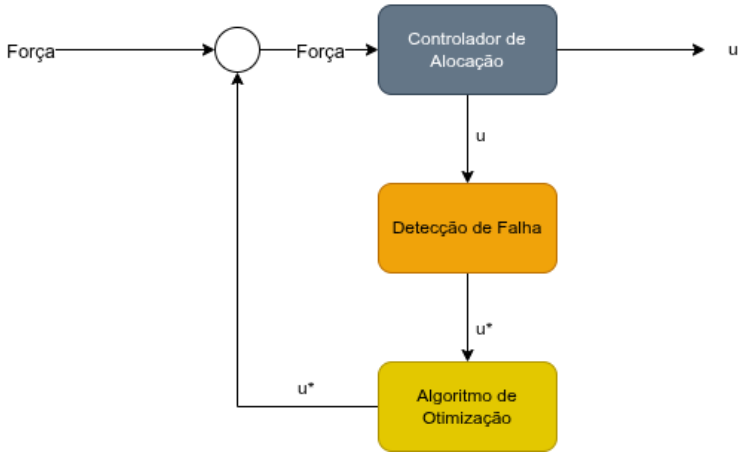
A metodologia proposta neste trabalho consiste no desenvolvimento e validação de um algoritmo de otimização voltado à redistribuição de forças em veículos subaquáticos (*BlueROV2*), com o objetivo de manter a navegabilidade do sistema em caso de falha total de um ou mais *thrusters*.

Os experimentos foram conduzidos em ambiente de simulação, utilizando o sistema operacional Ubuntu 22.04, o simulador Gazebo Ignition e o framework ROS 2 Control. O modelo *BlueROV2 Standard* foi adotado como base para representar o comportamento dinâmico do veículo e verificar o desempenho do método proposto. O controle do ROV é feito por um controlador de alocação de esforços, responsável por distribuir as forças entre os *thrusters*. O algoritmo de otimização será integrado em paralelo com o controle de alocação, de modo a realizar a redistribuição adaptativa das forças quando houver falha em um ou mais propulsores. Vale destacar que o reconhecimento de falha não é abordado neste trabalho, por ser um desafio à parte da otimização das forças.

O diagrama do fluxo de controle proposto é apresentado na Figura 1, destacando a interação entre o controlador de alocação e o algoritmo de otimização, bem como as entradas e saídas do sistema. O desempenho do algoritmo será avaliado com base na comparação entre o erro medio obtido com e sem o algoritmo e o desvio padrão do erro, considerando dois critérios principais: a velocidade de movimentação do ROV em determinada direção e o erro de deslocamento, em caso de falha de um ou mais *thrusters*, para determinada direção.

Figura 1: Fluxo de controle.



Fonte: Autores.

A partir dessas análises é esperado verificar a robustez e a eficiência do método de otimização proposto em comparação ao controle convencional sem a tolerância a falhas.

0.1 Metodologia experimental

A validação do algoritmo de otimização será realizada por meio do seguinte procedimento experimental:

Foram estabelecidos três cenários de teste, cinco casos de falhas e quatro trajetórias, conforme detalhado na Tabela 1:

Tabela 1: Configuração experimental: cenários, casos de falhas e trajetórias

Cenários		**Casos de Falhas**		**Trajetórias**
ID	**Descrição**	**ID**	**Tipo de Falha**	**Movimento**
1	Sem falhas e sem otimização	1	Dois *thrusters* horizontais coincidentes	1. *Surge* (X)
2	Com falhas e sem otimização	2	Dois *thrusters* horizontais paralelos	2. *Sway* (Y)
3	Com falhas e com otimização	3	Dois *thrusters* horizontais diagonais	3. *Heave* (Z)
		4	Um *thruster* horizontal e um vertical	4. *Yaw* (rot. Z)
		5	Um *thruster* horizontal	

Fonte: Autores.

Os testes foram realizados para o intervalo de tempo de 30 segundos para cada tipo de trajetória, a fim de observar o comportamento do sistema ao longo do tempo. Cada combinação de cenário, caso e tempo foi repetida cinco vezes.

O metodo de teste para *Heave* e *Yaw* foram diferentes dos outros, devido a limitações fisicas do ROV. Para o movimento de *Heave*, foi somente ralizado o caso 4, pois esse é o unico que afeta o movimento vertical do ROV, e para o movimento de *Yaw*, foram realizados (tipos de teste para yaw a definir)

As trajetorias são realizadas a partir de uma força de *wrench* aplicada nos *thrusters* do ROV, o que gera um movimento relacionado ao tipo de trajétorio, ou seja, a trajétoria *surge* não se trata do movimento perfeito naquela direção, mas sim de uma força em X, gerando um movimento naquela direção, ainda estando sujeito as forças presentes no ambiente simulado.

Os dados coletados serão analisados estatisticamente, utilizando métricas como o erro medio e desvio padrão, para analisar o efeito do algoritmo de otimização em relação a cada tipo de falta em relação as trajetórias quando comparadas ao caso original de pleno funcionamento. Essas análises permitirão avaliar a eficácia do algoritmo de otimização na manutenção da navegabilidade do ROV em diferentes cenários de falhas.