# Graph Grammar-Based Automatic Design for Heterogeneous Fleets of Underwater Robots

Allan Zhao[1], Jie Xu[1], Juan Salazar[1], Wei Wang[1], Pingchuan Ma[1], Daniela Rus[1], and Wojciech Matusik[1]

***Abstract*— Autonomous underwater vehicles (AUVs) are specialized robots that are commonly used for seafloor surveying and ocean water sampling. Computational design approaches have emerged to reduce the effort required to design both individual AUVs as well as fleets. As the number and scale of underwater missions increases beyond the capabilities of a single vehicle, fleet level design will become more important. Depending on the mission, the optimal fleet may consist of multiple distinct types of AUVs designed to a variety of specifications. Moreover, the AUVs may differ in both continuous parameters (such as battery capacity) and discrete parameters (such as number and model of thrusters). In this work, we present a computational pipeline for designing these heterogeneous AUV fleets. Using a novel shape design space based on a graph grammar and deformation cages, we can express a variety of AUV architectures with different topologies, component selections, and dimensions. We search this space using a combination of discrete graph search and gradient-based continuous optimization, enabled by a differentiable AUV simulator. Finally, we formulate heterogeneous fleet design as a modified knapsack problem, and solve it using an efficient backtracking-based algorithm. We evaluate our pipeline on a simulated mission with nonuniform design requirements—surveying a section of seafloor with varying depth—and show that the best heterogeneous fleet outperforms the best fleet composed of a single vehicle type.**

## I. Introduction

Autonomous underwater vehicles (AUVs) have many industrial applications, such as search and rescue, spatiotemporal sampling, sea floor mapping, and offshore oil and gas installations [1], [2]. These applications have driven the design and development of AUVs in a variety of sizes, shapes, thruster configurations, and working depth limits. Optimizing for different mission objectives (such as cost, speed, and endurance) and operational conditions (such as deep sea, seafloor, and extreme weather) translates into significant variation in AUV designs [3]. The resulting design space for AUVs becomes overly expansive and complex for a person to explore without significant expertise. It includes discrete parameters (such as topology and component selection) as well as continuous parameters (such as component locations and dimensions). Engineers designing AUVs for any combination of the conditions mentioned above use methods that tend to rely on their collective knowledge and experience [4]. These manual design approaches offer limited trade-off possibilities.

Prior works on automating the design of AUVs [4]–[9] have limited expressiveness over structures. A key challenge is to find representations that can encode discrete and continuous parameters while supporting effective search strategies. Inspired by previous work on graph grammars [10]–[14] and shape deformation [15]–[17], we present a novel AUV shape representation combining a graph grammar and cage-based deformation. The grammar encodes AUV components and topologies while cage-based deformation allows for continuous shape variation—e.g., changing the length, diameter, or thickness of a hull segment.

We then build upon our single AUV design approach to rapidly design fleets containing multiple vehicle types for tasks subject to heterogeneous constraints—for example, large scale surveying at different depths. Much of the current multi-AUV literature assumes the AUV fleet is homogeneous, or composed of a single vehicle type [18], [19]. Heterogeneous AUV fleets could potentially make full use of individual abilities to satisfy the task requirements with greater efficiency. Prior works on marine exploration using heterogeneous AUV fleets [20]–[22] do not formulate the task as an optimization problem with constraints nor optimize the individual AUV designs along with the multi-AUV task.

In contrast, we begin by optimizing a number of individual AUV designs, each with a different mass limit and depth rating. Each design, consisting of discrete and continuous parameters, is evaluated on a task-specific objective function using a differentiable simulator. Our optimization pipeline uses a combination of discrete graph search and gradient-based continuous optimization to update each AUV's design parameters to minimize the objective. The result from these previous steps is a diverse library of AUV designs that feature unique pairings of mass and depth rating. Finally, this library is fed into our combinatorial fleet optimization algorithm, which outputs the optimal fleet according to the task specifications.

In the development of our computational fleet design pipeline, we present the following contributions. Our first contribution is a novel graph-grammar for AUVs that captures a diverse space of topologies. Our second contribution is that we formulate heterogeneous fleet design as a modified knapsack problem and propose an efficient, exact algorithm to solve it. We demonstrate our fleet design pipeline on a simulated mission with nonuniform design requirements: surveying a section of seafloor with varying depth.

## II. Design Space

We describe possible AUV topologies using a custom graph grammar and parameterize the hull shape with deformable cages. A single AUV design is represented as a

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (email to Allan Zhao: azhao@csail.mit.edu).

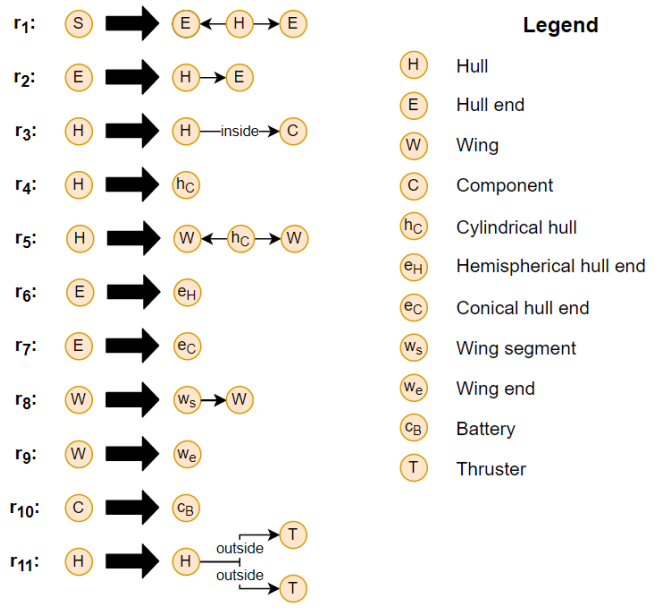


Fig. 1. The rules comprising our AUV grammar. Symbols are either in uppercase (representing abstract subgraphs) or in lowercase (representing concrete components).

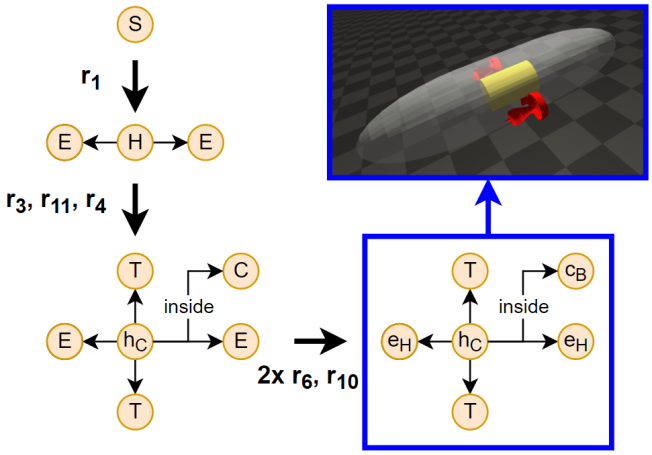


Fig. 2. Sequential growth of a propeller-driven AUV from applying 7 of our grammar rules.

graph where the nodes are components and the edges are the inter-component connections.

Our AUV grammar is illustrated in Fig. 1. Some rules encode components such as underwater thrusters, hull segments, and a parametric battery. Each component has properties based on its real-life counterpart, such as maximum thrust, mass, and energy density. The remaining rules control how nodes in the design graph can expand into subgraphs. Fig. 2 illustrates how a design is constructed by applying a sequence of these grammar rules.

The hull shape is composed of multiple segments, each of which is parameterized using a deformation cage. A cage initially forms an axis-aligned bounding box surrounding its segment, and the positions of its corners are functions of the continuous parameters. As the corners of a cage move, the points on the hull inside the cage are recalculated using trilinear interpolation. By combining these deformation cages with our graph grammar, we can represent AUVs with many different topologies and shapes.

## III. Simulation

### A. Dynamic Model

We first establish the inertial and body frames to describe the motion of an AUV. We adopt the North-East-Down (NED) frame as the inertial frame. The body frame is defined at the center of mass of an AUV with the three axes pointing forward ($x$), right ($y$), and down ($z$). We define an AUV's state as a 12-dimensional vector $\mathbf{s}$:

$$\mathbf{s} = (\mathbf{x}, \mathbf{\Phi}, \mathbf{v}_B, \mathbf{\Omega}) \tag{1}$$

where $\mathbf{x}$ stands for the position of the AUV in the inertial frame, $\mathbf{\Phi}$ stands for the attitude in the form of Euler angles, $\mathbf{v}_B$ denotes the linear velocity in the body-fixed frame, and $\mathbf{\Omega}$ is the angular velocity in the inertial frame. The following relation between the defined linear velocities holds:

$$\mathbf{v}_B = \mathbf{T}_{\mathrm{I}}^{\mathrm{B}} \dot{\mathbf{x}} \tag{2}$$

where $\mathbf{T}_{\mathrm{I}}^{\mathrm{B}}$ is the rotation matrix expressing the transformation from the inertial frame to the body-fixed frame. Moreover, the following relation between the defined angular velocities holds:

$$\mathbf{\Omega} = \mathbf{J}(\mathbf{\Phi})\dot{\mathbf{\Phi}} \tag{3}$$

where $\mathbf{J}(\mathbf{\Phi})$ is a Jacobian matrix. It is useful to collect the kinematic equations in six-dimensional matrix forms. We define the vector $\boldsymbol{\nu} \in \mathbb{R}^6$ as

$$\boldsymbol{\nu} = \begin{pmatrix} \mathbf{v}_B \\ \mathbf{\Omega} \end{pmatrix}. \tag{4}$$

According to Newton's law and Euler's equations, the overall equations of motion can therefore be written in matrix form as

$$\mathbf{M}_v \dot{\boldsymbol{\nu}} + \mathbf{C}_v(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}_v + \mathbf{g}_v(\mathbf{T}_{\mathrm{B}}^{\mathrm{I}}) = \boldsymbol{\tau}_v + \boldsymbol{\tau}_{\mathrm{E}}, \tag{5}$$

where $\mathbf{M}_v \in \mathbb{R}^{6\times6}$ represents the symmetric positive-definite added mass and inertia matrix; $\mathbf{C}_v(\mathbf{v}) \in \mathbb{R}^{6\times6}$ is the skew-symmetric vessel matrix of the Coriolis and centripetal terms; $\boldsymbol{\tau}_{\mathrm{E}} \in \mathbb{R}^{6\times1}$ represents environmental disturbances from wind, currents, and waves (which we assume to be zero for simplicity); $\mathbf{D}_v \in \mathbb{R}^6$ is the hydrodynamic damping; $\mathbf{g}_v(\mathbf{T}_{\mathrm{B}}^{\mathrm{I}})$ represents the forces and moments due to gravity and buoyancy in the body-fixed frame. $\boldsymbol{\tau}_{\mathbf{v}} = [\mathbf{f}^T \ \boldsymbol{\tau}^T]^T \in \mathbb{R}^6$ is the sum of forces and moments exerted by all thrusters. More details of the dynamic model can be found in [1].

For each thruster, we compute its orientation and relative position with respect to the center of mass, which gives us the direction of its thrust and induced torque. The magnitude of the thrust is an optimization variable.

For our hydrodynamic model, we follow the practice from [23], [24] by discretizing the hull geometry into a

triangle mesh and computing the lift and drag force on each surface triangle as follows:

$$\boldsymbol{f}_{\text{drag}} = \frac{1}{2}\rho A C_d(\phi)\|\boldsymbol{v}_{\text{rel}}\|^2 \boldsymbol{d} \tag{6}$$

$$\boldsymbol{f}_{\text{lift}} = -\frac{1}{2}\rho A C_l(\phi)\|\boldsymbol{v}_{\text{rel}}\|^2 \boldsymbol{n} \tag{7}$$

where $\rho$ is the fluid density, $A$ is the triangle's area, $\boldsymbol{v}_{\text{rel}} = \boldsymbol{v}_{\text{fluid}} - (\boldsymbol{v} + \boldsymbol{\Omega} \times \boldsymbol{x}_{\text{rel}})$ is the fluid velocity relative to the triangle, $\boldsymbol{d} = \frac{\boldsymbol{v}_{\text{rel}}}{\|\boldsymbol{v}_{\text{rel}}\|^2}$, and $\boldsymbol{n}$ is the triangle's normal. We ignore the indices of the surface triangles for clarity. $C_d(\phi)$ and $C_l(\phi)$ are the coefficients of drag and lift respectively, where $\phi = \cos^{-1}(\boldsymbol{n} \cdot \boldsymbol{v}_{\text{rel}}) - \frac{\pi}{2}$ is the angle of attack. We define $C_d(\phi)$ and $C_l(\phi)$ to resemble typical drag and lift curves, which are symmetric and monotonically increasing functions respectively:

$$\begin{aligned} C_d(\phi) = {} & 0.05 \cdot (1 - |\phi|)^3 + 0.05 \cdot 3 \cdot |\phi|(1 - |\phi|)^2 \\ & + 1.85 \cdot 3 \cdot |\phi|^2(1 - |\phi|) + 2.05 \cdot |\phi|^3 \end{aligned} \tag{8}$$

$$\begin{aligned} C_l(\phi) = {} & -0.8 \cdot (1 - \phi)^3 - 0.5 \cdot 3 \cdot \phi(1 - \phi)^2 \\ & + 0.1 \cdot 3 \cdot \phi^2(1 - \phi) + 2.5 \cdot \phi^3, \end{aligned} \tag{9}$$

The total hydrodynamic damping $\mathbf{D}_v = [\mathbf{f}_{\text{H}}^T \ \boldsymbol{\tau}_{\text{H}}^T]^T$ is computed by adding together the forces on each triangle and their resulting torques:

$$\boldsymbol{f}_{\text{H}} = \sum (\boldsymbol{f}_{\text{drag}} + \boldsymbol{f}_{\text{lift}}), \tag{10}$$

$$\boldsymbol{\tau}_{\text{H}} = \sum (\boldsymbol{x}_{\text{rel}} \times (\boldsymbol{f}_{\text{drag}} + \boldsymbol{f}_{\text{lift}})), \tag{11}$$

where $\boldsymbol{x}_{\text{rel}} = \boldsymbol{x}_{\text{tri}} - CG$ is the position of a triangle relative to the vehicle's center of mass.

Eqn. (5) provides us enough information to compute the time derivatives of $\mathbf{s}$. We can now represent the dynamics in a compact form M:

$$\dot{\mathbf{s}} = \text{M}(\mathbf{s}, \mathbf{a}, \text{G}, \theta). \tag{12}$$

Here, $\mathbf{a}$ is the action vector consisting of forces exerted by each thruster. The design graph and shape parameters $(\text{G}, \theta)$ determine the configuration of thrusters and the hull geometry. In short, given the current design $(\text{G}, \theta)$, the current state $\mathbf{s}$, and the current action $\mathbf{a}$, the dynamic model M computes $\dot{\mathbf{s}}$ that evolves the dynamic system.

### B. Trim State and Control

To control an AUV design, we compute a trim state and execute a simple open-loop control strategy. A trim state is a state with zero linear and angular accelerations:

$$(\dot{\bar{\mathbf{x}}}, \dot{\bar{\boldsymbol{\Phi}}}, 0, 0) = \text{M}(\bar{\mathbf{s}}, \bar{\mathbf{a}}, \text{G}, \theta), \tag{13}$$

By constantly applying the control action $\bar{\mathbf{a}}$ to the vehicle in the trim state, the vehicle maintains its linear and angular velocity. This results in a straight line motion that allows us to extrapolate the vehicle's power consumption over a long time horizon and calculate the resulting endurance.

TABLE I

SIMULATION PARAMETERS FOR THE SURVEYING MISSION

| Parameter | Value | Description |
|---|---|---|
| $\rho$ | 1027 kg/m$^3$ | Density of seawater |
| $a_{\text{survey}}$ | 4000 km$^2$ | Total survey area |
| $m_{\text{total}}$ | 4000 kg | Total fleet mass |
| $d_{\text{transit}}$ | 100 km | Transit distance |
| $t_{\text{charge}}$ | 24 hr | Charging time |
| $\eta$ | 0.5 | Propulsion efficiency |
| $p_{\text{core}}$ | 8 W | Compute and sensor power |
| $p_{\text{sonar}}$ | 70 W | Sonar power consumption when active |
| $h_{\text{sonar}}$ | 400 m | Sonar scan width |
| $\alpha$ | 0.5 | Sonar swath overlap fraction |
| $\gamma$ | 0.8 | Usable fraction of battery capacity |

## IV. OPTIMIZATION

Our pipeline employs multiple optimization techniques to generate an optimal fleet for a given task. In the precomputation stage, we optimize a library of vehicles for multiple mass limits and depth ratings using a combination of discrete search and gradient-based continuous optimization. In the scheduling stage, we then solve a modified knapsack problem using the library of precomputed designs to minimize the mission time under a total mass constraint.

### A. Task Specification

We demonstrate our complete pipeline on a simulated surveying mission, where the vehicles in a fleet work together to scan a region of seafloor with area $a_{\text{survey}}$ using sonar. The area being surveyed ranges in depth from 200 to 1000 meters, so at least one of the vehicles must be capable of the full depth. We assume that the survey area is sloped so that there is an equal, infinitesimal fraction of its area at each possible depth.

The goal is to complete the survey using as little time as possible, subject to a total fleet mass constraint $m_{\text{total}}$ (which relates to the cost to construct and transport the vehicles). Vehicles start at a shore-based facility, and navigate their way to the survey area located a distance $d_{\text{transit}}$ away. The survey does not need to be completed on a single battery charge; vehicles may return to the facility to recharge before resuming the survey. Recharging a vehicle takes a predefined amount of time $t_{\text{charge}}$. The vehicles must return to the shore facility by the end of the mission.

We assume each vehicle travels at one of two possible speeds: $v_{\text{transit}}$ when transiting to or from the survey area, and $v_{\text{survey}}$ when actively surveying. These speeds may vary between different vehicles in a fleet.

The specific values of the parameters used in our experiments are listed in Table I.

### B. Precomputation of Individual Vehicles

Before running the fleet optimization, we first generate a library of vehicles for multiple combinations of mass limits and depth ratings. The mass limits are chosen so that they roughly form a geometric sequence, while the depth ratings are chosen so that an equal amount of the survey area lies within each depth bin. In our experiments, we select four

depth cutoffs $d_1 = 400,\ d_2 = 600,\ d_3 = 800,\ d_4 = 1000$ m resulting in equal areas in each depth bin $\hat{A}_k = 1000$ km$^2$. Within each depth bin, we optimize vehicles for mass limits of 200, 300, 500, 1000, 1500, and 2000 kilograms.

For each mass and depth pair, we run our single vehicle design pipeline consisting of a discrete search algorithm paired with gradient-based continuous optimization. The discrete search operates as an outer loop that samples different topologies, and is only concerned with the types of components and how they are connected. Continuous optimization is then responsible for finding locally optimal continuous parameters for each topology. The continuous parameters of a design include not only the positions, orientations, and dimensions of each component $\boldsymbol{\theta}$, but also the steady state speeds in the transit and survey phases $v_{\text{transit}}, v_{\text{survey}}$ and the control inputs $\mathbf{a}$ to the thrusters.

The objective function for continuous optimization is composed of two terms:

- a survey rate term $L_{\text{rate}}$ that maximizes the area surveyed by the vehicle per unit time (including transit and charging time between sorties),
- and a penalty term $L_{\text{penalty}}$ that helps the optimization converge towards a positive survey time.

Our formulation also includes several constraints:

- linear and angular acceleration are zero in both the transit and survey phases, meaning that the thrust force exactly counteracts the forces from drag and buoyancy,
- the survey time $t_{\text{survey}}$ is positive, ensuring that the vehicle has sufficient battery capacity to approach and return from the survey area,
- and the total mass $m$ of the vehicle is no greater than the mass limit $m_{\max}$.

The single-vehicle optimization problem can be formally described as

$$\begin{aligned} \min_{\boldsymbol{\theta}, v_{\text{transit}}, v_{\text{survey}}, a} \quad & \underbrace{-\frac{a_{\text{sortie}}}{t_{\text{sortie}}}}_{L_{\text{rate}}} + \lambda \underbrace{e^{-t_{\text{survey}}}}_{L_{\text{penalty}}} \\ \text{s.t.} \quad & \dot{v}_{\text{transit}}, \dot{\omega}_{\text{transit}} = 0 \\ & \dot{v}_{\text{survey}}, \dot{\omega}_{\text{survey}} = 0 \\ & t_{\text{survey}} \geq 0 \\ & m \leq m_{\max} \end{aligned} \tag{14}$$

where $\lambda = 1 \times 10^3$ and $t_{\text{sortie}} = t_{\text{transit}} + t_{\text{survey}} + t_{\text{charge}}$ is the total time per sortie. The objective and constraints, which involve computing the dynamics of the underwater vehicle, are all differentiable and can be optimized by a gradient-based solver.

### C. Fleet Optimization

We now describe our algorithm to find the best fleet (Alg. 1) from the 24 precomputed vehicle candidates in Section IV-B, such that the total mass of the fleet is within a limit and the entire region is surveyed in minimal time. Let $\{C_i\} = \{(m^i, d^i, v^i_{\text{survey}}, t^i_{\text{survey}}, t^i_{\text{transit}})\}$ be the set of vehicles in the fleet, and let $N_i$ be the number of survey sorties scheduled for the $i$-th vehicle. The fleet optimization problem can be formally described as:

$$\min_{\{(C_i, N_i, \tilde{T}_i)\}} \max_i \ \tilde{T}_i \tag{15}$$

$$\text{s.t.} \ \ C_i \in \text{candidate designs} \tag{16}$$

$$\sum_i m^i \leq m_{\max} \tag{17}$$

$$\tilde{T}_i \geq t^i_{\text{transit}} \times N_i + t_{\text{charge}} \times (N_i - 1) \tag{18}$$

$$\tilde{T}_i \leq (t^i_{\text{transit}} + t^i_{\text{survey}}) \times N_i + t_{\text{charge}} \times (N_i - 1) \tag{19}$$

$$\text{Entire survey area is covered}$$

$$N_i \in \mathbb{N},$$

where $\tilde{T}_i$ is the total working time for the $i$-th vehicle, and (18-19) are the validity constraints for this surveying time. Specifically, a vehicle has to spend time $t^i_{\text{transit}}$ for each sortie, time $t_{\text{charge}}$ between two consecutive sorties to recharge, and no more than time $t^i_{\text{survey}}$ to scan the area during each sortie.

We perform binary search on the optimal time $T^*$ to complete the survey task, converting the min-max optimization problem in (15-19) into a feasibility problem with an additional constraint $\tilde{T}_i \leq T^*$. In this way, we find the smallest time limit $T^*$ in which a fleet can complete the mission.

Note that a vehicle with depth rating $d^i$ can survey areas with depth less than or equal to $d^i$. Given a maximal allowed time $T^*$, let $a^i$ be the maximal area that can be surveyed by a vehicle $i$ within its depth range. Consequentially, a fleet that can finish the task on time must satisfy:

$$\forall_d \sum_{d_k \geq d} \hat{A}_k \leq \sum_{d^i \geq d} a^i, \tag{20}$$

where $\hat{A}_k$ is the total area required to be surveyed at depth $d_k$.

To compute the maximal survey area $a^i$ for each vehicle, we can formulate the following mixed integer linear programming problem:

$$\max_{N_i, \tilde{T}^i} a^i \tag{21}$$

$$\begin{aligned} \text{s.t.} \quad & \\ & a^i = \Big(\tilde{T}_i - t^i_{\text{transit}} \times N_i - t_{\text{charge}} \times (N_i - 1)\Big) v^i_{\text{survey}} \\ & \tilde{T}_i \geq t^i_{\text{transit}} \times N_i + t_{\text{charge}} \times (N_i - 1) \\ & \tilde{T}_i \leq (t^i_{\text{transit}} + t^i_{\text{survey}}) \times N_i + t_{\text{charge}} \times (N_i - 1) \\ & \tilde{T}_i \leq T^*,\ N_i \in \mathbb{N} \end{aligned}$$

Note that the analytical solution of (21) can be computed using standard algebraic techniques. Specifically, the maximum is achieved when $N_i = \lfloor \frac{T^* + t_{charge}}{t^i_{transit} + t^i_{survey} + t_{charge}} \rfloor$ or $\lceil \frac{T^* + t_{charge}}{t^i_{transit} + t^i_{survey} + t_{charge}} \rceil$, and $\tilde{T}_i = \min\{T^*, (t^i_{\text{transit}} + t^i_{\text{survey}}) \times N_i + t_{\text{charge}} \times (N_i - 1)\}$.

The feasibility problem can now be regarded as a modified knapsack problem, where each type of candidate vehicle can contribute $a^i$ survey area in its depth range with a weight cost of $m^i$, and we need to select the correct number of vehicles of each type so that their total surveyed area satisfies (20). We

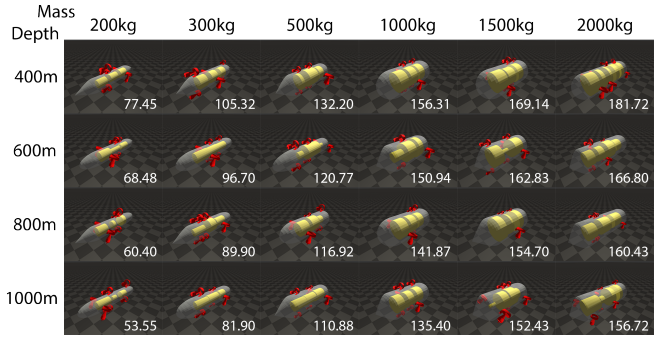


Fig. 3. The best designs found by our single-vehicle optimization pipeline for each combination of mass limit and depth rating. The survey rate (in $m^2/s$) is indicated next to each design.

convert this modified knapsack problem into a graph search problem and solve it by breadth first search with several pruning techniques.

Specifically, we denote a search node $(i,A,M)$ as achievable if and only if there exists a fleet composed of the first $i$ types of vehicles that can survey an area $A$ within mass budget $M$. Given an achievable node $(i,A,M)$, we can consider how many vehicles of type $i+1$ to select. If we select $k$ vehicles of type $i+1$, we can further label the node $(i+1,A+a^{i+1}k,M+m^{i+1}k)$ as achievable. Note that the area constraint (20) would be hard to check if we consider the vehicles in arbitrary order, since the whole fleet selection would need to be encoded in our node state instead of just three values $(i,A,M)$. By sorting the candidate vehicles in descending order of their depths, we can easily prune nodes that violate (20) while keeping the node state compact.

Our search algorithm starts with only node $(0,0,0)$ being achievable. The feasible solution of the problem exists if and only if there exists an achievable node $(n,A,M)$ where $A \geq \sum_k \hat{A}_k$ and $M \leq m_{\max}$. Since the problem is combinatorial, the search space is extremely large. In addition to the *area constraint pruning* mentioned earlier, we also apply *Pareto pruning* to further speed up the algorithm. Given a set of all achievable nodes after considering the first $i$ types of vehicles, a node $\{(i,A_j,M_j)\}$ can be removed from the set if it is dominated by another achievable node $\{(i,A_k,M_k)\}$; *i.e.*, $A_j \leq A_k$ and $M_j \geq M_k$ and the inequality holds for at least one equation. In other words, we only keep the achievable nodes on the Pareto front of the two metrics $A$ and $M$ after considering each additional vehicle type.

Note that our fleet optimization algorithm provides an exact solution to the problem described in (15-19). Only nodes which cannot be a part of the solution are pruned.

## V. Results

In this section, we provide an end-to-end demonstration of our heterogeneous fleet optimization pipeline, and show that it produces fleets that complete the simulated surveying mission more quickly than fleets composed of a single vehicle type.

**Algorithm 1:** Fleet Optimization

**Data:** $n$ candidate vehicles, weight limit $m_{\max}$, area to be surveyed at each depth $\hat{A}_k$.
**Result:** The optimal fleet and the minimum time to complete survey task $T$.

```
sort candidate vehicles in descending order of depth.
T_l ← 0, T_r ← a large enough number.
while T_l < T_r − ε do                    /* binary search */
    T* ← (T_l + T_r)/2
    compute a^i for each vehicle type by solving (21).
    // breadth first search
    success ← False
    S = {(0,0,0)}                         /* achievable set */
    for i = 1 → n do
        S' = {}
        for (i−1, A, M) ∈ S do
            for k = 0 → ⌊(m_max − M)/m^i⌋ do
                S' = S' ∪ {(i, A + a^i k, M + m^i k)}
            end for
        end for
        remove nodes from S' by checking area
         constraint in Eqn. (20).
        remove nodes from S' by Pareto pruning.
        if ∃(i, A, M) ∈ S', A ≥ Σ_k Â_k then
            success ← True
            save the fleet configuration.
            break
        end if
    end for
    // halve the searching space
    if success then
        T_r ← T*
    else
        T_l ← T*
    end if
    T ← T_l
    fleet ← last saved fleet configuration.
end while
```

### A. Single-Vehicle Optimization

Figure 3 shows the results of the precomputation stage, where we generate a library of designs for multiple mass and depth rating combinations. One of the most noticeable trends is the increasing size and number of battery modules (represented by the yellow cylinders) as the allowed mass increases. The corresponding increase in battery capacity allows higher mass vehicles to spend more time surveying as opposed to recharging, resulting in greater survey rate.

As the required depth rating increases, the thickness and mass of the hull also increases. In order to maintain close to neutral buoyancy, less of the internal volume can be used for batteries. The survey rate reduces accordingly. This trend is noticeable at the higher end of the depth range, as more space is left between the batteries and the hull.

Thruster placement (shown in red) does not appear to follow a clear pattern. One feature stands out however:

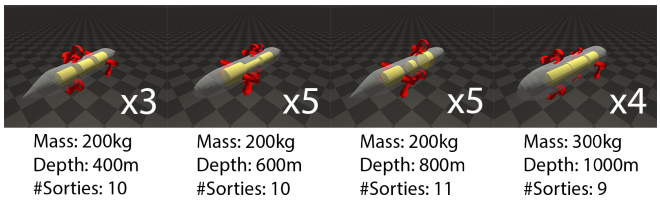


Fig. 4. The best fleet found by our fleet optimization algorithm using the precomputed design library, along with the number of sorties performed by each vehicle of a given type. The fleet consists of 4 different designs, with the number of instances of each design indicated.

the upward orientation of some thrusters. Interestingly, the optimization tends to produce negatively buoyant designs that require thrust to maintain a constant depth. Although this may seem counterintuitive (since it results in additional power consumption), the endurance of the vehicles is increased. Reducing the displacement (and therefore mass) of the hull allows allocating more of the mass budget to batteries, resulting in greater endurance compared to a neutrally buoyant design. If recoverability of the vehicles in case of failure is important, an additional constraint enforcing neutral buoyancy (omitted here) could be imposed.

### B. Fleet Optimization

Using the precomputed library of vehicles, we run our fleet optimization algorithm (Alg. 1) to produce an optimal fleet for the mission (Fig. 4). We notice that the fleet includes a large number of lightweight vehicles as opposed to fewer, more capable vehicles. This makes sense, as overall survey rate of the fleet scales roughly linearly with the number of vehicles. The vehicle type chosen to cover the deepest portion of the survey area is not the lightest possible, however. This may be due to the fact that the hull of the lightest vehicle capable of 1000 m depth weighs 145 kg alone, leaving very little of the mass budget for batteries.

Taking advantage of varying design requirements within the same mission (in our case, the depth rating) enables a better solution composed of multiple vehicle types. The heterogeneous fleet designed by our pipeline completes the mission in only 39 days, as opposed to the best achievable using a single vehicle type under the same constraints: 44 days using 10 identical vehicles of 400 kg each. The results in Table II suggest that parallelizing across multiple identical vehicles can decrease mission time, but only until a certain point. As the number of vehicles increases (and the mass budget per vehicle decreases), the hull represents a nontrivial fraction of the vehicle mass.

### C. Implementation Details

The majority of our pipeline is implemented in Python using the PyTorch library [25] for autodifferentiation. Performance-critical components of the continuous optimization objective, namely the underwater vehicle dynamics, are implemented in C++ with manually derived gradients.

We choose to use random search as the discrete search algorithm for simplicity, and SLSQP from the NLopt package [26] for gradient-based continuous optimization. Each iteration of the discrete search, which involves up to 1000 SLSQP iterations, takes an average of 8 seconds per topology on a Google Cloud N2 instance with 24 cores. The continuous optimization is single-threaded, but multiple design searches for different mass and depth combinations can be run in parallel. The entire precomputation stage can be completed in less than 5 hours on the 24-core machine.

With the 24 precomputed designs (Fig. 3) as input, our fleet optimization algorithm then produces the same fleet as brute force enumeration in less than a second.

TABLE II

MISSION TIMES USING A SINGLE VEHICLE TYPE

| Number of vehicles | Mass per vehicle (kg) | Mission time (days) |
|---|---|---|
| 1 | 4000 | 187.5 |
| 2 | 2000 | 97.6 |
| 3 | 1333 | 72.3 |
| 4 | 1000 | 59.8 |
| 5 | 800 | 49.5 |
| 6 | 666 | 49.0 |
| 7 | 571 | 46.9 |
| 8 | 500 | 45.3 |
| 10 | 400 | 44.0 |
| 12 | 333 | 49.2 |
| 14 | 285 | 49.1 |
| 16 | 250 | 54.4 |
| 18 | 222 | 57.5 |
| 20 | 200 | 62.5 |
| 22 | 181 | 68.3 |
| 24 | 166 | 76.1 |

## VI. CONCLUSION AND FUTURE WORK

Designing optimal heterogeneous fleets of autonomous underwater vehicles (AUVs) for a given task requires both vehicle-level and fleet-level optimization. Individual vehicles are described by both discrete parameters, such as topology and component selection, as well as continuous parameters such as the placement and sizing of the components.

In this work, we propose a novel shape design space for AUVs that encompasses both discrete parameters (represented in a graph) and continuous parameters associated with the nodes of the graph. We further contribute a complete pipeline for heterogeneous AUV fleet design, and evaluate it on a simulated surveying task covering a range of depths. The resulting solution, consisting of multiple distinct AUV designs, can complete the surveying task in less time than the best fleet composed of a single design.

One possible direction for future work would be a more principled method for selecting the mass and depth bins. We also rely on a simple analytical drag model to estimate the power needed for propulsion. Modeling drag using computational fluid dynamics (CFD) would enable more detailed shape design. Finally, designs in our framework are specified at a high level of abstraction. This allows for mission-level planning, but still leaves many details of fabrication to the human designer.

## ACKNOWLEDGMENTS

This work is supported by Defense Advanced Research Projects Agency (DARPA) grant No. FA8750-20-C-0075.

## REFERENCES


[1] G. Antonelli, T. I. Fossen, and D. R. Yoerger, *Modeling and Control of Underwater Robots*. Cham: Springer International Publishing, 2016, pp. 1285–1306. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_51

[2] K. L. Vasudev, "Review of autonomous underwater vehicles," in *Autonomous Vehicles*, G. Dekoulis, Ed. Rijeka: IntechOpen, 2020, ch. 2. [Online]. Available: https://doi.org/10.5772/intechopen.81217

[3] K. Alam, T. Ray, and S. G. Anavatti, "A brief taxonomy of autonomous underwater vehicle design literature," *Ocean Engineering*, vol. 88, no. Complete, pp. 627–630, 2014.

[4] M. Martz and W. L. Neu, "Multi-objective optimization of an autonomous underwater vehicle," in *OCEANS 2008*, 2008, pp. 1–9.

[5] C. Brown and R. P. Clark, "Using a novel vehicle conceptual design utility to evaluate a long-range, large payload uuv," in *OCEANS 2010 MTS/IEEE SEATTLE*, 2010, pp. 1–10.

[6] K. Alam, T. Ray, and S. G. Anavatti, "Design optimization of an unmanned underwater vehicle using low- and high-fidelity models," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 11, pp. 2794–2808, 2017.

[7] ——, "An evolutionary approach for the design of autonomous underwater vehicles," in *AI 2012: Advances in Artificial Intelligence*, M. Thielscher and D. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 279–290.

[8] ——, "Design of a toy submarine using underwater vehicle design optimization framework," in *2011 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS) Proceedings*, 2011, pp. 23–29.

[9] A. Alvarez, V. Bertram, and L. Gualdesi, "Hull hydrodynamic optimization of autonomous underwater vehicles operating at snorkeling depth," *Ocean Engineering*, vol. 36, no. 1, pp. 105–112, 2009, autonomous Underwater Vehicles. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801808001765

[10] K. Sims, "Evolving virtual creatures," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 15–22.

[11] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik, "Robogrammar: graph grammar for terrain-optimized robot design," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–16, 2020.

[12] J. Xu, A. Speilberg, A. Zhao, D. Rus, and W. Matusik, "Multi-objective graph heuristic search for terrestrial robot design." IEEE, 2021.

[13] F. R. Stöckli and K. Shea, "A simulation-driven graph grammar method for the automated synthesis of passive dynamic brachiating robots," in *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2015.

[14] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros, "Learning to control self-assembling morphologies: a study of generalization via modularity," *arXiv preprint arXiv:1902.05546*, 2019.

[15] J. Xu, T. Chen, L. Zlokapa, M. Foshey, W. Matusik, S. Sueda, and P. Agrawal, "An End-to-End Differentiable Framework for Contact-Aware Robot Design," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[16] A. Jacobson, I. Baran, J. Popović, and O. Sorkine, "Bounded biharmonic weights for real-time deformation," *ACM Trans. Graph.*, vol. 30, no. 4, Jul. 2011. [Online]. Available: https://doi.org/10.1145/2010324.1964973

[17] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," *ACM Trans. Graph.*, vol. 24, no. 3, p. 561–566, Jul. 2005. [Online]. Available: https://doi.org/10.1145/1073204.1073229

[18] Z. Zhou, J. Liu, and J. Yu, "A survey of underwater multi-robot systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, p. 18, 2021.

[19] J. Yu, C. Wang, and G. Xie, "Coordination of multiple robotic fish with applications to underwater robot competition," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1280–1288, 2016.

[20] R. Thenius, D. Moser, J. C. Varughese, S. Kernbach, I. Kuksin, O. Kernbach, E. Kuksina, N. Mišković, S. Bogdan, T. Petrović, A. Babić, F. Boyer, V. Lebastard, S. Bazeille, G. W. Ferrari, E. Donati, R. Pelliccia, D. Romano, G. J. Van Vuuren, C. Stefanini, M. Morgantin, A. Campo, and T. Schmickl, "subcultron - cultural development as a tool in underwater robotics," in *Artificial Life and Intelligent Agents*, P. R. Lewis, C. J. Headleand, S. Battle, and P. D. Ritsos, Eds. Cham: Springer International Publishing, 2018, pp. 27–41.

[21] T. Schmickl, R. Thenius, C. Moslinger, J. Timmis, A. Tyrrell, M. Read, J. Hilder, J. Halloy, A. Campo, C. Stefanini, L. Manfredi, S. Orofino, S. Kernbach, T. Dipper, and D. Sutantyo, "Cocoro – the self-aware underwater swarm," in *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, 2011, pp. 120–126.

[22] D. Sutantyo, P. Levi, C. Möslinger, and M. Read, "Collective-adaptive lévy flight for underwater multi-robot exploration," in *2013 IEEE International Conference on Mechatronics and Automation*, 2013, pp. 456–462.

[23] S. Min, J. Won, S. Lee, J. Park, and J. Lee, "Softcon: Simulation and control of soft-bodied animals with biomimetic actuators," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019.

[24] P. Ma, T. Du, J. Z. Zhang, K. Wu, A. Spielberg, R. K. Katzschmann, and W. Matusik, "Diffaqua: A differentiable computational design pipeline for soft underwater swimmers with shape interpolation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, p. 132, 2021.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[26] S. G. Johnson, "The nlopt nonlinear-optimization package," 2014.