

RESEARCH REPORT

Music-Guided Aerobatic Flight for F3P Aircraft

Learning to turn music into flight

AUTHOR

Eliau van Schalkwijk

SUPERVISORS

Ruihan Jia
Hongze Wang
Prof. Dario Floreano

DATE

September 2026

EPFL

LIS

 **LALDLAW**
SCHOLARS

Abstract

This report presents a system that turns a song into an aerobatic routine for a small indoor aeroplane, and then flies it. The strongest beats of the music become points in space, joined by a smooth trajectory timed to the beat. A neural network, trained by reinforcement learning, learns to fly this trajectory, and a physics solver checks that every instant could be performed by a real aircraft. On a real song, the learned controller reaches the beats seven times more accurately than a classical controller, and comes within a few millimetres of its targets on simpler figures. The project also uncovered a simple law linking musical tempo to how sharp a turn can be, which explains why a good choreography, like a good dancer, should not turn on every beat.

1. Introduction

Watch an F3P pilot fly to music and the aircraft seems alive: it snaps into a turn on a drum hit, climbs as the melody rises, and rolls through the chorus. F3P is a discipline of indoor precision aerobatics, in which light fixed-wing aeroplanes are flown inside a gymnasium through routines choreographed to music; judges reward both precision and how well the manoeuvres match the song. The aircraft used in this project weighs 100 g and spans 65 cm (Figure 1).



Figure 1. The F3P aircraft used in this project: a 100 g fixed-wing aeroplane with a 65 cm span, flown indoors.

Could a machine compose such a routine from a song, and then fly it? Two very different questions hide behind this one. The first is creative: how should a piece of music become a path through the air? The second is physical: that path must then be flown precisely, by a machine that obeys aerodynamics. The same pair of challenges appears in drone light shows and in robots that move in time with people.

The physical side is genuinely hard for an aeroplane. Its wings produce lift only while it moves, so slowing down, braking and turning sharply all compete with staying in the air. The music fixes *when* the aircraft must be somewhere; aerodynamics decide *how fast* it can get there and how tightly it can turn. Sharp corners call for low speed, while flying well calls for speed: the two pull in opposite directions.

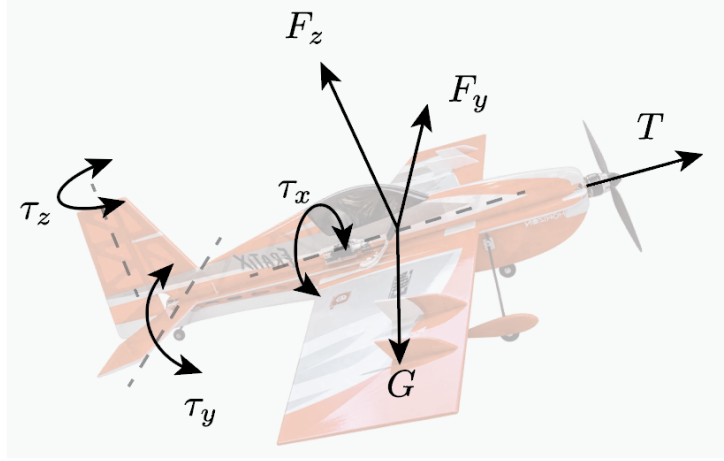
The research question was therefore:

can a flight routine be generated from a piece of music alone, and flown by a learned controller accurately enough that the aircraft reaches each beat on time, while every instant of the flight stays physically achievable?

2. Research design

2.1 How an F3P aircraft flies

An F3P aircraft is light, slow and remarkably agile. It weighs 100 g, flies between 0.5 and 7 m/s, and its propeller can pull twice its weight, enough to climb straight up. Its motion results from a small set of forces and torques (Figure 2): the thrust T of the propeller along the nose, the weight G , the aerodynamic forces on the airframe, and the forces of the control surfaces, which also produce the torques τ that make it roll, pitch and turn.



$$\mathbf{F}^b = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix} - V^2 \begin{bmatrix} k_x \\ k_y \sin \beta \\ k_z \sin \alpha \end{bmatrix} + c V_p^2 \begin{bmatrix} 0 \\ \sin \delta_r \\ -\sin \delta_e \end{bmatrix}$$

$$\boldsymbol{\tau}^b = -V^2 \begin{bmatrix} 0 \\ k_z \ell_H \sin \alpha \\ -k_y \ell_V \sin \beta \end{bmatrix} - V_p^2 \begin{bmatrix} c_a \ell_a & 0 & 0 \\ 0 & c \ell_t & 0 \\ 0 & 0 & c \ell_t \end{bmatrix} \begin{bmatrix} \sin \delta_a \\ \sin \delta_e \\ \sin \delta_r \end{bmatrix}$$

Figure 2. Forces and torques acting on the F3P aircraft, and the collective force $\mathbf{F}^b$ and torque $\boldsymbol{\tau}^b$ in the aircraft's own frame. V : airspeed (V_p : propwash); α : angle of attack; β : sideslip angle; T : thrust; δ : control surface deflection; k : airframe force coefficient; c : control surface force coefficient; ℓ : moment arm.

Two features shape everything the aircraft can do. First, every aerodynamic force grows with the square of the airspeed V : at twice the speed, the air pushes four times harder, so speed matters in every manoeuvre. Second, the tail sits in the airflow blown by the propeller, of speed V_p . Even when the aircraft hangs almost still, this propwash sends about 10 m/s of air over the tail, which is what lets F3P aircraft fly so slowly and so precisely.

The wings behave like flat plates. The force they produce grows with the angle of attack α , the angle between the wing and the oncoming air: at small angles the aircraft glides with little drag, while a wing turned across the flow becomes a powerful air brake. Section 3.3 shows how this shapes the way the aircraft can slow down.

This model is the foundation of the project. The physics judge solves its force equation at every instant of every flight (Section 2.7), and the limits of the training simulator, such as the maximum turn rate, were measured with it.

2.2 A team of specialists

The system works like a small team of specialists, each doing one job well (Figure 3). A *choreographer* listens to the song and decides where the aircraft should be on each strong beat. A *planner* draws a smooth, flyable path through those points, timed to the music. A *pilot*, the neural network, flies that path. Finally, a *judge* replays every instant of the flight and checks that a real aircraft could have performed it. Splitting the problem this way made each part simpler to build, and meant that whenever something went wrong, it was clear which member of the team to look at.

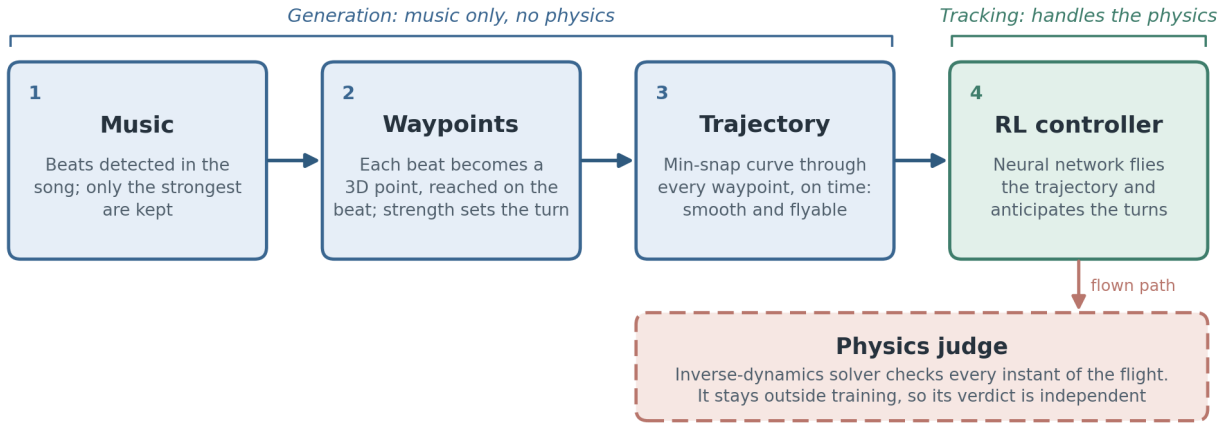


Figure 3. The four stages of the system. Generation uses the music only; the controller handles the physics; an independent judge checks the flown path.

2.3 From music to waypoints

A beautiful routine does not move on every beat. A dancer marks the accents: a strong drum hit, the start of a phrase. Turning on every beat at 200 beats per minute would mean more than three turns per second, impossible to fly and meaningless to watch. The song is therefore analysed for its *onsets*, the moments where the sound energy rises sharply [4], and each beat receives a time and a strength (Figure 4). The generator keeps the strongest beats, leaving a minimum time between two turns. Section 3.4 shows that this choice, made for aesthetic reasons, is also a physical necessity.

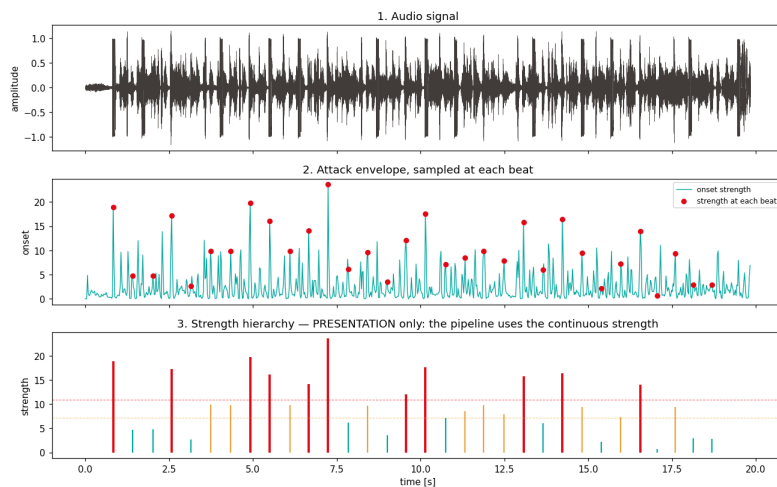


Figure 4. From sound to beats: the audio signal, its onset strength at each beat, and the beats ranked by strength. Only the strongest become turns.

Each selected beat becomes a *waypoint*: a point in 3D space that the aircraft must reach at that exact instant. The stronger the beat, the sharper the turn, from 30° to 120°. Turns are not limited to left and right: the beat strength sets both the angle and the plane of the turn, which can point left, right, up, down or anywhere in between, so the routine uses all three dimensions of the room. The generator searches these directions and keeps one that leaves the route inside a 10 m × 10 m × 10 m arena. A first version simply pushed back any point that crossed a wall, which bent the geometry and disturbed the whole trajectory; steering *before* reaching the wall solved the problem at its source.

The distance between waypoints follows the time between beats, which keeps the speed consistent. Climbs are made shorter and dives longer, because the aircraft can sustain a steep climb only slowly and a steep dive only fast.

2.4 From waypoints to a trajectory: why minimum-snap

The natural first idea was to join the waypoints with straight lines. Measuring this reference revealed its flaw: at every corner the direction changes instantly, which would require infinite acceleration. No aircraft can do that, so any controller is forced to cut corners, and is then penalised for failing at something impossible.

What was needed was a curve that passes through every waypoint at its exact beat time, is smooth enough to be flown, and is a true mathematical function of time, so that the required speed, acceleration and attitude can be obtained simply by differentiating it. *Minimum-snap* trajectories, a standard tool for agile drones [2], have exactly these properties. Each segment is a polynomial, and the polynomials are chosen to minimise the *snap*, the fourth derivative of position:

$$\min \int \left\| \frac{d^4 \mathbf{x}}{dt^4} \right\|^2 dt \quad \text{subject to} \quad \mathbf{x}(T_i) = \mathbf{W}_i$$

In practice, each segment is a seventh-order polynomial written in a normalised time u , which runs from 0 to 1 between two waypoints:

$$u = \frac{t - T_i}{\Delta t_i}, \quad p_i(u) = \sum_{k=0}^7 c_{i,k} u^k$$

Each segment therefore has eight coefficients per axis, and speed and acceleration follow by simple differentiation. The seventh order is not an arbitrary choice: it is precisely the kind of curve that minimises snap. Normalising time gives every segment the same scale, whatever its duration, which keeps the computation numerically stable. Neighbouring segments are joined so that position, speed and acceleration match where they meet. Added up over all segments, the snap becomes a quadratic function of these coefficients, written $c^\top Q c$. The matrix Q depends only on the durations of the segments, which are set by the music, and the problem is easy to minimise.

Two adaptations made it suitable for an aeroplane. First, the velocity is imposed at every waypoint, at cruise speed: standard minimum-snap starts and stops at rest, which suits a drone but not an aeroplane. Second, two *anchor* points around each corner confine the turn between them, halving the turning radius.

2.5 Teaching a neural network to fly

The pilot of the system is a neural network, and it learns to fly the way a person learns a skill: by practising and being told how well it did. This approach is called *reinforcement learning* (RL). The

network flies a simulated aircraft, receives a score after every move, and slowly adjusts itself to earn higher scores. We used a well-established RL algorithm called PPO [1].

Every tenth of a second, the network looks at 50 numbers: where the aircraft is and how it is oriented, how far it is from where it should be, where the next four waypoints lie and when they must be reached, and what it did a moment ago. From these, it decides four commands: how fast to pitch, turn and roll, and what speed to hold.

Inside the network. A neural network is a chain of simple calculators called *neurons*. Each one takes the numbers it receives, weighs them, adds them up, and passes the result through a smooth function called $\tanh$, which gently squeezes any value into the range between -1 and 1 :

$$\mathbf{h} = \tanh(W\mathbf{x} + \mathbf{b})$$

On its own, a neuron can do very little. Stacked in layers, here two layers of 64 neurons, they can represent remarkably complex behaviour. The last layer produces the four commands, squeezed once more by $\tanh$ so that they can never exceed what the aircraft is able to do. A second network of the same shape plays the role of an instructor: it estimates how much score the aircraft can still expect from where it is. Together, the two networks hold 15,177 adjustable numbers, the weights W and offsets $\mathbf{b}$.

How it learns. At first, the weights are random and the aircraft flies aimlessly; a little random noise keeps it trying new moves. After every batch of 16 simulated flights, the algorithm compares what happened with what the instructor expected: moves that went better than expected become more likely, and worse ones less likely. A method called *backpropagation* then traces this feedback back through the layers, works out how much each weight contributed, and nudges every one of them slightly in the right direction. The steps are kept small on purpose, so that one unlucky batch cannot erase what has been learned. After 1,200 such rounds, in under an hour on a standard processor and with no expert pilot to copy, the network flies the routine.

2.6 Building the reward, step by step

Most of the design effort went into the reward, whose final form has five terms:

$$r_t = \underbrace{r_{\text{beat}}}_{\text{punctuality}} + \underbrace{W_{\text{ref}} e^{-e_{\text{ref}}/1.5}}_{\text{tracking}} + \underbrace{W_{\text{sch}} (|\ell_{t-1}| - |\ell_t|)}_{\text{schedule}} \\ + \underbrace{W_{\mu} e^{-|\Delta\mu|}}_{\text{roll}} - \underbrace{W_{\text{dact}} \|a_t - a_{t-1}\|^2}_{\text{smoothness}}$$

Each term was added to solve a problem observed in training:

1. **Tracking** (second term). The first reward paid only for being close to the path. Far from it, the reward was almost zero, so the network had no signal telling it which way to go. A reward must *speak* at every distance.
2. **Schedule** (third term). A reward for progress along the trajectory gives that signal at every step. It belongs to a family of terms proven not to change the best possible behaviour, only to guide learning towards it [3].
3. **A moving target.** Chasing each waypoint as a fixed post caused overshoots and loops. The

Term	Formula	In words
Punctuality	$r_{\text{beat}} = \sum_{i: T_i \in [t-\delta t, t]} W_{\text{TIME}} \exp\left(-\frac{\ \mathbf{p}(T_i) - \mathbf{W}_i\ }{D_{\text{TIME}}}\right)$	A bonus paid once, at the exact instant of each beat, that shrinks as the aircraft gets further from the waypoint.
Tracking	$e_{\text{ref}} = \ \mathbf{p}_t - \mathbf{x}_{\text{ref}}\ $	The distance, in metres, between the aircraft and its reference point on the trajectory.
Schedule	$\ell_t = (\mathbf{p}_t - \mathbf{x}_{\text{ref}}(t)) \cdot \hat{\mathbf{d}}_{\text{ref}}(t)$	How far the aircraft is ahead of or behind schedule along the path. The term pays whenever this gap shrinks.
Roll	$\Delta\mu = \text{atan2}(\sin(\mu - \mu_{\text{ref}}), \cos(\mu - \mu_{\text{ref}}))$	The gap between the roll angle the music asks for and the actual one, kept between -180° and 180° .
Smoothness	$\ a_t - a_{t-1}\ ^2$	How much the four commands a_t change from one step to the next.

Here $\mathbf{p}$ is the position of the aircraft, T_i and $\mathbf{W}_i$ the time and position of beat i , and δt the 0.1 s control step.

target became a point sliding along the trajectory, reaching each waypoint on its beat.

4. **Punctuality** (first term), the heart of the reward: a bonus paid at the exact instant of each beat, decreasing with the distance to the waypoint. Its first weight looked dominant, yet it fired only a few times per flight while the other terms fired at every step, so it represented just 8% of the total reward, and the network flew the right path at the wrong time. A weight means nothing without its frequency; the reward was rebalanced until punctuality became the largest share.
5. **Roll** (fourth term) makes the aircraft follow the rolls requested by the music.
6. **Smoothness** (last term) penalises abrupt changes of command. On the cube, it cut the jerkiness of the commands by 38% with no loss of accuracy.

2.7 Test cases and the physics judge

The system was validated on increasingly demanding cases. The **cube**, proposed by the supervisors, links eight corners with sharp 90° turns and vertical edges: the ideal test for crisp corners and for climbing and diving. The **figure-of-eight** tests smooth, continuous curvature. The **song** is the real application.

As a baseline, the learned controller is compared with a classical controller designed by hand, which steers towards a point slightly ahead on the trajectory. Finally, every instant of every flight is passed to an inverse-dynamics solver provided by the lab. It searches for an angle of attack, a sideslip angle and a thrust that balance all the forces; if none exists within the aircraft's limits, the instant is counted as physically impossible.

3. Analysis and findings

3.1 The learned controller flies the routines

Accuracy is measured as the distance between the aircraft and the waypoint at the exact instant of each beat, never the closest approach within a time window, which would hide lateness.

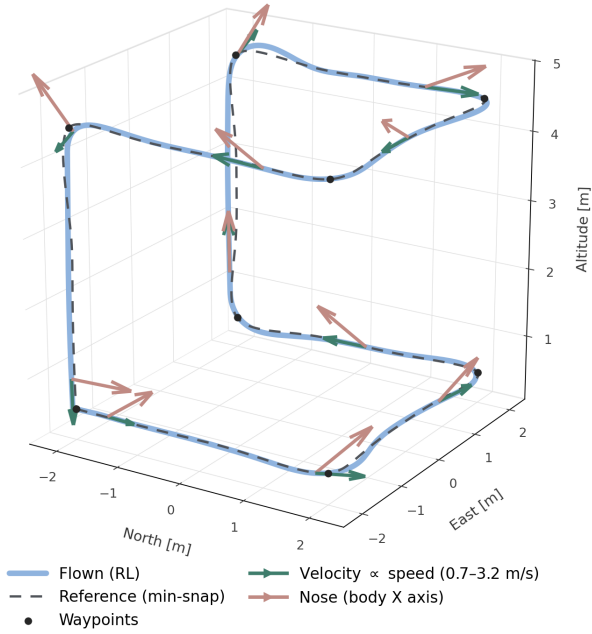


Figure 5. The cube: waypoints (black dots), reference (dashed) and flown path (blue). Green arrows show the velocity, red arrows the direction of the nose.

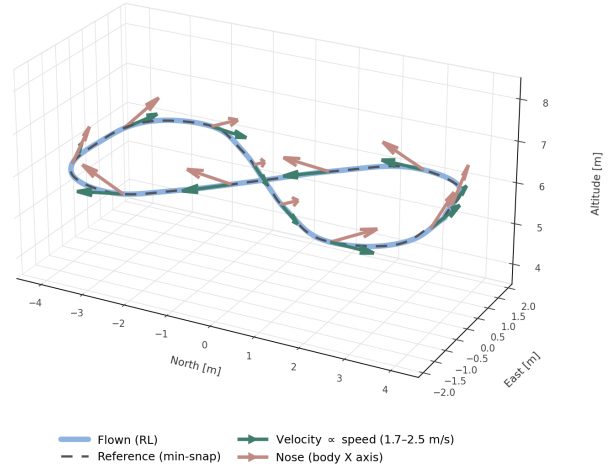


Figure 6. The figure-of-eight: reference trajectory (dashed) and path flown by the learned controller (blue), which overlap almost exactly.

Test case	Learned controller	Classical controller	Gain
Figure-of-eight	0.007 m	0.041 m	6×
Song, 32 beats (20 s)	0.31 m	2.15 m	7×
Second song, 35 beats (53 s)	0.25 m	0.75 m	3×

Mean error at the beat instant.

On the cube, the aircraft flies its sharp 90° corners with a mean error of about 4 cm. On the song, the result holds across repeated training runs and across three different choreographies generated from it: the method is not a lucky configuration, and training on a new routine gives a good result again.

The learned controller wins because it anticipates. The classical controller only reacts to its current error: it discovers a turn when it reaches it, overshoots, then corrects. The network sees the next four waypoints and learns to prepare, starting a turn early or adjusting its speed before a climb, so that it arrives on the beat instead of catching up after it.

3.2 Turning points

The most important progress did not come from tuning, but from spotting wrong assumptions:

- **Letting the network fly.** For several days, the network was only allowed to add small corrections to the classical controller, and training waited until that controller performed well enough. This was backwards: anticipating is precisely what the classical controller cannot do and the network can. Once trained from scratch, with the classical controller as a baseline to beat rather than a gate, the error on the song dropped by a factor of 9.5 in a single training run.

- **Giving the network real authority.** A first way of turning the network’s outputs into commands let it use only 7% of the aircraft’s maximum turn rate in practice. A simpler, linear mapping quadrupled its authority.

3.3 Flying it for real

The physics judge showed that almost all difficult instants occur in dives, where the aircraft must brake while descending. Pitching the nose up to brake with drag made things 48 to 700 times worse: at high angles, drag behaves like a switch rather than a dial, jumping from about 1 N to 5–7.5 N when only 0.14 N of extra braking was needed. A small reverse thrust of 20% of the maximum, which electric F3P motors can produce, works precisely: it changes only the instants that needed it, and brings the share of physically impossible instants on the song down to 0%.

3.4 A law for sharp turns

On the song, the turns looked rounder than hoped. The controller was not to blame: it was already using its full command budget, within 2% of the tightest turn physically possible. Two intuitive fixes, slowing down in the corners or asking for gentler turns, both failed.

The explanation is a short piece of algebra. At these speeds, the tightest turn is limited by how fast the aircraft can rotate: its minimum radius is its speed divided by its maximum turn rate ω . But the speed is set by the music, since a segment of length L must be covered in the time Δt between two beats. Combining the two gives:

$$\frac{R_{\min}}{L} = \frac{1}{\omega_{\max} \Delta t}$$

How much the aircraft turns is set by the strength of the beat, and how tightly it can take that turn depends on how fast it can rotate. The equation shows that **the time between turns also plays a major role**: the size of the figure, the segment length and the speed all cancel out, while the timing remains. The law was confirmed exactly on a test cube. With one second between turns, the cube’s corners come out twice as sharp as the song’s, where turns follow each other every 0.58 s.

This closes the loop with Section 2.3. Selecting the strongest beats is not only an aesthetic choice: by leaving more time between turns, it gives the aircraft the room it needs for sharp corners.

4. Conclusion

This project set out to show that an aerobatic aeroplane could fly a routine generated from music alone, on the beat, within the laws of physics. The complete pipeline now works end to end, and on a real song the learned controller reaches the beats seven times more accurately than a classical one.

The project also produced a result of wider interest: besides the strength of the beat and the agility of the aircraft, the time between beats plays a major role in how sharp a turn can be. This turns timing into a design rule for choreographers, and gives a physical reason to do what dancers already do, marking the accents rather than every count.

Several directions would take the work further. Counting full rotations in the roll reward would let the aircraft perform the barrel rolls the music asks for. Making the generator aware of the

aircraft's limits, while keeping the music in charge, would remove most remaining difficulties. And the natural next step is to fly the routines on the real aircraft.

On a personal level, the most valuable lesson was about method. The biggest steps forward came from questioning an assumption rather than tuning a parameter. Looking at the actual flight rather than only at averages, testing one variable at a time and repeating each experiment made the difference between a result that looks good and one that holds.

Acknowledgements

I would like to thank Ruihan Jia and Hongze Wang for their daily guidance and for providing the physical model and solver at the heart of this work, and Prof. Dario Floreano for hosting me at the Laboratory of Intelligent Systems (LIS), EPFL. This research was supported by the Laidlaw Scholars programme.

References

- [1] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv:1707.06347.
- [2] Mellinger, D. and Kumar, V. (2011). *Minimum snap trajectory generation and control for quadrotors*. IEEE International Conference on Robotics and Automation.
- [3] Ng, A. Y., Harada, D. and Russell, S. (1999). *Policy invariance under reward transformations: theory and application to reward shaping*. International Conference on Machine Learning.
- [4] McFee, B. et al. (2015). *librosa: Audio and music signal analysis in Python*. Proceedings of the 14th Python in Science Conference.