Recent decades have witnessed a dramatic growth of Autonomous Unmanned Systems (AUSs) which have the ability to function independently without human interaction. Examples of these AUSs include self-driving cars, unmanned aerial vehicles (UAVs), unmanned ground vehicles (UGVs) and unmanned underwater vehicles (UUVs), etc. These systems have fundamentally changed the industries of transportation, manufacturing and logistics [1].

One of the critical components in AUSs is control software, which is responsible for controlling the behaviours of AUSs to complete given tasks, while satisfying multiple critical requirements, including safety, security, privacy, etc. [2]. During operation, the control software of the AUS is required to handle conditions that differ from the ones it was initially designed for, for example because hardware elements of AUSs like sensors that degrade over time, becoming less accurate or consuming more energy than expected [3]. Moreover, the AUS should execute with incomplete knowledge of the operating environment (e.g., unknown obstacles and privacy restricted regions [4]) and face changing or updated goals at runtime [5]. The effects produced by these sources of uncertainty can compound and thereby inhibit the system from fully satisfying its requirements, i.e., the emergency of goal violation [6]. For example, to avoid new detected privacy restricted regions, a UAV may not be able to reach its destination on time [4]. To preserve the battery, a robot may decrease the localization accuracy when its sensors degrade over time, consuming more energy than expected [3]. Therefore, solutions to handling multiple goals online in response to those composed sources of uncertainties are required for the optimal goal satisfaction overall.

To address those real-time uncertainties and ensure goal satisfaction, self-adaptation might be a useful strategy and can are mostly realized in a way that use the feedback from the system and environment to plan and execute appropriate actions accordingly and ensure that the requirements of software are met [7]. To guarantee the system is robust in the face of dynamic behaviours, control theory provides the possibility for designing robust adaptive systems that operate with formal guarantees, so that desired properties can be guaranteed by designed [8], [9]. So far, most research on control-based adaptation focused on ad-hoc solutions to control the lower-level elements and resources of the system (e.g., CPU, storage, bandwidth, etc.) [10]. These solutions require well understanding of mathematical system models and are often done on a per-problem basis, discouraging flexibility and generality. Thus, general and automated methods for robust control are proposed like AMOCS [11], SimCA [12], SimCA\* [5] and AMOCS-MA [13], etc. However, these methods have limitations in optimality and efficiency when handling with multiple goals simultaneously for AUSs to mitigate runtime uncertainties.

Existing approaches seldom considering of trade-offs between goals, which leads to a lack of guarantee for optimality of solutions. Besides, they usually employ predefined priorities to maximize the satisfaction of multiple goals [11], [13]. In AMOCS [11], goals are ordered and managed by a cascaded single-goal controller by priority, while the utility function to optimize in AMOCS-MA [13] are combined with a weighted sum is susceptible to biases as the weights predefined by human experts are usually subjective [14]. There is a non-negligible possibility of violating all the goals, even though the degree of violation of each goal is small. It is also possible that several feasible goals are sacrificed to guarantee goals with higher priority.

In addition, existing approaches merely consider runtime performance (e.g., computation overhead) to maximize the satisfaction of multiple goals. In the adaptation of AUS, the system can not only determine the physical motions [15] (e.g., acceleration, velocity), but also adjust the system configurations (e.g., sensor parameters) [3], [12]. Thus, the AUS is highly configurable, whose configuration space is exponentially large. For the AUSs to make timely decisions for the optimal goal satisfaction especially when the available computational capacity is limited, the runtime adaptation space should be reduced regardless of the number of goals to optimize [16] and millions of possible configurations [3].

To address those concerns, in this paper, we propose a new control-based goal adaptation approach (Captain), which focused on the adaptation of AUSs to autonomously detect and minimize of goal violations to ensure optimal goal satisfaction in response of runtime uncertainties. Specifically, Captain first utilizes Model Predictive Control (MPC) strategy to check whether all the goals can be achieved in the immediate future based on a goal satisfaction model established at design time. For a set of goals, Captain evaluates their satisfaction degrees and divides the goal set into two subsets: violated and non-violated goals. If the violated goal set is none, a feasible solution can be found such that all goals can be achieved under current situation. Otherwise, Captain transforms violated goals into objective functions and optimize their satisfaction by adjusting the expectation to enlarge the search domain, while non-violated goals are maintained as constraints to guarantee. As a result, an updated goal satisfaction optimization problem is dynamically generated, as only violated goals need optimization, Captain can guarantee the minimal adaptation space for real-time performance and overall goal satisfaction.

Captain is general-purpose and applicable to several AUS applications. It exploits a three-stepped control-based adaptation mechanism for mitigating uncertainties while guaranteeing optimality and efficiency in handling multiple goals: Goal Satisfaction Checking, Goal Violation Analysis and Goal Satisfaction Optimization. For the optimality and efficiency, Captain only transforms goals which predicted to be violated into objective functions to optimizes their satisfaction, while the satisfaction of other non-violated goals or critical attributes are viewed as constrains to guarantee. The adaptation space is scaled such that an optimal solution can be figured out efficiently. Simulation results show that it outperforms state-of-the-art control-based solutions (e.g., AMOCS-MA) in dealing with multiple goals for motion and configuration planning in complex environments. Captain is practical too: we demonstrate its effectiveness in completing real-world tasks. These tests were carried out by implementing Captain on a DJI Matrice 100 UAV.

The main contributions of our work are summarized as follows.

We formalize the behaviour model and requirement model for AUSs. In particular, the requirements are classified into three categories i.e., task, hard constraints and soft goals based on their characteristics and differential expectation of their satisfaction.

We propose a real-time requirement evaluation approach to detect goal violations and evaluate their degree of violation based on a goal satisfaction model. In particular, goal satisfaction is checked and analysed proactively, such that actions are taken to minimize such violation in advance of accidents.

We propose real-time and multi-objective planner which allows sensor reconfiguration and motion planning of AUSs simultaneously.

We conduct comprehensive experimental study of Captain in both simulators and real systems, using various AUSs, to evaluate the effectiveness and efficiency of our approach. Promising results have been shown that Captain can improve adaptation flexibility and optimal goal satisfaction with acceptable performance overhead.