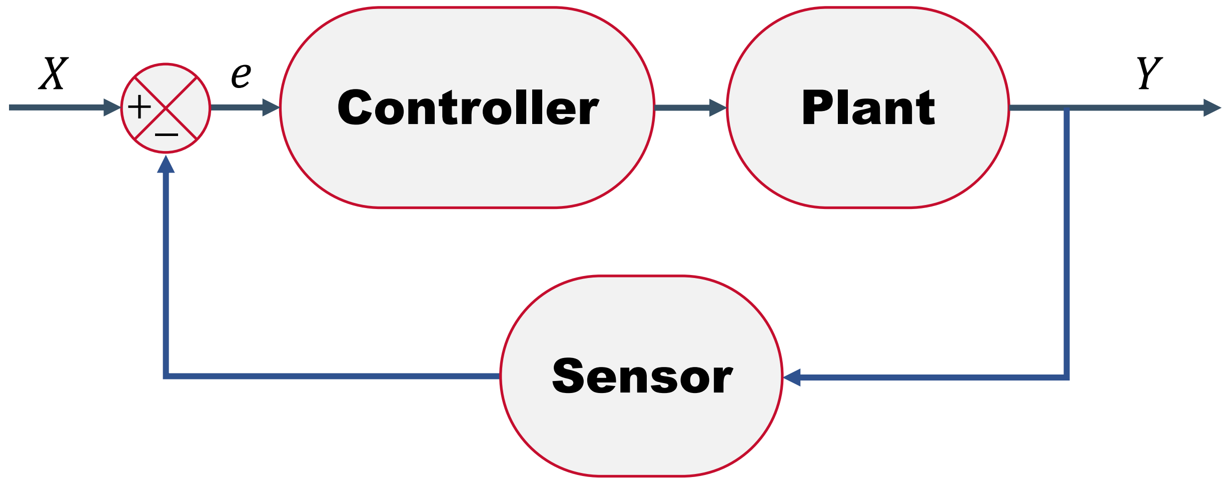
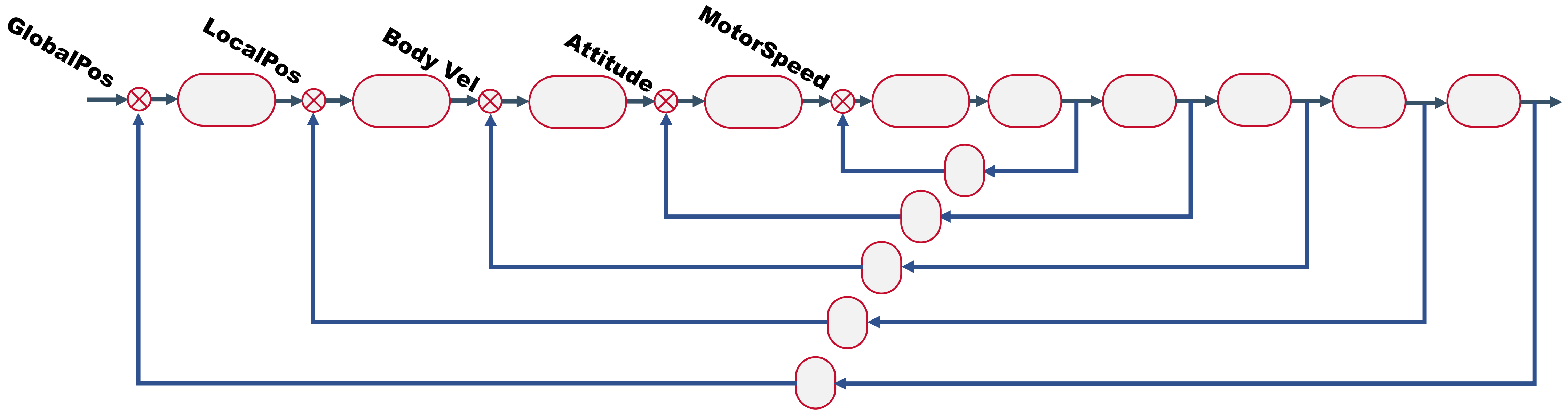
SAI

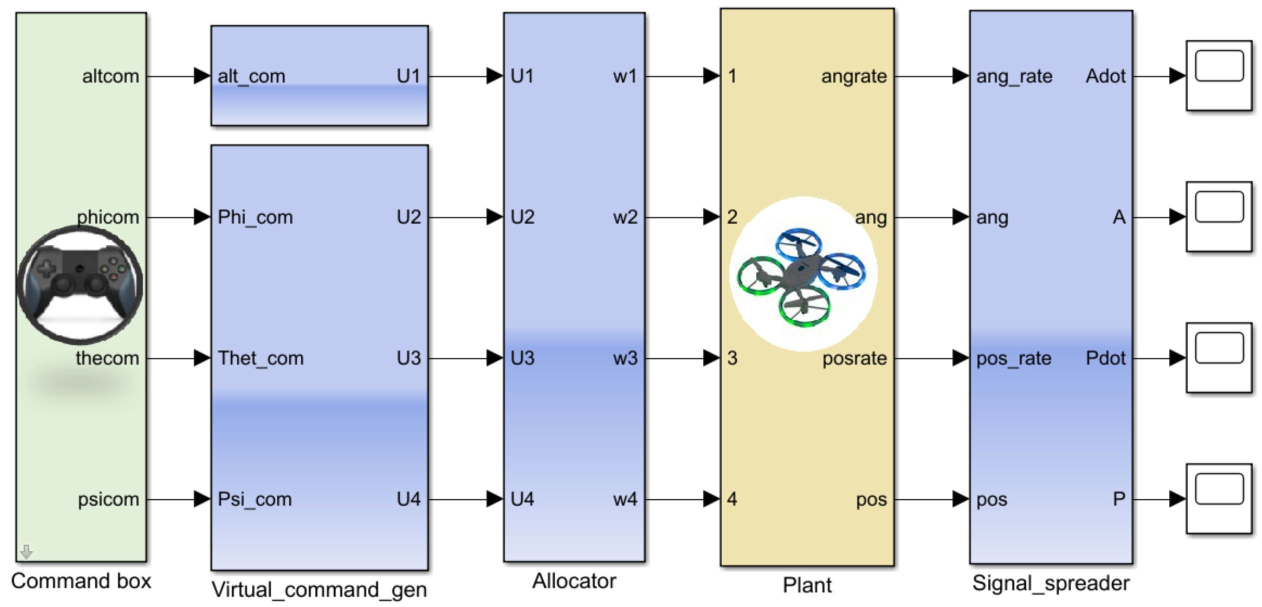
Control of a Quadrotor with Reinforcement Learning

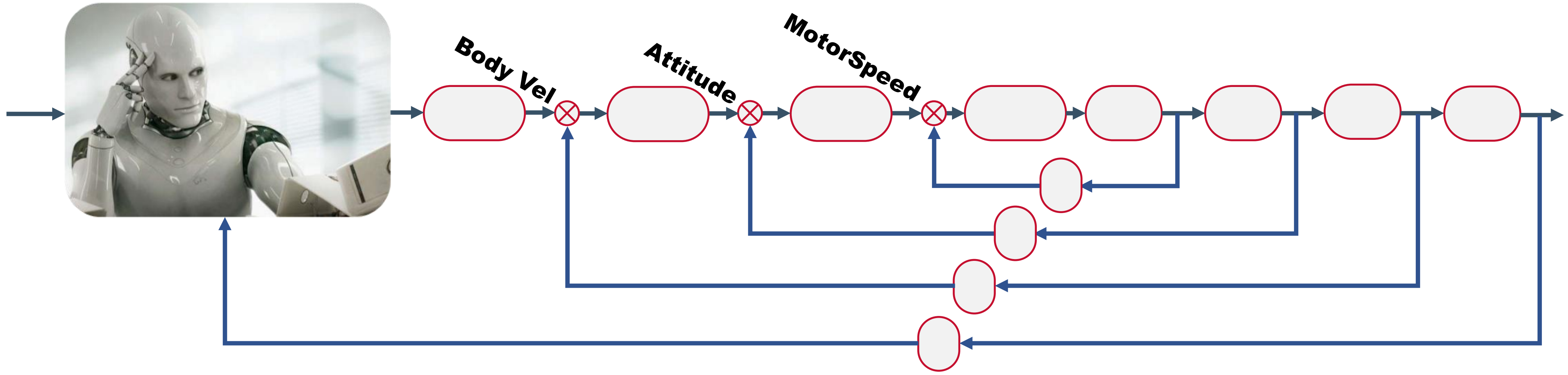
Graduate school of Sejong University
Maters' course in Aerospace engineering
Hong, Daseon

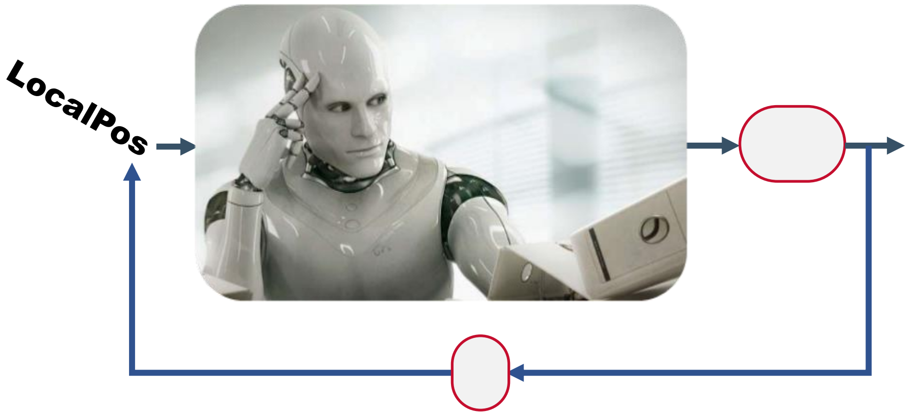
세종대학교
SEJONG UNIVERSITY

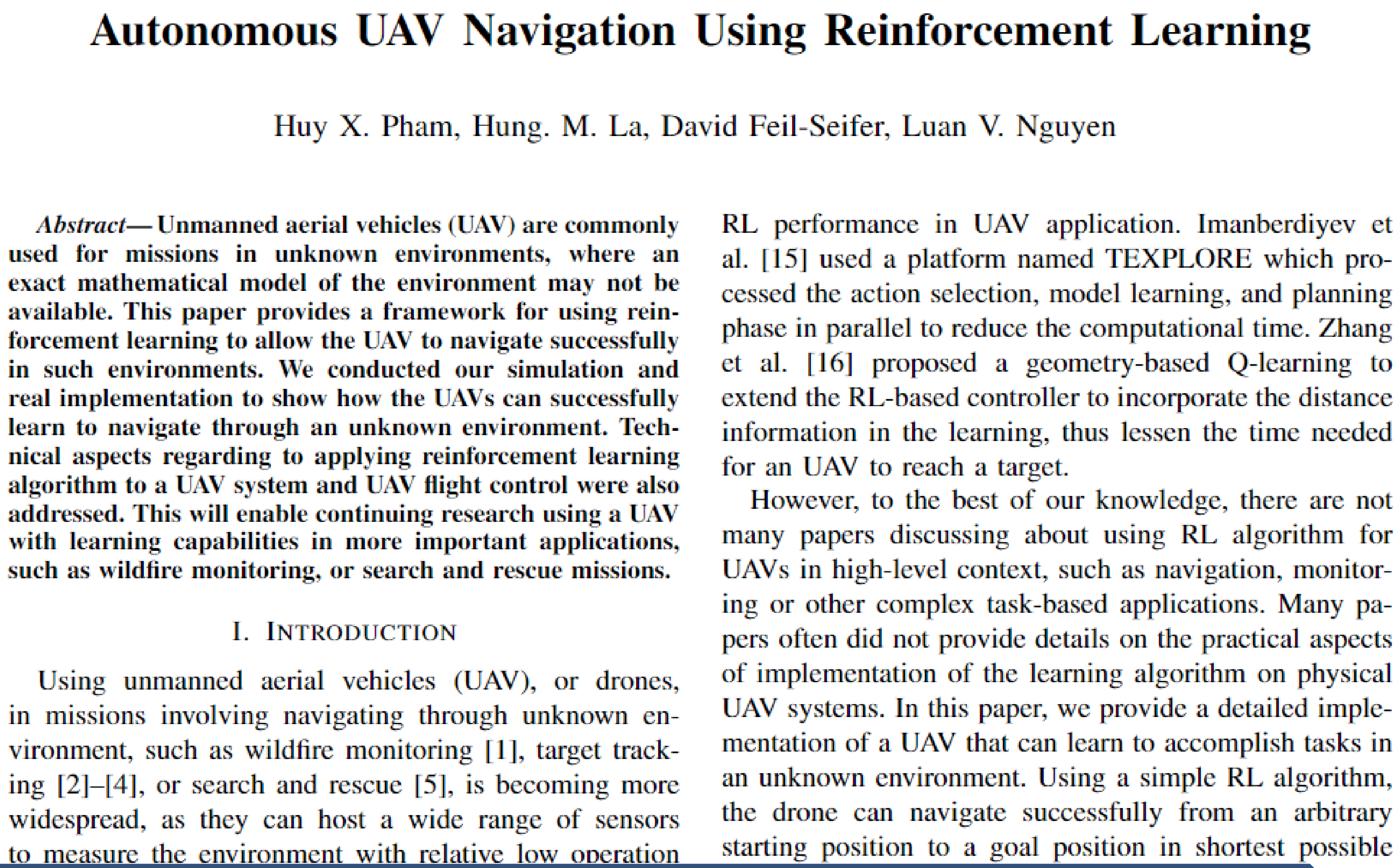

X
$+$
$-$
e
Controller
Plant
Y
Sensor


GlobalPos
LocalPos
Body Vel
Attitude
MotorSpeed


altcom
phicom
thecom
psicom
Command box
alt_com
U1
Phi_com
U2
Thet_com
U3
Psi_com
U4
Virtual_command_gen
U1
w1
U2
w2
U3
w3
U4
w4
Allocator
1
angrate
2
ang
3
posrate
4
pos
Plant
ang_rate
Adot
ang
A
pos_rate
Pdot
pos
P
Signal_spreader


Body Vel
Attitude
MotorSpeed


LocalPos



Autonomous UAV Navigation Using Reinforcement Learning

Huy X. Pham, Hung. M. La, David Feil-Seifer, Luan V. Nguyen

***Abstract*— Unmanned aerial vehicles (UAV) are commonly used for missions in unknown environments, where an exact mathematical model of the environment may not be available. This paper provides a framework for using reinforcement learning to allow the UAV to navigate successfully in such environments. We conducted our simulation and real implementation to show how the UAVs can successfully learn to navigate through an unknown environment. Technical aspects regarding to applying reinforcement learning algorithm to a UAV system and UAV flight control were also addressed. This will enable continuing research using a UAV with learning capabilities in more important applications, such as wildfire monitoring, or search and rescue missions.**

I. Introduction

Using unmanned aerial vehicles (UAV), or drones, in missions involving navigating through unknown environment, such as wildfire monitoring [1], target tracking [2]–[4], or search and rescue [5], is becoming more widespread, as they can host a wide range of sensors to measure the environment with relative low operation

RL performance in UAV application. Imanberdiyev et al. [15] used a platform named TEXPLORE which processed the action selection, model learning, and planning phase in parallel to reduce the computational time. Zhang et al. [16] proposed a geometry-based Q-learning to extend the RL-based controller to incorporate the distance information in the learning, thus lessen the time needed for an UAV to reach a target.

However, to the best of our knowledge, there are not many papers discussing about using RL algorithm for UAVs in high-level context, such as navigation, monitoring or other complex task-based applications. Many papers often did not provide details on the practical aspects of implementation of the learning algorithm on physical UAV systems. In this paper, we provide a detailed implementation of a UAV that can learn to accomplish tasks in an unknown environment. Using a simple RL algorithm, the drone can navigate successfully from an arbitrary starting position to a goal position in shortest possible

II. Problem Formulation

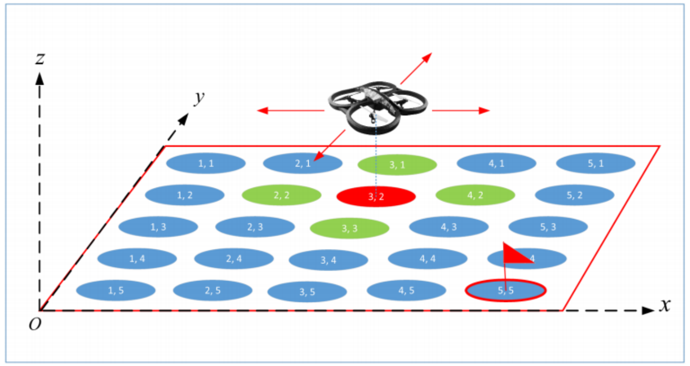


Fig. 1. A UAV navigating in closed environment with discretized state space, represented by discrete circles. The red circle is the UAV's current state, the green circles are the options that the UAV can choose in the next iteration. The goal is marked by a red flag.

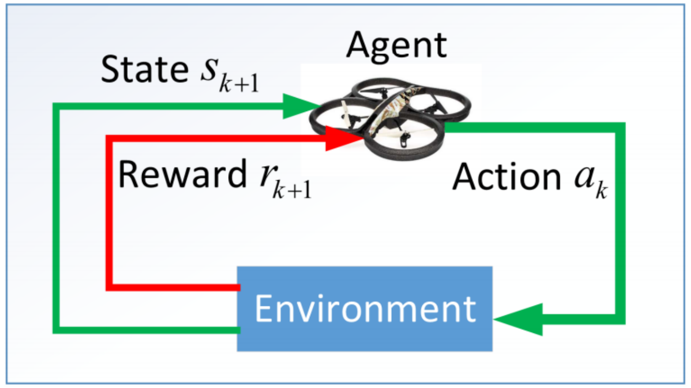
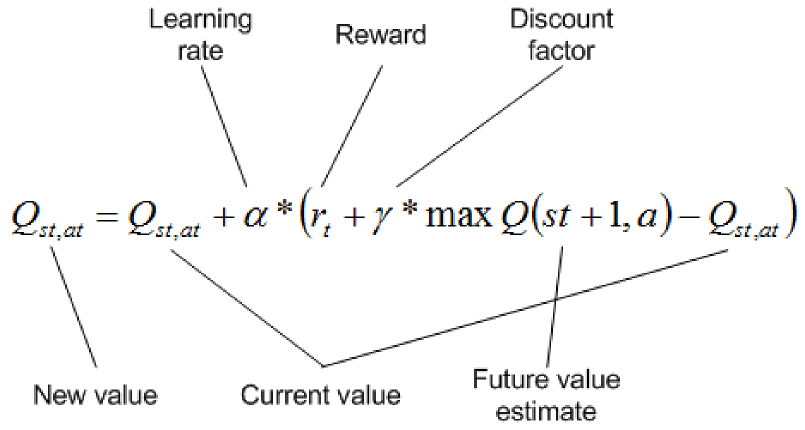


Fig. 3. Reinforcement Learning model.



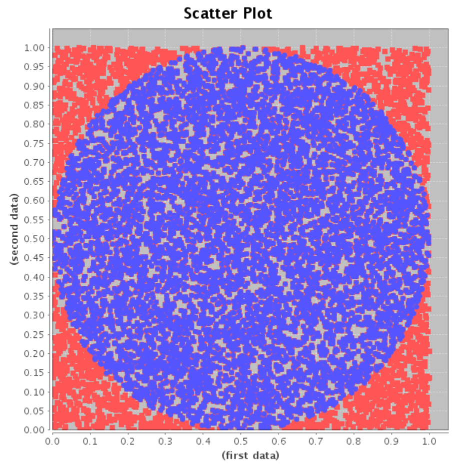
$$Q_{st,at} = Q_{st,at} + \alpha * \left(r_t + \gamma * \max Q(st+1, a) - Q_{st,at}\right)$$

$$\underline{Q^{\pi}(s_t, a_t)} = \underline{E}[\underline{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots}|\underline{s_t, a_t}]$$

Q value for that state given that action

Expected discounted cumulative reward ...

given that state and that action



```
Algorithm 1: PID + Q-LEARNING.
Input: Learning parameters: Discount factor $\gamma$,
       learning rate $\alpha$, number of episode $N$
Input: Control parameters: Control gains
       $K_p, K_p, K_d$, error radius $d$
1  Initialize $Q_0(s,a) \leftarrow 0, \forall s_0 \in S, \forall a_0 \in A$;
2  for $episode = 1 : N$ do
3      Measure initial state $s_0$
4      for $k = 0, 1, 2, ...$ do
5          Choose $a_k$ from $A$ using policy (2)
6          Take action $a_k$ that leads to new state $s_{k+1}$:
7          for $t = 0, 1, 2, ...$ do
8              $u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt}$
9          until $||p(t) - s_{k+1}|| \leq d$
10         Observe immediate reward $r_{k+1}$
11         Update:
12         $Q_{k+1}(s_k, a_k) \leftarrow (1-\alpha)Q_k(s_k, a_k) + \alpha[r_{k+1} + \gamma \max_{a\prime} Q_k(s_{k+1}, a\prime)]$
13     until $s_{k+1} \equiv G$
```

$$r_{k+1} = \begin{cases} 100, & if\ s_{k+1} \equiv G \\ -1, & otherwise. \end{cases}$$

IV. Controller Design and Algorithm

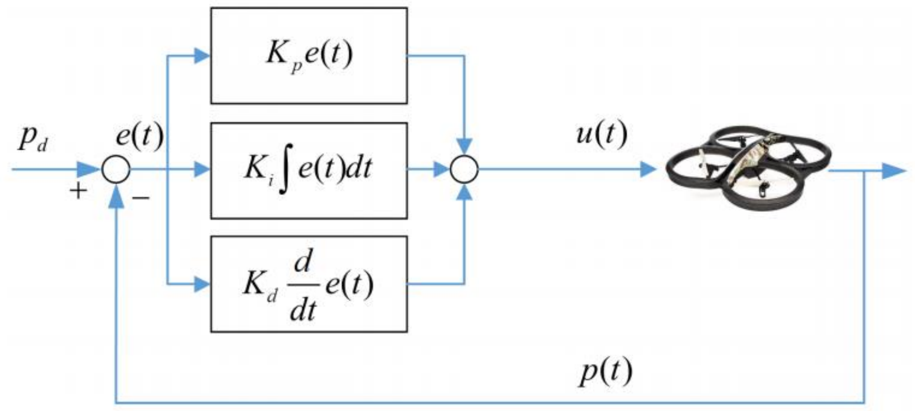
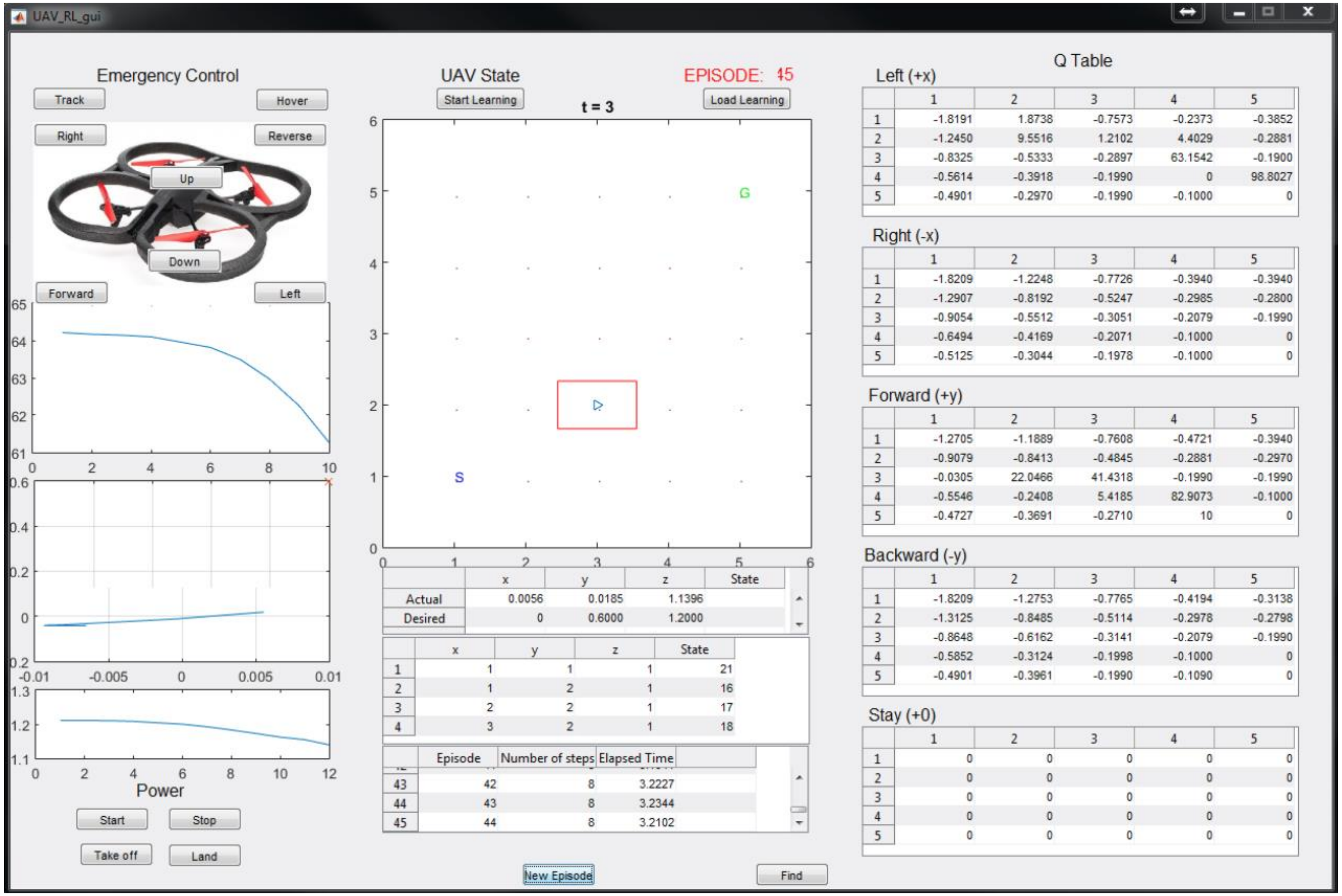


Fig. 4. The PID control diagram with 3 components: proportional, integral and derivative terms.



	x	y	z	State
Actual	0.0056	0.0185	1.1396	
Desired	0	0.6000	1.2000	

	x	y	z	State
1	1	1	1	21
2	1	2	1	16
3	2	2	1	17
4	3	2	1	18

	Episode	Number of steps	Elapsed Time
43	42	8	3.2227
44	43	8	3.2344
45	44	8	3.2102

Q Table

Left (+x)

	1	2	3	4	5
1	-1.8191	1.8738	-0.7573	-0.2373	-0.3852
2	-1.2450	9.5516	1.2102	4.4029	-0.2881
3	-0.8325	-0.5333	-0.2897	63.1542	-0.1900
4	-0.5614	-0.3918	-0.1990	0	98.8027
5	-0.4901	-0.2970	-0.1990	-0.1000	0

Right (-x)

	1	2	3	4	5
1	-1.8209	-1.2248	-0.7726	-0.3940	-0.3940
2	-1.2907	-0.8192	-0.5247	-0.2985	-0.2800
3	-0.9054	-0.5512	-0.3051	-0.2079	-0.1990
4	-0.6494	-0.4169	-0.2071	-0.1000	0
5	-0.5125	-0.3044	-0.1978	-0.1000	0

Forward (+y)

	1	2	3	4	5
1	-1.2705	-1.1889	-0.7608	-0.4721	-0.3940
2	-0.9079	-0.8413	-0.4845	-0.2881	-0.2970
3	-0.0305	22.0466	41.4318	-0.1990	-0.1990
4	-0.5546	-0.2408	5.4185	82.9073	-0.1000
5	-0.4727	-0.3691	-0.2710	10	0

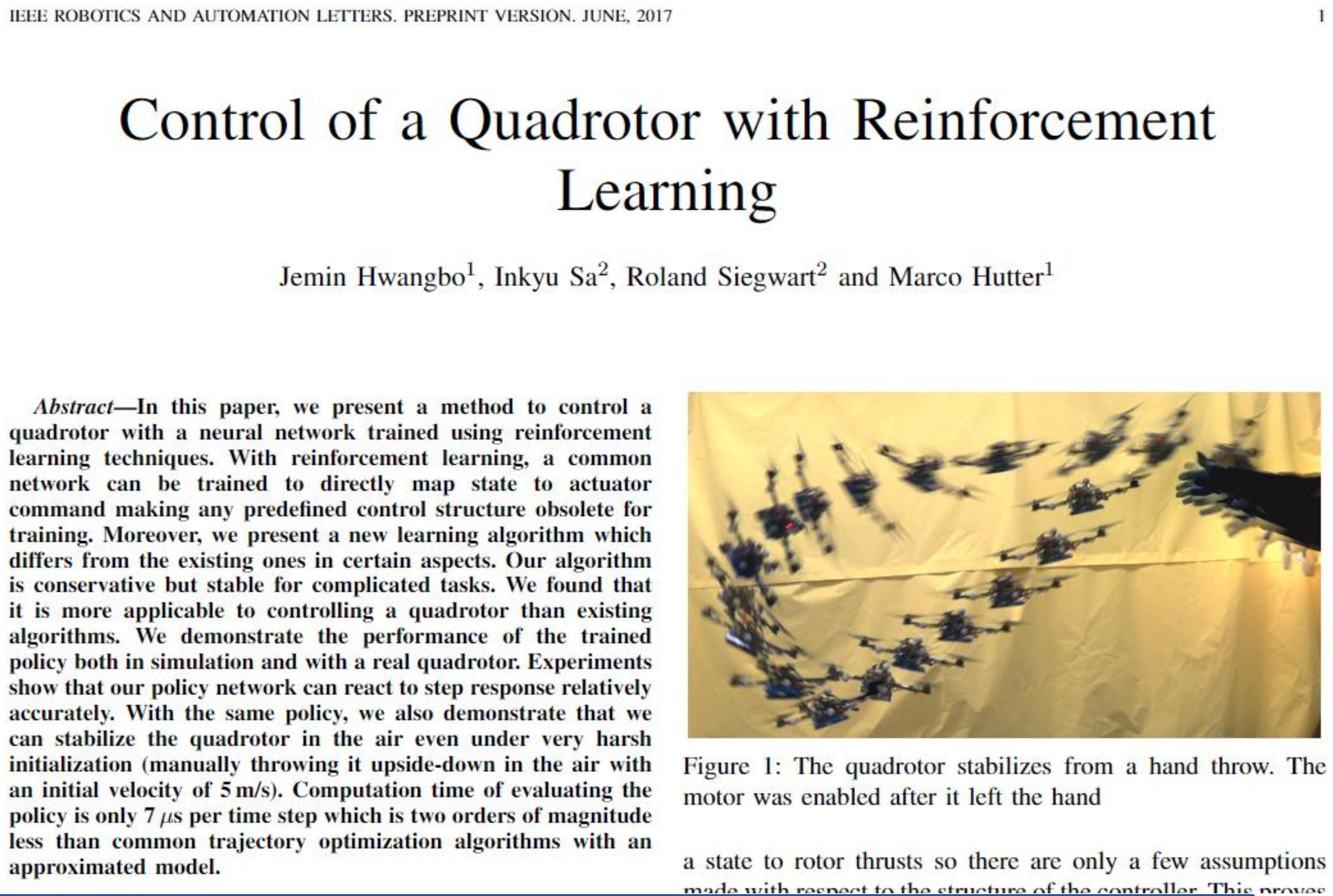
Backward (-y)

	1	2	3	4	5
1	-1.8209	-1.2753	-0.7765	-0.4194	-0.3138
2	-1.3125	-0.8485	-0.5114	-0.2978	-0.2798
3	-0.8648	-0.6162	-0.3141	-0.2079	-0.1990
4	-0.5852	-0.3124	-0.1998	-0.1000	0
5	-0.4901	-0.3961	-0.1990	-0.1090	0

Stay (+0)

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

SAI



Control of a Quadrotor with Reinforcement Learning

Jemin Hwangbo[1], Inkyu Sa[2], Roland Siegwart[2] and Marco Hutter[1]

***Abstract*—In this paper, we present a method to control a quadrotor with a neural network trained using reinforcement learning techniques. With reinforcement learning, a common network can be trained to directly map state to actuator command making any predefined control structure obsolete for training. Moreover, we present a new learning algorithm which differs from the existing ones in certain aspects. Our algorithm is conservative but stable for complicated tasks. We found that it is more applicable to controlling a quadrotor than existing algorithms. We demonstrate the performance of the trained policy both in simulation and with a real quadrotor. Experiments show that our policy network can react to step response relatively accurately. With the same policy, we also demonstrate that we can stabilize the quadrotor in the air even under very harsh initialization (manually throwing it upside-down in the air with an initial velocity of 5 m/s). Computation time of evaluating the policy is only 7 μs per time step which is two orders of magnitude less than common trajectory optimization algorithms with an approximated model.**

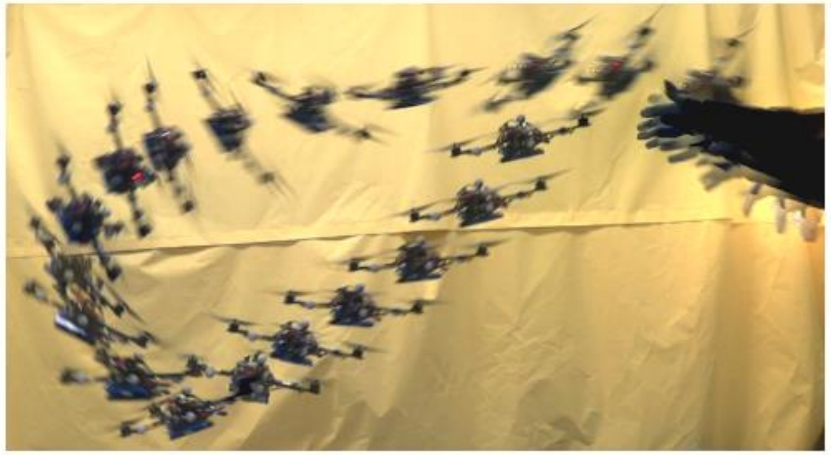
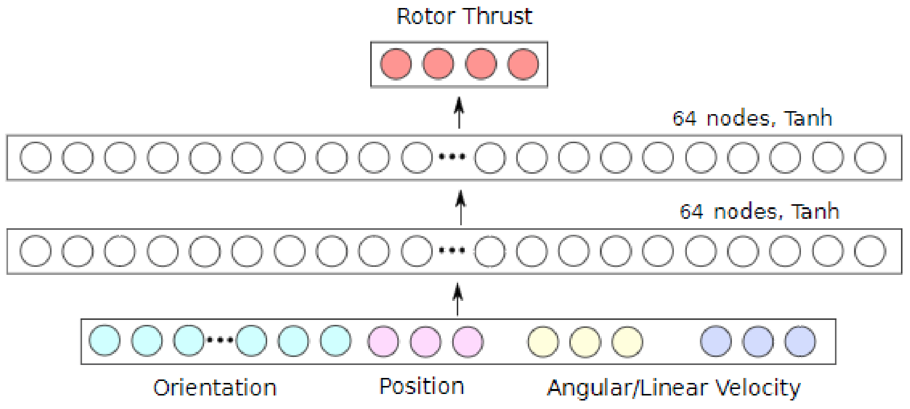


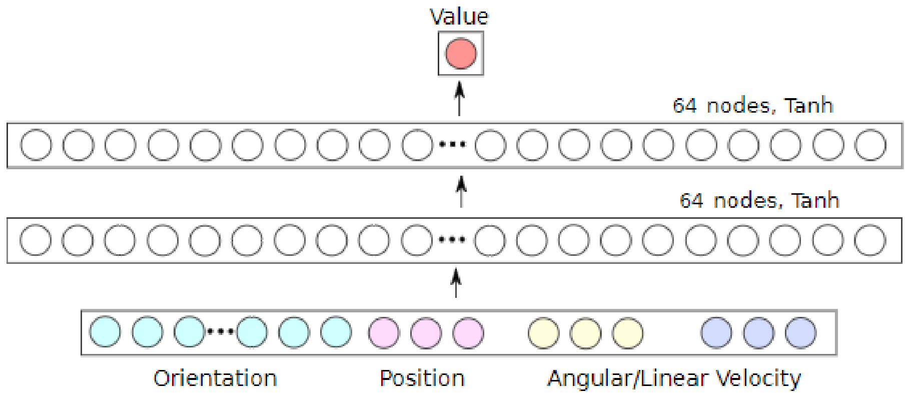
Figure 1: The quadrotor stabilizes from a hand throw. The motor was enabled after it left the hand

a state to rotor thrusts so there are only a few assumptions made with respect to the structure of the controller. This proves

세종대학교
SEJONG UNIVERSITY

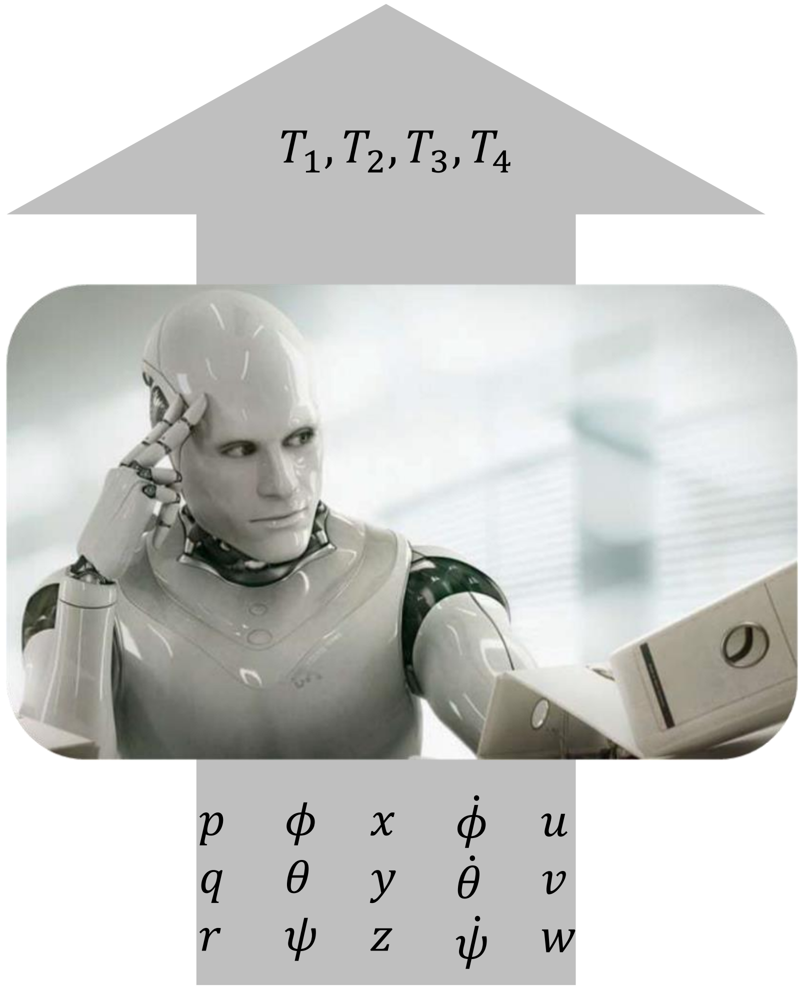


(a) Policy network.



(b) Value network.

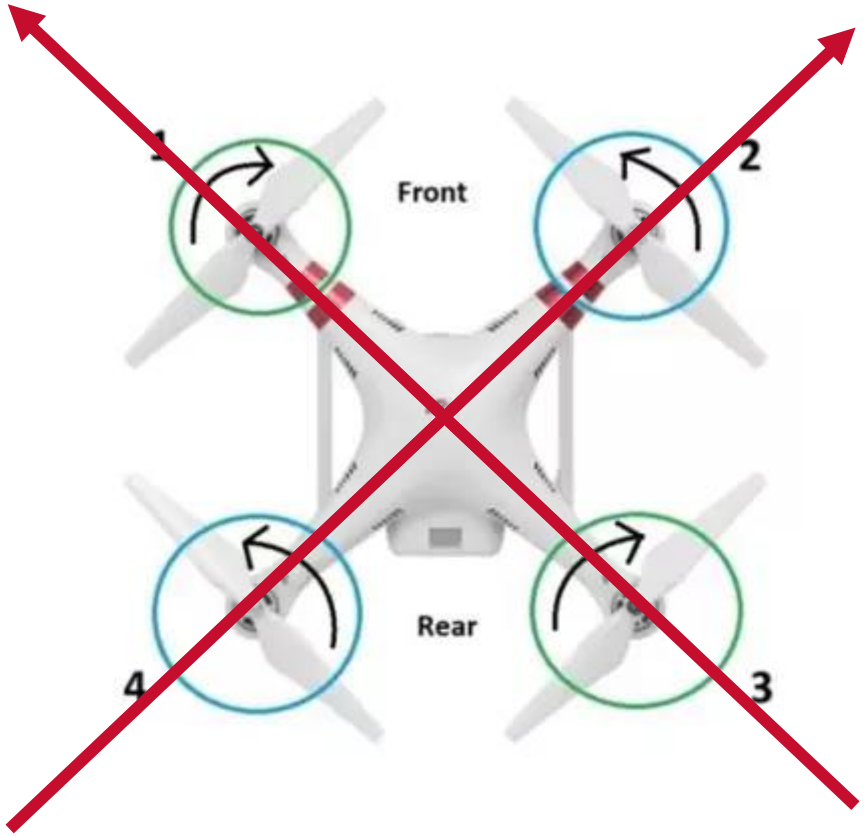
Figure 2: The two neural networks used in this work are shown.


T_1, T_2, T_3, T_4
p ϕ x $\dot{\phi}$ u
q θ y $\dot{\theta}$ v
r ψ z $\dot{\psi}$ w

$T_i = b\Omega_i^2 \Rightarrow$ *The thrust force*

$D_i = d\Omega_i^2 \Rightarrow$ *The drag moment*

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$
$$U_2 = lb(-\Omega_2^2 + \Omega_4^2)$$
$$U_3 = lb(\Omega_1^2 - \Omega_3^2)$$
$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$



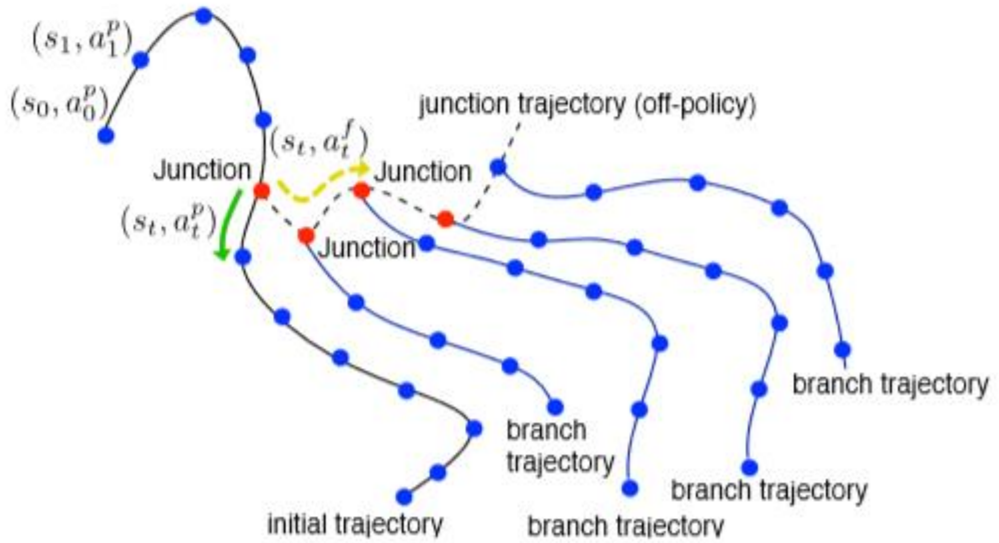
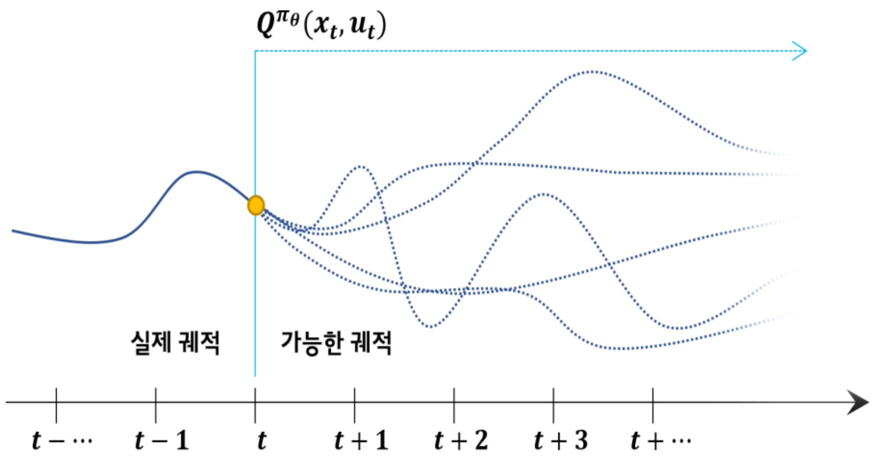


Figure 3: Exploration strategy.



$$A^{\pi}(s_i, a_i^f) = r_i^f + \gamma v_{i+1}^f - v_i^p,$$

$$\bar{L}(\theta) = \sum_{k=0}^{K} A^{\pi}(s_k, \pi(s_k|\theta)),$$

$$\theta_{j+1} \leftarrow \theta_j - \frac{\alpha}{K} \sum_{k=0}^{K} n_k,$$

$$s.t.\ (\alpha n_k)^T D_{\theta\theta}(\alpha n_k) < \delta, \quad \forall k,$$

where n_k is per-sample natural gradient defined as a vector that satisfies $D_{\theta\theta} n_k = g_k$, where D is the squared Mahalanobis distance and the double subscript means that it is a Hessian w.r.t. the subscripted variable. We use i for time index, k for junction index, and j for iteration index. We denote the on-

Advantage Function

$$A^{\pi_\theta}(x_t, u_t) = Q^{\pi_\theta}(x_t, u_t) - V^{\pi_\theta}(x_t)$$

$$A^{\pi_\theta}(x_t, u_t) \approx r(x_t, u_t) + \gamma V^{\pi_\theta}(x_{t+1}) - V^{\pi_\theta}(x_t)$$

$$L(\pi_\theta) = \mathop{\mathbb{E}}_{s_0, s_1 \dots} [\sum_{k=0}^{\infty} \gamma^k r_{t+k}]$$

where, $V^{\pi_\theta}(s) = \mathop{\mathbb{E}}_{s_{t+1}, s_{t+2} \dots} [\sum_{t=t} \gamma^t r_t | s_t = s, \pi_\theta]$

$$\theta^* = \operatorname{arg}max\, J(\theta)$$

$$J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T} \gamma^t r(x_t, u_t)\right] = \int p_\theta(\tau)\left(\sum_{t=0}^{T} \gamma^t r(x_t, u_t)\right) d\tau$$

$$p_\theta(\tau) = (x_0) \prod_{t=0}^{T} \pi(u_t | x_t) p(x_{t+1} | x_t, u_t)$$

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T}\left(\nabla_\theta \log \pi_\theta(u_t | x_t)\left(\sum_{k=t}^{T} \gamma^t r(x_k, u_k)\right)\right)\right]$$

$$= E_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T}\left(\gamma^k \nabla_\theta \log \pi_\theta(u_t | x_t)\left(\sum_{k=t}^{T} \gamma^{k-t} r(x_k, u_k)\right)\right)\right]$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \qquad G_t = \sum_{k=t}^{T} \gamma^{k-t} r(x_t, u_t)$$

$$\nabla_\theta J(\theta) \approx \nabla_\theta \frac{1}{M} \sum_{m=1}^{M}\left[\sum_{t=0}^{T}\left\{ \log \pi_\theta\left(u_t^{(m)} | x_t^{(m)}\right) G_{t(m)} \right\}\right]$$

$$loss = -\sum_{t=0}^{T}\left(\log \pi_\theta\left(u_t^{(m)} | x_t^{(m)}\right) G_t^{(m)}\right) \quad ,$$

$$\because \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \approx \theta - \alpha \nabla_\theta \sum_{t=0}^{T}\left(\log \pi_\theta\left(u_t^{(m)} | x_t^{(m)}\right) G_t^{(m)}\right)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{M}\sum_{m=1}^{M}\sum_{t=0}^{T}\left(\nabla_\theta \log\pi_\theta\left(u_t^{(m)}|x_t^{(m)}\right)A^{\pi_\theta}\left(x_t^{(m)},u_t^{(m)}\right)\right)$$

$$\approx \frac{1}{M}\sum_{m=1}^{M}\sum_{i=t}^{t+N-1}\left(\nabla_\theta \log\pi_\theta\left(u_i^{(m)}|x_i^{(m)}\right)A^{\pi_\theta}\left(x_i^{(m)},u_i^{(m)}\right)\right)$$

$$E_{x\sim p(x)}[f(x)] = E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]$$

$$\frac{p_\theta(x_t)}{p_{\theta_{old}}(x_t)} \approx 1$$

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T}\left(E_{x_t\sim p_{\theta_{old}}(x_t)}\left[E_{u_t\sim \pi_{\theta_{old}}(u_t|x_t)}\left[\gamma^t \frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}\nabla_\theta \log\pi_\theta(u_t|x_t)A^{\pi_\theta}(x_t,u_t)\right]\right]\right)$$

$$J(\theta) - J(\theta_{old}) = \sum_{t=0}^{\infty}E_{\tau\sim p_\theta(\tau)}\left[\gamma^t A^{\pi_{\theta_{old}}}(x_t,u_t)\right]$$

$$\sum_{t=0}^{\infty}E_{\tau\sim p_\theta(\tau)}\left[\gamma^t A^{\pi_{\theta_{old}}}(x_t,u_t)\right] = \sum_{t=0}^{\infty}E_{x_t\sim d_\theta(x_t)}\left[E_{u_t\sim \pi_\theta(u_t|x_t)}\left[A^{\pi_{\theta_{old}}}(x_t,u_t)\right]\right]$$

$$L(\theta) = J(\theta) - J(\theta_{old}) = \sum_{t=0}^{\infty}E_{x_t\sim \pi_{\theta_{old}}(x_t)}\left[E_{u_t\sim \pi_{\theta_{old}}(u_t|x_t)}\left[\frac{\pi_\theta(u_t|x_t)}{\pi_{\theta_{old}}(u_t|x_t)}\gamma^t A^{\pi\theta_{old}}(x_t,u_t)\right]\right]$$

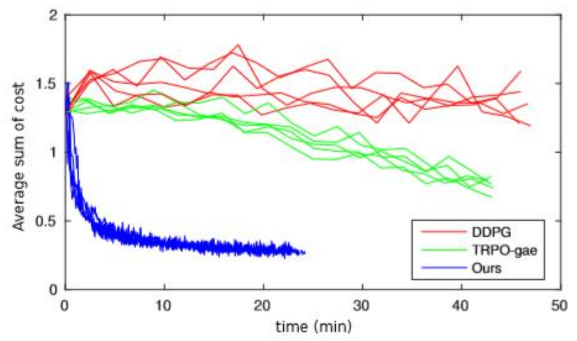


Figure 4: Learning curves for optimizing the policy for three different algorithms and 5 different runs per each algorithm.

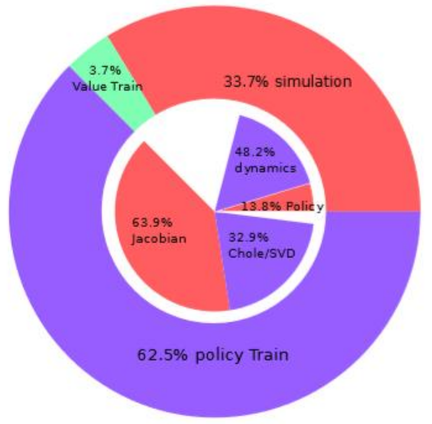


Figure 5: Computation resource consumption is shown. The outer ring represents the fraction of each categories and the inner ring is for that of their subcategories.

Table I: Quadrotor physical parameters

Parameter	value
mass	0.665 kg
dimension	0.44 m, 0.44 m, 0.12 m
I_{xx}, I_{yy}, I_{zz}	0.007, 0.007, 0.012 kgm^2

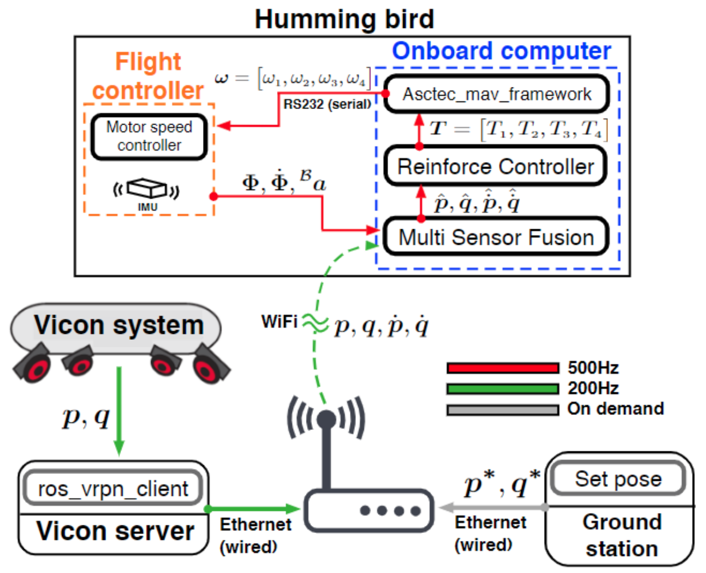
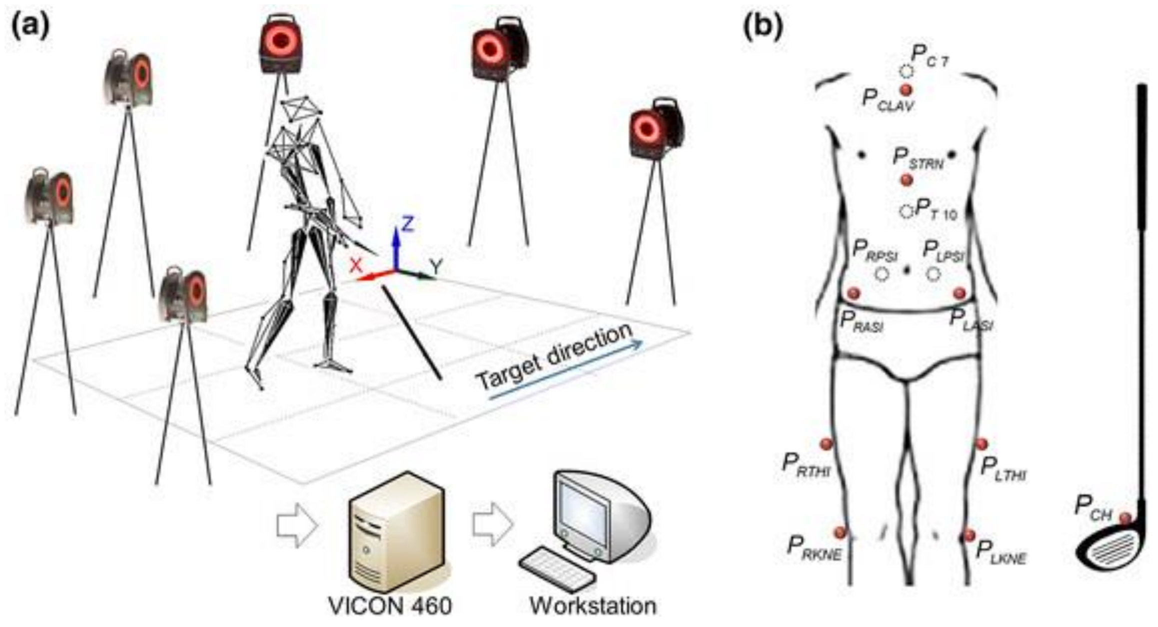


Figure 6: System diagram used for the experiments. Different colors denote the corresponding rate respectively. $\boldsymbol{p}, \boldsymbol{q}, \dot{\boldsymbol{p}}, \dot{\boldsymbol{q}}$ are position, orientation and their velocities. $\boldsymbol{p}^*$ and $\hat{\boldsymbol{p}}$ denote the desired goal and estimated position. $\boldsymbol{T}$ and $\boldsymbol{\omega}$ are thrust in N and rotor speed in rad/s. $\boldsymbol{\Phi}, \dot{\boldsymbol{\Phi}}, {}^{B}\boldsymbol{a}$ represent IMU measurements; orientation, angular velocity, and linear acceleration in body frame.



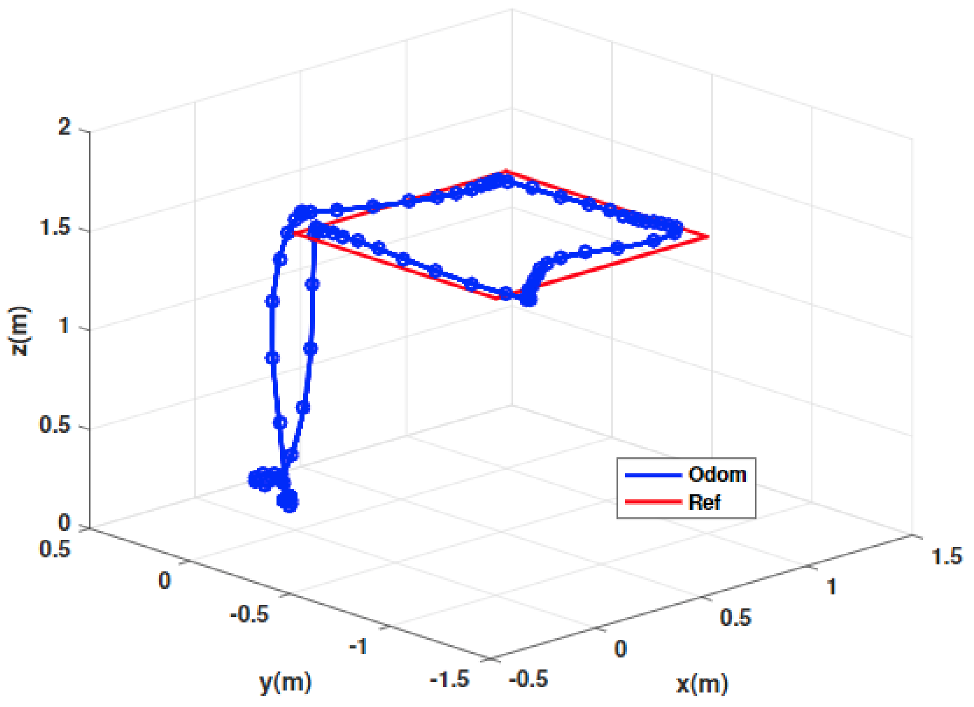


Figure 7: Trajectory while performing waypoint tracking.

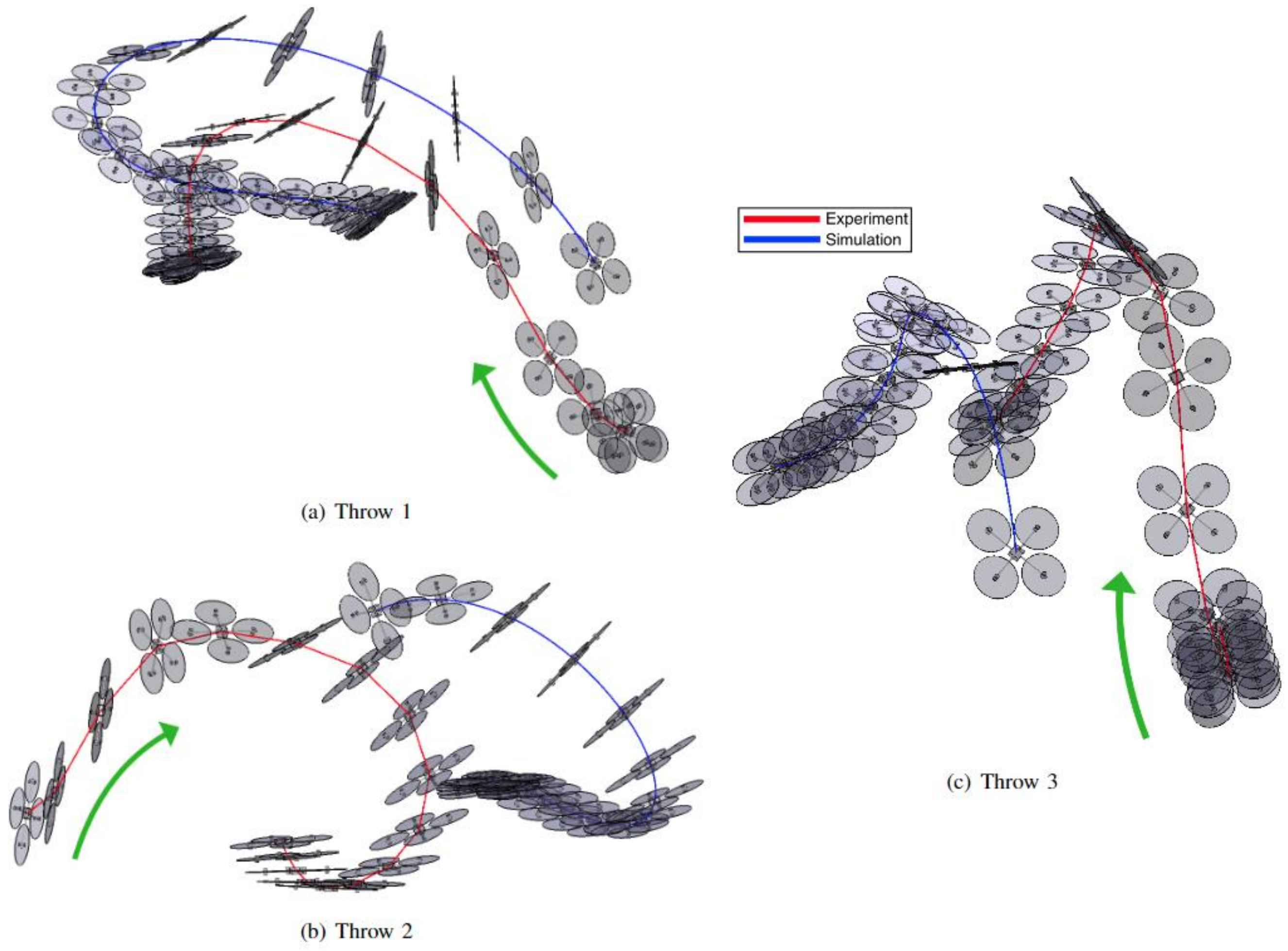


Figure 8: Experimental/simulation results are shown. All trajectories start close to upside-down configuration and the quadrotor flips over in midair. All launches started with initial velocity around 5 m/s. The stroboscopic images are generated with 0.1 s intervals.

끝