Model-Free trajectory planning for a rotary-wing unmanned aerial vehicle with an uncertain suspended payload

Annika Hohbach¹, Hendrik Willem Jordaan¹ and Japie Engelbrecht¹
¹Electrical and Electronic Engineering Department, Stellenbosch University, Banghoek Road, Stellenbosch, 7600, South Africa

Abstract. Payload transport using rotary-wing unmanned aerial vehicles (RUAVs) has grown in popularity. Suspending the payload from the RUAV expands the range of use cases but does so at the expense of changing the dynamics that may result in instability. This paper explores how a model-free solution can be used to plan robust trajectories that account for the added oscillations if the payload is varied. The twin-delayed deep deterministic policy gradient (TD3) algorithm is a model-free reinforcement learning method and is used to train an agent to be implemented as a local planner to plan optimal trajectories that can complete two defined circuits whilst minimizing the swing of the payload. A non-linear model predictive controller (NMPC) is implemented as a model-based solution to evaluate the capabilities of the model-free results determining its viability, without choosing a superior approach. The results indicate that the model-free TD3 agent has comparable results to the model-based NMPC for two defined circuits, with increased robustness to payload uncertainty when trained with different payload parameters.

1 Introduction

Rotary-wing unmanned aerial vehicles (RUAVs) are increasingly used to complete hazardous or challenging tasks. Most current use cases fix a payload to the RUAV body, to ensure that the dynamics are minimally affected. This attachment method limits the shape and size of the payload as it must match the capability of the RUAV. A larger variety of payloads can be transported if suspended via a rod or cable. However, this causes the system model to significantly change as the swinging payload adds oscillations that could lead to instability if aggressive maneuvers are performed. The two commonly discussed solutions are to design a controller that actively dampens vibrations during flight or to plan for swing-minimizing trajectories. Both of these are frequently observed to use a model, either for controller design or to predict future behavior. The model's accuracy has a significant impact on how well a solution works, thus any uncertainties can result in performance degradation. Since suspended payload transport increases the variability in the payload, uncertainties in the model are expected.
Most current research lies in active damping control techniques which aim to reduce residual oscillations at every time step. This might not be the best behavior for a task. In addition to damping the oscillations, trajectory planning allows for task-specific planning that provides the system with some foresight about decisions that must be made before the controllers execute them. A swing-minimizing pre-planned trajectory also allows the standard PID controllers, which are found on most commercial RUAVs, to be used for trajectory tracking. Further consideration should be given to trajectory planning with variable planning models, which could provide additional robustness to the changing or uncertain payload parameters that could result from suspending a payload from a RUAV.

The specifications used to define an optimal trajectory in the context of what must be achieved, determine the chosen trajectory planning method. Optimization methods are popular due to being able to solve for specific non-linear dynamics and a variety of path and system constraints [1]. Many current trajectory optimization implementations are for the RUAV without a suspended payload case [2, 3].

A cost function that describes specific tasks can be used to generate short optimal trajectories, with many implementations not addressing the execution thereof [3]. The problem of execution can be addressed with feedback controllers, but accurate system knowledge or iterative refinement of the controllers is essential for the successful execution of trajectories, especially through narrow spaces [2]. Since the computational burden of optimization methods is a big concern for online planning, the definition of the dynamics and path constraints must be adjusted to ensure fast computation [1, 3]. In many cases this consists of linearizing the system dynamics [4, 5]. The attached payload introduces non-linearities that are important to the system's dynamics, and linearized planning or tracking models cause a loss of system information that compromises the trajectory's execution. An extensive model of the system can be established by analyzing the three-dimensional kinematic equations and making use of the payload parameters such as mass and cable length [6]. Non-linear models to predict future motion are a better representation of the physical system dynamics [8], but the payload parameters must still be known to ensure good performance.

Model-free methods have the potential to address the performance deterioration caused by model uncertainties. Model-free reinforcement learning (RL) has shown to have benefits in these situations as its sample-based properties allow for robust trajectory generation even if the model of the system is unknown [7]. Only a few RL-based solutions for the specific case of a RUAV and suspended payload are found in the literature [9, 10, 11, 12, 13].

Good results using RL without the use of neural networks have been seen [9, 10, 11], but these usually make use of discrete action spaces which limits the system's ability to perform a variety of tasks. RL has recently been paired with deep learning to accommodate larger state spaces and continuous action spaces, which is not possible with regular RL. In a hybrid system, RL is occasionally used to tune the controller to complete the desired goal rather than planning or executing the trajectory [12]. This setup is beneficial for controllers that have a rigorous design process such as non-linear controllers but usually require all system states to be available in real-time. Additionally, this setup underutilizes the RL agent's ability to plan for specific environments.

The state-of-the-art deep RL algorithm for continuous autonomous control is currently the twin-delayed deep deterministic policy gradient (TD3) [14]. This algorithm has been applied to RUAV navigation [15, 16], but, to the best of our knowledge, only one application to a multi-rotor with a suspended payload is available [13]. The TD3 agent is used to replace the
position PID controller as a swing-minimizing controller and shows good tracking of a variety of 3D waypoints with significant swing reduction. This paper emphasizes the minimal prerequisites needed for implementation on different RUAVs as only the position controller is substituted. The onboard controllers on most commercial RUAVs are integrated into the flight control firmware, making it difficult to replace parts of it. Thus, adding trajectory planning allows for an even simpler implementation. This work also does not comment on the effect of varying payload parameters which come with uncertainty in the payload.

In this paper the TD3 algorithm is applied to the RUAV carrying a suspended payload problem as a local planning solution and compared to the well-researched non-linear model predictive controller (NMPC), to assess the viability of a model-free solution for trajectory planning. To better understand the performance enhancement that trajectory planning gives, both are contrasted with the RUAV's baseline behavior. Unlike the previous research the TD3 agent is also evaluated for different payload parameters, to ascertain the offered robustness to varying payloads.

2 Problem Formulation

A system overview of conventional RUAV systems is shown in Fig. 1. The global planner receives a map of the environment and produces course waypoints to reach a goal location. It is assumed in this paper that a global planner exists which produces an initial solution through an obstacle-filled environment. The TD3 agent and the NMPC method, that are described in Section 4, are trajectory planning methods that make up the local planner block and are the focus of the paper. The tracking controller block consists of simple PID controllers, which are used to track the generated trajectories and are part of the flight control software. The baseline behavior of the system is defined as the response produced by excluding the local planner from the architecture and only using controllers to track the global waypoints. The first set of tests is performed on a mathematical non-linear testing model with drag and added sensor noise implemented in MATLAB and Simulink, which is based on an available practical system and is further discussed in Section 3. In addition, a Robot Operating System (ROS) with Gazebo simulation is then used for testing as it runs the PX4 flight software found on the practical RUAV.

The model-free and model-based methods are tested and compared on two different circuits. The first circuit consists of a four-waypoint circuit in the $\mathbf{x},\mathbf{y}$-plane and shows no change in altitude. The second circuit tests the local planner in a more constrained environment such as a $4 \times 4$ m tunnel with horizontal and vertical changes of direction.

3 Modelling

![Fig. 1. The system architecture of conventional autonomous RUAV systems to indicate where the trajectory planning solutions are implemented.](image_url)
3.1 Full non-linear testing model

The mathematical model of the RUAV used to produce the simulation results in this paper is based on an available practical system, depicted in Fig. 2, built by [17] and the derivation thereof is based on the work done by [18].

![Fig. 2. The practical RUAV, Honeybee, of the Electronic Systems Laboratory at Stellenbosch University with an attached suspended payload.](image)

The RUAV and suspended payload system is modelled as two separate systems, with both being rigid bodies. The relevant axis used to describe the dynamics are the inertial frame, \( I\{\hat{x}_I, \hat{y}_I, \hat{z}_I\} \) defined in the North-East-Down (NED) direction, and the body frame, \( B\{\hat{x}_B, \hat{y}_B, \hat{z}_B\} \), that has its origin at the RUAV centre of mass (CoM) as depicted in Fig. 3. The forces and moments acting on the RUAV body (\( F_B \) and \( M_B \)) comprise of the gravity force, the aerodynamic forces and moments caused by drag and wind, the thrust forces and moments and the forces caused by the acceleration of the payload. Newton's second law states that the resulting effect causes motion in the body frame:

\[
F_B = m_r \dot{v}_B + \omega_B \times m_r v_B , \quad \text{and} \quad M_B = I_\omega \dot{\omega}_B + \omega_B \times I_r \dot{v}_B ,
\]

where \( \dot{v}_B, v_B, \dot{\omega}_B \) and \( \omega_B \) are the RUAV acceleration, velocity, angular acceleration and angular velocity vectors in the body frame, and \( m_r \) and \( I_r \) are the mass and inertia matrix of the RUAV.

The motion of the body frame is then related to the inertial frame by making use of the direct-cosine-matrix (DCM) as follows:

\[
v_B = R^T_B v_I ,
\]

where the DCM, \( R^T_B \), makes use of quaternions.

3.2 Reduced planning model

To plan optimal trajectories, model-based solutions make use of a reduced model due to the high computational burden that predicting an online trajectory can have. The derivation of a planning model is made possible under the assumption that the attitude changes of the RUAV have little effect on the system's response to the additional forces of the suspended payload. Thus, the rotational dynamics of the RUAV are decoupled from the translational dynamics. This assumption is proven to be valid in [19]. The full non-linear testing model is depicted on the left in Figure 3 and the reduced planning model is seen on the right, with the payload swing angles, \( \alpha \) and \( \beta \), shown for each model. In addition to being used to plan trajectories on the left in Figure 3 and the reduced planning model is seen on the right, with the payload swing angles, \( \alpha \) and \( \beta \), shown for each model. In addition to being used to plan trajectories...
in the model-based NMPC method, the planning model can be used for accelerated training of the model-free TD3 agent.

A few assumptions about the planning model are that the point of attachment is modelled as a low-friction joint, the payload is a rigid point mass and is attached to the RUAV by a rigid rod and the rod is massless in comparison to the much larger payload and RUAV masses. The Lagrangian of the suspended payload system consists of subtracting the potential energy from the kinetic energy:

\[ \mathcal{L} = \frac{1}{2} m |\mathbf{p}_p|^2 + mgz_p, \tag{4} \]

where \( m \) is the mass of the payload and \( g \) is the gravitational constant. The payload position can be defined with respect to the RUAV CoM position in the inertial coordinate frame \( \mathbf{p}_p = [x_N \ x_E \ x_D]^T \) as:

\[
\mathbf{p}_p = \begin{bmatrix}
\dot{x}_p \\
\dot{y}_p \\
\dot{z}_p
\end{bmatrix}
= \begin{bmatrix}
x_N + l \cos \alpha \sin \beta \\
x_E + \sin \alpha \\
x_D + l \cos \alpha \cos \beta
\end{bmatrix}, \tag{5}
\]

The Lagrangian is then used in the Euler-Lagrange equation described by:

\[
\frac{d}{dt} \left( \frac{\partial \mathcal{L}}{\partial \dot{s}_i} \right) - \frac{\partial \mathcal{L}}{\partial s_i} = Q_i, \tag{6}
\]

where \( s_i \) refers to the different payload states which are:

\[
\mathbf{s}_p = [x_p \ y_p \ z_p \ \alpha \ \beta]^T, \tag{7}
\]

and \( Q_i \) specifies the non-conservative forces, which are chosen as the forces exerted in the \( \mathbf{x}_3, \mathbf{y}_3 \) - and \( \mathbf{z}_3 \) -directions \( (f_x, f_y \ \text{and} \ f_z) \) as seen in Figure 3. The friction of the attachment point and any damping caused by the shape of the payload are also added to the non-conservative forces resulting in the following set of equations:

\[
Q = \begin{bmatrix}
f_x - \frac{1}{2} \rho C_p \dot{x}_p^2 \\
f_y - \frac{1}{2} \rho C_p \dot{y}_p^2 \\
f_z - \frac{1}{2} \rho C_p \dot{z}_p^2 \\
-c \dot{\beta} - \frac{1}{2} \rho C_p \dot{y}_p^2 \cos \alpha \\
-c \dot{\alpha} - \frac{1}{2} \rho C_p \dot{x}_p^2 \cos \beta
\end{bmatrix}, \tag{8}
\]
where $\rho$ is the air density, $c$ is a friction coefficient and $C_p$ is a drag coefficient that is defined to be the same for all directions given the assumption that the payload is a rigid point mass.

4 Trajectory Planning

In the field of robotics, trajectory planning is used to guide a robot to accomplish a goal by selecting the best course of action [20]. The type of system and what must be achieved determines how the optimality of the trajectory is defined. The trajectory in this paper should complete a defined circuit with robustness to varying payload parameters whilst minimizing the swing of the oscillating payload to ensure stable flight.

4.1 Model-free trajectory planning

Given the defined problem, an appropriate model-free RL algorithm is chosen. A continuous action space allows for a more optimal solution to be found. The increased dimensionality of the system due to the added payload must also be accounted for. Actor-critic methods are most suited if an optimal solution is required given complex systems and tasks. The TD3 algorithm has shown the best stability given large action spaces [21] and is thus chosen.

Figure 4 shows the setup of the training environment. The planning model is used to train the TD3 agent in simulation and allows for faster training due to its simplicity. The PID controllers also form part of the training environment, since the TD3 agent only functions as a local planner and the training environment must accurately represent the rest of the system components depicted in Figure 1.

![Fig. 4. The TD3 setup used to train an agent as a local planner with a reduced training environment of a RUAV and suspended payload system.](image-url)

The algorithm makes use of six neural networks which each have two fully connected hidden layers of 200 neurons each and are activated using the rectified linear unit (ReLU) activation function for better training and to account for non-linearities. These choices are based on other autonomous vehicle applications of this algorithm found in [22].
The input layer to the actor network consists of 14 normalized observations:
\[
s = [x_j \ v_j \ \dot{a} \ \dot{x}_j^{\text{ref}} \ \dot{x}_j^{\text{ref}+1}]^T,
\]
which are the inertial position and velocity vectors, the angular velocities of the payload, the upcoming waypoint to reach and the one thereafter.

It is frequently noted that RL results are difficult to predict and interpret even if the solution converges, demonstrating that it is not necessarily the most dependable option. Implementing the agent as an end-to-end solution can lead to system instability or failure, especially in a practical application. To make sure that the TD3 agent does not have full control over the system, it is implemented as a local planner, which motivates the choice of the action space. The actions space is thus chosen to be continuous velocity references in the inertial frame, bounded between -4 m/s and 4 m/s.

The choice in reward function is essential for proper training. A scalar value is returned by the environment to guide the learning process and to specify the benefit of a particular action. The reward function consists of intermediate rewards and continuous rewards at every time step and can be described as:

\[
r(s, a) = \begin{cases} 
    r_1 + r_2 + r_3 & \forall t \in [t_0, t_f] \\
    r_4 & \text{if } d_k \leq d_{k-1} \\
    r_5 & \text{if } |v_t| \leq v_m \\
    r_6 & \text{if waypoint reached} \\
    r_7 & \text{if goal reached} \\
    r_8 & \text{if out of bounds} 
\end{cases} \tag{10}
\]

The first reward, \(r_1\), is a negative scalar, \(-k_1\), awarded at every time step to penalise the agent for taking too many time steps. The next two rewards, \(r_2\) and \(r_3\), penalise the energy added to the system by the swinging payload and the change in direction at every time step as such:

\[
r_2 = -k_2 \sqrt{\mathcal{T}_a^2 + \mathcal{T}_\beta^2}, \quad \text{and} \tag{11}
\]
\[
r_3 = -k_3 \sqrt{a_N^2 + a_E^2 + a_D^2}, \quad \text{and} \tag{12}
\]

where \(\mathcal{T}_a^2\) and \(\mathcal{T}_\beta^2\) are the kinetic energies added by the swinging payload and \(a_N, a_E\) and \(a_D\) are the acceleration values of the RUAV in the inertial frame.

The fourth reward, \(r_4\), is a positive scalar, \(k_4\), that rewards the agent when the current distance to the reference waypoint is smaller than the previous distance, incentivising it to move forward. The fifth reward, \(r_5\), is another negative scalar \(-k_5\) given when the magnitude of the velocity is smaller than a chosen limit to ensure that the agent does not choose to remain stationary rather than attempt to find a solution. The last three rewards, \(r_6, r_7\) and \(r_8\) are all scalar values, \(k_6, k_7\) and \(-k_8\), received if the specified conditions are satisfied.

The hyper-parameters that are chosen through trial and error can be seen in Table 1. They consist of the number of episodes trained, the mini-batch size \(D\), the update constant \(\tau\), the reward discount factor \(\gamma\), the sampling time of the environment \(t_s\), the percentage noise added to the selected action specified by the standard deviation \(\sigma\) and its clipping value \(k\).
4.2 Model-based trajectory planning

Figure 5 shows an overview of what the NMPC implementation consists of. The planning model with PID controllers is used as before. The control inputs are kept the same as the action space of the TD3 implementation. The optimizer solves for an optimal trajectory given a set number of nodes, state constraints as well as environmental constraints depending on the circuit and as specified by the global planner. The trajectory is iteratively calculated from each new state after a single control input is executed. The states consist of the inertial position, the inertial velocity, the payload swing angle, and its angular rate. The interior point optimizer (IPOPT) is used as it is a popular solver for large-scale non-linear programming problems [23].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>5000</td>
</tr>
<tr>
<td>D</td>
<td>64</td>
</tr>
<tr>
<td>(\tau)</td>
<td>0.005</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>0.95</td>
</tr>
<tr>
<td>(t_s)</td>
<td>0.1</td>
</tr>
<tr>
<td>(\sigma)</td>
<td>0.05</td>
</tr>
<tr>
<td>(k)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 1. Training hyper-parameters

5 Simulation Results

This section shows the simulation results for the model-free TD3 agent, as well as the model-based NMPC solution and compares their results. The results on the first four waypoint testing circuit make use of the non-linear testing model as described in Section 3 and are also compared to the baseline behavior of the RUAV with a suspended payload. The results through the tunnel circuit then make use of a Software-in-the-loop (SITL) Gazebo simulation that runs the same flight controller software as the practical system known as PX4.
4.2 Model-based trajectory planning

Figure 5 shows an overview of what the NMPC implementation consists of. The planning model with PID controllers is used as before. The control inputs are kept the same as the action space of the TD3 implementation. The optimizer solves for an optimal trajectory given a set number of nodes, state constraints as well as environmental constraints depending on the circuit and as specified by the global planner. The trajectory is iteratively calculated from each new state after a single control input is executed. The states consist of the inertial position, the inertial velocity, the payload swing angle, and its angular rate. The interior point optimizer (IPOPT) is used as it is a popular solver for large-scale non-linear programming problems [23].

Table 1. Training hyperparameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>5000</td>
</tr>
<tr>
<td>$m_{DOF}$</td>
<td>64</td>
</tr>
<tr>
<td>$\tau$</td>
<td>0.005</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.95</td>
</tr>
<tr>
<td>$t_s$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>0.05</td>
</tr>
<tr>
<td>$k$</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Simulation Results

This section shows the simulation results for the model-free TD3 agent, as well as the model-based NMPC solution and compares their results. The results on the first four waypoint testing circuit make use of the non-linear testing model as described in Section 3 and are also compared to the baseline behavior of the RUAV with a suspended payload. The results through the tunnel circuit then make use of a Software-in-the-Loop (SITL) Gazebo simulation that runs the same flight controller software as the practical system known as PX4.

The TD3 agent was initially trained using a constant $m = 1$ kg and $l = 1$ m payload and delivered the results for varying payload parameters as seen in Figure 6a. To improve the agent's robustness against varying payload parameters, the agent was retrained by randomly selecting payload parameters within a 0.4 kg to 1 kg range for the payload mass and a 1 m to 2 m range for the cable length after each episode. The results in Figure 6b indicate a significant improvement in tracking performance as the trajectories for each selected payload are now much more similar.

Fig. 5. All particulars that make up the NMPC local planner, required by the solver to produce optimal trajectories for a RUAV and suspended payload system consisting of the initial states, the final states, the node, path and system constraints, the cost function, and the planning model.

Fig. 6. Circuit completion of the four-waypoint circuit in the $\tilde{x}_J$-$\tilde{y}_J$-plane for the basic TD3 agent trained with a constant payload in (a) and the robust TD3 agent trained with changing payloads in (b) with variable payload parameters.

The comparison of the robust TD3 agent and the NMPC execution with the baseline behavior can be seen in Figure 7a. The baseline approaches the waypoints with straight paths, and the NMPC behavior completes the circuit similarly. The robust TD3 agent takes larger curves and thus shows a very different approach. However, both the TD3 and NMPC executions come down to tuning and can be adapted to perform in a more similar way. Figure 7b shows that the NMPC method reduces the swing angle by 39.9 deg, but achieves this by taking twice as long to complete the circuit, which indicates a slow execution on the testing model with a final time of 29.0 s. The robust TD3 agent reduces the swing angle from the baseline by 37.1
deg and has an execution time of 18.2 s, that is only 2.9 s slower than the baseline, despite the longer path taken.

![Graph showing the comparison between baseline, robust TD3, and NMPC methods in terms of position and angle.](image1)

**Fig. 7.** The circuit completion of the four-waypoint circuit in (a) and the resulting swing angle in (b) of the robust TD3 agent and the NMPC method with a $m = 1$ kg and $l = 1$ m payload.

The performance difference of the basic TD3 agent and the robust TD3 agent in comparison to the NMPC method applied to a tunnel-based circuit is seen in Figure 8a. The trajectory of the robust TD3 agent is closer to that of the NMPC method. The improved performance is also evident when looking at the swing angles depicted in Figure 8b, which shows the robust TD3 agent reducing the maximum swing angle by 9.43 deg and closer resembling the NMPC behavior. The fact that the SITL Gazebo model does not result in delayed NMPC execution, as seen with the testing model, suggests that the system on which the NMPC is implemented can influence the quality of the generated input commands. The TD3 agent shows more consistent behavior for the different models as the agent produces input commands more deterministically that do not require an iterative solving process form each new state.

![Graph showing the comparison between basic and robust TD3 agents and the NMPC method in terms of position and angle.](image2)

**Fig. 8.** The circuit completion of the tunnel circuit in (a) and the resulting swing angle in (b) of the two TD3 agents and the NMPC method with a $m = 1$ kg and $l = 1$ m payload.

Figure 9 shows very good robustness to varying payload parameters for the robust TD3 agent, which aids in avoiding the tunnel edges for this kind of circuit. This indicates that training with changing payload parameters improves the agent's robustness and performance, and more comparable results to the NMPC method are achieved.
6 Conclusion

A model-free TD3 agent was trained as a local planner using a reduced system model to stabilize the flight of a RUAV and suspended payload system and a model-based NMPC was implemented as a comparison. The suitability of trajectory planning as a solution was proven on a four-waypoint testing circuit, where the TD3 agent produced comparable results to a model-based NMPC method with regards to achieving a significant swing reduction from the baseline. The ability of both local planning solutions to incorporate path limitations was further proven using a tunnel-based circuit.

By training the agent with changing payload parameters after each episode, which introduced a method of developing variable planning models, the TD3 agent's robustness to uncertain payload parameters was increased. The performance was also improved and a trajectory that better avoided the tunnel edges was achieved with reduced swing angles from the basic TD3 agent.

This paper not only provides evidence that a TD3 agent is a viable local planning solution to stabilize the flight of a RUAV with an uncertain suspended payload, with comparable results to the state-of-the-art NMPC solution, but also shows that a reduced model can be used to train an agent that can be transferred to two different much more realistic models.

We would like to acknowledge the Stellenbosch University School for Data Science and Computational Thinking for partially funding this research.

References


