A Literature Review of Learning and Optimization Methods Applied to Quadrotor Control

Ted Xiao

Spring 2016

Abstract

Various methods of applying learning and optimization methods to quadrotor control have been explored in the past years. The challenges faced in quadrotor control systems represent those faced in other scenarios: high dimensional input data, complex feedback control systems, and sensitive dynamic trajectories. Control approaches vary depending on the constraints and goals of individual scenarios, so understanding the current state of research requires an overview of a wide spectrum of literature. This literature review surveys existing work in this field at a broad level, and then examines two specific approaches in real-time control scenarios: learning-based model predictive control (LBMPC) and reachability-based safe learning.

Contents

List of Figures

List of Tables

1 Introduction

Quadrotor helicopters (UAVs with 4 rotors) are a very popular platform for research in many fields related to control systems and robotics. From path-planning to trajectory control to multi-agent systems to feedback control, quadrotors provide a testbed for not just direct control but also higher level theory. In real world applications, quadrotors are seeing increasing prominence as well, from commercial drones to extensive use in security and surveillance ([1], [2]). As such, quadrotors have been a classic choice for researchers in control and robotics.

Various methods of applying learning and optimization methods to quadrotor control have been explored in the past years ([3], [4]). Results and lessons from applying these methods to quadrotors are useful in a plethora of different applications beyond UAV control, such as power control in HVAC systems [5]. The challenges faced in quadrotor control systems represent different aspects faced in other scenarios: high dimensional input data, complex feedback control systems, and sensitive dynamic trajectories.

Problems usually involve learning system dynamics, and applying controls based on learned models. Research often also encompasses secondary goals, which may be completing a task or generating a trajectory. Advances in machine learning and reinforcement learning methods have been applied to these sorts of formulations ([3], [6]).

The goal of this literature review is to survey these numerous approaches towards the large field of quadrotor control, and to focus on comparing two approaches for real-time control in scenarios with safety constraints: learning-based model predictive control (LBMPC) and reachability-based safe learning.

2 Background on Quadrotor Dynamics and Notation

A quadrotor flies through thrust generated from lift forces produced by controlling the speeds of 4 fixed-angle rotors. A total thrust equal to the weight of the system stabilizes the system in hover. To change angle with regards to the axis, differential thrust is applied to a combination of rotors to roll about specified axis [7].

Clearly understanding quadrotor dynamics is crucial to further extrapolation to quadrotor control. While papers use different notations and dynamics models[1], the core components of all quadrotor models are the same. In models, sufficient state is necessary to uniquely describe any feasible quadrotor position.

While there are always 4 inputs into the system (thrust for each rotor), output states requires not only the standard 3D Cartesian coordinates, but also the Euler angles of pitch, yaw, and roll. These are illustrated in Figure 1, along with thrust.

[1]Many experiments will use a subset of the variables described. For example, experiments that assume a near-hover state may use a simplified model [8]. Other experiments may hold yaw constant [9]

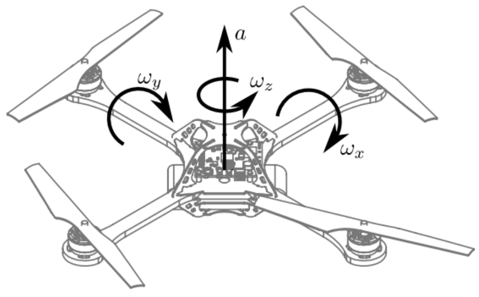


Figure 1: Pitch, Yaw, Roll, and Thrust

In addition to static variables that describe the snapshot position of the quadrotor, additional variables also describe the time derivatives of each measure to capture the velocity of the state. The notation for the time derivative of a variable x is $\dot{x} = \frac{\delta x}{\delta t}$. In all, there are 12 variables:

$$s = (x, \dot{x}, \omega_x, \dot{\omega_x}, y, \dot{y}, \omega_y, \dot{\omega_y}, z, \dot{z}, \omega_z, \dot{\omega_z})^T \in \mathbb{R}^{12}$$

Symbol	Alternate Symbol	Meaning
x	x_1	Position along X-axis
y	x_2	Position along Y-axis
z	x_3	Position along Z-axis
ω_x	ϕ	Roll
ω_y	θ	Pitch
ω_z	ψ	Yaw

Table 1: Dynamics Symbol Definitions

In Figure 1, a represents thrust, which is the relative net force pushing perpendicular from the quadrotor (up from the quadrotor's perspective). Thrust can be described with a combination of symbols shown above [1]. Also, note that the symbols and alternate symbols in Table 1 are used interchangeably depending on context.

In experiments, state is measured through some subset of the above 12 variables. The mechanical system that runs the quadrotor is able to adjust rotor speeds for each of the four motors to affect the trajectory of the quadrotor. However, experimental control inputs usually do not control the motor speeds exactly; instead, inputs describe the net effect on the state that the motors should provide [10]. In particular, control inputs vary depending the specific model the experiment uses to track the quadrotor. These individual models will be described later in the contexts of their experiments. However, in general, the format of the transfer function to update state x_t to state x_{t+1} is:

$$x_{t+1} = Ax_t + Bu_t$$

Various approaches will alter this update in cases with noise or disturbance.

3 Quadrotor Control Approaches

While there are enough results in current quadrotor control research to fill several textbooks, this review will focus on surveying control approaches in a concise but representative fashion. Approaches are roughly outlined in chronological order of development and publication, but are mainly ordered in a way as to set the stage for the on-line control methods of LBMPC and reachability-bsed safe learning.

3.1 Proportional-Integral-Derivative Controller (PID)

Proportional-Integral-Derivative Controllers (PID) are one of the simplest classical controllers seen in many textbook applications. PID controller are control loop feedback mechanisms that directly adjust control values with a closed-form formula based on derivative, integral, and proportional gains.

In the setting of quadrotor control, a PID formulation were examined in [11]. Beginning with an analysis of physical and mechanical properties of quadrotors, the paper described an implementation of a PID control. The model inputs were described as follows:

$$\begin{aligned} U_1 &= (Th_1 + Th_2 + Th_3 + Th_4)/m \\ U_2 &= l(-Th_1 - Th_2 + Th_3 - Th_4)/I_1 \\ U_3 &= l(-Th_1 + Th_2 + Th_3 - Th_4)/I_2 \\ U_4 &= C(Th_1 + Th_2 + Th_3 + Th_4)/I_3 \end{aligned} \tag{1}$$

Where Th_i's are thrusts generated by four rotors and can be considered as the real control inputs to the system, C the force to moment scaling factor, and I_i's are the moment of inertia with respect to the axes.

After some simplification, the model was formulated into subsytems as follows:

$$\begin{bmatrix} \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} u_1 \cos\phi \cos\psi - g \\ u_4 \end{bmatrix} + \begin{bmatrix} -K_3 \dot{z}/m \\ -K_6 \dot{\phi}/I_3 \end{bmatrix}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} u_1 \cos\phi & u_1 \sin\phi \\ u_1 \sin\phi & -u_1 \cos\phi \end{bmatrix} \begin{bmatrix} \sin\theta \cos\psi \sin\psi \end{bmatrix} + \begin{bmatrix} -K_1 \dot{x}/m \\ -K_2 \dot{y}/m \end{bmatrix}$$

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} -IK_4 \dot{\theta}/I_1 \\ -IK_5 \dot{\psi}/I_2 \end{bmatrix}$$

Classical PID control was then applied to this model using the following parameters, in Table 2.

Performance for the PID controller is explored in depth in [11]. In general, PID controls have been found to struggle with aggressive maneuvers, trajectory tracking, and robustness [12]. Relevant to this review is PID's relative performance towards LQR and other methods, which will be discussed in further detail in the next section.

3.2 Linear-Quadratic Regulator (LQR)

A Linear-Quadratic Regular (LQR) is a feedback controller based on the solution to the Linear-Quadratic-Gaussian Problem. This review will focus on approaches outlined in [12] and [13].

The standard dynamics model is assumed for the quadrotor. For some state input $x \in \mathbb{R}^6$ and output $\dot{x} \in \mathbb{R}^6$, we have the transfer function

$$\dot{x} = Ax + Bu$$

Parameter	Value	Unit
I_1	1.25	N_s^2/rad
I_2	1.25	N_s^2/rad
I_3	2.5	N_s^2/rad
K_1	0.010	N_s^2/m
K_2	0.010	N_s^2/m
K_3	0.010	N_s^2/m
K_4	0.012	N_s^2/rad
K_5	0.012	N_s^2/rad
K_6	0.012	N_s^2/rad
m	2	kg
I	0.2	m
G	9.8	m/s^2

Table 2: PID Parameters and Initial Condition

$$A \in \mathbb{R}^{6x6}, B \in \mathbb{R}^{6} \text{ determined empirically}$$

The LQR controller then seeks to minimize control input u for the quadratic cost function

$$\begin{aligned} J &= \int_0^\infty (x^T Q x + u^T R u) dt \\ u &= -Kx \end{aligned} \tag{2}$$

Q and R are weighting matrices that are computed to minimize the cost function J. Coefficients of Q limit the amplitude of the state variables, and the coefficients of R limit the amplitude of the inputs. The larger these coefficients, the smaller the state/input variables. Usually, it is observed that Q and R are diagonal matrices.

The results of the implemented LQR controller are expanded in depth in [12] and [13]. At a high level, when comparing PID, LQR, and LQR-PID control schemes, it seems that LQR achieves faster performance than PID in terms of simulated step responses when applying LQR to tune the PID controller. LQR controllers have been found to be fairly robust and produce a low steady state error, but have large transition delays and when using many feedback gains (as large as the input dimension), the control does not work well for scenarios requiring fast parameter updates or with incomplete state information. As such, LQR controllers are often used in experiments near hover, but may not perform as robustly in aggressive maneuvers.

As an extension, in combination with a Linear Quadratic Estimator (LQE) and Kalman Filter, the LQR algorithm transforms into the Linear Quadratic Gaussian (LQG), which is able to be implemented against noisy or incomplete state information. The LQG obtained good results for near-hover stabilization ([7], [14]).

3.3 Reinforcement and Apprenticeship Learning

Techniques from robotics have been successfully applied to helicopter control, particularly in the domain of aggressive maneuvering and trajectory tracking. While the dynamics of a helicopter system are different from quadrotor dynamics, the models are similar enough that quadrotor control approaches are almost identical to helicopter control. Thus, in the scope of this review, insights from these research approaches for helicopter control are treated as directly applying to quadrotor control.

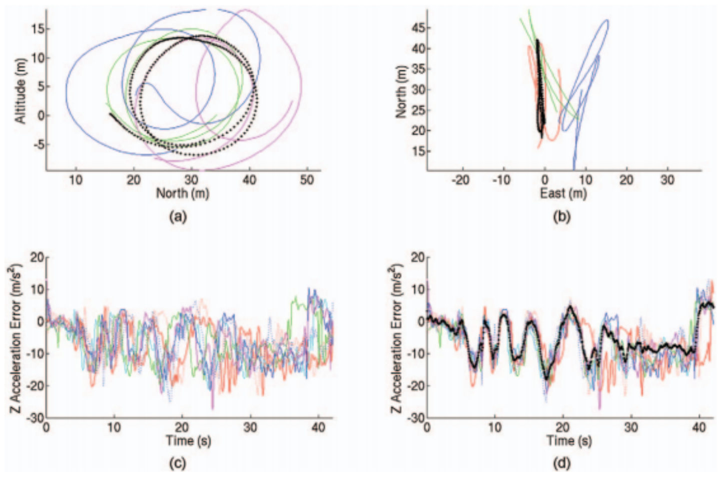


Figure 2: Colored lines: demonstrations. Black dotted line: trajectory inferred by our algorithm.

In [6] and [3], Reinforcement Learning (RL) and Apprenticeship Learning (Inverse Reinforcement Learning) methods are applied to teach a helicopter the controllers for advanced aerobatic maneuvers.

While RL and Apprenticeship methods are not directly comparable to the other control algorithms discussed, the novel approaches demonstrated in these approaches were able to apply proven Artificial Intelligence (AI) methods to actually learn on top of existing control algorithms. So while RL and Apprenticeship learning are not actual control algorithms, they are able to learn optimal controllers through learning parameters, reward functions, and then learning the trajectories through applying variations of LQR [2].

These approaches are surveyed because they provide an insight to the power of AI methods when used on top of existing control algorithms. In this case, these methods were able to learn controllers required for very complex aerobatic maneuvers with a high degree of accuracy (see Figure 2.

4 Robust, Safe, and Real-time Quadrotor Control

While many quadrotor control methods exist for a variety of scenarios, many real-world cases are very different from laboratory environments. Many algorithms do not guarantee rigorous safety conditions while learning or optimizing control, and others may not be robust enough to operate in noisy or partially unknown states. Thus despite academic successes of many models, their must be limited to scenarios in which either safety and robustness are not critical, or a large number of trials can be conducted before deployment to verify the systems safety. Two approaches, Linear-Based Model Predictive Control and Reachability-based Safe Learning, have shown success in providing online robustness and safety guarantees.

[2]The approach used was a Gauss-Newton LQR, or differential dynamic programming. It includes one additional higher-order term in the approximation.

4.1 Learning-Based Model Predictive Control (LBMPC)

Learning-Based Model Predictive Control (LBMPC) is a model-based control strategy that combines model predictive control with online model updates to improve the learned model, instead of solely relying on a nominal model. A major drawback of many of the surveyed control methods is that learning safely is not inherently built into the model, if learning is at all implemented. In LBMPC, learning is an integral part of the model, while the nominal model is used to guarantee safety constraints.

In [8], [15], and [16], the LBMPC approach is introduced and then applied to live quadrotor experiments with success.

In the experiment, a simplified quadrotor dynamics model is used:

$$\begin{aligned} x &= (x_1, \dot{x_1}, \theta_1, \dot{\theta_1}, x_2, \dot{x_2}, \theta_2, \dot{\theta_2}, x_3, \dot{x_3})^T \in \mathbb{R}^{10} \\ x_{t+1} &= Ax_t + Bu_t + k + h(x_t, u_t) \\ y &= Cx + \epsilon \\ F_{\mathcal{N}} &= Ax_t + Bu_t + k \end{aligned} \tag{3}$$

The simplified dynamics model with $\mathbb{R}^{10}$ vectors is used because the system is maintained around hover. Yaw is assumed fixed. In additional, the pitch and roll dynamics are assumed to be decoupled and identical. The control input at each timestep is a vector $u = (u_1, u_2, u_3 \in \mathbb{R}^3$, representing the commanded pitch, roll, and thrust. From these input commands, a full set of motor commands and state updates can be computed. The nominal model $F_{\mathcal{N}}$ is the initial prior model of the system dynamics, which remains unchanged throughout. The $h(x_t, u_t)$ term is then the unmodeled dynamics of the system, where LBMPC is able to learn information to update the holistic model. The model guarantees safety under the assumption that the uncertainty is bounded within a convex, compact polytope.

The LBMPC controller is composed of two parts: (i) state estimation and learning the unmodeled dynamics and (ii) closed loop control based on the updated model.

The state estimation and learning portion is the "Learning" part of LBMPC. An oracle $\mathcal{O}_m(x_t, u_t) = F(\beta)x_t + H(\beta)u_t + z(\beta)$ is used as a black box in the context of the state update under learned dynamics

$$x_{t+1} = F_{\mathcal{O}} := F_{\mathcal{N}}(x_t, u_t) + \mathcal{O}_m(x_t, u_t) = (A + F)x_t + (B + H)u_t + k + z$$

The oracle is modeled as a linear combination of entries of the nominal dynamics matrix; the entries are weighted by parameters β, which is the vector that is updated at each iteration of state estimation and learning.

Then, the closed-loop control is the solution of a QP. Out of the N controls outputted as the optimal solution for the N-step time horizon, only the first control $\hat{u}_m$ is applied. Then, in the next iteration through the loop, the entire process of generating N new controls is applied with the updated F, H, z matrices.

Essentially, at each time step, the oracle is updated to learn dynamics of the system based on observations, and then a standard MPC QP problem is solved to generate a control input (computed for optimality at a fixed receding horizon).

Multiple experiments are used to demonstrate the performance of this LBMPC algorithm on quadrotor control:

1. Learning the ground effect
   The ground effect is an aerodynamic effect that increasing lift with close to the ground. The LBMPC algorithm was able to learn the ground effect and maintain hover when the quadrotor descended close enough to the ground to experience the ground effect. In comparison, a standard linear MPC algorithm (nominal model only) was not able to maintain hover.

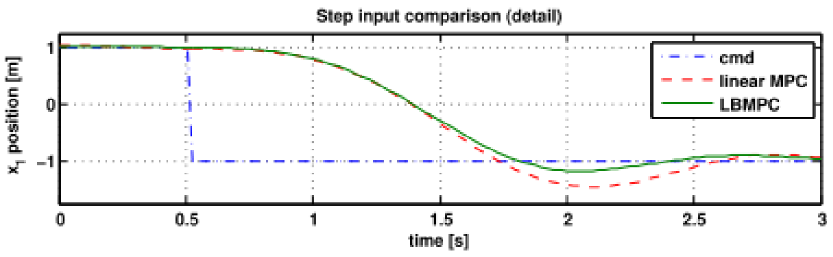


Figure 3: Overshoot of Nominal Model only compared with the LBMPC Learned Model

2. Decreased overshoot in step response
   A linear MPC algorithm (nominal model only) was again compared with LBMPC in the setting of a quadrotor oscillating along the x_1 axis between two points. The metric of success was amount of overshoot in this step response. As seen in Figure 3, the LBMPC model experienced 62% less overshoot than the linear MPC model.
3. Robustness to "incorrect learning"
   This experiment challenged the safety constraints of the quadrotor by adding copious amounts of noise into the oracle, causing the control system to learn incorrectly. However, the nominal model in LBMPC is able to maintain stability.
4. Precise maneuvering: ball catching
   The ball catching experiment applied LBMPC control to a task of catching a hand-thrown ball. The task is challenging because the quadrotor must quickly and accurately arrive at the ball's protected location, even as that location is being updated. The ball's trajectory was computed by a separate physics model through observations, and then fed to the quadrotor. The quadrotor was able to catch the ball 90% of the time, showing that the LBMPC method's practical computational speed matches the speed of linear MPC. Being able to run in real-time in dynamically changing scenarios is important for on-line control algorithms, especially in scenarios involving safety and learning.

LBMPC's split of the model into a nominal and learned model enables safety guarantees, an updated learned model, and the computational runtime of linear MPC.

4.2 Reachability-based Safe Learning

As discussed earlier in the RL/Apprenticeship Learning methods, many advances in AI and Machine Learning techniques have become popular tools in autonomous systems. However, many algorithms only guarantee performance asymptotically. In [9], [17], and [18], Reachability-based Safe Learning Methods attempt to provide safety guarantees to any online-learning algorithm.

The main idea behind Guaranteed Safe Online Learning via Reachability (GSOLR) is a sort of dual-mode operation. When the system is determined to be reaching a state that may lead to unsafe behavior, an optimal safety controller determines the control to keep the quadrotor in a safe space. When the quadrotor is not at risk of breaking safety constraints, any control algorithm may be implemented (even one without safety guarantees).

Through advances in Hamilton-Jacobi-Isaacs (HJI) reachability, the unsafe backwards reachable set can be computed to determine regions of the state space from which the system can be guaranteed to remain safe. Formally, the unsafe backwards reachable set $Pre_\tau(\mathcal{K})$ is defined as the set of all states x such that, for any control input u, the disturbance d can drive

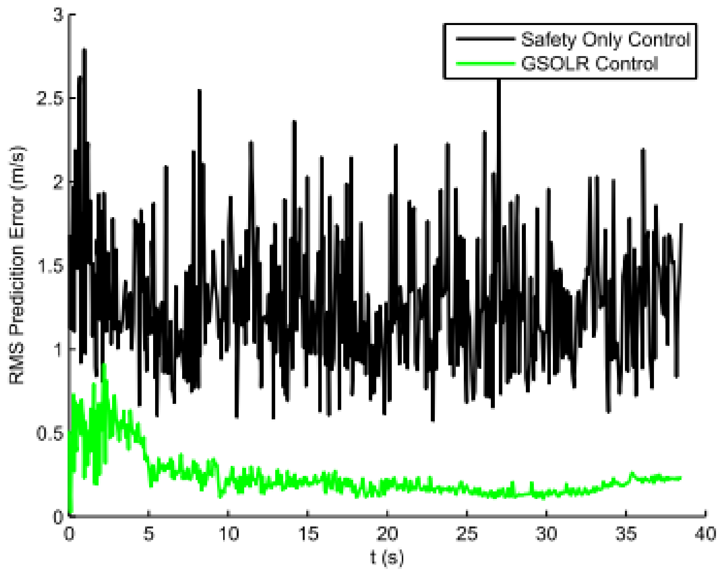


Figure 4: Prediction error target's velocity of GSOLR vs. Safety-only

the system state into an unsafe region within τ time steps. $Pre_\tau(\mathcal{K})$ can be calculated through a dynamic game formulation, which is solved through a HJI partial differential equation

$$\frac{\delta J}{\delta t} + \min 0, H^*(x, \frac{\delta J}{\delta x}) = 0, J(x, \tau) = l(x)$$

$$H^*(x,p) = \max_{u\in\mathcal{U}} \min_{d\in\mathcal{D}} p^T f(x,u,d)$$

Then, the solution to the optimal Hamiltonian H^* provides the optimal control actions $u*$ which must be taken to maintain safety when x is on the border of $Pre_\tau(\mathcal{K})$. GSOLR is a least-restrictive safe controller in the sense that the safety-based control actions only need to be enforced when the system is on the border of $Pre_\tau(\mathcal{K})$.

The GSOLR algorithm was applied to an experiment of an observer quadrotor learning the dynamics of a target ground vehicle. Details on the system dynamics can be found in [9], but they were fairly similar to models discussed previously in this review.

Safety and learning were put in tension because a higher hovering height for safety resulted in less information gained for learning. Additionally, a learning-only model and a safety-only model were compared with the GSOLR algorithm. As expected, the GSOLR algorithm was able to maintain safety guarantees of the safety-only model while still learning the dynamics model of the observed target.

Some extensions on GSOLR have been made in [18] using Gaussian Processes. The nominal model in GSOLR assumes worst-case disturbances, which reduces the computed safe region and thereby the region where the system can learn. Additionally, GSOLR decouples safety and learning, which could lead to the learning controller to attempt to drive the system into unsafe regions. 'Chattering' behavior was indeed observed, which resulted when the system oscillated rapidly between being pushed toward and then away from the unsafe set.

In [18], additional learning is incorporated by learning the disturbances of the system from data to prevent an overly conservative reachability analysis. Prior disturbance models assumptions are updated based on online data through a Gaussian process model. This approach showed learning improvements over unaltered GSOLR.

4.3 Comparison of LBMPC and GSOLR

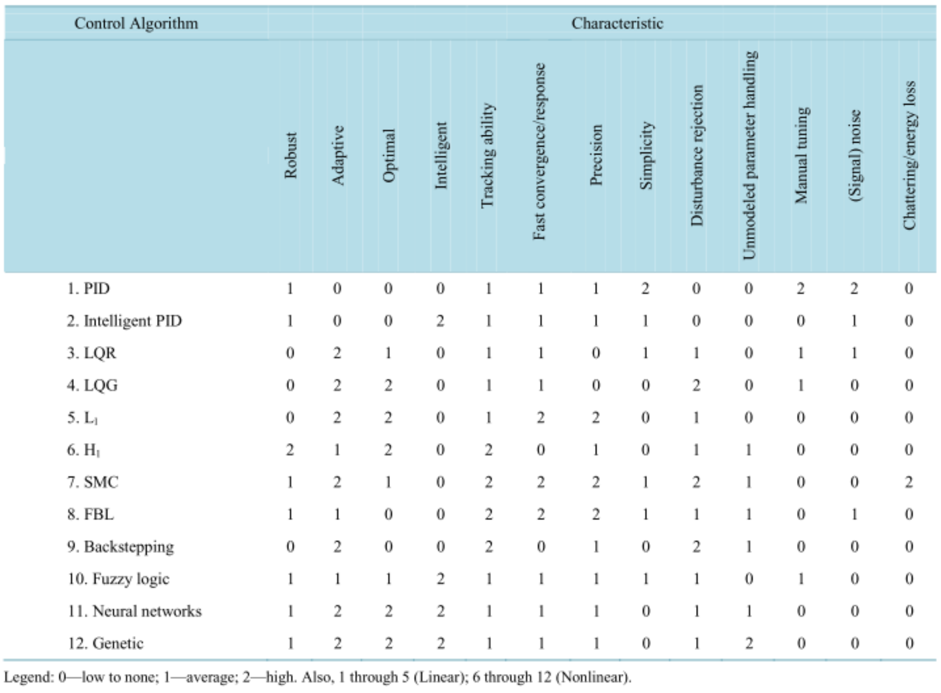
LBMPC and GSOLR are both approaches that seek to address the challenge of providing robust safety guarantees in algorithms that are able to run online. Both approaches are able to achieve empirical success, and are vaguely similar at a high-level because both approaches use a dual approach, with one model guaranteeing safety and one model providing learning and robustness.

However, LBMPC has a specific nominal model that builds in the safety constraints to the control process, whereas GSOLR must switch to the optimal safety control when the system is at risk. Additionally, the computation for the methods is approached through very different means. The HJI computation for the reachable set in GSOLR actually has no analytical solution, and so numerical solvers must be applied to compute the set. After computing the set, however, it can be saved and used for computationally efficient online use. In comparison, the LBMPC calculation has a similar efficiency as linear MPC. On the other hand, LBMPC may not be as general as GSOLR because LBMPC requires a linear model of the system dynamics for its safety guarantees to hold, which is not necessary under GSOLR.

Thus, providing better safety guarantees under LBMPC and improving computational efficiency for GSOLR are areas requiring further research.

5 Conclusion

This review serves as an overview of control methods, a look into robust online safe control methods, and a comparison between LBMPC and reachability-based safe learning. As a survey over existing methods, this review highlights shortcomings of existing approaches and what future novel methods should focus on addressing. Namely, current research has found many successful controls for controlled environments, but in noisy real-world environments many methods fail. Robust and intelligent methods are needed to bring many advances from laboratories to the wider public. A similar survey that focused on a wider variety of general control methods found similar results [7]



Control Algorithm	Characteristic												
	Robust	Adaptive	Optimal	Intelligent	Tracking ability	Fast convergence/response	Precision	Simplicity	Disturbance rejection	Unmodeled parameter handling	Manual tuning	(Signal) noise	Chattering/energy loss
1. PID	1	0	0	0	1	1	1	2	0	0	2	2	0
2. Intelligent PID	1	0	0	2	1	1	1	1	0	0	0	1	0
3. LQR	0	2	1	0	1	1	0	1	1	0	1	1	0
4. LQG	0	2	2	0	1	1	0	0	2	0	1	0	0
5. L_1	0	2	2	0	1	2	2	0	1	0	0	0	0
6. H_1	2	1	2	0	2	0	1	0	1	1	0	0	0
7. SMC	1	2	1	0	2	2	2	1	2	1	0	0	2
8. FBL	1	1	0	0	2	2	2	1	1	1	0	1	0
9. Backstepping	0	2	0	0	2	0	1	0	2	1	0	0	0
10. Fuzzy logic	1	1	1	2	1	1	1	1	1	0	1	0	0
11. Neural networks	1	2	2	2	1	1	1	0	1	1	0	0	0
12. Genetic	1	2	2	2	1	1	1	0	1	2	0	0	0

Legend: 0—low to none; 1—average; 2—high. Also, 1 through 5 (Linear); 6 through 12 (Nonlinear).

Figure 5: Comparison of Control Methods by A. Zulu and S. John

Finally, in the specific focus of robust, online safe control methods, LBMPC and GSOLR have both shown recent success in quadrotor control. Further research may seek to provide algorithmic efficiency and more general safety guarantees.

References

[1] Gabriel M. Hoffmann, Haomiao Huang, Steven L. Waslander, and Claire J. Tomlin. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, 19(9):1023–1036, 2011.

[2] Mo Chen, Jaime F Fisac, Shankar Sastry, and Claire J Tomlin. Safe Sequential Path Planning of Multi-Vehicle Systems via Double-Obstacle Hamilton-Jacobi-Isaacs Variational Inequality. *Proceedings of the 14th European Control Conference (to appear)*, pages 3309–3314, 2015.

[3] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous Helicopter Aerobatics through Apprenticeship Learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[4] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3277–3282, 2009.

[5] Anil Aswani, Neal Master, Jay Taneja, Andrew Krioukov, David Culler, and Claire Tomlin. Energy Efficient Building HVAC Control Using Hybrid System LBMPC. 2012.

[6] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. *Education*, 19:1, 2007.

[7] Andrew Zulu and Samuel John. A Review of Control Algorithms for Autonomous Quadrotors. *Open Journal of Applied Sciences*, 4(December):547–556, 2014.

[8] Patrick Bouffard, Anil Aswani, and Claire Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. *2012 IEEE International Conference on Robotics and Automation*, pages 279–284, 2012.

[9] Jeremy H. Gillula and Claire J. Tomlin. Guaranteed safe online learning via reachability: Tracking a ground target using a quadrotor. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2723–2730, 2012.

[10] Jun Li and Yuntang Li. Dynamic analysis and PID control for a quadrotor. *2011 IEEE International Conference on Mechatronics and Automation*, pages 573–578, 2011.

[11] A. Zul Azfar and D. Hazry. A simple approach on implementing IMU sensor fusion in PID controller for stabilizing quadrotor flight control. In *Proceedings - 2011 IEEE 7th International Colloquium on Signal Processing and Its Applications, CSPA 2011*, pages 28–32, 2011.

[12] C Balas. Modelling and Linear Control of a Quadrotor. *Interface*, 2007.

[13] Lucas M. Argentim, Willian C. Rezende, Paulo E. Santos, and Renato A. Aguiar. PID, LQR and LQR-PID on a quadcopter platform. *2013 International Conference on Informatics, Electronics and Vision, ICIEV 2013*, 2013.

[14] Ly Dat Minh and Cheolkeun Ha. Modeling and control of quadrotor MAV using vision-based measurement. *2010 International Forum on Strategic Technology, IFOST 2010*, pages 70–75, 2010.

[15] Anil Aswani, Patrick Bouffard, and Claire Tomlin. Extensions of Learning-Based Model Predictive Control for Real-Time Application to a Quadrotor Helicopter. *Proc American Control Conference ACC to appear*, pages 4661–4666, 2012.

[16] Anil Aswani, Humberto Gonzalez, S. Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[17] Jh Gillula and Cj Tomlin. Reducing Conservativeness in Safety Guarantees by Learning Disturbances Online: Iterated Guaranteed Safe Online Learning. *Robotics: Science and Systems*, 2012.

[18] A. K. Akametalu, S. Kaynama, J. F. Fisac, M. N. Zeilinger, J. H. Gillula, and C. J. Tomlin. Reachability-Based Safe Learning with Gaussian Processes. *IEEE Conference on Decision and Control*, pages 1424–1431, 2014.