Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Real-Time Stochastic Optimal Control for Multi-Agent Quadrotor Systems

Vicenç Gómez[1], Sep Thijssen[2], Andrew Symington[3], Stephen Hailes[4], Hilbert J. Kappen[2]

[1]Universitat Pompeu Fabra. Barcelona, Spain (vicen.gomez@upf.edu)
[2]Radboud University Nijmegen, the Netherlands ({s.thijssen,b.kappen}@donders.ru.nl)
[3]University of California Los Angeles, USA (andrew.c.symington@gmail.com)
[4]University College London, United Kingdom (s.hailes@cs.ucl.ac.uk)

Abstract

This paper presents a novel method for controlling teams of unmanned aerial vehicles using Stochastic Optimal Control (SOC) theory. The approach consists of a centralized high-level planner that computes optimal state trajectories as velocity sequences, and a platform-specific low-level controller which ensures that these velocity sequences are met. The planning task is expressed as a centralized path-integral control problem, for which optimal control computation corresponds to a probabilistic inference problem that can be solved by efficient sampling methods. Through simulation we show that our SOC approach (a) has significant benefits compared to deterministic control and other SOC methods in multimodal problems with noise-dependent optimal solutions, (b) is capable of controlling a large number of platforms in real-time, and (c) yields collective emergent behaviour in the form of flight formations. Finally, we show that our approach works for real platforms, by controlling a team of three quadrotors in outdoor conditions.

1 Introduction

The recent surge in autonomous Unmanned Aerial Vehicle (UAV) research has been driven by the ease with which platforms can now be acquired, evolving legislation that regulates their use, and the broad range of applications enabled by both individual platforms and cooperative swarms. Example applications include automated delivery systems, monitoring and surveillance, target tracking, disaster management and navigation in areas inaccessible to humans.

Quadrotors are a natural choice for an experimental platform, as they provide a safe, highly-agile and inexpensive means by which to evaluate UAV controllers. Figure 1 shows a 3D model of one such quadrotor, the *Ascending Technologies Pelican*. Quadrotors have non-linear dynamics and are naturally unstable, making control a non-trivial problem.

Stochastic optimal control (SOC) provides a promising theoretical framework for achieving autonomous control of quadrotor systems. In contrast to deterministic control, SOC directly captures the uncertainty typically present in noisy environments and leads to solutions that qualitatively depend on the level of uncertainty (Kappen 2005). However, with the exception of the simple Linear Quadratic Gaussian case, for which a closed form solution exists, solving the SOC problem requires solving the Hamilton Jacobi Bellman (HJB) equations. These equations are generally intractable, and so the SOC problem remains an open challenge.

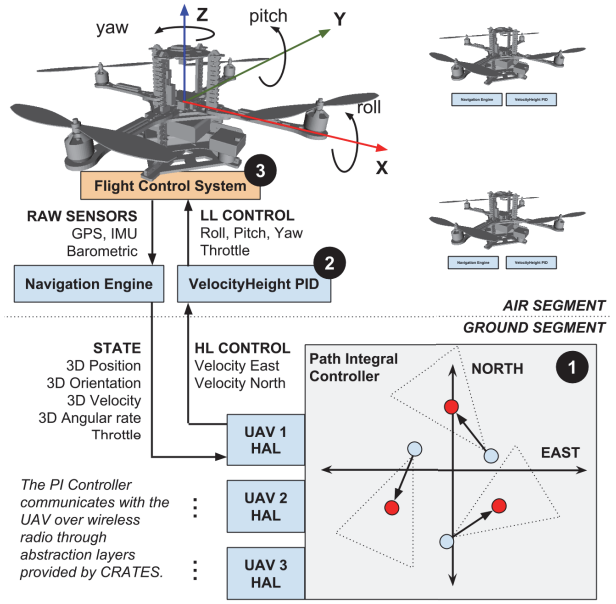

Figure 1: Control hierarchy: The path-integral controller (1) calculates target velocities/heights for each quadrotor. These are converted to roll, pitch, throttle and yaw rates by a platform-specific Velocity Height PID controller (2). This control is in turn passed to the platform's flight control system (3), and converted to relative motor speed changes.

In such a complex setting, a hierarchical approach is usually taken and the control problem is reduced to follow a state-trajectory (or a set of way points) designed by hand or computed offline using trajectory planning algorithms (Kendoul 2012). While the planning step typically involves a low-dimensional state representation, the control methods use a detailed complex state representation of the UAV. Examples of control methods for trajectory tracking are the Proportional Integral Derivative or the Linear-Quadratic regulator.

A generic class of SOC problems was introduced in Kappen; Todorov (2005; 2006) for which the controls and the cost function are restricted in a way that makes the HJB equation linear and therefore more efficiently solvable. This class of problems is known as path integral (PI) control, linearly-solvable controlled diffusions or Kullback-Leibler control, and it has lead to successful robotic applications, e.g. (Kinjo, Uchibe, and Doya 2013; Rombokas et al. 2012; Theodorou, Buchli, and Schaal 2010). A particularly interesting feature of this class of problems is that the computation of optimal control is an inference problem with a solution given in terms of the passive dynamics. In a multi-agent system, where the agents follow independent passive dynamics, such a feature can be exploited using approximate inference methods such as variational approximations or belief propagation (Kappen, Gómez, and Opper 2012; Van Den Broek, Wiegerinck, and Kappen 2008).

In this paper, we show how PI control can be used for solving motion planning tasks on a team of quadrotors in real time. We combine periodic re-planning with receding horizon, similarly to model predictive control, with efficient importance sampling. At a high level, each quadrotor is modelled as a point mass that follows simple double integrator dynamics. Low-level control is achieved using a standard Proportional Integral Derivative (PID) velocity controller that interacts with a real or simulated flight control system. With this strategy we can scale PI control to ten units in simulation. Although in principle there are no further limits to experiments with actual platforms, our first results with real quadrotors only include three units. To the best of our knowledge this has been the first real-time implementation of PI control on an actual multi-agent system.

In the next section we describe related work. We introduce our approach in Section 3 Results are shown on three different scenarios in Section 4 Finally, Section 5 concludes this paper.

2 Related Work on UAV Planning and Control

There is a large and growing body of literature related to this topic. In this section, we highlight some of the most related papers to the presented approach. An recent survey of control methods for general UAVs can be found in Kendoul (2012).

Stochastic optimal control is mostly used for UAV control in its simplest form, assuming a linear model perturbed by additive Gaussian noise and subject to quadratic costs (LQG), e.g. (How et al. 2008). While LQG can successfully perform simple actions like hovering, executing more complex actions requires considering additional corrections for aerodynamic effects such as induced power or blade flapping (Hoffmann et al. 2011). These approaches are mainly designed for accurate trajectory control and assume a given desired state trajectory that the controller transforms into motor commands.

Model Predictive Control (MPC) has been used optimize trajectories in multi-agent UAV systems (Shim, Kim, and Sastry 2003). MPC employs a model of the UAV and solves an optimal control problem at time t and state $x(t)$ over a future horizon of a fixed number of time-steps. The first optimal move $u^*(t)$ is then applied and the rest of the optimal sequence is discarded. The process is repeated again at time $t + 1$. A quadratic cost function is typically used, but other more complex functions exist.

MPC has mostly been used in indoor scenarios, where high-precision motion capture systems are available. For instance, in Kushleyev et al. (2013) authors generate smooth trajectories through known 3-D environments satisfying specifications on intermediate waypoints and show remarkable success controlling a team of 20 quadrotors. Trajectory optimization is translated to a relaxation of a mixed integer quadratic program problem with additional constraints for collision avoidance, that can be solved efficiently in real-time. Examples that follow a similar methodology can be found in Turpin, Michael, and Kumar; Augugliaro, Schoellig, and D'Andrea (2012; 2012). Similarly to our approach, these methods use a simplified model of dynamics, either using the 3-D position and yaw angle Kushleyev et al.; Turpin, Michael, and Kumar (2013; 2012) or the position and velocities as in Augugliaro, Schoellig, and D'Andrea (2012). However, these approaches are inherently deterministic and express the optimal control problem as a quadratic problem. In our case, we solve an inference problem by sampling and we do not require intermediate trajectory waypoints.

In outdoor conditions, motion capture is difficult and Global Positioning System (GPS) is used instead. Existing control approaches are typically either based on Reynolds flocking (Bürkle, Segor, and Kollmann 2011; Hauert et al. 2011; Vásárhelyi et al. 2014; Reynolds 1987) or flight formation (Guerrero and Lozano 2012; Yu et al. 2013). In Reynolds flocking, each agent is considered a point mass that obeys simple and distributed rules: separate from neighbors, align with the average heading of neighbors and steer towards neighborhood centroid to keep cohesion. Flight formation control is typically modeled using graphs, where every node is an agent that can exchange information with all or several agents. Velocity and/or position coordination is usually achieved using consensus algorithms.

The work in Quintero, Collins, and Hespanha (2013) shares many similarities with our approach. Authors derive a stochastic optimal control formulation of the flocking problem for fixed-wings UAVs. They take a leader-follower strategy, where the leader follows an arbitrary (predefined) policy that is learned offline and define the immediate cost as a function of the distance and heading with respect to the leader. Their method is demonstrated outdoors with 3 fixed-wing UAVs in a distributed sensing task. As in this paper, they formulate a SOC problem and perform MPC. However, in our case we do not restrict to a leader-follower setup and consider a more general class of SOC problems which can include coordination and cooperation problems.

Planning approaches Within the planning community, Bernardini, Fox, and Long (2014) consider search and tracking tasks, similar to one of our scenarios. Their approach is different to ours, they formulate a planning problem that uses used *search patterns* that must be selected and se-

quenced to maximise the probability of rediscovering the target. Albore et al. (2015) and Chanel, Teichteil-Knigsbuch, and Lesire (2013) consider a different problem: dynamic data acquisition and environmental knowledge optimisation. Both techniques use some form of replanning. While Albore et al. (2015) uses a Markov Random Field framework to represent knowledge about the uncertain map and its quality, Chanel, Teichteil-Knigsbuch, and Lesire (2013) rely on partially-observable MDPs. All these works consider a single UAV scenario and low-level control is either neglected or deferred to a PID or waypoint controller.

Recent Progress in Path-Integral Control There has been significant progress in PI control, both theoretically and in applications. Most of existing methods use parametrized policies to overcome the main limitations (see Section 3.1). Examples can be found in Theodorou, Buchli, and Schaal; Stulp and Sigaud; Gómez et al. (2010; 2012; 2014). In these methods, the optimal control solution is restricted by the class of parametrized policies and, more importantly, it is computed offline. In Rawlik, Toussaint, and Vijayakumar (2013), authors propose to approximate the transformed cost-to-go function using linear operators in a reproducing kernel Hilbert space. Such an approach requires an analytical form of the PI embedding, which is difficult to obtain in general. In Horowitz, Damle, and Burdick (2014), a low-rank tensor representation is used to represent the model dynamics, allowing to scale PI control up to a 12-dimensional system. More recently, the issue of state-dependence of the optimal control has been addressed (Thijssen and Kappen 2015), where a parametrized state-dependent feedback controller is derived for the PI control class.

Finally, model predictive PI control has been recently proposed for controlling a nano-quadrotor in indoor settings in an obstacle avoidance task (Williams, Rombokas, and Daniel 2014). In contrast to our approach, their method is not hierachical and uses naive sampling, which makes it less sample efficient. Additionally, the control cost term is neglected, which can have important implications in complex tasks involving noise. The approach presented here scales well to several UAVs in outdoor conditions and is illustrated in tasks beyond obstacle avoidance navigation.

3 Path-Integral Control for Multi-UAV planning

We first briefly review PI control theory. This is followed by a description of the proposed method used to achieve motion planning of multi-agent UAV systems using PI control.

3.1 Path-Integral Control

We consider continuous time stochastic control problems, where the dynamics and cost are respectively linear and quadratic in the control input, but arbitrary in the state. More precisely, consider the following stochastic differential equation of the state vector $\mathbf{x} \in \mathbb{R}^n$ under controls $\mathbf{u} \in \mathbb{R}^m$

$$d\mathbf{x} = \mathbf{f}(\mathbf{x})dt + \mathbf{G}(\mathbf{x})(\mathbf{u}dt + d\boldsymbol{\xi}), \tag{1}$$

where $\boldsymbol{\xi}$ is $m-$dimensional Wiener noise with covariance $\Sigma_u \in \mathbb{R}^{m\times m}$ and $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^n$ and $\mathbf{G}(\mathbf{x}) \in \mathbb{R}^{n\times m}$ are arbitrary functions, $\mathbf{f}$ is the drift in the uncontrolled dynamics (including gravity, Coriolis and centripetal forces), and $\mathbf{G}$ describes the effect of the control $\mathbf{u}$ into the state vector $\mathbf{x}$.

A realization $\boldsymbol{\tau} = \mathbf{x}_{0:dt:T}$ of the above equation is called a (random) path. In order to describe a control problem we define the cost that is attributed to a path (cost-to-go) by

$$S(\boldsymbol{\tau}|\mathbf{x}_0, \mathbf{u}) = r_T(\mathbf{x}_T) + \sum_{t=0:dt:T-dt} \left(r_t(\mathbf{x}_t)dt + \frac{1}{2}\mathbf{u}_t^\top \mathbf{R}\mathbf{u}_t \right) dt, \tag{2}$$

where $r_T(\mathbf{x}_T)$ and $r_t(\mathbf{x}_t)$ are arbitrary state cost terms at end and intermediate times, respectively. $\mathbf{R}$ is the control cost matrix. The general stochastic optimal control problem is to minimize the expected cost-to-go w.r.t. the control

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \mathbb{E}[S(\boldsymbol{\tau}|\mathbf{x}_0, \mathbf{u})].$$

In general, such a minimization leads to the Hamilton-Jacobi-Bellman (HJB) equations, which are non-linear, second order partial differential equations. However, under the following relation between the control cost and noise covariance $\Sigma_u = \lambda\mathbf{R}^{-1}$, the resulting equation is *linear* in the exponentially transformed cost-to-go function. The solution is given by the Feynman-Kac Formula, which expresses optimal control in terms of a Path-Integral, which can be interpreted as taking the expectation under the optimal path distribution (Kappen 2005)

$$p^*(\boldsymbol{\tau}|\mathbf{x}_0) \propto p(\boldsymbol{\tau}|\mathbf{x}_0, \mathbf{u}) \exp(-S(\boldsymbol{\tau}|\mathbf{x}_0, \mathbf{u})/\lambda), \tag{3}$$

$$\langle \mathbf{u}_t^*(\mathbf{x}_0) \rangle = \langle \mathbf{u}_t + (\boldsymbol{\xi}_{t+dt} - \boldsymbol{\xi}_t)/dt \rangle, \tag{4}$$

where $p(\boldsymbol{\tau}|\mathbf{x}_0, \mathbf{u})$ denotes the probability of a (sub-optimal) path under equation (1) and $\langle\cdot\rangle$ denotes expectation over paths distributed by p^*.

The constraint $\Sigma_u = \lambda\mathbf{R}^{-1}$ forces control and noise to act in the same dimensions, but in an inverse relation. Thus, for fixed λ, the larger the noise, the cheaper the control and vice-versa. Parameter λ act as a temperature: higher values of λ result in optimal solutions that are closer to the uncontrolled process.

Equation (4) permits optimal control to be calculated by probabilistic inference methods, e.g., Monte Carlo. An interesting fact is that equations (3, 4) hold for all controls $\mathbf{u}$. In particular, $\mathbf{u}$ can be chosen to reduce the variance in the Monte Carlo computation of $\langle \mathbf{u}_t^*(\mathbf{x}_0) \rangle$ which amounts to importance sampling. This technique can drastically improve the sampling efficiency, which is crucial in high dimensional systems. Despite this improvement, direct application of PI control into real systems is limited because it is not clear how to choose a proper importance sampling distribution. Furthermore, note that equation (4) yields the optimal control for all times t averaged over states. The result is therefore an open-loop controller that neglects the state-dependence of the control beyond the initial state.

3.2 Multi-UAV planning

The proposed architecture is composed of two main levels. At the most abstract level, the UAV is modeled as a 2D point-mass system that follows double integrator dynamics.

Algorithm 1 PI control for UAV motion planning

function PICONTROLLER($N, H, dt, r_t(\cdot), \Sigma_u, \bullet_{t:t+H}$)
  for $k = 1, \ldots, N$ **do**
    Sample paths $\boldsymbol{\tau}_k = \{\mathbf{x}_{t:dt:t+H}\}_k$ with Eq. (5)
  end for
  Compute $S_k = S(\boldsymbol{\tau}_k | \mathbf{x}_0, \mathbf{u})$ with Eq. (2)
  Store the noise realizations $\{\boldsymbol{\xi}_{t:dt:t+H}\}_k$
  Compute the weights: $w_k = e^{-S_k/\lambda} / \sum_l e^{-S_l/\lambda}$
  for $s = t : dt : t + H$ **do**
    $\mathbf{u}^*_s = \mathbf{u}_s + \frac{1}{dt} \sum_k w_k \left(\{\boldsymbol{\xi}_{s+dt}\}_k - \{\boldsymbol{\xi}_s\}_k \right)$
  end for
  Return next desired velocity: $\mathbf{v}_{t+dt} = \mathbf{v}_t + \mathbf{u}^*_t dt$ and $\mathbf{u}^*_{t:dt:t+H}$ for importance sampling at $t + dt$
end function

At the low-level, we use a detailed second order model that we learn from real flight data (De Nardi and Holland 2008). We use model predictive control combined with importance sampling. There are two main benefits of using the proposed approach: first, since the state is continuously updated, the controller does not suffer from the problems caused by using an open-loop controller. Second, the control policy is not restricted by any parametrization.

The two-level approach permits to transmit control signals from the high-level PI controller to the low-level control system at a relatively low frequencies (we use 15Hz in this work). Consequently, the PI controller has more time available for sampling a large number of trajectories, which is critical to obtain good estimates of the control. The choice of 2D in the presented method is not a fundamental limitation, as long as double-integrator dynamics is used. The control hierarchy introduces additional model mismatch. However, as we show in the results later, this mismatch is not critical for obtaining good performance in real conditions.

Ignoring height, the state vector $\mathbf{x}$ is thus composed of the East-North (EN) positions and EN velocities of each agent $i = 1, \ldots, M$ as $x_i = [p_i, v_i]^\top$ where $p_i, v_i \in \mathbb{R}^2$. Similarly, the control $\mathbf{u}$ consists of EN accelerations $u_i \in \mathbb{R}^2$. Equation (1) decouples between the agents and takes the linear form

$$dx_i = Ax_i dt + B(u_i dt + d\xi_i),$$
$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{5}$$

Notice that although the dynamics is decoupled and linear, the state cost $r_t(\mathbf{x}_t)$ in equation (2) can be any arbitrary function of all UAVs states. As a result, the optimal control will in general be a non-linear function that couples all the states and thus hard to compute.

Given the current joint optimal action $\mathbf{u}^*_t$ and velocity $\mathbf{v}_t$, the expected velocity at the next time t' is calculated as $\mathbf{v}_{t'} = \mathbf{v}_t + (t' - t)\mathbf{u}^*_t$ and passed to the low-level controller. The final algorithm optionally keeps an importance-control sequence $\mathbf{u}_{t:dt:t+H}$ that is incrementally updated. We summarize the high-level controller in Algorithm 1.

The importance-control sequence $\mathbf{u}_{t:dt:t+H}$ is initialized using prior knowledge or with zeros otherwise. Noise is dimension-independent, i.e. $\Sigma_u = \sigma_u^2 \mathrm{Id}$. To measure sampling convergence, we define the *Effective Sample Size* (ESS) as ESS $:= 1/\sum_{k=1}^{N} w_k^2$, which is a quantity between 1 and N. Values of ESS close to one indicate an estimate dominated by just one sample and a poor estimate of the optimal control, whereas an ESS close to N indicates near perfect sampling, which occurs when the importance- equals the optimal-control function.

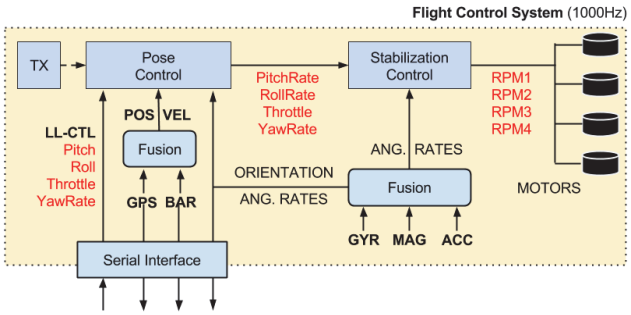

Figure 2: The flight control system (FCS) is comprised of two control loops: one for stabilization and the other for pose control. A low-level controller interacts with the FCS over a serial interface to stream measurements and issue control.

3.3 Low Level Control

The target velocity $\mathbf{v} = [v_E\ v_N]^\top$ is passed along with a height $\hat{p}_U$ to a Velocity-Height controller. This controller uses the current state estimate of the real quadrotor $\mathbf{y} = [p_E\ p_N\ p_U\ \phi\ \theta\ \psi\ u\ v\ w\ p\ q\ r]^\top$, where (p_E, p_N, p_U) and (ϕ, θ, ψ) denote navigation-frame position and orientation and $(u, v, w), (p, q, r)$ denote body-frame and angular velocities, respectively. It is composed of four independent PID controllers for roll $\hat{\phi}$, pitch $\hat{\theta}$, throttle $\hat{\gamma}$ and yaw rate $\hat{r}$. that send the commands to the flight control system (FCS) to achieve $\mathbf{v}$.

Figure 2 shows the details of the FCS. The control loop runs at 1kHz fusing triaxial gyroscope, accelerometer and magnetometer measurements. The accelerometer and magnetometer measurements are used to determine a reference global orientation, which is in turn used to track the gyroscope bias. The difference between the desired and actual angular rates are converted to motor speeds using the model in Mahony, Kumar, and Corke (2012).

An outer pose control loop calculates the desired angular rates based on the desired state. Orientation is obtained from the inner control loop, while position and velocity are obtained by fusing GPS navigation fixes with barometric pressure (BAR) based altitude measurements. The radio transmitter (marked TX in the diagram) allows the operator to switch quickly between autonomous and manual control of a platform. There is also an acoustic alarm on the platform itself, which warns the operator when the GPS signal is lost or the battery is getting low. If the battery reaches a critical level or communication with the transmitter is lost, the platform can be configured to land immediately or alternatively, to fly back and land at its take-off point.

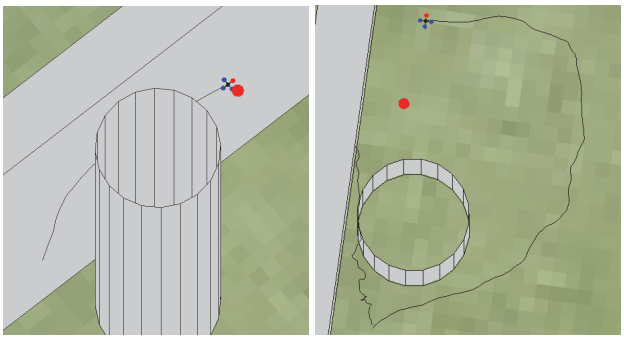

Figure 3: Drunken Quadrotor: a red target has to be reached while avoiding obstacles. (Left) the shortest route is the optimal solution in the absence of noise. (Right) with control noise, the optimal solution is to fly around the building.

3.4 Simulator Platform

We have developed an open-source framework called CRATES[1]. The framework is a implementation of QRSim (De Nardi 2013; Symington et al. 2014) in Gazebo, which uses Robot Operating System (ROS) for high-level control. It permits high-level controllers to be platform-agnostic. It is similar to the Hector Quadrotor project (Meyer et al. 2012) with a formalized notion of a hardware abstraction layers.

The CRATES simulator propagates the quadrotor state forward in time based on a second order model (De Nardi and Holland 2008). The equations were learned from real flight data and verified by expert domain knowledge. In addition to platform dynamics, CRATES also simulates various noise-perturbed sensors, wind shear and turbulence. Orientation and barometric altitude errors follow zero-mean Ornstein-Uhlenbeck processes, while GPS error is modeled at the pseudo range level using trace data available from the International GPS Service. In accordance with the Military Specification MIL-F-8785C, wind shear is modeled as a function of altitude, while turbulence is modeled as a discrete implementation of the Dryden model. CRATES also provides support for generating terrain from satellite images and light detection and ranging (LIDAR) technology, and reporting collisions between platforms and terrain.

4 Results

We now analyze the performance of the proposed approach in three different tasks. We first show that, in the presence of control noise, PI control is preferable over other approaches. For clarity, this scenario is presented for one agent only. We then consider two tasks involving several units: a flight formation task and a pursuit-evasion task.

We compare the PI control method described in Section 3.2 with iterative linear-quadratic Gaussian (iLQG) control (Todorov and Li 2005). iLQG is a state-of-the-art method based on differential dynamic programming, that iteratively computes local linear-quadratic approximations to the finite horizon problem. A key difference between iLQG and PI control is that the linear-quadratic approximation is certainty equivalent. Consequently, iLQG yields a noise independent solution.

[1]CRATES stands for 'Cognitive Robotics Architecture for Tightly-Coupled Experiments and Simulation'. Available at https://bitbucket.org/vicengomez/crates

4.1 Scenario I: Drunken Quadrotor

This scenario is inspired in Kappen (2005) and highlights the benefits of SOC in a quadrotor task. The Drunken Quadrotor is a finite horizon task where a quadrotor has to reach a target, while avoiding a building and a wall (figure 3). There are two possible routes: a shorter one that passes through a small gap between the wall and the building, and a longer one that goes around the building. Unlike SOC, the deterministic optimal solution does not depend on the noise level and will always take the shorter route. However, with added noise, the risk of collision increases and thus the optimal noisy control is to take the longer route.

This task can be alternatively addressed using other planning methods, such as the one proposed by Ono, Williams, and Blackmore (2013), which allow for specification of user's acceptable levels of risk using chance constraints. Here we focus on comparing deterministic and stochastic optimal control for motion planning. The amount of noise thus determines whether the optimal solution is go through the risky path or the longer safer path.

The state cost in this problem consists of hard constraints that assign infinite cost when either the wall or the building is hit. PI control deals with collisions by killing particles that hit the obstacles during Monte Carlo sampling. For iLQG, the local approximations require a twice differentiable cost function. We resolved this issue by adding a smooth obstacle-proximity penalty in the cost function. Although iLQG computes linear feedback, we tried to improve it with a MPC scheme, similar as for PI control. Unfortunately, this leads to numerical instabilities in this task, since the system disturbances tend to move the reference trajectory through a building when moving from one time step to the next. For MPC with PI control we use a receding horizon of three seconds and perform re-planning at a frequency of 15 Hz with $N = 2000$ sample paths. Both methods are initialized with $\mathbf{u}_t = 0, \forall t$. iLQG requires approximately 10^3 iterations to converge with a learning rate of 0.5%.

Figure 3 (left) shows an example of real trajectory computed for low control noise level, $\sigma_u^2 = 10^{-3}$. To be able to obtain such a trajectory we deactivate sensor uncertainties in accelerometer, gyroscope, orientation and altimeter. External noise is thus limited to aerodynamic turbulences only. In this case, both iLQG and PI solutions correspond to the shortest path, i.e. go through the gap between the wall and the building. Figure 3 (right) illustrates the solutions obtained for larger noise level $\sigma_u^2 = 1$. While the optimal reference trajectory obtained by iLQG does not change, which results in collision once the real noisy controller is executed (left path), the PI control solution avoids the building and takes the longer route (right path). Note that iLQG can find both solutions depending on initialization. However, However, it will always choose the shortest route, regardless of nearby obstacles. Also, note that the PI controlled unit takes a longer route to reach the target. The reason is that the con-

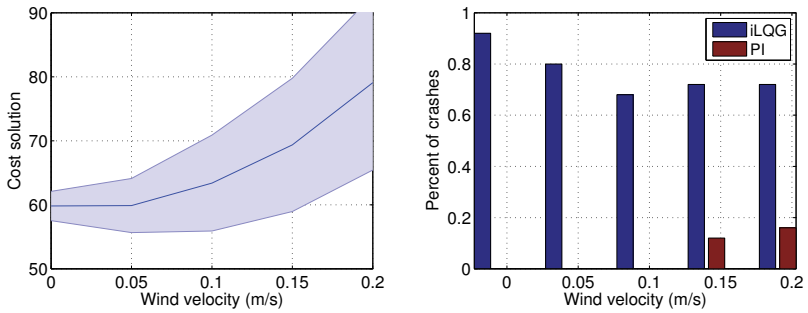

Figure 4: Results: Drunken Quadrotor with wind: For different wind velocities and fixed control noise $\sigma_u^2 = 0.5$. (Left) cost of the obtained solutions and (Right) percentage of crashes using iLQG and PI.

trol cost $\mathbf{R}$ is set quite high in order to reach a good ESS. Alternatively, if $\mathbf{R}$ is decreased, the optimal solution could reach the target sooner, but at the cost of a decreased ESS. This trade-off, which is inherent in PI control, can be resolved by incorporating feedback control in the importance sampling, as presented in Thijssen and Kappen (2015).

We also consider more realistic conditions with noise not limited to act in the control. Figure 4 (a,b) shows results in the presence of wind and sensor uncertainty. Panel (a) shows how the wind affects the quality of the solution, resulting in an increase of the variance and the cost for stronger wind. In all our tests, iLQG is not able to bring the quadrotor to the other side. Panel (b) shows the percentage of crashes using both methods. Crashes occur often using iLQG control and only occasionally using PI control. With stronger wind, the iLQG controlled unit does occasionally not even reach the corridor (the unit did not reach the other side but did not crash either). This explains the difference in percentages of Panel (b). We conclude that for multi-modal tasks (tasks where multiple solution trajectories exist), the proposed method is preferable to iLQG.

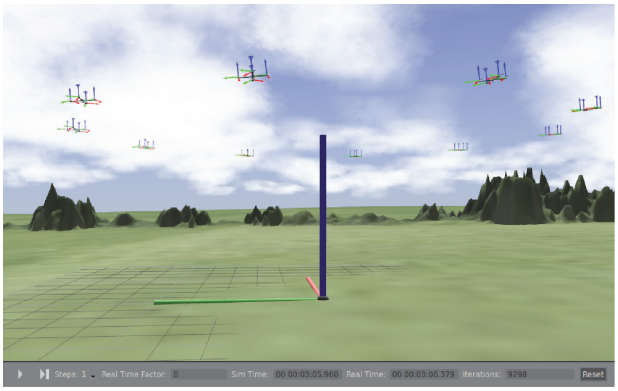

Figure 5: Holding pattern in the CRATES simulator. Ten units coordinate their flight in a circular formation. In this example, $N = 10^4$ samples, control noise is $\sigma_u^2 = 0.1$ and horizon $H = 1$ sec. Cost parameters are $v_{\min} = 1, v_{\max} = 3, C_{\text{hit}} = 20$ and $\mathrm{d} = 7$. Environmental noise and sensing uncertainties are modeled using realistic parameter values.

4.2 Scenario II: Holding Pattern

The second scenario addresses the problem of coordinating agents to hold their position near a point of interest while keeping a safe range of velocities and avoiding crashing into each other. Such a problem arises for instance when multiple aircraft need to land at the same location, and simultaneous landing is not possible. The resulting flight formation has been used frequently in the literature (Vásárhelyi et al. 2014; How et al. 2008; Yu et al. 2013; Franchi et al. 2012), but always with prior specification of the trajectories. We show how this formation is obtained as the optimal solution of a SOC problem.

Consider the following state cost (omitting time indexes)

$$r_{\text{HP}}(x) = \sum_{i=1}^{M} \exp\left(v_i - v_{\max}\right) + \exp\left(v_{\min} - v_i\right) + \exp\left(\| p_i - \mathrm{d} \|_2\right) + \sum_{j>i}^{M} C_{\text{hit}} / \| p_i - p_j \|_2 \tag{6}$$

where $v_{\max}$ and $v_{\min}$ denote the maximum and minimum velocities, respectively, d denotes penalty for deviation from the origin and C_{hit} is the penalty for collision risk of two agents. $\| \cdot \|_2$ denotes ℓ-2 norm.

The optimal solution for this problem is a circular flying pattern where units fly equidistantly from each other. The value of parameter d determines the radius and the average velocities of the agents are determined from $v_{\min}$ and $v_{\max}$. Since the solution is symmetric with respect to the direction of rotation (clockwise or anti-clockwise), only when the control is executed, a choice is made and the symmetry is broken. Figure 5 shows a snapshot of a simulation after the flight formation has been reached for a particular choice of parameter values [2]. Since we use an uninformed initial control trajectory, there is a transient period during which the agents organize to reach the optimal configuration. The coordinated circular pattern is obtained regardless of the initial positions. This behavior is robust and obtained for a large range of parameter values.

[2] Supplementary video material is available at http://www.mbfys.ru.nl/staff/v.gomez/uav.html

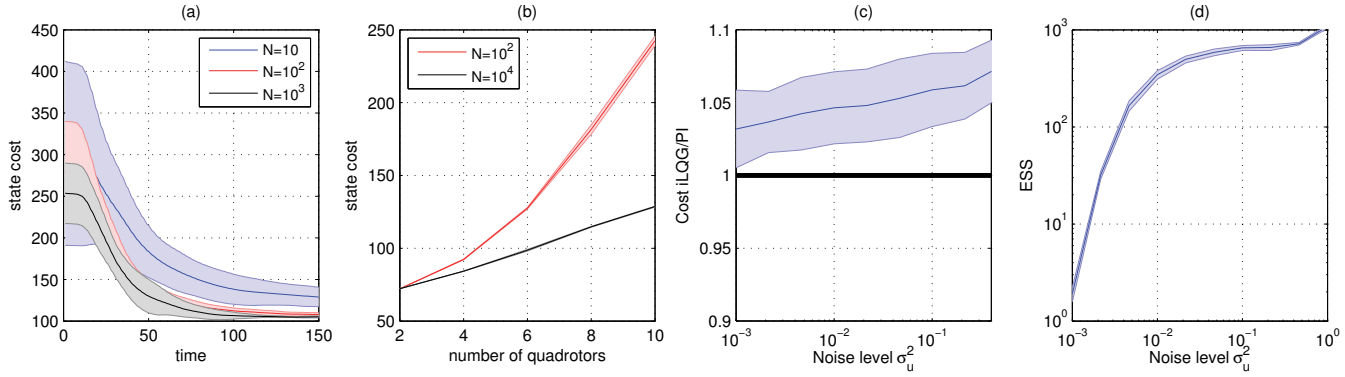

Figure 6: Holding pattern: (a) evolution of the state cost for different number of samples $N = 10, 10^2, 10^3$. (b) scaling of the method with the number of agents. For different control noise levels, (c) comparison between iLQG and PI control (ratios > 1 indicate better performance of PI over iLQG) and (d) Effective Sample Sizes. Errors bars correspond to ten different random realizations.

Figure 6(a) shows immediate costs at different times. Cost always decreases from the starting configuration until the formation is reached. This value depends on several parameters. We report its dependence on the number N of sample paths. For large N, the variances are small and the cost attains small values at convergence. Conversely, for small N, there is larger variance and the obtained dynamical configuration is less optimal (typically the distances between the agents are not the same). During the formation of the pattern the controls are more expensive. For this particular task, full convergence of the path integrals is not required, and the formation can be achieved with a very small N.

Figure 6(b) illustrates how the method scales as the number of agents increases. We report averages over the mean costs over 20 time-steps after one minute of flight. We varied M while fixing the rest of the parameters (the distance d which was set equal to the number of agents in meters). The small variance of the cost indicates that a stable formation is reached in all the cases. As expected, larger values of N lead to smaller state cost configurations. For more than ten UAVs, the simulator starts to have problems in this task and occasional crashes may occur before the formation is reached due to limited sample sizes. This limitation can be addressed, for example, by using more processing power and parallelization and it is left for future work.

We also compared our approach with iLQG in this scenario. Figure 6(c) shows the ratio of cost differences after convergence of both solutions. Both use MPC, with a horizon of 2s and update frequency of 15Hz. Values above 1 indicate that PI control consistently outperforms iLQG in this problem. Before convergence, we also found, as in the previous task, that iLQG resulted in occasional crashes while PI control did not. The Effective Sample Size (ESS) is shown in Figure 6(d). We observe that higher control noise levels result in better exploration and thus better controls. We can thus conclude that the proposed methodology is feasible for coordinating a large team of quadrotors.

For this task, we performed experiments with the real platforms. Figure 7 shows real trajectories obtained in outdoor

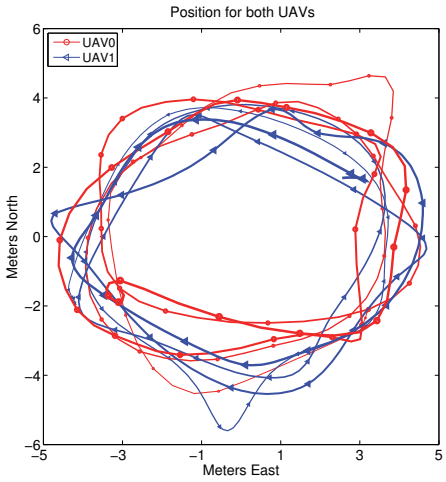

Figure 7: Resulting trajectories of a Holding Pattern experiment using two platforms in outdoors conditions.

conditions (see also the video that accompanies this paper for an experiment with three platforms). Despite the presence of significant noise, the circular behavior was also obtained. In the real experiments, we used a Core i7 laptop with 8GB RAM as base station, which run its own ROS messaging core and forwarded messages to and from the platforms over a IEEE 802.11 2.4GHz network. For safety reasons, the quadrotors were flown at different altitudes.

4.3 Scenario III: Cat and Mouse

The final scenario that we consider is the cat and mouse scenario. In this task, a team of M quadrotors (the cats) has to catch (get close to) another quadrotor (the mouse). The mouse has autonomous dynamics: it tries to escape the cats by moving at velocity inversely proportional to the distance to the cats. More precisely, let p_{mouse} denote the 2D position of the mouse, the velocity command for the mouse is

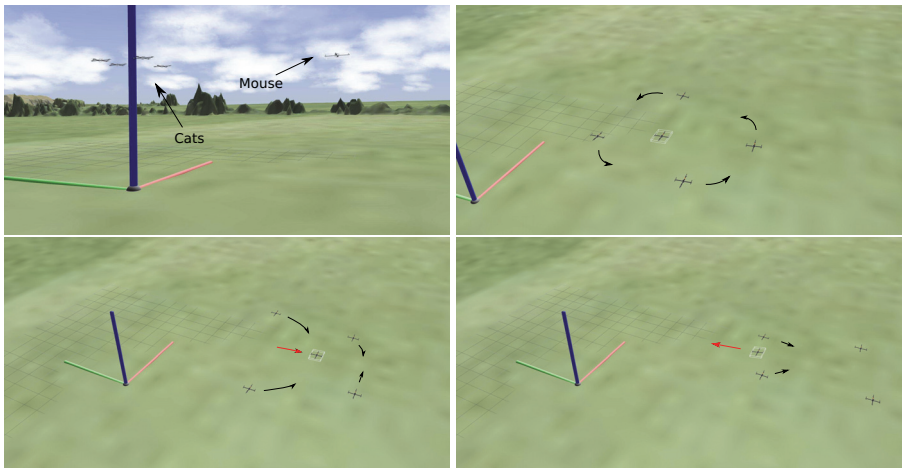

Figure 8: Cat and mouse scenario: (Top-left) four cats and one mouse. (Top-right) for horizon time $H = 2$ seconds, the four cats surround the mouse forever and keep rotation around it. (Bottom-left) for horizon time $H = 1$ seconds, the four cats chase the mouse but (bottom-right) the mouse manages to escape. With these settings, the multi-agent system alternates between these two dynamical states. Number of sample paths is $N = 10^4$, noise level $\sigma_u^2 = 0.5$. Other parameter values are d $= 30$, $v_{\text{min}} = 1$, $v_{\text{max}} = 4$, $v_{\text{min}} = 4$ and $v_{\text{max mouse}} = 3$.

computed (omitting time indexes) as

$$v_{\text{mouse}} = v_{\underset{\text{mouse}}{\text{max}}} \frac{v}{\| v \|_2}, \quad \text{where } v = \sum_{i=1}^{M} \frac{p_i - p_{\text{mouse}}}{\| p_i - p_{\text{mouse}} \|^2}.$$

The parameter $v_{\underset{\text{mouse}}{\text{max}}}$ determines the maximum velocity of the mouse. As state cost function we use equation (6) with an additional penalty term that depends on the sum of the distances to the mouse

$$r_{\text{CM}}(x) = r_{\text{HP}}(x) + \sum_{i=1}^{M} \| p_i - p_{\text{mouse}} \|_2 .$$

This scenario leads to several interesting dynamical states. For example, for a sufficiently large value of M, the mouse always gets caught (if its initial position is not close to the boundary, determined by d). The optimal control for the cats consists in surrounding the mouse to prevent collision. Once the mouse is surrounded, the cats keep rotating around it, as in the previous scenario, but with the origin replaced by the mouse position. The additional video shows examples of other complex behaviors obtained for different parameter settings. Figure 8 (top-right) illustrates this behavior.

The types of solution we observe are different for other parameter values. For example, for $M = 2$ or a small time horizon, e.g. $H = 1$, the dynamical state in which the cats rotate around the mouse is not stable, and the mouse escapes. This is displayed in Figure 8 (bottom panels) and better illustrated in the video provided as supplementary material. We emphasize that these different behaviors are observed for large uncertainty in the form of sensor noise and wind.

5 Conclusions

This paper presents a centralized, real-time stochastic optimal control algorithm for coordinating the actions of multiple autonomous vehicles in order to minimize a global cost function. The high-level control task is expressed as a Path-Integral control problem that can be solved using efficient sampling methods and real-time control is possible via the use of re-planning and model predictive control. To the best of our knowledge, this is the first real-time implementation of Path-Integral control on an actual multi-agent system.

We have shown in a simple scenario (Drunken Quadrotor) that the proposed methodology is more convenient than other approaches such as deterministic control or iLQG for planning trajectories. In more complex scenarios such as the Holding Pattern and the Cat and Mouse, the proposed methodology is also preferable and allows for real-time control. We observe multiple and complex group behavior emerging from the specified cost function. Our experimental framework `CRATES` has been a key development that permitted a smooth transition from the theory to the real quadrotor platforms, with literally no modification of the underlying control code. This gives evidence that the model mismatch caused by the use of a control hierarchy is not critical in normal outdoor conditions. Our current research is addressing the following aspects:

Large scale parallel sampling– the presented method can be easily parallelized, for instance, using graphics processing units, as in Williams, Rombokas, and Daniel (2014). Although the tasks considered in this work did not required more than 10^4 samples, we expect that this improvement will significantly increase the number of application domains and system size.

Distributed control– we are exploring different distributed formulations that take better profit of the factorized representation of the state cost. Note that the costs functions considered in this work only require pairwise couplings of the agents (to prevent collisions). However, full observability of the joint space is still required, which is not available in a fully distributed approach.

References

Albore, A.; Peyrard, N.; Sabbadin, R.; and Teichteil Königsbuch, F. 2015. An online replanning approach for crop fields mapping with autonomous UAVs. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Augugliaro, F.; Schoellig, A.; and D'Andrea, R. 2012. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *Intelligent Robots and Systems (IROS)*, 1917–1922.

Bernardini, S.; Fox, M.; and Long, D. 2014. Planning the behaviour of low-cost quadcopters for surveillance missions. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Bürkle, A.; Segor, F.; and Kollmann, M. 2011. Towards autonomous micro UAV swarms. *J. Intell. Robot. Syst.* 61(1-4):339–353.

Chanel, C. C.; Teichteil-Knigsbuch, F.; and Lesire, C. 2013. Multi-target detection and recognition by UAVs using online POMDPs. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

De Nardi, R., and Holland, O. 2008. Coevolutionary modelling of a miniature rotorcraft. In *10th International Conference on Intelligent Autonomous Systems (IAS10)*, 364 – 373.

De Nardi, R. 2013. The QRSim Quadrotors Simulator. Technical Report RN/13/08, University College London.

Franchi, A.; Masone, C.; Grabe, V.; Ryll, M.; Bülthoff, H. H.; and Giordano, P. R. 2012. Modeling and control of UAV bearing-formations with bilateral high-level steering. *Int. J. Robot. Res.* 0278364912462493.

Gómez, V.; Kappen, H. J.; Peters, J.; and Neumann, G. 2014. Policy search for path integral control. In *European Conf. on Machine Learning & Knowledge Discovery in Databases*, 482–497.

Guerrero, J., and Lozano, R. 2012. *Flight Formation Control*. John Wiley & Sons.

Hauert, S.; Leven, S.; Varga, M.; Ruini, F.; Cangelosi, A.; Zufferey, J.-C.; and Floreano, D. 2011. Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. In *Intelligent Robots and Systems (IROS)*, 5015–5020.

Hoffmann, G. M.; Huang, H.; Waslander, S. L.; and Tomlin, C. J. 2011. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control. Eng. Pract.* 19(9):1023 – 1036.

Horowitz, M. B.; Damle, A.; and Burdick, J. W. 2014. Linear Hamilton Jacobi Bellman equations in high dimensions. *arXiv preprint arXiv:1404.1089*.

How, J.; Bethke, B.; Frank, A.; Dale, D.; and Vian, J. 2008. Real-time indoor autonomous vehicle test environment. *IEEE Contr. Syst. Mag.* 28(2):51–64.

Kappen, H. J.; Gómez, V.; and Opper, M. 2012. Optimal control as a graphical model inference problem. *Mach. Learn.* 87:159–182.

Kappen, H. J. 2005. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment* 2005(11):P11011.

Kendoul, F. 2012. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *J. Field Robot.* 29(2):315–378.

Kinjo, K.; Uchibe, E.; and Doya, K. 2013. Evaluation of linearly solvable Markov decision process with dynamic model learning in a mobile robot navigation task. *Front. Neurorobot.* 7:1–13.

Kushleyev, A.; Mellinger, D.; Powers, C.; and Kumar, V. 2013. Towards a swarm of agile micro quadrotors. *Auton. Robot.* 35(4):287–300.

Mahony, R.; Kumar, V.; and Corke, P. 2012. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics & Automation Magazine* 20–32.

Meyer, J.; Sendobry, A.; Kohlbrecher, S.; and Klingauf, U. 2012. Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo. *Lecture Notes in Computer Science* 7628:400–411.

Ono, M.; Williams, B. C.; and Blackmore, L. 2013. Probabilistic planning for continuous dynamic systems under bounded risk. *J. Artif. Int. Res.* 46(1):511–577.

Quintero, S.; Collins, G.; and Hespanha, J. 2013. Flocking with fixed-wing UAVs for distributed sensing: A stochastic optimal control approach. In *American Control Conference (ACC)*, 2025–2031.

Rawlik, K.; Toussaint, M.; and Vijayakumar, S. 2013. Path integral control by reproducing kernel Hilbert space embedding. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 1628–1634. AAAI Press.

Reynolds, C. W. 1987. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.* 21(4):25–34.

Rombokas, E.; Theodorou, E.; Malhotra, M.; Todorov, E.; and Matsuoka, Y. 2012. Tendon-driven control of biomechanical and robotic systems: A path integral reinforcement learning approach. In *International Conference on Robotics and Automation*, 208–214.

Shim, D. H.; Kim, H. J.; and Sastry, S. 2003. Decentralized nonlinear model predictive control of multiple flying robots. In *IEEE conference on Decision and control (CDC)*, volume 4, 3621–3626.

Stulp, F., and Sigaud, O. 2012. Path integral policy improvement with covariance matrix adaptation. In *International Conference Machine Learning (ICML)*.

Symington, A.; De Nardi, R.; Julier, S.; and Hailes, S. 2014. Simulating quadrotor UAVs in outdoor scenarios. In *Intelligent Robots and Systems (IROS)*, 3382–3388.

Theodorou, E.; Buchli, J.; and Schaal, S. 2010. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.* 11:3137–3181.

Thijssen, S., and Kappen, H. J. 2015. Path integral control and state-dependent feedback. *Phys. Rev. E* 91:032104.

Todorov, E., and Li, W. 2005. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference (ACC)*, 300–306 vol. 1. IEEE.

Todorov, E. 2006. Linearly-solvable Markov decision problems. In *Advances in neural information processing systems (NIPS)*, 1369–1376.

Turpin, M.; Michael, N.; and Kumar, V. 2012. Decentralized formation control with variable shapes for aerial robots. In *International Conference on Robotics and Automation (ICRA)*, 23–30.

Van Den Broek, B.; Wiegerinck, W.; and Kappen, H. J. 2008. Graphical model inference in optimal control of stochastic multi-agent systems. *J. Artif. Intell. Res.* 32:95–122.

Vásárhelyi, G.; Virágh, C.; Somorjai, G.; Tarcai, N.; Szorenyi, T.; Nepusz, T.; and Vicsek, T. 2014. Outdoor flocking and formation flight with autonomous aerial robots. In *Intelligent Robots and Systems (IROS)*, 3866–3873.

Williams, G.; Rombokas, E.; and Daniel, T. 2014. GPU based path integral control with learned dynamics. In *Autonomously Learning Robots - NIPS Workshop*.

Yu, B.; Dong, X.; Shi, Z.; and Zhong, Y. 2013. Formation control for quadrotor swarm systems: Algorithms and experiments. In *Chinese Control Conference (CCC)*, 7099–7104.