Waldock, A., Greatwood, C., Richardson, T., & Salama, F. (2018). Learning to Perform a Perched Landing on the Ground Using Deep Reinforcement Learning. *Journal of Intelligent and Robotic Systems*, 92(3-4), 685-704. <https://doi.org/10.1007/s10846-017-0696-1>

Peer reviewed version

Link to published version (if available):
[10.1007/s10846-017-0696-1](https://doi.org/10.1007/s10846-017-0696-1)

[Link to publication record in Explore Bristol Research](#)
PDF-document

This is the accepted author manuscript (AAM). The final published version (version of record) is available online via Springer at <https://doi.org/10.1007/s10846-017-0696-1>. Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research

General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
<http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/>

Learning to perform a Perched Landing on the Ground using Deep Reinforcement Learning

Antony Waldock · Colin Greatwood ·
Francis Salama · Thomas Richardson

Received: date / Accepted: date

Abstract A UAV with a variable sweep wing has the potential to perform a perched landing on the ground by achieving high pitch rates to take advantage of dynamic stall. This study focuses on the generation and evaluation of a trajectory to perform a perched landing on the ground using a non-linear constraint optimiser (Interior Point OPTimizer) and a Deep Q-Network (DQN). The trajectory is generated using a numerical model that characterises the dynamics of a UAV with a variable sweep wing which was developed through wind tunnel testing. The trajectories generated by a DQN have been compared with those produced by non-linear constraint optimisation in simulation and flown on the UAV to evaluate performance. The results show that a DQN generates trajectories with a lower cost function and have the potential to generate trajectories from a range of starting conditions (on average generating a trajectory takes 174 milliseconds). The trajectories generated performed a rapid pitch up before the landing site is reached, to reduce the airspeed (on average less than 0.5m/s just above the landing site) without generating an increase in altitude, and then the nose dropped just before hitting the ground to allow the aircraft to be recovered without damaging the tail. The trajectories generated by a DQN produced a final airspeed (when it hit the ground)

Antony Waldock
BMT Defence Services
E-mail: awaldock@bmtdsl.co.uk

Colin Greatwood
University of Bristol
E-mail: Colin.Greatwood@bristol.ac.uk

Francis Salama
Blue Bear Systems
E-mail: francis.salama@bbsr.co.uk

Thomas Richardson
University of Bristol
E-mail: Thomas.Richardson@bristol.ac.uk

of 3.25m/s (with a standard deviation of 0.97m/s) in the downward direction, which would allow the aircraft to be safely recovered and significantly less than a traditional landing ($\sim 10m/s$).

Keywords Deep Q-Network · UAV · Perched Landing

Nomenclature

C_L	Coefficient of lift
C_{L_0}	Coefficient of nominal lift
C_{L_α}	Slope of $C_L - \alpha$ plot
C_{L_q}	Slope of $C_L - q$ plot
C_D	Coefficient of drag
C_{D_0}	Zero lift drag coefficient
F_L	Lift force on aircraft
F_D	Drag force on aircraft
ρ	Air density
V	Airspeed
S	Wing surface area
M_y	Pitching moment
η	Elevator angle
Λ	Wing sweep
$\dot{\eta}$	Rate of change of elevator angle
$\dot{\Lambda}$	Rate of change of wing sweep
C_m	Pitching moment coefficient
C_{m_0}	Pitching moment coefficient about aerodynamic centre of wing
C_{m_α}	Slope of $C_m - \alpha$ plot
C_{m_q}	Slope of $C_m - q$ plot
C_{m_η}	Slope of $C_m - \eta$ plot
C_{m_Γ}	Slope of $C_m - \Gamma$ plot
q	Pitch rate
c	Wing chord
x_e, y_e, z_e	Position in earth coordinates (z down)
ϕ_e, θ_e, ψ_e	Orientation in earth coordinates
U_b, V_b, W_b	Velocity in aircraft body axis
q_b, p_b, r_b	Rotational velocity in body axis
$\theta^{traj}(t)$	Desired pitch angle for computed trajectory at time t
$\theta^{err}(t)$	Error between desired and measured pitch angle at time t
$\eta^{traj}(t)$	Desired elevator angle for computed trajectory at time t
$\Delta\eta(\theta^{err})$	Change in elevator deflection due to pitch angle error
u	Control Actions (Wing sweep rate and Elevator angle rate)
$Q(x, u)$	State-action value function
$V(x)$	state value function
$R(x)$	Reward in state x
N	Number of time steps in the trajectory
γ	Discount rate

$Q(x, u)^*$ Optimal state-action value function that maximises the expected reward R

1 Introduction

Perching is a difficult problem for a fixed wing aircraft as it requires the aircraft to effectively stop flying when the landing location is reached with a sensible orientation (e.g. not upside down). A fixed wing UAV that is capable of sweeping the wings forwards and backwards is used here to perform perching manoeuvres taking inspiration from birds landing on flat ground. By sweeping the wings forwards the aircraft is capable of producing large pitching moments and achieving high pitch rates, which in turn enable the aircraft to decelerate through dynamic stall. A longitudinal model of the aircraft is identified from wind tunnel measurements and is used to compute optimal perching trajectories. The paper develops and compares two nonlinear optimisation processes for generating these trajectories and flight test results are presented.

The problem of performing a perched landing has been discussed by several others, but typically these have had the perch raised above the ground allowing the aircraft to “swoop” from below the perch to reduce airspeed. For example, Cory et al [5] showed how a small fixed wing glider could be controlled to land on a 2cm perch using optimal control (linear-quadratic regulator trees). In recent years, the same approach has been used by Moore et al [11] and has shown a success rate of 95 percent in flight experiments from a range of initial conditions. The aircraft was lightweight (85g) and started with an initial speed of between 6 and 8m/s at 3.5m from the perch.

Another example of perching has recently been demonstrated by Meckstroth et al [9]. A small UAV (200g) was shown to be able to perform a perched landing at higher speeds (12m/s) from a position 20m from the perch using elevon controls. Meckstroth et al built a numerical model of the aircraft using flight data from a range of trim conditions and then produced a Bezier Curve to represent the trajectory depending on the initial starting conditions (height and airspeed) [8]. The results showed that 12 out of 13 flights resulted in a successful perch where the one unsuccessful flight was due to state estimation errors in the VICON measurement system.

Previous research has shown the benefits of performing a perched landing but have been limited to situations where the aircraft is able to drop below the perch to reduce airspeed [12] and limited outdoor environments. The focus of this study is on the development of an aircraft and controller to perform a perched landing on the ground where dipping down below the perch is not possible i.e. landing on the ground. To be able to achieve this, the hypothesis is that the aircraft must take advantage of the properties of dynamic stall to reduce the airspeed to close to zero without gaining altitude. Performing a perched landing in this study is a two stage process where the aircraft is first controlled to hit an entry gate and then a perching trajectory is executed. The overall approach is inspired by Meckstroth et al [9] where a numerical model

of a fixed wing UAV with a sweep wing is first built and then a non-linear optimiser and machine learning algorithms are used to generate a perching landing trajectory. The trajectory (sequence of control inputs and desired aircraft states) is then combined with a closed loop controller to test the accuracy on the aircraft in series of flight trials.

This study uses two optimisation approaches. The first optimisation method was inspired by [5] where the perching problem is presented as a non-linear constraint optimisation. The second combines neural networks and Q-Learning [7] which has recently seen a resurgence due to the addition of deep learning methods to form Deep Q-Networks (DQN) [10]. This study first outlines the development of a numerical model for a sweeping wing UAV (based on a Bixler 2 airframe) followed by an introduction to machine learning and non-linear optimisation. The results section then presents a series of flight trials using different trajectories using both an open loop and closed loop controller.

2 Numerical Model

In order to generate a perching trajectory, it is necessary to identify a model of the UAV first. This section provides details on the numerical model developed. The UAV is based upon an off-the-shelf airframe (Bixler 2), but modified to have variable wing sweep and independent wing tip incidence control. The aircraft designed and built for evaluating these manoeuvres is shown in Figure 1. The wing sweep enables the aircraft to generate large pitching moments due to changing the centre of lift relative to the centre of mass. The sweeping of the wings is slow when compared to the elevator due to their inertia; the elevator is therefore used for disturbance rejection to keep the aircraft on the desired trajectory using a closed loop controller.

Fig. 1 Bixler 2 Airframe, fitted with variable sweep wing

The coefficient of lift (C_L) and drag (C_D) are defined by

$$C_L = C_{L_0} + C_{L_\alpha}\alpha + C_{L_q}q \quad (1)$$

$$C_D = C_{D_0} + C_{D_\alpha}\alpha \quad (2)$$

where C_{L_q} is the increase in lift due to pitch rate q , C_{L_α} is the increase in lift due to the angle of attack α , C_{L_0} is the nominal lift and C_{D_0} is the nominal drag. From these coefficients, the lift and drag forces may be evaluated as

$$F_L = \frac{1}{2}C_L\rho V^2 S \quad (3)$$

$$F_D = \frac{1}{2}C_D\rho V^2 S \quad (4)$$

Where ρ is the density, V is the airspeed and S is the surface area of the wing. The pitching moment (M_y) is a function of the elevator (η) and sweep (Λ) as well as the incidence and pitch rate. The pitching moment and pitching moment coefficient are defined as

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_\eta}\eta + C_{m_\Lambda}\Lambda + C_{m_q}q \quad (5)$$

$$M_y = \frac{1}{2}C_m\rho V^2 S c \quad (6)$$

In this study, the aircraft is not powered during the perching manoeuvre and hence thrust is not considered in this model. The forces and moments are combined with the mass and inertia of the aircraft to evaluate the body accelerations and rates. Euler Angles are then used to derive the velocity and position in body and earth coordinates. Wind tunnel data was used to identify all of the coefficients. Figure 2 shows the aircraft in the wind tunnel connected to the data acquisition software.

A nonlinear model that captures the dynamics over a wide range of states was desired in order to fully exploit the capabilities of the aircraft. It was therefore decided to identify the coefficients as a function of aircraft state. The identified coefficients are given in the appendix and are encoded within the model as piece-wise linear functions. The numerical model was implemented in Matlab and Java to allow the response to control surface deflections to be simulated. The numerical model was then used to generate a trajectory i.e. set of control deflections (wing sweep and elevator) that would allow the aircraft to perform a perched landing. The wing tips are currently used to stabilise roll during the perching manoeuvre using a PID controller but future work would incorporate these into the perching manoeuvre.

3 Generating a Perching Trajectory using Interior Point OPTimizer

A successful perched landing is defined in this paper as the arrival at the perch with a near-zero velocity (such that the aircraft is recoverable) and

Fig. 2 UAV being tested in the Wind Tunnel with data capture screen in the foreground

pitch angle (such that the tail does not hit the ground first). In this section, the method for generating a trajectory for perched landings uses a non-linear constraint optimisation approach similar to that defined in [11]. The vector used to represent the aircraft state is:

$$x = [x_e, y_e, z_e, \phi_e, \theta_e, \psi_e, U_b, V_b, W_b, q_b, p_b, r_b, \Lambda, \eta]$$

where $[x_e, y_e, z_e]$ is the aircraft position in earth coordinates (z down), $[\phi_e, \theta_e, \psi_e]$ is the aircraft orientation in earth coordinates, $[U_b, V_b, W_b]$ is the translational velocity in body axis, $[q_b, p_b, r_b]$ are the rotation velocity in body axis and Λ is the delta sweep angle from neutral (positive forwards) and η is the delta elevator angle from neutral (positive down). Roll control was performed using the wing twist using a standard PID controller and was independent of the trajectory generated.

The dynamics of the model are highly non-linear with large changes in the forces and moments during the perching manoeuvre. In this study, an optimal trajectory is generated using IPOPT (Interior Point OPTimizer) [16] that implements an interior-point line-search filter method [15], which is particularly suitable for large problems with up to millions of variables and constraints, assuming that the Jacobian matrix of constraint function is sparse, but also small and dense problems can be solved efficiently. The trajectories generated in this study used IPOPT configured to use the linear solver MA27 optimiser [2] for the constraint optimisation.

The optimisation treats the control inputs $u = [\dot{\Lambda}, \dot{\eta}]$ and the system states $[x_e, y_e, z_e, \dots, \Lambda, \eta]$ as variables in the optimisation routine where constraints can be placed on the states and the inputs. The optimisation starts at a fixed

starting state (defined in the results section) and then minimise the following objective function.

$$\begin{aligned} \min_{x,u,h} J &= \sum_{n=0}^{N-1} (x_n^T Q x_n + h u_n^T R u_n) + x_N^T Q_f x_N \\ \text{subject to} & \quad x_{n+1} = x_n + h f(x_n, u_n) \end{aligned} \quad (7)$$

Where N is the number of time steps (fixed at a maximum of 50 in this study), Q is the cost of the state, R is the cost on the control, h is the time step (0.1 seconds in this study) and Q_f represents the cost on the final time step. As per [11], Q was zero and Q_f was used to describe if the aircraft had successfully performed the perched landing. Different cost functions are evaluated in the results section and are shown in Figure 4. The resultant trajectory is a sequence of control inputs (elevator and wing sweep rates) that achieve a perched landing (e.g. reduce the cost function) with the corresponding predicted aircraft states. The challenge with this approach is that it relies on a fixed set of starting conditions. This assumption could be overcome using a library of trajectories (as suggested by [5]) but this could potentially require generating and storing a large set depending on the variation in starting conditions. In this paper, the authors frame the problem as a Reinforcement Learning (RL) problem in order to learn a model that can be queried from a wide range of starting conditions.

4 Perching as a Reinforcement Learning Problem

In this section, the generation of a perching trajectory is posed as a Reinforcement Learning (RL) problem where the reward or penalty is only known when the perch is reached or not. The goal is to learn a model that can be queried from any state such that a trajectory can be generated from a range of starting conditions. The aircraft starts in a known (but not fixed) state x_t and the controller performs the control inputs u_t at which point the aircraft transitions to the next state x_{t+1} . At the end of the flight (or episode in reinforcement learning terms), the aircraft receives a numerical reward r_t (in this paper between -1 and 1) based on the desired goal.

The aim in reinforcement learning is to learn a control policy that maximises the *expected long-term reward* by performing trial and error actions. For a continuing task, such as control, the expected return R_t at time t can be written as the discounted sum of the immediate rewards r_t over an infinite number of time steps:

$$\begin{aligned} R_t &= \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \\ \forall \gamma &\in [0, 1] \end{aligned} \quad (8)$$

where γ is the discount rate and determines the importance of future rewards. When γ is set to zero, the controller is concerned with only maximising

the immediate reward i.e. the next action. As γ is increased, a higher weighting is placed on the future rewards until $\gamma=1$ when all rewards are equally valued.

The majority of reinforcement learning algorithms use a dynamic programming [3] approach to approximate the *expected value* of each state-action pair. The expected value, $Q(x_t, u_t)$ of a state-action pair is the sum of the immediate reward $r_t(x_t, u_t, x_{t+1})$ after taking control action u_t in state x_t to transition to x_{t+1} plus the expected return r_t if the policy described by the optimal state-action values (argmax) is followed from then on:

$$\begin{aligned} Q(x_t, u_t) = & E[r_t(x_t, u_t, x_{t+1}) \\ & + \gamma r_{t+1}(x_{t+1}, \underset{u_{t+1}}{\operatorname{argmax}} Q(x_{t+1}, u_{t+1}), x_{t+2}) \\ & + \gamma^2 r_{t+2}(x_{t+2}, \underset{u_{t+2}}{\operatorname{argmax}} Q(x_{t+2}, u_{t+2}), x_{t+3}) \dots] \end{aligned} \quad (9)$$

where E denotes the expected value of $Q(x_t, u_t)$ if the policy determined by $\operatorname{argmax}_u Q(x, u)$ is followed. A control policy that maximises the expected action-value function Q^* can be found by rearranging Equation 9 to satisfy the *Bellman optimality equation* [4]:

$$Q^*(x_t, u_t) = E[r_t(x_t, u_t, x_{t+1}) + \gamma \max_{u_{t+1}} Q^*(x_{t+1}, u_{t+1})] \quad (10)$$

where $\max_{u_{t+1}} (Q^*(x_{t+1}, u_{t+1}))$ is value $V(x_{t+1})$ of the optional action in state x_{t+1} . Assuming the policy followed is determined by $\operatorname{argmax}_u Q^*(x, u)$, the optimal policy is guaranteed. Q -Learning [17] provides a method of computing the Q^* while interacting with the environment. On every time step t , the value of each state-action pair $Q(x_t, u_t)$ is updated based on experiences in the form of $(x_t, u_t, r_{t+1}, x_{t+1})$ using:

$$Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \alpha [r_{t+1} + \gamma \max_{u'} Q(x_{t+1}, u') - Q(x_t, u_t)] \quad (11)$$

Where α is referred to as the learning rate and is used to determine the rate at which the state-action values are updated. Q -Learning does not explicitly represent the control policy but instead uses the values of all actions in all states to derive an appropriate policy. The state-action value function, Q , directly approximates Q^* , the optimal state-action value function, independent of the control policy followed.

In large or continuous control problems, such as those found in flight control, storing the state-action values perfectly in a tabular representation becomes unmanageable (in this problem our state vector has 14 input dimensions). The memory needed to store the large tables is not the only issue but also the time and training data needed to fill them accurately. In real world control problems, it is typically not possible to exactly store the value function perfectly so instead it must be approximated using function approximation techniques. Function approximation techniques attempt to model an entire function, in Q -Learning this is the state-action value function, by generalising from previously seen state-action values to state-action values that

have not yet been experienced. When the state or action spaces include continuous variables, exactly the same state-action value is extremely unlikely to occur twice and generalisation becomes essential.

In fact, Barto and Sutton argue that the only way to learn anything at all on these tasks is to generalise from previously experienced states to ones that have never been seen [14]. Finding an approximation of the state-action values is the key to Q -Learning for control problems with continuous states and actions. For example, to learn a mapping from aircraft states to control actions u requires learning a high-dimensional model where the exact aircraft state and response is unlikely to have been encountered before.

5 Deep Q-Network

In recent years, advances in deep q-learning have shown promise to model the Q function for high-dimensional problems. For example, Mnih et al [10] showed how a deep q-network (DQN) could be used to play different Atari games using the screen image as the input and the normalised score as the reward function. Deep neural networks are capable of modelling high-dimensional functions such that reinforcement learning algorithms can perform effectively [7]. However, previous work in combining simple online RL algorithms with deep neural networks produced instabilities i.e. the network would overfit in some areas and diverge in others. A variety of solutions have been proposed to stabilise the algorithm which share a common concept: the sequence of observed data encountered by an online RL agent is non-stationary, and online RL updates are strongly correlated. By storing the agents data in an experience replay memory, the data can be randomly sampled [10] from different time-steps. Aggregating over memory in this way reduces non-stationary and reduces the correlations in the updates, but at the same time limits the methods to off-policy reinforcement learning algorithms. Deep RL algorithms based on experience replay have achieved unprecedented success in challenging domains such as Atari 2600 but requires significant amounts of memory and training times. However for this perching problem, the training phase is based on a numerical model of the aircraft which can be processed off-line (i.e. away from the aircraft) using multiple CPUs or GPUs. In a similar approach to [10], the perching of a fixed wing aircraft is seen by the Deep Q-Network as a game where the score relates to the final state of the aircraft.

6 Deep Q-Network applied to Perched Landing

This section outlines the application of a deep q-network to the generation of a trajectory to perform a perched landing on a variable sweep wing UAV. The actions (u_t) are defined as the control inputs $u = [\dot{A}, \dot{\eta}]$ where in this paper the control inputs u are the rate of change of the variable sweep of the wing and of the elevator. A deep Q-Network is used to model the Q function and hence

define the policy for the controller in any given state. In this experimentation, the neural network model uses an input layer of 14 nodes (the size of the aircraft state) with three fully connected hidden layers each with 512 nodes which are each followed by a rectified linear layer. The output layer is then added using a full connected layer from 512 hidden nodes to the 49 outputs nodes that represent the Q values for the different combinations of possible output values. The number of nodes and layers was determined experimentally but the remainder of the parameters used are the same as reported in [10].

The experience memory contains 1 million experiences in the form $(x_t$ and u_t to x_{t+1} with r_t) and the exploitation rate is decayed from 1 to 0.1 between 10000 to 1 million time steps. The exploitation rate controls the probability of selecting a random action or following the learnt policy ($Q(x, u)$). Therefore at the beginning of the experiment, the optimisation is purely following random actions and exploring the reward received and then as the experiment continues starts to follow the optimal policy found. The discount rate is 0.99 and the learning rate is 0.00025. The experiment was run over a total of 5 million steps and the performance of the controller evaluated every 5000 steps. The network built can be queried to extract a trajectory consisting of control inputs and predicted aircraft states from the particular starting condition.

7 Approach Controller

The perch is performed in two stages as inspired by hand glider landings: approach and then perching. An approach controller has been constructed to drive the aircraft to the required height and airspeed at the entry gate position at which point the perching trajectory generated can be followed. The role of the approach controller is therefore to take the UAV from a position lined up with the landing site - from a few tens of metres up and of the order of 100m away - down to the entry gate. The final speed and position of the UAV at the end of the approach is critical as it will directly contribute to the ability for the aircraft to accurately follow the perch trajectory.

The experiments were conducted using an open source flight controller called ArduPlane [1]. In this work, the ArduPlane software is modified to use a single input/single output controller to control the airspeed during this approach phase - based upon the technique described in Example 4.6-4 of Ref. [13]. Airspeed is measured from the pitot tube and the throttle is adjusted from its cruise set-point via a proportional feedback term. A separate controller pitches the aircraft by sending pitch demands through the autopilot's attitude controller to drive it back onto the commanded glide-slope. The geometry of the glide-slope is shown in Figure 3, where the current aircraft position $P(t_1)$ is compared against a point projected along the current glide-slope at some point in the future $P(t_2)$. The perpendicular distance between the desired glide-slope and $P(t_2)$ is taken as the distance error d . A Proportional Derivative (PD) controller is used to drive the error d to zero and hence guide the aircraft along the glide-slope. Whilst it is difficult to measure d directly, $\dot{d}$ can be computed

using the aircraft velocity and flight path angle error. This derivative is then integrated in order to determine d . Further details of the glide slope geometry are given in Ref. [13].

Fig. 3 Glide-slope geometry, schematic from Example 4.6-4 of Ref. [13]

8 Closed Loop Trajectory Controller

Once the aircraft passes through the entry gate, it switches from the approach control law into following the control input sequence previously computed by IPOPT or DQN. Flying such a dynamic manoeuvre as perching using only a predefined control input sequence does not take into account external disturbances (*e.g.* gusts) which will cause the expected and actual responses to diverge. A closed loop controller was therefore wrapped around the open loop elevator control sequence. The closed loop controller compared the difference between the expected pitch angle $\theta^{traj}(t)$ and the measured pitch angle θ , finding the pitch angle error

$$\theta^{err} = \theta^{traj}(t) - \theta \quad (12)$$

The autopilot's attitude controller (of standard linear PID construction) is used to find the required change in elevator deflection to correct the pitch attitude. This change in elevator deflection calculated by the attitude controller $\Delta\eta(\theta^{err})$, as a function of the pitch angle error, is added to the elevator angle computed by IPOPT or DQN. Hence, if the optimised control input at time t is defined as $\eta^{traj}(t)$ then the elevator input demanded from the control surface becomes a function of time t and measured pitch angle θ ,

$$\eta(t, \theta) = \eta^{traj}(t) + \Delta\eta(\theta^{err}) \quad (13)$$

9 Results

This section presents experimental results on performing a perched landing on a UAV with a sweeping wing in an outdoor environment following a trajectory

built using IPOPT and DQN. Before conducting outdoor trials, a number of different cost functions were evaluated in simulation to quantitatively determine which parameters produced trajectories that appeared a feasible perched landing.

9.1 Trajectories Generated using IPOPT

This section presents an evaluation of different cost functions using IPOPT. The aim was to evaluate which cost functions produced a feasible perched landing in terms of reaching the perch with a minimal velocity and with an appropriate orientation. Using a series of exploratory flights (under manual control), the aircraft was capable of performing a perch type manoeuvre when approximately 40m away from the perch at a height of 2m with an airspeed of 13m/s. Therefore it is assumed that the perching manoeuvre will be initiated at this location and hence, a trajectory will be generated with these starting conditions $x_{t0} = [-40, 0, -2, 0, 0, 0, 13, 0, 0, 0, 0, 0, 0]$. The perch is assumed to be located at $[0, 0, 0]$ with the trajectory concluding when the aircraft height goes negative (where height is defined as $-z_e$) or the maximum time limit is reached (5 seconds). Figure 4 shows three trajectories generated using IPOPT with different cost functions. The first cost function (solid line) was inspired by the work of Cory [5] and Moore [11] and only considered position and velocity with a 1:10 ratio and no weighting on the control inputs. The trajectory generated lands 62cm away from the perch with an overall airspeed of 5.83m/s (with a vertical velocity component of 5.39m/s) but the control inputs contained rapid movements which is not ideal for a real aircraft. For the second cost function (dashed line), the control inputs were weighted the same as the velocity to reduce the rapid changes. The trajectory generated was similar in profile to the first but landed further away from the perch at 1.37m but with a similar final overall airspeed (5.96m/s). Both of these trajectories had a positive pitch angle when landing (19.8 and 16.9 degrees nose up correspondingly) which would mean that the tail was likely to hit the ground first.

Therefore, a third cost function (dotted line) was evaluated where the pitch of the aircraft was penalised the same as the airspeed and the control inputs. The trajectory generated landed 6cm away from the perch with an overall airspeed of 6.21m/s (with a vertical speed of 5.07 m/s) with a pitch angle of only 10.3 degrees. In particular, the perching manoeuvre results in a very low forward speed and essentially drops on to the perch from approximately a metre above the ground. This cost function was used in the outdoor aircraft trials as it produced a trajectory that appeared a feasible perched landing.

9.2 Flight Tests using IPOPT

This section presents the results of following the trajectory generated by IPOPT on a UAV with a variable sweep wing. The experiments were con-

Fig. 4 Trajectories generated by IPOPT using the simulation of the aircraft dynamics derived from the wind tunnel data collection with three different cost functions 1 - Solid) airspeed is the most important followed by position $Q_f = \text{diag}[1, 0, 1, 0, 0, 0, 10, 0, 10, 0, 0, 0, 0, 0]$ and $R = \text{diag}[0, 0]$, 2 - Dash) airspeed and smooth actions are the most important followed by position $Q_f = \text{diag}[1, 0, 1, 0, 0, 0, 10, 0, 10, 0, 0, 0, 0, 0]$ and $R = \text{diag}[10, 10]$ and 3 - Dotted) airspeed, smooth actions and a small pitch angle are the most important followed by position $Q_f = \text{diag}[1, 0, 1, 0, 10, 0, 10, 0, 10, 0, 0, 0, 0, 0]$ and $R = \text{diag}[10, 10]$

ducted using the entry conditions of 13m/s and 60m above the ground (although the graphs have been scaled to fit with the trajectory which started at 2m above the ground). The trials were conducted at altitude so the trajectory could be performed a number of times to gain an understanding of repeatability without having to relaunch the aircraft every time. Figure 5 shows the aircraft states for three open and two closed loop flights compared with the numerical model. The results show that the closed loop flights follow the optimised trajectory more closely (in terms of pitch and pitch rate) and achieve the large final pitch angles required, the latter having the effect of reducing the airspeed to approximately 2m/s at the landing point. The pitch angles and rates achieved can be seen in Table 1. The trajectory generated by IPOPT aimed to reduce the airspeed initially by pitching up (at around 0.5 seconds) and then maintain the airspeed at around 10m/s (just above the stall speed of this aircraft) until the final pitch up was performed. Figure 5 shows that the controller was unable to maintain the airspeed above 10m/s throughout all this time (the airspeed drops below after 3 seconds) and therefore the aircraft stalled early (this was also visually observed during the trials) and fell short of the landing site. The stalling of the aircraft is likely due to inaccuracies in

Fig. 5 A comparison of the trajectory generated by the IPOPT (dotted line) with the trajectory flown using an open loop (dashed) and closed loop controller (solid). The figure presents the pitch angle, pitch rate, sweep and elevator angle (sweep and elevator angle are the same for the IPOPT and open loop trajectory) and the height and airspeed of the aircraft. The height has been scaled to show the trajectory when initiated 2m above the ground - the aircraft would therefore land at a 0m height.

Parameter	Open Loop			Closed Loop	
Initial Airspeed (m/s)	13.3	12.8	12.7	12.4	12.5
Minimum Airspeed (m/s)	2.0	-0.56	0.42	-0.77	-0.38
Max Pitch Angle (deg)	14.32	39.53	12.6	32.65	34.95
Max Pitch Rate (deg/s)	78.31	106.05	89.24	142.40	161.91

Table 1 The table presents a summary of the key flight parameters shown in Figure 5. Note: The airspeed is never negative in reality but reports a negative value when close to zero due to calibration errors

the numerical model or small disturbances during flight. If the aircraft stalls early, there is no means to reach the perch because the aircraft was gliding. To reduce the likelihood of the aircraft stalling early, the trajectories were regenerated with a small amount of noise added to the numerical model to prevent the trajectory getting too close to the stall point.

9.3 Flight Tests using IPOPT with noise

To prevent the optimisation from generating a trajectory that could stall early a small amount of noise has been added to accelerations calculated by the nu-

merical model. To evaluate the impact on the trajectory of adding noise to the numerical model, a set of trajectories were generated for different noise levels (0, 0.3, 0.5, 1.0 m/s²). Figure 6 shows trajectories for the different amounts of noise added. As noise is added to the numerical model, the airspeed on the approach to the perch is increased away from the stall point until a constant decrease in airspeed is achieved with a noise level of 1.0m/s².

Fig. 6 Comparison of the different trajectories generated by IPOPT when Gaussian noise is added to the acceleration term in the wind tunnel model. solid line = no noise (as before), dashed line = 0.3m/s², dotted line = 0.5m/s² and long dashed line = 1.0m/s².

The trajectory generated with a small amount of noise (0.3m/s²) was flown on the UAV to validate that the aircraft was able to follow the trajectory. Figure 7 presents the aircraft states for two open loop and five closed loop flights compared with the numerical model. The results show that the controller is able to closely follow the trajectory compared to the open loop experiments in terms of pitch angle and pitch rate. The airspeed is maintained above the stall point except at the perch when the airspeed is reduced to a mean of 4.59m/s. The open loop controller typically stalls earlier (at ~ 3.5 seconds) than the closed loop controller resulting in falling short of the perch.

Figure 7 also shows the altitude (height above the ground) of the aircraft during the perch. The height follows a similar profile to the numerical model but the aircraft loses more height than expected at the beginning. This extra reduction in height is due to the entry conditions of the aircraft because the initial descent for the trajectory is assumed to be zero (in the optimisation)

Fig. 7 Comparison of the trajectory generated by IPOPT with 0.3 m/s^2 (dotted line) of added noise with open loop (dashed) and closed loop controller flights (solid). The figure presents the pitch angle, pitch rate, sweep and elevator angle (sweep and elevator angle are the same for the IPOPT and open loop trajectory) and the height and airspeed of the aircraft. The height has been scaled to show the trajectory when initiated 2m above the ground - the aircraft would therefore land at a 0m height.

where in the flights conducted the aircraft initial descent is around 2m/s . To evaluate the overall performance of the trajectory, the airspeed, pitch angle and position of the UAV are shown for nine closed loop perching manoeuvres in Figure 8.

The results in Figure 8 show that the airspeed is being maintained above the stall speed ($\sim 10\text{m/s}$) during the initial part of the manoeuvre and then quickly reduced to around 6m/s towards the end when the pitch angle reaches nearly 30° . The aircraft ends the manoeuvre with a mean airspeed of 4.82m/s (standard deviation of 0.41m/s) and a mean pitch angle of -8.5° (standard deviation 4.01°). The graph on the right shows a top down view of the trajectory of the aircraft during the perching manoeuvre.

The results show that the aircraft lands (the dots) with a mean spread of 2.2m with a standard deviation of 1.98m and the maximum distance from the centre of the points is 6.5m . Considering this is in an outdoor environment with highly variable conditions and the trajectory was primarily optimised for airspeed and pitch angle on landing this is an extremely encouraging result. A sequence of images showing the manoeuvre is shown in Figure 9 for three different runs.

Fig. 8 Comparison of nine flights using the IPOPT trajectory generated with $0.3m/s^2$ noise with the closed loop controller on the elevator only for stabilisation. The figure shows the airspeed, pitch, top down position (x,y) and roll angle during each of the manoeuvres. The aircraft ends the manoeuvre with a mean airspeed of $4.82m/s$ (standard deviation of $0.41m/s$) and a mean pitch angle of -8.5° (standard deviation 4.01°) with the position clustered with a mean spread of $2.2m$.

The spread of results is probably due to the variation in the initial conditions and a preference to roll to the right at the start of the pitch up manoeuvre, which can be seen at the end of the top down plot (Fig. 8). The right hand wing had a tendency to stall just before the left hand wing. The mechanism was installed in the wings by hand without a jig and so the incidence of the right hand wing is slightly greater than the left. The effects of the asymmetry could be reduced by increasing the gains on the roll controller during this final section of the flight or by changing the incidence of the wing tip.

This following sections present results of using a Deep Q-Network to generate perching trajectories for a UAV with a sweeping wing and elevator from a range of starting conditions. One of the advantages of this approach is that a trajectory can be built as the aircraft exits the approach and hence can take into account the deviation between the desired and actual state as it passes through the entry gate. The flight trials in this section have shown that the aircraft was descending as it reached the entry gate but further analysis of the aircraft state as it reached the entry gate are presented in the next section.

9.4 Reaching the Entry Gate

This section presents a brief analysis of the approach controller in terms of the key parameters required to perform a perch manoeuvre (height, airspeed, pitch angle, pitch rate). In particular, the results show the variability of the entry conditions to the trajectory and highlight the likely need for trajectories from different starting conditions. The aircraft was flown in a circuit at

(a) Flight 1

(b) Flight 2

(c) Flight 3

Fig. 9 Sequence of images showing a perched landing at altitude using a trajectory generated by the non-linear constraint optimiser (IPOPT). The aircraft is flying from right to left and the sequence of images go from left to right across the rows. The images are 0.9 seconds apart.

approximately 65m above the ground with a cruising speed of 15m/s. The approach controller was then initialised with a target height of 50m and airspeed of 13m/s to be achieved within 100m.

Fig. 10 Evaluation of the height, airspeed, pitch angle, pitch rate, descent and throttle during the approach. The three right most plots show error in the height vs airspeed, pitch vs pitch angle and descent vs throttle with a target height of 50m and airspeed of 13m/s.

The results of the entry gate experiment are shown in Figure 10 where the change in the height, airspeed, pitch angle, pitch rate, descent and throttle are shown during the approach with the final results shown as an error on the right-hand side. The error plots show that the approach controller achieves a height error between -1m and 2m with an airspeed error of 0m/s and 2m/s. The error in airspeed was most likely due to the descent, as it was noticed that the throttle was cut at the beginning of the manoeuvre to reduce height rapidly. Although the final airspeed could be improved using a different (e.g. TECS [6]) or more aggressive approach controller, this estimate of the uncertainty in entering the gate will be used to validate the DQN model when generating multiple trajectories from a range of starting conditions.

9.5 Trajectories Generated using Deep Q-Network

This section presents the trajectories generated using a Deep Q-Network for a fixed starting condition. Initially, it is assumed that the perching manoeuvre

will be initiated at a fixed location (as per IPOPT) and hence, a trajectory will be generated with these starting conditions:

$$[-40, 0, -2, 0, 0, 0, 13, 0, 0, 0, 0, 0, 0]$$

but this is relaxed later in this section where the initial airspeed of the aircraft varying based on the variance seen in test flights.

The perch is assumed to be at $[0, 0, 0]$ with the simulation concluding when either the aircraft height goes negative ($-z_e$ is positive), x_e is out of bounds ($>50\text{m}$ or $<-10\text{m}$) or the pitch angle ϕ exceeds $\pi/2$ (upright). The reward function $r(x_t)$ at the terminal step is given by negating and normalising the cost function $x_N^T Q_f x_N$ (as define in Equation 7 where Q_f is the cost associated with the final state) between -1 and $+1$ or returning -1 when ϕ exceeds $\pi/2$. The cost function is the same as IPOPT $([1, 0, 1, 0, 10, 0, 10, 0, 10, 0, 0, 0, 0])$. For all non-terminal states, the reward function is defined as $u_n^T R u_n$ where u_n are the control inputs and R is $1e - 6$ (as defined in Equation 7).

In this study, the continuous control inputs are divided into 7 discrete values for both elevator and wing sweep over the range from -60 degrees/second to 60 degrees/second (which is a sensible range for the servos in the aircraft). Therefore, at any time step the controller can select one of the 49 different control actions which represent two control inputs (deflection of the elevator and sweep angle). The resultant trajectories are compared with the those generated by a non-linear constraint optimiser (IPOPT) in the previous section where the noise added to the numerical model is 0.3m/s^2 .

The two methods (DQN and IPOPT) were compared using the same cost function to understand the two trajectories in Figure 11. The IPOPT (dotted) trajectory scored 547.2 while the DQN (solid) trajectory with the same cost function ($w_\theta = 10$) scored 340.9. The major difference between the two trajectories is the final airspeed achieved. The DQN trajectory manages to reduce the airspeed to 4.92m/s while IPOPT achieved 6.33m/s . The difference in airspeed appears to be due to the pitch angle achieved; DQN achieves a pitch angle of > 60 degrees just before landing while IPOPT has a pitch angle of ~ 30 degrees. A DQN generated the trajectory in on average 174 milliseconds (with a standard deviation of 23 miliseconds) on a virtual machine with 8 Intel CPUs.

Both methods used the same cost function (in terms of x, z, dx, dz and θ). The trajectory generated by DQN with the same cost function as IPOPT had a final pitch angle of ~ 60 degrees which would be undesirable for this type of aircraft (i.e. the tail would hit the ground first). Hence, the cost function was altered to weight the pitch angle by an extra factor of 10 ($w_\theta = 100$). The new DQN trajectory is shown in Figure 11 where the final pitch angle is reduced to 22 degrees with an airspeed of 5m/s with a score of 471.8 (using the same cost function as previously used for comparison with IPOPT).

The Q function learnt by DQN is shown in Figure 12 during the trajectory for the different control actions. The black squares indicate the worst combination of actions to take and white indicates the best. At the beginning of the flight, the extreme actions are highly undesirable but between 1.5 and

Fig. 11 Comparison of a trajectory generated using IPOPT (dotted line) with Deep Q-Networks using two different cost functions. The cost function is the same as IPOPT $([1, 0, 1, 0, w_\theta, 0, 10, 0, 0, 0, 0, 0])$ apart from the pitch angle where w_θ is either 10 (solid line) or 100 (dashed line).

3 seconds the desire to perform a pitch up can be seen by the white in the bottom left hand corner of the plots. Once the Q function was learnt, a new trajectory for a different starting condition can be generated on average in 174 milliseconds.

9.6 Flight Tests using DQN

Two trajectories generated by DQN (where w_θ in the cost function is 10 and 100) were flown on the aircraft to evaluate performance in real-world conditions. The aircraft was flown at a 65m altitude in a pre-defined circuit. The aircraft then performed an approach manoeuvre and then followed the trajectory generated by DQN. In the trial, the trajectory had been generated prior to the flight using a fixed starting condition but in the future the trajectory on-board would be generated using the measured starting conditions as it passed through the entry gate. The experiment was repeated five times, three times using a closed loop controller (i.e. follow the control inputs but adjusted based on desired pitch angle) and three using an open loop controller (i.e. follow the control inputs only).

Figure 13(a) and 13(b) show the aircraft pitch angle, pitch rate, sweep deflection angle, elevator deflection angle, height (the 48m is subtracted from

Fig. 12 The Q function learnt by Deep Q-Network for the trajectory shown in Figure 11 with the elevator deflection rate on the x-axis (-60,-10,-5,0,5,10,60 degrees per second) and sweep deflection rate on the y-axis (-60,-10,-5,0,5,10,60 degrees per second) with black as the lowest Q value and white as the highest Q value.

the entry gate height of 50m to get the same starting height as the trajectory (2m)) and airspeed for DQN with the two different cost functions ($w_\theta=10$ and $w_\theta=100$ correspondingly). In Figure 13(a), the controller is unable to maintain the desired pitch angle for any length of time and the aircraft is seen to roll off to one side as one wing stalls before the other. The aircraft is shown in the image sequence in Figure 14 performing the perched landing at altitude.

The remainder of this section focuses on the cost function where $w_\theta = 100$ as expected this produced more feasible perched landings because the pitch angle is lower. The results show that the aircraft can achieve pitch angles in excess of 33 degrees which led to a significant reduction in airspeed (mean 3.25m/s and standard deviation 0.97m/s). The airspeed achieved is 1.57m/s less than flights conducted with a non-linear optimiser (IPOPT was mean 4.82m/s and standard deviation of 0.41m/s). In fact, the airspeed is actually close to zero when the aircraft is at the maximum pitch angle achieved (see table 2) but then increases again as the nose drops due to the requirement to have the fuselage level. The airspeed measurements at low speeds (below 2m/s) are extremely uncertain and are dominated by noise. With the closed loop controller, the control inputs were followed without adaption for approximately

(a) $w_\theta = 10$ (b) $w_\theta = 100$

Fig. 13 Results of the flight tests for a trajectory generated by Deep Q-Networks (dotted line) with cost function $[1, 0, 1, 0, w_\theta, 0, 10, 0, 10, 0, 0, 0, 0]$ and the flights with an open loop controller (dashed) and closed loop controller on the elevator (solid lines). The flights were conducted at 50m altitude and the height adjusted to appear at ground level.

Fig. 14 A sequence of images showing the DQN trajectory flown on the modified Bixler 2 aircraft. The sequence starts in the top right and goes from right to left. The images sections are 1 second apart and were taken from Figure 13(a).

2 seconds when the elevator increased in an attempt to reach the high pitch angles demanded before reducing slightly to allow the nose to drop at the end.

Parameter	Open Loop			Closed Loop		
Initial Airspeed (m/s)	12.88	13.07	12.78	11.95	12.93	12.37
Minimum Airspeed (m/s)	3.41	3.44	-1.41	-1.23	-0.25	2.87
Max Pitch Angle (degs)	28.65	22.92	41.83	36.10	33.23	29.79
Max Pitch Rate (degs/s)	110.09	95.53	106.88	105.25	90.70	97.96

Table 2 The table presents a summary of the key flight parameters shown in Figure 13(b). Note: The airspeed is never negative in reality but reports a negative value when close to zero due to calibration errors

In future work, the controller could be used on both the elevator and sweep deflection to archive the demanded pitch angles. The height profiles in the bottom left graph appear to have the same profile as the numerical model although the approach controller was consistently under the desired height and hence the trajectories were under the desired height. The closed loop controller gains more height than the open loop controller due to the influence of the elevator to achieve the higher pitch angles. Future work will look at how the closed loop controller could be extended to control the sweep wing in tandem with the elevator.

In terms of the cost function, Figure 15 show the airspeed, pitch angle and position of the aircraft from a top down perspective for the cost function where $w_\theta = 100$. The DQN trajectory has achieved a minimal final airspeed and pitch angle (after the flare) but the top-down position of the aircraft shows a significant roll to the right during the pitch up manoeuvre. The roll is likely to be due to small changes in the aircraft construction, as previously mentioned, which is more pronounced at high pitch angles as the roll controller was still functioning throughout the flight.

Fig. 15 Comparison of nine flights using the Deep Q-Network trajectory generated with $0.3m/s^2$ noise with the closed loop controller on the elevator only for stabilisation with cost function $w_\theta=100$. The figure shows the airspeed, pitch and the position during each of the manoeuvres. The aircraft ends the manoeuvre with a mean airspeed of $3.25m/s$ (standard deviation of $0.97m/s$).

9.7 DQN with different starting conditions

To evaluate the impact of variable starting conditions, the DQN algorithm was trained with variable starting conditions (the initial airspeed was varied between $13m/s$ and $15m/s$) as determined by the entry gate analysis. A set of trajectories for different airspeeds were then generated and compared with the output of DQN with a fixed starting condition and a non-linear optimiser (IPOPT).

The left hand side of Figure 16 shows the position, airspeed and pitch angle for the trajectories generated by DQN with different starting conditions (from $13m/s$ to $15m/s$ in $0.1m/s$ intervals) when trained with a fixed starting condition (solid line) and variable starting conditions (dotted line). The graphs on the right show the output from IPOPT when the fixed trajectory is followed with different initial airspeeds.

For IPOPT, as the airspeed is increased from $13m/s$ to $15m/s$, the final airspeed starts to increase because the aircraft pitches up more with the increased speed. The DQN trajectories show a consistent final airspeed of below $6m/s$ although the position is variable at around $-5m$ for the variable and between $\pm 10m$ for the fixed. A significant feature of the DQN is that, after training, it has the ability to generate trajectories in milliseconds. These preliminary results therefore indicate that DQN would be able to generate suitable trajectories for variable starting conditions in less than a second while onboard the aircraft.

10 Conclusion and Future Work

This study has presented a numerical model for a micro UAV with a wing that is capable of sweeping forwards and backwards. Additionally, it has been

Fig. 16 Trajectories generated in simulation by DQN (left hand side) and IPOPT (right hand side) with variable starting conditions (airspeed of 13m/s to 15m/s) when DQN is trained with fixed (solid line) and variable starting conditions (dot and dashed).

shown that the numerical model can be integrated into a non-linear constraint optimiser and deep reinforcement learning to generate suitable trajectories to perform a perched landing. The trajectories generated have been flown at altitude with the UAV to evaluate the repeatability.

The perched landing manoeuvre generated by IPOPT has a low final forward speed and a level fuselage such that the aircraft can be safely recovered without a specially designed landing mechanism. The resultant trajectory produced by IPOPT consistently produced a final airspeed of 4.82m/s (standard deviation 0.41m/s) with the majority in the vertical direction as it drops on to the landing site. The aircraft finished with a slightly negative pitch angle (-8.5°) and a mean spread in the final position of 2.2m (standard deviation of 1.98m).

This study then showed how a Deep Q-Network could be used to generate a perched landing trajectory for a UAV with sweep wing. For this particular problem, the trajectories generated by the DQN algorithm have a lower cost function than those generated by a non-linear constraint optimiser (IPOPT) and have been shown in real-world flight trials to achieve a lower airspeed (mean of 3.25m/s with a standard deviation of 0.97m/s which is 1.57m/s less than IPOPT) during the perched landing by gaining high pitch angles and rates. Finally, the DQN algorithm was shown to be resilient to variable starting conditions even when not explicitly trained.

This study has shown that a UAV with variable sweep wing is capable of repeatedly performing a perched landing manoeuvre that results in small landing velocities. The perched landing manoeuvre in this study was optimised for an aircraft landing on the ground with a small pitch angle (to prevent damage to the tail and to land on the main body of the vehicle) but other perched landing trajectories could be considered using this approach. In future work, the focus will be on generating a selection of trajectories from a range of starting conditions to cope with those small deviations at the end of the approach.

Acknowledgements This research has been funded under the Autonomous System Underpinning Research programme from DSTL. The authors are grateful to Alan Davies from BAE Systems for helpful discussions during the project.

References

1. Arduplane autopilot software. <https://github.com/diydrones/ardupilot>.
2. Hsl. a collection of fortran codes for large scale scientific computation. <http://www.hsl.rl.ac.uk/ipopt/>. Accessed: 2016-06-20.
3. R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.
4. R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.
5. Rick E Cory. *Supermaneuverable perching*. Massachusetts Institute of Technology, 2010.
6. LF Faleiro and AA Lambregts. Analysis and tuning of a ‘total energy control system’ control law using eigenstructure assignment. *Aerospace science and technology*, 3(3):127–140, 1999.
7. Chris Gaskett. *Q-Learning for robot control*. Australian National University, 2002. This thesis is openly accessible from the link to the Australian National University’s institutional repository above.
8. Christopher Meckstroth and Gregory W. Reich. Near-optimal perching trajectory selection using bézier curve interpolation. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 5107. American Institute of Aeronautics and Astronautics, 8 2013.
9. Christopher M. Meckstroth and Gregory W. Reich. Aerodynamic modeling of small uav for perching experiments. In *31st AIAA Applied Aerodynamics Conference*, page 3191. American Institute of Aeronautics and Astronautics, 5 2013.
10. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
11. Joseph Moore, Rick Cory, and Russ Tedrake. Robust post-stall perching with a simple fixed-wing glider using lqr-trees. *Bioinspiration & Biomimetics*, 9(2):025013, 2014.
12. DMKK Venkateswara Rao, Hui Tang, and Tiauw Hiong Go. A parametric study of fixed-wing aircraft perching maneuvers. *Aerospace Science and Technology*, 42:459–469, 2015.
13. Brian L Stevens and Frank L Lewis. *Aircraft control and simulation*, volume 2. Wiley New York, 2003.
14. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.
15. Andreas Wächter. *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering*. PhD thesis, PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2002.

-
16. Andreas Wächter and T. Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005.
 17. C. Watkins. *Learning with delayed rewards*. PhD thesis, University of Cambridge, England, 1989.

11 Appendix A - Aircraft Data

This section presents the coefficients used in the numerical model for completeness.

Coefficient	Incidence range	Airspeed (m/s)				
		14	12	10	8	6
C_{L_0}	$-5^\circ \leq \alpha < 5^\circ$	0.4300	0.4100	0.3900	0.3771	0.3417
C_{L_0}	$5^\circ \leq \alpha < 10^\circ$	0.5930	0.5860	0.6040	0.6106	0.6340
C_{L_0}	$\alpha \geq 10^\circ$	1.0313	1.0260	1.0296	1.0770	1.1030
C_{L_α}	$-5^\circ \leq \alpha < 5^\circ$	4.3144	4.4576	4.5321	4.6507	4.8701
C_{L_α}	$5^\circ \leq \alpha < 10^\circ$	2.2345	2.2345	2.1199	2.0283	1.7131
C_{L_α}	$\alpha \geq 10^\circ$	-0.2750	-0.2865	-0.3209	-0.6474	-0.9740
C_{D_0}	$-5^\circ \leq \alpha < 5^\circ$	0.0840	0.0870	0.0765	0.0750	0.0682
C_{D_0}	$5^\circ \leq \alpha < 10^\circ$	0.0456	0.0394	0.0259	-0.0054	-0.0213
C_{D_0}	$\alpha \geq 10^\circ$	-0.0390	-0.0600	-0.0715	-0.0167	0.0027
C_{D_α}	$-5^\circ \leq \alpha < 0^\circ$	0.0286	0.0172	0.0172	0.0000	0.0344
C_{D_α}	$0^\circ \leq \alpha < 5^\circ$	0.2498	0.2372	0.3346	0.2653	0.3839
C_{D_α}	$5^\circ \leq \alpha < 10^\circ$	0.6898	0.7827	0.9144	1.1866	1.4078
C_{D_α}	$\alpha \geq 10^\circ$	1.1746	1.3522	1.4725	1.2519	1.2702
C_{m_0}	$-5^\circ \leq \alpha < 14^\circ$	-0.0060	0.0100	0.0200	0.0215	0.0104
C_{m_0}	$\alpha \geq 14^\circ$	-0.3065	-0.3070	-0.3230	-0.3040	-0.2908
C_{m_α}	$-5^\circ \leq \alpha < 0^\circ$	-0.7013	-0.6039	-0.6600	-0.6589	-0.6790
C_{m_α}	$0^\circ \leq \alpha < 14^\circ$	-1.2342	-1.2972	-1.4037	-1.3321	-1.2324
C_{m_α}	$\alpha \geq 14^\circ$	0.0000	0.0000	0.0000	0.0000	0.0000

Table 3 Static coefficients for different incidences and airspeeds

Coefficient	Sweep angle	Airspeed (m/s)		
		6	8	10
C_{L_q}	0°	0.5798	0.4699	0.3180
C_{L_q}	10°	0.6973	0.4653	0.4183
C_{L_q}	20°	0.5758	0.4715	0.4416
C_{L_q}	30°	0.5789	0.4946	0.3935
C_{m_q}	0°	-0.1410	-0.1349	-0.1324
C_{m_q}	10°	-0.1905	-0.1541	-0.1362
C_{m_q}	20°	-0.2136	-0.1738	-0.1588
C_{m_q}	30°	-0.2682	-0.2223	-0.1674

Table 4 Dynamic coefficients for different wing sweep angles and airspeeds

Coefficient	Incidence (degrees)				
	-5	-2.5	0	2.5	5
C_{m_η}	-0.3068	-0.3919	-0.4236	-0.4270	-0.4427

Table 5 Elevator pitching moment coefficient

Coefficient	Incidence	Airspeed (m/s)				
		6	8	10	12	14
C_{m_Λ}	-5°	-0.1730	-0.1266	-0.0476	-0.0716	-0.1942
C_{m_Λ}	0°	0.0427	0.2080	0.5094	0.8021	0.6394
C_{m_Λ}	5°	0.1923	0.4796	0.9580	1.5075	1.4290
C_{m_Λ}	10°	0.2704	0.6629	1.2932	2.0890	-
C_{m_Λ}	13°	0.2892	0.6824	1.3338	2.2638	-
C_{m_Λ}	15°	0.2968	0.6131	1.4066	-	-
C_{m_Λ}	20°	0.3025	0.7105	1.3522	-	-
C_{m_Λ}	25°	0.3283	0.7506	1.5928	-	-

Table 6 Wing sweep pitching moment coefficient. Lower right data points omitted to avoid damage to aircraft - outside aircraft flight envelope