Requirements-Driven Online Adaptation to Tackle Uncertainty for Autonomous Unmanned Systems

APPENDIX A
DETAILS OF Captain

A. Requirement Satisfaction Modeling

We formulate the functions to evaluate the requirement satisfaction in Section III-B. Here, we classified the soft goals into three categories ($\mathrm{DS}^1(\mathcal{X}_i)$, $\mathrm{DS}^2(\mathcal{X}_i)$, $\mathrm{DS}^3(\mathcal{X}_i)$) based on their relationship with target values, i.e., LESS THAN, MORE THAN and AS CLOSE AS POSSIBLE. Correspondingly, the functions to evaluate requirement satisfaction, and the detailed fuzzy membership functions for $\mathrm{DS}^1(\mathcal{X}_i)$, $\mathrm{DS}^2(\mathcal{X}_i)$, and $\mathrm{DS}^2(\mathcal{X}_i)$ are listed as follows:

$$\mathrm{DS}^1(\mathcal{X}_i) = \begin{cases} \frac{ub_i - \mathcal{X}_i}{ub_i - g_i} & g_i < \mathcal{X}_i \leq ub_i \\ 1 & \mathcal{X}_i \leq g_i \\ 0 & otherwise \end{cases}$$

$$\mathrm{DS}^2(\mathcal{X}_i) = \begin{cases} \frac{\mathcal{X}_i - lb_i}{g_i - lb_i} & lb_i \leq \mathcal{X}_i < g_i \\ 1 & \mathcal{X}_i \geq g_i \\ 0 & otherwise \end{cases}$$

$$\mathrm{DS}^3(\mathcal{X}_i) = \begin{cases} \frac{\mathcal{X}_i - lb_i}{g_{i,1}} & lb_i \leq \mathcal{X}_i < g_{i,1} \\ 1 & g_{i,1} \leq \mathcal{X}_i \leq g_{i,2} \\ \frac{ub_i - \mathcal{X}_i}{ub_i - g_i} & g_{i,2} < \mathcal{X}_i \leq ub_i \\ 0 & otherwise \end{cases}$$

B. Requirement Satisfaction Checking

To evaluate the requirement satisfaction online, as illustrated in Section III-B, we construct slacked requirement constraints $\psi_{i,k}$ for each requirement R_i at any time instant k. Thus, to evaluate the requirement satisfaction degree, is to figuring out the least sum of slack variables which satisfy all the slacked requirement constraints. If the sum of slack variables is zero, it means that all requirement can be satisfied at current time instant. The detailed constraints formulations are listed as follows:

$$\psi_{i,k}: \begin{cases} \mathcal{X}_i(k) \in [lb_i, g_i + \delta_{i,k}] \wedge \delta_{i,k} \in [0, ub_i - g_i] \\ \quad \forall R_i \in \mathcal{S}^1 \\ \mathcal{X}_i(k) \in [g_i - \zeta_{i,k}, ub_i] \wedge \zeta_{i,k} \in [0, g_i - lb_i] \\ \quad \forall R_i \in \mathcal{S}^2 \\ \mathcal{X}_i(k) \in [g_{i,1} - \zeta_{i,k}, g_{i,2} + \delta_{i,k}] \wedge \\ \zeta_{i,k} \in [0, g_{i,1} - lb_i] \wedge \delta_{i,k} \in [0, ub_i - g_{i,2}] \\ \quad \forall R_i \in \mathcal{S}^3 \\ \mathrm{DS}^0(\mathcal{X}_i(k)) = 1, \forall R_i \in \mathcal{H} \cup \mathcal{M} \end{cases}$$

In the UAV delivery scenario, the Requirements Satisfaction Checking problem can be formulated as follows:

$$\begin{aligned} \min_{s,\boldsymbol{\delta},\boldsymbol{\varepsilon}} \quad & f_k = \sum_{k=k_0+1}^{k_0+N} \Big(\sum_{o \in \mathcal{O}_k} \varepsilon_{o,k} + \sum_{c \in \mathcal{C}_k} \varepsilon_{c,k} \Big) + \frac{\delta_{\xi,T}}{\Delta - \Delta_t} \\ & + \frac{\delta_{E,T}}{E - E_t} + \frac{\varepsilon_{\varphi,T}}{A_t - A} \\ s.t. \quad & \forall k \in [k_0, k_0 + N - 1], \\ & \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{v}_k \tau, \\ & \boldsymbol{v}_k \in [\boldsymbol{v}_{min}, \boldsymbol{v}_{max}], \boldsymbol{w}_k \in [\boldsymbol{w}_{min}, \boldsymbol{w}_{max}], \\ & \boldsymbol{x}_{k_0+N+1} = \boldsymbol{x}_{k_0+N} + \boldsymbol{v}_{k_0+N}(T - k_0 - N)\tau = \boldsymbol{x}_d; \\ & \forall k \in [k_0 + 1, k_0 + N], \\ & 1 - \varepsilon_{o,k} \leq \mathcal{X}_{o,k},\ 0 \leq \varepsilon_{o,k} \leq 1, \forall o \in \mathcal{O}_k, \\ & 1 - \varepsilon_{c,k} \leq \mathcal{X}_{c,k},\ 0 \leq \varepsilon_{c,k} \leq 1, \forall c \in \mathcal{C}_k, \\ & \mathcal{X}_{\xi,T} \leq \Delta_t + \delta_{\xi,T}, 0 \leq \delta_{\xi,T} \leq \Delta - \Delta_t, \\ & \mathcal{X}_{E,T} \leq E_t + \delta_{E,T}, 0 \leq \delta_{E,T} \leq E - E_t, \\ & A_t - \varepsilon_{\varphi,T} \leq \mathcal{X}_{\varphi,T},\ 0 \leq \varepsilon_{\varphi,T} \leq A_t - A. \end{aligned}$$

C. Requirements Satisfaction Optimizing

Depends on the partition of *ε-Flexible* set of requirements ($\mathcal{R}_f$) and *ε-Inflexible* set of requirements ($\mathcal{R}_{nf}$), the *Requirements Satisfaction Optimizing* in Section III-B is to maximize satisfaction of soft goals in $\mathcal{R}_f$ while guarantee the achievement of requirements $\mathcal{R}_{nf}$. The requirements satisfaction optimization problem (Eq. (3)) can be formulated as below in details:

$$\begin{aligned} \min_{\boldsymbol{u}} h_k = & \sum_{\substack{R_i \in \mathcal{R}_M \cap \mathcal{R}_f \\ j \in \{1,2,3\}}} \frac{1}{N} \sum_{k=k_0+1}^{k_0+N} \mathrm{DS}^j(\mathcal{X}_i(k)) \\ & + \sum_{\substack{R_i \in \mathcal{R}_A \cap \mathcal{R}_f \\ j \in \{1,2,3\}}} \mathrm{DS}^j(\mathcal{X}_l(T)) \\ s.t. \, & \forall R_i \in \mathcal{R}_{\mathcal{M}} \cap \mathcal{R}_{nf}, \forall k \in [k_0 + 1, ..., k_0 + N], \\ & \mathrm{DS}^j(\mathcal{X}_i(k)) = 1,\ j \in \{1, 2, 3\}; \\ & \forall R_j \in \mathcal{R}_{\mathcal{A}} \cap \mathcal{R}_{nf},\ \mathrm{DS}^j(\mathcal{X}_i(T)) = 1,\ j \in \{1, 2, 3\}. \end{aligned}$$

Given the prediction horizon N, for requirement $R_i \in \mathcal{R}_{\mathcal{M}} \cap \mathcal{R}_f$, their satisfaction should be optimized during the predict horizon, i.e., $[k_0 + 1, k_0 + N]$, while for $R_i \in \mathcal{R}_{\mathcal{M}} \cap \mathcal{R}_{nf}$, their satisfaction should be maintained during this period. Similarly, for $R_i \in \mathcal{R}_{\mathcal{A}} \cap \mathcal{R}_f$, their satisfaction is optimized based on the estimation of the final state of

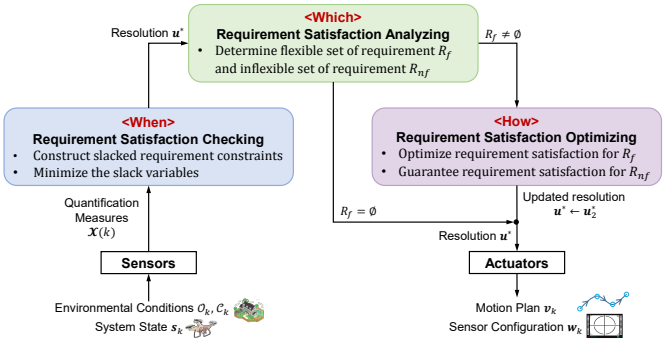


Fig. 1: Requirements-driven adaptation in UAV scenario.

AUS (assuming that AUS will keep the action plan $\boldsymbol{u}_{k_0+N-1}$ until the end of the mission at $k = T$), while guarantee $R_i \in \mathcal{R}_{\mathcal{A}} \cap \mathcal{R}_{nf}$ at the end of the mission.

APPENDIX B
UAV DELIVERY SCENARIO

In this part, we illustrate the details of implementation and supplemental experimental results for the UAV delivery case.

A. Overview

The workflow in UAV scenario is shown in Fig. 1. Based on the requirements set up at design time, at every time instant k, the AUS first monitor the changes in the environment and system itself and calculate Quantification Measures of requirements $\boldsymbol{\mathcal{X}}(k)$. To mitigate runtime uncertainties and achieve the optimal requirement satisfaction overall, `Captain` first check the satisfaction of each requirement based on the established requirement satisfaction functions $\mathcal{DS}$ to answer the question *when* the AUS fail to achieve all the requirements. After that, the analysis of unsatisfied requirement is launched and the initial requirement set is split into ε-*Flexible* and ε-*Inflexible* sets of requirements and only the soft goals in ε-*Flexible* set of requirements are relaxed and optimized. An updated requirement satisfaction optimization problem is generated where soft goals in ε-*Flexible* set are transformed into part of objective function and ε-*Inflexible* ones are viewed as constraints to guarantee, for the achievement of the optimal overall requirement satisfaction in response of uncertainties. Finally, motion plan $\boldsymbol{v}_k$ and sensor configuration $\boldsymbol{w}_k$ for the AUS to take are decided and deployed the through solving such optimization problem.

B. Requirements Satisfaction Modeling

We formulate the requirement satisfaction functions for requirements listed in Section IV.

- Safety: The indicator to evaluate the safety requirement is the collision risk of UAV during the flight. Supposing that the obstacles detected by the UAV at time instant k is $\mathcal{O}_k$, while the current state of UAV is s_k. Thus, the QM of safety is $\mathcal{X}_{S_{o,k}} = \frac{\|\boldsymbol{x}_k - \boldsymbol{x}_o\|_2 - r_a - r_o}{D_o}, \forall o \in \mathcal{O}_k$. Such that the average distance between UAV and the center of obstacle reflects the safety risk.

$$\mathrm{DS}^2(\mathcal{X}_{S_{o,k}}) = \begin{cases} 1, & \mathcal{X}_{S_{o,k}} \geq 1 \\ 0, & \mathcal{X}_{S_{o,k}} < 0 \\ \mathcal{X}_{S_{o,k}}, & otherwise \end{cases}$$

- Timelines: The total traveling time from time instant i to j is denoted as $\xi_{ij} = \sum_{k=i}^{j-1} \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2}{\boldsymbol{v}_k}$. The indicator of timeliness is $\mathcal{X}_\xi = \xi_{0T}$, the degree of satisfaction of the timeliness requirement of the whole trajectory DS_ξ is:

$$\mathrm{DS}^1(\mathcal{X}_\xi) = \begin{cases} 1, & \mathcal{X}_\xi \leq \Delta_t \\ \dfrac{\Delta - \mathcal{X}_\xi}{\Delta - \Delta_t}, & \Delta_t < \mathcal{X}_\xi \leq \Delta \\ 0, & \mathcal{X}_\xi > \Delta \end{cases}$$

- Accuracy: The average quality of the information collected during the mission is denoted as $\mathcal{X}_\varphi = \frac{1}{\xi_{0T}} \sum_{k=0}^{T-1} \|\boldsymbol{w}\| \tau$, the degree of satisfaction is DS_φ is:

$$\mathrm{DS}^2(\mathcal{X}_\varphi) = \begin{cases} 1, & \mathcal{X}_\varphi \geq A_t \\ \dfrac{\mathcal{X}_\varphi - A}{A_t - A}, & A \leq \mathcal{X}_\varphi < A_t \\ 0, & \mathcal{X}_\varphi < A \end{cases}$$

- Energy-saving: The total energy consumption from time instant i to j is denoted as $e_{ij} = \sum_{k=i}^{j-1} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 + \eta_1 \cdot \|\boldsymbol{v}_{k+1} - \boldsymbol{v}_k\|_2 + \eta_2 \cdot \|\boldsymbol{w}_k\| \tau$. The indicator of energy consumption is $\mathcal{X}_E = e_{0T}$, the degree of satisfaction of energy requirement DS_E is:

$$\mathrm{DS}^1(\mathcal{X}_E) = \begin{cases} 1, & \mathcal{X}_E \leq E_t \\ \dfrac{E - \mathcal{X}_e}{E - E_t}, & E_t < \mathcal{X}_e \leq E \\ 0, & \mathcal{X}_E > E \end{cases}$$

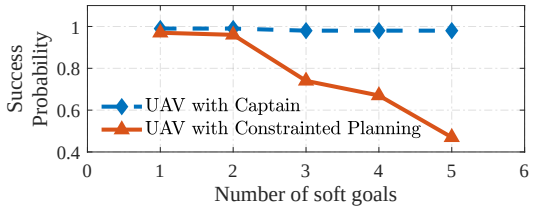


Fig. 2: Success rate with the number of soft goals increases.

C. Experiment Results

We show the details of the experimental results in the UAV delivery scenarios discussed in the paper.

1) Effectiveness: To illustrate that `Captain` can obtain solutions effectively with increasing numbers of soft goals, we simulated uncertain conditions and diverse scenarios with 100 randomly generated test cases from $\{\mathcal{S}_S\}$ to $\{\mathcal{S}_S, \mathcal{S}_P, \mathcal{S}_\xi, \mathcal{S}_\varphi, \mathcal{S}_e\}$. The test case is marked as fail if any hard constraint is violated or the mission is not finished. We also compared the results with the UAV implemented with constrainted planning where all of the requirements must

TABLE I: Requirements-driven adaptation results for different scales of environment

Scale		*Accuracy* [%]	*Traveling Time* [s]	*Energy Consumption* [unit]	*Safety Risk*	*Privacy Risk*	*Real-time Performance*
50	$\mathcal{S}$	90	60	100	0	0	Adaptation Rate
	$\mathcal{H}$	80	90	150	1	1	6/120
	$\mathcal{X}$	90	60.00	105.57	0	0	Overhead
	$\mathcal{DS}$	100%	100%	88.85%	100%	100%	Avg. 0.066s Std. 0.139s
100	$\mathcal{S}$	90	90	200	0	0	Adaptation Rate
	$\mathcal{H}$	80	150	300	1	1	3/203
	$\mathcal{X}$	90	101.50	210.94	0	0.1534	Overhead
	$\mathcal{DS}$	100%	80.83%	89.06%	100%	99.98%	Avg. 0.132s Std. 0.180s

be totally satisfied. The success rate of constrained planning decreases when the number of soft goals increases, while `Captain` can maintain a high success rate, greater than 95%. It is because based on requirements modeling, requirements are classified into soft goals and hard constraints, and it is allowed to transform violated soft goals into optimizable objectives flexibly when necessary.

2) Scalability: To demonstrate the scalability of `Captain` with respect to different environments, we further simulated the UAV case on two selected real urban environments from the open building dataset of Portland in USA [1]. We used ArcGIS map to set up a 3D model based on the method in [2]. Their original spaces are $500 \times 500 \times 100m^3$ and $10^3 \times 10^3 \times 100\ m^3$, and compressed into $50 \times 50 \times 10\ m^3$ and $100 \times 100 \times 10\ m^3$, respectively. In the dataset, we can also obtain the longitude and latitude of the center of each building, as well as its average height and building types (i.e., industrial and commercial buildings, houses and apartments for living). The buildings for industrial and commercial use are viewed as obstacles, while houses and apartments are viewed as private regions. We used ArcGIS map to set up a 3D model according to the method introduced in [2]. As shown in Fig. 4, the process of real urban scenarios modeling is explained. We set the range of longitude and latitude of working space on the ArcGIS map, and all the building are marked as blue (Fig. 4(a)). Then the ArcGIS map is turned into a binary map. Thus a 3D model of the urban scene is set up with the scale of $500m \times 500m \times 100m$ in Fig. 4(c).

In each case, the flight task of UAV is to travel from the position $[0, 0, 0]$ to the destination $[49, 49, 0]$ and $[99, 99, 0]$ respectively, within the budget of accuracy, time and energy, as well as minor safety and privacy risk. The trajectory, state transition and requirement achievement of the UAV at each time instant are shown in Figure 4(d), Figure 4(e) and Figure 4(f). So as to the scale of $1000m \times 1000m \times 100m$ in Figure 5. Table I summarizes the requirement adaptation results in these two settings. we find that the states generated from `Captain` and the PD controller are almost the same, indicating that the planning results of `Captain` can be translated by the PD controller and executed by the UAV effectively.

From our simulation results, in Setting 2, the flight task is completed in 60.07 s, consuming 106.27 units of energy without safety and privacy risk. In Setting 3, the task is finished in 101.5 s, consuming 210.94 units of energy. It does not have safety risk, but gets 4 points along the trajectory where the soft goal of privacy-preserving is violated. This results in an average 99.98% requirements satisfaction along the path. `Captain` has high scalability as the two important steps *Requirements Satisfaction Checking* and *Requirements Satisfaction Optimization* are solved by SQP, which can handle large-scale optimization problems. In contrast, AMOCS-MA fails to compute a motion plan at real-time when the size of workspace increases.

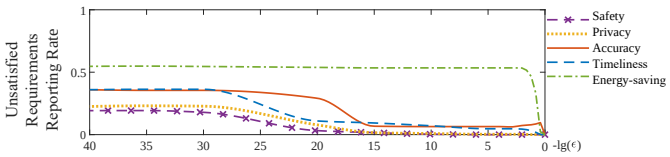
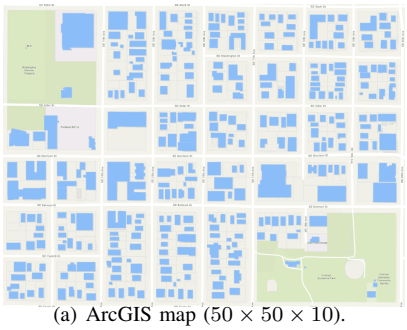


Fig. 3: Unsatisfied requirements reporting rate. (UAV case, $\rho_o = 2\%, \rho_p = 2\%$)

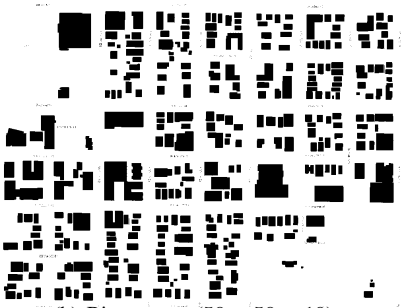
3) Choice of violation tolerance: Given a violation tolerance, the average rate of time instants violating soft goals is recorded per simulation, as shown in Fig. 3. We choose $\epsilon_{SR,PR,\varphi,\xi,e} = \{10^{-20}, 10^{-20}, 10^{-10}, 10^{-20}, 0.005\}$ for the UAV case, the average rates of time instants that do not report violation of the soft goals of safety, privacy, timeliness, accuracy, and energy-saving in the UAV scenario are 16.0%, 14.7%, 29.0%, 25.5%, and 1.1% respectively. These results show that choosing the right violation tolerance parameters can help determine the appropriate set of unsatisfied requirements with a smaller impact on the algorithm's efficiency. In the case where the choice of tolerance parameter may affect the convergence in the *Requirements Satisfaction Optimization* step and result in no feasible solutions, `Captain` can directly leverage the planning results from the *Requirements Satisfaction Checking* step, as illustrated in lines 13 and 14 in Algorithm 1.

APPENDIX C
UUV OCEANIC SURVEILLANCE

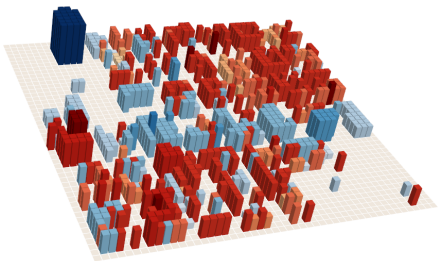
This example originates from [3], [4]. Rather than considering single-objective optimization for the UUV scenario in [3], [4], we extended it to achieve multiple dynamic requirements under uncertainties and disturbances through `Captain`, while other configurations like sensors are kept the same as [3]. The requirements to achieve in this scenario are listed as follows:



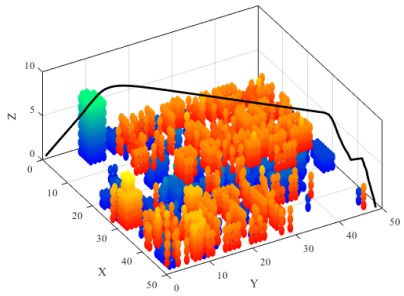
(a) ArcGIS map ($50 \times 50 \times 10$).



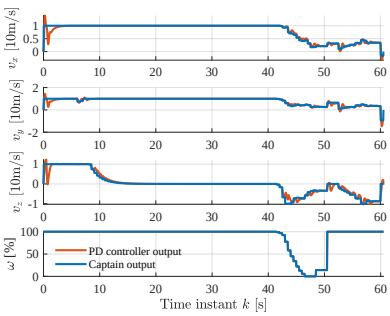
(b) Binary map ($50 \times 50 \times 10$).



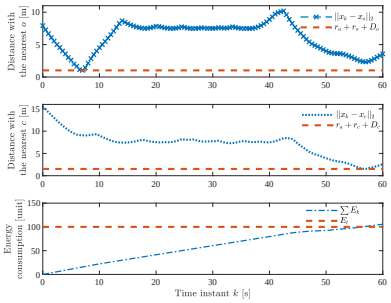
(c) Grid map ($50 \times 50 \times 10$).



(d) Flight Trajectory of UAV.

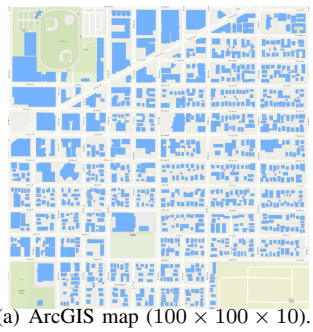


(e) State of UAV during the flight.

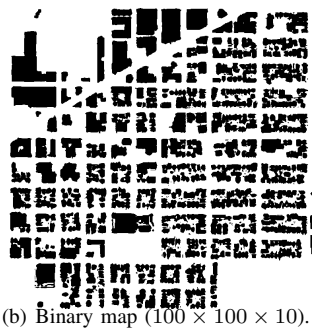


(f) Requirement achievement of UAV during the flight.

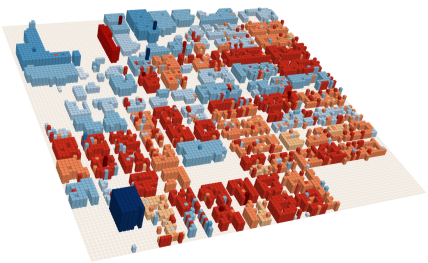
Fig. 4: Environment modeling with the scale of $500m \times 500m \times 100m$.



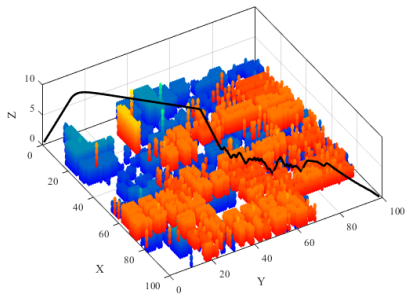
(a) ArcGIS map ($100 \times 100 \times 10$).



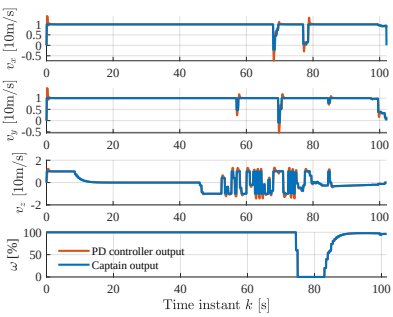
(b) Binary map ($100 \times 100 \times 10$).



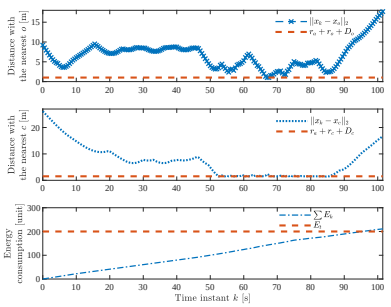
(c) Grid map ($100 \times 100 \times 10$).



(d) Flight Trajectory of UAV.



(e) State of UAV during the flight.



(f) Requirement achievement of UAV during the flight.

Fig. 5: Environment modeling with the scale of $1000m \times 1000m \times 100m$.

- Scanning Distance (R_l): A segment of surface over a distance of $L_t = 100$ km is expected to be examined by the UUV within $\Delta = 10$ hours, while the threshold of surveillance distance is $L = 90$ km.
- Energy Consumption (R_e): A total amount of energy $E_t = 5.4$ MJ is expected to be consumed, while the maximum amount of energy is $E = 6$ MJ.
- Accuracy (R_φ): The accuracy of sensor measurements is targeted at $A_t = 90\%$, while the accuracy threshold is set as $A = 80\%$.

A. Requirements Satisfaction Modeling

The UUV is equiped with 5 sensors for ocean surveillance. The scanning time 10 hours is 360 time instance, $x_i, i \in [1, 5]$ is the portion of time the sensor i should be used during system operation in each instance. Acc_i is the accuracy of sensor i; E_i is the energy consumed by sensor; V_i is the scanning speed of sensor. q_i is portion of accuracy of sensor and p_i is for scanning speed respectively in decimals. The energy consumed is related with working accuracy and speed of sensor. The corresponding measures are listed as follows: $\mathcal{X}_l = \sum_{k=0}^{T}\sum_{i=0}^{N} x_i q_i V_i \tau$, $\mathcal{X}_e = \sum_{k=0}^{T}\sum_{i=0}^{N} x_i E_i \cdot \frac{e^{p_i+q_i}-1}{e^2-1}\tau$, and $\mathcal{X}_\varphi = \sum_{k=0}^{T}\sum_{i=0}^{N} x_i p_i Acc_i$, where $T = 360$, i.e., adaptations is performed every 100 surface measurements of the UUV state,, and the time instance k incremented by $1 \sim 100$. The requirement satisfaction functions are listed as follows:

- Scanning distance: A segment of surface over a distance of $L_t = 100$ km is expected to be examined by the UUV within $\Delta = 10$ hours, while the distance threshold is $L = 90$ km.

$$\mathrm{DS}^2(\mathcal{X}_l) = \begin{cases} 1, & \mathcal{X}_l \geq L_t \\ \dfrac{\mathcal{X}_l - L}{L_t - L}, & L \leq \mathcal{X}_l < L_t \\ 0, & \mathcal{X}_l < L \end{cases}$$

- Energy-saving: A total amount of energy $E_t = 5.4$ MJ is expected to be consumed, while the maximum amount of energy is $E = 6$ MJ.

$$\mathrm{DS}^1(\mathcal{X}_e) = \begin{cases} 1, & \mathcal{X}_e \leq E_t \\ \dfrac{E - \mathcal{X}_e}{E - E_t}, & E_t < \mathcal{X}_e \leq E \\ 0, & \mathcal{X}_e > E \end{cases}$$

- Accuracy: The accuracy of sensor measurements is targeted at $A_t = 90\%$, while the accuracy threshold is set as $A = 80\%$.

$$\mathrm{DS}^2(\mathcal{X}_\varphi) = \begin{cases} 1, & \mathcal{X}_\varphi \geq A_t \\ \dfrac{\mathcal{X}_\varphi - A}{A_t - A}, & A \leq \mathcal{X}_{\varphi_k} < A_t \\ 0, & \mathcal{X}_\varphi < A \end{cases}$$

B. Experiment Results

To demonstrate the generality of `Captain`, we applied it to a UUV case described in [3]. The requirement satisfaction models in this scenario are $\mathrm{DS}^1(\mathcal{X}_l)$ (scanning distance), $\mathrm{DS}^2(\mathcal{X}_e)$ (energy consumption) and $\mathrm{DS}^2(\mathcal{X}_\varphi)$ (accuracy). There are trade-offs between these requirements, e.g., when sensors (e.g., sensor 1) with a higher quality of surveillance is chosen, more energy is consumed, resulting in less distance scanned.

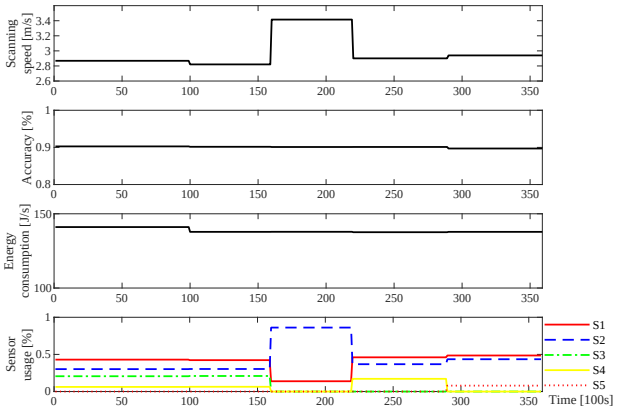


Fig. 6: Requirements-Driven Adaptation Planning Results in the UUV Case.

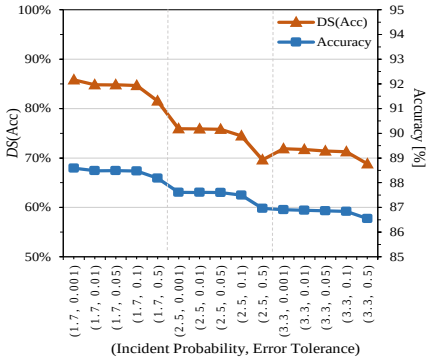
TABLE II: Results of Requirements Satisfaction in the UUV Case.

Req.	***Strategies***	# of incidents				
		3	6	9	12	15
$\mathcal{X}_\varphi$ [%]	AMOCS-MA	88.9	87.5	85.8	84.4	82.4
	GSlack	89.5	88.2	86.8	85.0	82.6
	`Captain`	***89.6***	***88.4***	***87.0***	***85.4***	***83.2***
$\mathcal{X}_l$ [km]	AMOCS-MA	99.9	99.2	98.0	96.6	94.5
	GSlack	103.9	103.4	103.0	100.5	98.8
	`Captain`	***104.2***	***104.0***	***103.5***	***101.0***	***98.9***
$\mathcal{X}_e$ [MJ]	AMOCS-MA	5.34	5.33	5.31	5.34	5.34
	GSlack	5.24	5.19	5.16	5.17	5.18
	`Captain`	***5.24***	***5.18***	***5.14***	***5.16***	***5.16***

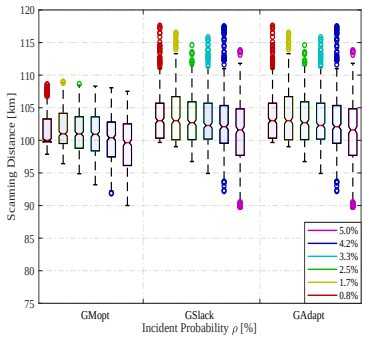
[a] The violation tolerance are $\epsilon_\varphi = 10^{-3}$, $\epsilon_l = 0$, and $\epsilon_e = 0$.

1) Effectiveness: The Fig. 6 shows the requirements-driven adaptation process of the UUV during operation with the case in [3]. At $k = 100$ we change the available energy change from 5.4 to 5.0 MJ, at $k = 160$ we change the distance to be scanned from 100 to 105 km. The plots show that these changes in requirements lead to corresponding changes in the arrangement of sensor usage, as the time portion for $S2$ increases. Figure 6 also shows how `Captain` reacts to changes in sensor parameters and sensor failures. At $k = 220$, the measurement accuracy of sensor $S3$ drastically decreases from 83% to 43%, at $k = 290$, $S4$ stops working, and leads to an optimal solution that $S1$ is more exploited. At last, the mission end with average measurement accuracy at 90.1%, scanning distance at 106.7 km, energy consumption at 4.98 MJ.

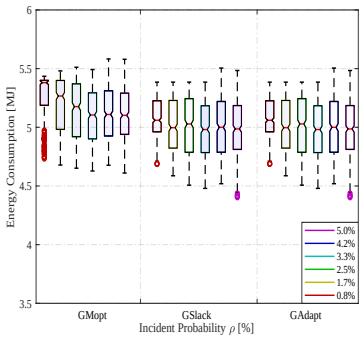
2) Robustness: In this scenario, the performance of the three methods are compared while adding random failures to parameters of sensors, i.e., sensor accuracy, scanning speed



(a) Scanning accuracy with various incidents and ϵ. (`Captain`)



(b) Variance of scanning distance.



(c) Variance of energy consumption.

Fig. 7: Scanning accuracy, distance and energy consumption with various probability of incidents and violation tolerance.

and energy consumption. For each method, we simulated UUV motion by adding different frequencies of random disturbances at different time instants. For each frequency of disturbances, we simulated 500 rounds and computed the average accuracy, scanning distance and energy consumption. The results are shown in Table II. We can see that under `Captain`, the UUV can scan a longer distance with higher accuracy and less energy consumption than AMOCS-MA and GSlack. Moreover, the motion generated by AMOCS-MA can hardly satisfy all the three requirements, while the motion from `Captain` and GSlack can satisfy the timeliness and energy requirements, only slightly violating the accuracy requirement. The reason behind this is that at the *Requirements Satisfaction Analysis* stage, the accuracy is selected as the requirement for adaptation while the other two remain as their original soft goals as constraints. Besides, `Captain` outperforms GSlack, especially in situations with very frequent disturbances. The variance of scanning distance and energy consumption of three strategies are compared in Figure 7(b) and Figure 7(c).

TABLE III: Statistics on Overhead Data.

Cases	***Approaches***	***Average*** [s]	***Standard Deviation*** [s]
UAV	AMOCS-MA	0.2115	0.0879
	GSlack	0.0544	0.0637
	`Captain`	***0.0811***	***0.0821***
UUV	AMOCS-MA	0.0423	0.0092
	GSlack	0.0180	0.0078
	`Captain`	***0.0344***	***0.0217***

3) Real-time Performance: Finally, we analyze the computation overhead of `Captain` in generating an optimal self-adaptive plan. Table III shows the empirical distribution of the computation time for 10000 executions of each method in either simulation, where the red stars represent the average computation time. From Tablee III, the difference in average computation time between AMOCS-MA and `Captain` is about 7 ms (0.0423 s versus 0.0344 s) in the UUV case, and 130 ms (0.2115 s versus 0.0811 s) in the UAV case.

This is because AMOCS-MA considers the optimization of all requirements while `Captain` only considers the optimization of unsatisfied requirements, as requirements predicted to be satisfied are constraints. So AMOCS-MA requires more computation especially when there are more requirements to be considered. `Captain` has larger variance since the number of requirements to be optimized may vary at runtime. In the worst case, when all the requirements need to be adjusted, the computation time of `Captain` will be larger than AMOCS-MA. This is very rare from our experiment results. Hence, `Captain` is flexible and capable of accommodating multiple requirements at runtime. Compared with GSlack, `Captain` requires more time to solve *Requirements Satisfaction Optimizing* problem. This gives us a trade-off between optimality and computation time: GSlack is faster but its solution is not optimal, while `Captain` needs more time to generate the solution to guarantee the overall optimality. Since the extra optimization in `Captain` can be solved based on results of the previous step within an acceptable time, we believe `Captain` is the best choice considering both performance and optimality.

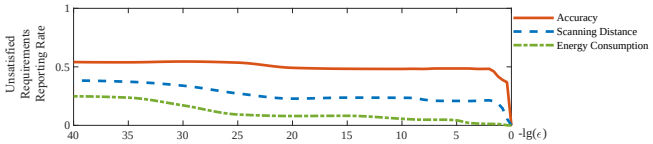


Fig. 8: Unsatisfied requirements reporting rate. (UUV case, # of incidents = 9)

4) Choice of violation tolerance: As discussed in the paper, the choice violation tolerance ϵ determine which requirement needs relaxation, while the larger the value is, the less number of requirements in the *Unsatisfied* requirement set, along with the underlying higher risk of no feasible solution. Thus the value of ϵ should be determined based on experimental data. The choice of different violation tolerance ϵ and the performance scanning accuracy achievement are illustrated in Figure 7(a). As the number of incidents increases, the impact of ϵ on requirement satisfaction is more obvious. Figure 8

illustrates adaptation rate of each soft requirements in UUV case, as the increase of violation tolerance. In the UUV case, we choose $\epsilon_{\varphi,e,l} = \{10^{-3}, 0, 0\}$, when the adaptation rate of each requirement tends to be gentle.

REFERENCES

[1] S. J. Burian, S. P. Velugubantla, K. Chittineni, S. R. K. Maddula, and M. J. Brown, "Morphological analyses using 3D building databases: Portland, Oregon," Utah. LA-UR, Los Alamos National Laboratory, Los Alamos, NM, Tech. Rep., 2002.

[2] A. Hidalgo-Panagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach," *Soft Computing*, vol. 21, no. 4, pp. 949–964, 2017.

[3] S. Shevtsov and D. Weyns, "Keep it simplex: Satisfying multiple goals with guarantees in control-based self-adaptive systems," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2016, pp. 229–241.

[4] S. Shevtsov, D. Weyns, and M. Maggio, "SimCA*: A control-theoretic approach to handle uncertainty in self-adaptive systems with guarantees," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 13, no. 4, p. 17, 2019.