Thruster Placement Optimization

Ian Wilhite

November 16, 2025

1 Introduction

We consider N body-fixed thrusters mounted on or near a reference sphere of radius ρ. Each thruster produces a force along its own local axis, and we seek to (a) map these forces to a global wrench about the origin and (b) optimize placement/orientation for strong, well-conditioned 6-DoF authority. This document standardizes frames, derives the required transformations, builds the allocation matrix, and proposes a differentiable evaluation function for gradient-based design.

While a minimum of six thrusters are required for full 6-DoF control, this system utilizes eight. The two additional thrusters provide redundancy, enchancing the vehicle's fault tolerance and allowing it to maintain control with failure. This redundancy was evaluated using a 'leave-one-out' analysis, which accounts for a worst-case single-point failure, aiding the evaluation of not only complete control authority, but designing redundancy into the system.

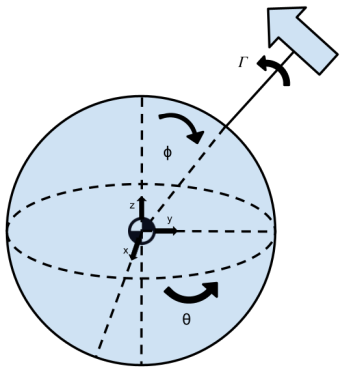

Figure 1: non-Gimbaled Thruster ROV Convention

2 Optimization Process

The problem of thruster placement optimization on the ROV, as defined in this document, is to find a thruster configuration that provides well-conditioned 6-DoF control with wrench bias under single-point failures.

Here is a narrative of the optimization process:

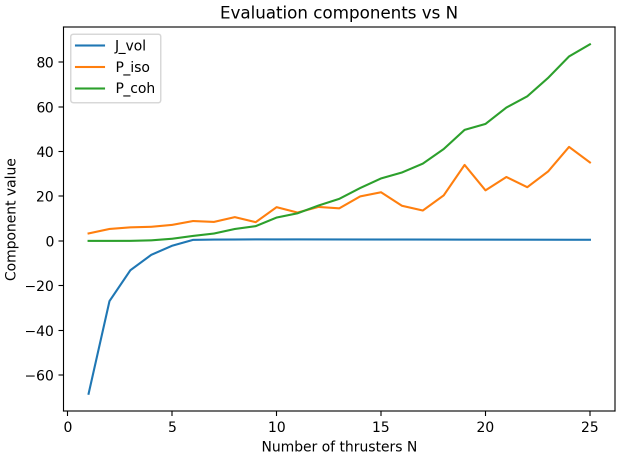

Figure 2: Evaluation Components

2.1 Stage 1: Initial Exploration (`optimizer.py`)

The process follows standard stochastic gradient descent methods. An evalaution function was developed which evaluated

2.2 Stage 2: Mathematical Formulation and Core Metrics (`optimizer2.py`)

This script marks the true beginning of the rigorous optimization process. It established the core mathematical framework by defining:

- **The Allocation Matrix (A):** The central element of the problem. This 6xN matrix maps the forces of the N individual thrusters to the resulting 6-DoF wrench (3 forces + 3 torques) on the ROV's body.
- **Wrench Metrics:** To evaluate the quality of a given thruster layout, a set of metrics based on the Singular Value Decomposition (SVD) of the allocation matrix was implemented:
  - `sigma_min`: The smallest singular value, representing the maneuverability in the weakest direction. Maximizing this is key to ensuring the ROV has no "bad" directions.
  - `manipulability`: The product of the singular values. A measure of the total volume of the achievable wrench space.
  - `cond` (Condition Number): The ratio of the largest to the smallest singular value. A value close to 1 indicates that the ROV is equally agile in all directions (isotropy).
- **Objective Function (J):** A scalar objective function `J` was created to be minimized. It was formulated to simultaneously maximize `sigma_min` and `manipulability` while minimizing the `condition number`.
- **Random Search:** A `quick_random_search` function was implemented to sample thousands of random thruster configurations to find a promising starting point for a more refined search.

2.3 Stage 3: Gradient Descent and Visualization (`optimizer3.py` and `optimizer4.py`)

Building on the mathematical framework of the previous stage, these scripts (which are identical) introduced an iterative optimization method:

- **Gradient Descent:** A gradient descent loop was implemented to refine the thruster parameters. The gradient of the complex objective function was calculated numerically using a finite-difference method. In each iteration, the parameters (thruster angles) are adjusted to "descend" towards a better objective score.
- **3D Visualization:** The script added `matplotlib` visualizations to plot the thruster positions and orientations on a 3D sphere, providing immediate visual feedback on the optimizer's progress.
- **Effectiveness Evaluation:** A critical function, `evaluate_effectiveness`, was added. It uses linear programming to calculate the maximum achievable force/torque in each of the 12 cardinal directions (e.g., +Fx, -Fx, +Fy, etc.), taking into account realistic asymmetric thrust limits. This provides a concrete measure of the final design's real-world performance.

2.4 Stage 4: Structured, Goal-Oriented Optimization (`optimizer5.py`)

This final script represents a significant leap in sophistication by introducing structure and user-defined goals:

- **Reduced Parameter Space:** Instead of optimizing 24 independent parameters for 8 thrusters, the problem was constrained. The thrusters were grouped into 4 pairs, with the pair centers fixed in a symmetric tetrahedral arrangement. This reduced the optimization space to just 12 parameters (`spacing`, `tilt`, and `rotation` for each pair), making the search more efficient.
- **Goal-Oriented Objective Function:** The objective function was enhanced to be goal-driven. A `w_target` (target wrench) vector was introduced, allowing the user to specify desired performance characteristics (e.g., "more forward thrust than vertical thrust"). The objective function now includes a penalty for any deviation from this target, guiding the optimizer to a solution that is not only well-conditioned but also tailored to the specific mission requirements of the ROV.
- **Advanced Diagnostics:** The script includes tools to plot the convergence of the objective function and to quantitatively compare the final achieved wrench capabilities against the desired target wrench, providing a clear report on the success of the optimization.

In essence, the process evolved from a simple idea to a powerful, goal-driven optimization framework that leverages advanced mathematical concepts and numerical methods to design a bespoke, high-performance thruster configuration for the ROV.

3 Conclusion

This narrative consolidates geometry, transforms, wrench mapping, the allocation matrix, and a differentiable objective suitable for gradient-based thruster placement. It is intended to be self-contained and implementation-ready.