31st Congress of the International Council
of the Aeronautical Sciences

Belo Horizonte, Brazil; September 09-14, 2018

PRACTICAL IMPLEMENTATION OF A TRAJECTORY PLANNING ALGORITHM FOR AN AUTONOMOUS UAV

Jean-Pierre Theron* , Laurent Dala** , Daniel N. Wilke* , Patrick Barrier***
*University of Pretoria , **Northumbria University Newcastle , ***UAS consultant

Keywords: *Inverse dynamics based planning; Near real-time planning; Numerical optimisation; Obstacle avoidance; Fixed-wing autonomous unmanned aerial vehicle*

Abstract

A near real-time optimal trajectory planning framework for UAVs is presented and tested in a series of low altitude obstacle avoidance planning scenarios. The framework uses the Inverse Dynamics Trajectory Optimisation approach with a quaternion point-mass aircraft dynamic model and a hybrid Differential Evolution and Sequential Quadratic Programming based Interior-Point optimisation strategy.

It was found that the new framework was able to successfully find a feasible (if not optimal) trajectory and to do so as efficiently as possible. However, it was also concluded that at this stage the framework is not yet fit to be used on a UAV, as the framework tends to take longer to plan a trajectory than it takes the UAV to fly it.

Ultimately it was concluded that with some further work, the Hybrid framework could be a viable near real-time trajectory planner for UAVs.

Nomenclature

		Units
Ξ	Design vector	
η	Propulsion efficiency	
γ	Flight path angle	
ϕ	Roll angle	
ψ	Heading angle	
ζ	Parameter tolerance	
$\boldsymbol{A}$	Coefficient matrix	
$\boldsymbol{L}$	Lower bound vector for Ξ	
$\boldsymbol{P}$	Bezier curve construction vector	
$\boldsymbol{r}$	Inertial postion vector	m
$\boldsymbol{U}$	Upper bound vector for Ξ	
$\boldsymbol{X}$	Boundary condition vector	
C	Constraint function	
C_D	Drag coefficient	
C_L	Lift coefficient	
$C_{D_1}, C_{D_2}, C_{D_3}$	Drag polar coefficients	
CR	DE crossover probability	
F	Cost function	kg
G	DE differential weight	
H	Population size	
J	Total number of constraints	
K	Total number of obstacles	
M	Load factor	g
m	Mass	kg
N	Total number of trajectory discritization points	
$nFval$	Number of objective function evaluations	
q_1,q_2,q_3,q_4	Quaternion elements	
R	Sphere radius	m
S	Aircraft reference surface	m^2
SFC	Specific fuel consumption	$kg.W^{-1}.s^{-1}$
T	Thrust	N
t	Time	s
x,y,z	Inertial coordinates	m

Subscripts

0	Initial value
μ	Average
σ	Standard deviation
$calc$	Calculation time
CV	Coefficient of variation
f	Final value
h	Population member counter
j	Constraint counter
k	Obstacle counter

n	Trajectory discritization point counter
O	First order optimality
obs	Obstacle
p	Penalty function
Superscripts	
$\dot{\ }$	Time derivative

1 Introduction

1.1 Motivation

A key component for an autonomous Unmanned Aerial Vehicle (UAV), is the ability for the UAV to develop a complete flight plan to achieve its given mission objectives. Part of this process is planning the trajectory that the UAV must fly in order to reach its objective. This can be done via a variety of methods, as presented in survey papers [1], and planning can be done both optimally and non-optimally. Additionally, given that the UAV has to do the planning, it implies that the planning process has to be conducted on-board the UAV and in near real-time. Finally, the UAV will also have to avoid obstacles during its flight. Especially during low altitude missions.

This leads to the interesting problem of implementing an optimal trajectory planning algorithm that is fast to plan trajectories and robust enough to work in any planning scenario.

1.2 Inverse Dynamics trajectory planning

In this work, the Inverse Dynamics Trajectory Optimisation (IDTO) approach is followed. The basis of the method was developed by Taranenko [2] (as presented by Yakimenko [3]) and relies on using differentially flat aircraft dynamic models to determine aircraft control histories from specified aircraft trajectories. Trajectory optimisation is then performed by adapting the specified aircraft trajectories until some optimality criterion is satisfied.

The attractiveness of the IDTO approach lies in the fact that one has direct control over the aircraft trajectory. That is one derives the controls histories using algebraic equations which in comparison to differential equations, that requires integration to compute the trajectories, is computationally more efficient to work with. Therefore, aircraft trajectories can be optimised much more rapidly with IDTO, which makes it an ideal candidate approach for real-time application on autonomous UAVs.

From Taranenko's work a variety of adaptations of the IDTO method have been developed and tested in simulation. Some examples include Yakimenko's own work [3] and Lei et al. [4].

This project, however, builds primarily on the work done by Drury [5, 6]. In his thesis, Drury, developed a complete IDTO framework that featured a quaternion inverse aircraft dynamic model. This was done in response to previous works, such as those mentioned earlier [3, 4], which all used an Euler-angle based aircraft model. It is well known that using Euler-angles for aircraft orientation representation is problematic given the gimbal lock phenomenon. Additionally, the presence of these discontinuities can also severely affect the numerical behaviour of the objective function and constraints of the trajectory optimisation problem. This, in turn, will affect the performance of the optimisation process. By using Drury's Quaternion model, these pitfalls could be avoided. Thus ensuring good performance for the trajectory planning process.

Drury, however, only did the foundational work in the development of his framework. Aside from the validation tests that he performed, Drury did not implement and test his framework in a more practical scenario, like for example an obstacle avoidance scenario. The purpose of this study is three-fold:

1. To extend Drury's framework to include obstacle avoidance planning.
2. To improve computational efficiency of the framework.
3. To evaluate the suitability of the framework for on-board route planning on UAVs.

1.3 Flight mission profile

The focus of this work was placed on doing planning during the cruise section of low altitude missions.

The framework was developed to work as a short time-scale planner of a receding horizon trajectory planner. Therefore, only a section of the entire mission flight path is planned within a planning cycle. Planning cycles are then repeated at a set frequency for the entire duration of the flight. Thus, only a limited amount of time is available for planning within a cycle.

2 Framework description

2.1 Inverse aircraft dynamic model

Drury's paper [5] gives a complete description of the inverse quaternion aircraft model. But it should be noted that for this work a different output vector was used. Where as Drury works towards an output control vector consisting of the acceleration and aircraft body angular rates, this work required an output control vector consisting of the engine thrust and orientation quaternion.

2.2 Space curve parametric functions

For the time t parametrised functions of x, y, z that define the 3-dimensional space curve of the aircraft's flight, it was opted to use 7th order Bezier curves. Therefore, 8 construction parameters P_i have to be specified per dimension, which results in a total of 24 construction parameters for a space curve in three dimensional space. In addition, specifying the final time t_f results in a 25 dimensional optimisation problem. This can, however, be reduced by enforcing initial and final conditions.

For each dimension the Bezier curves and their time derivatives (up to the third derivative) can be evaluated at the initial and final conditions, after which all the P_i can be separated out. The linear system in Equation (1) can be then constructed. The x dimension is used as an example but similar systems can be constructed for y and z.

$$
\begin{aligned}
\boldsymbol{AP} &= \boldsymbol{X} \\
\boldsymbol{P} &= \begin{bmatrix} P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 \end{bmatrix}^T \\
\boldsymbol{X} &= \begin{bmatrix} x_0 & \dot{x}_0 & \ddot{x}_0 & \dddot{x}_0 & x_f & \dot{x}_f & \ddot{x}_f & \dddot{x}_f \end{bmatrix}^T
\end{aligned}
\tag{1}
$$

$x_0\ x_f\ \dot{x}_0\ \dot{x}_f\ \ddot{x}_0\ \ddot{x}_f$ are known quantities. The initial conditions are known through measurement, while the end conditions are defined for the planning process. However, $\dddot{x}_0$ and $\dddot{x}_f$ are not. These two unknown parameters will act as design variables for the numerical optimisation. By specifying these two values, in conjunction with a specified t_f and the boundary conditions, a unique parametric curve and its derivatives are defined. Therefore, the final design vector for the numerical optimisation consists of $\boldsymbol{\Xi} = \begin{bmatrix} \dddot{x}_0 & \dddot{x}_f & \dddot{y}_0 & \dddot{y}_f & \dddot{z}_0 & \dddot{z}_f & t_f \end{bmatrix}^T$.

The curves were discretized in the t domain by using N equidistant nodes. The consequence is that δt between the nodes will differ between iterations as t_f changes over different iterations of the numerical optimisation. Thus, the accuracy of the trajectories and derived time histories is not consistent between iterations. Additionally, the value of N will also affect the performance of the algorithm. Hence, there exists a classical trade off scenario. Using a smaller value of N will reduce the calculation time of the planning process, since there are fewer nodes to process per iteration. However, this increased performance comes at the cost of losing accuracy in the results.

2.3 Optimisation problem

The aim is to find the optimal trajectory that minimizes the fuel consumption under mission and aircraft constraints which can be expressed by [7]:

$$
\min_{\boldsymbol{\Xi}} F = \int_0^{t_f} \log\left(\frac{T(t)V(t)SFC}{4.5\eta}\right) \mathrm{d}t \tag{2}
$$

subject to

$$
\begin{gathered}
\boldsymbol{\Xi} \in [\boldsymbol{L}; \boldsymbol{U}] \\
C_j = \max\left(c_{j,n}\right)_j \leq 0;\, n \in [1, \ldots, N]\,;\, j \in [1, \ldots, J]
\end{gathered}
$$

Where c is a constraint evaluated at every single node of the trajectory.

The first set of constraints that were enforced were 10 aircraft performance related constraints. They are:

$$-1000\text{ m} \leq z \leq 0\text{ m} \tag{3}$$

$$0\text{ N} \leq T \leq 400\text{ N} \tag{4}$$

$$V_{stall} \leq V \leq 70\text{ m.s}^{-1} \tag{5}$$

$$-1\text{ g} \leq M \leq 2.5\text{ g} \tag{6}$$

$$-0.8601 \leq 2(q_1q_3 - q_2q_4) \leq 0.8601 \tag{7}$$

Constraint 7 is derived from the constraint $-90° \leq \gamma \leq 90°$, which in itself is derived from the need to enforce the dynamics model's assumption that the magnitude of the lift vector cannot be 0.

Obstacle avoidence were also enforced through the use of constraints. For the obstacles it was opted to only use spheres, since they could easily be characterised by the coordinates of the centre point and their radius. Therefore, the obstacle avoidence cosntraints were characterised as:

$$-\left\|\boldsymbol{r} - \boldsymbol{r}_{obs_k}\right\|_2 \leq R_{obs_k} + 1\,;\, k \in [1,...,K] \tag{8}$$

2.4 Optimisation strategy

Having a limited amount of time available for a planning, emphasizes the need to carefully consider the optimisation strategy. To that end the optimisation strategy was build around a 3 tier solution hierarchy upon which the final result of the planning process would land. The higher up the trajectory is, the more ideal the trajectory is.

1. Tier 1 is optimal trajectories for the given scenarios.

2. Tier 2 is feasible trajectories for the given scenarios.

3. Tier 3 is infeasible trajectories that minimise the constraint violations.

Given the nature of how numerical optimisation algorithms operate, the optimisation process will naturally progress to produce final results from Tier 3 to 2 to 1. However, the strategy can be designed to maximise the probability of ending with a solution from tier 2 or 1.

With regards to Tier 1's optimal solution, no specification was made that the solution should be the global optimum. Given the high non-linear nature of the problem, focusing on finding the global optimum would take a lot of time and thus it is not practical for near real-time planning.

Additionally, during a planning cycle it is possible for the trajectory planner to find a solution so quickly that there is still some calculation time available to search for a better local optimum. This work, however, did not include this additional improvement scenario in its scope. The focus was placed only on finding the first feasible local minimum, after which the planning cycle would stop.

The nature of the starting initial guess plays a significant role in the performance of the trajectory planner and thus it affects the design of the strategy. Two scenarios were considered. The first is using the result of the previous planning cycle as the initial guess of the next cycle. This is the preferred type of initial guess, since it has already been optimised. Thus, the optimisation process starts off from a known optimal trajectory and only has to correct for the differences in the planning environment between the two planning cycles.

The second scenario is using a random initial guess. This would typically be used in scenarios where the result from a previous planning cycle cannot be used, for example the very first planning cycle. Random initial guesses would also be used in cases where attempts are made to find better local minima. This is more a difficult position to start from, since the location of the feasible domain, let alone an optimal solution, is not known. Additionally, the optimisation process has no information about the numerical landscape of the objective and constraint functions between the initial guess and the optimal point. Thus, the planning process must take into account the possibility of the optimisation process getting stuck in the infeasible domain due to discontinuities or local minima/saddle points etc. Even if the numerical landscape was characterised, it would only be helpful for one planning cycle, since, during flight, the numerical landscape is constantly changing between planning cycles.

The near real-time requirement, solution hierarchy and the nature of the initial guess, pushes the focus on optimisation strategies that are fast to optimise the problem but also robust against getting stuck in the infeasible domain of the design space. Lai et al. [4] used a 2-stage approach where they used the first stage to find the feasible domain and to provide the second stage with an improved initial guess. Stage 2 was then used to quickly close in on the optimal value. The same hybrid approach was used in this work.

For stage 1, numerical robustness, especially for the random initial guess case, was the main design requirement. The Differential Evolution (DE) algorithm was ultimately selected for this stage. Since DE uses an entire population of agents to perform its optimisation, it is very robust against the numerical landscape of the problem. If one or more of its agents get stuck at some local minima or numerical feature, DE can still rely on the remaining agents to complete the optimisation process. Additionally, DE also enhances the chance that a global solution is found. DE, however, has slow convergence properties which translates into long calculation times.

For the second stage, a Sequential Quadratic Programming (SQP) based Interior-Point (IP) method was used. As a gradient based solver, IP can very quickly close in on an optimal value by using the gradient information of the objective and constraint functions. However, it is susceptible to getting stuck on local minima/numerical features. So while it is fast, its robustness is less than that of the DE.

During the simulation experiments of Section 3, prebuilt numerical optimisation packages were used for the two optimisers. Buehren's DE package for MATLAB [8] was used for DE stage, while MATLAB's own IP package [9] (via the fmincon function) was used for the IP stage.

The DE stage was set up to only search for the feasible domain by optimising the following penalty function:

$$F_p = \sum_{j=1}^{J} \left[\max\left(0, \max\left(C_{j,n}\right)_j\right)\right]^2 ; n \in [1, \ldots, N] \tag{9}$$

Hence no attempt is made by the DE to optimise the problem of Equation (2).

For the stopping criteria, 2 metrics were used in the DE stage. The first was:

$$\min\left(F_{p_h}\right) = 0 ; h \in [1, \ldots, H] \tag{10}$$

The second criterion was a simple allocation of 30 seconds of calculation time, as measured by MATLAB's tic-toc function. Irrespective of which stopping criterion was reached, the design related to the best member from the DE's final population was used as input to the second stage. Thus, in the case where the DE ran out of calculation time, infeasible designs were sent to the second stage.

The DE algorithm was configured to operate with a DE/rand/1/exp mutation strategy, $G = 1.5$ differential weight, $CR = 0.3$ crossover probability and a population of $10 \times \text{length}(\boldsymbol{\Xi}) = 70$ members. For the first generation 69 members are randomly generated using a uniformly distributed random process. The 70th member is the supplied initial guess. For the rest of the configurable parameters Buehren's default values were used. It should be noted that this optimiser's settings are not optimised for implementation.

The complete optimisation problem of Problem 2 to 8, was optimised by the IP stage. Three metrics were used as stopping criteria for the IP stage:

1. First order optimality tolerance of $\zeta_O \leq 10^{-3}$.
2. Design step tolerance of $\zeta_{\boldsymbol{\Xi}} \leq 10^{-3}$
3. Maximum objective function evaluation count of 1000 evaluations.

MATLAB's default values were used for the IP's settings, except for the constraint violation tolerance which was set to $\zeta_C \leq 10^{-3}$. Thus, the IP stage used finite differences for gradient estimation and BFGS for Hessian approximation. This set up is also not optimised for implementation.

For any given planning scenario, there are multiple locally optimal solutions. Since, the strategy gives no special attention towards finding the globally optimal solution, the planning process will deliver different trajectories between repeat runs of the same planning scenario. With Stopping Criterion 10, the DE stage is stopped as soon as the first feasible design is found and the IP stage would tend to close in on a locally optimal solution in the surrounding section of the feasible design space. Therefore, if the feasible design space is segmented into pockets, there is no possibility for optimums in other pockets, to be considered during the IP stage. The random population generation and mutation strategy of the DE stage enhances the likelihood that between different planning runs, different pockets are found resulting in a variety of final trajectories.

3 Experimental tests

The aircraft data and model parameters used during the experiments are tabulated in Equation (11). These values are derived from the RoadRunner aircraft developed by the Paramount Group [10]. Also, a constant air density was assumed for the experiments.

$$\begin{aligned}
&m = 45 \text{ kg} &&; C_{D_1} = 0.1754\\
&S = 2.7 \text{ m}^2 &&; C_{D_2} = -0.0168\\
&V_{stall} = 11.547 \text{ m.s}^{-1} &&; C_{D_3} = 0.0258\\
&\rho = 1.225 \text{ kg.m}^{-3} &&; N = 400\\
&\eta = 0.7\\
&SFC = 1.1111 \times 10^{-7} \text{ kg.W}^{-1}\text{.s}^{-1}
\end{aligned} \qquad (11)$$

3.1 Simple planning tests

As a demonstration of the capabilities of the algorithm a series of 4 experiments were conducted. The first 3 experiments were simple planning scenarios. In each case the UAV was given a set of boundary conditions and the algorithm had to plan the optimal trajectory between the start and end points. Random initial guesses were used for these 3 experiments. The distance between the end and start point was chosen to be small enough so that the end point would arguably be within the planning range of a single planning cycle. Therefore, the entire trajectory of the mission will be planned in one planning cycle.

Parameter	Exp 1		Exp 2		Exp 3	
	0	f	0	f	0	f
V [m.s^{-1}]	27.6026					
γ [°]	35.2656					
ψ [°]	45					
ϕ [°]	0					
x [m]	1	100	1	200	1	200
y [m]	1	100	1	200	1	200
z [m]	-1	-100	-1	-200	-1	-200
x_{obs} [m]	50.5		$\begin{bmatrix}50.5\\150.5\end{bmatrix}$		$\begin{bmatrix}50.5\\150.5\\110\end{bmatrix}$	
y_{obs} [m]	50.5		$\begin{bmatrix}50.5\\150.5\end{bmatrix}$		$\begin{bmatrix}50.5\\150.5\\90\end{bmatrix}$	
z_{obs} [m]	-50.5		$\begin{bmatrix}-50.5\\-150.5\end{bmatrix}$		$\begin{bmatrix}-50.5\\-150.5\\-100\end{bmatrix}$	
R_{obs} [m]	10		$\begin{bmatrix}10\\10\end{bmatrix}$		$\begin{bmatrix}10\\10\\10\end{bmatrix}$	

Table 1 Boundary conditions used during the experiments

Obstacles were placed on the direct line connecting the end and start points, ensuring the algorithm would have to plan for obstacle avoidance. Experiment 1 (Exp 1) was given only one sphere to dodge, while Experiment 2 (Exp 2) and 3 (Exp 3) were constructed to be environmentally more complex by having more spheres to dodge.

Table 1 Lists all the boundary conditions and obstacle data used during Exp 1-3.

From these boundary conditions $\boldsymbol{L}$ and $\boldsymbol{U}$ for each experiment were defined. This process was done on a trail-and-error approach and the final bounds were selected based on whether the extreme bound trajectories were contained in an approximately ±2000 m in x,y and z box. For all three the experiments the bounds $\boldsymbol{L} = \begin{bmatrix}-10 & -10 & -10 & -10 & -10 & -10 & 1\end{bmatrix}^T$ and $\boldsymbol{U} = \begin{bmatrix}10 & 10 & 10 & 10 & 10 & 10 & 60\end{bmatrix}^T$ were found to be sufficient to define the bound constrained ***LU*** domain.

Given the random element inherent to the Hybrid trajectory planner, each experiment repeated the planning process a 100 times. The 100 random starting points were sampled using Latin hyper cube sampling over the entire ***LU*** domain. This ensures that the entire bounded design domain is included in the experiments.

To provide additional levels of comparison and validation of the optimisation strategy's design choices, all three experiments were also conducted with trajectory planners that only used either the DE or the IP optimisation solvers.

The pure DE solver was tasked with optimising Equation (2), with prospective designs only being checked for feasibility. With this objective function, 3 new stopping criteria were defined:

1. $\left|F_{h_{best}} - F_{h_{worst}}\right| \leq \times 10^{-3}$
2. $\left\|\boldsymbol{\Xi}_{h_{best}} - \boldsymbol{\Xi}_{h_{worst}}\right\|_2 \leq \times 10^{-3}$
3. Maximum generation count of 100 generations.

The same mutation strategy, differential weight, crossover probability and population size as that of the Hybrid DE was used.

The pure IP solver was set up to use the same settings as the Hybrid IP solver. However, the pure IP solver was given an increased maximum function evaluation count of 3000 evaluations.

A planning attempt was considered failed if the final solution was still within the infeasible domain. This could only really occur in cases where the framework ran out of function evaluations/generations.

Finally, for every experimental run the success rate, the calculation time t_{calc} (as measured by MATLAB's tic-toc function) and the number of objective function evaluations $nFval$ were recorded. Additionally, the t_f values from the final designs were also recorded. With these t_f values the suitability of the configuration will be evaluated by counting the amount of runs where the calculation time was less than the final planned flight time. For the system to be suitable it should generally plan trajectories faster than it would take to fly them.

All the experiments were conducted on a Lenovo ThinkPad X250 with an Intel Core I7-5600U 2.6 GHz 4 core CPU, 8 GB PC3-12800 DDR3L RAM and Samsung SSD 850 EVO 500 GB hard drive Windows 8.1 Pro Version 6.3 Build 9600 was used for the operating system.

3.1.1 Results

Table 2 lists a summary of the final results. Except for the success rate, only the successful runs were used in calculating these results. Looking at the success rate and the averages of $t_{calc_{\mu}}$ and $nFval_{\mu}$ over all three cases, one can clearly see the expected result of the DE configuration generally being slowest but 100% successful, while the IP configuration was the fastest but with the lowest success rate. With the Hybrid falling in the middle of two, one can conclude that the design works as intended.

What is interesting to note is the dramatic improvement in performance for the Hybrid configuration between Exp 1 and Exp 2-3. This result is counter-intuitive when considering that the planning environment of Exp 1 is a lot simpler than that of Exp 2-3. The scale of the trajectory that has to be planned, could be a possible explanation. Table 1 shows that the distance between the start and end points for Exp 1 is a lot smaller than the distance for Exp 2-3. It is possible that this smaller scale makes the associated numerical landscape of Exp 1 more difficult to traverse during the optimisation process. Thus, increasing the calculation time.

As for the consistency in the performance of the various configurations, the coefficient of variance $CV = \sigma/\mu$ ratios of $t_{calc_{CV}}$ and $nFval_{CV}$ in Table 2, show that in Exp 1 the Hybrid method had the most consistent performance since it had the lowest CV value. The DE had the most erratic performance with the highest CV value. However, in Exp 2 and Exp 3 the roles are reversed and the Hybrid and IP frameworks have about the same consistency. The scaling problem discussed above, could also be a possible explanation for

	Exp 1			Exp 2			Exp 3		
Parameter	**Hybrid**	**DE**	**IP**	**Hybrid**	**DE**	**IP**	**Hybrid**	**DE**	**IP**
Success rate [%]	92	100	57	92	100	74	94	100	68
$t_{calc_{\mu}}$ [s]	32.549	103.8115	3.5966	23.2699	206.1538	4.2409	22.2099	213.2413	4.0184
$t_{calc_{CV}}$ [s]	0.1611	1.0732	0.4294	0.5464	0.2655	0.5975	0.5917	0.1814	0.5369
$nFval_{\mu}$	3534	2868	267	2473	7221	316	2388	7420	295
$nFval_{CV}$	0.1557	0.9132	0.4357	0.5549	0.1543	0.6077	0.5974	0.0158	0.5486
$t_{f_{\mu}}$ [s]	12.4687	16.7343	22.4361	14.1904	13.6186	23.5792	14.8695	13.6007	21.7234
Suitability [%]	2.1739	7	100	29.3478	0	100	31.9149	0	100

Table 2 Exp 1-3 results

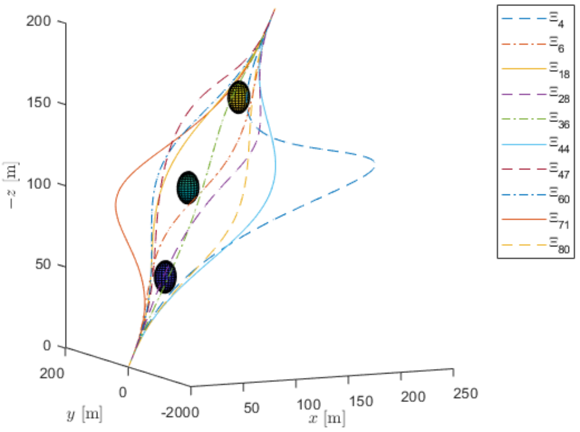

Fig. 1 Exp 3 sample set of trajectories.

this phenomenon.

Finally, the suitability values show that the Hybrid framework is again the middle choice between the three configurations. Unsurprisingly the DE configuration did the worst of the three, while the IP configuration had a 100 % rate for all three experiments. Again there is a massive improvement phenomena in the Hybrid's suitability between Exp 1 and Exp 2-3. This is a direct consequence of the dramatic improvement in calculation time discussed previously. Despite this, the results of the experiments does cast doubt on the viability of the Hybrid configuration.

However, as mentioned in Section 2.4, the solver settings of the two optimisers have not been optimised to this type of application. Additionally, by using dedicated compiled programming code instead of pre-built packages for the optimisers, one would be able to gain some additional performance. Hence, despite the underwhelming suitability values, the Hybrid configuration still has the potential to a viable method for robust and fast optimal trajectory planning and thus warrants further investigation.

Figure 1 is a collection of 3D plots of 10 randomly chosen trajectories from the Hybrid pool of successful trajectories. There are 2 distinct groups of trajectories First there is a group of trajectories that can be considered "reasonable". Trajectories that closely follow the direct line between the start and end point, and only curve to avoid the obstacle. An example is Ξ_6. In contrast to this are "unreasonable" trajectories. Trajectories that either momentarily completely veer off course for no apparent reason, or perform manoeuvres that are completely unnecessary. Like for example Ξ_4. While these trajectories are unreasonable, they are valid since they meet all the constraints. Thus, they are included in the range of possible solutions

There are few options as to how to eliminate these trajectories. The first option would be to use constraints, like for example reducing the optimisation bounds $\boldsymbol{L}$ and $\boldsymbol{U}$, or adding additional constraints. However, care should be taken so that the planning of reasonable trajectories are not inhibited.

The alternative approach would be to add an optimisation metric that is minimised alongside the fuel cost. Like for example minimising the distance of each point on the trajectory to its corresponding point on the direct line between the start and end point. Following this approach, however, transforms the optimisation problem into a multi-objective problem and then the competing interests of this metric and fuel consumption have to be balanced.

3.2 Replanning test

Experiment 4 (Exp 4) is almost exactly the same as the previous 3 experiments. The focus of this test, however, was to emulate the replanning scenario where one would use the result of a previous planning cycle as the initial guess. This was emulated by repeating Exp 3 but with the successful Hybrid configuration results of Exp 2 as input. In this case the third obstacle of Exp 3 is a new object that has to be accounted for.

For this experiment 3 types of results can be expected. The first is cases where the original Exp 2 trajectory would collide with the new obstacle and thus the replanning process would replan to avoid the new obstacle. The second type of result is cases where the Exp 2 trajectory would not have collided with the new obstacle, but the replanning process further improved the trajectory. The third group of results are cases where the Exp 2 trajectory would not have collided but the replanning process did not result in a new design.

Due to these possible solutions, multiple optimisation configurations were again tested to see if one could still further improve one's performance. In this case since one already has a baseline trajectory that is either feasible (Group 2 and 3) or infeasible (in the obstacle avoidance sense), one might find that one can get by without having to do a DE stage type search. Therefore, for this experiment only the Hybrid and pure IP configurations were tested. In both cases the same solver settings used during Exp 1-3 were used in this experiment as well.

3.2.1 Results

The results of Exp 4 is listed in Table 3.

Looking at the "Number of trajectories" row in Table 3, it should be noted that since the results of Exp 2 were used as input, no control was exercised over the distribution of the experimental results between the three types of results. Therefore, only very preliminary interpretations can be drawn. Particularly for the results of small numbered groups.

Focussing first on calculation speed via the average values of t_{calc} and $nFval$, it is clear that the pure IP configuration performed the best of the two configurations. As expected, its average calculation times and function evaluation counts are less than that of the Hybrid method. However, in contrast to Exp 1-3, the IP configuration has a 100 % success rate for all three groups. This suggests that for the replanning scenario, the IP configuration is the superior option.

The coefficient of variance ratios of t_{calc} and $nFval$ show that the Hybrid configuration still has a more consistent performance. Of the two configurations it has the lowest values over all three groups.

Finally, the suitability values are, in almost all the cases and for both configurations, a 100 %. Again the only exception was the Hybrid Crash group and this is most probably due to the low number of trajectories in this group. Ultimately, based on this result and the results of t_{calc} and $nFval$, it can be concluded that in the replanning scenario, the pure IP configuration is the better option.

To further illustrate the point that the replanning scenario results in better performance than the random initial guess, a comparison simply needs to be drawn against the results of the Hybrid configuration from Exp 3. This was done in Table 3. Both the Hybrid and IP methods had better success rates, $t_{calc_{\mu}}$ and $nFval_{\mu}$ values and suitability. The relative differences between the averages of $t_{calc_{\mu}}$ and $nFval_{\mu}$ and those of Exp 3 show improvements of at least 43.0585 %.

4 Conclusion

Drury's framework was successfully adapted and implemented in the form of the Hybrid framework. Testing showed that in principle, the new system worked in an obstacle avoidance setting. The tests also showed that the system struck a good balance between successfully finding a feasible (if not optimal) trajectory and doing it in the least amount of time.

However, in its current form, testing also

	Crash		Improved		Same		Exp 3
Parameter	**Hybrid**	**IP**	**Hybrid**	**IP**	**Hybrid**	**IP**	**Hybrid**
Number of trajectories	2	2	87	40	3	50	100
Success rate [%]	100	100	100	100	100	100	94
t_{calc_μ} [s]	12.6467	1.098	1.9351	0.9069	1.5136	0.1241	22.2099
$t_{calc_{CV}}$	0.9532	1.0971	0.4153	1.0645	0.0087	0.2382	0.5917
Relative difference w.r.t. to t_{calc_μ} of Exp 3 - Hybrid [%]	-43.0585	-95.0562	-91.2872	-95.9165	-93.1852	-99.4411	0
$nFval_\mu$	1218	81	175	65	148	8	2388
$nFval_{CV}$	0.9159	1.1331	0.3251	1.0732	0.0039	0.2044	0.5974
Relative difference w.r.t. to $nFval_\mu$ of Exp 3 - Hybrid [%]	-49.0159	-96.629	-92.6534	-97.2603	-93.7884	-99.6491	0
t_{f_μ} [s]	13.407	14.839	14.0395	14.8659	12.96	13.4595	14.8695
Suitability [%]	50	100	100	100	100	100	31.9149

Table 3 Exp 4 results

showed that the Hybrid configuration is not yet suitable for use on UAVs. However, by optimising the settings of the optimisation solvers and improving the programming code, the performance figures of the system could be improved.

Ultimately it is concluded that the Hybrid framework still has the potential to develop into a viable near real-time trajectory planner for UAVs.

References

[1] Goerzen C, Kong Z and MettlerB. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, Vol. 57, No. 1, pp 65-100, 2009.

[2] Taranenko V T. Experience of employment the Ritz's, Poincare's, and Lyapunov's methods for solving the problems of flight dynamics. *Air Force Engineering Academy Prof. N. Zhukovskiy Press*, 1968.

[3] Yakimenko O A. Direct method for rapid prototyping of near-optimal aircraft trajectories. *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 5, pp 865-875, 2000.

[4] Lai C K, Lone M, Thomas P, Whidborne J F and Cooke A. On-board trajectory generation for collision avoidance in unmanned aerial vehicles. *2011 IEEE Aerospace Conference*, Big Sky MT USA, pp 1-14, 2011

[5] Drury R G. Trajectory generation for autonomous unmanned aircraft using inverse dynamics. Phd thesis, Cranefield University. 2010.

[6] Drury R G and Whidborne J F. Quaternion-based inverse dynamics model for expressing aerobatic aircraft trajectories. *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 4, pp 1388-1391, 2009

[7] Phillips W F. *Mechanics of Flight.* 2nd edition, Wiley, 2004

[8] Buehren M. Differential evolution, Ver. 1.16. MathWorks - File Exchange, https://www.mathworks.com/matlabcentral/fileexchange/18593-differential-evolution (visited on 2018-02-14)

[9] MathWorks. Constrained nonlinear Optimization Algorithms. https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html#brnpd5f (visited on 2018-06-05)

[10] Paramount Group. Roadrunner. http://www.paramountgroup.com/capabilities/air/roadrunner/ (visited on 2018-06-17)

Contact Author Email Address

jeanpierre.jtheron@gmail.com

Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.