Avanços característicos de AUV com o ROV BlueROV2

Henrique Abrantes

Maio 2025

Índice

Coordenador de Projeto Henrique Abrantes
Professor Pedro Teodoro
Equipa ROV

1 Introdução ao ROV

Desenvolvido pela BlueRobotics o BlueROV2 é um drone subaquático que foi adquirido pela faculdade. Demonstra ter uma plataforma para começar a aprender e aplicar tecnologia marítima. O artigo colocado nas referências 'Survey on advances on terrain based navigation for autonomous underwater vehicles'[MM17] demonstrou ser necessário para introduzir a tecnologia marítima nas fases iniciais da cadeira e do projeto e está disponível também uma recensão crítica feita por Henrique Abrantes [Abr25a].
Desde desenvolver software para INS autónomo, visão (processamento de imagem) e controlo este relatório visa partilhar em detalhe o trabalho realizado, desafios encontrados e conhecimentos adquiridos na cadeira de Robótica Marítima com o BlueROV2. Com várias capacidades para pesquisa demonstra ser uma ferramenta versátil para adquirir conhecimentos essenciais na área da robótica marítima e eventualmente fazer avanços tecnológicos.
Tem os seguintes sensores.

Sensor	**Tipo/Função**
Ping Sonar Altimeter and Echosounder	Medição de profundidade/distância
Ping360 Scanning Imaging Sonar	Sonar de varrimento em 360°
Newton Subsea Gripper	Garra para manipulação
Sensores IMUs	Movimento e orientação (inercial)
Sensor de pressão	Medição precisa de profundidade
Câmara	Visão subaquática em tempo real

Table 1: Sensores do BlueROV2

Para saber mais sobre os como os sensores funcionam e como o ROV se aplica na robótica marítima refiro ao artigo [MM17] e mais especificamente sobre o ROV a recensão crítica [Abr25a].

Figure 1: Caixa ROV [Dan25]

Figure 2: ROV [Dan25]

2 Cuidados a ter

Para manusear o ROV é preciso ter muitos cuidados dentro e fora de água, antes e depois de ser utilizado. Esta parte do relatório pretende fornecer um guia prático e seguro para a operação do veículo subaquático remotamente operado (ROV) BlueROV2 da BlueRobotics. Serão abordadas as melhores práticas relativas ao uso de baterias, cuidados operacionais, manutenção preventiva e corretiva, bem como um checklist de preparação para mergulho (pre-dive checklist).

2.1 Tipo de Baterias e Cuidados Associados

O BlueROV2 utiliza baterias de polímero de lítio (LiPo), geralmente do tipo 4S ou 6S com capacidade entre 10.000 a 18.000 mAh. [Lia22]

2.1.1 Cuidados com Baterias LiPo

É necessário verificar visualmente se as baterias têm sinais de inchaço, danos físicos ou fugas antes de cada utilização. No carregamento de baterias é sempre preciso supervisão e é apenas permitido utilizar carregadores LiPo balanceados (como o disponível na caixa do ROV). Se a bateria não for usada durante longos períodos de tempo (como, por exemplo, o fim do ano letivo) a bateria tem que estar com cerca de 50% de carga e armazenada em sacos LiPo à prova de fogo. Na utilização das baterias verifique sempre se está firmemente presa e se os cabos não tão em risco de serem danificados.

2.2 Cuidados na Operação do BlueROV2

É necessário manter sempre uma distância segura entre obstáculos dentro e fora de água, evitando sempre colisões. Evitem contacto bruto com o fundo (da piscina ou do mar). Verifique sempre se o cabo tether está livre de enrolamentos e evite tensão excessiva. Evite operar o ROV em temperaturas fora da faixa recomendada (0-40ºC). [Blu24a]

2.3 Manutenção Preventiva e Corretiva

Após cada missão é obrigatório lavar o ROV com água doce para preservar o ROV (principalmente os propulsores), passe com água e gire com a mão cada propulsor do ROV. Pode deixar o ROV dentro da caixa com ele ligeiramente molhado, porém mantenha a caixa aberta. Para manutenção preventiva verifique o estado dos O-rings (têm que estar lubrificados) e lubrifique com o silicone apropriado. [Blu24b]

2.3.1 Manutenção Periódica (mensal ou a cada 10 mergulhos)

É necessário abrir os compartimentos e inspecionar por condensação ou infiltração. Substituir os O-rings gastos e testar o funcionamento dos motores (thrust balance). Atualizar o firmware e software via BlueOS.

2.4 Pre-Dive Checklist

Segue-se a lista de verificação recomendada pela BlueRobotics para garantir a segurança da operação [Blu24d].

1. **Verificação Visual Geral do ROV**
    - Ausência de danos físicos.
    - Thrusters limpos e desobstruídos.
    - Tether devidamente conectado e sem danos.
2. **Integridade de Vedações**
    - O-rings em bom estado e lubrificados.
    - Compartimentos fechados e bem selados. (teste de vácuo)
3. **Sistema Elétrico e Eletrônico**

- Bateria carregada e corretamente instalada.
- Verificar tensão da bateria via interface BlueOS.
- Comunicação estabelecida com BlueOS / QGroundControl.
- Sensores operacionais (IMU, barômetro, altímetro, sonar).

4. **Testes em Terra**
   - Thrusters respondem corretamente aos comandos.
   - Câmara transmite vídeo com qualidade.
   - Luzes e outros periféricos funcionais.
5. **Planeamento da Missão**
   - Verificar condições meteorológicas e marítimas.
   - Trajeto e profundidade máxima definidos.
   - Equipamento de recuperação de emergência disponível.

A operação segura e eficiente do BlueROV2 exige uma abordagem sistemática e disciplinada. A manutenção regular, a preparação cuidadosa e o conhecimento dos limites do equipamento são essenciais para prolongar a vida útil do ROV e garantir missões bem-sucedidas. [Blu24c; Blu24a]

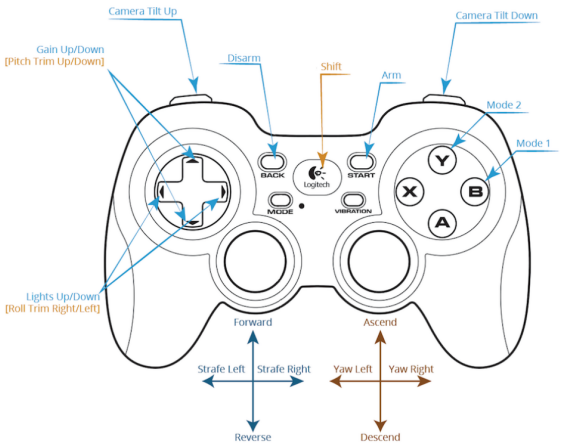

Figure 3: Funcionalidades padrão comando [Blu24a]

Nota: Este documento pode ser atualizado conforme novas recomendações da BlueRobotics ou alterações no sistema do BlueROV2. Consulte sempre a documentação oficial em: `https://bluerobotics.com` [Blu24c; Blu24a]

3 Descrição do trabalho desenvolvido

Antes de explicar em detalhe o algoritmo e trabalho desenvolvido é preciso explicar de maneira breve o software presente no ROV.

O BlueOS é uma plataforma moderna baseada em web, desenvolvida especificamente para veículos subaquáticos como o BlueROV2. É executado a bordo do ROV (normalmente num Raspberry Pi) e oferece uma interface acessível via navegador que permite a configuração e monitorização do sistema em tempo real.

Funcionalidades principais incluem:

- Gestão de dispositivos: detecção e configuração de sensores (IMU, sonar, GPS, etc.).
- Atualizações e extensões: sistema de plugins e atualizações automáticas que facilitam a manutenção.
- Integração com o QGroundControl via MAVLink, facilitando a comunicação entre o ROV e o operador.
- Acesso remoto: suporte para acesso remoto ao sistema, útil para testes e manutenção fora do local.

O BlueOS é particularmente relevante para quem pretende integrar novos sensores ou algoritmos personalizados no ROV, pois permite fácil acesso aos dados e à estrutura de comunicação do sistema. Foi utilizado para criar ativar um MAVLink endpoint para podermos aceder via script Python o ROV e utilizado para saber quais os IPs/porta série dos sensores e da câmara.

O QGroundControl (QGC) é uma estação de controlo de solo cross-platform utilizada no computador de bordo do ROV e usa o protocolo MAVLink. É utilizado para planear, monitorizar e controlar as missões do ROV em tempo real.[Pro24]

Principais características:

- Interface gráfica rica e intuitiva, com visualização de dados em tempo real.
- Planeamento de missões autónomas, com definição de pontos de passagem (waypoints) e ações associadas.
- Calibração de sensores como IMUs, magnetómetros e barómetros.
- Visualização de telemetria, mapas, gráficos e mensagens do sistema.

As etapas/projetos que o professor Pedro Teodoro nos propôs foram:

- Manter a profundidade.
- Colocar o ROV a navegar autonomamente através de INS.
- Utilizar a Câmara do ROV para o mesmo passar dentro de um arco dentro de água de maneira autónoma.

3.1 Métodos de comunicação

Para conseguir alcançar os objetivos propostos é preciso conseguir comunicar com o ROV sem ser a utilizar o software QGroundControl e o BlueOS. Os métodos de comunicação usados foram os seguintes,

Método de comunicação	**Sensor**
Comunicação Série	Ping360 Scanning Imaging Sonar, Ping Sonar Altimeter and Echosounder, Câmara
MAVLink	Sensores IMUs, Sensores de pressão

Table 2: Métodos de comunicação

Comunicação serie Para descobrir qual o endereço dos sensores utilizou-se o PingViewer, pois ele faz uma busca da rede e dá o endereço e a porta dos sensores Ping disponíveis, porém também se pode usar o software BlueOS na aba dos sensores.

PyMavLink PyMavLink foi utilizado para estabelecer uma conexão com o ROV via Script de Python. Pymavlink é uma biblioteca de processamento de mensagens MAVLink de baixo nível e de uso geral.

Para utilizar MAVLink primeiro ativou-se no BlueOS um MAVLinkendpoint. O MAVLinkendpoint usado foi o GCSServerLink (Ground Control Station Server Link). É um endpoint UDP que permite um GCS (Ground Control

Station) conectar ao veículo via protocolo MAVLink. O GCS envia comandos para o ROV que solicitam dados de telemetria, recebendo em troca os dados pedidos, estados do sistema, dados dos sensores puro ou dados dos sensores previamente tratados. Por ser um Endpoint UDP foi preciso fazer validação dos dados antes de enviar para o algoritmo.

3.1.1 MAVLink Router

O MAVLink Router é uma aplicação que distribui mensagens MAVLink entre múltiplos pontos de conexão. Permitindo que pacotes MAVLink sejam distribuídos para um único destino ou para múltiplos endpoints, dependendo do endereço de destino.

Tivemos que utilizar MAVLink Router porque o QGroundControl fechava o único endpoint disponível para comunicação com o ROV. Para configurá-lo tivemos que instalar o MAVLink router no computador do ROV que serviu de router entre o QGroundControl e os Scripts de Python.

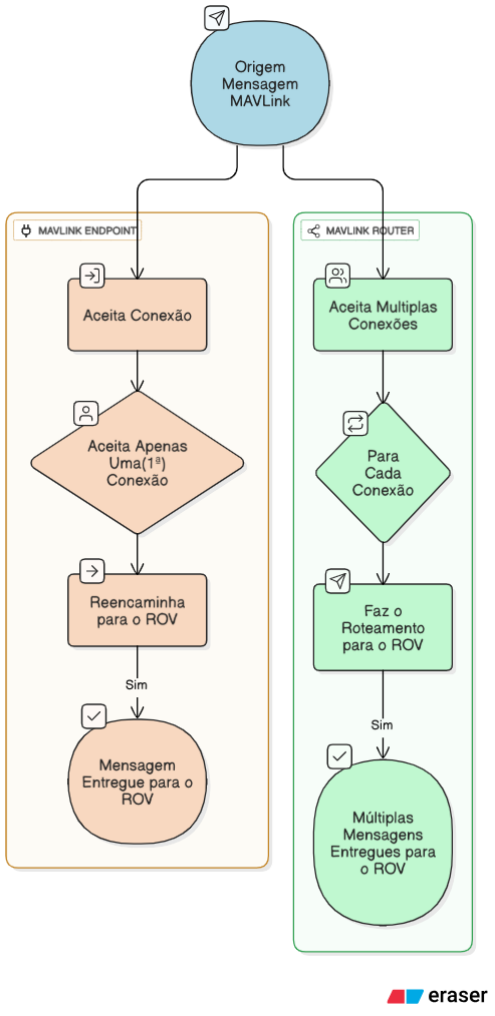

Figure 4: MAVLink endpoint vs MAVLink Router

3.1.2 Passos de configuração / Manual de funcionamento

Nesta secção vamos abordar os comandos essenciais/usados, os IPs e a configuração geral realizada no MAVLink router e no computador do Host. Para manual de funcionamento/comandos to MAVLink refiro ao site do MAVLink [Con25]

Inicialmente foi configurado um MAVLink endpoint no BlueOS com o IP 0.0.0.0 : 14550. Para utilizar o MAVLink Router tivemos que o instalar no WSL (Windows Subsystem for Linux). Para o configurar criámos o ficheiro *main.conf* onde nele colocamos as configurações pretendidas. Ficheiro *main.conf*,

```
[General]
ReportStats=false
MavlinkDialect=auto
Log=/tmp/logs
LogMode=always

[UdpEndpoint blue_rov]
Mode=Normal
Address=0.0.0.0
Port=14550

[UdpEndpoint qgroundcontrol]
Mode=Normal
Address=0.0.0.0
Port=14551

[UdpEndpoint python_script]
Mode=Normal
Address=0.0.0.0
Port=14552
```

Porquê 0.0.0.0 e não outro IP? O IP 0.0.0.0 ouve de qualquer interface. Não à problemas de segurança envolvidos porque estamos num ambiente isolado. Para colocar tudo a funcionar em sintonia configurámos uma ponte entre o computador host e o WSL.

Inicialmente para colocar o MAVLink Router a funcionar era preciso fazer este comando, $./build/src/mavlink-routerd -c ./main.conf$. Para facilitar o uso configurei-o como serviço configurando o ficheiro *mavlink-router.service*,

```
[Unit]
Description=MAVLink Router
After=network.target
[Service]
ExecStart=/home/tespria/mavlink-router-master/build/mavlink-routerd -c /home/tespria/m
Restart=on-failure
User=tespria
[Install]
WantedBy=multi-user.target
```

Localização do ficheiro: $/etc/systemd/system/mavlink-router.service$

Para colocar o MAVLink Router a funcionar é necessário executar os seguintes comandos, $sudo systemctl daemon-reload$, $sudo systemctl restart mavlink-router$, $sudo systemctl status mavlink-router$.

Para verificar a configuração do MAVLink/analisar o tráfego utilizei o WireShark no windows e o mavproxy no ubuntu (WSL). $/.local/bin/mavproxy.py --master = udp : 127.0.0.1 : PORTA$

Ao colocar o MAVLink Router como serviço consegue-se analisar melhor o seu funcionamento bem como fica com um método de funcionamento mais otimizado.

3.2 Análise de telemetria

A análise de telemetria dos sensores é essencial para saber se é necessário aplicar um filtro nos dados que estão a ser recebidos dos sensores. O sensor sob analise é o Ping Sonar Altimeter and Echosounder. Para fazer a análise desse sensor foi preciso manter a profundidade do ROV, para o fazer utilizou-se um controlador PID.

3.3 Controlador PID (Proporcional integral derivativo)

"O controlador PID desempenha um papel fundamental na estabilidade e precisão de sistemas automatizados como o BlueROV2. Este tipo de controlador é essencial para garantir que o ROV consiga manter a sua profundidade, orientação e velocidade de forma eficiente, mesmo em ambientes instáveis como o fundo do mar, onde existem correntes, alterações de pressão e outras perturbações imprevisíveis. O PID permite que o ROV reaja automaticamente a estas mudanças, ajustando os motores de forma contínua e precisa, sem necessidade de intervenção humana constante. Com os parâmetros corretamente ajustados, o sistema é capaz de manter o equilíbrio e a direção desejada, o que é crucial para operações de inspeção, recolha de dados ou tarefas técnicas em ambientes subaquáticos." [Fra25]

Um controlador PID, na área de controlo, é um algoritmo de feedback muito usado pela sua versatilidade, facilidade de implementação e resiliência durante a sua utilização. Combina três tipos de ações, a ação proporcional (Kp), ação integral (Ki) e ação derivativa (Kd). Para alcançar um melhor desempenho do controlador PID é necessário ajustar esses valores, esse ajuste pode ser feito através do método tentativa e erro até o controlador alcançar um bom comportamento, ou, através de inteligência artificial ou algoritmos de otimização para encontrar os melhores parâmetros durante a utilização do controlador, criando um ambiente dinâmico. Apesar de ser versátil e fácil de implementar existem desafios associados, como, por exemplo, ruídos no sistema (que será abordado a seguir).[25b]
Variações do controlador PID, [Fra25]

- **PI (Proporcional + Integral)**: utilizado quando o sistema é particularmente sensível a ruído.
- **PD (Proporcional + Derivativo)**: indicado para sistemas que já são naturalmente estáveis.
- **PID adaptativo**: ajusta automaticamente os seus parâmetros ao longo do tempo.
- **PID com inteligência artificial**: recorre a redes neuronais ou lógica fuzzy para tomar decisões mais inteligentes.

Onde u(t) é o sinal de saída,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_p \frac{de(t)}{dt}$$

Transformada de Laplace: (s = frequência complexa)

$$L_s = K_p + K_i/s + K_d s$$

3.3.1 Ação proporcional

A ação proporcional produz um sinal de saída que é proporcional à amplitude do erro e(t), sendo Kp a constante de proporcionalidade:

$$P_{\text{saída}} = K_p e(t)$$

3.3.2 Ação integral

A ação integral produz um sinal de saída que é proporcional à magnitude e à duração do erro, ou seja, ao erro acumulado. Isso fornece uma alternativa para corrigir o erro de off-set gerado pela ação proporcional e acelera a resposta do sistema, permitindo-o chegar ao valor de referência mais rapidamente. O sinal de saída do controlador PI pode ser descrito por:

$$I_{\text{saída}} = K_i \int_0^t e(\tau)d\tau$$

onde Ki é o ganho integral

3.3.3 Ação derivativa

A ação derivativa produz um sinal de saída que é proporcional à velocidade de variação do erro:

$$D_{\text{saída}} = K_d \frac{de(t)}{dt}$$

onde Kd é o ganho derivativo.

Como a mudança de cada parâmetro afeta a performance,

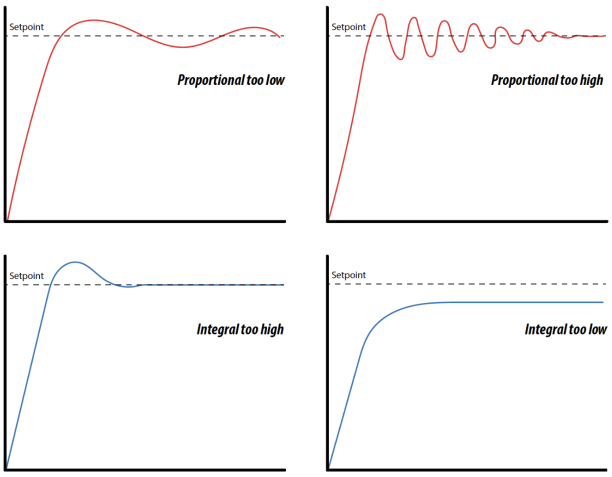

Figure 5: PID [25b]

3.3.4 Resultados e Conclusões PID

(Código no Github [Abr25d])

Nesta secção vamos analisar o comportamento do ROV bem como a telemetria obtida do PID. Os dados que vão ser analisados correspondem ao sensor Ping1d e foram obtidos na piscina da faculdade.

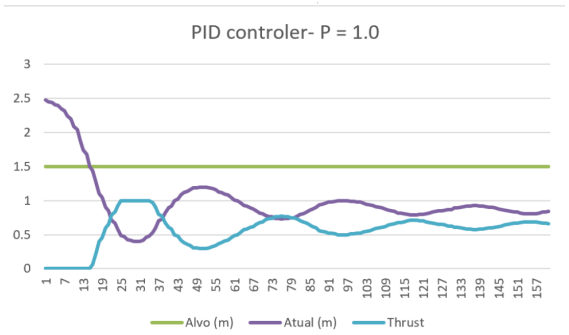

Figure 6: Primeira análise PID [Abr25b]

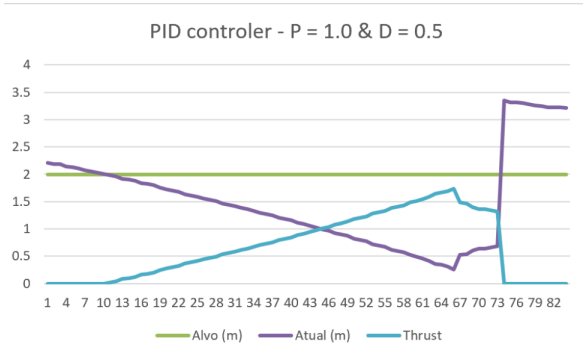

Figure 7: Segunda análise PID [Abr25c]

Podemos analisar no primeiro gráfico que o ROV chega a um nível de $thrust < 0,5$ e desce de profundidade. Convém relembrar que o sinal de controlo dos propulsores do ROV é, $u(t) = 0 = FullReverse$, $u(t) = 0,5 = Neutro$, $u(t) = 1 = Fullpropulsores$. Para normalizar este valor é preciso que quando o controlador PID quiser colocar os motores no neutro ou abrandar que esse valor não fica inferior a $0,5$. Para fazer isso é preciso que,

$$e = 0 \rightarrow u(t) = 0 \rightarrow u'(t) = 0,5$$

$$e > 0 \rightarrow u(t) > 0 \rightarrow u'(t) > 0,5$$

$$e < 0 \rightarrow u(t) < 0 \rightarrow u'(t) < 0,5$$

Sendo $u(t)$ a saída que o PID iria ter (já normalizada) e $u'(t)$ é a saída que precisa de ter para evitar o problema referido e analisado.

O ROV é apenas centrado em $0,5$ no thruster vertical (eixoZ), no thruster do eixoX (frente/trás) e no eixoY (lateral) é centrado em 0.

Acerca da mudança repentina de profundidade(Figura 4), deve-se ao facto de o sensor chegar muito perto do fundo da piscina e pela sua superfície lisa o sensor não consegue obter um valor correto, podemos analisar na 'Confiança' do sensor que é um dado recolhido do próprio.

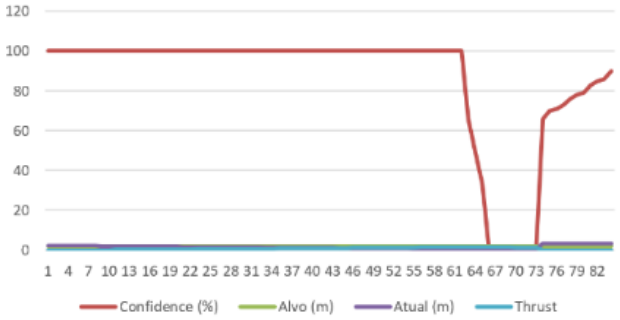

Figure 8: Análise da confiança PID [Abr25c]

Último teste PID,

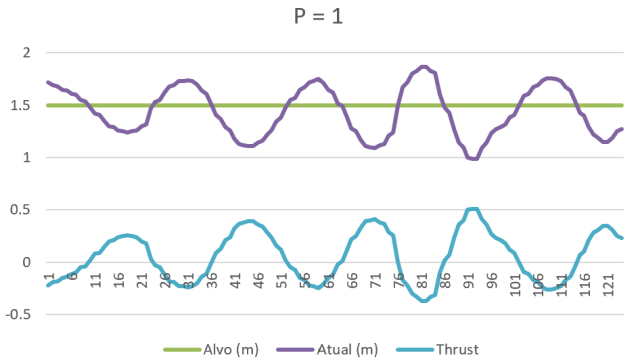

Figure 9: Última análise PID

Podemos analisar que a saída estava muito agressiva por o *P* estar muito alto. Contudo o objetivo foi concluído pois conseguimos manter o ROV "estável" mitigando o problema da primeira análise PID, pois a oscilação está centrada na profundidade alvo.

3.4 Visão (Processamento de Imagem)

O processamento de imagem foi necessário para concluir um dos objetivos do professor Pedro Teodoro.

Consistiu em receber a stream da Câmara embutida no ROV e detetar o arco vermelho na água, de seguida detetar o centroide do arco, compará-lo com o centroide da Câmara e colocar a diferença no PID, normalizar os valores e atuar os propulsores.

3.4.1 Análise da imagem

Quando recebemos a imagem do ROV temos de detetar o centroide, fazemos da seguinte maneira. A área vermelha (do arco) tem uma quantidade de pixeis do objeto, com esse valor dividimos pelo valor de pixeis totais e o resultado é a área desse objeto.

$$A_{\text{arco}} = \frac{pixeisObjeto}{pixeisTotais}$$

Esta conta não é utilizada no código, pois verificamos pixel a pixel e adicionamos $+1$ à área total do arco (processo automatizado com uma biblioteca).

Para saber as coordenadas do centroide é preciso da área do arco e uma função que pegue as coordenadas de todos os j (x) e os some ($\sum$) e divide pela área, dando o j (x) do centroide. Para saber o i (y) é fazer o mesmo, mas com todas as coordenadas do i.

$$j_{\text{centroide}} = \frac{\sum j_{\text{centroide}}}{A_{\text{arco}}}$$

$$i_{\text{centroide}} = \frac{\sum i_{\text{centroide}}}{A_{\text{arco}}}$$

A comparação entre o centroide do objeto (arco vermelho) e o centroide da Câmara é feita ao calcular a diferença entre suas coordenadas ($j_{\text{centroide objeto}}$, $i_{\text{centroide objeto}}$) e ($j_{\text{centroide Câmara}}$, $i_{\text{centroide Câmara}}$). Essa diferença representa o deslocamento necessário para alinhar o ROV ao arco.

Com a diferença calculada, o sistema chama o controlador PID, que processa o erro gerado pelo desvio e ajusta dinamicamente os comandos de movimento do ROV. O objetivo do PID é minimizar essa diferença, garantindo que o centroide da câmara coincida com o centroide do arco, permitindo uma passagem precisa dentro da estrutura subaquática.

3.4.2 Normalização HSV e Máscaras

HSV é um sistema de cores formadas pelas componentes hue (matriz), saturation (saturação) e value (valor). Este sistema define o espaço de cor da seguinte maneira: Matriz (tonalidade): Verifica o tipo de cor, abrangendo todas as cores do espectro. Saturação: Quanto menor este valor, mais o tom de cinza aparecerá na imagem. Valor (brilho): Define o brilho da cor. Todos estes valores são normalizados de 0 a 100 (%)

Porquê HSV e não RGB? No modelo HSV a matriz representa a cor pura, enquanto o valor indica o brilho, facilitando a correção de iluminação variável em ambientes subaquáticos. Ajuda também na redução de interferências pois a água absorve e dispersa luz de forma inconsistente afetando a perceção das cores, o HSV tem a saturação para compensar essa distorção. Como o HSV separa a cor e a intensidade é mais fácil identificar/destacar elementos específicos como o arco, sem ser afetado por sombras ou reflexos.

As máscaras servem para aplicar filtros e transformações em regiões específicas de uma imagem. Deste a suavização e remoção de ruído, deteção de padrões e correção de iluminação.

3.4.3 Operações morfológicas

Operações morfológicas foram utilizadas para analisar a estrutura do arco na imagem. São baseadas na morfologia matemática e são úteis para segmentação, filtragem e análise de formas.

3.5 Resultados e Conclusões Visão

(Código no Github [Abr25d])

Para abrir a câmara do ROV tivemos os seguintes desafios. O Stream da câmara está a ser feito diretamente no USB na porta $\backslash dev \backslash dev2$ ou no IP 192.168.2.1 : 14???, porém o computador host não consegue abrir a câmera.

Para o fazer foi preciso utilizar o VLC. No BlueOS temos a opção de extrair o ficheiro SDP da câmara (que é um ficheiro informativo usado para descrever sessões de transmissão de media) e o abrimos através do VLC. A biblioteca CV2 e o VLC não são compatíveis então para resolver este problema tive que converter cada frame do VLC para uma variável no código,

```
def conversion(player):
    snapshot_path = "vlc_snapshot.png"
    result = player.video_take_snapshot(0, snapshot_path, 640, 480)
    if result != 0:
        raise RuntimeError("Erro ao capturar frame do VLC.")
    frame = cv2.imread(snapshot_path)
    if frame is None:
        raise RuntimeError("Erro ao ler snapshot.")
    return frame
```

Aplicando o algoritmo abordado nos temas anteriores temos os seguintes resultados:

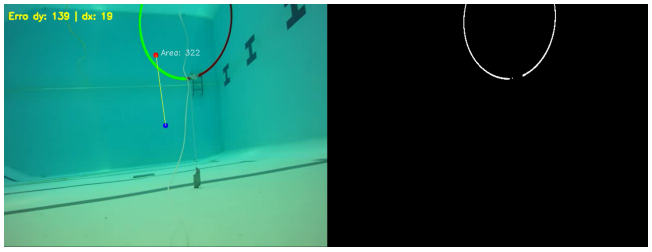

Figure 10: ROV a detetar o arco

O ROV detetou com sucesso o arco, porém, ao atuar podemos observar que o ROV fica descontrolado. A configuração PID usada foi com o $P = 1$, (**o mesmo do que o último teste PID abordado neste relatório**), logo a saída do PID estava muito agressiva. Por essas razões o ROV atuou com os propulsores para a frente e para cima ao mesmo tempo causando uma ligeira inclinação para cima, essa inclinação causou com que o ROV detetasse o teto da piscina e começasse a aplicar o algoritmo no que estava a detetar, causando a instabilidade e o insucesso do objetivo.

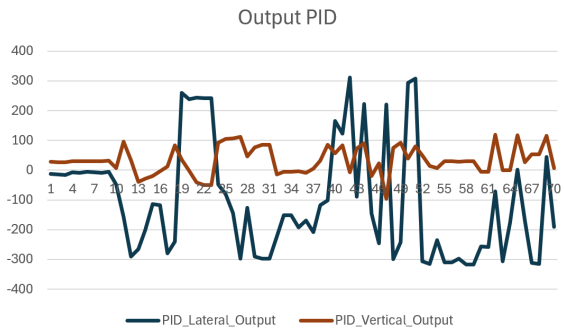

Figure 11: Saída PID

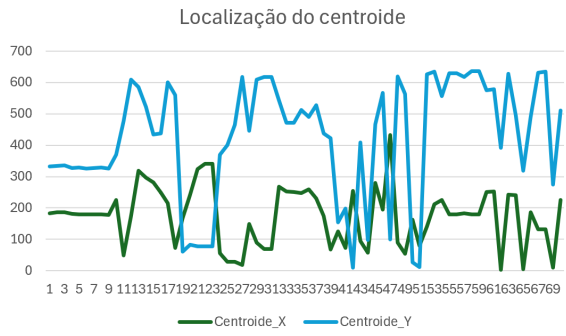

Figure 12: Localização do Centroide

Contudo, foi bem sucedido o experimento pois conseguimos que o ROV detetasse o arco e atuasse mediante a localização do centroide. Vídeo disponível no GitHub do projeto [Abr25d].

4 Conclusão

Foi me dada a responsabilidade e a confiança de ser o técnico responsável pela manutenção do ROV, e, consequentemente na cadeira de robótica marítima, ter ficado a gerir o projeto e as diferentes equipas de desenvolvimento e de pesquisa.

Foi muito interessante ter aprendido tecnologias como TBN e navegação acústica apesar de não ter sido possível fazer o desenvolvimento por não haver os sensores adequados para a recolha dos dados necessários. Foi também muito recompensador ver as as equipas a se adaptarem à mudança da direção do projeto.

Este projeto foi não só um desafio de aprendizagem e adaptação a novas tecnologias, mas também um grande desafio de gestão de equipas.

Equipa	Membros
Pesquisa	Francisca Grazina, Diogo Soares
Visão	Daniel Mendes, João Stoffel, Francisco Azevedo
Manutenção	Rafael, Eurico Martins
Comunicação	Guilherme Santos, André
Suporte Operacional	Bruno Machado, Daniel Mendes, Stoffel
TBN, Nav. Acústica	Daniel Mendes, João Stoffel
Desenvolvimento e Implementação	Tiago Ferreira, Henrique Abrantes

Table 3: Lista de Equipas

Membros
Francisca Grazina, Diogo Soares
João Stoffel, Francisco Azevedo
Rafael, Eurico Martins
Guilherme Santos, André
Bruno Machado, Daniel Mendes
Tiago Ferreira, Henrique Abrantes

Table 4: Lista de Membros

(Todas as referências a relatórios ou documentos entregues pelas respetivas equipas foram feitas)

Sugestão para um próximo projeto com o ROV: Um bom projeto seria fazer a nossa própria plataforma baseada em Python para controlar o ROV. Não refiro nada complexo de front-end mas o nosso programa controla o ROV, sem precisar do QGroundControl.

References

[25a] *Modelo de Cores HSV*. Baseado em aulas do professor Pedro Teodoro e caderno. 2025. URL: https://pt.wikipedia.org/wiki/HSV.

[25b] *PID - Controlador Proporcional Integral Derivativo*. Baseado em aulas do professor Pedro Teodoro e caderno. 2025. URL: https://pt.wikipedia.org/wiki/Controlador_proporcional_integral_derivativo.

[Abr25a] Henrique Abrantes. *Recensão crítica sobre "Survey on advances on terrain based navigation for autonomous underwater vehicles"*. Tech. rep. Recensão Crítica. Escola Náutica Infante Dom Henrique, 2025.

[Abr25b] Henrique Abrantes. *Recolha de Dados1*. telemetria da recolha de dados do PID. 2025.

[Abr25c] Henrique Abrantes. *Recolha de Dados2*. telemetria da recolha de dados do PID. 2025.

[Abr25d] Henrique Abrantes. *Repositório GitHub*. Acesso privado. 2025.

[Blu24a] BlueRobotics. *BlueOS Documentation*. Acesso em maio de 2025. 2024. URL: https://docs.bluerobotics.com/blueos/.

[Blu24b] BlueRobotics. *BlueROV2 Maintenance Guide*. Acesso em maio de 2025. 2024. URL: https://bluerobotics.com/learn/bluerov2-maintenance-checklist/.

[Blu24c] BlueRobotics. *BlueROV2 Technical Documentation*. Acesso em maio de 2025. 2024. URL: https://bluerobotics.com/learn/bluerov2-technical-details/.

[Blu24d] BlueRobotics. *Pre-Dive Checklist for BlueROV2*. Acesso em maio de 2025. 2024. URL: https://bluerobotics.com/learn/bluerov2-pre-dive-checklist/.

[Coi] Miguel Tavares Coimbra. *Capítulo III – Processamento de Imagem*. URL: https://www.dcc.fc.up.pt/~mcoimbra/lectures/PSI_1011/Aula%204%20-%20Capitulo%20III.pdf.

[Con25] MAVLink Router Contributors. *MAVLink Router: Route MAVLink packets between endpoints*. A tool for routing MAVLink messages between multiple endpoints. 2025. URL: https://github.com/mavlink-router/mavlink-router.

[Dan25] Francisco Azevedo Daniel Mendes João Stoffel. *Relatório do Projeto 'ROV'*. Equipa de Visão, TBN e Nav. Acústica. 2025.

[Fra25] Diogo Soares e Francisca Grazina. *Controlador PID*. Equipa de Pesquisa. 2025.

[Lia22] Oscar Liang. *LiPo Battery Safety Guide: Tips and Precautions*. Acesso em maio de 2025. 2022. URL: https://oscarliang.com/lipo-battery-guide/.

[MM17] José Melo and Aníbal Matos. "Survey on advances on terrain based navigation for autonomous underwater vehicles". In: *Ocean Engineering* 139 (2017), pp. 250–264. DOI: 10.1016/j.oceaneng.2017.04.047.

[Pro24] QGroundControl Project. *QGroundControl User Guide*. Acesso em maio de 2025. 2024. URL: https://docs.qgroundcontrol.com/.

[Vin25] Vinicius. *Modelos de Cores HSV e LAB*. Acesso em maio de 2025. 2025. URL: https://www.monolitonimbus.com.br/modelos-de-cores-hsv-e-lab/.