An Nonlinear Model Predictive Control Approach to Autonomous UAV Racing Trajectory Generation & Control

S.T. Spronk*, S. Li†,
G.H.C.E de Croon†

*Section Control & Simulation, Faculty of Aerospace Engineering,
Technical University Delft, The Netherlands*

Februari 2020

Abstract

When observing an Autonomous Unmanned Aerial Vehicle(UAV) race, one would be hard-pressed to call it racing as the actual velocities attained are extremely low. This article addresses this shortcoming by proposing a method of generating and executing a racing trajectory for a UAV, through a series of position objectives representative of a racing environment, with the goal of significantly improving the velocity when compared to the current norm of PID controllers. The method consists of applying Nonlinear Model Predictive Control with the capability of dynamically updating the position goal based upon internal state estimation to generate a set of inputs for a UAV. To prove the viability of the proposed method we test by using numerical simulations, a flight simulator environment(Gazebo) and a series of real-world flight tests on the Bebop1 UAV. Through 2 iterations of the testing process it is proven that the method is able to significantly($\approx 1s$) decrease the flight time through both simple and more complex short range manoeuvres($2m - 4m$). However model errors and an inability to fully control thrust on the UAV introduce a significant and consistent position error.

1 Introduction

Autonomous UAV racing is a very new discipline in the field of control systems. These races are attended alsmost exclusively by academic teams [1] [2], and the first ever autonomous UAV race was held as recent as 2016. It has a promising future however as piloted First Person View(FPV) racing is already a well established and professionally looking, albeit niche, sport [1]. The FPV pilots are able to control their crafts with velocities of over $150km/h$ through sufficiently large arenas with known gate positions. However since all autonomous UAV races are held in very small arenas the main limiting factor on speed is the distance between gates. As these distances generally do not exceed 4 meters velocities exceeding $\approx 8km/h$ are considered extremely impressive [3]. The current trend is to generally forgo any optimisation in terms of trajectory and/or inputs and use basic control methods such as PID controllers [3] [4] [5] [6]. Since the foremost issue limiting UAV racing velocities is the accurate and quick identification of gates [7] [8]. However as progress is made in this the need for more aggressive manoeuvres will increase. To facilitate these developments faster control schemes and racing strategies must be developed. Recently a trend of applying Model Predictive Control(MPC) to UAV flight can be observed [9] [10] [11] [12]. Nonlinear Model Predictive Control(NMPC) implementations are less explored but have shown the capacity of impressive high-velocity aggressive manoeuvres [13]. This article proposes a method of generating a close to optimal trajectory and a corresponding set of inputs for a UAV in a complex racing environment. With the aim of significantly reducing the flight time between objectives in comparison with a PID controller. The core of the approach is the use of NMPC. Selected for its capability to react to on-line changing objectives. As racing environments consist of a series of gates with generally known locations the method provides a basis for a computationally efficient on-line racing controller able to adapt to changes in the environment and generating a close-to time-optimal set of inputs for the UAV.

Section 2 will describe the method used to detect and avoid the objects; Section 3 will describe the tests that were performed; Section 4 will show the results and finally the conclusions are drawn and recommendations are put forward in Section 5.

2 Method

This section will explain the method used to turn a set of positions/attitudes representing a racing trajectory into a set of attitude inputs for a UAV. The process consists of several steps. First choices made in the problem framework such as the selection of reference frames and dynamic model will be explained. followed by an overview of the functioning of an NMPC. And finally an explanation of the specific implementation of said NMPC.

2.1 Framework

The first choice was that of a UAV model. The UAV model used for testing was a Parrot Bepop 1 with an assumed weight of 530 grams. All values and results are based on the capabilities and performance of this UAV. The proposed approach uses 2 reference frames, namely the body and inertial frame. All positions and velocities are tracked in the

*Student
†Supervisor
[1] `https://theUAVracingleague.com/`

inertial, also called the earth, frame while all accelerations are tracked in the body frame. These reference frames are illustrated in Figure 1

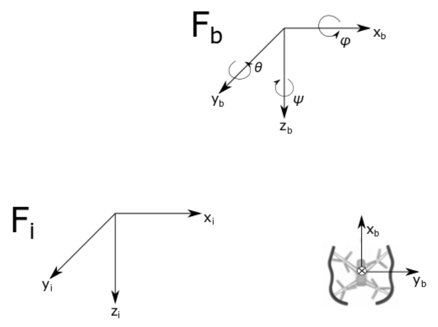

Figure 1: Illustration of inertial frame F_i, body frame F_b and body frame to UAV orientation

To model the UAV a set of ordinary differential equations(ODE) have to be defined. The full set, basis provided by Li, S[2] used for the project can be found in Equation 1.

$$\dot{x} = cos(\psi)*cos(\theta)*v_x - sin(\psi)*cos(\phi)*v_y \quad (1a)$$
$$\dot{y} = cos(\psi)*cos(\phi)*v_y + sin(\psi)*cos(\theta)*v_x \quad (1b)$$
$$\dot{z} = cos(\theta)*cos(\phi)*v_z \quad (1c)$$
$$\dot{v_x} = \frac{-sin(\theta)*g*m}{m} - (q*v_z - r*v_y) - (C_d*v_x)^* \quad (1d)$$
$$\dot{v_y} = \frac{sin(\phi)*cos(\theta)*g*m}{m} - (p*v_z - r*v_x) - (C_d*v_y)^* \quad (1e)$$
$$\dot{v_z} = \frac{cos(\phi)*cos(\theta))*g*m+T}{m} - (p*v_y - q*v_x) - (C_d*v_z)^* \quad (1f)$$
$$\dot{\phi} = p + q*tan(\theta)*sin(\phi) + r*tan(\theta)*cos(\phi) \quad (1g)$$
$$\dot{\theta} = q*cos(\phi) - r*sin(\phi) \quad (1h)$$
$$\dot{\psi} = q*(sin(\phi)/cos(\theta)) + r*(cos(\phi)/cos(\theta)) \quad (1i)$$
$$\dot{T} = \delta_T \quad (1j)$$

These equations are slightly simplified from the full transformation from body to inertial under the assumptions of small angles. As will be presented in Subsection 2.4.2 roll and pitch angles are constrained to not exceed 25 degrees, which makes this a valid assumption. Furthermore the bracket terms with an asterisk (*) in Equations 1d, 1e and 1f denote the drag terms, assumed to be $C_d = 0.5$, which are not present in all implementations as will be explained in Section 3

2.2 UAV Racing Considerations

In order for the proposed method to function in a racing environment there are some considerations to be made. There is a certain way a UAV behaves, or should behave taking into account autonomous racing conditions. This problem is best supported with an illustration and can be found in Figure 2

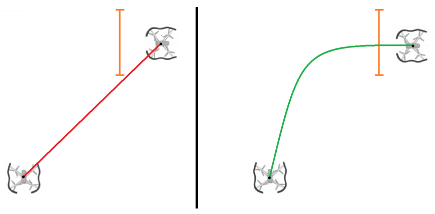
Figure 2: Racing optimality problem example

As can be seen in the figures while the objective is the same the results are very different. While the left trajectory is mathematically optimal being a straight line between two points it is not optimal in a racing environment where gates have to be cleared. One of the greatest challenges of this research was to first define and the try to capture 'good' racing behaviour in a set of cost function weights. The adopted solution to this problem is presented in Subsubsection 2.4.2. An example of this problem is that early in development the UAV initial conditions included that it started on the ground with no thrust. While this makes sense on first glance what would happen is that even with a constraint that prohibits an altitude lower than 1 the constraints on the rate of increase of thrust would conflict with this and the UAV would in mathematical terms fall through the floor. Although this would in all likelihood not have an issue when considering a real flight test it does illustrate the problem well. It is very ill-defined what good racing behaviour is and, this might be stating the obvious, the program does not have a frame of reference or any built in parameters that relate to UAV racing. So building on the earlier example if one would like to move from a (x=0,y=0) to a (x=3,y=3) position it would be easily proven that simply rolling and pitching would generate a straight line between those points, the strictly fastest trajectory to the target. In racing conditions however one should consider that the UAV would need to be able to detect a gate or goal and should thus have a sensor pointed in the direction of flight. This traditionally takes the form of a camera pointing forward, which will be the main consideration for the research. The pure mathematical optimal solution is thus very often not optimal in actual racing conditions. The second problem is that no clear guide on how to tune a 3D NMPC for autonomous racing UAVs exists as far as the author could find. This thus necessitated the following approach. First to define a set of desirable global behaviours, such as pointing toward the goal and preferring pitching. It is therefore important to keep in mind that the values used to define optimal are highly subjective and due consideration to the real-world environment where this method would be applied is extremely important. The chosen values and their motivation are presented in Subsubsection 2.4.2.

2.3 NMPC

The core of the proposed solution is an NMPC. This subsection will explain the theory and the implementation of the used NMPC algorithm and its numeric simulation environment.

[2]`https://github.com/ls90911/drone_simulation`

an NMPC is a special case of a normal Optimal Control Problem(OCP) [14] [15]. In essence an NMPC repeatedly solves an off-line OCP to create an on-line control solution. The formal mathematical problem definition of an NMPC that needs to be solved can be found in equations 2,3,4a and 4b.

$$
\begin{aligned}
min\ J = \sum_{i=0}^{N} ||h(x(t_i),u_i) - y_{ref}||^2 W \\
+ ||h(x(T),u_T) - h_{ref}||^2 R
\end{aligned} \tag{2}
$$

subject to:

$$
u \in U\ \forall t \in [t+T],\ x \in \chi\ \forall t \in [t+T] \tag{3}
$$

where:

$$
U := u \in \Re^m | u_{min} \leq u \leq u_{max} \tag{4a}
$$

$$
\chi := x \in \Re^n | x_{min} \leq x \leq x_{max} \tag{4b}
$$

where T is the horizon length, $x(t_i)$ is a vector containing the following UAV differential states x,y,z,v_x,v_y,v_z,ϕ,θ,ψ sampled at interval length t_i, ui consist of the input r. The choice of penalizing only the yaw rate is to possibly avoid numerical ambiguity in the optimization. The other inputs of the system, p, q and δ_{T_r} are not included in the cost function as to not restrict the possibility of aggressive manoeuvres. W is a matrix containing the weights associated with each state and output. y_{ref} is a matrix containing the desired value for each interval. h_{ref} is a vector containing the set of goals of y_{ref} corresponding to the final interval. R is also mostly equal to W_i, unless the system needs a higher terminal value penalty to enforce stability in which case R becomes a multiple of W_i. Only one input is present in the cost function because it would impact the performance of the UAV in terms of time. The exception is made for the yaw angle as its impact on travel times is negligible. The possibility of a penalty for yaw rate was added as an extra security to stop the UAV from yawing 'unnecessarily', in context of desired racing behaviour. This kind of optimization problem is solved numerically by sequential quadratic programming(SQP) algorithm [16]. In addition to the NMPC an OCP solver to find the true optimal time was also constructed for comparison. This OCP solution was constructed to optimize for time. An OCP is not restricted by a fixed interval length and to compare a roughly equivalent amount of intervals amount of 20 was set. This number is chosen to roughly provide the same number of intervals as the NMPC, which an interval size of 0.1 seconds. The reason the OCP was not chosen as a basis for the approach instead of the NMPC is the fact that it is unable to dynamically change objective, always having only 1 set of initial states and 1 set of end states [17].

2.4 ACADO

The implementation of the NMPC was done using the ACADO toolkit [18] [19] [20] [21] [22]. This software package is focused primarily on solving both linear and nonlinear optimal control problems, of which model predictive control is a special case. In addition to providing syntax to ease the setting up of such a model it also facilitates the running of a (N)MPC in a numerical simulation environment. This being the method of generating a trajectory/UAV input data.

2.4.1 The Adaptive Reference System

In order for it to function for a racing application a custom add-on for the reference had to be made. When planning a flight path the original release requires a pre-determined time for every change of objective y_{ref}, presented in Equation 2. Since the objective of the NMPC is to find a time-optimal path in complex environments this was critically important to fix. The adaptive reference trajectory module was created and successfully implemented in the ACADO toolkit. The adaptive system checks the internally estimated states to decide whether to advance the goals. Thus always having only a singe goal for all intervals. While with a fixed goal switch time this is not always the case. In Figure 3 a comparison of results of a trajectory simulated with ACADO using the new adaptive and the old static reference system. The objective was to fly a square with 3 meter long sides. The objective switch times provided to the static reference system were set to 2 seconds.

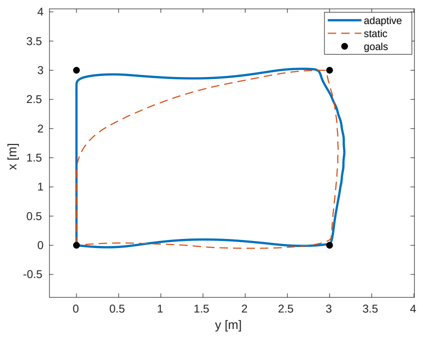

Figure 3: Adaptive reference trajectory vs static reference trajectory comparison

It can be seen that the simulation using the adaptive reference trajectory system follows the set trajectory far more accurately than the static version. The error is caused by cost function terminal cost term. causing the static system to disregard imminent objectives in favour of later ones as the last state on the horizon is considered to be semi-infinite constant [14]. In addition the adaptive reference system also completely eliminates the need for pre-determining times of any sort.

The current release does have a couple of limitations and/or shortcomings which will be briefly discussed. Currently the goal is considered reached, and the decision made to switch to the next goal, if the x, y and z position of the UAV is within an 5% error region of the desired position for 5 consequetive sampling instances. This means, for example, that a reference trajectory that only contains attitude changes will not be executed properly. And that the current system is ill-suited for use when flying larger distances. Also any internally estimated position errors will propagate, but this problem is ubiquitous in

[3]https://github.com/stspronk/ACADOtoolkit-Adaptive-Reference-Trajectory-V1.0

UAV control so therefore not of great immediate concern for this project. The source code of the adaptive reference trajectory add-on for the ACADO toolkit can be found here[3]. The NMPC is coded in C++ , the compiled program is therefore very computationally efficient. While currently still containing a numerical simulation of the process it would be possible to substitute this with the estimated states of the real UAV. Thus enabling an on-line version running on the UAV.

2.4.2 Horizon Size, Cost Function Weights & Constraints

The final values for the racing optimized NMPC Cost Function Weights & Constraints are presented here. The are the values for W_i found in Equation 2 and the range of U and χ found in Equation 4. Also the horizon size and interval denoted by the size of t_i, and T in Equation 2 is presented here. The horizon length T is 1 second consisting of 10 intervals of 0.1 seconds. The allowed sampling time for the state estimator is half the horizon interval at 0.05 seconds. The constraints, as they complement the dynamic model presented Equation 1 to fully describe the UAV capabilities. The constrained values and their limits are listed in Table 1. The time dependent values are adjusted in-code to fit the horizon intervals.

Table 1: NMPC constraint values.

	lower limit	upper limit
z[m]	-1.4	-1.6
ϕ[rad]	-0.436	0.436
θ[rad]	-0.436	0.436
ψ[rad]	-0.436	0.436
p[rad/s]	-21	21
q[rad/s]	-21	21
r[rad/s]	-33.6	33.6
T[N]	-19.621	0
δ_T[N/s]	-80	80

While constraining the attitude this tightly seems counter-intuitive at first when one considers a racing application. It was a necessity of the limited flight testing space available, where the velocities higher attitude angles would enable are simply unfeasible. These constraints should be relaxed if a sufficiently large testing area can be acquired. It additionally was assumed that the UAV is unable to generate positive thrust. Finally the cost function values are presented in Table 2.

These values were obtained after extensive testing. The testing was almost exclusively done by trial and error, as explained in Subsection 2.2, considering 2 objectives. The first being appropriate racing behaviour, an expression of this is the lower cost of θ in comparison to the other attitudes. This means a doubled likelihood of the UAV pointing towards the set reference position. The absolute value of the weights is not very important, it is the relative weights that matter. The choice of using integers simplifies this process.

Table 2: NMPC cost function weight values.

Term	Value
x	20
y	20
z	20
v_x	3
v_y	3
v_z	3
ϕ	10
θ	5
ψ	10
r	0

Since the context is UAV racing the position must be of greatest relative importance in the cost function and the velocity must be lowest. The attitude values are chosen as such to slightly nudge the behaviour into desired behaviour. To perform complex racing manoeuvres it was found that smart choices in reference trajectory was a far more efficient way of achieving good racing behaviour than re-tuning the cost function for each manoeuvre. It must be pointed out however that this is a possibility. The second objective of choosing the weights is for the solution to be as close to optimal as possible. This objective reinforces the need for the position cost to be the comparative highest. It also requires the velocity to be slightly constrained, comparatively lowest. This will not significantly slow down the flight and eliminates any overshoot.

2.4.3 Integrator Options & Limits

The NMPC requires the choice of an integrator in order for the prediction process. In addition there are tolerances and limits for when the computed solution accuracy is deemed acceptable. The integration scheme selected is a Runge-Kutta-Dormand-Prince integrator with order 4 with a step size control order of 5. With an absolute and relative tolerance of $1e^{-8}$. The maximum number of optimization iterations was set to 3. Generally the algorithm only needs 1 to achieve convergence so this is purely for extra redundancy. The Karush-Kuhn-Tucker(KKT) [23] tolerance, a first order derivative test of interval match, was set to $1e^{-10}$.

3 Testing

The performed research consisted of several parts. The first consists of proving the possible time gains of the application of an NMPC, verifying the approach. Followed by an overview of the validation set-up. And finally a step-by-step explanation of the testing process and its implications.

3.1 Verification

Before any real testing can be done a proof of possible improvement in terms of time needed to be given. This was achieved by comparing the NMPC with both a PID controller, provided by Li, S. Controller PID gain values can be found in the release[4], and a time optimized OCP of a set trajectory for the UAV. For the NMPC there are 2 versions. An older

[4]https://github.com/ls90911/drone_simulation

version used for the first comparison during development and the final ,optimized for the problem at hand, version. A side-by-side comparison of the performance of each method can be found in Figure 4.

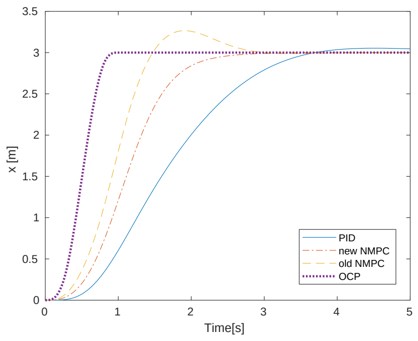

Figure 4: Trajectory control method performance comparison

The differences in performance are obvious. It is of note that the OCP solution magnifies any assumptions and model errors made and it is very sensitive to the change in the amount of intervals, in this case 20 as explained in Subsection 2.3. This explains the far larger accelerations of the OCP. Furthermore the difference between the NMPC solutions is mostly due to how the velocity is represented in the cost function. While it may seem logical to leave velocity unconstrained when optimizing for time in practice the overshoot will cause it to be slightly slower. Finally, as expected, the PID controlled solution performs worst in both terms of time and accuracy.

To more formally test possible time-gains solutions were computed for the OCP, (most recent)NMPC and PID methods for 36 different trajectories. These desired trajectories were spaced out on a 6-by-6, 1 meter grid. The times recorded are the times where the computed positions have an error smaller than 1% w.r.t. the goal. A histogram of the results can be found in Figure 5.

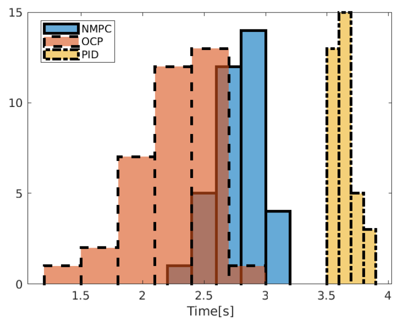

Figure 5: Trajectory times of control methods comparison

Mirroring the results presented in the side-by-side example the possible time gain of applying the NMPC technique compared to a PID approach is substantial. It also confirms that the NMPC does not reach the true optimal trajectory. It is however deemed close enough to merit the trade-off for more flexibility.

3.2 Validation

To validate the proposed method 2 stages of further testing are performed. First a test in the Gazebo[5] flight simulator followed by an real-world flight test in the TUDelft Cyberzoo. As stated before the platform used is a Bebop1 UAV. Running on the platform is the open-source autopilot Paparazzi[6]. The Cyberzoo environment is equipped with a high fidelity optical tracking system(OptiTrack) to take accurate measurements of the performance of the UAV. In Figure 6 an overview of the testing environment and the OptiTrack systems is provided.

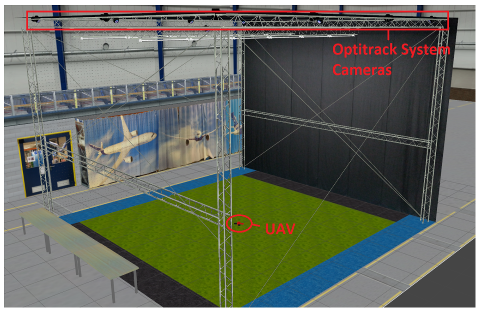

Figure 6: Cyberzoo testing environment; Gazebo visualization with highlighted OptiTrack measurement system components

The test trajectories will be the exact same as the ones tested in simulation. To facilitate the execution of the NMPC generated commands an adjustment of the stabilization loop of Paparazzi was made. This adjustment forced the UAV to follow the attitude commands as dictated. Sadly the thrust was not directly controllable. An attempt was made to scale the thrust commands, this however lead to an apparent positive feedback loop causing the thrust to keep increasing beyond an intended stable value. Therefore the thrust during testing was fully controlled by the Paparazzi indi stabilization and not the NMPC calculated values.

There is however a consideration to be made when using the OptiTrack system and the Simulation of the Cyberzoo. The neutral heading(0degrees) is set to point due north. In simulation this can simply be adjusted by tweaking the environment model. The Optitrack system however does not allow for easy modification. This unfortunately means that setting heading angles is not as straightforward as one would like. The choice made here is to mostly ignore this when building the NMPC or setting test trajectories since as long as the trajectory still fits in the 'rotated' test environment the results will be the same. This means that when looking at the graphs for the flight tests it seems that there is a huge heading bias present at initialization. This will, as stated, not impact test integrity or performance in any way.

[5]http://gazebosim.org/
[6]https://wiki.paparazziuav.org/wiki/Main_Page

3.3 First Flight Test Results

In this subsection the results of the gazebo simulations and real world flight tests will be explained. This is done by splitting the results into the first and second flight test. Not all test trajectories simulated/flown will be shown only a representative subset. This subset consists of a time history of the x-position and a x-position error histogram. The choice of the x-position is made since a displacement in x-direction is present in every test trajectory and is representative of overall behaviour. For reference all data and plots of every test can be found here[7].

3.3.1 First Simulation Test

The testing process consisted of several steps. The first was to establish a set of desired trajectories that would be used as a reference during the tests. Then these would be used as the reference trajectories to the NMPC implemented in ACADO to generate a set of inputs that could be used in both simulation and the real world. First these inputs were provided to the GAZEBO flight simulator in order to test the stability of the UAV executing said inputs. In order for the autopilot to execute these pre-determined inputs changes were made to the software running on the UAV. These included the reading of a text file containing the inputs, saving said inputs in RAM for quick access. And interrupting the stabilization module to insert the desired attitude commands. The goal for the simulation of the first flight test was to test whether the adjustment of the stabilization module of Paparazzi worked correctly and, more importantly, to check whether the aircraft would follow the set trajectory without becoming unstable. First the procedure was tested through checking the attitude response to the NMPC calculated attitude commands illustrated in figure 18 in Appendix B. The commands are followed well, only some initial biases are present. in Figure 7 a histogram of the error of the pitch is presented, which is representative of the performance as the initial bias is smallest.

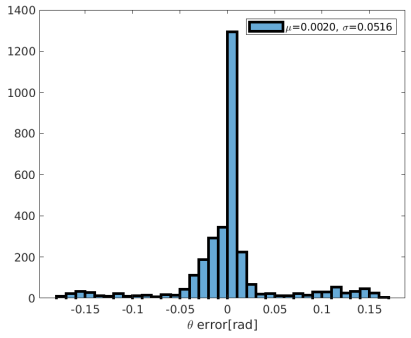

Figure 7: θ input error simulation test 1

These results show clearly that the UAV is able to follow the commanded attitude commands with satisfactory accuracy($\sigma \approx 3deg$). While the influence of input delay is noticeable the error is acceptable. While a minor position error is to be probable it is expected to fall within $\approx 0.1m$. It is recommended however that in further development the effects of input lag should be ameliorated. The position plot, found in Figure 8 however does display some significant error from the intended path.

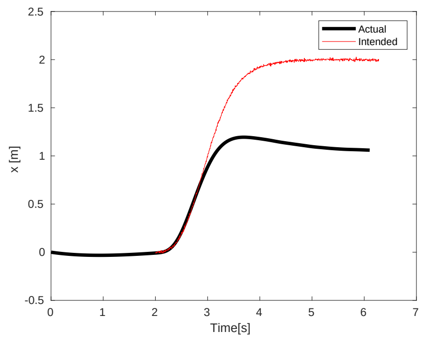

Figure 8: Simulation flight test 1 Trajectory 1, x-position intended-actual comparison

The response is almost exactly half of what was expected. That being said the simulated results are very stable. This points to a model mismatch where the acceleration is less in simulation but the overall behaviour is consistent and stable enough for the real flight test. While the large undershoot in term of position was cause for concern it was deemed preferable to overshoot in terms of stability and safety. This behaviour can be observed in almost all planned trajectories except for the trajectories 4 and 9. They both crashed in simulation and where thus deemed unsuitable for real-life testing. In figure 9 a histogram of the position error in x-direction is shown.

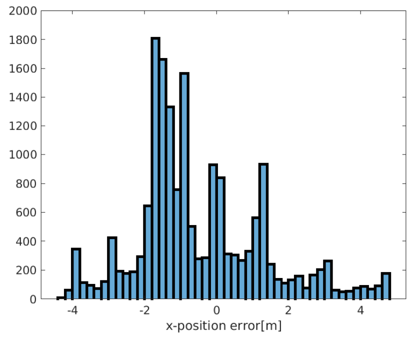

Figure 9: Simulation test 1, x-position error histogram

This data clearly shows a trend of undershoot. The peak around $-2m$ to $-1m$ confirms the consistent nature of this undershoot. The simulated results where stable enough to go ahead in testing however, with the expectation of observing similar behaviour in the real world.

3.3.2 First Real-World Flight Test

With acceptable simulation results for most of the pre-determined trajectories the first flight test was conducted. In Figure 10 a histogram of the pitch input error is shown.

[7]https://github.com/stspronk/Thesis-Test-Data

The corresponding attitude response plot can be found in Figure 19 in Appendix B.

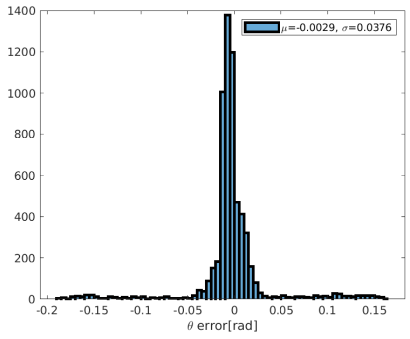

Figure 10: θ Input error flight test 1

The input error of the actual flight is small enough($\sigma \approx 2deg$) to conclude both that the adjustment of the stabilization algorithm is functioning well and that the NMPC input constraints are representative of the performance of the UAV. However during testing an unexpected behavioural trend was observed, an example of this trend is presented in Figure 11.

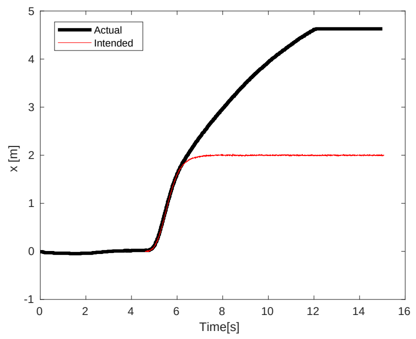

Figure 11: Real world flight Test 1 Trajectory 1, x-position intended-actual comparison

The behaviour is modelled well by the NMPC until the UAV needs to decelerate and come to a complete stop. This points to a significant mismatch between the model used in gazebo and paparazzi and it's real life counterpart. The model used by the NMPC looks thus to better match the gazebo simulation than the real life aircraft. From this first example it is clear that overshoot is a problem. Of special note is the velocity in x direction v_x which never returns to 0. While the UAV does follow the commands and the commands do prompt the UAV in the right direction, the overall performance is unacceptable in a racing environment. The same behaviour can be observed when in the other test trajectories. The histogram of the cumulative position error in x-direction can be found in Figure 12.

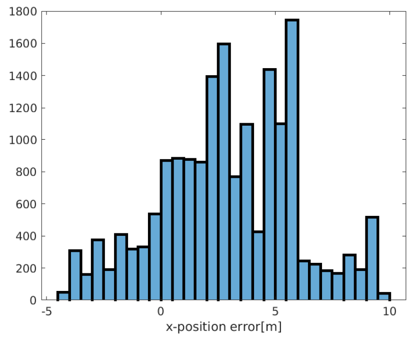

Figure 12: Real world flight test 1, x-position error histogram

A few conclusions can be made based on these results. The first being, as stated before, that the model used in simulation(both Gazebo and the NMPC) does not fit reality accurately enough to enable the crisp execution of these aggressive manoeuvres. The aspect of the model that seem most influential to this problem is the inclusion of drag. As can be seen from the attitude and velocity graphs the breaking manoeuvre is significantly smaller than the acceleration. It was decided that the model be adjusted to generate a larger deceleration manoeuvre to ameliorate the problem, and that this would be tested in a second flight test. The second conclusion is that the rest of the behaviour is modelled well and that the proposed method of racing is thus tentatively viable.

3.4 Second Flight Test Results

This subsection contains the results of the second round of tests both simulation and flight tests. As before first the changes made were tested in simulation and their results explained first. Followed by the results of the actual second flight test.

3.4.1 Second Tests Preparations

During the first flight test a couple of problems were identified. Namely a large post manoeuvre drift. This behaviour leads to large errors and a condemns the used inputs to be judged as unfeasible for a racing application. Ameliorating these issues was deemed to be a fairly simple process. Therefore it was decided work to be put into improving the performance of the flight behaviour. To enable this several changes were implemented. The first being to eliminate the terms describing drag from the model ODE's. Secondly it was decided to remove the noise from the generated NMPC input as robustness had been proven and it could be concluded from the first simulation and flight tests that it did not impact performance. Lastly it was decided that the test trajectories would be amended slightly for the second test. The first trajectory's goal was extended from $2m$ in x-direction to $4m$, and renamed 11. This was done to be able to more clearly observe any behaviours present. Furthermore 2 more trajectories were added in order to test influence of differences in heading on the same position goals. The updated list can be found in Table 5 in Appendix A.

3.4.2 Second Simulation Test

Since the first flight test showed a unacceptable amount of overshoot changes to the model used by the NMPC were made. With the changes the objective for the second round of simulation is to observe a behaviour of 'over-breaking', where the simulated UAV reverses over the set trajectory. This would indicate that the overshoot observed in the real world would be either eliminated or at least severely lessened. Some lessoned learned from the first flight test were also implemented. The foremost being the realization that the system while promising does not have the required accuracy to generate more complex manoeuvre inputs. Therefore most of the pre-determined test trajectories were deemed to complex and not included in te second test. To compensate for this two additional manoeuvres were added. This to still have a large enough basis of tracks for drawing conclusions.

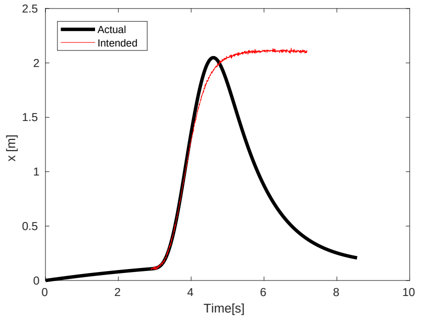

Figure 13: Simulation Flight Test 2 Trajectory 1, x-position intended-actual comparison

As can be seen the objective of the second simulation test is achieved. The x-position can be seen to almost completely return to its starting position meaning the breaking manoeuvre is far more aggressive. A clear indication that the overshoot during actual flight would be reduced or eliminated. This way of predicting flight behaviour is not a very solid approach but trying to identify the exact differences between the actual flight dynamics and the Paparazzi model was deemed beyond the scope of the project. It is hypothesized that the model is not very well optimized for simulating aggressive behaviour dictated by time optimal trajectories.

Some trajectories however performed extremely well in simulation which was cause for concern. trajectory 3, track details found in Table 4 in Appendix A, is a prime example of this. This result was reason for this trajectory not to be included in the actual flight test due to the high likelihood of overshoot and collision with the Cyberzoo borders. The 5th trajectory also exhibited 'too good' behaviour. However since this manoeuvre is less complex than the third and far less likely to thus crash it was decided to include it in the actual flight test.

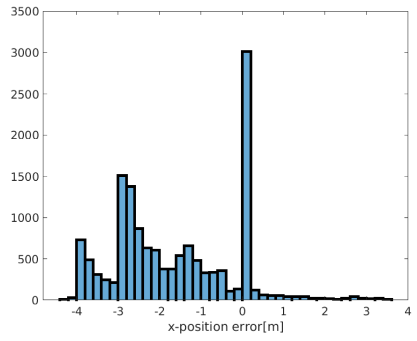

Figure 14: Simulation test 2, x-position error histogram

All other results showed similar results to trajectory 1, as proven by Figure 14. Exhibiting an over-breaking trend. Therefore the changes made for the second simulation test seem to be successful, having the intended effect.

3.4.3 Second Real-World Flight Test

This section will present and explain the data gathered during the second flight test. The main aim of the test was to confirm that the changes made to the model used by the NMPC to generate the trajectories were effective. As with the other simulation and flight test results relevant examples will be provided and discussed here. Figure 15 shows the representative result of the second test.

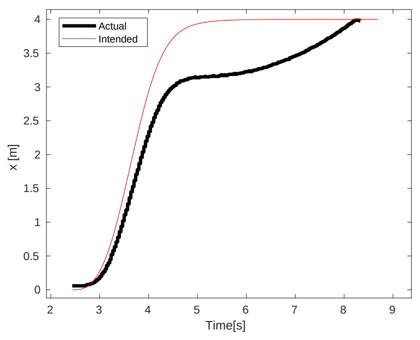

Figure 15: Real World Flight Test 2 Trajectory 11, x-position intended-actual comparison

It can be seen that the drift is still present but much reduced. Overshoot is replaced by a slight undershoot, but the intended trajectory is followed far more closely than before. Overall the performance is far more stable. The results of the other test trajectories show the same overall behaviour. In Figure 16 the histogram of the x-position is illustrated, showing a large general improvement when compared to Figures 9, 12 and 14. This is proof that the elimination of the drag term has had a positive effect on performance. The undershoot present does imply that the inclusion of drag cannot be eliminated wholesale, but that it's effect is limited. It is recommended that for further development a more accurate drag coefficient is identified and implemented.The error can be attributed to the model mismatch again as the inputs are followed extremely accurately.

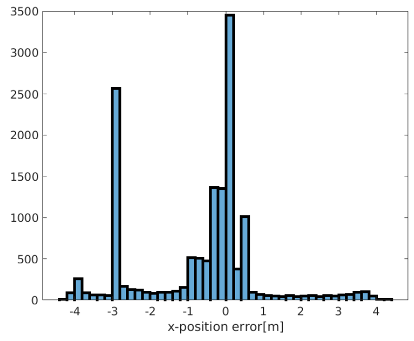

Figure 16: Real world flight test 2, x-position error histogram

In addition to this the inability to control the thrust to the same level as in the NMPC creates another source of error. From the test data it can be clearly observed that altitude decreases in every simulation and flight test while this not being the case for the NMPC intended trajectory. Therefore it is expected that if the thrust could be controlled directly the performance would also increase. These are very promising results as it is believed that the performance can be greatly increased by improving the model accuracy. It is evidence that the approach is valid and that it can be used to improve the achievable velocities, even in complex manoeuvres.

3.5 PID Controller Flight Test

This section will present the trajectory flown using the Paparazzi autopilot built-in flight planner. Which uses a PID controller. The default gain values for the bebop1 provided by the Paparazzi autopilot are presented in Table 3

Table 3: Paparazzi autopilot guidance PID gains

Gain	Vertical	Horizontal
P	283	79
I	82	100
D	20	30

The result of the test is presented in Figure 17

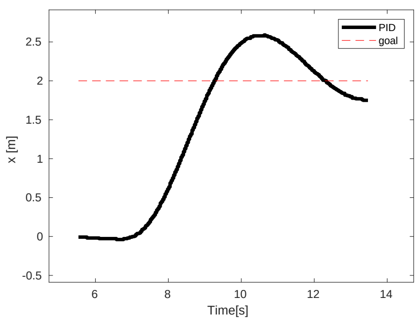

Figure 17: PID Test, x-position result-goal comparison

The difference when comparing to the results of the NMPC flight tests is stark. The NMPC provided trajectory does exhibit overshoot and drift due to it being an open-loop manoeuvre. Unexpectedly the PID also exhibits a large overshoot of the goal in addition to a far larger deviation from the intended path in y-direction. Overall the velocities are also significantly lower. The PID reaches a maximum velocity in x-direction of $1.2m/s$, while the NMPC reaches a maximum velocity in x-direction of $2.6m/s$. Considering this difference will decrease if the overshoot of the NMPC is reduced due to model improvements it is still an extreme difference, and it is expected to stay as such.

4 Results

The final result is a system consisting of various parts which will be addressed separately. First an implementation of an NMPC that is able to implicitly optimize for time through a complex and dynamically changing set of position objectives enabled by the adaptive reference trajectory system. Secondly The final implementation of the proposed racing trajectory generation and control method is able to perform a varied set of racing objectives in significantly less time than the more classical PID control method. However there are persistent, yet consistent, model errors causing improper deceleration and thus position errors. The proposed method has shown to be able to decrease flight times by $\approx 1s$ in very short range manoeuvres ($2m - 4m$). The difference in velocity confirms this as the PID reaches a maximum velocity of $1.2m/s$ while the NMPC reaches $2.6m/s$.

5 Conclusion & Recommendations

From the results it can be concluded that the proposed approach is capable of improving the overall performance in terms of time of an autonomous racing UAV . While the real-world results definitely show the desired behaviour, the error is still too high to be deemed racing-viable. This error can be attributed to model errors, in the form of an unknown drag coefficient and the lack of direct thrust control on the real-world UAV. Furthermore it has been shown that the Gazebo simulation using the Paparazzi model for the chosen UAV does not reflect reality in any acceptable capacity. This combination of flight simulator and dynamic model should therefore not be used to predict the behaviour of the kind of aggressive manoeuvres tested.

It is recommended to perform a thorough systems identification on the actual UAV that will be used to race. The acquisition of accurate UAV weight and drag coefficient values is a priority in this endeavour. Also in further development of this method the thrust difference should be addressed either by reflecting the Paparazzi thrust behaviour in the NMPC or by taking more direct control in the thrust control of Paparazzi to better execute the NMPC commands. The effects of input lag should also be reduced. The system is capable of being adapted in an on-line capacity through adapting the ACADO NMPC implementation to be able to interpret the Paparazzi provided state estimation. Code profiling should be performed to asses computational viability. Additionally the system can be used as a basis for a neural network approximation of the proposed NMPC racing method.

References

[1] Iros 2018 autonomous drone racing competition. `http://rise.skku.edu/iros2018racing/index.php/rules-regulations/`. accessed: 28-11-2018.

[2] Alphapilot – lockheed martin ai drone racing innovation challenge. `https://www.herox.com/alphapilot/82-teams`. accessed: 18-07-2019.

[3] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido CHE de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. *arXiv preprint arXiv:1905.10110*, 2019.

[4] Sunggoo Jung, Sungwook Cho, Dasol Lee, Hanseob Lee, and David Hyunchul Shim. A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge. *Journal of Field Robotics*, (February 2017):146–166, 2017.

[5] Sunggoo Jung, Hanseob Lee, Sunyou Hwang, and David Hyunchul Shim. Real Time Embedded System Framework for Autonomous Drone Racing using Deep Learning Techniques. *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, (January), 2018.

[6] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018.

[7] Pete Florence, John Carter, and Russ Tedrake. Integrated Perception and Control at High Speed: Evaluating Collision Avoidance Maneuvers Without Maps. *Wafr*, 2016.

[8] Hector D. Escobar-Alvarez, Neil Johnson, Tom Hebble, Karl Klingebiel, Steven A.P. Quintero, Jacob Regenstein, and N. Andrew Browning. R-ADVANCE: Rapid Adaptive Prediction for Vision-based Autonomous Navigation, Control, and Evasion. *Journal of Field Robotics*, 35(1):91–100, 2018.

[9] Michael Neunert, Cédric De Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June(ICRA):1398–1404, 2016.

[10] P Bouffard, A Aswani, and C Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. *IEEE International Conference on Robotics and Automation (ICRA), 2012*, pages 279–284, 2012.

[11] Chi-Kin Lai, Mudassir Lone, Peter Thomas, James Whidborne, and Alastair Cooke. On-board trajectory generation for collision avoidance in unmanned aerial vehicles. *2011 Aerospace Conference*, pages 1–14, 2011.

[12] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019.

[13] Daniel Mellinger, Alex Kushleyev, and Vijay Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 477–483, 2012.

[14] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux meeting on systems and control*, volume 11, pages 119–141. Technische Universiteit Eindhoven Veldhoven Eindhoven, The Netherlands, 2002.

[15] D.E. Seborg, T.F. Edgar, D.A. Mellichamp, and F.J. Doyle III. *Process Dynamics and Control*. 3rd edition, 2015.

[16] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[17] B. Houska, H.J. Ferreau, M. Vukov, and R. Quirynen. ACADO Toolkit User's Manual. http://www.acadotoolkit.org, 2009–2013.

[18] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[19] B. Houska, H.J. Ferreau, and M. Diehl. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10):2279–2285, 2011.

[20] M. Vukov, W. Van Loock, B. Houska, H.J. Ferreau, J. Swevers, and M. Diehl. Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation. In *The 2012 American Control Conference, Montreal, Canada.*, 2012.

[21] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl. Auto-generated Algorithms for Nonlinear Model Predicitive Control on Long and on Short Horizons. In *Proceedings of the 52nd Conference on Decision and Control (CDC)*, 2013.

[22] H.J. Ferreau, T. Kraus, M. Vukov, W. Saeys, and M. Diehl. High-speed moving horizon estimation based on automatic code generation. In *Proceedings of the 51th IEEE Conference on Decision and Control (CDC 2012)*, 2012.

[23] Hsien-Chung Wu. The karush–kuhn–tucker optimality conditions in an optimization problem with interval-valued objective function. *European Journal of Operational Research*, 176(1):46–59, 2007.

A Test Trajectories

Table 4: Test 1 Trajectories

Track #	x-position[m]	y-position[m]	Heading[degrees]	Special Conditions
1	2	0	0	-
2	4	0	-90	-
	4	4	-90	
3	2	0	0	-
	2	2	0	
	0	2	0	
	0	0	0	
4	3	0	90	-
	3	3	180	
	0	3	270	
	0	0	0	
5	3	2	180	-
6	4	4	45	-
7	4	0	180	-
	4	3	180	
	0	3	180	
8	4	0	0	-
	4	0	90	
9	4	0	0	Roll and Pitch constraints of 45deg
10	4	0	0	No noise
	4	0	90	

Table 5: Test 2 Trajectories

Track #	x-position[m]	y-position[m]	Heading[degrees]	Special Conditions
2	4	0	-90	-
	4	4	-90	
3	2	0	0	-
	2	2	0	
	0	2	0	
	0	0	0	
5	3	2	180	-
11	4	0	0	-
12	3	2	0	
13	3	0	0	-
	3	0	90	

B Attitude Response-Command Comparisons

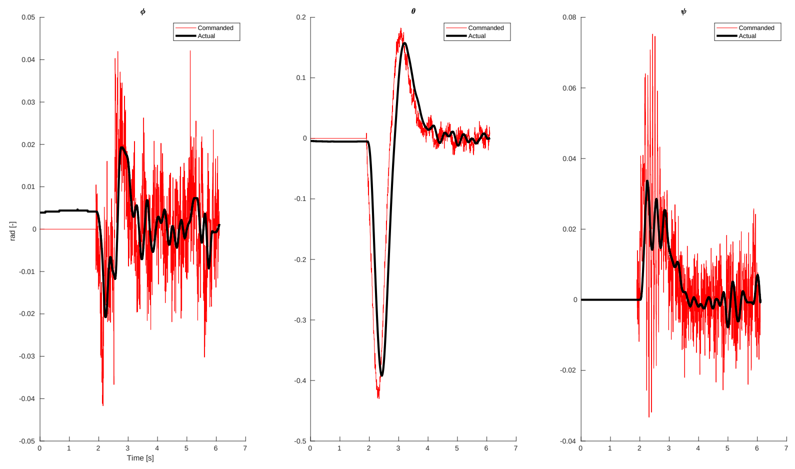

Figure 18: Simulation test 1 track 1, attitude command-response comparison

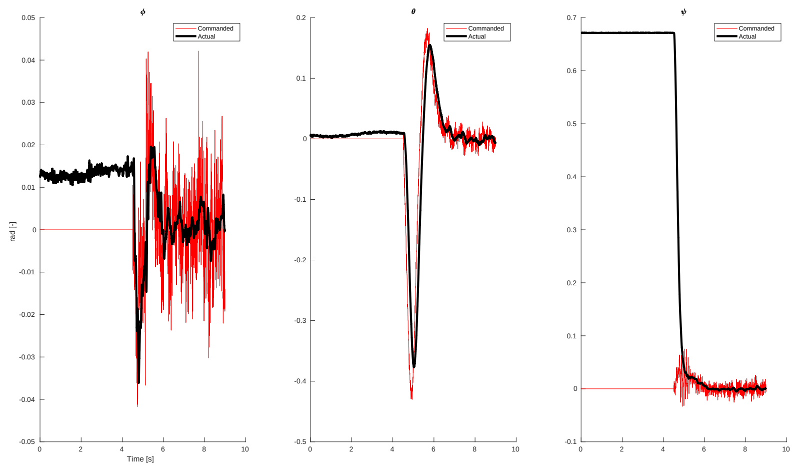

Figure 19: Real world flight test 1 track 1, attitude command-response comparison