Proceedings of the 37th Chinese Control Conference
July 25-27, 2018, Wuhan, China

Hybrid Path Planning of A Quadrotor UAV Based on Q-Learning Algorithm

Tianze Zhang, Xin Huo, Songlin Chen, Baoqing Yang, Guojiang Zhang
Control and Simulation Center, Harbin Institute of Technology, Harbin 150080, P. R. China
E-mail: zhangtianze_jlu@163.com, huoxin@hit.edu.cn

Abstract: The hybrid methodology is an emerging technology for control solution of nonlinear control systems with infinite states, moreover, by utilizing the approach the system can be transformed to a finite one based on discrete abstractions. In this paper, the problem of path planning of a quadrotor unmanned aerial vehicle (UAV) is investigated in the framework of hybrid methodology. With the kinematics model of the Quadrotor UAV and the abstraction of the environment in the form of grid world, the design procedure is presented by utilizing the Q-learning algorithm, which is one of the reinforcement method. In this process, an optimal or suboptimal safe flight trajectory will be obtained by learning constantly to maximize the reward. Matlab software is used for computation, and the effectiveness of the proposed method is illustrated by a typical example.

Key Words: Hybrid System, Path Planning, Q-Learning, Quadrotor UAV

1 Introduction

Recent advances made in the field of autonomous vehicles suggest that, in a near future, the Unmanned Air Vehicles (UAVs) will be more and more deployed in order to achieve various missions such as mapping, surveillance, intelligence, tracking operations or search and rescue. Applications of UAV have already been expanded to civil domains such as agriculture, aerial photography and package delivery [1].

Over the past several decades, autonomous path-planning techniques have become significantly and increasingly important to the UAV, as the conventional remotely piloted techniques cannot offer sufficient accuracy and perfect timing for complex missions nowadays [2, 3]. Path planning is a challenging issue for UAVs, aiming at finding an optimal or suboptimal safe flight trajectory within a proper time, while accomplishing the prearranged tasks and avoiding the obstacles. Geometry based methods [4] provide a closed form solution but for a simplified UAV model. Methods based on temporal logic hybrid system [5] assume that all obstacle aircraft observe the proposed temporal logic. The idea proposed by [6] is to synthesize a global integrated task and motion plan through composing simple local moves and actions, and to achieve its performance guarantee through modular and incremental verifications. They model the controllers, which are abstracted in Counter Linear Temporal Logic over Constraint System CLTLB($\mathcal{D}$), and verify their safety through formulating them as Differential Dynamic Logic (dL) formula. Nonstandard search patterns that have a random trajectory are easier to program in cooperative search and coverage scenario. Some algorithms used to generate the nonstandard search patterns are A^* and traveling salesmen, which are heuristic techniques [7]. The path planning problem is not restricted to UAVs. In reality, it is much more consumingly investigated in the field of robotics, where path planning is normally considered as motion planning [8]. A self-adaptive learning particle swarm optimization (SLPSO) with different learning strategies is proposed to address path planning problem in [9]. The path planning problem is transformed into a minimisation multi-objective optimization problem and a novel self-adaptive learning mechanism is developed to select the most suitable search strategies at different stages of the optimization process adaptively and the new bound violation handling schemes is utilized to restrict the position and velocity of each particle.

In the wake of the development of machine learning, more and more methodologies on the basis of deep learning (DL) and reinforcement learning (RL) are promoted to solve path planning problem. [10] proposes a learning architecture, that is able to do reinforcement learning based on raw visual input data, and it has accomplished a slot car racer task. [11] proposes an end-to-end deep reinforcement learning (DRL) method towards cognitive exploration in an unfamiliar environment by taking depth image as the input and control commands as the output. Based on the Deep Q-Network framework [12], the raw depth image is taken as the only input to estimate the Q values corresponding to all moving commands. The training of the network weights is end-to-end. [13] presentes a CNNs-based data-driven end-to-end motion planning approach for a robotic platform. Given local laser range findings and a relative target position, the approach is able to compute the required steering commands for a differential drive platform.

In this paper, the hybrid path planning method of a quadrotor unmanned aerial vehicle is investigated by using the notions of abstraction, which is illustrated in the well-developed hybrid control theory [14]. The approach is an emerging technology for nonlinear control systems, which is based on symbolic models or discrete abstractions. A hybrid system combines the state machine models of discrete control with differential-equation models of continuous dynamics. This research discipline brings together researchers from software engineering and formal methods, control theory, and applications such as robotics and systems biology. [17] presents an abstraction and refinement methodology for the automated controller synthesis to enforce general predefined specifications and a path planning problem for an

This work was supported by the National Natural Science Foundation of China under Grants 61773138 and 61203191, China Postdoctoral Science Foundation under Grant 2015M571415, Heilongjiang Postdoctoral Foundation under Grant LBH-Z14088, and the Fundamental Research Funds for the Central Universities under Grant HIT.NSRIF.201626.

autonomous vehicle is demonstrated.

The rest of the paper is organized as follows. In Section 2, the preliminaries are elaborated, consisting of kinematics model of a quadrotor UAV established in the form of differential equations, the approach to describe the system, and the advantage of reinforcement Learning in the application of path planning. The concrete procedure of environment description and path planning is presented in Section 3, which is the main contribution of this paper. Simulation and analysis of astringency are developed by utilizing matlab in Section 4, which show the validity of the proposed approach. Section 5 concludes the whole paper.

2 Preliminaries

2.1 Kinematics model of a Quadrotor UAV

Usually the control structure of a UAV is hierarchical, in the architecture of which the attitude of the UAV is controlled in the lower level and then, the path planning algorithm is responsible for the motion of the UAV in the higher level of this framework [18]. Consequently, if the generated path respects the dynamic constraints of the UAV motion, the lower level controller of the UAV is able to track it in this structure [19]. Hence, for UAVs, or a large class of autonomous helicopters, it is rational to consider the path planner dynamics of them as a mass point model with the following dynamics:

$$\dot{x} = u \quad x \in \mathbb{R}^2, \quad x \in U \subseteq \mathbb{R}^2 \tag{1}$$

where x is the position of the UAV, u is the velocity, and U is the constraint set. Here, we assume that the UAV is flying in some certain altitude, and the altitude control works well.

2.2 System Description

The objective investigated in this paper consists of discrete logics and continuous variable plant, i.e., the specification used throughout the paper is described in a form of discrete logic formulas and continuous variable dynamics. To bridge the gap, a hierarchical control structure consisting of a discrete planning layer on top of a continuous implementation layer is usually adopted. The basic idea is to obtain an equivalent finite abstraction of the environment.

The effectiveness of the abstraction-based method depends on whether there exists an equivalent finite-state discrete abstracted model for the original system. If a implement in the abstracted model satisfying the specification can be found, thus there must exist corresponding continuous trajectories for the original robotic system satisfying the same specification. Therefore, significant efforts have been devoted to developing the equivalent abstract models, mainly using simulation-, bisimulation- and approximate bisimulation-based abstractions, see details in [14–16] and the references therein.

Among many different mathematical models used to describe dynamical phenomena, we are especially interested in models with states belonging to finite sets, infinite sets, and combinations thereof. By a finite-state system we mean a system described by finitely many states. The finite-state machines used to model digital circuits are one such example. We also consider infinite state systems described by difference or differential equations with solutions evolving in infinite sets such as $\mathbb{R}^2$. Hybrid systems, combining aspects of finite-state and infinite-state systems, consist of another class of systems that can be described by the notion of system adopted in this paper.

Definition 2.1: A system S is a sextuple

$$(X, X_0, U, F, Y, H) \tag{2}$$

consisting of:

- a set of states X;
- a set of initial states X_0;
- a set of inputs U;
- a transition relation $F \subseteq X \times U \times X$;
- a set of outputs Y;
- an output map $H : X \to Y$.

States in X are regarded as internal to the system whereas outputs are externally visible. The set of initial states may be a proper subset $X_0 \subset X$, a fixed initial state $x_0 \in X$, or the whole set of states $X_0 = X$. A system is called finite-state if X is a finite set. Inputs in U can represent choices to be made by a controller, choices to be made by the environment, or they can simply describe the passage of time.

2.3 Reinforcement Learning

As an significant machine learning method, reinforcement learning is an effective way for robotics to learn and make control decision and policies. The learner should discover the actions which correspond to the highest reward through exploring among them. Actions to be utilized will affect the immediate reward as well as the next situation and, through that, all subsequent rewards. The main advantage of reinforcement learning is the completed independence from human labeling. Motivated by the trial-and-error interaction with the environment, the estimation of the action-value function is self-driven by taking the robot states as the input of the model. Conventional reinforcement learning methods improved the controller performances in path-planning of robot-arms and controlling of helicopters [20].

Problems referred above can be solved by finite Markov decision processes (finite MDPs). MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. A basic reinforcement learning framework is shown in Fig. 1.

Beyond the agent(UAV) and the environment, one can identify four main subelements of a reinforcement learning system: a policy, a reward signal, a value function, and, optionally, a model of the environment.

- A *policy* defines the learning agent's way of behaving at a given time. Summarily speaking, a policy maps perceived states to actions to be taken in the environment when in those states.
- A *reward* signal defines the sole goal in a reinforcement learning problem. The reinforcement learning agent interact with the environment on each time step, and the number sent to the agent from environment is

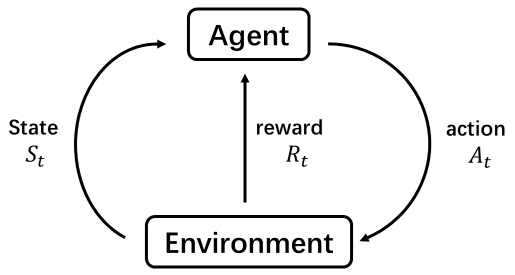

Fig. 1: A basic reinforcement learning framework

called reward. The agent's fundamental objective is to maximize the total reward it receives over the long run. Thus, the reward defines what are the good and bad events for the agent.

- The *value* of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states.
- A *model* of the environment is a process that simulates the behavior of the agent in the environment, or more detailedly, that allows inferences to be made about how the environment will behave.

In this paper, the UAV is considered as an agent, and its flight space can be seen as state space, and it is the action that denotes the movement of the UAV, i.e., which grid it will move to on the next step. The methodology proposed in subsequent sections is a direct reinforcement learning without model.

3 Path Planning Design

3.1 Mission Description

In this paper, we consider a typical two dimension path planning problem, which is described in the sequel.

There are a series of obstacles on the way from the start state to the target region A, the UAV should reach the target region while avoiding the obstacles, as shown in Fig. 2.

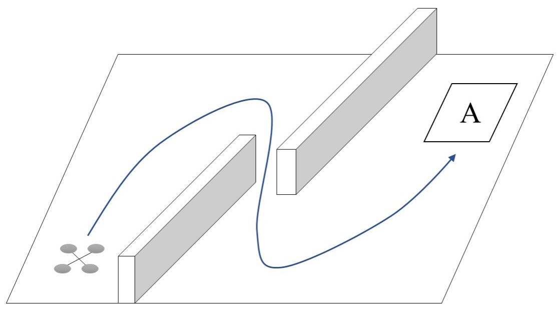

Fig. 2: Mission description

3.2 Finite Abstraction of Environment

As described in the previous section, the quadrotor UAV in this paper is considered to hover at a certain height, i.e., in horizontal plane. An effective method of state space partitioning is necessary, on basis of which we can make a clear description of the environment where the quadrotor UAV moves.

The modeling methods for flight environment of UAV include the geometric modeling method, the unit decomposition modeling and topology method, etc. In this section, we'll achieve a system that is abstracted to a finite model by utilizing a grid-based partitioning method. Considering the area from start point to destination as a square, in which the UAV hovers with infinite number of states including position X position Y. Then, the square is partitioned into a certain number of grids as shown in Fig. 3.

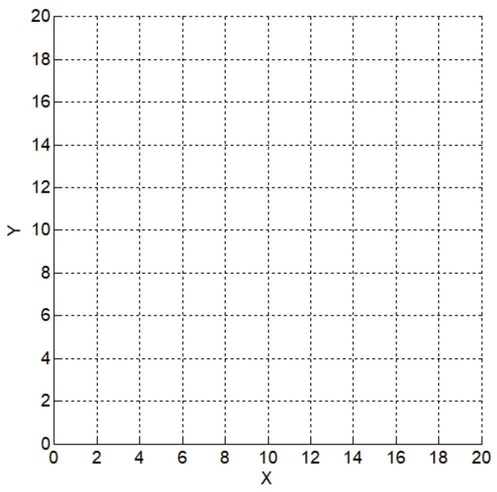

Fig. 3: Finite Abstraction of Environment

Clearly, choosing large partitions reduces the maneuverability of the UAV in the partitioned space. Therefore, it is desired to have smaller partitions. Theoretically we can choose very small partitions for the system with a mass point model, but this increases the computation cost due to the increase in the number of discrete states, and also it may cause collision between two UAVs or obstacles that are in adjacent regions. Therefore, the size of partitions should be bigger than the size of the UAV.

3.3 Q-Learning-based Path Planning

One of the early breakthroughs in reinforcement learning was the development of an off-policy TD control algorithm known as Q-learning, defined by

$$\begin{aligned} Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \times \\ max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \end{aligned} \tag{3}$$

In this case, the learned action-value function, Q, directly approximates q, the optimal action-value function, independent of the policy being followed. This dramatically simplifies the analysis of the algorithm and enabled early convergence proofs. The policy still has an effect in that it determines which state-action pairs are visited and updated. However, all that is required for correct convergence is that all pairs continue to be updated. This is a minimal requirement in the sense that any method guaranteed to find optimal behavior in the general case must require it. Under this assumption and a variant of the usual stochastic approximation conditions on the sequence of step-size parameters, Q has been shown to converge with probability 1 to q.

α is the *learning rate*, determining the rate that the new value covers the old one. If $\alpha = 0$, it means the algorithm does not learn any new values, and $\alpha = 1$ means not saving old values.

γ is a parameter, $0 \leq \gamma \leq 1$, called the *discount factor*. The discount rate determines the present value of future reward. If $\gamma = 0$, it denotes that the algorithm calculates the current reward only without future reward, while $\gamma = 0$ means the current reward and the future one are equally important. In the case $0 < \gamma < 1$, more older the reward is, more importantly it plays in the algorithm.

In this project, a map is constructed between the Q matrix and the flight environment, in other words, one or some elements in the Q matrix correspond to one grid in the environment consisting of grid world. It is assumed that the UAV could move forward, backward, left, right and staying invariantly, i.e., there are five actions for the UAV to choose. So in this case five matrix elements are configured to provide formulations to the q value of one grid, which illustrate the rewards of five actions. Without doubt, the number of actions can also be expended to nine. It's just not the main distribution of this paper.

The whole learning procedure could be denoted as finite Markov decision processes. As we discussed in Section 2, with the description of the environment in form of (2), a Markov process can run well in this framework. The concrete algorithm is shown as followed.

Algorithm 1 Q-learning path planning algorithm

```
Input: $Q(s,a), S$
Require: all $s \in \mathcal{S}$; $a \in \mathcal{A}(s)$, arbitrarily
  $Q(terminal - state, :) = 0$                         ▷ Initialize
  for all step number < max step number do
      Choose $A$ from $S$ using policy derived from $Q$
      (A $\epsilon$-greedy is selected)
      Take action $A$
      Observe $R, S'$
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \times max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$
      if current state == terminal then
          break                         ▷ Repeat until $S$ is terminal
      end if
  end for
```

ϵ-greedy is a basic random policy of reinforcement learning, which could keep balance between exploitation and exploration.

$$\pi(a \mid s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & a = \text{argmax} Q(s,a) \\ \frac{\epsilon}{|A(s)|} & a \neq \text{argmax} Q(s,a) \end{cases} \tag{4}$$

thereinto, exploitation is to maximize the action-value function and exploration means it is possible for the suboptimal actions to explore, namely, interact with environment to obtain new information. It means that the probability of selecting the action corresponding to the maximum action-value function equals to $1 - \epsilon + \frac{\epsilon}{|A(s)|}$. The probability of implement the other actions is $\frac{\epsilon}{|A(s)|}$. $|A(s)|$ denotes the number of actions to be selected at state s.

4 Simulations and Discussion

4.1 Simulation Set-up

The simulation runs in the environment of Matlab R2014, and a GUI interphase is programmed, in which the original position, destination and the obstacles can be set flexibly by clicking the left mouse button. Therefore, it is easy to set different environments to examine the method proposed. The concrete parameter configuration of one examination is shown in the table 1.

Table 1: Parameter configuration

Grid number	20×20
Learning rate	$\alpha = 0.3$
Discount factor	$\gamma = 0.95$
Research parameter	$\epsilon = 0.5$
Max trails times	5000
Max Iteration times	1800
Convergence goal	0.25

4.2 Results and Discussion

In this case, the initial state is selected arbitrarily which is denoted by a blue $\star$, and the target arbitrarily too, which is denoted by a red $*$. The simulation result is shown in Fig. 4, and it can be seen from the figure that, the UAV sets off from the origin region, and avoids the obstacles successfully and reaches the target region eventually. The analysis of convergence is shown in Fig. 5. Moreover, it indicates that the UAV will consume less iterations to arrive at the terminal state with the trail times increasing in Fig. 6.

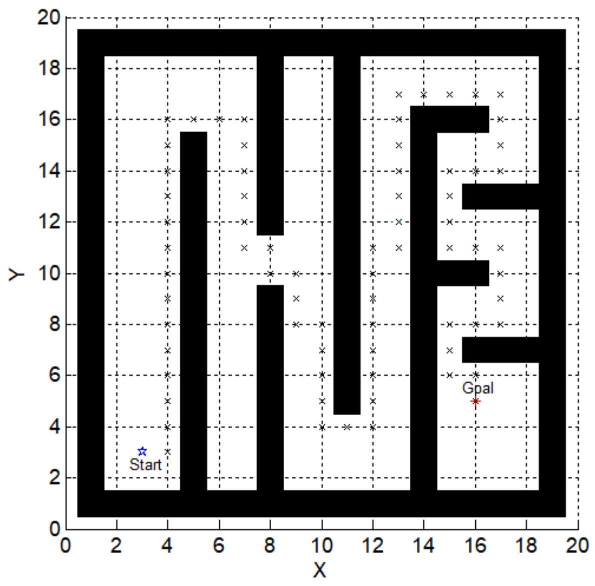

Fig. 4: The simulation of path planning in a maze

In the given mission, the UAV is only considered to hover in a horizontal plane at a certain height. Furthermore, the UAV can only select motion within five actions. In the

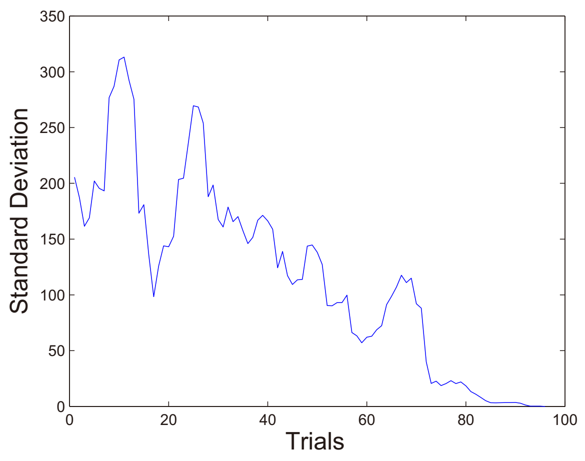

Fig. 5: The analysis of convergence

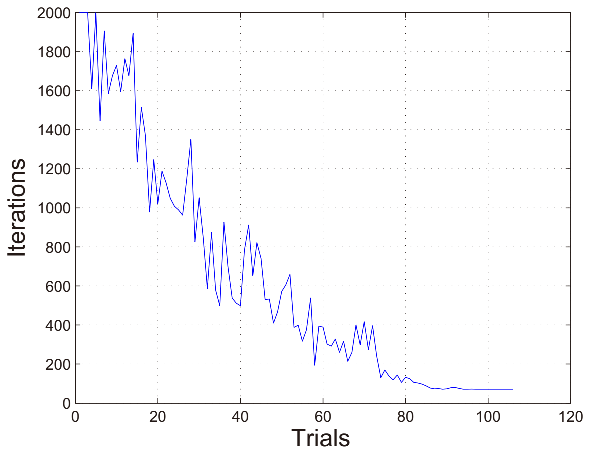

Fig. 6: The iterations needed per trail

follow-up work, we will consider simulating the UAV hovering in three-dimensional space with more actions and then, take a test in practice.

5 Conclusion

The quadrotor UAV is a typical underactuated system, which is also a nonlinear multi-freedom control system. In this paper, a 2D kinematics model of the Quadrotor UAV is simplified in form of a line motion without angular movement. The path planning of the UAV is studied in the framework of hybrid methodology, by the description of which the environment is abstracted to a grid world with finite states. And the concrete path planning algorithm is based on the Q-learning, one of the reinforcement learning approaches. A typical simulation example is illustrated to verify the feasibility of the method. The ϵ-greedy policy is to be modified to improve the efficiency of the methodology. Furthermore, the extension of 2D-planning to 3D-planning and some applications of the proposed approach, including its implementation based on the physical UAV system, should be investigated in the future.

References

[1] Yucong Lin and Srikanth Saripalli, Sampling-Based Path Planning for UAV Collision Avoidance, *IEEE Transactions on Intelligent Transportation Systems*, 2017: 3179-3192.

[2] C. Goerzen, Z. Kong, and B. Mettler, A survey of motion planning algorithms from the perspective of autonomous UAV guidance, *J. Intell. Robot. Syst.*, 2010: 65-100.

[3] C. Zheng, L. Li, F. Xu, F. Sun, and M. Ding, Evolutionary route planner for unmanned air vehicles, *IEEE Trans. Robot.*, 2005: 609-620.

[4] X. Yang, L. M. Alvarez, and T. Bruggemann, A 3D collision avoidance strategy for UAVs in a non-cooperative environment, *J. Intell. Robot. Syst.*, 2013: 315-327.

[5] Tianze Zhang and Xin Huo, Path planning and control of a quadrotor UAV: a symbolic approach, *ASCC* 2017, accepted.

[6] R. R. da Silva, B. Wu, and H. Lin, Formal Design of Robot Integrated Task and Motion Planning, *2016 IEEE 55th Conference on Decision and Control*, 2016: 6589 - 6594.

[7] Y. Guo and M. Balakrishnan, Complete coverage control for nonholonomic mobile robots in dynamic environments, in Proceedings of the *IEEE International Conference on Robotics and Automation*, 2006: 1704-1709.

[8] S. M. LaValle, Motion planning-Part I: The essentials, *IEEE Robot. Autom. Mag.*, 2011: 79-89.

[9] Li G S, Chou W S, Path planning for mobile robot using self-adaptive learning particle swarm optimization. *Sci China Inf Sci*, 2018, 61(5): 052204.

[10] Sascha Lange, Martin Riedmiller, Arne Voigtländer, Autonomous reinforcement learning on raw visual input data in a real world application, *WCCI 2012 IEEE World Congress on Computational Intelligence*, 2012: 10-15.

[11] Lei Tai and Ming Liu, Towards cognitive exploration through deep reinforcement learning for mobile robots, arXiv:1610.01733v1 [cs.RO] 6 Oct 2016.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature*, 2015: 529-533.

[13] Pfeiffer M, Schaeuble M, Nieto J, et al. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots, *IEEE International Conference on Robotics and Automation*, 2017:1527-1533.

[14] P. Tabuada, Verification and Control of Hybrid Systems. New York: *Springer*, 2009.

[15] R. Alur, T. Henzinger, G. Lafferriere and G. J. Pappas, Discrete abstractions of hybrid systems, in *Proc. IEEE: Special Issue on Hybrid Systems*, Vol. 88, ed. P. J. Antsaklis, 2000: 971-984.

[16] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins and G. J. Pappas, Symbolic planning and control of robot motion, *IEEE Robot. Autom. Mag.*, 2007: 61-70.

[17] G. Reißig, A. Weber, and M. Rungger, Feedback Refinement Relations for the Synthesis of Symbolic Controllers. *IEEE Transactions on Automatic Control*. 2015: 1781 - 1796

[18] Karimoddini A, Cai G, Chen BM, Lin H, Lee TH, Multi-layer flight control synthesis and analysis of a small-scale Uav helicopter. in *Proceedings of 4th IEEE international conference on robotics, automation and mechatronics*, Singapore, 2010: 321-326.

[19] Bayraktar S, Fainekos G, Pappas G, Experimental cooperative control of fixedwing Unmanned Aerial Vehicles. in *43rd IEEE conference on decision and control*, 2004: 4292-8.

[20] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, Autonomous inverted helicopter flight via reinforcement learning, in *Experimental Robotics IX. Springer*, 2006: 363-372.