项目编号：S370PRP25001

上海交通大學

本科生研究计划（PRP）研究论文

（第 25 期）

论文题目：自适应性四旋翼飞行器

项目负责人：焦子元　　学院（系）：密西根学院

指导教师：马澄斌　　学院（系）：密西根学院

参与学生：沈骏杰、张弋

项目执行时间：2014 年 2 月 至 2014 年 9 月

自适应性四旋翼飞行器

密西根学院　F5123709192　沈骏杰

指导老师：密西根学院　马澄斌

摘 要

四旋翼飞行器拥有优良的运动性能，能广泛适应复杂地形地貌，完成高难度姿态下的飞行任务。四旋翼飞行器如果可以搭载各种功能的模块，便有很广阔的利用前景。研究制作可以搭载各种功能模块的，而且可以自己适应各种模块的四旋翼，未来可以广泛应用于灾后搜救，未知区域勘探，公共区域监控等领域，具有可观的应用前景。实现四旋翼飞行器的高精度抗大幅扰动飞行控制算法，可以更换多种功能模块，优化系统架构，使后续维护和二次开发更高效。

关键词：四旋翼，高精度，控制算法

ABSTRACT

Quadrotor has excellent motion performance so that it can adapt to complex terrains as well as accomplish flight missions of superior difficulty. With modules of various functions, quadrotor has great potential in many fields, such as search and rescue, prospection of zone of ignorance and monitor of public territory. The realization of the new control algorithm allows the quadrotor to resist large excitation with high accuracy.

Keywords: quadrotor, high accuracy, control algorithm

1 Description

Quadrotor is a type of Unmanned Mini Aerial Vehicle, which is a nonlinear coupling dynamics system and therefore hard to control. We apply state-of-art machine learning techniques control problem of quadrotor attitude stabilization. Based on NI myRIO's powerful computational capability and extreme portability, this algorithm is implemented as a model-free, online controller tuning procedure, which improves the controller performance, requiring no detail about the dynamical model of the vehicle.

2 Products

NI Hardware: NI myRIO-1900

NI Software: LabVIEW2013
myRIO module
Real-Time module
FPGA module
MathScript RT module

Other Hardware: InvenSense MPU 9150 9-Axis Sensor
Sunnysky V2216 KV900
SkyWalker Quattro ESC
SHARP IR Sensor (GP2Y0A02YK0F)
XBee 1mW Wire Antenna Zigbee

Other Software: Matlab R2014a

3 Challenge

In spite of the advantage of the flexible maneuverability, quad-rotor is a dynamically unstable, nonlinear, strongly coupled system that has to be stabilized by a elaborately designed or tuned control system. Traditionally, quad-rotor applications use manually tuned PID or LQR controllers derived from a simplified linear model. These controllers require exhausting parameter identification and provide no guarantee for stability while tracking aggressive paths, especially in face of sensor noise, nonlinear disturbance and inaccurate model.

4 Solution

This paper applies a state-of-art reinforcement learning algorithm called Policy Gradient via Signed Derivative (PGSD) to the control problem of quad-rotor attitude stabilization. Based on NI myRIO's powerful computational capability and extreme portability, this algorithm is implemented as a model-free, online controller tuning procedure, which improves the controller performance, requiring no detail about the dynamical model of the vehicle. This proves to be a both adaptive and optimal control strategy, greatly overcoming nonlinearity, modeling error, environmental variants. Experiments on both simulation and hardware display the validity of the solution.

4.1 Controller Parametrization

In spite of strongly coupled dynamics, controller policy of a quad-rotor can be roughly separated into three less dependent channels, namely the pitch, roll and yaw channels.
We consider a linear controller with coupling term involved so that control inputs for each channel are represented as

$$u_\theta = Kp_\theta\ (\theta_t^* - \theta_t) - Kd_\theta\ \dot{\theta} + Kc_\theta\ \dot{\psi}_t\dot{\varphi}_t$$

$$u_\varphi = Kp_\varphi\ (\varphi_t^* - \varphi_t) - Kd_\varphi\ \dot{\varphi}_t + Kc_\varphi\ \dot{\theta}_t\dot{\psi}_t$$

$$u_\theta = Kp_\theta\ (\theta_t^* - \theta_t) - Kd_\theta\ \dot{\theta}_t + Kc_\theta\ \dot{\psi}_t\dot{\varphi}_t$$

where the angles with asterisks are the target we want to achieve at each time step.

Note that this controller parametrization is actually a strengthened PID control scheme, which also takes the nonlinear coupling terms into account. The addition of coupling terms can be proved to make the controller stable not only around equilibrium point.

In matrix form, the controller policy can be simplified as

$$u_t = \begin{bmatrix} u_\theta \\ u_\varphi \\ u_\psi \end{bmatrix} = w^{\mathbf{T}}\phi(s_t)\ ,\ w = \begin{bmatrix} Kp_\theta & 0 & 0 \\ 0 & Kp_\varphi & 0 \\ 0 & 0 & Kp_\psi \\ Kd_\theta & 0 & 0 \\ 0 & Kd_\varphi & 0 \\ 0 & 0 & Kd_\psi \\ Kc_\theta & 0 & 0 \\ 0 & Kc_\varphi & 0 \\ 0 & 0 & Kc_\psi \end{bmatrix}$$

The nine controller parameters remain to be tuned. Instead of exhaustively determining their value by manual trails or heuristics, we adopt the machine learning algorithm to automatically tune their values, based on the data collected during flight. The controller is guaranteed to converge to an optimal one after a few iterations.

4.2 Auto-Tuing using Reinforcement Learning Algorithm

In this section we briefly describe the reinforcement learning algorithm which runs online on NI myRIO-1900, which improves controller performance during flight. The powerful computation capability of NI myRIO-1900 makes the online execution possible with little extra effort.

Following the controller parametrization above, we consider a control cost induced by deviation from the target state at each discrete time step.

$$C_t(s_t, u_t) = (s_t - s_t^\star)^T Q_t (s_t - s_t^\star) + u_t^T R_t u_t$$

The weighting matrices Q and R are selected to be semi-definite so that the furthe the quad-rotor deviates the target state, the more cost is induced at the time step.
Consider a control task for the quadrotor to following a series of attitude states

$$\{s_0^*, s_1^*, \ldots, s_H^*\}$$

The total cost, which is the sum of all costs at each time step, is the criterion of the controller performance. Hence our aim now becomes to find a appropriate parameter matrix such that

$$w_{opt} = \arg\min \sum_{t=1}^{H} C_t(s_t, u_t)$$

where the state and control data s and u are all derived from the flight data. This problem setting is actually a abbreviation of Markov Decision Process(MDP).

The procedure of calculating the optimal parameter matrix is based on the gradient descent method, and uses no details about dynamics model of the quad-rotor at all.

Algorithm 1 Policy Gradient w/ Signed Derivative (PGSD)

Input:

$S \in \mathbb{R}^{m \times n}$: signed derivative matrix
$H \in \mathbb{Z}_+$: horizon
$Q_t \in \mathbb{R}^{n \times n}, R_t \in \mathbb{R}^{m \times m}$: diagonal cost function matrices
$\alpha \in \mathbb{R}_+$: learning rate
$\phi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^k$: feature vector function
$w \in \mathbb{R}^{k \times m}$: initial policy parameters

Repeat:

1. Execute policy for H steps to obtain $u_0, s_1, \ldots, u_{H-1}, u_H$.
2. Compute approximate gradients w.r.t. controls:

$$\widetilde{\nabla}_{u_t} J(s_0, \Theta) \leftarrow \sum_{t'=t+1}^{H} S^T Q_{t'}(s_{t'} - s_{t'}^\star) + R_t u_t$$

3. Update parameters:

$$w \leftarrow w - \frac{\alpha}{H} \sum_{t=0}^{H-1} \phi(s_t, t)(\widetilde{\nabla}_{u_t} J(s_0, \Theta))^T$$

Note that in the algorithm a signed derivative matrix is still needed. This matrix can be viewd as a very rough approximation to the

Jacobian matrix of the derivative of future state with respect to current control input, with the signs retained but concrete values discarded. Hence, encoded in this matrix is a rough, naive model description, which is sufficient for the algorithm to converge efficiently. In the application of our quad-rotor, the signed derivative matrix is set to

$$S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.3 Hardware Experiment

NI myRIO-1900 is used as flight control processor, responsible for sensor data acquisition, data fusion, algorithm computation, rotor control as well as communication. The huge number of channels on NI myRIO and their reconfigurability make it possible to integrate multiple sensors, needed on a quad-rotor, into a unit processing center.

As the default development environment for NI myRIO, NI LabVIEW2013 provides our team with a variety of customized functionalities, extremely simplifying the development and commissioning process and allowing us to focus consistently on the high level design of the project.

Fig 1. Overview of our quadrotor.

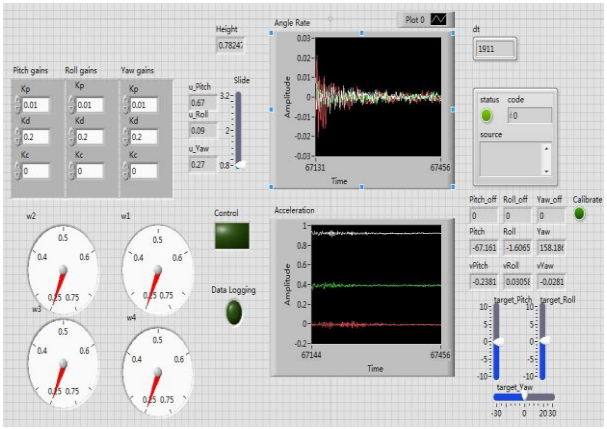

Fig.2 Overview of VI front panel.

In order to display the efficiency of this algorithm, we set the control loop frequency to a medium level, 100 Hz, which is commonly used on commercial quad-rotor, although myRIO FPGA allows much higher frequency.

The procedure of the experiment is as follows. At first, we choose a reasonable but not possibly optimal set of controller parameters, for instance,

$$Kp_\theta = Kp_\varphi = Kp_\psi = 0.01$$

$$Kd_\theta = Kd_\varphi = Kd_\psi = 0.10$$

$$Kc_\theta = Kc_\varphi = Kc_\psi = 0$$

Then, we collect data while the quad-rotor is aviating for each 10 seconds, which is equivalent to H=10*100=1000 time steps. Based on the data collected, the algorithm described in section 2 updates the controller parameters automatically. This data-acquisition, update iteration runs for a few times. According to our result, 10 iteration would be sufficient for apparent improvement on controller performance to be seen. The initial performance is shown in Fig.3 and the ultimate optimal performance is shown in Fig.4.

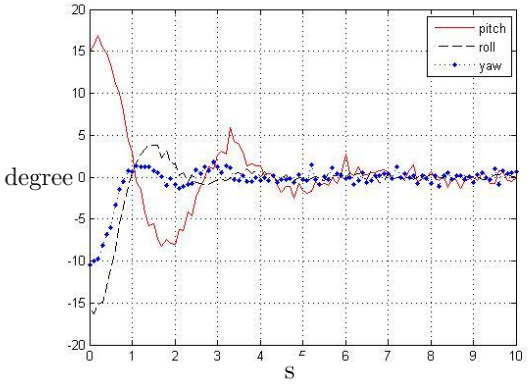

Fig.3 Initial performance

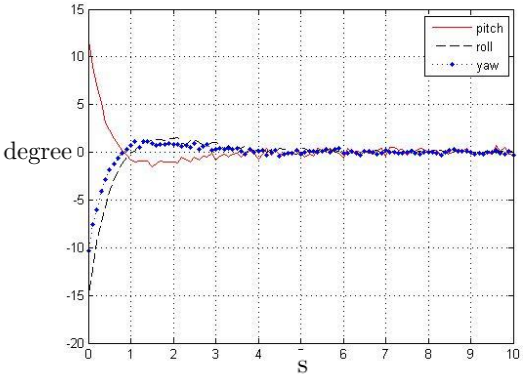

Fig.4 Ultimate performance

5 Revision

1. The linear controller parametrization could be replaced by a nonlinear one, for example, artificial neural network.
2. The Controller VI could be programmed into FPGA to release RT resource and improve performance.
3. Position control of the quad-rotor could be added.
4. The VI user interface could be strengthened.

6 Reference

The preceding parts are all our own works, so there's no reference.

7 Acknowledge

We want to appreciate anyone who has helped us in this PRP. In particular, we are grateful to Prof. Ma Quanbing who gave us relative knowledge. Without his help, it's impossible for us to finish our project.