Automatic quadcopter launching under unstructured conditions

Zipan He
Xubo Lyu
MARS Lab, Simon Fraser University, Autonomy Lab, Simon Fraser University
zhe118@syr.edu xlv@sfu.ca

Abstract

Automatic quadropter launching under unstructured conditionsFlexible launching of a quadrotor system remains challenging due to its highly dynamical nature. Generally, the popular commercial quadcopter (e.g. DJI) requires a trained operator to execute the launching on a plain ground with remote control. Such requirements cannot be always satisfied under certain extreme environments or emergencies. In this project, the objective is to design an automatic launching module using reinforcement learning (RL) for a nano-quadrotor system (Crazyflie 2.1), which supports more flexible launching under unstructured conditions, such as rugged ground and hand-throwing.
During the process of this project, it teaches a quadrotor how to launch from a plain ground. The Gazebo simulator can provide accurate physical modelling of quadrotor and allow for efficient and safe training for this task. The project uses neural network as a powerful, non-linear controltler and select the state-of-art RL algorithm, either model-free or model-based one, to optimize the controller to achieve stable launching in simulation. In the future, the controller needs to be transferred and fully fine-tuned from simulation to real system. Finally, put this special quadrotor on the ground and witness the magic of neuro-based fully automatic launching!

Keywords: Quadcopter, Crazyflie 2.1, Gazebo simulator, Neural network, Reinforcement learning algorithm, Tensorflow, Pytorch

1 Introduction

Unmanned Aerial Vehicles (UAV), initially mainly used in national defense and military, such as reconnaissance aircraft and drones. In recent years, interest and attention related to civilian use of UAV has significantly increased. Today, drones are used in aerial photography, agriculture, plant protection, micro selfies, express transportation, disaster rescue, observation of wild animals, monitoring of infectious diseases, surveying and mapping, news reports, power inspections, disaster relief, film and television shooting, romance and other fields Applications. All of these greatly expand the use of the drone itself.

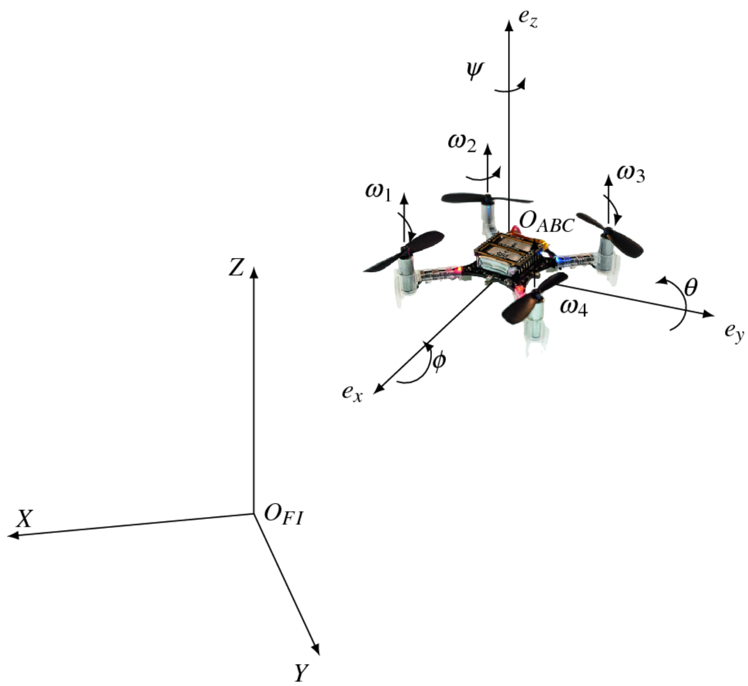


Figure 1: Crazyflie2.1 Nano- quadcopte

Many existing algorithms for autonomous control and navigation have been studied by lots of researchers [1] [2] [3] [4], but there are still great challenges to enable UAVs to work autonomously in restricted and unknown environments or indoors. Therefore, it comes with the need for some simulation tools to understand what will happen in the unfamiliar environment and manage the complexity and heterogeneity of the hardware and the applications

In this project, we mainly use ROS with Gazebo 7 to complete our simulation task. Robot Operating System (ROS) is a flexible framework for writing robot software, and Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.

With these support , we use ROS package CrazyS [5] [6] in order to modeling, developing and integrating the Crazyflie 2.1 nano-quadcopter in the physics based simulation environment, and then designed an automatic launching module using reinforcement learning(RL). Therefore, our goal that launching from rugged ground or hand-throwing could be achieved.

2 Motivation

• Quadcopter autonomous stabilization remains a challenge in many aggressive and non-aggressive scenarios such as automatic takeoff, hand and throw launch, communication loss, and crash recovery.
• Previous works [7] [8] have tried to achieve this, but they often require extensive modelling and designing of quadcopter dynamics and high-level controller, and only can be applied to non-aggressive cases.

3 Algorithm

3.1 Algorithm Selection

To better explore the possible solutions, we decide to select two types of RL algorithm: PPO [9] and D4PG [10] that PPO is a on-policy model-free method and D4PG is an off-policy model-free method.

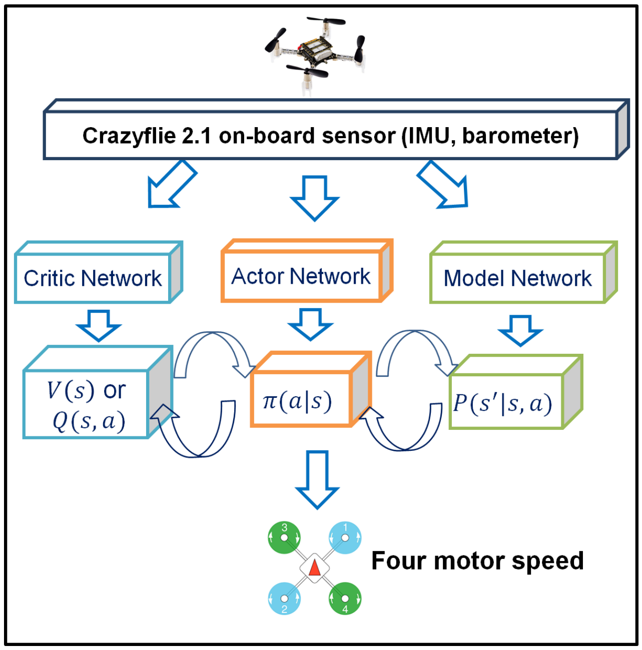


Figure 2: Important modules of our method

3.1.1 Proximal policy optimization (PPO)

As in reinforcement learning , finding the most satisfied result isn't that obvious, because the algorithms have many moving parts that are hard to debug, and they require substantial effort in tuning in order to get good results. Proximal policy optimization(PPO) strikes a balance between ease of implementation, sample complexity, and ease of tuning, trying to compute an update at each step that minimizes the cost function while ensuring the deviation from the previous policy is relatively small.

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right]$$

- θ is the policy parameter
- $\hat{E}_t$ denotes the empirical expectation over timesteps
- r_t is the ratio of the probability under the new and old policies, respectively
- $\hat{A}_t$ is the estimated advantage at time t
- ε is a hyperparameter, usually 0.1 or 0.2

Figure 3: Proximal policy optimization algorithm

However, as the PPO algorithm comes from TRPO, and TRPO is to solve the problem of difficult selection of the step size in the strategy gradient algorithm. Therefore, the PPO algorithm is essentially a policy gradient algorithm. So PPO has the limitations of the policy gradient algorithm that is the parameters are updated along the direction of the policy gradient. Because the parameter itself has its own spatial structure, and the direction of the strategy gradient does not consider the spatial structure of the parameter itself, so the update speed will be slow.

3.1.2 Distributed Distributional Deep Deterministic Policy Gradient(D4PG)

In general, D4PG adds the following points on the basis of DDPG:

Algorithm 1 D4PG

Input: batch size M, trajectory length N, number of actors K, replay size R, exploration constant ϵ, initial learning rates α_0 and β_0

```
Initialize network weights (θ, w) at random
Initialize target weights (θ', w') ← (θ, w)
Launch K actors and replicate network weights (θ, w) to each actor
for t = 1, ..., T do
    Sample M transitions (x_{i:i+N}, a_{i:i+N-1}, r_{i:i+N-1}) of length N from replay with priority p_i
    Construct the target distributions Y_i = (Σ_{n=0}^{N-1} γ^n r_{i+n}) + γ^N Z_{w'}(x_{i+N}, π_{θ'}(x_{i+N}))
    Note, although not denoted the target Y_i may be projected (e.g. for Categorical value distributions).
    Compute the actor and critic updates
        δ_w = (1/M) Σ_i ∇_w (R p_i)^{-1} d(Y_i, Z_w(x_i, a_i))
        δ_θ = (1/M) Σ_i ∇_θ π_θ(x_i) E[∇_a Z_w(x_i, a)] |_{a=π_θ(x_i)}
    Update network parameters θ ← θ + α_t δ_θ, w ← w + β_t δ_w
    If t = 0 mod t_target, update the target networks (θ', w') ← (θ, w)
    If t = 0 mod t_actors, replicate network weights to the actors
end for
return policy parameters θ
```

$$Y_i = \left(\sum_{n=0}^{N-1} \gamma^n r_{i+n}\right) + \gamma^N Z_{w'}(\mathbf{x}_{i+N}, \pi_{\theta'}(\mathbf{x}_{i+N}))$$

$$\delta_w = \frac{1}{M}\sum_i \nabla_w (Rp_i)^{-1} d(Y_i, Z_w(\mathbf{x}_i, \mathbf{a}_i))$$

$$\delta_\theta = \frac{1}{M}\sum_i \nabla_\theta \pi_\theta(\mathbf{x}_i)\, \mathbb{E}[\nabla_{\mathbf{a}} Z_w(\mathbf{x}_i, \mathbf{a})]\big|_{\mathbf{a}=\pi_\theta(\mathbf{x}_i)}$$

Actor

```
repeat
    Sample action a = π_θ(x) + εN(0, 1)
    Execute action a, observe reward r and state x'
    Store (x, a, r, x') in replay
until learner finishes
```


Figure 4: Distributed Distributional Deep Deterministic Policy Gradien Algorithm

1. Distributional RL Because it is an off-policy algorithm, multiple actors can be used to sample distributed, and then stored in the same replay buffer. The learner samples from the buffer, and then synchronizes the weights to each actor after updating.

2. N-step returns

$$(\mathcal{T}_\pi^N Q)(\mathbf{x}_0, \mathbf{a}_0) = r(\mathbf{x}_0, \mathbf{a}_0) + \mathbb{E}\Big[\sum_{n=1}^{N-1} \gamma^n r(\mathbf{x}_n, \mathbf{a}_n) + \gamma^N Q(\mathbf{x}_N, \pi(\mathbf{x}_N)) \,\big|\, \mathbf{x}_0, \mathbf{a}_0\Big]$$

Figure 5: Distributed Distributional Deep Deterministic Policy Gradien Algorithm

As we all know, N-step return is generally better than one-step return because it better balances bias and variance. However, as an off-policy method, DDPG uses experience replay to update critics. this combines N-step return but does not consider off-policy correction.

3. Distributed sampling (APEX) and Prioritized Experience Replay (PER)

These two improvements have improved the algorithm from the source and use of samples, and have obtained higher-quality critic and actor training. Reward Design

3.2 Reward Design

To train the agent more efficiently, we also design an “Extended Time-To-Reach” reward based on [11]. This reward takes an approximate quadcopter dynamics into consideration and can accelerate learning

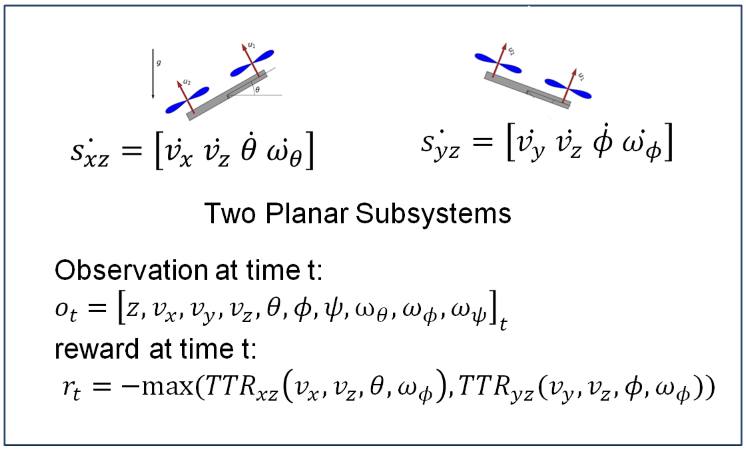


Figure 6: Extended Time-To-Read Reward

3.3 Model Learning

We also investigate using deep RL method for learning the drone’s dynamic model similar to [12]. The learned model can help to alleviate exploration cost by planning reasonable trajectories

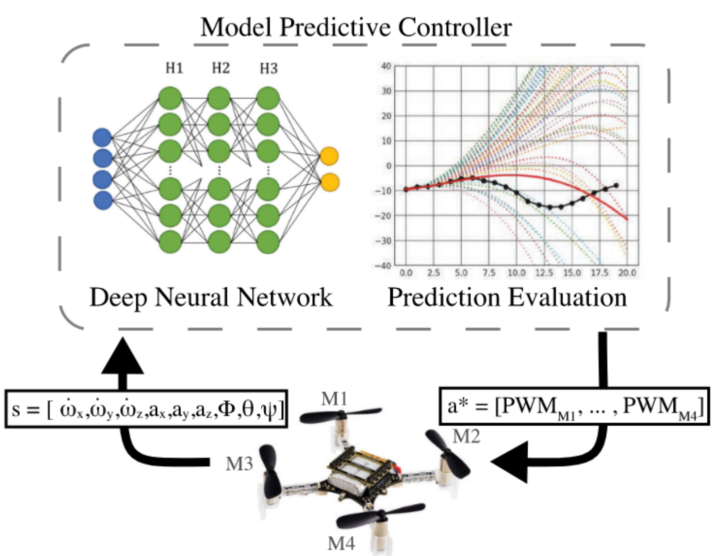
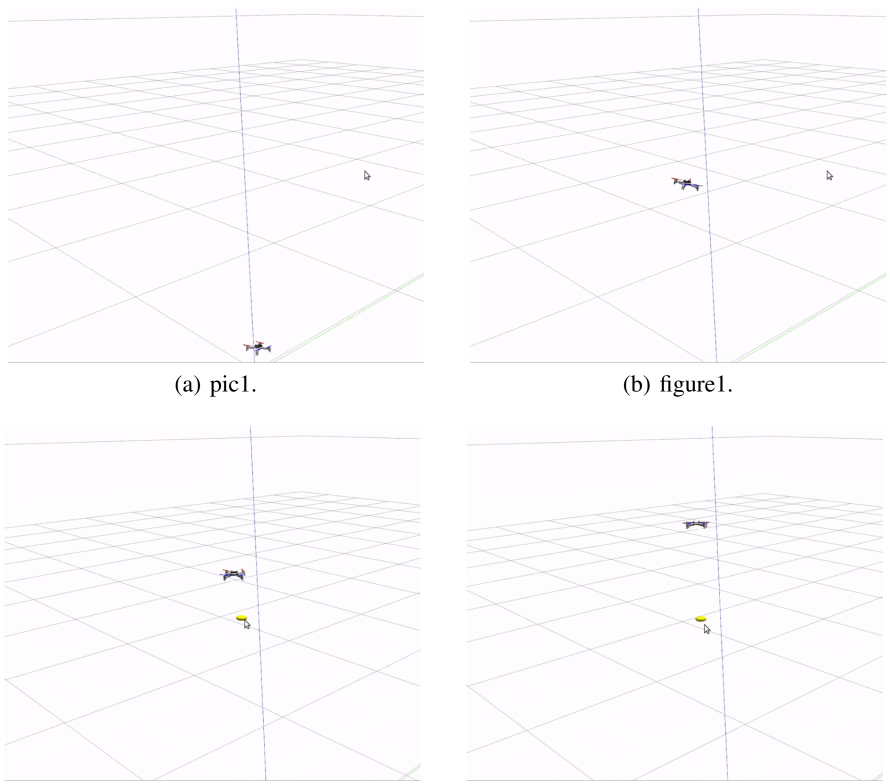


Figure 7: The model predictive control loop used to stabilize the Crazyflie

4 Experiments

We choose a simulated Crazyflie 2.1 nano quadcopter as testbed and first apply our methods on "automatic takeoff and hover" task to test the performance. After training around 600k timesteps, our method can successfully stabilize the quadcopter up to 20s in the air

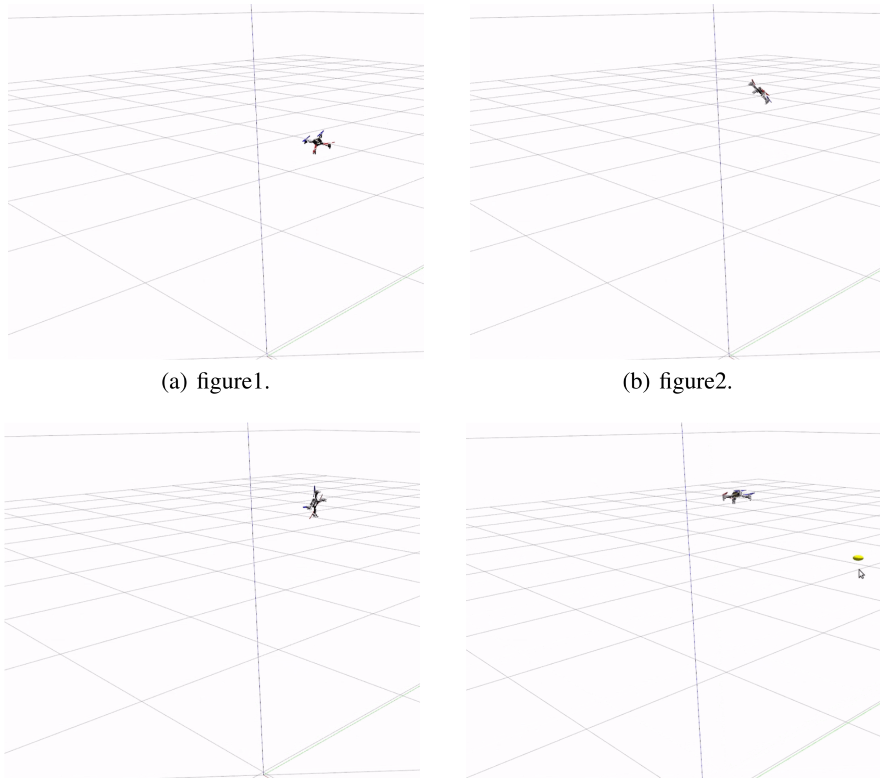
Find github link for more information CMPT726 Final Project



(a) pic1. (b) figure1.

(c) figure2. (d) figure3.

Figure 8: Crazyflie perfect hovering



(a) figure1. (b) figure2.

(c) figure3. (d) figure4.

Figure 9: Crazyflie recover from bad state

5 Conclusion and future work

In this paper, we illustrated how to expand the functionalities of the ROS package CrazyS for modeling and simulate the nano-quadcopter Crazyflie 2.1 in a simulation platform Gazebo7, we achieved a complete simulation of automatic taking off and hovering of drone. With added challenges of the static instability and fast dynamics of the Crazyflie in simulation, it showed the capabilities and future potential of reinforcement learning Algorithm PPO and D4PG. Future directions for this works can include several aspects. Firstly, difficulty of launching could be increased. We could set quadcopter under unstructured conditions in simulation such as turning flat surface into inclined plane, or taking wind into consider. Secondly, combine other RL algorithm to the current algorithm. For instance, we could combine Hindsight Experience Relay(HER) [13] with current module to get better performance and faster convergence. Thirdly, investigate different methods to transfer our model to a real quadcopter. For example, by fine tuning the current model or learning a new model for dynamics of the real quadcopter. Moreover, all proposed algorithms should be tested in real-world experiments on the real Crazyflie platform in different scenarios Last but not the least, it may be possible to look for some improvements of the inner loop (on-board controller) after having been tested on CrazyS.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[1] B. Landry (2015) Planning and control for quadrotor flight through cluttered environments, *Master's thesis, MIT*,

[2] D. Ferreira de Castro &D. A. dos Santos (2016) A Software-in-the-Loop Simulation Scheme for Position Formation Flight of Multicopters, *Journal of Aerospace Technology and Management*, **8**(4):431–440,

[3] M. Mancini G. Costante P. Valigi T. A. Ciarfugli J. Delmerico D. Scaramuzza (2017)Toward Domain Independence for Learning-Based Monocular Depth Estimation /it IEEE Robotics and Automation Letters, **2**(3): 1778–1785

[4] T. Hinzmann J. L. Schonberger M. Pollefeys and R. Siegwart (2018)Mapping on the Fly: Real- ¨Time 3D Dense Reconstruction, Digital Surface Map and Incremental Orthomosaic Generation for Unmanned Aerial Vehicles *Field and Service Robotics. Springer International Publishing* :383–396.

[5] Giuseppe Silano & Luigi Iannelli. (2019)CrazyS: a software-in-the-loop simulation platform for the Crazyflie 2.0 nano-quadcopter, in Robot Operating System (ROS): The Complete Reference (Volume 4), K. Anis, Ed. Springer International Publishing, 81-115, Chapter 4, 2019.

[6] Giuseppe Silano Emanuele Aucone Luigi Iannelli. CrazyS: A Software-In-The-Loop Platform for the Crazyflie 2.0 Nano-Quadcopter. 2018 26th Mediterranean Conference on Control and Automation (MED), 352-357, 2018, Zadar Croatia.

[7] M. Faessler F. Fontana C. Forster D. Scaramuzza "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 1722-1729.[2]

[8] H. Abaunza & P. Castillo, "Quadrotor Aggressive Deployment, Using a Quaternion-based Spherical Chattering-free Sliding-mode Controller," in IEEE Transactions on Aerospace and Electronic SystemsAutomatic quadcopter launching under unstructured conditions remains challenging due to its highly dynamic nature.

[9] Schulman John et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[10] Barth-Maron Gabriel et al. "Distributed distributional deterministic policy gradients." arXiv preprint arXiv:1804.08617 (2018)

[11] Xubo Lyu & Mo Chen. "TTR-Based Reward for Reinforcement Learning with Implicit Model Priors." arXiv preprint arXiv:1903.09762 (2019).

[12] Lambert NO Drew DS Yaconelli J Levine S, Calandra R Pister KS. Low-Level Control of a Quadrotor With Deep Model-Based Reinforcement Learning. IEEE Robotics and Automation Letters. 2019 Jul 23;4(4):4224-30.

[13]Marcin Andrychowicz Filip Wolski Alex Ray Jonas Schneider Rachel Fong Peter Welinder Bob McGrew Josh Tobin Pieter Abbeel Wojciech Zaremba. "Hindsight Experience Replay" arXiv preprint arXiv:1707.01495..

[14]Artem Molchanov Tao Chen Wolfgang Hönig James A. Preiss Nora Ayanian Gaurav S. Sukhatme "Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors arXiv preprint arXiv:1903.04628

[15]Tianhao Zhang Gregory Kahn Sergey Levine Pieter Abbeel"Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search" arXiv preprint arXiv:1509.06791

[16]William Koch Renato Mancuso Azer Bestavros "Neuroflight: Next Generation Flight Control Firmware" arXiv preprint arXiv:1901.06553

[17]M. Faessler F. Fontana C. Forster and D. Scaramuzza "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 1722-1729, doi: 10.1109/ICRA.2015.7139420.