DOI: 10.48011/asba.v2i1.1040

Sistema de Alertas e Operação Assistida de um Robô para a Inspeção de Ambientes Confinados - EspeleoRobô ⋆

Israel Amaral * **Carolina Fany** * **David Simon** * **Lucas Matos** *
Álvaro Araújo * **Lucca Leão** * **Adriano Rezende** *
Héctor Azpúrua †,‡ **Gustavo Pessin** ‡ **Gustavo Freitas** *

* *Escola de Engenharia, Universidade Federal de Minas Gerais*
† *Dept. Ciência da Computação, Universidade Federal de Minas Gerais*
‡ *Instituto Tecnológico Vale*
(israelfilipe,carol-duarte,davidsimon,lucasvini,ara1557,luccaleao98, gustavomfreitas)@ufmg.br; (hector.azpurua,gustavo.pessin)@itv.org

Abstract: The EspeleoRobô is a teleoperated mobile robotic platform, employed by Vale S.A. to explore confined environments. During inspection operations, the robot deals with adversities including risk of collision, rollover and loss of connection with the control base. This article proposes new alert and assisted operation systems for the EspeleoRobô, seeking to facilitate the remote operation of the device. To this end, five algorithms are implemented for (i) Collision Risk Alert, (ii) Rollover Risk Alert, (iii) Connection Loss Alert, (iv) Autonomous Local Navigation for Radio Signal Restoration and (v) Assisted Navigation in Galleries. The article describes these algorithms, as well as the validations performed through simulation and laboratory and field experiments. The results obtained show that the functionalities developed can contribute effectively to the robot operational safety.

Resumo: O EspeleoRobô é uma plataforma robótica móvel teleoperada, utilizada pela Vale para a exploração de ambientes confinados. Durante operações de inspeções, é comum o robô se deparar com adversidades incluindo risco de colisão, capotamento e perda da conexão com a base de controle. Este artigo propõe novos sistemas de alerta e operação assistida do EspeleoRobô, buscando facilitar a operação remota do dispositivo. Para tal, são implementados cinco algoritmos para (i) Alerta de Risco de Colisão, (ii) Alerta de Risco Capotamento, (iii) Alerta de Perda de Conexão com a base de controle, (iv) Navegação Autônoma Local para Restabelecimento de Conexão e (v) Navegação Assistida em Galerias. O artigo descreve os algoritmos, assim como as validações realizadas através de simulação e experimentos em laboratório e campo. Os resultados obtidos mostram que as funcionalidades desenvolvidas podem contribuir de forma efetiva para a segurança operacional do robô.

Keywords: Mobile Robots; Assisted Operation; Alert System; Service Robots.
Palavras-chaves: Robôs Móveis; Operação Assistida; Sistema de Alertas; Robôs de Serviço.

1. INTRODUÇÃO

A exploração e inspeção de ambientes confinados representam riscos aos operadores envolvidos. Muitas estruturas de difícil acesso, incluindo sistemas de tubulações, galerias de drenos e barragens, moinhos e silos requerem manutenção constante, aumentando a exposição destes operadores a condições adversas.

Outro exemplo são as cavernas naturais, comumente encontradas em regiões de formação de ferro e áreas de mineração, que devem ser previamente investigadas antes de realizar qualquer atividade de exploração nas proximidades. Cavidades naturais podem apresentar vários riscos, como a presença de animais peçonhentos, gases nocivos, excrementos de morcegos e o risco de colapso estrutural.

Uma possível solução consiste na utilização de um dispositivo robótico móvel capaz de investigar estes ambientes de forma remota, evitando a exposição dos operadores a riscos. Nesse contexto, o Instituto Tecnológico Vale (ITV), em conjunto com a Universidade Federal de Minas Gerais (UFMG), vem desenvolvendo o dispositivo robótico denominado de EspeleoRobô. O robô teleoperado foi originalmente desenvolvido para a inspeção de cavernas naturais, e agora está sendo utilizado para monitorar galerias de barragens e drenos, e moinhos de bolas.

O projeto EspeleoRobô teve início em 2015, e desde então a plataforma robótica sofreu mudanças no projeto mecânico

⋆ Este trabalho foi parcialmente financiado pela Vale S.A. e Instituto Tecnológico Vale (ITV), Universidade Federal de Minas Gerais (UFMG), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Código de Financiamento 001. O trabalho contou com a participação de membros do PETEE-UFMG.

e instrumentação embarcada (Cota et al., 2017; Azpurua et al., 2019). A Figura 1 ilustra a evolução do robô.

Figura 1. Diferentes modelos do EspeleoRobô.

Operar uma plataforma robótica em campo de forma remota exige constante consciência situacional, além de módulos de percepção eficazes e mecanismos flexíveis para o planejamento de movimento e controle (Marion et al., 2017). É importante destacar que mesmo com bons módulos de percepção, é um desafio compreender o ambiente de operação através da percepção restrita de um robô. A teleoperação do dispositivo assume que o operador possa superar esses problemas cognitivos, geralmente com a ajuda de interfaces de usuário bem projetadas (Murphy et al., 2016). Sendo assim, o EspeleoRobô conta com uma GUI (*Graphical User Interface*) intuitiva para o operador comandar o dispositivo.

Essa GUI conta com alertas visuais que indicam ao usuário sobre riscos envolvidos na operação; os alertas são definidos conforme a gravidade da situação (Galindo et al., 2006). A GUI também possui botões para habilitar e desabilitar funções de operação assistida, buscando facilitar a exploração remota de ambientes confinados.

Em (Beer et al., 2014) são propostos dez graus de autonomia para um robô: tele operação manual; suporte de ação; tele operação assistida; processamento em lote; suporte de decisão; estratégias de controle: compartilhado com iniciativa humana, compartilhado com iniciativa do robô, de supervisão, executivo; e autonomia completa. No caso da operação assistida, o usuário assiste todos os aspectos da tarefa; contudo, o robô é responsável pelo sensoriamento do ambiente e intervém em alguns casos.

Buscando aumentar o grau de autonomia do EspeleoRobô, estão sendo implementados algoritmos através de pacotes de ROS (*Robot Operating System*) para assistir à operação remota do dispositivo. Visando ajudar o operador a evitar situações que possam comprometer o funcionamento do robô, foram criados pacotes para avaliar riscos de colisão e capotamento que geram alertas visuais na interface gráfica de usuário. Também foram desenvolvidos algoritmos para mitigar o problema de perda de conexão do robô com a base de controle, indicando a qualidade da conexão através de um alerta visual na GUI, e em casos críticos controlando o robô de forma autônoma até o restabelecimento da comunicação. Além disso, um pacote capaz de auxiliar na navegação do robô em galerias está sendo implementado, considerando as operações de inspeção em campo realizadas rotineiramente pelo robô.

O restante do artigo está organizado na seguinte forma: na Seção 2 é apresentada uma breve descrição dos dispositivos robóticos utilizados. Já a Seção 3 descreve os algoritmos implementados através de pacotes de ROS. A Seção 4 apresenta simulações e experimentos realizados em laboratório e em campo para a validação das novas funcionalidades. Por fim, a Seção 5 destaca as conclusões e propõe trabalhos futuros.

2. PLATAFORMAS ROBÓTICAS UTILIZADAS

As novas funcionalidades desenvolvidas para a operação assistida foram validadas através de simulações, bem como experimentos com robôs reais. A Figura 2 apresenta os robôs utilizados nos experimentos: (a) modelo simulado do EspeleoRobô no CoppeliaSim; (b) EspeleoRobô; (c) Robô auxiliar da UFMG, composto por uma plataforma móvel Pioneer 3-AT mais instrumentação embarcada equivalente.

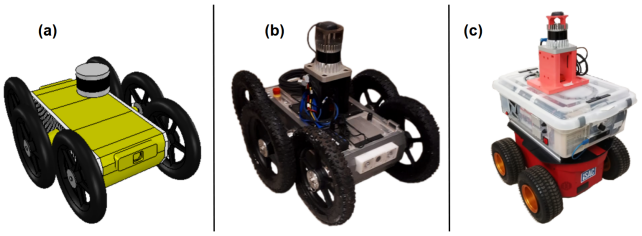

Figura 2. Robôs utilizados para validação dos experimentos: (a) EspeleoRobô simulado, (b) EspeleoRobô real, (c) Robô auxiliar implementado na UFMG.

O EspeleoRobô foi projetado com dimensões reduzidas (0,55x0,25x0,14m), para possibilitar sua mobilidade em ambientes estreitos. Além disso, seu peso é de aproximadamente 28kg, com capacidade de carga útil de 10kg extras.

Capaz de se adaptar às condições do terreno, o robô pode se mover com diferentes mecanismos de locomoção, que são facilmente substituídos através de pinos de engate rápido. As diferentes configurações incluem pernas como as do robô multitarefa da *Boston Dynamics* RHEX (Saranli et al., 2001), além de rodas e esteiras.

O robô é equipado com: seis conjuntos de engrenagens de redução planetária e motores MCD EPOS da *Maxon Motors*, baterias militares de alta-densidade da *Ben-Tronics*, um par de câmeras HD Axis-P12 e um computador mini PC Intel Nuc Core I5 rodando ROS com Ubuntu 16.04.

Para estimar a pose do robô e mapear o ambiente ao redor, são utilizados um laser OusterOS1-16, câmeras Intel RealSense Depth e Tracking, e uma IMU *(Inertial Measurement Unit)* Xsens MTi-G-710. A comunicação entre plataforma robótica e base de comando é realizada utilizando rádios AIR Rocket M900 da Ubiquiti operando na frequência de 900MHz.

3. ALGORITMOS DE ALERTAS E OPERAÇÃO ASSISTIDA

A seguir são apresentados os algoritmos desenvolvidos para a operação assistida do EspeleoRobô. Entre os pacotes de ROS implementados, estão implementadas as seguintes funcionalidades: alerta de risco de colisão, alerta de risco de capotamento, alerta de risco de perda de conexão, navegação local para restabelecimento de conexão e navegação assistida em galerias.

A teleoperação do robô é feita com o auxílio de uma interface gráfica. As funcionalidades descritas nesta seção foram integradas a essa GUI, permitindo que o operador visualize os alertas emitidos e possa habilitar as funcionalidades de operação assistida. A Figura 3 mostra a GUI durante

uma inspeção de galerias do EspeleoRobô. O retângulo vermelho destaca os alertas visuais e botões para habilitar ou desabilitar os pacotes de operação assistida.

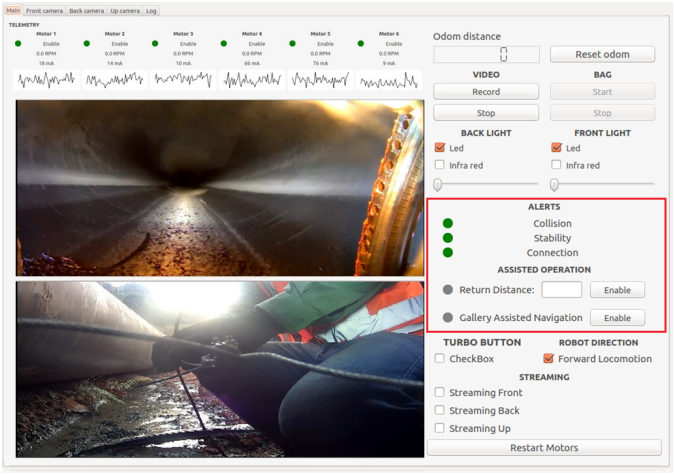

Figura 3. Interface gráfica de usuário do EspeleoRobô.

3.1 Alerta de Risco de Colisão

O algoritmo de alerta de risco de colisão é responsável por processar uma nuvem de pontos e detectar obstáculos que representem ameaça ao EspeleoRobô, gerando mensagens de alerta alterando a cor do LED *collision* presente na GUI conforme o risco durante a operação. A Figura 4 ilustra o funcionamento do algoritmo.

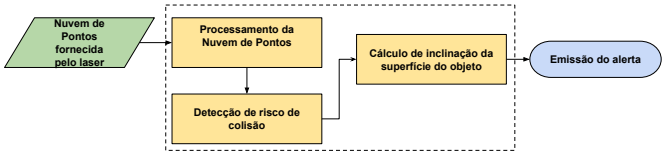

Figura 4. Representação gráfica do pacote para alerta de risco de colisão.

Uma nuvem de pontos do ambiente é gerada pelo laser OusterOS1-16, com campo de visão de 360° horizontal e 33.2° vertical. Os dados da nuvem são processados pelo ROS e representados por $\mathbf{p}_i \in \mathbb{R}^3$, contendo as coordenadas cartesianas de cada ponto i medido pelo laser. Dessa forma é possível calcular a distância d_{3D} de um obstáculo qualquer no ambiente em relação ao robô utilizando a formula da distância euclidiana (linha 3 do Algoritmo 1) e, caso esse obstáculo esteja muito próximo do robô, o ponto é classificado com possível risco de colisão.

Para filtrar corretamente os obstáculos que representam risco de colisão, o algoritmo calcula o ângulo de inclinação entre pontos da superfície deste obstáculo. Esta estratégia é utilizada para evitar que estruturas como pisos ou terrenos com pequenas inclinações não sejam classificadas como obstáculos que representem risco de colisão. Esse cálculo é realizado considerando as distâncias d_{2D} no plano (linha 5 do Algoritmo 1) e as alturas $p_{i,z}$ entre pontos consecutivos classificados como risco de colisão. Com esses dados, é aplicada a fórmula do arco tangente (linha 6) para calcular o angulo ϑ entre os pares de pontos.

A classificação de um obstáculo como risco de colisão é feita de acordo com as linhas 7 do Algoritmo 1 onde, se $\vartheta > \vartheta_{limit}$ e $d_{2D_i} < d_{2D_{limit}}$, o algoritmo classifica o obstáculo como risco de colisão. Dessa forma, o LED *collision* presente na GUI altera sua cor para amarelo ou vermelho, de acordo a distância d_{2D} entre o obstáculo e o robô.

Cabe ressaltar que os valores limites de $d_{3D_{limit}}$, $d_{2D_{limit}}$ e ϑ_{limit} utilizados para a classificação de obstáculos que representam risco de colisão são configuráveis em um arquivo de parâmetros de formato *".yaml"* presente no pacote contendo o algoritmo. Assim como nos outros algoritmos apresentados, são utilizados dois valores limites (nesse caso $d_{2D_{limit}}$) para definir o nível de alerta emitido.

Entradas: $pointCloud, d3D_{limit}, \vartheta_{limit}$
Saídas: $collisionFlag$
1: $collisionFlag \leftarrow FALSE$
2: **for** $p_i \in pointCloud$ **do**
3: $\quad d3D_i = \sqrt{(p_{i,x})^2 + (p_{i,y})^2 + (p_{i,z})^2}$
4: $\quad$ **if** $d3D_i \leq d3D_{limit}$ *and* $i > 0$ **then**
5: $\quad\quad d2D_i = \sqrt{(p_{i,x})^2 + (p_{i,y})^2}$
6: $\quad\quad \vartheta = atan2\left(\frac{p_{i,z} - p_{i-1,z}}{d2D_i - d2D_{i-1}}\right)$
7: $\quad\quad$ **if** $\vartheta > \vartheta_{limit}$ *and* $d2D_i < d2D_{limit}$ **then**
8: $\quad\quad\quad collision\_flag \leftarrow TRUE$
9: $\quad\quad\quad$ **break**
10: **return** $collisionFlag$

Algoritmo 1: Alerta de risco de colisão.

3.2 Alerta de Risco de Capotamento

O algoritmo de alerta de risco de capotamento tem por objetivo avaliar a estabilidade do robô através do cálculo dos ângulos de tombamento. Tais ângulos indicam o limite máximo em que o robô pode ser rotacionado antes do tombamento. Com base nesses ângulos, o objetivo final é gerar uma mensagem de alerta para situações de iminência de capotamento. A Figura 5 apresenta o funcionamento do algoritmo.

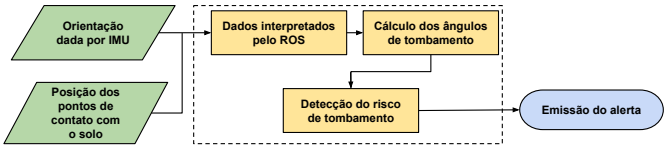

Figura 5. Representação gráfica do pacote para alerta de risco de capotamento.

Inicialmente, é necessário obter a localização dos pontos de contato $\mathbf{p}^R_{C(i)}$ do robô com o terreno, com respeito ao sistema de coordenadas do robô que é coincidente com seu Centro de Massa (CM). Os contatos estão localizados nas partes inferiores das rodas que tocam o terreno.

Esses pontos devem ser recalculados com respeito a um sistema de coordenadas inercial. Para isso, é utilizada a orientação disponibilizado pela IMU do robô (linha 1 do Algoritmo 2). Os novos pontos representados por $\mathbf{p}^I_{Ci}$ são computados utilizando a matriz de rotação $\mathbf{R} \in SO(3)$, conforme linha 2.

A partir do invólucro convexo do polígono formado pelos m pontos $\mathbf{p}^R_{C(i)}$, é definido o polígono de sustentação do robô (Papadopoulos and Rey, 1996), conforme ilustrado na Figura 6.

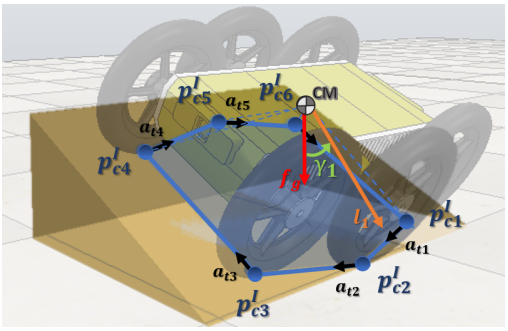

Figura 6. Polígono de sustentação do EspeleoRobô definido pelos pontos de contato ($\mathbf{p}^I_{C(i)}$). Os eixos de tombamento ($\mathbf{a}_{t(i)}$), a normal ($\mathbf{l}_{(1)}$) e o ângulo de tombamento ($\gamma_{(1)}$) são ilustrados.

As linhas que unem os pontos de contato compõem as arestas do polígono de sustentação, que correspondem aos eixos de tombamento $\mathbf{a}_{ti}$. Esses eixos são calculados subtraindo pontos $\mathbf{p}^I_{Ci}$ consecutivos (linha 5). Para determinar a distância entre os eixos de tombamento e o CM do robô, $\mathbf{p}^I_{C(i+1)}$ são projetados em $\mathbf{a}_{t(i)}$, definindo os vetores $\mathbf{l}_{(i)}$ normais à $\mathbf{a}_{t(i)}$ (linha 7). Cada eixo $\mathbf{a}_{t(i)}$ está associado a um ângulo de tombamento $\gamma_{(i)}$, que representa a rotação entre o vetor da força gravitacional $\mathbf{f}_g$ e a normal $\mathbf{l}_{(i)}$ (linha 13). Por fim, o menor dos ângulos de tombamento é definido como α; quanto mais próximo de zero for o seu valor, mais iminente o robô está do capotamento.

O alerta é ativado quando $\alpha < \alpha_{limit}$; neste caso, o LED *stability* muda de cor na GUI para amarelo ou vermelho conforme o valor de α, alertando o operador. O valor de α_{limit} é definido em um arquivo *".yaml"* de parâmetros.

Entradas: m, $\mathbf{q}$, $\mathbf{p}^R_{C(i)}$, $\mathbf{f}_g$
Saídas: γ, α, $tipOverFlag$
1: $R \leftarrow quat2rotm(\mathbf{q})$
2: $\mathbf{p}^I_{C(i)} \leftarrow \mathbf{R}\ \mathbf{p}^R_{C(i)}$
3: $\hat{\mathbf{f}}_g \leftarrow \mathbf{f}_g / norm(\mathbf{f}_g)$
4: **for** $i = 1, i = m, i = i + 1$ **do**
5: $\quad \mathbf{a}_{t(i)} \leftarrow \mathbf{p}^I_{C(i+1)} - \mathbf{p}^I_{C(i)}$
6: $\quad \hat{\mathbf{a}}_{t(i)} \leftarrow \mathbf{a}_{t(i)} / norm(\mathbf{a}_{t(i)})$
7: $\quad \mathbf{l}_{(i)} \leftarrow (\mathbb{I} - \hat{\mathbf{a}}_{t(i)} \hat{\mathbf{a}}^T_{t(i)}) \mathbf{p}^I_{C(i+1)}$
8: $\quad \hat{\mathbf{l}}_{(i)} \leftarrow \mathbf{l}_{(i)} / norm(\mathbf{l}_{(i)})$
9: $\quad$ **if** $(\hat{\mathbf{l}}_{(i)} \times \hat{\mathbf{f}}_g) \cdot \mathbf{a}_{t(i)} > 0$ **then**
10: $\quad\quad \sigma \leftarrow 1$
11: $\quad$ **else**
12: $\quad\quad \sigma \leftarrow -1$
13: $\quad \gamma_{(i)} \leftarrow \sigma \cos^{-1}(\hat{\mathbf{f}}_g \cdot \hat{\mathbf{l}}_{(i)})$
14: $\alpha \leftarrow min(\gamma)$
15: **if** $\alpha \leq \alpha_{limit}$ **then**
16: $\quad tipOverFlag \leftarrow TRUE$
17: **else**
18: $\quad tipOverFlag \leftarrow FALSE$
19: **return** $\gamma, \alpha, tipOverFlag$

Algoritmo 2: Alerta de risco de capotamento.

3.3 Alerta de Risco de Perda de Conexão

O objetivo deste algoritmo é emitir um alerta quando a conexão entre os rádios do EspeleoRobô e da base de controle apresentar falhas. A Figura 7 ilustra o funcionamento do algoritmo.

Um aspecto que dá grande importância a este algoritmo é o fato de que as cavidades as quais o robô deve explorar não

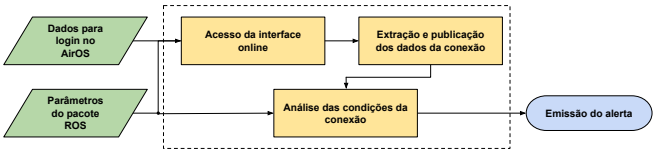

Figura 7. Representação gráfica do pacote para alerta de risco de perda de conexão.

tem um formato regular, o que torna difícil prever como ocorrerá a propagação das ondas de rádio dentro delas. Além disso, o fluxo de dados entre o dispositivo robótico e a base é consideravelmente grande, devido a todos os sensores instalados no robô, por exemplo as câmeras que transmitem informações de vídeo continuamente.

A fabricante dos rádios disponibiliza uma interface de monitoramento online, o AirOS, utilizada para checar os *status* de *quality*, *signal* e *noisef* da conexão entre base e robô. O status *quality* é uma medida da qualidade geral da conexão, dada pela porcentagem da banda total que está sendo utilizada para transmissão de dados. Já *signal* e *noisef* representam, respectivamente, os valores de sinal e ruído da conexão em dBi; caso a diferença entre eles seja pequena, significa que o ruído está causando muita interferência na transmissão de dados.

O AirOS é acessado utilizando o IP (*Internet Protocol*) do rádio, conforme implementado na linha 2 do Algoritmo 3. Depois disso são acessados os valores avaliados (linha 3).

Na linha 4 uma variável $diff$ recebe a diferença entre *signal* e *noisef*. Na linha 5 são analisados o valor de *quality* e $diff$, e caso estes estejam abaixo de valores limite ($quality_{limit}$ e $noise_{limit}$), é gerada uma mensagem de alerta. A cor do LED *Connection* na GUI muda conforme o valor de *quality* e $diff$, para notificar o operador. Os valores limite são configuráveis, e estão definidos em um arquivo *".yaml"* de parâmetros presente no pacote.

Entradas: $radioIP$, $credentials$, $quality_{limit}$, $noise_{limit}$
Saídas: $connectionFlag$
1: $connectionFlag \leftarrow FALSE$
2: $webDocument \leftarrow getURL(radioIP, credentials)$
3: $(quality, signal, noise) \leftarrow parse(webDocument)$
4: $diff \leftarrow (signal - noise)$
5: **if** $quality \leq quality_{limit}$ *or* $diff \leq noise_{limit}$ **then**
6: $\quad connectionFlag \leftarrow TRUE$
7: **return** $connectionFlag$

Algoritmo 3: Alerta de risco de perda de conexão.

3.4 Navegação Autônoma Local para Restabelecimento de Conexão

O objetivo do algoritmo de navegação autônoma local para restabelecimento de conexão é fazer com que, quando houver queda de conexão entre rádios da base de controle e dispositivo móvel, o robô seja capaz de retroceder seu percurso de modo autônomo até o reestabelecimento da conexão. A Figura 8 resume o funcionamento do pacote de navegação autônoma local.

É importante esclarecer que o caminho percorrido para o reestabelecimento da conexão deve ser o mesmo por onde o EspeleoRobô já passou. Isto porque o robô realiza inspeções em locais com terrenos não estruturados com

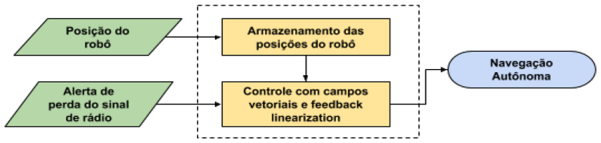

Figura 8. Representação gráfica do pacote para navegação autônoma local para restabelecimento de conexão.

risco de colisão ou tombamento. Assim, uma forma segura de navegação local é utilizar o caminho de ida como referência para o robô retroceder.

O funcionamento desse pacote ocorre em duas rotinas. A primeira é executada enquanto houver conexão entre base de comando e robô. Essa rotina é responsável por salvar o caminho percorrido pelo robô. Isso é feito armazenando a posição planar $\mathbf{p}_R$ do robô em um vetor de *buffer*. Para que a mesma coordenada não seja adicionada várias vezes ao *buffer*, é calculada a variação d_{disp} da distância entre pontos consecutivos (linha 3 do Algoritmo 4).

Caso d_{disp} seja maior que uma variação d_{limit}, as coordenadas $p_{R,x(i)}$ e $p_{R,y(i)}$ são adicionadas ao *buffer* (linha 5 do Algoritmo 4). Cabe ressaltar que o tamanho do *buffer* é fixo, especificado por um parâmetro no código; isso faz com que o robô possua uma distância máxima de retorno. Caso o *buffer* já tenha atingido o seu tamanho máximo de armazenamento, o dado mais antigo é apagado e o dado mais novo é adicionado, seguindo a lógica FIFO (*first in, first out*).

Uma vez identificada a perda de conexão, o algoritmo chama a rotina de navegação autônoma. Ela consiste em enviar todos os pontos do *buffer* na forma de uma curva parametrizada utilizada como referência para guiar o robô por meio de campos vetoriais artificiais e *Feedback Linearization*, conforme mostrado no Algoritmo 5. Caso a conexão de rádio seja reestabelecida, esta rotina pode ser interrompida, e o robô retorna ao modo teleoperado.

O controle por campos vetoriais se baseia na definição de uma velocidade de referência composta por duas componentes: (i) convergente; e (ii) tangente. A componente convergente é responsável por comandar o robô até a curva de referência; já a tangente é responsável por guiar o robô ao longo do caminho. A estrutura de campo vetorial adotada é semelhante à apresentada em (Gonçalves et al., 2010), utilizando como referência a curva de nível de uma função escalar α, ou seja, $\alpha(\mathbf{p}_R) = 0$. O campo é definido por:

$$\mathbf{F}(\mathbf{p}_R) = v_d \left(k_G(\alpha) \frac{\nabla \alpha}{\|\nabla \alpha\|} + k_H(\alpha) \frac{\nabla_H \alpha}{\|\nabla \alpha\|} \right), \qquad (1)$$

onde v_d é uma velocidade de referência para o robô, $k_G(\alpha) = -(2/\pi) atan(k_f \alpha)$ com ganho k_f positivo, e $k_H = \sqrt{1 - k_G^2}$. Além disso, $\nabla \alpha$ é o gradiente de α e $\nabla_H \alpha$ é o campo Hamiltoniano de α, ou seja, o vetor $\nabla \alpha$ rotacionado em 90°.

Para esta implementação, a curva de referência é definida por uma sequência de pontos, e não pela forma implícita $\alpha(\mathbf{p}_R) = 0$. Assim, a sequência de pontos foi utilizada para aproximar uma função α conforme o método numérico apresentado em (Gonçalves et al., 2010).

Esta metodologia fornece uma velocidade de referência para o robô. Porém, a ação de comando que o robô recebe é composta de velocidade linear v e angular ω. Como se trata de um robô não holonômico, não é possível computar v e ω tal que o robô alcance as velocidades $\mathbf{F}$ definidas pelo campo vetorial.

Para lidar com esse nível de controle, foi usado o *Feedback Linearization* (Siciliano et al., 2010). Ele computa v e ω necessários para fazer com que um ponto a uma certa distância a frente do robô alcance a velocidade $\mathbf{F}$ desejada.

Um botão na GUI permite o operador habilitar ou desabilitar a navegação autônoma local. Além disso, para garantir uma maior customização do pacote, foi criado um arquivo de parâmetros *".yaml"* onde é possível alterar os parâmetros de tamanho do *buffer*, d_{limit} e $d_{tolerance}$.

Entrada: $radioSignal$, $\mathbf{p}_R$
Saídas: $trajPoints$

1: $buffer \leftarrow \{\}$
2: $autoFlag \leftarrow FALSE$
3: $d_{disp} \leftarrow \sqrt{(p_{R,x(i)} - p_{R,x(i-1)})^2 + (p_{R,y(i)} - p_{R,y(i-1)})^2}$
4: **if** $d_{disp} \geq d_{limit}$ and $autoFlag$ is FALSE **then**
5: $\quad buffer.append(\mathbf{p}_{R(i)})$
6: **if** $radioSignal$ is $FALSE$ **then**
7: $\quad trajPoints \leftarrow buffer$
8: $\quad autoFlag \leftarrow TRUE$
9: $distance \leftarrow \sqrt{\mathbf{p}_R{}^2 - trajPoints[0]^2}$
10: **if** $distance \leq d_{tolerance}$ **then**
11: $\quad autoFlag \leftarrow FALSE$
12: $\quad buffer \leftarrow \{\}$
13: **return** $trajPoints$

Algoritmo 4: Navegação autônoma local.

Entrada: $trajPoints$, $radioSignal$, $\mathbf{p}_R$
Saídas: $v_{R,x}$, $\omega_{R,z}$

1: **if** $radioSignal$ is FALSE **then**
2: $\quad p_0 \leftarrow trajPoints[0]$
3: $\quad \alpha \leftarrow aproximaAlpha(trajPoints)$
4: $\quad$ **repeat**
5: $\quad\quad (v_{u,x}, v_{u,y}) \leftarrow campoVetorial(\mathbf{p}_R, \alpha)$
6: $\quad\quad (v_{R,x}, \omega_{R,z}) \leftarrow feedbackLin(v_{u,x}, v_{u,y}, \psi, d_{feedb})$
7: $\quad$ **until** $(\mathbf{p}_R != p_0)$
8: **return** $v_{R,x}$, $\omega_{R,z}$

Algoritmo 5: Controle por campos vetoriais.

3.5 Navegação Assistida em Galerias

O algoritmo de navegação assistida em galerias tem como objetivo facilitar a navegação do EspeleoRobô evitando colisões. É possível observar que, no interior de uma galeria, a aproximação das paredes laterais aumenta o risco de colisão. Para evitar isso, uma solução consiste em manter o robô o mais próximo possível do centro da galeria. Assim, se for identificado algum desvio da linha central, a direção do robô é corrigida. A representação gráfica do funcionamento do algoritmo está apresentada na Figura 9.

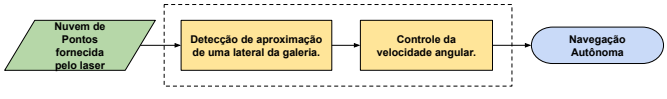

Figura 9. Representação gráfica do pacote para navegação assistida em dutos e galerias.

Na estratégia de navegação são utilizados os valores de distância medidos com um laser para identificar as laterais

da galeria; o robô se movimenta procurando manter a mesma distância entre os dois lados. O método implementado classifica as distâncias identificadas pelo laser em lado esquerdo e direito, e busca constantemente o menor valor de cada lado (linhas 1 e 2 do Algoritmo 6); esses pontos são utilizados como referências das paredes da galeria. Dessa forma, esse método independe da orientação do robô.

Os valores identificados como referências das paredes laterais são subtraídos, e o resultado é salvo numa variável $e(t)$ que representa o erro de posição do robô em relação ao centro do corredor (linha 3 do Algoritmo 6). Se $e(t) = 0$, significa que o robô está no centro do corredor; caso contrário, é definida uma velocidade angular para que o robô vire em direção ao centro.

A velocidade angular $\omega_{R,z}$ é calculada através de um controlador proporcional-integral-derivativo (PID), conforme linha 4 do Algoritmo 6. O objetivo do controle é garantir que $e(t) = 0$, ou seja, a distância entre as duas paredes permaneça igual. Os ganhos proporcional k_P, integral k_I e derivativo k_D podem ser sintonizados conforme apresentado em (Hägglund and Åström, 1995).

Um botão na GUI permite que o operador habilite ou desabilite a navegação em galerias. Os ganhos do controlador PID estão definidos num arquivo *".yaml"* de parâmetros, facilitando o ajuste dos valores de k_P, k_D e k_I.

Entradas: $laserScan$, k_P, k_I, k_D
Saídas: $\omega_{R,z}$

1: $d_{right} \leftarrow min(laserScan.ranges(-\pi/2, 0))$
2: $d_{left} \leftarrow min(laserScan.ranges(0, \pi/2))$
3: $e(t) \leftarrow d_{left} - d_{right}$
4: $\omega_{R,z} \leftarrow e(t)\ k_P + k_I \int e(t) + k_D\ \frac{de(t)}{dt}$
5: **return** $v_{R,x}, \omega_{R,z}$

Algoritmo 6: Navegação assistida em galerias.

4. SIMULAÇÕES E RESULTADOS EXPERIMENTAIS

Essa seção apresenta as simulações e experimentos realizados para validar os pacotes desenvolvidos. As simulações foram realizadas utilizando o V-REP e sua nova versão, o CoppeliaSim, integrados ao ROS. Parte das simulações foram realizadas utilizando ambientes do *Darpa Subterranean Challenge* [1], uma competição proposta pela agência de defesa americana para promover a utilização de robôs móveis na exploração de ambientes subterrâneos. Já os experimentos foram realizados utilizando o EspeleoRobô e a plataforma robótica auxiliar da UFMG. Um vídeo resumindo os experimentos está disponível em `https://www.youtube.com/watch?v=rDIOJJmsZE4`

• Alerta de risco de colisão:

As simulações e experimentos realizados para verificação do algoritmo de alerta de risco de colisão consistiram em controlar o robô num ambiente com obstáculos, e monitorar os alertas visuais gerados através do LED *collision*. Os parâmetros de risco de colisão foram definidos na simulação como $\vartheta_{limit} = 30°$, 0.80 m para alerta amarelo, e 0.40 m para alerta vermelho. Já no experimento as distâncias foram de 1.2 m para alerta amarelo e 1.0 para vermelho.

Na validação do algoritmo em ambiente simulado, foram monitoradas as mensagens de alerta geradas conforme o robô se aproxima das paredes de um corredor. Na Figura 10 é possível ver os resultados da simulação, onde a medida em que o robô se aproxima de uma das paredes, o LED altera sua cor.

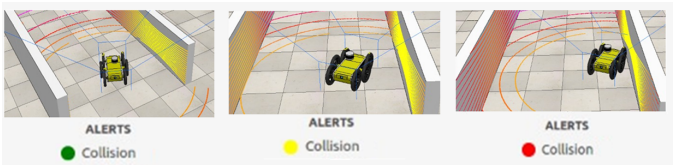

Figura 10. Validação por simulação do algoritmo de alerta de risco de colisão.

Para a validação experimental do algoritmo, o EspeleoRobô foi controlado em um percurso fechado pelos corredores do prédio da Escola de Engenharia da UFMG. A Figura 11 apresenta os resultados obtidos, onde o LED de alerta modifica sua cor conforme o robô se aproxima de um obstáculo, indicado pelos pontos vermelhos da nuvem de pontos.

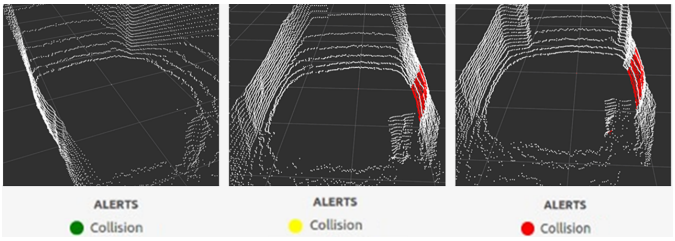

Figura 11. Validação experimental do algoritmo de alerta de risco de colisão.

• Alerta de risco de capotamento:

A verificação do algoritmo de alerta de risco de capotamento foi feita com base nos alertas visuais gerados através do LED *stability*, conforme a iminência de tombamento do robô em ambientes simulados e durante experimentos em laboratório. Os parâmetros de ângulos limite foram definidos como 25° para alerta amarelo, e 15° para alerta vermelho.

A validação em simulação foi feita em um cenário com terreno acidentado utilizado no *Darpa Subterranean Challenge*. Os resultados são demonstrados na Figura 12.

Para a verificação experimental do algoritmo, foi realizado um teste em laboratório. O objetivo foi emular o tombamento do robô de forma segura e sem riscos de danificar o equipamento. Para isso, o robô foi fixado por cintas de amarração em uma chapa de madeira e inclinado por duas pessoas em torno de cada um dos 6 eixos de tombamento, conforme ilustrado na Figura 13.

Figura 12. Validação por simulação do algoritmo de alerta de risco de capotamento.

Tanto na Figura 12 quanto na Figura 13, é possível ver que o LED alterou sua cor conforme a orientação do robô.

[1] https://www.subtchallenge.com/

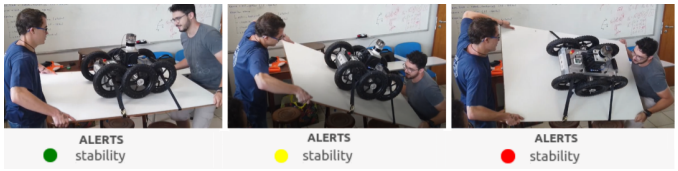

Figura 13. Validação experimental do algoritmo de alerta de risco de capotamento.

Desta forma, o operador é alertado de forma efetiva quanto ao risco de o robô tombar.

• Alerta de perda de conexão:

O algoritmo para alerta de perda de conexão foi validado em experimentos em que foram registrados os valores da intensidade do sinal de rádio, que combinados com as informações de posição do robô, possibilitam gerar gráficos de qualidade de conexão conforme ilustrado na Figura 14. Nesse teste, o robô percorreu um corredor da escola de engenharia da UFMG até que a conexão com a base fosse perdida. Os parâmetros de limite para alerta amarelo foram definidos como 40dB de *status quality* ou 40dB de diferença entre *signal* e *noisef*. Já para alerta vermelho, os parâmetros limites foram definidos como 20dB de *status quality* ou 25dB de diferença entre *signal* e *noisef*.

Na Figura 14, a posição do rádio da base de comando está marcada com um círculo vermelho. Para ilustrar a qualidade da conexão, é utilizada a convenção de que pontos onde o sinal é mais forte (*hot spots*) são marcados em vermelho, e pontos onde o sinal é mais fraco (*cold spots*) são marcados em azul (Kamakaris and Nickerson, 2005).

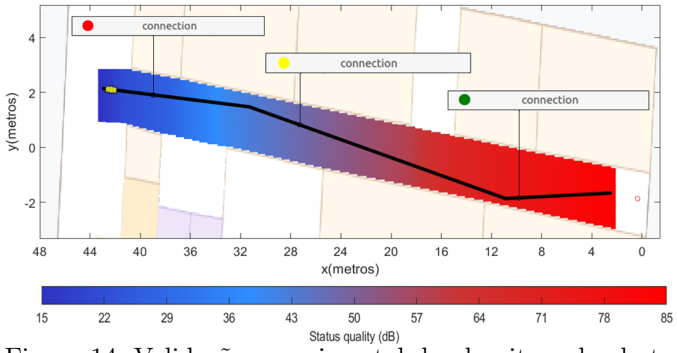

Figura 14. Validação experimental do algoritmo de alerta de risco de perda de conexão.

• Navegação autônoma local para restabelecimento de conexão

A validação do algoritmo de navegação autônoma local para restabelecimento da conexão de rádio em simulação consistiu em comandar o robô em um cenário industrial do *Darpa Subterranean Challenge*, e então simular a perda do sinal de rádio. A Figura 15 apresenta o resultado deste teste. Nela é possível ver o trecho em que o robô foi teleoperado (em azul) e o trecho que o robô navegou de forma autônoma (em vermelho).

Já a validação com o robô real ocorreu de forma análoga. Os testes foram realizados com o robô auxiliar Pioneer. O experimento consistiu em teleoperar o robô por um percurso através de obstáculos, marcados por um "X" na Figura 16, até a perda de conexão com a base de controle. A Figura 16 apresenta um gráfico contendo o percurso

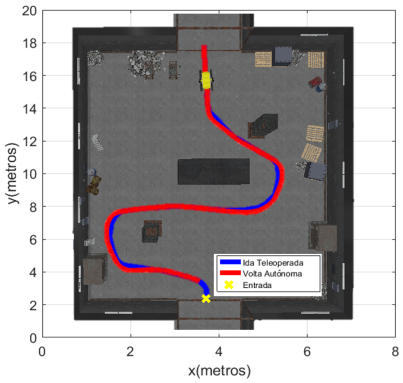

Figura 15. Validação por simulação do algoritmo de navegação autônoma local para restabelecimento de conexão.

em azul definido pelo operador por teleoperação, e em vermelho (após a perda da conexão) o caminho de volta realizado de maneira autônoma pelo robô.

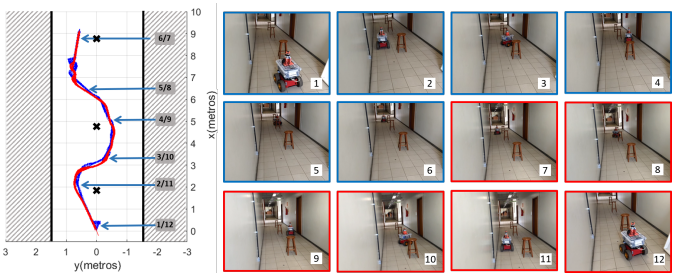

Figura 16. Validação experimental do algoritmo de navegação autônoma local para restabelecimento de conexão. O percurso azul foi realizado de forma teleoperada, e o vermelho de forma autônoma.

Analisando o gráfico da Figura 16, é possível notar que o caminho percorrido pelo robô de forma autônoma foi próximo do percurso teleoperado. As diferenças foram causadas por erros de localização, calculada utilizando apenas a odometria das rodas. Neste sentido, filtros probabilísticos e técnicas de localização e mapeamento simultâneos (SLAM) fundindo IMU, laser e câmeras já em implementação no EspeleoRobô deverão minimizar estes erros.

• Navegação assistida em galerias:

A verificação do algoritmo de navegação em galerias foi realizada em dois ambientes de simulação: uma réplica de um corredor da UFMG, e uma caverna virtual utilizada no *Darpa Subterranean Challenge*. Experimentos com o EspeleoRobô também foram realizados num corredor da UFMG, facilitando a comparação entre os resultados obtidos.

O percurso realizado pelo algoritmo no cenário de corredor criado no CoppeliaSim pode ser observado na Figura 17. O robô se manteve no centro do corredor durante todo o percurso, realizando as curvas de maneira suave e rapidamente retornando ao centro, sem a ocorrência de oscilações. Já no cenário da caverna, mesmo com o terreno bastante irregular, o robô se manteve próximo do centro da cavidade durante todo o percurso, acompanhando de forma satisfatória as curvas existentes no trajeto.

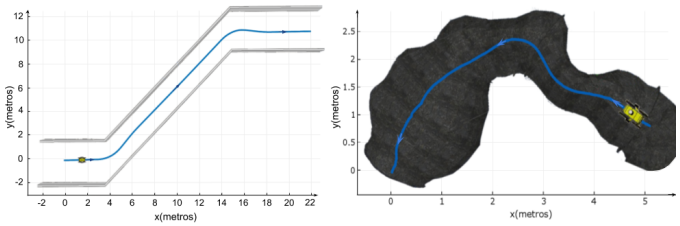

Figura 17. Validação por simulação do algoritmo de navegação assistida em galerias.

Já o percurso realizado durante experimento com o EspeleoRobô está representado na Figura 18. Apesar de utilizar os mesmos ganhos do controlador PID empregados na simulação, o caminho realizado apresenta oscilações. Isto pode ser explicado pela falta de ajustes e sintonias nos drivers de controle dos motores no dia do experimento.

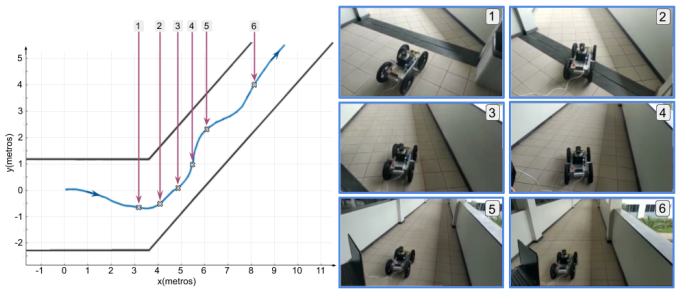

Figura 18. Validação experimental do algoritmo de navegação assistida em galerias.

5. CONCLUSÕES

Este artigo descreveu os novos sistemas de alerta e operação assistida desenvolvidos para o EspeleoRobô, buscando facilitar a operação remota do dispositivo. Os algoritmos implementados permitem identificar e alertar o operador sobre riscos de colisão, capotamento ou perda de conexão com a base de controle. Já os pacotes de navegação local para reestabelecimento de conexão e navegação assistida em galerias elevam o grau de autonomia do robô. As novas funcionalidades foram validadas através de simulações e experimentos, se mostrando promissoras ao auxiliar o operador durante operações de campo.

Transformar um dispositivo teleoperado num robô autônomo corresponde a um problema complexo a ser solucionado de maneira gradual; neste sentido, a operação assistida corresponde a uma etapa deste processo. O projeto do EspeleoRobô tem avançado nesse sentido, buscando desde a capacitação de profissionais para trabalhar com robôs (Duarte et al., 2019), até o desenvolvimento de técnicas de planejamento de caminhos ótimos (Santos et al., 2019).

Como trabalhos futuros, é possível mencionar o ajuste dos níveis de alerta de acordo com o local de operação, a complexidade da missão, e a possibilidade ou não de resgate do dispositivo de inspeção. Estes alertas deverão ser integrados aos sistemas de controle, de forma a impedir ações que possam danificar o robô. Também está prevista a integração entre os algoritmos de operação assistida e navegação com técnicas de planejamento de caminhos, visando a operação completamente autônoma do EspeleoRobô.

REFERÊNCIAS

Azpurua, H., Rocha, F., Garcia, G., Santos, A.S., Cota, E., Barros, L.G., Thiago, A.S., Pessin, G., and Freitas, G.M. (2019). EspeleoRobô - a robotic device to inspect confined environments. *ICAR - Int. Conf. on Advanced Robotics*.

Beer, J.M., Fisk, A.D., and Rogers, W.A. (2014). Toward a framework for levels of robot autonomy in human-robot interaction. *Journal of human-robot interaction*, 3(2), 74–99.

Cota, E., Torre, M.P., Rocha, F.A.S., Garcia, G., Ângelo Junior, Ramos, V., Queiroz, V., Zanini, V., Brito, G., Marques, A., Érica Pinto, Nogueira, L., Freitas, G., Miola, W., dos Reis, M.A., Araújo, R., and Brandi, I. (2017). Dispositivo de monitoramento remoto de cavidades - EspeleoRobô. *SBAI - Simpósio Brasileiro de Automação Inteligente*, 1761–1762.

Duarte, C.F.A., Amaral, I.F.S., Dutra, L.P.V., da Cruz Júnior, G.P., Azpúrua, H., and Freitas, G. (2019). Utilização do robot operating system (ROS) em conjunto com o kit didático lego mindstorms no ensino de robótica móvel. *SBAI - Simpósio Brasileiro de Automação Inteligente*.

Galindo, C., Gonzalez, J., and Fernandez-Madrigal, J.A. (2006). Control architecture for human–robot integration: application to a robotic wheelchair. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(5), 1053–1067.

Gonçalves, V.M., Pimenta, L.C., Maia, C.A., Dutra, B.C., and Pereira, G.A. (2010). Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4), 647–659.

Hägglund, T. and Åström, K.J. (1995). *PID Controllers: Theory, Design and Tuning*. International Society of Automation.

Kamakaris, T. and Nickerson, J.V. (2005). Connectivity maps: Measurements and applications. *38th Hawaii Int. Conf. on System Sciences*.

Marion, P., Fallon, M., Deits, R., Valenzuela, A., Pérez D'Arpino, C., Izatt, G., Manuelli, L., Antone, M., Dai, H., Koolen, T., et al. (2017). Director: A user interface designed for robot operation with shared autonomy. *Journal of Field Robotics*, 34(2), 262–280.

Murphy, R.R., Tadokoro, S., and Kleiner, A. (2016). Disaster robotics. In *Springer Handbook of Robotics*, 1577–1604. Springer.

Papadopoulos, E.G. and Rey, D.A. (1996). A new measure of tipover stability margin for mobile manipulators. *Proceedings of IEEE Int. Conf. on Robotics & Automation*, 4, 3111–3116.

Santos, A.S., Azpúrua, H., Pessin, G., and Freitas, G. (2019). Planejamento de caminhos para robôs móveis em ambientes acidentados. *SBAI - Simpósio Brasileiro de Automação Inteligente*.

Saranli, U., Buehler, M., and Koditschek, D.E. (2001). Rhex: A simple and highly mobile hexapod robot. *The Int. Journal of Robotics Research*, 20(7), 616–631.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2010). *Robotics: modelling, planning and control*. Springer Science & Business Media.