Learning Quadrotor Controls Using Data-Efficient Policy Search

Sahand Rezaei-Shoshtari
Department of Mechanical Engineering
McGill University
Montreal, Canada
sahand.rezaei-shoshtari@mail.mcgill.ca

***Abstract*— This research attempts to apply a data-efficient policy search method to control a quadrotor. The simulated quadrotor has no prior knowledge of the dynamical model, nor of the control policies. Over the course of learning, with use of the powerful PILCO method, the quadrotor models its dynamical behavior and optimizes its control policy. This method enables the quadrotor to learn how to take off and hover at a particular altitude. It slightly underperformed in the case of yawing, however, in a more complicated scenario, it was capable of controlling the quadrotor using only three actuators. As this method has shown promising results in the implemented scenarios, it can be applied to more complex tasks such as collision recovery.**

I. Introduction

This research studies the performance of learning quadrotor controller purely from the experience data. The research was inspired by several related works in the field of robotics; authors in [1] were successful to create adapting spider robot capable of moving after loss of an actuator. Additionally, current research thrust in Aerospace Mechatronics Laboratory at McGill University focuses on the collision recovery in quadrotors [2, 3]. Collision recovery is a highly non-linear and complicated task, given its complex dynamical equations [4]; therefore, designing a controller for this task is extremely demanding.

Furthermore, loss of an actuator adds multiple levels of complexity to the already arduous task of collision recovery. The idea here is to make use of data-driven methods to learn both the quadrotor dynamics and controllers undergoing the collision.

The controller learning approach is based on a policy search technique named PILCO (Probabilistic Inference for Learning Control) [5]. This method has been successfully applied to several complex robotic systems including a gait learning swimming Aqua robot [6]. The main advantage of PILCO is its ability to optimize control policies without specific knowledge of the underlying dynamical system of equations, making it perfectly suited for the problem at hand, namely flying with three propellers and collision recovery.

The learned controllers for the simulator quadrotor demonstrate successful performance at simple hovering tasks. However, the performance dropped dramatically in more complex tasks such as yawing. Surprisingly, the method was able to learn to fly with three propellers. It seems as if there are multiple points needed to be taken into consideration for a successful learning; the definition of the loss function and the parameters from the state-space which are being fed to the policy function are among the most impactful factors.

The remainder of this paper is structured as follows. Section II introduces related works. Section III describes the learning methodology used in this research. Section IV presents the details behind the learning scenarios and the simulation. Section V presents the results obtained from the simulation process.

II. Background

A. *Learning Quadrotor Controls*

Programming drone to fly autonomously is regarded as a highly challenging task. However, with the advances in artificial intelligence and applied machine learning, this arduous task has become easy over the years. P. Abbeel and A. Ng were able to fly a helicopter autonomously for the first time [7, 8], and their work has established a new path in the context of applying learning to flying robotics.

Several authors have applied reinforcement learning to the problem of finding suitable parameters for the controller of a quadrotor [9-14]. Some researchers have applied Bayesian Optimization and Gaussian processes to tune a quadrotor controller [15, 16]. Others have coupled Gaussian Processes with other learning algorithms to implement a powerful model predictive controller [17, 18]. With the recent advances in the area of deep learning, some authors have applied deep neural networks and deep reinforcement learning to control a quadrotor [19-21].

The previous works done in this area have all led to successful results, however, none of the papers have addressed the challenging task of flying a quadrotor with three actuators which is one of the main goals of this research. Inspired by [22], we are trying to control the position of a quadrotor once an actuator has been lost, after which the yaw angle cannot be controlled with three propellers.

B. *PILCO*

The PILCO algorithm [5] has been successfully applied to control several mechanical systems (e.g. cart pole double pendulum, unicycle, robotic manipulator and swimming Aqua robot [6]). This method is a promising choice for reinforcement learning on physical platforms due to its ability to handle continuous states and controls. Additionally, PILCO is applicable to high state dimensional systems and it achieves excellent data efficiency [6]. However, the computational cost of the method scales dramatically with the amount of experience and state dimensionality, increasing the learning time of each iteration to approximately one to two hours in the case of a quadrotor.

The PILCO algorithm consists of three high-level modules: model learning, policy learning, and policy application.

1) Model Learning

The forward model learned is a non-parametric, probabilistic Gaussian process [23]. The non-parametric property of the GP does not require an explicit task-dependant parametrization of the dynamics of the system. Furthermore, the probabilistic property of the GP reduces the effect of model errors [24].

Inputs to the GP are state-action pairs (x_t, u_t) in which t is the time stamp. Training targets are either successor states x_{t+1} or differences $\Delta_t = x_{t+1} - x_t$. The GP model is trained by evidence maximization, where high signal-to-noise ratios are penalized to maintain numerical stability [24].

2) Policy Learning

PILCO uses the learned GP forward model to compute approximate long-term predictions $p(x_1|\pi), \dots, p(x_T|\pi)$ for a given controller π. To follow this procedure, the analytical moment-matching approach proposed by [25] is applied and all $p(x_t|\pi)$ are approximated by Gaussians $\mathcal{N}(\mu_0, \Sigma_0)$. The expected long-term cost is then computed analytically for the cost function $c(x)$.

$$J^{\pi} = \sum_{t=1}^{T} \mathbb{E}[c(x_t)|\pi] \tag{1}$$

A Gaussian-shaped cost function has been recommended by the designers of the PILCO algorithm [5].

$$c(x) = 1 - exp\left(-\frac{1}{2}\left\|x - x_{target}\right\|_W^2 / \sigma_c^2\right), \tag{2}$$

Where $\left\|x - x_{target}\right\|_W^2$ is the Mahalanobis distance between x and x_{target}, weighted by W and σ_c is a scaling factor. The policy is then improved with the utilization of gradient-based Quasi-Newton optimization methods, such as BFGS. The gradients of the long-term cost with respect to the policy parameters are computed analytically, which allows the learning of low-level policies with hundreds of parameters [24]. However, the downside is the high computation cost of the optimization step.

3) Policy Application

The learned policy π is then applied either to the simulated system or the actual robot. The generated state-action pairs are used to feed the GP model and the previous steps are initiated again.

III. Learning Method

This section gives an overview of the system developed for this research project, its controls, and the loss function. In each learning episode, the information regarding the states of the quadrotor is fed to the RL control module. The RL control module accepts tasks from the user in the form of a loss function, and tries to reach the target position given its current state by applying low-level control commands, the RPMs of the motors, to each of the actuators. Initially, the quadrotor has no prior knowledge of the governing dynamics, nor of the control policy. Over the course of learning, experience data is used to learn both the dynamical model of the quadrotor and the control policies for the task. Each of the main elements of the system is further described in the following section.

A. Quadrotor State Space

At each time step, the quadrotor receives the components of its 13-dimensional state space from the simulator. The state space vector consists of the quadrotor velocity, body rates, position and attitude quaternions, respectively.

$$\mathbf{s} = [u, v, w, p, q, r, X, Y, Z, q_w, q_x, q_y, q_z]^T \tag{3}$$

Even though Euler angles provide an intuitive representation of the orientation of the quadrotor, attitude quaternions have been used so that the so called “gimbal lock” problem would not occur [26].

B. Quadrotor Control Space

The policy has low-level control over the rotational speed of each propeller $\mathbf{u} = [u_1, u_2, u_3, u_4]$. It is worth noting that the rotational speeds suggested by the control policy cannot be instantly applied to the actuator; instead they are passed to the simulator and the actual speed of each motor is altering smoothly, ensuring the critical assumption underlying Gaussian processes, namely the smoothness of the data.

C. Loss Function

The loss function is considered one of the key factors in convergence of the learning process. This function must be defined by a human designer in terms of the state-space. Following the recommendation of authors in [5], the loss function is created by a unity minus Gaussian function with width σ centered at the specified goal to ensure smoothness. The width needs to be set manually using design heuristics [6].

IV. Simulation Methodology

First, this section describes the details of the approach in evaluating the learning performance in three sample tasks. The simulator created by Gareth Dicker and Fiona Chui [26, 27], former McGill students in Aerospace Mechatronics Laboratory[1], has been used in this research. This section continues by explaining the details behind each scenario and the simulator.

[1] http://aerospacemechatronics.lab.mcgill.ca/

A. Sample Tasks

A total of three sample tasks have been tested on the quadrotor to evaluate its learning performance. The first two tasks are rather simple relative to the third one:

1. **Hovering:** take off using four actuators and hover in a particular altitude without any change in the yaw angle.
2. **Yawing:** take off using four actuators and hover in a particular altitude while changing the yaw angle.
3. **Hovering with three propellers:** take off using three actuators and hover at a particular altitude.

A quadrotor has four control outputs, whereas it has a total of six degrees of freedom, thereby, it is an under-actuated system in which it is impossible to control all six degrees of freedom simultaneously. As a result, four degrees of freedom should be chosen to be controlled by the four control outputs. A typical choice is the position vector and the yaw angle ψ or yaw rate $\dot{\psi}$ [26]. Nevertheless, all of the states in the 13-dimensional state-space is included in the loss function in the first and second sample tasks, since eventually the quadrotor needs to remain stable in a specified position.

However, this is not the case with the third sample task. With the loss of an actuator, only a total of three degrees of freedom can be controlled at each time stamp. Therefore, the position is chosen to be controlled and the yaw angle ψ is free to change. This has been applied to the loss function by excluding two of the attitude quaternions from the loss equation. As we would want to make the quadrotor hover in a particular altitude, pitch θ and roll ϕ angles should remain zeros. Given that, two of the quaternions (q_y and q_z) would remain zero while the two others change with the yaw angle ψ. The two non-zero quaternions are excluded from the loss function and the two zero quaternions are included to apply the constraint of zero pitch θ and roll ϕ angles to the learning method.

B. Quadrotor Simulator

Gareth Dicker and Fiona Chui have implemented the simulator in MATLAB using its standard libraries and the MATLAB Aerospace Toolbox [26, 27]. The dynamical model takes into account most important dynamic effects, namely the terms due to propeller acceleration and saturation limits of this acceleration. In addition, the thrust and drag force were computed as a multiple of square of the propeller speeds [26].

Table 1. Dynamics and simulation parameters [26].

Parameter	Value	Units	Description
g	9.81	m/s^2	Gravitational acceleration
m	1.096	kg	Quadrotor mass
I	diag [0.008 0.008 0.017]	$kg.m^2$	Quadrotor moments of inertia
J_r	2.208e-05	$kg.m^2$	Propeller moment of inertia
l	22	cm	Arm length
k_t	8.7e-8	N/RPM^2	Thrust coefficient
k_d	8.7e-9	Nm/RPM^2	Drag coefficient
dt	0.025	s	Simulation time step

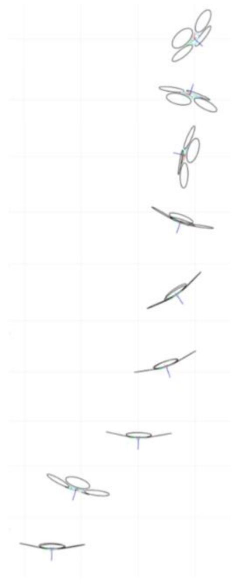
Figure 1. Snapshots of visualization from a simulation trial [26].

The modeled quadrotor, named Navi, has been designed based on a quadrotor called Spiri manufactured by Pleiades Inc. Table 1 summarizes the key simulation parameters and Figure 1 displays the snapshots of a sample simulation trial [26].

V. Simulation Results

This section presents the results obtained from the learned policy applied to the quadrotor. For each task, the first 10 trials are created by following a random policy. The goal of these trials is to gather enough experience for the initial dataset to train the GP. Once enough data has been collected, the following iterations are all accompanied by GP update and policy optimization.

The prediction horizon time is set to 2.5 seconds in all of the learning iterations. However, once the quadrotor hits the ground constraint, the trial finishes earlier than the prediction horizon time. This mostly happens in the first 10 trials in which the random policy is used, therefore, the shorter horizon time does not pose any problem in the learning process.

A. Task 1: Hovering

The quadrotor learns to take off and perform the hovering maneuver after 22 trials (including the initial random trials). Total time of interaction with the environment is 36 seconds. The Q matrix in the loss function is chosen as follows. The intuition behind that is to put more focus on static states (i.e. position and attitude quaternions) and less emphasis on the velocity and body rates so that the quadrotor would be allowed to speed up to reach its goal.

$$Q = diag\ [0.1\ 0.1\ 0.1\ 0.5\ 0.5\ 0.5\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$$

Figure 2 displays the immediate cost once the task is learned, and Figure 3 shows the strong performance of the GP in learning the dynamical model. A video showing this learning scenario is available online[2].

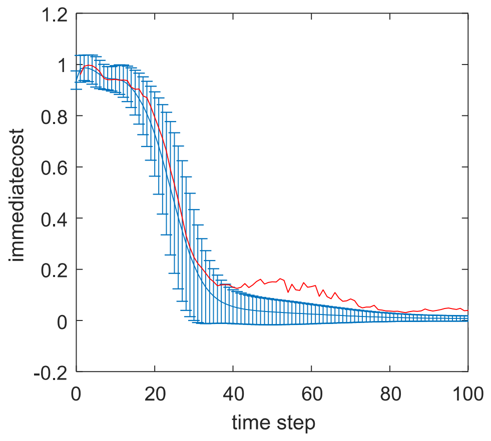

Figure 2. Immediate cost function in the last trial of task 1. The motors spin fast at first so that the quadrotor flies upward as fast as possible. Once it gets near the goal, the propellers are slowed than.

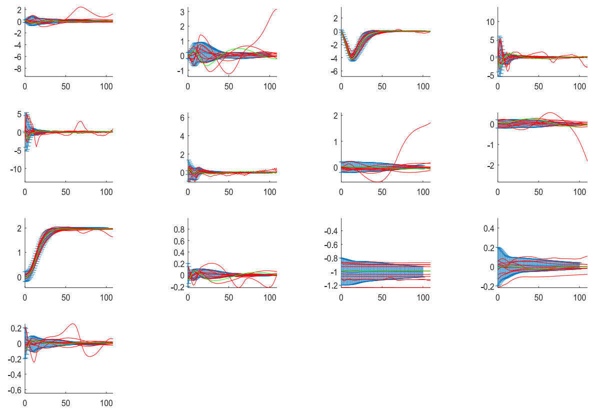

Figure 3. Performance of the Gaussian process in modeling the dynamical system in task 1. Each diagram corresponds to one of the states.

B. *Task 2: Yawing*

The target in this task is to hover at an altitude with a 180° change in the yaw angle. After a total of 29 trials (including the initial random trials) and gathering 55.5 seconds of data, the quadrotor was not able to completely learn the yawing task. However, it has achieved the goal to some extent by reaching the desired position and trying to change the yaw angle. It seems as if in more trials, it would have been able to fulfill the task. Unfortunately, due to the time constraint, the run had to be terminated.

The same loss function from task 1 has been utilized here. Figures 4 and 5 display the results obtained from the learned policy. An online video showing this task is also available[3].

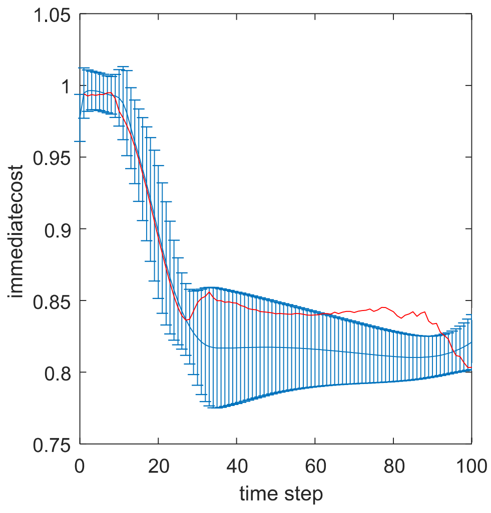

Figure 4. Immediate cost function in the last trial of task 2. Although the learning process is not fully successful and the quadrotor is not able to change its yaw angle, it reaches the target position and changes the yaw angle slightly. Optimization of the loss function have prematurely converged to a sub optimal policy.

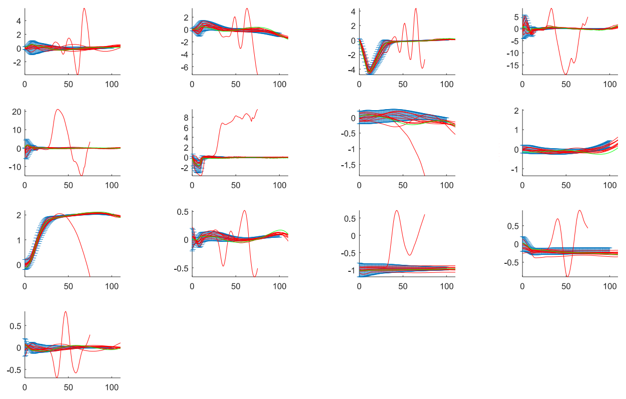

Figure 5. Performance of the Gaussian process in modeling the dynamical system in task 2. Each diagram corresponds to one of the states.

C. *Task 3: Hovering with Three Propellers*

The target in this task is to take off and hover the quadcopter using only three actuators. In order to compensate for the loss of an actuator, the maximum rpm of the rotors have been increased slightly. After a total of 27 trials and gathering 41 seconds of data, the quadrotor was able to stabilize its position to some extent. However, due to the instability in the yaw angle, it cannot remain at the desired position for a long time. The Q matrix in the loss function was chosen as follows. This is the same as the previous tasks, but with no constraint on the yaw angle.

$$Q = diag\ [0.1\ 0.1\ 0.1\ 0.5\ 0.5\ 0.5\ 1\ 1\ 1\ 0\ 0\ 1\ 1]$$

Figure 6 and 7 display the performance of the quadrotor in this task. A video of this task is also available online[4].

[2] https://youtu.be/DMLAIIAM2RA
[3] https://youtu.be/Vi_xmHiWcOU
[4] https://youtu.be/5SIGOiIapPM

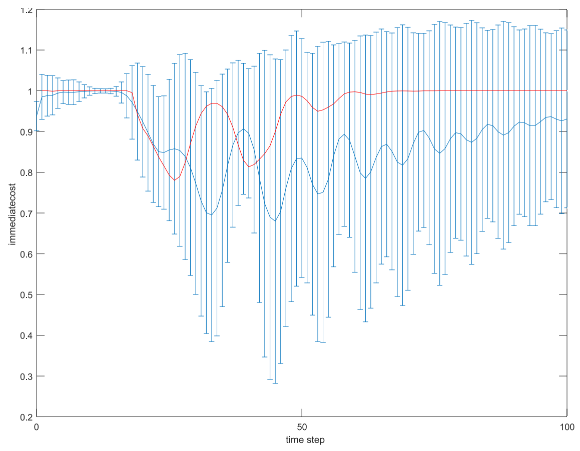

Figure 6. Immediate cost function in the last trial of task 3. The cost function decreases at first when the quadrotor reaches the desired position. However, oscillations in yaw angle drags the quadrotor away from the target and the cost function increases. Perhaps, use of other cost functions would make the learning process more efficient.

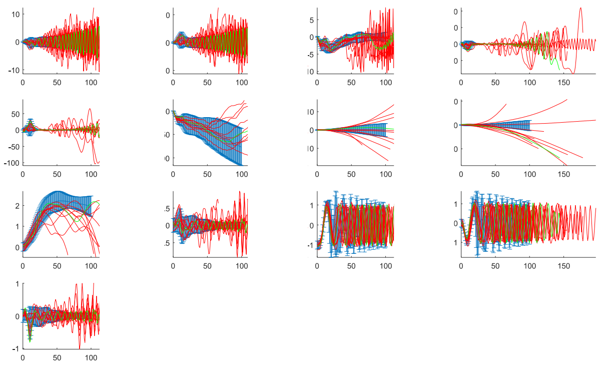

Figure 7. Performance of the Gaussian process in modeling the dynamical system in task 3. Each diagram corresponds to one of the states.

VI. Conclusion

This research presented a method to allow quadrotors to learn multiple flying tasks. The method was based on the PILCO algorithm, which was adapted to find the rotational speed of the motors in each task. The quadrotor was able to take off and hover at a particular altitude. Although the performance was not very successful at changing the yaw angle, the quadrotor was capable of flying with three propellers.

Similar to other learning methods, the definition of the loss function and the parameters chosen to be passed to the policy function have direct impact on the overall performance. Therefore, with changes to these underlying modules, the method can work on far more complicated scenarios.

Future work consists of applying the PILCO method to the collision recovery of a quadrotor. The complexity nature of this task may require further investigation, as the smoothness condition necessary for the Gaussian processes may not be satisfied. Another direction of the future works would be to study the application of hierarchical control in connecting multiple learned policies, each with its own strength.

References

[1] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature,* vol. 521, no. 7553, pp. 503-507, 2015.

[2] A. Battiston, I. Sharf, and M. Nahon, "Attitude estimation for normal flight and collision recovery of a quadrotor UAV," in *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*, 2017, pp. 840-849: IEEE.

[3] G. Dicker, F. Chui, and I. Sharf, "Quadrotor collision characterization and recovery control," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017, pp. 5830-5836: IEEE.

[4] F. Chui, G. Dicker, and I. Sharf, "Dynamics of a quadrotor undergoing impact with a wall," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, 2016, pp. 717-726: IEEE.

[5] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465-472.

[6] D. Meger, J. C. G. Higuera, A. Xu, P. Giguere, and G. Dudek, "Learning legged swimming gaits from experience," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 2332-2338: IEEE.

[7] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research,* vol. 29, no. 13, pp. 1608-1639, 2010.

[8] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in neural information processing systems*, 2007, pp. 1-8.

[9] M. Shaker, M. N. Smith, S. Yue, and T. Duckett, "Vision-based landing of a simulated unmanned aerial vehicle with fast reinforcement learning," in *Emerging Security Technologies (EST), 2010 International Conference on*, 2010, pp. 183-188: IEEE.

[10] J. Valasek, J. Doebbler, M. D. Tandale, and A. J. Meade, "Improved adaptive–reinforcement learning control for morphing unmanned air vehicles," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 38, no. 4, pp. 1014-1020, 2008.

[11] B. Zhang, Z. Mao, W. Liu, and J. Liu, "Geometric reinforcement learning for path planning of UAVs," *Journal of Intelligent & Robotic Systems,* vol. 77, no. 2, pp. 391-409, 2015.

[12] H. Bou-Ammar, H. Voos, and W. Ertel, "Controller design for quadrotor uavs using reinforcement learning," in *Control Applications (CCA), 2010 IEEE International Conference on*, 2010, pp. 2130-2135: IEEE.

[13] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 1642-1648: IEEE.

[14] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters,* vol. 2, no. 4, pp. 2096-2103, 2017.

[15] R. R. Duivenvoorden, F. Berkenkamp, N. Carion, A. Krause, and A. P. Schoellig, "Constrained Bayesian Optimization with Particle Swarms for Safe Adaptive Controller Tuning," 2017.

[16] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016, pp. 491-496: IEEE.

[17] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *Control Conference (ECC), 2015 European*, 2015, pp. 2496-2501: IEEE.

[18] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian Process Model Predictive Control of an Unmanned Quadrotor," *Journal of Intelligent & Robotic Systems,* vol. 88, no. 1, pp. 147-162, 2017.

[19] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016, pp. 528-535: IEEE.

[20] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," *arXiv preprint arXiv:1610.06283,* 2016.

[21] R. Polvara *et al.*, "Autonomous Quadrotor Landing using Deep Reinforcement Learning," *arXiv preprint arXiv:1709.03339,* 2017.

[22] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 45-52: IEEE.

[23] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006.

[24] M. P. Deisenroth, A. McHutchon, J. Hall, and C. E. Rasmussen, "PILCO Code Documentation v0. 9," 2013.

[25] J. Q. Candela, A. Girard, J. Larsen, and C. E. Rasmussen, "Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, 2003, vol. 2, pp. II-701: IEEE.

[26] G. Dicker, "Quadrotor Reorientation Control for Collision Recovery," Master of Engineering, Department of Mechanical Engineering, McGill University, 146878, 2017.

[27] F. Chui, "Quadrotor Collision Dynamics and Fuzzy Logic Characterization," Master of Engineering, Department of Mechanical Engineering, McGill University, 146839, 2017.