

Teleoperação Gestual do Dobot CR3 com Prevenção de Colisões e Singularidades

José Leandro Caires Mirante
Universidade Federal de Viçosa
Departamento de Engenharia Elétrica
Email: jose.mirante@ufv.br

Alexandre Santos Brandão
Universidade Federal de Viçosa
Departamento de Engenharia Elétrica
Email: alexandre.brandao@ufv.br

Resumo—Este trabalho apresenta um sistema de teleoperação gestual para o cobot Dobot CR3, baseado em *MediaPipe Hands*, ROS 2 Humble e MoveIt Servo, com prevenção em tempo real de duas classes de risco: (i) singularidades cinemáticas, monitoradas por meio da medida de manipulabilidade de Yoshikawa, $w = \sqrt{\det(JJ^T)}$, a 20 Hz; e (ii) colisões geométricas com objetos do ambiente, prevenidas por meio de restrições inspiradas no paradigma de *virtual fixtures*. O sistema reconhece duas mãos simultaneamente: a mão primária comanda o movimento e o *gripper*, enquanto a mão secundária controla discretamente o nível de velocidade (*slow/normal/fast*). A arquitetura contempla um caminho de simulação (Gazebo + MoveIt Servo) e um caminho para o robô real (Dobot TCP API MoveJog), compartilhando um monitor de segurança que sinaliza estados de risco. Em vez de utilizar a parada de emergência do firmware, a contenção do movimento é realizada por bloqueio e fuga direcional, permitindo a recuperação automática quando o operador conduz o robô para fora da região de risco. Os resultados experimentais avaliam a latência do pipeline ROS, a prevenção de singularidades e colisões e a robustez do reconhecimento de gestos.

Abstract—This work presents a gesture-based teleoperation system for the Dobot CR3 cobot, based on *MediaPipe Hands*, ROS 2 Humble, and MoveIt Servo, with real-time prevention of two classes of risks: (i) kinematic singularities, monitored using Yoshikawa's manipulability measure, $w = \sqrt{\det(JJ^T)}$, at 20 Hz; and (ii) geometric collisions with objects in the environment, prevented through constraints inspired by the *virtual fixtures* paradigm. The system simultaneously recognizes two hands: the primary hand commands the robot motion and the *gripper*, while the secondary hand discretely controls the speed level (*slow/normal/fast*). The architecture comprises a simulation path (Gazebo + MoveIt Servo) and a real-robot path (Dobot TCP API MoveJog), sharing a safety monitor that signals risk states. Instead of relying on the firmware emergency stop, motion containment is performed through blocking and directional escape, allowing automatic recovery when the operator commands the robot away from the risk region. Experimental results evaluate the ROS pipeline latency, singularity and collision prevention, and the robustness of gesture recognition.

Palavras-chave—Teleoperação gestual, cobot, MediaPipe, ROS 2, MoveIt Servo, manipulabilidade de Yoshikawa, virtual fixtures, prevenção de colisão, Dobot CR3.

Index Terms—Gesture teleoperation, cobot, MediaPipe, ROS 2, MoveIt Servo, Yoshikawa manipulability, virtual fixtures, collision avoidance, Dobot CR3.

I. INTRODUÇÃO

A. Contexto e motivação

A consolidação da Indústria 4.0 ampliou a presença de robôs colaborativos (cobots) em ambientes de manufatura, nos quais máquinas e operadores humanos compartilham o mesmo espaço de trabalho sem barreiras físicas. Nesse cenário, a forma convencional de comandar o manipulador, o *teach-pendant*, mostra-se pouco natural e exige treinamento específico, o que motiva o estudo de interfaces mais intuitivas. A teleoperação gestual, baseada apenas em uma câmera RGB e em reconhecimento de mãos por visão computacional [1], [2], surge como alternativa de baixo custo que dispensa sensores dedicados, luvas instrumentadas ou dispositivos hápticos, permitindo que o operador comande o robô com gestos das próprias mãos.

Além do desenvolvimento da interface de controle gestual, este trabalho considera como aspecto metodológico a validação prévia do sistema em ambiente simulado, visando facilitar sua posterior implementação no robô físico e reduzir os riscos associados aos ensaios experimentais. A realização de testes diretamente na plataforma robótica pode envolver custos operacionais, demandar maior tempo de desenvolvimento e expor o equipamento e o ambiente a riscos decorrentes de colisões ou configurações cinemáticas singulares. Nesse contexto, a utilização de um ambiente de simulação baseado em Gazebo e MoveIt [3] permite avaliar previamente os módulos de controle e segurança antes de sua implementação na plataforma física. Para que essa estratégia seja efetiva, entretanto, é desejável que a arquitetura desenvolvida permita a transferência entre os ambientes simulado e real com o mínimo de modificações nos componentes previamente validados.

Este trabalho aborda esse objetivo por meio de uma arquitetura ROS 2 *dual-path*, concebida para compartilhar os principais componentes entre os ambientes simulado e real. O módulo de reconhecimento de gestos, a interface de comunicação e os mecanismos de segurança são reutilizados em ambos os caminhos, reduzindo as modificações necessárias durante a transição entre simulação e plataforma física. Essa abordagem permite que os módulos de controle e segurança sejam previamente desenvolvidos e avaliados em simulação e, posteriormente, integrados ao robô real, preservando a lógica de teleoperação e os mecanismos de prevenção de singularidades e colisões.

B. Problema

Na teleoperação gestual, os comandos de movimento são gerados continuamente a partir dos gestos realizados pelo operador, sem que uma trajetória completa seja necessariamente planejada a priori. Embora essa característica proporcione uma interação mais direta e intuitiva, os comandos produzidos refletem essencialmente a intenção instantânea do usuário e, por si só, não incorporam informações sobre as limitações cinemáticas do manipulador ou sobre as restrições geométricas impostas pelo ambiente.

Nesse contexto, a execução direta dos comandos gestuais pode conduzir o manipulador a configurações ou regiões inadequadas de seu espaço de trabalho. Neste trabalho, são considerados dois problemas principais:

- 1) **Singularidades cinemáticas:** correspondem a configurações nas quais o Jacobiano $J(q)$ perde posto, reduzindo a capacidade do manipulador de produzir velocidades cartesianas em determinadas direções. À medida que o robô se aproxima dessas configurações, pequenas velocidades desejadas no espaço cartesiano podem exigir velocidades articulares elevadas, comprometendo a estabilidade e a execução adequada do movimento. Em um sistema de teleoperação, no qual o operador não possui necessariamente conhecimento das configurações singulares do manipulador, torna-se necessário identificar sua proximidade e restringir comandos que conduzam o sistema a essas regiões.
- 2) **Colisões com o ambiente:** os comandos derivados dos gestos do operador não consideram, por si só, a posição e a geometria dos obstáculos presentes no espaço de trabalho. Consequentemente, um comando válido do ponto de vista cinemático pode conduzir o efetuador em direção a objetos, superfícies ou outras regiões restritas do ambiente. A prevenção desse tipo de ocorrência requer que as informações geométricas da cena sejam incorporadas ao processo de execução dos comandos, permitindo restringir movimentos de aproximação antes que ocorra o contato.

Dessa forma, o problema abordado consiste em permitir que os comandos provenientes da interface gestual sejam executados de maneira contínua, preservando a capacidade de intervenção do operador, ao mesmo tempo em que são impostas restrições destinadas a evitar configurações cinemáticas singulares e colisões com o ambiente.

C. Objetivos

O objetivo deste trabalho é desenvolver e validar um sistema de teleoperação gestual para o cobot Dobot CR3, capaz de prevenir situações de singularidade cinemática e colisões com o ambiente durante sua operação, tanto em simulação quanto no robô físico.

Para alcançar o objetivo geral, busca-se desenvolver uma interface baseada no reconhecimento simultâneo de duas mãos por meio do MediaPipe Hands, permitindo ao operador comandar o movimento do manipulador e controlar diferentes

níveis de velocidade. Pretende-se também implementar o monitoramento em tempo real da manipulabilidade de Yoshikawa, com frequência de operação de pelo menos 20 Hz, de modo a identificar a aproximação de configurações cinemáticas singulares. Adicionalmente, propõe-se desenvolver uma estratégia de prevenção de colisões baseada na verificação geométrica dos obstáculos presentes no espaço de trabalho, aplicável tanto ao ambiente simulado quanto à plataforma robótica real. Por fim, pretende-se avaliar experimentalmente o sistema quanto à latência do pipeline ROS, à capacidade de prevenção de singularidades e colisões e à robustez do reconhecimento de gestos.

D. Contribuições

As principais contribuições deste trabalho estão relacionadas ao desenvolvimento de uma arquitetura de teleoperação gestual que integra mecanismos de prevenção de singularidades e colisões e que pode ser utilizada tanto em ambiente simulado quanto no robô físico. A arquitetura ROS 2 proposta adota uma abordagem *sim-to-real*, compartilhando entre os dois ambientes os principais componentes de reconhecimento de gestos, comunicação e segurança. Essa estrutura permite que o sistema seja inicialmente desenvolvido e avaliado em simulação e, posteriormente, transferido para a plataforma física com modificações reduzidas na arquitetura previamente validada.

Outra contribuição consiste na integração dos mecanismos de prevenção de singularidades cinemáticas e colisões em um mesmo sistema de teleoperação gestual. A prevenção de colisões é baseada em restrições geométricas inspiradas em *forbidden-region virtual fixtures* [4], [5], enquanto o monitoramento de singularidades utiliza a medida de manipulabilidade do manipulador. A atuação desses mecanismos restringe movimentos em direção às regiões consideradas de risco, preservando, sempre que possível, movimentos que permitam ao operador conduzir o manipulador para configurações seguras.

Por fim, o sistema desenvolvido é avaliado experimentalmente em simulação e no robô físico, considerando aspectos relacionados à latência do pipeline ROS, ao desempenho dos mecanismos de prevenção de singularidades e colisões e à robustez do reconhecimento de gestos. Essa avaliação permite verificar tanto o desempenho dos componentes individuais quanto o funcionamento integrado da arquitetura proposta em uma tarefa de teleoperação gestual.

E. Organização

O restante deste trabalho está organizado da seguinte forma. A Seção II apresenta uma revisão dos trabalhos correlatos, enquanto a Seção III aborda os fundamentos teóricos necessários ao desenvolvimento da proposta. A arquitetura do sistema é descrita na Seção IV, e a estratégia de prevenção de colisões aplicada ao robô real é detalhada na Seção V. Na sequência, a Seção VI apresenta a metodologia experimental adotada para avaliação do sistema, e a Seção VII apresenta e discute os resultados obtidos. Por fim, a Seção VIII apresenta as conclusões e as perspectivas para trabalhos futuros.

II. TRABALHOS CORRELATOS

A. Teleoperação gestual de manipuladores

A captura da intenção gestual do operador pode ser realizada por diferentes modalidades de sensoriamento, cada uma apresentando características distintas quanto à intrusividade e à robustez. Entre essas modalidades, destacam-se as luvas instrumentadas, as câmeras de profundidade, os sensores ópticos dedicados e as câmeras RGB monoculares associadas a técnicas de aprendizado de máquina.

As **luvas instrumentadas (IMU/flex sensors)** fornecem medições diretas e precisas da postura da mão, com baixa latência e imunidade a oclusões e variações de iluminação. Em contrapartida, são intrusivas, pois exigem que o operador vista o dispositivo, além de demandarem calibração específica para cada usuário [6], [7]. As **câmeras de profundidade (RGB-D)**, como o Microsoft Kinect e o Intel RealSense, permitem recuperar a geometria 3D da cena e realizar o rastreamento de mãos e do corpo sem contato. Entretanto, podem apresentar sensibilidade a oclusões e às condições de iluminação infravermelha, além de exigirem um sensor dedicado para a aquisição das informações de profundidade [8].

Outra possibilidade é o emprego de **sensores ópticos dedicados**, como o Leap Motion, que permitem realizar o rastreamento óptico-infravermelho das mãos com alta precisão. Entretanto, esses dispositivos operam em um volume de trabalho restrito, devido ao curto alcance e ao campo de visão limitado, além de exigirem hardware específico para a aquisição dos movimentos [9].

Por fim, a **câmera RGB monocular com aprendizado de máquina** utiliza imagens adquiridas por uma câmera convencional como entrada para um modelo de visão responsável por estimar os pontos-chave da mão. Entre as soluções disponíveis, destaca-se o *MediaPipe Hands* [1], [2], capaz de estimar 21 pontos-chave (*landmarks*) 2D/3D da mão em tempo real. Essa abordagem dispensa o uso de sensores específicos e apresenta baixa intrusividade, embora seu desempenho dependa da robustez do modelo diante de variações de iluminação, fundo e condições de aquisição das imagens.

O MediaPipe Hands tem sido empregado em diferentes aplicações de teleoperação de sistemas robóticos, seja no comando de braços robóticos acionados por servomotores a partir do reconhecimento das mãos em imagens monoculares [10], seja na teleoperação de múltiplos robôs por meio da integração entre MediaPipe, ROS, Gazebo e RViz [11], ou ainda na teleoperação de robôs quadrúpedes equipados com braços manipuladores [12].

No presente trabalho, o MediaPipe Hands é empregado na teleoperação de um braço robótico e integrado a mecanismos ativos de prevenção de singularidades cinemáticas e colisões com o ambiente. A abordagem é desenvolvida de forma a permitir sua aplicação tanto em simulação quanto no robô físico, preservando os principais componentes de controle e segurança entre os dois ambientes.

B. Prevenção de singularidades

As medidas escalares clássicas de proximidade de singularidade derivam todas da decomposição em valores singulares (SVD) do Jacobiano, $J(q) = U\Sigma V^T$, cujos valores singulares $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_m \geq 0$ são os semieixos do elipsoide de manipulabilidade. Uma singularidade corresponde ao colapso desse elipsoide em alguma direção, isto é, $\sigma_{\min} \rightarrow 0$. Na literatura, são empregadas três medidas principais:

- **Manipulabilidade de Yoshikawa** [13]: $w = \sqrt{\det(JJ^T)} = \prod_{i=1}^m \sigma_i$, proporcional ao volume do elipsoide. Anula-se em qualquer singularidade, é barata (apenas um determinante) e tem interpretação física direta. Permanece a referência canônica.
- **Número de condicionamento**: $\kappa = \sigma_{\max}/\sigma_{\min}$, que mede a anisotropia do elipsoide ($\kappa = 1$ isotrópico, $\kappa \rightarrow \infty$ na singularidade). É adimensional e independente de escala (vantagem sobre w), mas exige a SVD completa.
- **Menor valor singular**: $\sigma_{\min}$, que captura diretamente a direção que está colapsando, usando a SVD, sem ser mascarado pelas demais.

Este trabalho adota a manipulabilidade de Yoshikawa, por sua interpretação como volume de destreza, evitando a dependência de escala definida por calibração empírica dos limiares.

C. Prevenção de colisão e virtual fixtures

O paradigma de *virtual fixtures* foi introduzido em [4] como uma estratégia para impor restrições virtuais ao movimento durante tarefas de teleoperação, funcionando de maneira análoga a guias mecânicas que orientam ou restringem a ação do operador. Posteriormente, o emprego de *virtual fixtures* em tarefas de manipulação assistida foi formalizado em [5], incluindo restrições associadas a regiões proibidas. Na variante *forbidden-region*, o movimento em direção à região restrita é impedido, enquanto movimentos tangenciais ou de afastamento permanecem permitidos. Outra estratégia de prevenção de colisões em teleoperação é apresentada em [14], na qual a componente da velocidade comandada na direção do obstáculo mais próximo é modificada de modo a evitar a aproximação do manipulador. Em [15], são empregadas *forbidden-region virtual fixtures* construídas a partir de nuvens de pontos, permitindo estabelecer restrições de movimento com base nas informações geométricas do ambiente.

Uma abordagem distinta para a prevenção de obstáculos é apresentada em [16], por meio do método de campos potenciais artificiais (*Artificial Potential Fields* – APF). Nesse método, forças virtuais repulsivas associadas aos obstáculos são utilizadas para modificar o movimento do robô. Diferentemente dos APFs, as *virtual fixtures* do tipo *forbidden-region* atuam diretamente na restrição das componentes de movimento que conduzem o manipulador em direção às regiões proibidas.

Neste contexto, este trabalho emprega o conceito de prevenção de colisão, baseado em *forbidden-region virtual fixtures*, para a prevenção de colisões durante a teleoperação. O método utiliza primitivas geométricas para representar os obstáculos e restringe apenas as componentes do movimento que conduzem o manipulador em direção às regiões proibidas.

III. FUNDAMENTAÇÃO TEÓRICA

A. Cinemática diferencial e Jacobiano

Seja $q \in \mathbb{R}^n$ a configuração articular de um manipulador serial de $n = 6$ graus de liberdade. A relação entre velocidades articulares $\dot{q}$ e o twist cartesiano do efetuador $v \in \mathbb{R}^6$ é

$$v = J(q) \dot{q}, \quad (1)$$

em que $J(q) \in \mathbb{R}^{6 \times n}$ é o Jacobiano geométrico.

B. Medida de manipulabilidade de Yoshikawa

A medida proposta por Yoshikawa [13] resume em um único número a destreza do robô em cada configuração:

$$w(q) = \sqrt{\det(J(q) J(q)^\top)}. \quad (2)$$

Pode-se imaginar w como o volume de um elipsoide que representa a facilidade de mover o efetuador em cada direção: quanto maior w , mais o robô consegue se mover livremente. Perto de uma singularidade esse elipsoide se achata e $w(q) \rightarrow 0$, indicando que o robô perde a capacidade de mover-se em alguma direção. Monitorar w em tempo real dá, assim, um aviso contínuo de proximidade da singularidade.

Como w depende das unidades e do comprimento da cadeia modelada, não há um limiar universal: os limiares de alerta e bloqueio são calibrados empiricamente para o robô em questão (Seção IV).

C. MediaPipe Hands

MediaPipe Hands [1] é uma solução de visão computacional que localiza as mãos e estima a sua postura a partir de uma única imagem RGB, sem sensor de profundidade. O reconhecimento ocorre em duas etapas encadeadas: primeiro um detector de palma (*palm detector*) localiza a região da mão na imagem; em seguida, um modelo de landmarks estima, dentro dessa região, 21 pontos-chave anatômicos da mão (pontas dos dedos, articulações e pulso), cada um com coordenadas 2D na imagem e uma profundidade relativa estimada. O conjunto dos 21 pontos descreve completamente a configuração da mão: posição, orientação e quais dedos estão estendidos ou recolhidos (Figura 1).

A separação em duas etapas é o que viabiliza a operação em tempo real: o detector de palma, mais custoso, não precisa ser executado a cada quadro: uma vez localizada a mão, o rastreamento dos landmarks a acompanha nos quadros seguintes, e o detector só é reinvocado quando a mão é perdida [2]. Isso permite estimar a postura de até duas mãos simultaneamente a taxas de vídeo em CPU comum, sem GPU dedicada. A interpretação desses landmarks como comandos de movimento é descrita na Seção IV.

D. MoveIt Servo e Virtual Fixtures

O MoveIt Servo é o componente do MoveIt 2 [3] destinado à movimentação cartesiana contínua (*jogging*) em tempo real. Em vez de planejar uma trajetória completa, o componente recebe um fluxo contínuo de comandos de velocidade (TwistStamped) e os converte, a cada ciclo, em comandos

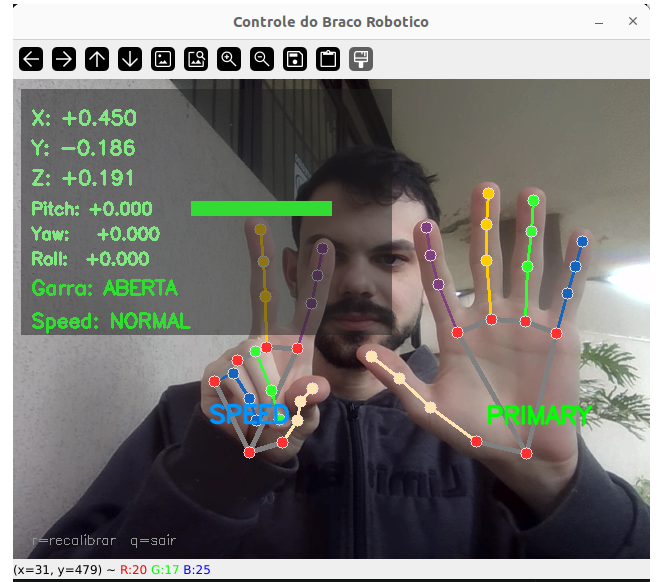


Figura 1: Janela de visão do sistema: o MediaPipe Hands estima os 21 landmarks da mão a partir de imagem RGB monocular (pontos sobrepostos), e o GestureProcessor converte o desvio em relação à pose neutra nos comandos exibidos em tela: as três translações (x, y, z), os três ângulos de orientação, o estado da garra (aberta/fechada) e o nível de velocidade.

articulares por meio da cinemática diferencial, tornando-se adequado à teleoperação, na qual o destino não é necessariamente conhecido *a priori*. O MoveIt Servo também incorpora mecanismos de proteção, como suavização dos comandos, redução automática da velocidade nas proximidades de singularidades e limites articulares, além da verificação de colisões com base na *Planning Scene*.

No caminho do robô real, entretanto, os comandos são enviados por meio do MoveJog, sem a utilização do MoveIt Servo e, consequentemente, sem acesso direto a esses mecanismos. Para incorporar restrições de movimento nesse caminho, este trabalho emprega o conceito de *virtual fixtures*, apresentado na Seção II, como base para a prevenção de colisões.

Formalmente, considerando um comando de velocidade v_c e o vetor normal $\hat{n}$ à fronteira de uma região proibida, uma *forbidden-region virtual fixture* restringe apenas a componente do movimento direcionada à região proibida, preservando as componentes tangenciais e de afastamento. Essa operação é definida por

$$v_{\text{safe}} = v_c - \max(0, \hat{n}^\top v_c) \hat{n}, \quad (3)$$

em que v_{safe} representa o comando de velocidade após a aplicação da restrição.

IV. ARQUITETURA DO SISTEMA

A. Plataforma robótica: Dobot CR3

O manipulador utilizado é o **Dobot CR3**, um robô colaborativo (cobot) serial de seis graus de liberdade, com juntas



Figura 2: O cobot Dobot CR3, manipulador serial de seis graus de liberdade utilizado neste trabalho, com a garra magnética (eletroímã) montada no flange.

rotativas, voltado a tarefas leves de manipulação em ambientes compartilhados com operadores humanos (Figura 2). Por ser um cobot, embute proteções de segurança de fábrica (como a reação a colisão por monitoramento de torque) e é acionado por uma API TCP do fabricante, usada neste trabalho no caminho do robô real. Sua arquitetura serial de seis juntas o torna sujeito a singularidades cinemáticas em parte do espaço de trabalho, o que motiva o monitor de manipulabilidade descrito adiante. O mesmo modelo do CR3 é empregado tanto na simulação (via URDF) quanto nos ensaios físicos.

A Figura 3 ilustra o pipeline: uma frente comum de reconhecimento gestual que se ramifica nos caminhos de simulação e do robô real.

B. Processador de gestos

O `GestureProcessor` é o componente que traduz os 21 landmarks da mão em um vetor de comando `/arm/cmd` de oito `float32`: $[v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \text{grripper}, \text{speed_scale}]$.

O reconhecimento é relativo a uma pose neutra: ao iniciar, o operador mantém a mão parada por alguns instantes e o sistema registra, pela média de vários quadros, uma referência de posição, tamanho aparente e orientação da palma. A partir daí, o comando é gerado pelo desvio da mão em relação a essa referência, de modo que a mão parada na pose neutra não comanda movimento. Tomando como ponto de referência a base do dedo médio, cada grau de liberdade é obtido assim:

- **Translação no plano** (v_x, v_z): deslocamento horizontal e vertical do ponto de referência em relação à pose neutra.
- **Translação em profundidade** (v_y): variação do tamanho aparente da mão: aproximar a mão da câmera a faz parecer maior, o que é interpretado como avanço.

- **Orientação** ($\omega_x, \omega_y, \omega_z$): constrói-se um sistema de coordenadas da palma a partir do pulso e da base dos dedos (indicador, médio e mindinho); o seu desvio angular em relação à orientação neutra fornece roll, pitch e yaw.
- **Gripper**: conta-se quantos dedos estão estendidos (a ponta mais distante do pulso que a base); mão aberta abre a garra, mão fechada a fecha.
- **Velocidade** (`speed_scale`): contagem de dedos da mão secundária (Seção seguinte).

Cada componente passa ainda por uma média móvel dos últimos quadros (que reduz o tremor da estimativa) e por uma normalização para $[-1, +1]$, na qual um deslocamento confortável da mão corresponde ao comando máximo. Sobre esse valor atua uma zona morta: o robô só começa a se mover quando o desvio normalizado da mão ultrapassa um limiar de 0.3, permanecendo parado abaixo dele. O desvio define a direção do movimento; a velocidade, por sua vez, é dada pelo nível discreto selecionado pela mão secundária (SLOW/NORMAL/FAST, Seção seguinte). Assim, o operador mantém o robô estável enquanto não afasta a mão claramente da pose neutra, e pequenos movimentos involuntários não geram comando.

A contagem de dedos da mão secundária é mantida por 10 quadros consecutivos antes que uma mudança no nível de velocidade seja confirmada, reduzindo transições decorrentes de oscilações no reconhecimento dos gestos. São definidos três níveis de velocidade: {SLOW:0,35, NORMAL:0,70, FAST:1,40}.

C. Monitor de segurança

O nó `safety_monitor.py` vigia continuamente a proximidade de singularidade: a cada nova leitura das juntas, calcula a manipulabilidade $w(q)$ (Equação (2)) e, comparando-a com limiares calibrados, classifica o estado do robô em NORMAL, WARNING (aproximando-se de uma singularidade) ou BLOCKED (singularidade crítica). Esse monitoramento roda a 20 Hz, frequência suficiente para reagir antes que o operador conduza o robô a uma configuração singular.

Para obter $w(q)$ é preciso o Jacobiano, que depende da cadeia cinemática do robô. O nó a constrói em tempo de execução a partir do URDF (com `urdf_parser_py`) e calcula o Jacobiano com o `PyKDL`, porque o pacote que faria essa ponte automaticamente (`kdl_parser_py`) não está disponível no ROS 2 Humble.

O nó publica quatro tópicos, consumidos pelo `collision_guard` no robô real:

- `/safety/manipulability`: o valor de w a cada ciclo;
- `/safety/computation_time`: o tempo de cálculo (ms), para aferir o custo do monitoramento;
- `/safety/status`: o estado em texto (NORMAL/WARNING/BLOCKED);
- `/safety/state_code`: o mesmo estado em código numérico, usado pelos demais nós para reagir.

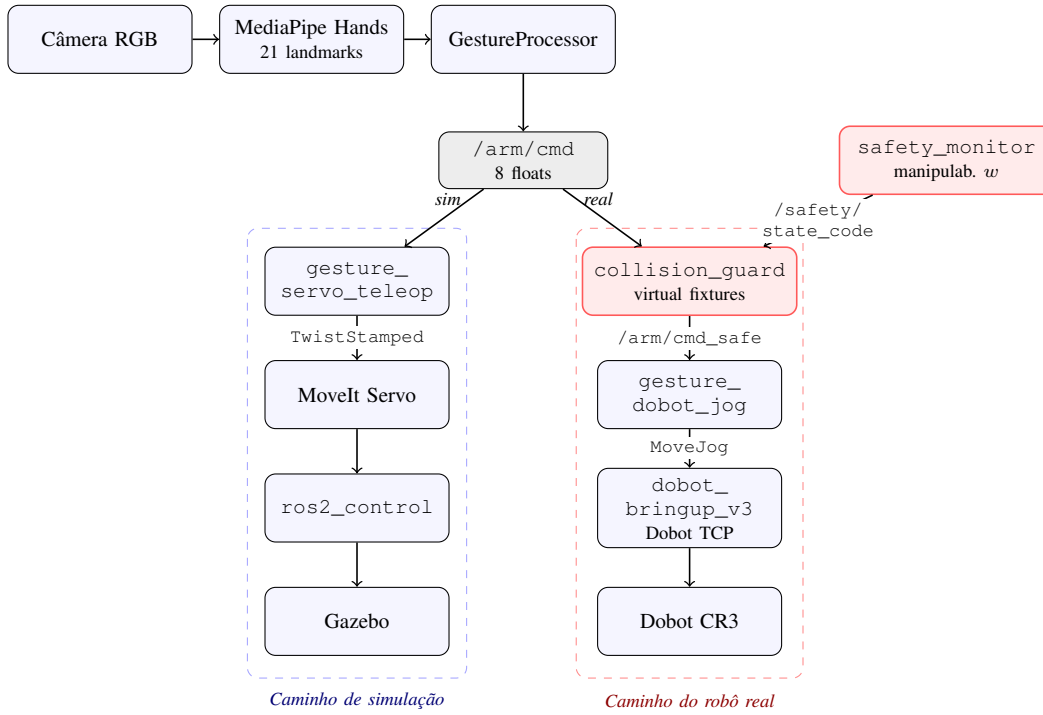


Figura 3: Pipeline dual-path. A frente comum (câmera → MediaPipe Hands → GestureProcessor → /arm/cmd) alimenta o caminho de simulação (gesture_servo_teleop → MoveIt Servo → ros2_control → Gazebo) e o caminho do robô real (collision_guard → gesture_dobot_jog → MoveJog → Dobot CR3). Os nós em vermelho (safety_monitor e collision_guard) compõem a camada de segurança proposta; o safety_monitor sinaliza singularidade via /safety/state_code.

D. Prevenção de colisão em simulação

Na simulação, os obstáculos do ambiente são descritos como primitivas geométricas (caixa, cilindro ou esfera) em um arquivo de cena: o nó scene_publisher as insere na Planning Scene do MoveIt, tornando-as visíveis no RViz. A prevenção de colisão é feita pelo mesmo collision_guard empregado no robô real (Seção V), interposto no caminho de comando (/arm/cmd → /arm/cmd_safe): ele lê a mesma descrição de cena, calcula a distância do efetuador a cada obstáculo e veta o movimento de aproximação antes do contato. Dessa forma, uma única implementação de prevenção, leve e portátil, atende tanto à simulação quanto ao robô real.

A Figura 4 mostra a cena usada nos testes, vista no RViz: o CR3 sobre o pedestal, o chão e uma parede frontal (arquivo scene_wall.yaml). Qualquer comando de gesto que levaria o efetuador a atravessar a parede é barrado pelo collision_guard antes do contato, enquanto o operador permanece livre para afastar o braço.

V. PREVENÇÃO DE COLISÃO NO ROBÔ REAL

A. Motivação

No robô real, os comandos de gesto são enviados diretamente ao robô pela API TCP do fabricante (Dobot), por meio do comando de jogging MoveJog, a forma nativa de acioná-lo. Diferentemente da simulação, o robô físico não é controlado por um controlador ros2_control, mas pelo driver

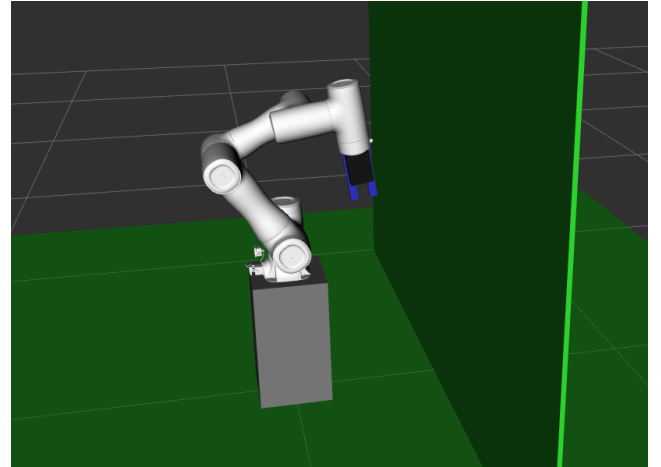


Figura 4: Cena de colisão na simulação, vista no RViz: o CR3 sobre o pedestal e as primitivas inseridas na Planning Scene pelo scene_publisher: o chão e a parede frontal. O collision_guard verifica a distância do efetuador a esses objetos e impede o robô de atravessar a parede.

do fabricante. O MoveIt Servo, por sua vez, foi projetado para comandar justamente um controlador ros2_control, interface que esse driver não expõe; conduzir o robô real através do Servo exigiria uma camada adicional de tradução,

que reintroduziria a latência da sua pilha e que, na prática, esbarrou em conflitos de comunicação com o driver oficial. Por isso o caminho real emprega o `MoveJog` diretamente.

A consequência é que esse caminho não passa pelo `MoveIt Servo` nem pelas proteções nativas que ele oferece em simulação. Resta somente a proteção do próprio firmware do robô: uma reação por torque que freia o braço depois de tocar o obstáculo, e não antes. Este trabalho reintroduz ao caminho real uma prevenção geométrica que atua antes do contato.

B. O nó `collision_guard`

A solução é um nó intermediário, o `collision_guard`, inserido entre os gestos e o robô: ele intercepta o comando de velocidade em `/arm/cmd` e reemite uma versão filtrada em `/arm/cmd_safe`, que então segue para o robô. O nó reaproveita o mesmo arquivo de descrição da cena de colisão e o leitor já usados na simulação, evitando duplicação. A posição da garra é obtida em tempo real pela árvore de transformadas (TF) do ROS, a 30–50 Hz. Como referência espacial adota-se a base do robô, e não a raiz do modelo: usar esta última criaria árvores de TF desconectadas, quebrando a consulta da posição.

C. Alinhamento da cena com a base do robô

O robô é montado sobre um pedestal, de modo que sua base fica 36 cm acima do chão. Como o `collision_guard` calcula as distâncias a partir da base, enquanto o arquivo de cena pode descrever os obstáculos em relação ao chão, essa diferença de altura precisa ser compensada: do contrário, um obstáculo no chão seria tratado como se estivesse 36 cm acima, deslocando toda a verificação.

Para que o mesmo arquivo de cena sirva tanto à simulação quanto ao robô real, o `collision_guard` converte, ao iniciar, cada obstáculo para o referencial da base. Como os obstáculos são estáticos, isso é feito uma única vez; até a conversão terminar, o nó permanece em modo seguro, bloqueando qualquer movimento.

D. Checagem geométrica

O efetuador é aproximado por uma esfera envolvente (bounding sphere) centrada na garra. No robô montado em bancada, o raio dessa esfera é de 0.13 m (cobrindo a ponta do gripper, a 13 cm do flange) e a margem de segurança é de 0.03 m. Para cada obstáculo P_i da cena:

- 1) Calcula-se a distância mínima d_i entre a esfera e P_i .
- 2) Se $d_i < d_{\text{margin}} \approx 3$ cm, computa-se o vetor normal $\hat{n}_i$ da fronteira mais próxima.
- 3) O componente translacional do twist é vetado conforme a Eq. (3).

E. Fuga direcional em singularidade

O `collision_guard` também responde ao monitor de singularidade: ao receber o estado `BLOCKED`, em vez de zerar todos os comandos (o que prenderia o operador na própria singularidade, bloqueando inclusive o gesto de fuga), adota-se uma *fuga direcional*. Ao entrar no estado, o nó memoriza

a última direção de movimento efetiva (a “direção proibida”, que conduziu à singularidade) e veta apenas a componente do comando alinhada com ela, aplicando a mesma operação da Eq. (3) com o vetor normal $\hat{n}$ substituído pela direção proibida. Isso preserva o sentido reverso e as direções ortogonais, de modo que o operador sai da singularidade gesticulando no sentido oposto. Esse veto é a única prevenção de singularidade no caminho real, já que o `MoveJog` não dispõe da parada automática do `MoveIt Servo`. O estado da garra e o nível de velocidade são sempre preservados.

F. Acionamento da garra e tarefa de *pick-and-place*

No robô real, a garra não é acionada pela API de movimento do braço, mas por um atuador independente controlado por uma Raspberry Pi. O efetuador empregado nos ensaios é um eletroímã, comutado por um relé: a Raspberry Pi executa um pequeno nó ROS que se inscreve em um tópico dedicado e, ao receber a mensagem de texto “`fecha`”, aciona o relé que energiza o eletroímã (atraindo a peça ferromagnética); a mensagem “`abre`” desliga o relé e libera a peça. O gesto de fechar a mão, já reconhecido pelo `GestureProcessor`, é o que dispara a publicação dessa mensagem, de modo que o operador continua comandando a preensão exatamente como na simulação.

A consequência de projeto mais relevante é a facilidade de integração de outros tipos de garra. Como o braço se comunica com o efetuador apenas por uma mensagem de texto (“`abre`”/“`fecha`”) em um tópico ROS, trocar o eletroímã por uma garra pneumática, de dois dedos ou de sucção exige apenas reescrever o script que roda na Raspberry Pi para traduzir essa mensagem no acionamento físico correspondente: nada muda no módulo de gestos, na prevenção de singularidade ou na de colisão. O acionamento fica, assim, totalmente desacoplado do controle cartesiano do braço e preserva a arquitetura dual-path: o mesmo gesto que controla a garra em simulação comanda o atuador real sem alterar uma linha do pipeline de visão.

Esse acionamento habilitou a tarefa de validação ponta a ponta do sistema, uma operação de *pick-and-place* executada por teleoperação gestual: conduzir o efetuador até uma caixa por gestos, fechar a garra (energizar o eletroímã) para agarrá-la, transportá-la até uma posição-alvo e reabrir a garra (desenergizar) para soltá-la, com a prevenção de singularidade e de colisão ativas durante todo o percurso. A demonstração foi realizada com sucesso no robô físico (Seção VII), comprovando que o controle gestual, o acionamento da garra e as duas camadas de prevenção operam de forma integrada em uma tarefa de manipulação completa.

VI. METODOLOGIA EXPERIMENTAL

Para avaliar o sistema foram planejados os experimentos descritos a seguir. Cada um define o que se mede e como o resultado é julgado.

A. Exp. A: Latência do pipeline ROS

Mede-se o atraso introduzido pela parte ROS do pipeline, entre o comando de gesto publicado em `/arm/cmd` e o início

do movimento articular (`/joint_states`), decomposto no trecho da ponte de teleoperação e no da cadeia Servo/controle/-simulador. Reporta-se a média, a mediana e o percentil 95, de modo a capturar tanto o comportamento típico quanto os piores casos. A latência da etapa de visão (câmera $\rightarrow$ MediaPipe $\rightarrow$ `/arm/cmd`) fica fora deste escopo.

B. Exp. B: Robustez do reconhecimento de gestos

Mede-se a taxa de detecção da mão pelo MediaPipe Hands (a fração de quadros em que a mão é efetivamente localizada) sob variações de iluminação, distância da câmera e ângulo da mão. Essa taxa é o melhor preditor da disponibilidade do controle gestual: quando a mão deixa de ser detectada, o operador perde momentaneamente o comando. Reporta-se a taxa por condição experimental e a taxa de quadros efetiva (FPS), identificando o envelope operacional em que o sistema se mantém utilizável.

C. Exp. C: Prevenção de singularidade

Em vez de usar a câmera, o robô é levado por uma sequência de poses conhecidas, escolhidas para cobrir desde regiões saudáveis até poses muito próximas de singularidades. Verifica-se com que frequência o sistema bloqueia corretamente as poses perigosas (taxa de prevenção) e com que frequência bloqueia poses que eram seguras (falso positivo).

D. Exp. D-sim: Prevenção de colisão (simulação)

No ambiente simulado, comanda-se o robô para tentar atravessar os obstáculos da cena e verifica-se se a penetração é prevenida antes do contato pela prevenção geométrica do `collision_guard` interposto no caminho de comando.

E. Exp. D-real: Prevenção de colisão (robô real)

Verifica-se no robô físico se a prevenção de colisão de fato funciona: conduz-se a garra por gestos em direção a cada obstáculo e observa-se se o `collision_guard` barra o movimento de aproximação antes do contato, sem que ocorra colisão física e preservando a capacidade do operador de afastar a garra. A avaliação é qualitativa, baseada na observação direta do comportamento em uso real.

Calibração de bancada validada: em ensaio preliminar, a ponta da garra parou a 3 cm da caixa, com folga lateral ainda maior, como esperado para a esfera de proteção conservadora. O modelo do robô foi ajustado às dimensões reais da montagem (pedestal da base de $0.20 \times 0.20 \times 0.36$ m e garra de 0.13 m). A validação de campo confirmou a prevenção em uso real, sem colisões físicas.

VII. RESULTADOS E DISCUSSÃO

Os resultados quantitativos dos experimentos em simulação e dos componentes de segurança, bem como a validação de campo no robô físico (prevenção de colisão e de singularidade em uso real e a demonstração de pick-and-place), são apresentados a seguir.

Tabela I: Latência do pipeline ROS (Exp. A, robô simulado; 13 pulsos; L_1 e L_2 com 11 amostras pareadas).

Trecho	Média	P50	P95	Unidade
L_1 (<code>/arm/cmd</code> $\rightarrow$ <code>twist</code>)	34,3	37,3	48,8	ms
L_2 (<code>twist</code> $\rightarrow$ <code>/joint_states</code>)	37,2	36,4	46,5	ms
L_{total}	70,5	69,3	87,1	ms

A. Latência do pipeline ROS (Exp. A)

Para situar o sistema em relação ao tempo real, mediu-se o atraso introduzido pela parte ROS do pipeline, no robô simulado (MoveIt Servo + `ros2_control` + Gazebo): do comando de gesto publicado em `/arm/cmd` até o início do movimento das juntas (`/joint_states`). É importante frisar que esta medida cobre apenas o trecho ROS; a latência da etapa de visão (câmera $\rightarrow$ MediaPipe $\rightarrow$ `/arm/cmd`) não foi medida e fica como limitação desta avaliação.

O atraso é decomposto em dois trechos (Tabela I): L_1 , da publicação do gesto à geração do comando de velocidade (`twist`) pela ponte de teleoperação; e L_2 , desse `twist` até a primeira mudança efetiva nas juntas. L_{total} é a soma: o tempo que um comando leva, dentro do ROS, para se tornar movimento. Para cada trecho, a tabela traz a média, a mediana (P50) e o percentil 95 (P95, o “pior caso típico”: 95 % das medições ficaram abaixo desse valor).

Na prática, o atraso total fica em torno de 70 ms (P95 87 ms), abaixo do alvo de 100 ms usualmente citado para teleoperação direta, e se reparte de forma equilibrada entre os dois trechos (≈ 34 ms e ≈ 37 ms), sem que nenhum domine sozinho. Ou seja, a parte ROS do sistema responde com folga dentro do orçamento de tempo real.

B. Robustez do reconhecimento de gestos (Exp. B)

A taxa de detecção da mão primária pelo MediaPipe Hands foi medida em 7 condições, variando iluminação (normal/baixa), distância (0.5 m, 1.0 m e 1.5 m) e ângulo (frontal/lateral 45°), incluindo um pior caso combinado (luz baixa + 1.5 m + lateral). Em todas as condições a taxa de detecção foi de 100 %: no envelope testado, o MediaPipe Hands mostrou-se robusto a iluminação, distância e ângulo. A taxa de quadros, por sua vez, ficou em torno de 9 Hz (média 9.25 quadros/s sobre as sete condições), o que evidencia a necessidade de uma CPU robusta para que o controle gestual responda de forma fluida.

C. Monitor de singularidade (Exp. C)

A Tabela II resume o desempenho do monitor de singularidade: o cálculo da manipulabilidade a cada amostra leva, em média, 0.27 ms; somando a entrega da mensagem pelo sistema, o atraso total do monitor fica em 5.17 ms em média. O nó roda de forma estável a 20 Hz e, em um conjunto de poses sabidamente seguras, não disparou nenhum alarme indevido, ou seja, taxa de falso positivo nula.

A eficácia da prevenção foi avaliada em simulação, conduzindo o efetuator repetidamente em direção à singularidade (8 eventos de aproximação). Para que o teste medisse de fato a

Tabela II: Desempenho do `safety_monitor`.

Métrica	Média	P95	Unidade
Tempo de cálculo de w	0,27	0,43	ms
Atraso total do monitor	5,17	10,29	ms
Frequência de operação	20,00		Hz
Taxa de falso positivo	0		%

prevenção, o robô foi movido por comandos de velocidade cartesiana, passando pelo MoveIt Servo. Nos 8 eventos o conjunto monitor + Servo impediu o robô de atingir a singularidade; em um deles o Servo chegou a acionar a parada por singularidade.

No robô real, onde o MoveJog contorna o MoveIt Servo e o limiar de manipulabilidade é a única barreira contra singularidades, é preciso calibrar esse limiar a um valor adequado para que o robô desacelere e pare antes de atingir a configuração singular. A validação de campo confirmou esse comportamento: ao conduzir o efetuador para o cotovelo estendido, a manipulabilidade caía abaixo do limiar e o avanço era barrado, com o operador recuperando o controle ao gesticular no sentido oposto.

D. Verificação geométrica de colisão (Exp. D-real)

A rotina que calcula a distância da garra a cada obstáculo e veta o movimento de aproximação foi verificada em um conjunto de oito casos de teste cobrindo chão, parede, caixa, cilindro e esfera, além do próprio veto de velocidade. Em todos eles a distância e a direção calculadas coincidiram com os valores esperados, confirmando que a base geométrica da prevenção está correta.

No robô físico, a prevenção foi exercitada em uso real, conduzindo a garra por gestos contra o chão e a caixa. Em todas as aproximações o `collision_guard` barrou o movimento antes do contato e nenhuma colisão física ocorreu: a garra parava ao atingir a margem de segurança configurada, com a ponta detendo-se a 3 cm da caixa e folga lateral ainda maior (comportamento esperado para a esfera de proteção conservadora que a envolve). O operador voltava a se mover livremente apenas afastando a garra, já que somente a componente de velocidade dirigida ao obstáculo é vetada. Confirma-se, assim, que a prevenção geométrica de colisão opera no robô físico, com zero colisões e parada na margem configurada.

E. Prevenção de colisão em simulação (Exp. D-sim)

A prevenção de colisão em simulação foi feita pelo mesmo `collision_guard` usado no robô real, interposto no caminho `/arm/cmd` → `/arm/cmd_safe` → `ponte` → `Servo`, mantendo uma única implementação de prevenção, leve e portátil, entre os dois caminhos.

O comportamento foi o mesmo observado no robô físico: nos 3 eventos de aproximação o robô parou antes do contato, com distância mínima de 2.0 cm (margem configurada de 3 cm) e nenhuma penetração. A latência de processamento do `collision_guard` por mensagem ficou em torno de 3.6 ms (P95 7.5 ms), de modo que o acréscimo de latência introduzido



Figura 5: Demonstração de pick-and-place no robô real: o CR3 segura a caixa pela chapa metálica superior com a garra magnética (eletroímã), conduzido por teleoperação gestual com a prevenção de singularidade e de colisão ativas.

pela verificação de segurança é desprezível e não compromete a resposta do controle gestual. Confirma-se, assim, que uma mesma verificação geométrica de colisão, leve e portátil, previne a penetração tanto no robô real quanto em simulação.

F. Demonstração de pick-and-place no robô real

A validação ponta a ponta do sistema foi concluída com uma operação de pick-and-place executada inteiramente por teleoperação gestual no robô físico. O efetuador utilizado é um eletroímã comutado por um relé, acionado por um nó ROS na Raspberry Pi (Seção V-F): o gesto de fechar a mão publica "fecha" no tópico da garra, energizando o eletroímã e prendendo a peça; o gesto inverso publica "abre" e a libera. Na demonstração, o operador conduziu a garra por gestos até uma caixa, energizou o eletroímã para agarrá-la, transportou-a até a posição-alvo e a soltou ao desenergizar, tudo com a prevenção de singularidade e a de colisão ativas durante todo o percurso (Figura 5). Como o eletroímã só adere a material ferromagnético, cada caixa recebe uma chapa metálica fixada na face superior, que serve de ponto de preensão para a garra.

A demonstração é significativa por integrar, em uma única tarefa de manipulação, todos os componentes desenvolvidos no trabalho: o reconhecimento gestual de duas mãos (direção e velocidade), o acionamento desacoplado da garra, o monitor de singularidade e o `collision_guard`. Vale destacar que o acoplamento por mensagem de texto em um tópico ROS torna a substituição do efetuador trivial: trocar o eletroímã por uma

garra de dedos, pneumática ou de sucção exige apenas adaptar o script que roda na Raspberry Pi, sem qualquer alteração no pipeline de visão nem nas camadas de prevenção.

G. Discussão

Os números obtidos sustentam o argumento central do trabalho: a segurança pode ser reintroduzida no caminho do robô por verificações leves, compatíveis com a teleoperação em tempo real. O pipeline ROS responde em 70 ms de média (P95 87 ms), abaixo do alvo de 100 ms; o monitor de singularidade custa ~ 0.3 ms por amostra e preveniu 100 % das aproximações testadas; e a verificação geométrica de colisão (~ 3.6 ms por mensagem) parou o robô na margem configurada, sem penetração, tanto no caminho real quanto no simulado. Esse último ponto é relevante: uma mesma verificação geométrica, própria e portátil, protege os dois caminhos, o que dispensa delegar a segurança ao Servo e mantém a prevenção idêntica em simulação e no robô real.

Quanto à usabilidade, o reconhecimento de gestos mostrou-se robusto (100 % de detecção no envelope testado), de modo que a fluidez do controle depende sobretudo da taxa de quadros (~ 9 Hz), que requer uma CPU robusta.

A validação no robô físico fechou o argumento: tanto a prevenção de colisão quanto a de singularidade operaram corretamente durante a teleoperação gestual, sem colisões físicas e com parada na margem configurada, e o sistema completou uma tarefa de pick-and-place ponta a ponta, da preensão por eletroímã ao transporte e à liberação, com as duas camadas de prevenção ativas. A estratégia de vetar apenas a componente de velocidade que aponta para o obstáculo segue a mesma linha de trabalhos anteriores de teleoperação com prevenção geométrica [14], [15], mas aplicada aqui a um cobot industrial comandado apenas por gestos captados por câmera, sem nenhum dispositivo físico de controle nas mãos do operador.

VIII. CONCLUSÃO E TRABALHOS FUTUROS

A. Conclusão

Este trabalho apresentou um sistema de teleoperação gestual do cobot Dobot CR3 que integra, em um único pipeline ROS 2, o reconhecimento de gestos por MediaPipe Hands e a prevenção em tempo real de singularidades e de colisões, tanto em simulação quanto no robô real. A principal contribuição é metodológica: uma arquitetura dual-path que compartilha o módulo de gestos, o contrato de tópicos, a cena de colisão e o monitor de segurança entre os dois caminhos, de modo que o sistema possa ser desenvolvido e validado sem o hardware e migrar para ele redirecionando apenas a saída do pipeline. Como o caminho do robô real emprega o MoveJog e contorna o MoveIt Servo, a prevenção de colisão foi reintroduzida por um verificador geométrico leve (o `collision_guard`), na forma de uma forbidden-region virtual fixture com fuga direcional que preserva a usabilidade.

As evidências experimentais sustentam a viabilidade da proposta. Em simulação, o monitor de singularidade e a

verificação geométrica de colisão operaram dentro do orçamento de tempo real, sem falsos positivos e prevenindo todas as aproximações testadas, e o reconhecimento de gestos mostrou-se robusto em todo o envelope avaliado. No robô físico, a validação de campo confirmou os três pilares do sistema em uso real: a prevenção de colisão deteve a garra na margem de segurança sem nenhuma colisão física; a prevenção de singularidade bloqueou o avanço para a configuração de cotovelo, com recuperação por fuga direcional; e o sistema completou uma tarefa de pick-and-place ponta a ponta por teleoperação gestual, com a garra (eletroímã acionado por relé via Raspberry Pi) e as duas camadas de prevenção operando de forma integrada.

B. Trabalhos futuros

- A) **Modelo mesh do efetuador via FCL.** Hoje a garra é aproximada por uma esfera de proteção, solução simples mas conservadora: o movimento é barrado mais cedo do que o necessário, sobretudo nas laterais. Substituir essa esfera pelo modelo mesh real do efetuador, com a distância calculada pela FCL [17], daria medidas mais precisas e permitiria ao robô se aproximar dos obstáculos onde for geometricamente seguro.
- B) **Biblioteca de cinemática dedicada.** O monitor de singularidade monta a cadeia KDL manualmente em tempo de execução, o que é frágil e específico para a cadeia revolvente do CR3. Trocá-la por uma biblioteca dedicada (como o Pinocchio) tornaria o código mais robusto, generalizaria o monitor para cadeias com outros tipos de junta (prismáticas e mimic) e abriria caminho para a manipulabilidade dinâmica, ponderada pela matriz de inércia do robô.
- C) **Integração com OctoMap (a mais importante).** Hoje os obstáculos são declarados manualmente em uma cena estática, com posições medidas de antemão. Com o OctoMap, alimentado por um sensor de profundidade, o robô passaria a desviar de obstáculos detectados em tempo real, sem cadastro prévio, tornando a prevenção de colisão muito mais geral e aplicável a ambientes não estruturados.

AGRADECIMENTOS

Agradeço, acima de tudo, aos meus pais, Maria das Graças Caires Mirante e José Oliveira Mirante, cujo esforço e dedicação tornaram possível a minha graduação e a realização deste sonho. À minha irmã Vanessa e ao meu cunhado Thasso, pelo apoio constante e pelo incentivo em todos os momentos. Estendo minha gratidão a toda a minha família e aos meus amigos que, de alguma forma, me deram forças para continuar e seguir em frente.

Agradeço ao professor Alexandre Santos Brandão pela orientação ao longo de todo o trabalho, e ao NERo (Núcleo de Especialização em Robótica) da Universidade Federal de Viçosa pela estrutura e pelo ambiente de pesquisa que tornaram este projeto possível. Agradeço também aos amigos Luciano

Gonçalves Moreira e Werikson Alves, pelas discussões e ideias que contribuíram para o desenvolvimento do sistema.

REFERÊNCIAS

- [1] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “MediaPipe Hands: On-device real-time hand tracking,” *arXiv preprint arXiv:2006.10214*, 2020.
- [2] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee *et al.*, “MediaPipe: A framework for building perception pipelines,” in *arXiv preprint arXiv:1906.08172*, 2019.
- [3] S. Chitta, I. Sucan, and S. Cousins, “MoveIt!: An introduction,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.
- [4] L. B. Rosenberg, “Virtual fixtures: Perceptual tools for telerobotic manipulation,” in *Proceedings of IEEE Virtual Reality Annual International Symposium*, 1993, pp. 76–82.
- [5] J. J. Abbott, P. Marayong, and A. M. Okamura, “Haptic virtual fixtures for robot-assisted manipulation,” *Robotics Research, Springer Tracts in Advanced Robotics*, vol. 28, pp. 49–64, 2007.
- [6] M. I. Kurpath, P. K. Adwai, J. Bodireddy, K. Chandrasekaran, and K. Navin Sam, “An IMUs and potentiometer-based controller for robotic arm-hand teleoperation,” *Sensors and Actuators A: Physical*, vol. 367, p. 115019, 2024.
- [7] B. Fang, F. Sun, H. Liu, and C. Liu, “3D human gesture capturing and recognition by the IMMU-based data glove,” *Neurocomputing*, vol. 277, pp. 198–207, 2018.
- [8] G. Du, P. Zhang, J. Mai, and Z. Li, “Markerless Kinect-based hand tracking for robot teleoperation,” *International Journal of Advanced Robotic Systems*, vol. 9, no. 2, p. 36, 2012.
- [9] I. Jang, J. Carrasco, A. Weightman, and B. Lennox, “Intuitive bare-hand teleoperation of a robotic manipulator using virtual reality and Leap Motion,” in *Towards Autonomous Robotic Systems (TAROS)*, ser. Lecture Notes in Computer Science. Springer, 2019, pp. 283–294.
- [10] M. Altayeb, “Hand gestures replicating robot arm based on MediaPipe,” *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, vol. 11, no. 3, 2023.
- [11] L. A. Zick, D. Martinelli, A. Schneider de Oliveira, and V. Cremer Kalempe, “Teleoperation system for multiple robots with intuitive hand recognition interface,” *Scientific Reports*, vol. 14, no. 1, p. 30230, 2024.
- [12] J. Xie, Z. Xu, J. Zeng, Y. Gao, and K. Hashimoto, “Human–robot interaction using dynamic hand gesture for teleoperation of quadruped robots with a robotic arm,” *Electronics*, vol. 14, no. 5, p. 860, 2025.
- [13] T. Yoshikawa, “Manipulability of robotic mechanisms,” *The International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.
- [14] J. T. Feddema and J. L. Novak, “Whole arm obstacle avoidance for teleoperated robots,” in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1994.
- [15] S. Nia Kosari, F. Rydén, T. S. Lendvay, B. Hannaford, and H. J. Chizeck, “Forbidden region virtual fixtures from streaming point clouds,” *Advanced Robotics*, vol. 28, no. 22, pp. 1507–1518, 2014.
- [16] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [17] J. Pan, S. Chitta, and D. Manocha, “FCL: A general purpose library for collision and proximity queries,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3859–3866.