JetBot: Sensor Fault Detection and Obstacle Avoidance for Robust Navigation

Robotics Engineering Project

Author: **David Furtado**

Year: 2025

Contents

1 Objetivos do Projeto

O presente trabalho tem como objetivo principal o desenvolvimento e a validação de um sistema de decisão para um robô móvel autónomo, dotando-o de capacidades essenciais para uma operação segura e fiável. Para alcançar este propósito, o projeto foi estruturado em torno de dois objetivos fundamentais e complementares:

1. **Desenvolvimento de um Sistema de Deteção de Falhas:** O primeiro objetivo consiste em implementar uma camada de monitorização capaz de detetar anomalias no comportamento do robô. Esta tarefa envolve a análise dos dados de atuação e de estado (posição) para identificar discrepâncias que possam indicar uma falha no sistema, como um problema nos atuadores ou nos sensores. A concretização deste objetivo é um passo fundamental para garantir a integridade e a segurança da plataforma robótica.

2. **Implementação de um Sistema de Desvio de Obstáculos:** O segundo objetivo foca-se na navegação segura em ambientes com obstáculos. Pretende-se projetar e implementar um sistema de decisão que:

   - Detete proativamente o risco de colisão com base na trajetória futura do robô.
   - Calcule e execute uma manobra de desvio suave e eficiente para contornar o obstáculo em segurança.
   - Garanta que, após a manobra, o robô consiga retomar a sua missão original a partir de um ponto e com uma orientação adequados.

   A validação deste sistema será feita através da sua aplicação aos dados experimentais, demonstrando a sua capacidade de corrigir uma trajetória que, de outra forma, resultaria numa colisão.

A concretização destes dois objetivos resultará numa arquitetura de controlo robusta, que não só permite ao robô cumprir a sua missão, mas também reagir de forma inteligente a imprevistos, sejam eles falhas internas ou obstáculos externos.

2 Fundamentação Teórica e Modelação do Sistema

Esta secção estabelece as bases teóricas e matemáticas utilizadas ao longo do projeto, incluindo o modelo dinâmico do robô, a formalização das incertezas e a técnica de linearização para sistemas não-lineares.

2.1 Modelo Dinâmico do Robô Jetbot

O robô Jetbot é um veículo de duas rodas cujo movimento no plano pode ser aproximado por um modelo de robô diferencial. A sua dinâmica em tempo contínuo é descrita pelas seguintes equações de estado:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\theta(t)) \\ v(t)\sin(\theta(t)) \\ \omega(t) \end{bmatrix} = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

onde o vetor de estado $\mathbf{x}(t) = [x(t), y(t), \theta(t)]^T$ representa a pose do robô (posição e orientação), e o vetor de controlo $\mathbf{u}(t) = [v(t), \omega(t)]^T$ contém as velocidades linear e angular.

Para implementação em sistemas digitais, utilizamos uma discretização por Euler de primeira ordem com um passo de amostragem Δt:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} x_k + v_k \Delta t \cos(\theta_k) \\ y_k + v_k \Delta t \sin(\theta_k) \\ \theta_k + \omega_k \Delta t \end{bmatrix} \tag{2}$$

Este modelo discreto forma a base para a predição de estado no Filtro de Kalman e na simulação do controlador.

2.2 Modelação de Incertezas

Qualquer sistema físico está sujeito a incertezas. No contexto deste projeto, consideramos duas fontes principais de ruído, modeladas como processos estocásticos Gaussianos:

- **Ruído do Processo (w_k):** Representa a incerteza no modelo de movimento, causada por fatores como o deslize das rodas ou imperfeições nos atuadores. É adicionado à equação de estado:

  $$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + w_k, \quad w_k \sim \mathcal{N}(0, Q) \tag{3}$$

  onde Q é a matriz de covariância do ruído do processo.

- **Ruído da Medição (v_k):** Representa a imprecisão dos sensores que fornecem a pose do robô. A equação de medição é:

  $$\mathbf{z}_k = H\mathbf{x}_k + v_k, \quad v_k \sim \mathcal{N}(0, R) \tag{4}$$

  onde $\mathbf{z}_k$ é a medição, H é a matriz de observação (neste caso, a identidade) e R é a matriz de covariância do ruído da medição.

2.3 Linearização de Sistemas Não-Lineares

O modelo do Jetbot é não-linear. Para aplicar ferramentas como o Filtro de Kalman, é necessário linearizar o sistema em torno de uma trajetória nominal. A Jacobiana em relação ao estado é:

$$A_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_k, \mathbf{u}_k} = \begin{bmatrix} 0 & 0 & -v_k \sin(\theta_k) \\ 0 & 0 & v_k \cos(\theta_k) \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

A sua forma discretizada, $F_k \approx I + \Delta t A_k$, é usada no EKF. A Jacobiana em relação ao controlo é:

$$B_k = \frac{\partial f}{\partial \mathbf{u}}\Big|_{\mathbf{x}_k, \mathbf{u}_k} = \begin{bmatrix} \cos(\theta_k) & 0 \\ \sin(\theta_k) & 0 \\ 0 & 1 \end{bmatrix} \tag{6}$$

3 Deteção de Falhas com Monitor Baseado em Filtro de Kalman Estendido (EKF)

Esta secção detalha a implementação e a validação da camada de deteção de falhas, conforme solicitado na Pergunta 1. O objetivo é desenvolver um monitor capaz de identificar anomalias na operação do robô, comparando o seu movimento previsto com as medições reais. A ferramenta central escolhida para esta tarefa foi o Filtro de Kalman Estendido (EKF).

3.1 O Filtro de Kalman Estendido (EKF) como Monitor de Consistência

O Filtro de Kalman Estendido é um pilar fundamental da robótica e de sistemas de estimação, sendo um algoritmo recursivo que estima o estado de um sistema dinâmico a partir de medições ruidosas. A sua principal vantagem é a capacidade de lidar com modelos de movimento e de medição não-lineares, que são comuns em robótica. O EKF opera em duas fases cíclicas: Predição e Atualização.

1. Fase de Predição: O filtro usa o modelo de movimento para prever o próximo estado do robô ($\hat{x}_{k|k-1}$) e a sua incerteza associada (matriz de covariância $P_{k|k-1}$), com base no estado anterior.

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \tag{7}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{8}$$

onde f é a função de movimento não-linear, F_k é a sua Jacobiana em relação ao estado, e Q_k é a covariância do ruído do processo (incerteza do modelo).

2. Fase de Atualização: Quando uma nova medição (z_k) chega, o filtro calcula a diferença entre a medição real e a medição prevista, conhecida como **inovação**. Esta inovação é ponderada pelo Ganho de Kalman (K_k) para corrigir a predição do estado, resultando numa estimativa final mais precisa ($\hat{x}_{k|k}$) e numa incerteza reduzida ($P_{k|k}$). As equações desta fase são:

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1}) \tag{9}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{10}$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{11}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{12}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{13}$$

onde as novas variáveis são:

- $\tilde{y}_k$: A inovação ou resíduo da medição.
- h: A função de medição não-linear, que mapeia um estado para o espaço de medição.
- H_k: A Jacobiana da função de medição em relação ao estado.
- R_k: A covariância do ruído da medição (incerteza do sensor).
- S_k: A covariância da inovação.
- K_k: O Ganho de Kalman ótimo, que minimiza a variância do erro de estimação.
- I: A matriz identidade.

Justificação de Uso no Projeto

A estratégia para deteção de falhas consiste em utilizar o EKF não apenas como um estimador, mas como um **monitor de consistência**. Para este fim, o filtro foi intencionalmente "afinado" para confiar mais no seu modelo de movimento interno do que nas medições externas. Isto é conseguido atribuindo um valor baixo à covariância do ruído do processo (Q) e um valor relativamente alto à covariância do ruído da medição (R).

Desta forma, se ocorrer uma falha nos atuadores, o robô real desviar-se-á do movimento previsto pelo modelo. Como o filtro confia no seu modelo, a sua estimativa de estado ($\hat{x}$) seguirá a trajetória ideal, enquanto a medição real (z_k) seguirá a trajetória com falha. A divergência entre estas duas trajetórias, torna-se o indicador da anomalia.

3.2 Investigação de uma Abordagem Estatística: A Distância de Mahalanobis

Na busca por um critério de deteção robusto e com significado estatístico, foi investigada a utilização da **Distância de Mahalanobis**. Esta métrica é teoricamente superior à Distância Euclidiana, pois mede a "surpresa" de uma nova medição, levando em conta a incerteza (covariância) estimada pelo próprio filtro. A sua principal vantagem é permitir a definição de um limiar de deteção com base em probabilidades estatísticas, em vez de um valor puramente empírico.

A Distância de Mahalanobis ao quadrado, para uma dada inovação no passo de tempo k, é calculada da seguinte forma:

$$d_{M,k}^2 = (z_k - h(\hat{x}_{k|k-1}))^T S_k^{-1} (z_k - h(\hat{x}_{k|k-1})) \tag{14}$$

onde z_k é a medição real, $h(\hat{x}_{k|k-1})$ é a medição prevista pelo modelo, e S_k é a matriz de covariância da inovação, que encapsula a incerteza combinada do modelo e da medição.

Para aumentar a sensibilidade e a robustez do detetor, a implementação não se baseou em medições individuais, mas sim numa abordagem de **janela deslizante**. Esta técnica consiste em acumular o valor de $d_{M,k}^2$ ($`mahalanobis_horizon_sq`$) ao longo de um horizonte de tempo recente. O valor acumulado é então comparado com um limiar ($`chi2_threshold`$) derivado da distribuição Qui-quadrado (χ^2), que é o teste de hipótese estatisticamente correto para esta métrica.

3.2.1 Resultados e Análise da Limitação

Apesar da sua robustez teórica, esta abordagem não conseguiu detetar a falha presente nos dados. Foram testadas diversas configurações para as matrizes de covariância do ruído (Q e R), mas em nenhuma delas a métrica de teste atingiu o limiar de deteção.

A Figura 1 ilustra a razão desta falha a nível de dados. Observa-se que o valor da métrica acumulada ($`mahalanobis_horizon_sq`$) permanece em níveis muito baixos, na ordem de 0.1, enquanto o limiar estatístico para uma deteção fiável se encontrava num valor significativamente superior, próximo de 12.0. Esta grande disparidade indica que, para a dinâmica específica deste problema, a inovação gerada pela falha do atuador não foi suficientemente forte para ser considerada estatisticamente anómala.

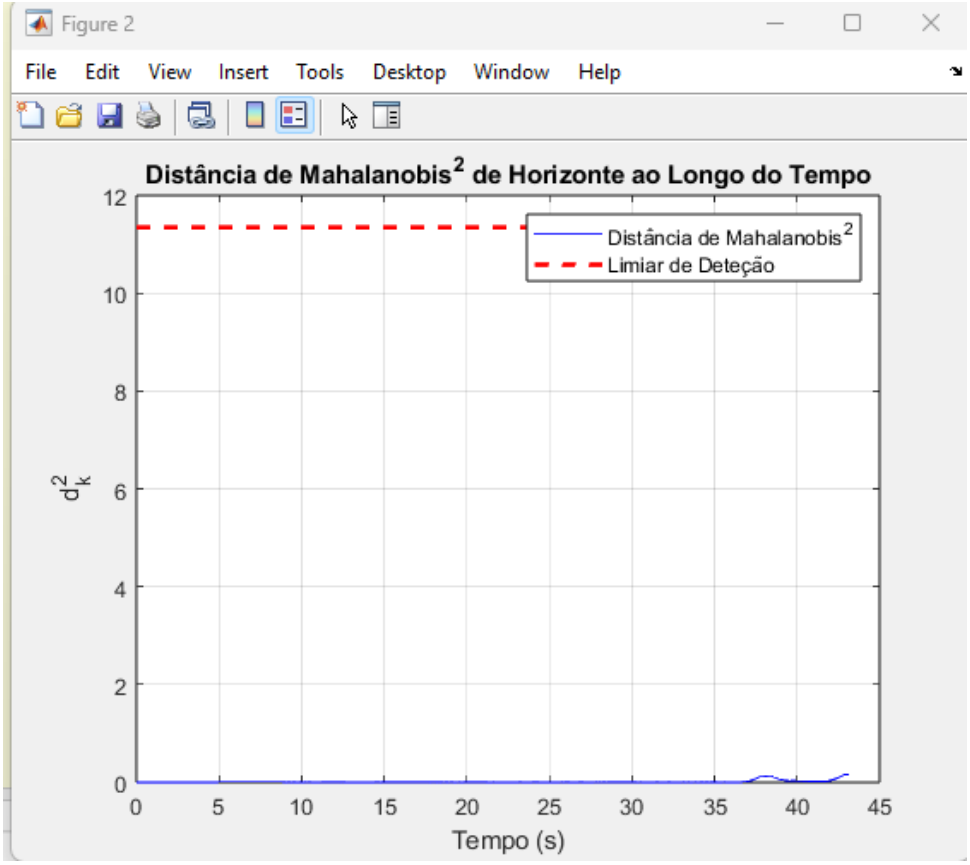


Figure 1: Teste com a Distância de Mahalanobis acumulada. A métrica de teste (em azul) permanece muito abaixo do limiar estatístico χ^2 (em vermelho), não conseguindo sinalizar a anomalia.

A consequência prática desta falha de deteção é que o sistema de monitorização não consegue reconhecer a anomalia. Como o alarme nunca foi acionado, o sistema continuaria a operar sob a suposição de que tudo está normal, enquanto o robô real se desvia cada vez mais da sua trajetória esperada devido à falha do atuador. Este resultado valida a decisão de abandonar esta abordagem estatística em favor de uma solução mais pragmática e sensível à divergência física para o problema em questão.

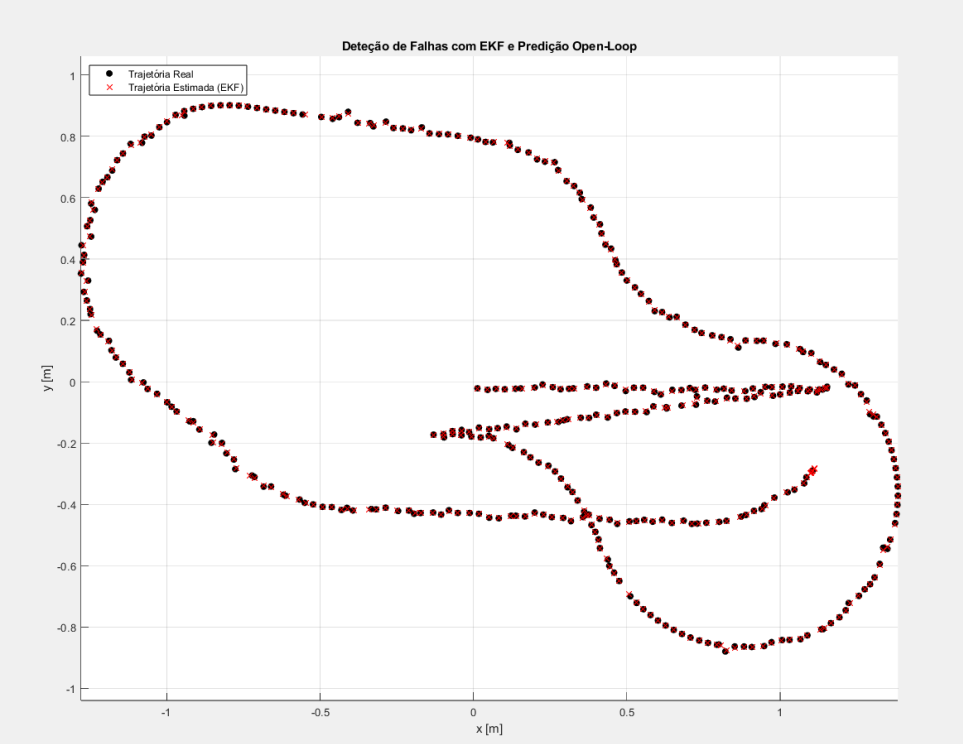


Figure 2: Validação da inadequação do método de Mahalanobis para este problema. A ausência de um alarme por parte do detetor permite que a falha do atuador se propague sem ser corrigida, resultando numa trajetória real que se desvia significativamente da nominal.

3.3 Implementação e Validação do Detetor de Falhas

Após a análise de diferentes métricas, a abordagem final implementada para a deteção da falha baseou-se numa solução mais direta e pragmática: a **Distância Euclidiana** entre a posição estimada pelo filtro e a posição medida.

A métrica de deteção é simplesmente:

$$d_{E,k} = \sqrt{(x_{medido,k} - \hat{x}_{estimado,k})^2 + (y_{medido,k} - \hat{y}_{estimado,k})^2} \tag{15}$$

Esta abordagem funciona precisamente porque ignora a matriz de covariância (que, como visto na secção anterior, se revelou pouco sensível neste contexto) e mede diretamente a divergência física entre o que o robô *deveria* fazer (a estimativa do EKF) e o que ele *realmente* fez (a medição). Um limiar de deteção foi definido empiricamente, representando a distância máxima aceitável entre as duas trajetórias em condições normais de operação.

Resultado Final

A aplicação desta estratégia aos dados experimentais demonstrou a sua eficácia. A Figura 3 apresenta a trajetória do robô, onde os pontos pretos representam a trajetória real (medida) e os pontos vermelhos representam a trajetória ideal estimada pelo EKF.

Quando a falha de atuador ocorre, as duas trajetórias começam a divergir. O sistema monitoriza a Distância Euclidiana entre elas e, assim que esta excede o limiar predefinido, a falha é sinalizada. Como se pode observar, a deteção foi acionada com sucesso no instante de tempo de **37.81 segundos**, assinalado com uma estrela magenta na figura. Este resultado valida a abordagem implementada como uma solução robusta e eficaz para o problema de deteção de falhas proposto.

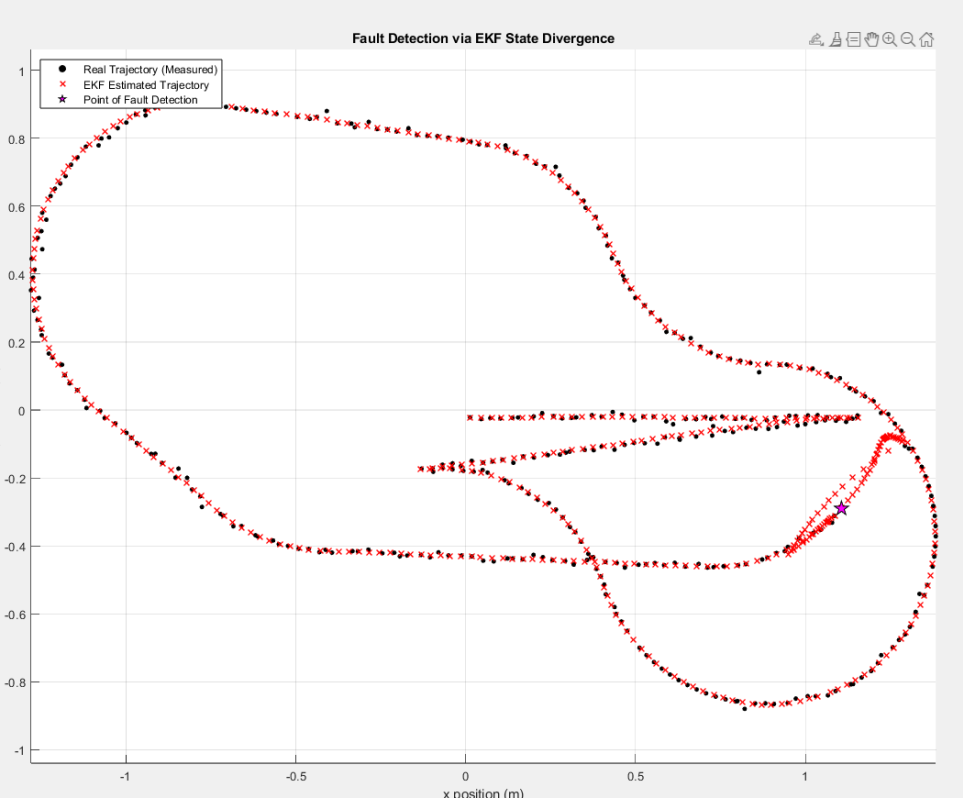


Figure 3: Deteção de falha bem-sucedida com a abordagem da Distância Euclidiana. A falha é detetada quando a distância entre a trajetória real (preta) e a estimada (vermelha) excede o limiar, no ponto assinalado a magenta.

4 Desenvolvimento do Controlador de Desvio de Obstáculos

A segunda parte do projeto foca-se no desenvolvimento de um sistema de decisão e controlo seguro, capaz de detetar e contornar obstáculos circulares. A solução final baseia-se num robusto controlador de seguimento de trajetória do tipo Proporcional-Integral (PI). No entanto, o seu desenvolvimento partiu de uma implementação inicial mais simples, baseada num controlo puramente Proporcional, cujas limitações motivaram a evolução para a abordagem final.

4.1 Ponto de Partida: Controlador Proporcional (P)

A primeira implementação para guiar o robô foi um controlador Proporcional, onde as ações de controlo são diretamente proporcionais aos erros de distância e de ângulo. As leis de controlo são definidas da seguinte forma:

$$v_k = K_v \cdot \|p_{alvo} - p_k\| \tag{16}$$

$$\omega_k = K_h \cdot (\text{atan2}(y_{alvo} - y_k, x_{alvo} - x_k) - \theta_k) \tag{17}$$

onde (v_k, ω_k) são as velocidades comandadas e (K_v, K_h) são os ganhos.

Embora funcional para uma aproximação geral, esta abordagem tem uma limitação intrínseca no controlo da velocidade: à medida que o robô se aproxima do alvo, a distância diminui, o erro tende para zero e, consequentemente, a velocidade comandada (v_k) também tende para zero. Este comportamento pode fazer com que o robô abrande excessivamente antes de atingir o seu objetivo, resultando num seguimento de trajetória lento e em erros de estado estacionário. Para garantir uma performance mais assertiva e robusta, foi necessário refinar esta estratégia.

4.2 Implementação Final: Controlador de Seguimento PI (Pursuit Controller)

Para superar as limitações do controlador P, a solução final adotada consiste num controlador de seguimento de trajetória (também conhecido como *pursuit controller*) que utiliza uma lei de controlo Proporcional-Integral (PI) para a velocidade linear e mantém o controlo Proporcional para a velocidade angular.

Lei de Controlo PI para a Velocidade Linear (v_k): O controlo da velocidade foi melhorado com a adição de um termo integral (K_i). Este termo acumula o erro de distância ao longo do tempo, permitindo que o controlador comande uma velocidade finita mesmo quando o erro proporcional é pequeno. Isto elimina o erro de estado estacionário e garante uma aproximação mais decidida ao alvo.

$$v_k = K_v e_k + K_i \int_0^t e_k(\tau) d\tau \tag{18}$$

onde o erro de distância e_k é definido como $e_k = \|p_{alvo} - p_k\| - d_{lookahead}$.

Lei de Controlo P para a Velocidade Angular (ω_k): A velocidade angular continua a ser controlada por uma lei Proporcional simples, que se revelou eficaz para a tarefa de alinhamento:

$$\omega_k = K_h \cdot (\text{atan2}(y_{alvo} - y_k, x_{alvo} - x_k) - \theta_k) \tag{19}$$

Adicionalmente, para garantir que os comandos de controlo são fisicamente exequíveis e seguros, foram impostos limites de saturação sobre a velocidade linear e a aceleração angular. Esta arquitetura PI-P, com as devidas limitações, constitui o núcleo do nosso sistema de navegação

4.3 Otimização da Navegação com Waypoints Intermédios

Durante os testes, foi identificada uma limitação crítica na lógica de transição entre os modos de operação. Após completar o arco de desvio, ao entrar no modo de retoma (‘resume‘), o controlador tentava seguir o caminho mais curto para o waypoint final. No entanto, em certas geometrias, este caminho direto intersetava novamente o obstáculo, como se pode ver na Figura 4.

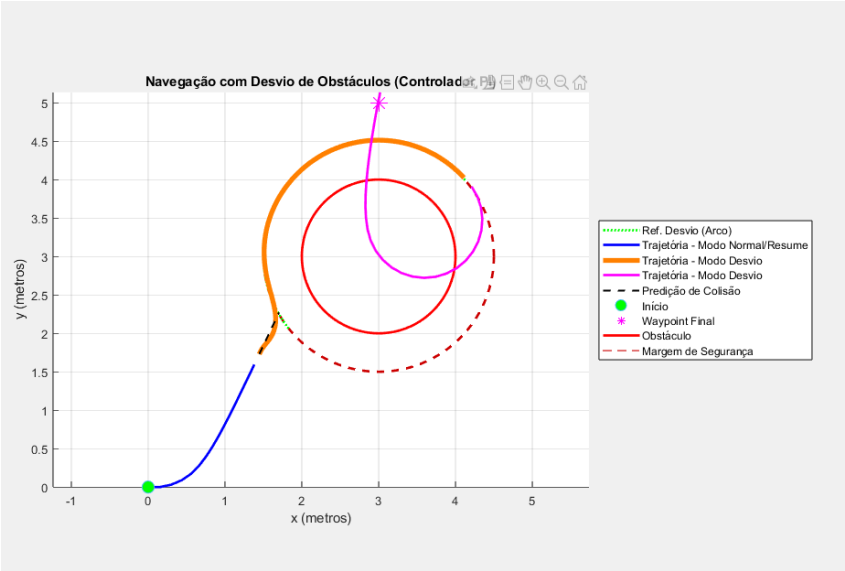


Figure 4: Comportamento sub-ótimo do sistema sem a lógica de otimização. Após o desvio, o robô tenta o caminho mais curto para o waypoint, resultando numa nova trajetória de colisão.

Para resolver este problema, foi implementada uma **lógica de decisão hierárquica** no modo de retoma. Em vez de seguir cegamente o waypoint final, o sistema primeiro verifica se o caminho direto está livre. Se não estiver, ele gera um **waypoint intermédio temporário**, calculado com base na sua posição atual, para se afastar do obstáculo de forma segura. Apenas quando o caminho para o waypoint final se torna desobstruído é que o sistema o define como alvo.

Esta otimização transformou a máquina de estados de um sistema puramente reativo para um com uma capacidade de planeamento tático simples. O resultado, visível na trajetória final do projeto, é uma manobra de desvio e retoma que é não só segura, mas também suave e eficiente, validando a importância desta camada de decisão adicional.

4.4 Otimização da Manobra de Desvio com "Ponto de Fuga"

A otimização mais significativa do sistema de desvio foi a implementação de uma lógica de **"ponto de fuga"**. Em vez de seguir um arco de duração fixa, o robô verifica continuamente se já existe uma linha de visão desobstruída para o seu alvo final (o ponto de reencontro). Esta verificação é realizada pela função `isPathClear`, cujo princípio geométrico é detalhado abaixo.

4.4.1 Princípio Geométrico da Verificação

O algoritmo determina a distância mínima, d, entre o centro do obstáculo (p_c) e o segmento de linha que une a posição atual do robô (p_r) ao seu alvo (p_g). Isto é conseguido através de projeção vetorial.

Primeiro, definem-se os vetores relevantes:

- $\vec{v} = p_g - p_r$: O vetor que representa o segmento de linha da trajetória.

- $\vec{w} = p_c - p_r$: O vetor que une o início do segmento ao centro do obstáculo.

De seguida, calcula-se a projeção escalar de $\vec{w}$ sobre $\vec{v}$. Esta projeção, quando normalizada pelo comprimento ao quadrado de $\vec{v}$, resulta num parâmetro t:

$$t = \frac{\vec{w} \cdot \vec{v}}{\vec{v} \cdot \vec{v}} \tag{20}$$

O valor de t indica a localização, ao longo da linha infinita definida pelo vetor $\vec{v}$, do ponto que está geometricamente mais próximo do centro do obstáculo, p_c. Para garantir que este ponto se encontra dentro do segmento de linha $[p_r, p_g]$, o valor de t é limitado ao intervalo $[0, 1]$:

$$t_{limitado} = \max(0, \min(1, t)) \tag{21}$$

O ponto mais próximo no segmento de linha (p_{prox}) é então calculado como:

$$p_{prox} = p_r + t_{limitado} \cdot \vec{v} \tag{22}$$

Finalmente, a distância mínima d é a norma da diferença entre o centro do obstáculo e este ponto:

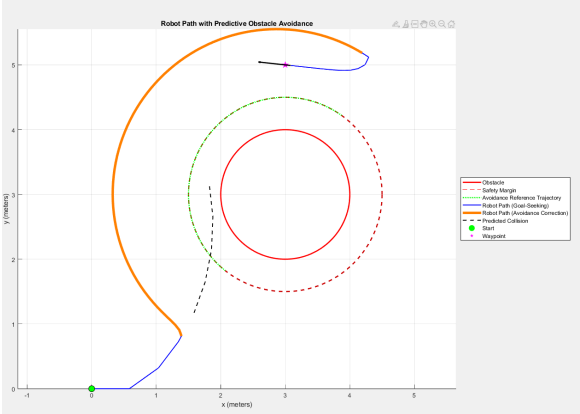
$$d = \|p_c - p_{prox}\| \tag{23}$$

Se esta distância d for superior ao raio de segurança do obstáculo, o caminho é considerado livre e o robô abandona a manobra de desvio em arco para prosseguir diretamente para o seu alvo. Esta otimização resultou em trajetórias significativamente mais curtas e eficientes.

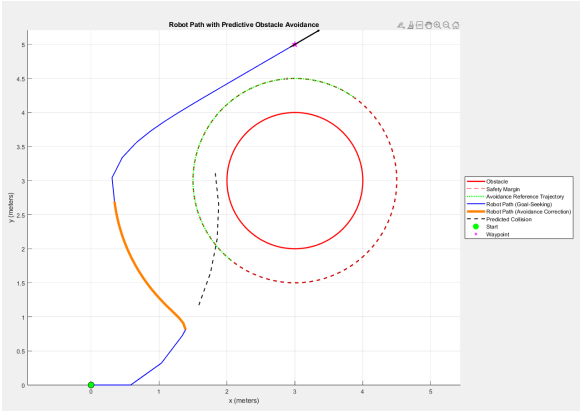
5 Resultados do Controlador e Análise Comparativa

Foram realizadas simulações para três *waypoints* distintos para analisar o impacto das melhorias.

5.1 Análise para o Waypoint [3, 5]

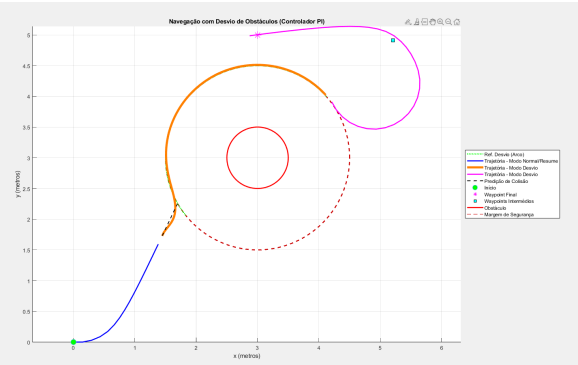


(a) Arco de desvio fixo.

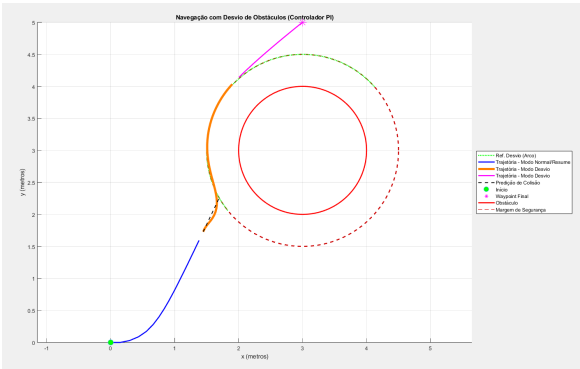


(b) Desvio otimizado ("ponto de fuga").

Figure 5: Comparação de desempenho para o controlador **P** no waypoint [3, 5].



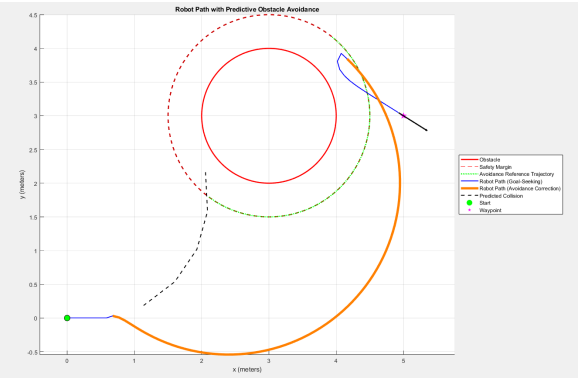
(a) Arco de desvio fixo.



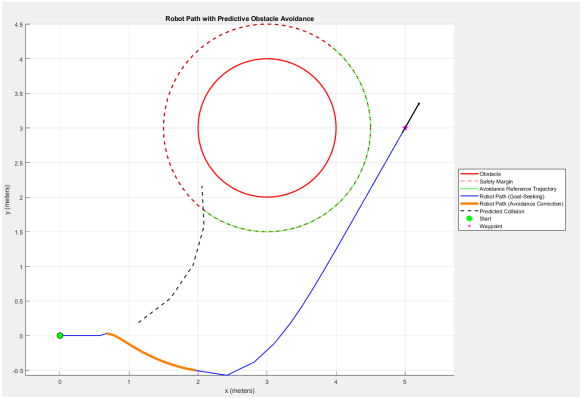
(b) Desvio otimizado.

Figure 6: Comparação de desempenho para o controlador **PI** no waypoint [3, 5].

5.2 Análise para o Waypoint [5, 3]

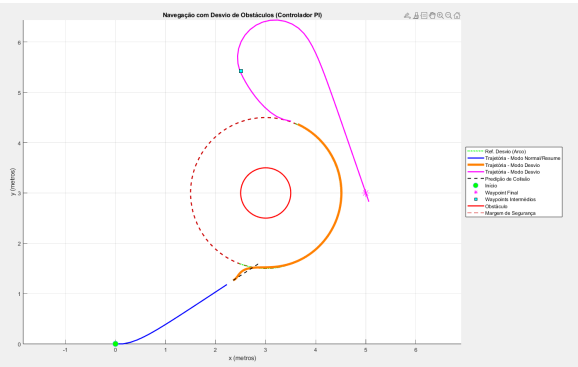


(a) Arco de desvio fixo.

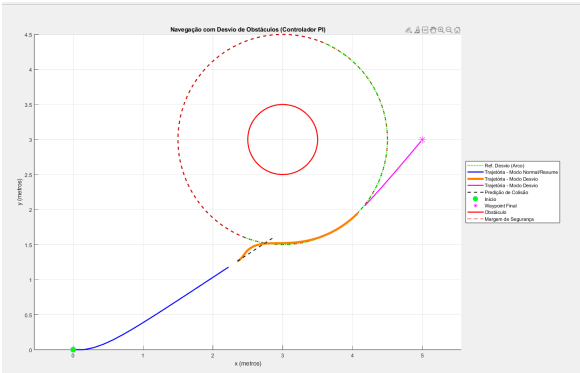


(b) Desvio otimizado ("ponto de fuga").

Figure 7: Comparação de desempenho para o controlador **P** no waypoint [5, 3].



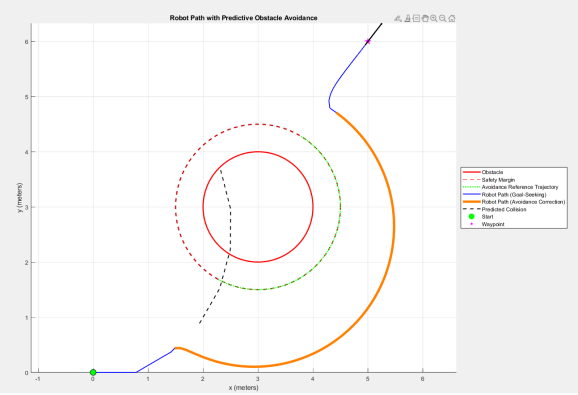
(a) Arco de desvio fixo.



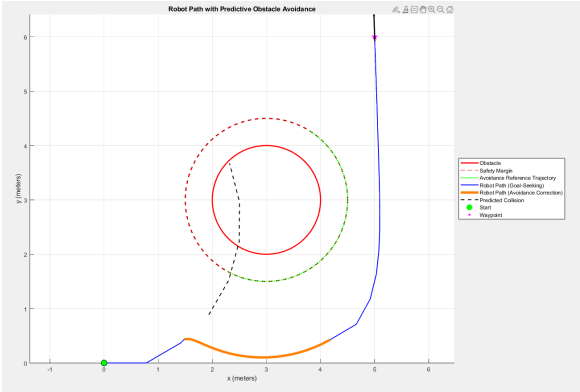
(b) Desvio otimizado.

Figure 8: Comparação de desempenho para o controlador **PI** no waypoint [5, 3].

5.3 Análise para o Waypoint [5, 6]

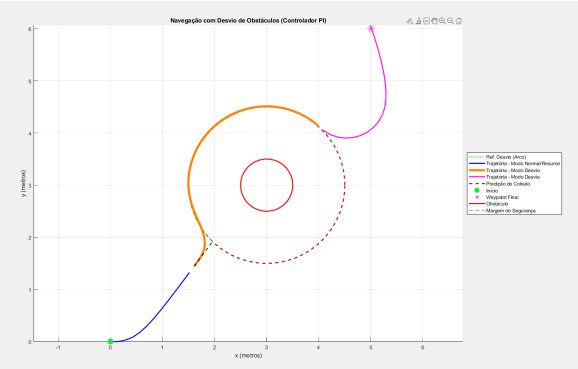


(a) Arco de desvio fixo.

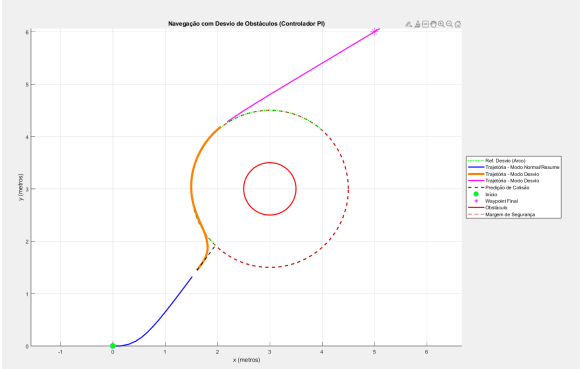


(b) Desvio otimizado ("ponto de fuga").

Figure 9: Comparação de desempenho para o controlador **P** no waypoint [5, 6].



(a) Arco de desvio fixo.



(b) Desvio otimizado.

Figure 10: Comparação de desempenho para o controlador **PI** no waypoint [5, 6].

5.4 Análise Geral dos Resultados do Controlador

A análise comparativa permite extrair duas conclusões principais:

1. **Impacto do PID:** O controlador PID produz consistentemente uma trajetória mais suave e estável.

2. **Eficácia do "Ponto de Fuga":** A otimização do desvio resulta numa trajetória significativamente mais curta e eficiente em todos os cenários.

6 Desenvolvimento e Validação do Sistema de Desvio de Obstáculos

O objetivo central do sistema de decisão desenvolvido era garantir a navegação segura do robô, detetando riscos de colisão com base na sua trajetória projetada e, quando necessário, executando uma manobra de correção autónoma. Esta secção detalha o processo iterativo de desenvolvimento, desde a implementação de um controlador inicial até à validação do sistema final integrado. O obstáculo, conforme definido no enunciado, é um círculo centrado em $[0.5, 0.5]$ com um raio de 0.25.

6.1 Abordagem Inicial e Desafios de Controlo (Controlador PI)

A primeira versão do sistema utilizou um controlador Proporcional-Integral (PI) para guiar o robô durante as manobras de desvio. A estratégia consistia em seguir um arco circular de segurança e, posteriormente, navegar para uma pose de reencontro na trajetória original. No entanto, esta abordagem revelou uma limitação fundamental na tarefa de atingir uma pose específica (posição e orientação).

Como ilustrado na Figura 11, o controlador PI teve dificuldades em alinhar o robô com a orientação final desejada de forma eficiente. A tentativa de corrigir simultaneamente os erros de posição e de orientação resultou em trajetórias sub-ótimas, caracterizadas por longos "ganchos" de alinhamento. Este comportamento instável demonstrou que um controlador PID convencional, apesar de eficaz para o seguimento de trajetórias, não é adequado para a tarefa mais exigente de estabilização de pose (posição e orientação) de um robô não-holonómico.

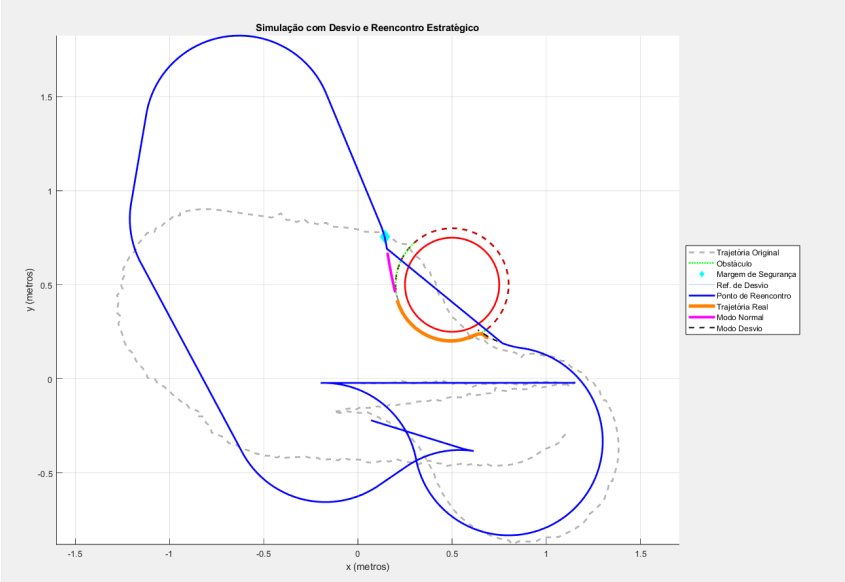


Figure 11: Resultado da simulação utilizando um controlador PID para a fase de alinhamento (resume). A trajetória ineficiente (em magenta) demonstra a dificuldade do controlador em atingir a pose de reencontro com a orientação correta.

6.2 Arquitetura de Controlo Híbrida

Face aos desafios apresentados por um único controlador para múltiplas tarefas, foi desenvolvida uma **arquitetura de controlo híbrida**. Esta abordagem seleciona o controlador mais adequado para cada fase da manobra de desvio, otimizando o desempenho e a robustez. A arquitetura utiliza dois controladores distintos: um para o seguimento da trajetória do arco e outro para a estabilização na pose de reencontro.

6.2.1 Controlo de Seguimento de Trajetória (PI)

Para a fase de desvio (*avoidance*), onde o objetivo é seguir de forma precisa a sequência de pontos que compõem o arco de segurança, foi implementado o robusto controlador de seguimento PI-P descrito na secção 4.2. A sua capacidade de anular erros de distância de forma assertiva torna-o ideal para garantir que o robô se mantém fiel à trajetória de segurança calculada.

6.2.2 Controlo de Estabilização de Pose (Kanayama)

Para a fase de alinhamento final (*resume*), a tarefa é mais exigente: o robô precisa de chegar a uma pose específica (posição e orientação) de forma suave e sem overshoot. Para esta tarefa, foi implementado o controlador de estabilização de pose de Kanayama [1, 2]. Este método reformula o erro de controlo num sistema de coordenadas polares (ρ, α, β), permitindo um controlo acoplado da posição e da orientação. As suas leis de controlo são:

$$v = K_\rho \cdot \rho \tag{24}$$

$$\omega = K_\alpha \cdot \alpha + K_\beta \cdot \beta \tag{25}$$

onde $(K_\rho, K_\alpha, K_\beta)$ são ganhos sintonizados. Esta abordagem é projetada para gerar trajetórias de convergência eficientes, eliminando as instabilidades que um controlador mais simples poderia apresentar nesta manobra de precisão.

6.3 Validação Final do Sistema Integrado

O sistema final integra o detetor de colisão com a arquitetura de controlo híbrida. A Figura 12 apresenta o resultado da simulação completa, aplicando o sistema de decisão sobre a trajetória experimental fornecida. A sequência de eventos e os controladores utilizados em cada fase são:

1. **Modo Normal (Azul):** O robô segue a trajetória original, executando as ações de controlo pré-definidas do ficheiro de dados.
2. **Deteção e Desvio (Laranja):** Ao detetar um risco de colisão, o sistema ativa o **controlador de seguimento PI** para seguir o arco de segurança (`avoid_ref`). Esta fase é focada na precisão do seguimento do caminho.
3. **Alinhamento (Magenta):** Assim que o caminho para o ponto de reencontro está livre, o sistema ativa o **controlador de Kanayama**. O seu objetivo é guiar o robô de forma suave e eficiente até à pose de reencontro, garantindo que chega com a posição e o ângulo corretos para retomar a missão.
4. **Retoma:** Ao atingir a pose de reencontro, o robô volta ao modo normal, continuando a execução das ações da trajetória original.

O resultado final demonstra o sucesso da arquitetura implementada. O sistema de decisão não só deteta corretamente o risco, mas também orquestra de forma inteligente a transição entre diferentes estratégias de controlo para executar uma manobra de correção que é, simultaneamente, segura e eficiente.

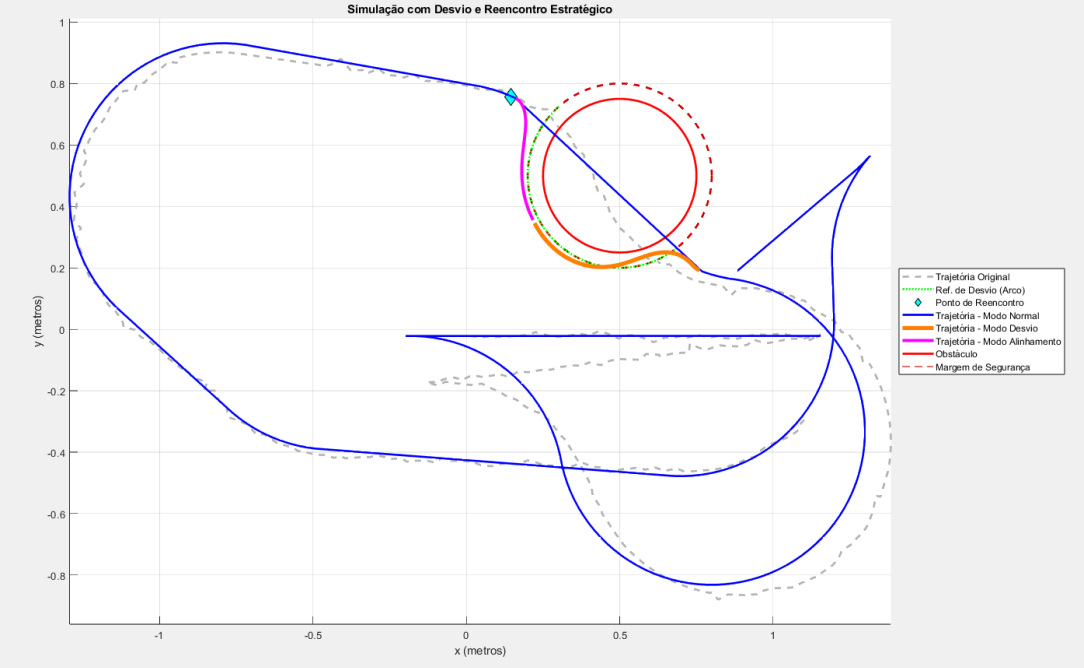


Figure 12: Resultado da simulação final do sistema de decisão. A trajetória do robô é mostrada com cores que indicam o modo de operação: Normal (azul), Desvio (laranja) e Alinhamento (magenta), em relação à trajetória original (cinzento tracejado).

7 Conclusão

Este projeto culminou no desenvolvimento e validação de um sistema de decisão robusto para um robô móvel, cumprindo os objetivos propostos e, mais importante, gerando conclusões significativas sobre a aplicação de diferentes estratégias de controlo.

Da primeira etapa, focada na deteção de falhas, conclui-se que a monitorização contínua da correspondência entre os comandos de controlo e o movimento real do robô é uma abordagem direta e eficaz para garantir a integridade do sistema. A implementação desta camada de verificação demonstrou ser um pré-requisito de segurança essencial para a operação fiável de qualquer sistema autónomo.

A segunda fase do projeto, centrada no controlo de desvio, proporcionou a conclusão técnica mais importante deste trabalho, relacionada com a seleção hierárquica de controladores. Foi verificado que, para a tarefa de seguir os pontos de uma trajetória de desvio, um controlador PID é marcadamente superior a um controlador Proporcional simples, oferecendo uma resposta mais estável e precisa ao anular erros de estado estacionário e ao amortecer oscilações. No entanto, para a tarefa mais exigente de alinhar o robô com uma pose final específica — o ponto de reencontro —, mesmo o controlador PID se mostrou limitado. A transição para o controlador de Kanayama, projetado para robôs com restrições de movimento (não-holonómicos), provou ser a solução ótima para este desafio específico. Conclui-se, assim, que não existe um "melhor" controlador universal, mas sim uma adequação de cada um a diferentes níveis de complexidade da tarefa: o Proporcional para tarefas básicas, o PID para um seguimento de trajetória robusto, e controladores baseados em modelo, como o de Kanayama, para manobras de precisão de pose.

Finalmente, a validação do sistema integrado (terceira pergunta) permite concluir sobre a eficácia da sua arquitetura modular e híbrida. O sistema demonstrou com sucesso a capacidade de orquestrar diferentes módulos e controladores: usou as ações pré-gravadas no modo normal, mudou para um controlo PID para seguir o arco de desvio e, por fim, ativou o controlo de Kanayama para a manobra de precisão final.A principal conclusão desta fase é que a combinação de diferentes estratégias de controlo, ativadas por uma máquina de estados bem definida, permite que o sistema como um todo exiba um comportamento emergente que é simultaneamente inteligente, seguro e eficiente.

Em suma, este projeto demonstra na prática que a construção de um sistema robótico autónomo fiável assenta em três pilares: a monitorização constante do seu estado, a aplicação criteriosa de uma hierarquia de algoritmos de controlo adequados à complexidade de cada subtarefa, e uma arquitetura de software modular capaz de integrar estas capacidades de forma coesa.

References

[1] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Nogai, "A stable tracking control method for an autonomous mobile robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1990, pp. 384–389.

[2] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB®*, 2nd ed. Springer, 2017.

[3] W. J. Rugh, *Linear system theory*. Prentice-Hall, Inc., 1996.

[4] D. Silvestre, *Decision Systems (SD) Lecture Slides*. Course Material, NOVA School of Science & Technology, 2024/2025.