Proceedings of the ASME 2019
Dynamic Systems and Control Conference
DSCC2019
October 9-11, 2019, Park City, UT, USA

DSCC2019-9086

SELF-REFLECTIVE LEARNING STRATEGY FOR PERSISTENT AUTONOMY OF AERIAL MANIPULATORS

Xu Zhou, Xiaoli Zhang[1]
Colorado School of Mines
Golden, CO

Jiucai Zhang
Guangzhou Automotive Group R&D Center
San Jose, CA

ABSTRACT

Autonomous aerial manipulators have great potentials to assist humans or even fully automate manual labor-intensive tasks such as aerial cleaning, aerial transportation, infrastructure repair, and agricultural inspection and sampling. Reinforcement learning holds the promise of enabling persistent autonomy of aerial manipulators because it can adapt to different situations by automatically learning optimal policies from the interactions between the aerial manipulator and environments. However, the learning process itself could experience failures that can practically endanger the safety of aerial manipulators and hence hinder persistent autonomy. In order to solve this problem, we propose for the aerial manipulator a self-reflective learning strategy that can smartly and safely finding optimal policies for different new situations. This self-reflective manner consists of three steps: identifying the appearance of new situations, re-seeking the optimal policy with reinforcement learning, and evaluating the termination of self-reflection. Numerical simulations demonstrate, compared with conventional learning-based autonomy, our strategy can significantly reduce failures while still can finish the given task.

Keywords: self-reflective learning strategy, reinforcement learning, persistent autonomy, unmanned aerial manipulator

1. INTRODUCTION

Autonomous aerial manipulators have great potentials to assist humans through manipulations in dangerous or remote locations, and to automate manual labor-intensive tasks such as aerial cleaning, aerial transportation, infrastructure construction and repair, and agricultural inspection and sampling. Due to the varieties of the system configurations, tasks, and environments, it is challenging for aerial manipulators to achieve persistent autonomy which requires aerial manipulators to safely operate in dynamic or unstructured environments for extended lengths of time with minimal human interventions**.**

Reinforcement learning [1] holds the promise for persistent autonomy of aerial manipulators because it can adapt to different situations by automatically learning optimal policies from the interactions between the aerial manipulator and environments. However, failures can be unavoidable during the learning process because reinforcement learning can only learn the outcome of an action by executing the action itself. These failures can lead to serious safety issues in practical systems such as the crash of aerial manipulators, which is not acceptable in persistent autonomy, especially for safety-critical systems. Although human interventions could address some failures, they are also not desirable in persistent autonomy. Thus, it is ideal to avoid these failures during the practical deployment of reinforcement learning on persistent autonomy.

While some safe reinforcement learning algorithms [2] could reduce failures by adding safety concerns into the optimization criteria or restricting to safe exploratory actions, they were still restricted within the scope of reinforcement learning itself. In other words, they assumed that they were always in the learning/training process to find a new optimal policy for a new situation. However, in persistent autonomy, the old policy may still work or even remain optimal when the situation changes to a new one. In this case, activating learning to re-find the optimal policy is actually unnecessary and unbeneficial because re-learning can bring more failures. Hence, it is safer to not activate learning and use current policy to finish the task. Similarly, even if the learning is activated, it could be sometimes enough for reinforcement learning to only find a sub-optimal policy with less failures as long as this policy can finish the task. From this high-level perspective focusing on the system safety, the practical deployment of reinforcement learning on persistent autonomy actually raises a fundamental but

[1] Contact author: xlzhang@mines.edu

unanswered question: ***when to start learning a new policy and when to terminate the learning***?

While this question is difficult to robots, humans can well solve this problem with their *self-reflection* capability [3][4]. More specifically, humans prefer doing nothing and remaining current behaviors if they are satisfied with the outcomes of their decisions or actions. If not satisfied, however, humans are highly likely to start changing behaviors and continue exploring new, alternative actions until the outcomes satisfy them again. Inspired by this *self-reflection* process, we present a novel self-reflective learning strategy for the aerial manipulator to smartly and safely operate in different situations. This strategy includes three steps. The first step is self-determining if the aerial manipulator is encountering a new situation by evaluating the necessity degree to change the control policy. If the necessity degree value is low, the strategy keeps its current policy because no new situation is identified. Once a new situation is detected, in the second step, the strategy makes new explorations about the new situation by using reinforcement learning. The maximum number of iterations is determined by the necessity degree value. The third step is deciding when to stop self-reflection. The termination criteria are related to task performances. With these three steps, the output policy is considered as safer for the new situation.

We consider our main contributions as: (1) Develop a new cognitive learning strategy with a self-reflective architecture for the aerial manipulator to step towards persistent autonomy. While still maintaining the capability to finish a given task, this strategy focuses on improving the system safety by smartly deciding the timing of activating and terminating learning. (2) Demonstrate our strategy can experience substantially fewer failures when finishing a given task with simulations. The results also show that learning can be sometimes unbeneficial and unnecessary for a practical system to finish a given task when situation changes.

2. RELATED WORK

In general, any unmanned aerial vehicle (UAV) equipped with any degree-of-freedom (DOF) robotic arm can be regarded as an aerial manipulator. For simplicity, this work investigates the quadrotor equipped with a 3-DOF robotic arm (the quadrotor-arm system) as a specific example due to its popularity in aerial manipulators. However, our strategy can be also used in other aerial manipulators or robots because it does not rely on specific robot dynamics.

While several methods [5,6] have been reported for the quadrotor-arm system to autonomously finish a task, they were limited to a well-defined situation without changes. If the situation such as the task or environment changes, their performances may reduce greatly. Reinforcement learning has shown great adaptabilities to different unknown situations in robotics [1], but it has not been really investigated in the aerial manipulator, which could be difficult due to the aerial manipulator's high complexity and nonlinearity. More importantly, failures in the learning process can result in serious consequences on the aerial manipulator like the crash.

With regard to safety and/or risks during the learning and/or deployment process, two fundamental tendencies of safe reinforcement learning methods [2] have been reported. The first one consists of transforming the traditional optimization criteria that maximizes the expectation of the return to more comprehensive criteria respecting learning safety such as the worst-case criterion [2], the risk-sensitive criterion [7], the constrained optimization criterion [8], and other optimization criteria in financial engineering [9]. Since exploratory actions could have serious consequences, the second tendency improved the safety with modifying the exploration process in two ways: (1) through the incorporation of external knowledge such as providing initial knowledge [10], deriving a policy from a finite set of demonstrations [11], and providing teach advice [12], and (2) through the use of a risk-directed exploration metric [13]. However, these methods restricted themselves within the training process of reinforcement learning. In contrast, our strategy reduces failures from a more general system's perspective with understanding learning can be unnecessary and unbeneficial and deciding when to activate or terminate learning to avoid such unbeneficial learnings.

3. SELF-REFLECTIVE LEARNING STRATEGY

As shown in Figure 1, the self-reflective control strategy includes three important steps. The first step is to determine if a new situation is identified and the necessity of adaptation to the new situation. The identification criteria are usually related to the system's maximum capability and situation change degree. If no new situation is detected, the self-reflective strategy believes the current policy is functional and no more action is needed. However, when a new situation is detected, the self-reflective strategy uses reinforcement learning technique to re-seek an optimal policy. New policies are tried with approximate value iteration (AVI) reinforcement learning method [14]. The maximum number of trials or iterations is proportional to the value of the necessity degree value to have more chances to find an optimal policy. During this process, the self-reflection can be terminated if a termination criterion is met, which is usually the satisfaction of new policy's performance.

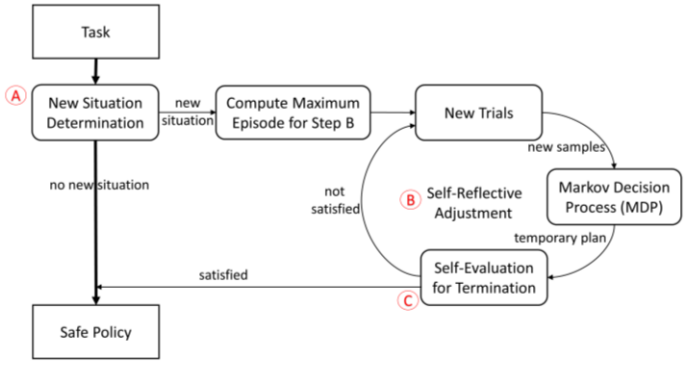

FIGURE 1: OVERALL SELF-REFLECTIVE CONTROL SCHEME

3.1 New Situation Determination

This step is to identify if there is a new situation that has not been met before. The basic idea is to compute a necessity degree value α based on the aerial manipulator system's maximum capability and situation change degree. As we use the quadrotor-arm system for example, the criteria can include the limit of the quadrotor's attitudes as well as the change of the quadrotor's CoG and end-effector's position. The limit of quadrotor's attitudes represents the maximum capability of the quadrotor-arm system to remain stable. The CoG change and position change indirectly reflect situation changes as different situations result in different variations of quadrotor's CoG and end-effect's position. If either of the two parameters is out of a reasonable range, indicating that the system behavior is strange and unexpected, then the current situation should be changed to a new one. The necessity degree value α, which is between 0 and 1, is calculated based on the deviation of the criteria from reasonable ranges. A threshold α_{th} is defined to determine if a new task is encountered and its value can be empirically defined for different manipulators. If $\alpha > \alpha_{th}$, then a self-reflection process is needed. Otherwise, the original policy is still considered as functional. Mathematically, α can be formulized as (1).

$$\alpha = \text{sigmoid}\begin{pmatrix} a_1(\phi - \phi_e) + a_2(\theta - \theta_e) \\ +a_3\left(\Delta C_Q - \Delta C_{Q_e}\right) + a_4 \frac{\Delta E_e}{\Delta E} \end{pmatrix} \quad (1)$$

where ϕ and θ indicates the quadrotor's roll and pitch angles, ΔC_Q is the change of quadrotor's CoG, ΔE is the end-effector's position change, a_i are the coefficients to adjust each term's importance, and all variables with subscript e are the critical values the system should obey. For example, $\phi_e = 60$ degrees means the quadrotor's roll angle is not supposed to exceed 60 degrees, otherwise the system is considered unstable. $\Delta E_e = 0.01$m indicates the end-effector is supposed to move effectively with at least a 0.01m change. Each criterion is normalized first, and their sum is then transferred into [0,1] with a sigmoid function. The reason choosing a sigmoid function is that any large difference between the actual measurements and comparison terms in (1) makes α approach 1 quickly, which means a new situation is more likely to appear. This can help detecting an unusual or new situation effectively.

3.2 Self-Reflective Adjustment

We formulate the quadrotor-arm self-reflective adjustment process as a Markov Decision Process (MDP), specified by the 5-tuple (S, A, $\{P_{sa}\}$, r, γ). Let S be the set of system states, where each state is a vector containing the positions, poses, and corresponding velocities. Let A be the set of actions the system can take from any state, where an action is a set of inputs including forces and torques. $\{P_{sa}\}$ describes the unknown state transition probabilities, capturing the dynamics of the system. Specifically, $P_{sa}(s')$ is the probability of transitioning to state $s' \in S$, given current state $s \in S$ and applied action $a \in A$. The reward function is defined as $r: S \longrightarrow \mathbb{R}$, representing the cost and reward structure of the state space. Let γ be a discount factor on rewards, chosen to be 0.99 in this case.

The appropriate action to take at a given state can be defined in terms of a fixed policy π, where $a = \pi(s)$. The total expected payoff (or value) of a given state can be defined as follows

$$V^{\pi}(s) = \mathbb{E}\begin{bmatrix} r(s_0) + \gamma r(s_1) + \\ \gamma^2 r(s_2) + \cdots | s_0 = s, \pi \end{bmatrix} \quad (2)$$

Equation (2) can formulate in the so-called Bellman Equations [27].

$$V^{\pi}(s) = r(s) + \gamma \textstyle\sum_{s' \in S} P_{sa}(s') V^{\pi}(s') \quad (3)$$

The optimal value function can then be defined as

$$V^{*}(s) = r(s) + \gamma \max_{a \in A} \textstyle\sum_{s' \in S} P_{sa}(s') V^{*}(s') \quad (4)$$

Value iteration is a commonly used RL technique where the optimal value function is iteratively found and used to compute an optimal policy [5]. The optimal policy can be obtained as

$$\pi^{*}(s) = \underset{a \in A}{\text{argmax}} \textstyle\sum_{s' \in S} P_{sa}(s') V^{*}(s') \quad (5)$$

However, the tabular representation of the value function is impractical or infeasible when the state and action spaces are large or continuous. Instead, we consider using approximate value iteration (AVI) method [14] to solve the above MDP problem. With this method, the value function is approximated with a linearly parameterized feature vector $F(s)$, which is specifically chosen for this problem including the quadrotor's displacement and its velocity magnitude, arm's end-effector displacement and its velocity magnitude. Mathematically, $F(s)$ can be expressed as $F(s) = [\|\boldsymbol{p}\|^2 \quad \|\dot{\boldsymbol{p}}\|^2 \quad \|\boldsymbol{p_e}\|^2 \quad \|\dot{\boldsymbol{p}}_{\boldsymbol{e}}\|^2]^T$. The reward function is designed to penalize the distance from the goal state for both the quadrotor and arm. The quadrotor's altitude is also penalized to provide a bounding box so that the whole system is enforced to stay above the ground. The form is $r(s) = [\beta_1 \quad \beta_2 \quad \beta_3][r_1(s) \quad r_2(s) \quad r_3(s)]^T$, where

$$r_1(s) = -\|\boldsymbol{p} - \boldsymbol{p}_r\|^2$$

$$r_2(s) = -\left\|\boldsymbol{p}_e - \boldsymbol{p}_{e_r}\right\|^2$$

$$r_3(s) = \begin{cases} -10000, & z < 0.3 \\ 0, & z \geq 0.3 \end{cases}$$

Because AVI is an iterative process, it only ends when the maximum iterative number (or maximum episode) is reached or the value function (3) is converged. The selection of maximum episode is important because it is better to be large enough to allow the value function to be converged. However, it cannot be too large if considering the practical availability. More specifically, when reinforcement learning cannot converge to an optimal policy for some new situations within the defined maximum episodes, increasing the maximum episode generally means a longer time, which may be not allowed in some situations. For example, a large maximum episode can give the aerial manipulator an ability to finish a new task in 15 minutes, but the task is required to be finished in 10 minutes. The sensitive 10-minutes requirement can make the selection of a large maximum episode meaningless. A convenient way to adjust the maximum episode can be based on an empirical constant and the necessity degree value α as in (6).

$$episode_{max}^{new} = \begin{cases} episode_{max}^{init} \times (1 + b\alpha), & if\ \alpha \geq \alpha_{th} \\ episode_{max}^{init}, & if\ \alpha < \alpha_{th} \end{cases} \quad (6)$$

where $episode_{max}^{init}$ is empirically set to be 10000, which corresponds to about 17 minutes using a PC with Intel i7-4790

dual Core CPU and 8G RAM. If there is no self-reflection (i.e., $\alpha < \alpha_{th}$), no maximum episode is needed due to no need of re-learning. However, we still assign a value ($episode_{max}^{init}$) for this situation just in case the system accidently learns a policy again when no self-reflection is needed. The coefficient b before α expands the maximum episode corresponding to the new level of situations. If α is close to 1, it means the current situation is very unlikely to be met before and hence the system may need a totally different policy. A new situation usually needs more trials, which means the convergence may be slower because more samples need to be collected to get the new situation well known. $\alpha = 1$ could increase the maximum episode to (1+b) of the empirical constant and we can still use a larger b to increase the maximum episode, which can be suitable for highly complex tasks that need a long time to learn out an optimal policy. In general, α_{th} mainly determines if re-finding control policy is needed. b and α decides how large the maximum episode will be.

3.3 Self-Evaluation for Termination

During the adjustment, we can use a self-evaluation module to decide whether the self-reflection should stop. In this paper, we define the stop criterion as (7).

$$TC = c_1\left(\left\|\boldsymbol{p_e} - \boldsymbol{p_{e_r}}\right\|^2 + \|\boldsymbol{p} - \boldsymbol{p_r}\|^2 + \|\boldsymbol{\Omega} - \boldsymbol{\Omega_r}\|^2\right) + c_2\left(t_s - t_{s_r}\right)^2 \quad (7)$$

where $\boldsymbol{p_e}$ is the end-effector's position, $\boldsymbol{p}$ is the quadrotor's position, $\boldsymbol{\Omega}$ is the quadrotor's attitude, t_s is the stabilization time, and all corresponding variables with subscript r are the desired values. c_1 and c_2 are two coefficients balancing the focus between the accuracy and stabilization time of quadrotor-arm system. If c_1 is larger, then the system focuses more on the accuracy of finishing a task. Otherwise, the system's stabilization time has a higher priority. It can be noticed that (7) and (1) have some overlapping terms such as ϕ and $\boldsymbol{\Omega}$. However, there are two major differences. One is the inclusion of more criteria about task performances such as t_s in (7). The other one is the different meaning of the similar comparison terms in (1) and (7). The comparison terms in (1) mean the system's maximum capability to keep stable, but those in (7) indicate a satisfactory performance of a given task. Equation (7) is generally stricter than (1). In addition, the termination criterion value (*TC*) has a value only if a new situation is considered, and the self-reflection can stop once *TC* is less than a threshold TC_{th}. If there is no new situation detected, *TC* is intentionally set to 0. The selection of TC_{th} is based on specific requirements of different users. For example, some users may only require the system not to crash, but some may need the system to perform the tasks with good performances like smaller deviation and shorter stabilization time. In general, TC_{th} should be smaller if the requirement is stricter. The use of TC_{th} may scarify some task performances as the learned policy does not necessarily need to be theoretically optimal. Instead, as long as the new policy can meet the requirement of finishing a task, it can be considered as safely functional, which implicitly considers the system's safety as an optimization criterion as well. Whatever TC_{th} is, for any specific case, there is always a maximum TC_{th}, defined as TC_{th_max}. TC_{th_max} represents the loosest termination criterion to keep a particular system stable, i.e., not crash. For a given system, TC_{th_max} is fixed and only depends on the system. As long as $TC \le TC_{th} \le TC_{th\_max}$, the learned control strategy can output an optimal control policy.

4. VALIDATION SIMULATIONS

In this work, all simulations are completed in MATLAB/Simulink. The parameters of the quadrotor-arm

TABLE I
QUADROTOR SIMULATION PARAMETERS

Parameters	Values	Descriptions
I_{bx}, I_{by}	1.24 kg · m²	Moment of inertia around quadrotor's X, Y axis
I_{bz}	2.48 kg · m²	Moment of inertia around quadrotor's Z axis
m_b	2 kg	Mass of quadrotor
γ_t	3.13*10^(-5)	Thrust factor
γ_d	7.5*10^(-7)	Drag factor
l	0.1 m	Distance between the rotor center and quadrotor center
m_1, m_2, m_3	0.1 kg	Mass of the first, second, and third link of the arm
L_1, L_2, L_3	0.1 m	Length of the arm's first, second, and third link
I_{x1}, I_{y1}	0.0025 kg · m²	Moment of inertia around first link's X, Y axis
I_{z1}	0.0075 kg · m²	Moment of inertia around first link's Z axis
I_{x2}, I_{y2}	0.0025 kg · m²	Moment of inertia around second link's X, Y axis
I_{z2}	0.0075 kg · m²	Moment of inertia around second link's Z axis
I_{x3}, I_{y3}	0.0025 kg · m²	Moment of inertia around third link's X, Y axis
I_{z3}	0.0075 kg · m²	Moment of inertia around third link's Z axis

TABLE II
SELF-REFLECTIVE CONTROL STRATEGY PARAMETERS

Parameters	Values
a_1, a_2, a_3, a_4	1
b	2
c_1, c_2	80, 20
γ	0.99
β_1, β_2, β_3	10000, 800, 50
α_{th}	0.5
TC_{th}	0.7
TC_{th_max}	0.95
ϕ_e, θ_e	60 degrees
ΔC_{Q_e}	0.15m
ΔE	0.01m
$\boldsymbol{p_{e_r}}$	[0.17, 0, 0.13] m
$\boldsymbol{p_r}$	[0, 0, 0.3] m
$\boldsymbol{\Omega_r}$	[0, 0, 0] degrees
t_{s_r}	10 s

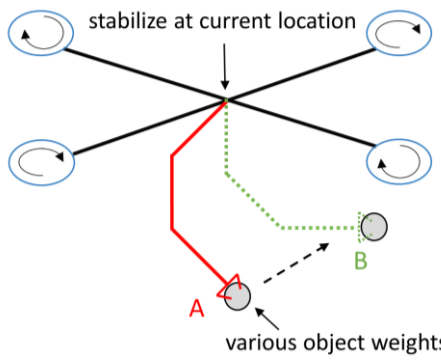

FIGURE 2: VISUALIZATION OF THE OBJECT-CARRY TASK

system [5] are listed in Table I. More modeling details about the quadrotor-arm system can be also seen in [5].

In order to validate the self-reflective strategy, the quadrotor-arm system is supposed to carry an object from position A ($\boldsymbol{\eta} = [-45, 45, 45]$ (degrees)) to position B ($\boldsymbol{\eta} = [0, 45, 45]$ (degrees)) as shown in Figure 2. According to the parameters in Table I, the quadrotor is supposed to be stabilized at a certain location ($\boldsymbol{p_r} = [0, 0, 0.3]$ (m), $\boldsymbol{\Omega_r} = [0, 0, 0]$ (degree)). The object weight can be 0.1kg, 0.15kg, 0.2kg, 0.25kg, and 0.3kg. An MDP controller is designed for a 0.1kg object beforehand, then the other weights can be considered as new tasks, depending on the self-reflective strategy's decision. Given the designed task, the parameters of the self-reflective strategy are listed in Table II. TC_{th_max} is assigned with 0.95 to remain the system stable. However, the task that is considered to be successfully finished should also include the object movement from A to B, which is why we choose TC_{th} to be 0.7.

5. RESULTS

5.1 New Situation Determination

The results of new situation determination for different weights are shown in Figure 3. The new situation determination values (reflection necessity degree values) for five different weights are 0.06, 0.20, 0.41, 0.84, and 1, respectively. The default threshold α_{th} (conservative threshold) is 0.5, so only two cases (0.25kg and 0.3kg) need self-reflections. However, if we decrease the threshold α_{th} to 0.3 (aggressive threshold), the 0.2kg case needs self-reflection as well. This is because a smaller α_{th} means more situations are considered as new situations, which indicates self-reflection is triggered easier (i.e., self-reflection is more needed). Instead of decreasing the threshold value α_{th}, another way to trigger self-reflection easier is to adjust constraints on the designed parameters, ϕ_e, θ_e, ΔC_{Q_e}, ΔE_e, in (1) to be stricter. More specifically, ϕ_e, θ_e, and ΔC_{Q_e} should be smaller while ΔE_e should be larger to cause an easier trigger. The result of reducing ΔC_{Q_e} from 0.15 to 0.05 can be found as the blue line in Figure 3. The necessity degree values of five weights are 0.09, 0.27, 0.55, 0.90, and 1, respectively. If

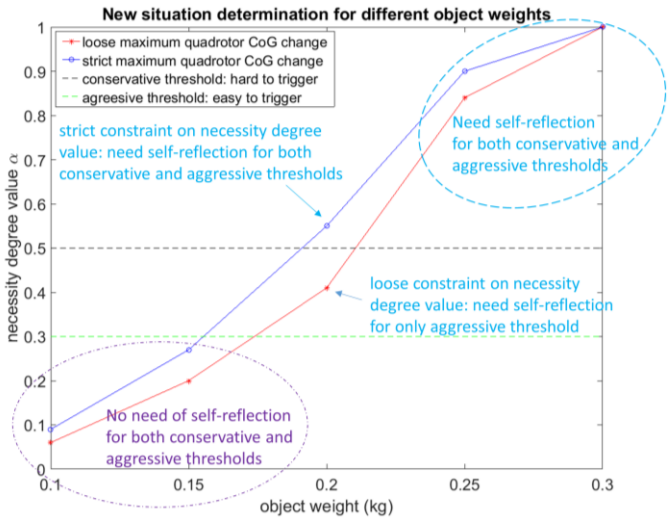

FIGURE 3: NEW SITUATION DETERMINATION FOR DIFFERENT OBJECT WEIGHTS. For 0.1kg and 0.15kg, there is no need of self-reflection as they are similar to pre-trained situations. For 0.25kg and 0.3kg, self-reflection is needed in all different parameter settings as they are greatly different from pre-trained situations. For 0.2kg, the necessity degree value is less with loose constraints than that with strict constraints. If given the conservative threshold, only the case with strict constraints needs self-reflection. But given an aggressive threshold, both cases with strict and loose constraints need self-reflection.

using the default threshold 0.5, one more case (0.2kg) needs self-reflection if compared with the loose maximum quadrotor CoG change (red line with $\Delta C_{Q_e} = 0.15$). It is hence convenient to adjust when a new situation should be considered or a self-reflection is needed.

5.2 Self-Reflective Adjustment

The change of maximum episode for different object weights is shown in Figure 4. Both the 0.1kg and 0.15kg case do not need self-reflection, so their maximum episodes are intentionally set to be the empirical constant 10000 according to (6). For the 0.2kg case, selections of parameters to calculate α and α_{th} are both important. If using the same α_{th}, a stricter selection of parameters in (1) needs to adjust its maximum episode because it needs self-reflection. Similarly, if keeping the selection of parameters the same, a small α_{th} needs to adjust its maximum episode too. For the 0.25kg case and 0.3kg case, they all need self-reflections and the maximum episodes of 0.3kg are all larger than 0.25kg because the necessity degree value α for 0.3kg is larger. In particular, the maximum episode of the 0.3kg case are all the same for four different combinations. This is because 0.3kg is too heavy for the quadrotor's current controller to handle, so ΔE in (1) equals to 0 and hence α is 1 and all maximum episodes become 30000.

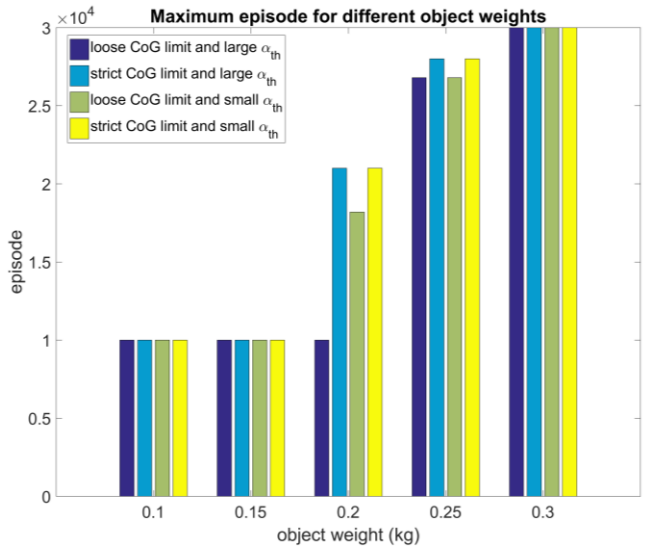

FIGURE 4: MAXIMUM EPISODE FOR DIFFERENT OBJECT WEIGHTS

5.3 Self-Evaluation for Termination

Table III shows termination criterion values and episodes with respect to different parameters. If using the default parameters in Table II, 0.1kg, 0.15kg and 0.2kg are not considered as new situations. Thus, their termination values are 0 and episodes used to find the optimal control policy are 10000. On the contrary, the termination criterion value for 0.25kg and 0.3kg are 0.68 and 0.69, respectively. The corresponding episodes are 20336 and 28212. More episodes are needed for 0.3kg because this case deviates more from the original controller designed for 0.1 kg object than 0.25kg and is hence much more difficult for the system to handle.

TC_{th} has an influence on the self-reflection results including final termination criterion value and episodes used for learning a new policy. Changing $\boldsymbol{p_r}$, $\boldsymbol{\Omega_r}$, t_{s_r}, c_1 ,and c_2 can also affect TC and so self-reflection results can be different. Because of various combinations of the related parameters, we only change c_1 and c_2 for simple examples. Also, both changes are applied to 0.25kg and 0.3kg cases because these two cases are easily switched between task success and task failure. The new termination criterion values and episodes are displayed in Table III after re-doing the simulations with new parameters. Reducing TC_{th} from 0.7 to 0.6 and keeping other parameters unchanged means the current task can be considered as successful only with better performances. With this new setting, the 0.25kg case can still work but the 0.3kg case fails because the termination criterion value can reach only 0. 66 within 30000 episodes. The failure means no optimal policy can be found with the given episodes and stricter performance criteria. If increasing the maximum episode number, the 0.3kg case may work again. For an additional test, we change c_1 to 20 and c_2 to 80, which means the controller focuses more on stabilization time. Although we use the same TC_{th} (0.7) as in Table II, the termination criterial values are still different from c_1=80 and c_2=20. The 0.25kg case needs 4766 more episodes, and the 0.3kg case fails. This is reasonable because limited time is a stronger constraint than good accuracy for the designed task if using the conclusion from the original 0.3kg case that the accuracy can be satisfied if given unlimited time.

TABLE III
TERMINATION CRITERION VALUE AND EPISODES COMPARISON

Object weight (kg)	Parameters	TC_{th_max}	TC_{th}	Termination criterion value	Episodes
0.1	c_1=80, c_2=20	0.95	0.7	0	10000
0.15	c_1=80, c_2=20	0.95	0.7	0	10000
0.2	c_1=80, c_2=20	0.95	0.7	0	10000
0.25	c_1=80, c_2=20	0.95	0.7	0.68	20336
0.3	c_1=80, c_2=20	0.95	0.7	0.69	28212
0.25	c_1=80, c_2=20	0.95	0.6	0.59	27482
0.3	c_1=80, c_2=20	0.95	0.6	0.66	30000
0.25	c_1=20, c_2=80	0.95	0.7	0.68	25102
0.3	c_1=20, c_2=80	0.95	0.7	0.74	30000

5.4 Task Performance

The task performances include both the quadrotor's performance and the arm's performance. For simplicity, only link 1's movements are shown in Figure 5 because link 1 has the most obvious movement. From this figure, it can be seen the original policy is good for 0.1kg. For the 0.15kg and 0.2kg cases, the overshoots and stabilization time increase greatly. Especially for the 0.2kg case, it is close to the unstable status. The unstable 0.25kg case is considered as a new situation and then a self-reflection starts. Although the performance after self-reflection cannot catch 0.1kg's performance, it is much better than 0.15kg

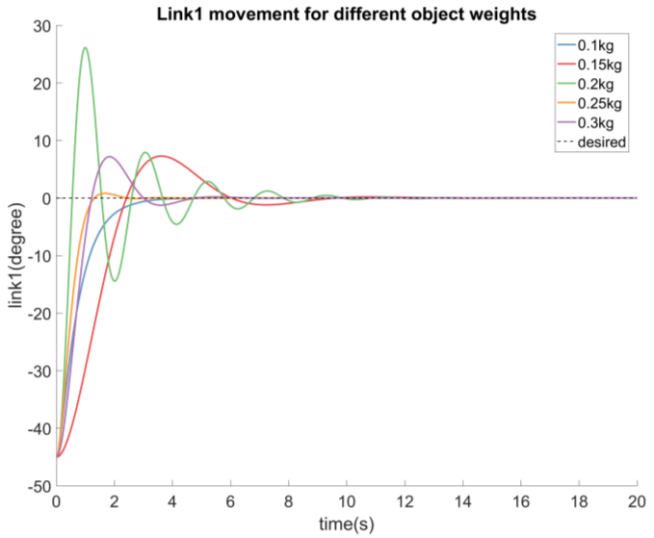

FIGURE 5: LINK1'S MOVEMENT FOR DIFFERENT OBJECT WEIGHTS. The movements with 0.15kg and 0.2kg are still acceptable using the pre-trained policy for 0.1kg. But the movement with 0.2kg is close to unstable due to the large overshoot and long stabilization time. Because self-reflection is applied for 0.25kg, the movement is stable again with a smaller overshoot and shorter setting time. The movement with 0.3kg becomes worse with using the strategy for 0.25kg but without self-reflection.

and 0.2kg. Carrying a 0.3kg object also needs self-reflection but the performance is worse than that of 0.25kg. This indicates that the self-reflection has a maximum capability to deal with new situations. If the situation difference is out of a system's maximum capability, the self-reflective strategy may not work. We keep increasing the object weight until the self-reflective strategy totally fails with settings in Table II. The corresponding maximum object weight is 0.34kg, which means the aerial manipulator fails to move an object over 0.34kg from A to B with an expected accuracy and time, but it does not mean that the aerial manipulator crashes since TC could still be smaller than TC_{th_max}.

In order to present the self-reflection process, the task performance is quantified with integral time squared error (ITSE). ITSE is a comprehensive evaluation criterion because it covers the overshoot, stabilization time, rising time, and the like. In detail, the task performance is calculated using $task\ performance = \sum_{dof=1}^{9} 100/\int_0^\infty te_{dof}(t)^2\,dt$, where 100 is the user-defined number to get the scale back into a reasonable range [0,1]. The summation is needed because ITSE can only apply to 1 DOF, and the aerial manipulator has 9 DOFs. The greater the ITSE value is, the better performance the system has. The task performances for different object weights can be seen in Figure 6. If not using the self-reflective strategy, the performance keeps decreasing as the object weight increases. According to (1), the performance drops from 0.1kg to 0.15kg and 0.15kg to 0.2kg are acceptable. But it is not acceptable with the drop from 0.2kg to 0.25kg, so this is when self-reflection is needed. With applying self-reflection, the performance is increased back to the acceptable range. Since self-reflection is not kept used during the change from 0.25kg to 0.3kg, the performance drops again although it is still acceptable.

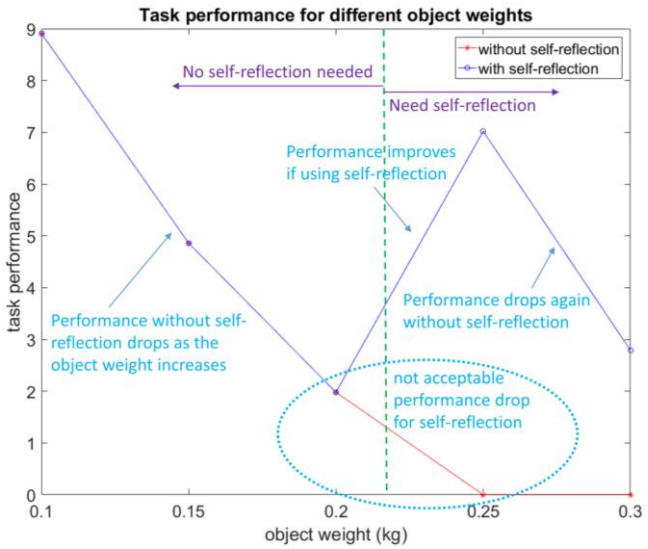

FIGURE 6: TASK PERFORMANCES FOR DIFFERENT OBJECT WEIGHTS. If self-reflection is not triggered, the task performance decreases as the object weight increases from 0.1kg to 0.2kg. Once self-reflection is needed and applied at 0.25kg due to an unacceptable performance drop, the task performance increases back to be acceptable. When addressing 0.3kg, the task performance drops again because self-reflection is determined not to be used.

5.5 Failure Costs and Computational Costs

Table IV shows the failure costs and computational costs of two different methods: (a) always using reinforcement learning with the maximum episode to be 30000 and (b) self-reflective method with the default parameters in Table II. Always using reinforcement learning means that reinforcement learning is always applied again for each time the situation changes. The definition of the failure is simplified as either ϕ or θ exceeds its limitation defined in Table II, which can be considered as a failure of physical instability. Also, a large deviation from the goal with $r(s) < -10000$ can be considered as a task failure. The computational cost is defined as the episodes used to converge to an optimal policy. The total numbers of failures and episodes using our method are much less than the corresponding numbers using always reinforcement learning. In most cases, our method has zero failures due to no need of re-learnings which are however necessary in always reinforcement learning. Even for the 0.25kg case that our method also re-learns, our method still has less failures and computational costs because the self-termination removes some unnecessary re-learnings. Given that both methods can finish the task, these re-learnings are actually unbeneficial because they result in more computational costs and failures.

5.6 Experimental Approach Stability Analysis

In order to experimentally analyze the stability of our method, we conduct one more simulation with the following scenario. After the self-reflective method adapts to the 0.25kg object, we test the five different object weights (0.1kg to 0.3kg) again on the newly learned policy. The corresponding reflection necessity degree values are 0.95, 0.48, 0.25, 0.05, 0.19, respectively. According to the default threshold α_{th}=0.5, only the 0.1kg object is seen as the new situation. This simulation experimentally shows that our method can be stable in dealing with new situations. In addition, it is interesting to note that a same weight change in the weight decreasing process has a larger necessity degree value than that in the weight increasing process. For example, the necessity degree value with the weight change from 0.25kg to 0.2kg is 0.06 larger than that with the change from 0.25kg to 0.3kg. This may indicate the weight decreasing

TABLE IV
FAILURE COSTS AND COMPUTATIONAL COSTS

Object Weight Change	Methods	Number of Failures	Episodes for Convergence
0.1kg to 0.15kg	Always reinforcement learning	3549	21448
	Self-reflective method	0	0
0.15kg to 0.2kg	Always reinforcement learning	4356	25633
	Self-reflective method	0	0
0.2kg to 0.25kg	Always reinforcement learning	5013	28702
	Self-reflective method	3208	20336
0.25kg to 0.3kg	Always reinforcement learning	5482	30000
	Self-reflective method	0	0

situations can be more likely to be considered as new situations than the weight increasing situations. One possible reason is that the policy designed for the old, heavy objects could be too strong for the new, light objects, resulting in larger deviations in (1).

6. DISCUSSIONS

Although our strategy uses reinforcement learning as the second step to ensure the system's adaptability to different situations, it more importantly improves the system's safety during the adaptation, which is important for practical safety-critical systems and results in significant differences between our strategy and traditional reinforcement learning methods [15]. The learned policies of traditional reinforcement learning based methods usually do not work for a new situation unless with a re-learning process. It is possible to keep reinforcement learning all the time to adapt to new situations, but the safety of adaptions may be compromised because introducing more learnings may lead to more failures due to possibly unavoidable unsafe explorations and intolerably high computational costs. Similarly, it is sometimes not safe to find a theoretically optimal policy for a task because the extra learnings from a sub-optimal policy that can already finish a task but may sacrifice some task performances to the theoretically optimal policy may also introduce more failures. With considering the safety from the whole system's perspective, our strategy can finish the task more safely by smartly avoiding the unnecessary and unbeneficial learnings with using self-detection (step one) and self-termination (step three),

One limitation of our work is that our strategy cannot guarantee zero failures because it still needs reinforcement learning to learn a new policy if the situation has a significant change. Considering that safe reinforcement learning methods [6] can reduce failures in the learning process, our strategy may be further improved by complementarily using safe reinforcement learning methods in the second step. In addition, introducing safe reinforcement learning may also relax the self-termination criteria to find a policy that is both theoretically optimal and safe. The combination of our strategy and safe reinforcement learning could be a promising way to achieve the persistent autonomy.

Another limitation of our work is the use of heuristic criteria for the self-detection and self-termination. This can require specific engineering knowledge for different aerial manipulators, although the criteria once designed can be used for a long time for different tasks and environments. This could be improved by automatically formulating different criteria using intrinsically-motivated learning [16]. Furthermore, some criteria that can be used for many situations may be stored in a memory so that it can be convenient to directly use them when the system encounters a similar situation in the future.

7. CONCLUSION

In this paper, a novel self-reflective learning strategy is developed for aerial manipulators to pursue persistent autonomy with mainly solving the problem of a smart and safe adaptation to new situations. While using reinforcement learning to achieve good self-adaptations, our method, more importantly, improves the system safety during the adaptations by determining the timing of triggering and terminating learning to reduce unnecessary and unbeneficial learnings. Numerical simulations using an aerial manipulator to carry different loads confirm the effectiveness of our method. In the future, we plan to apply our strategy in the physical aerial manipulator platform to get further validations. We also expect to replace the heuristic self-reflective criteria with more sophisticated criteria that may be changed online.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1652454.

REFERENCES

[1] Sutton, R. S., and Barto, A. G., 1998, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[2] Garcia, J., and Fernandez, F., 2015, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research,* vol. 16, no. 1, pp. 1437-1480.

[3] Grant, A. M., 2001, "Rethinking psychological mindedness: Metacognition, self-reflection, and insight," *Behaviour Change*, vol. 18, no. 1, pp. 8-17.

[4] Berzonsky, M. D., and Luyckx, K., 2008, "Identity Styles, Self-Reflective Cognition, and Identity Processes: A Study of Adaptive and Maladaptive Dimensions of Self-Analysis," *An International Journal of Theory and Research*, vol. 8, no. 3, pp. 205-219.

[5] Lippiello, V., and Ruggiero, F., 2012, "Cartesian impedance control of a UAV with a robotic arm," in *10th International IFAC Symposium on Robot Control*, Dubrovnik, Croatia, September 5-7, 2012.

[6] Orsag, M., Korpela, C., and Oh, P., 2013, "Modeling and control of mm-uav: mobile manipulating unmanned aerial vehicle," *Journal of Intelligent and Robotic Systems*, vol. 69, pp. 227–240.

[7] Geibel, P., and Wysotzki, F., 2005, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108.

[8] Moldovan, T. M., and Abbeel, P., 2012, "Safe exploration in markov decision processes," in *Proceedings of the 29th International Conference on International Conference on Machine Learning (ICML)*, pp. 1451-1458.

[9] Kashima, H., 2007, "Risk-sensitive learning via minimization of empirical conditional value-at-risk," *IEICE TRANSACTIONS on Information and Systems*, vol. 90, no. 12, pp. 2043–2052.

[10] Driessens, K., and Dˇzeroski, S., 2004, "Integrating guidance into relational reinforcement learning," *Machine Learning*, vol. 57, no. 3, pp. 271–304.

[11] Abbeel, P., Coates, A., and Ng, A. Y., 2010, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639.

[12] Garcia, J. and Fernandez, F., 2012, "Safe exploration of state and action spaces in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 515-564.

[13] Gehring, C., and Precup, D., 2013, "Smart exploration in reinforcement learning using absolute temporal difference errors," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems (AAMAS)*, pp. 1037-1044.

[14] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., 2010, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, 1st edition, Boca Raton, FL, USA: CRC Press, Inc.

[15] Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M., 2017, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters,* vol. 2, no. 4, pp. 2096-2103.

[16] Baldassarre, G., and Mirolli, M., 2013, *Intrinsically Motivated Learning in Natural and Artificial System*, Springer, Germany.