Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos 6627, CEP 31270-901,
Belo Horizonte, MG Brasil

DISSERTAÇÃO DE MESTRADO

Controle Servo Visual para Replanejamento da Tarefa de um Robô Manipulador para Inspeção de Peças Automotivas

Lucca Garcia Leão

Belo Horizonte
2023

Lucca Garcia Leão

Controle Servo Visual para Replanejamento da Tarefa de um Robô Manipulador para Inspeção de Peças Automotivas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Gustavo Medeiros Freitas

Belo Horizonte
2023

Leão, Lucca Garcia.
L437c Controle servo visual para replanejamento da tarefa de um robô manipulador para inspeção de peças automotivas [recurso eletrônico] / Lucca Garcia Leão. - 2023.
1 recurso online (107 f. : il., color.) : pdf.

Orientador: Gustavo Medeiros Freitas.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Bibliografia: f. 101-105.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Robôs - Teses. 3. Robótica - Teses. 4. Automóveis - Peças - Teses. 5. Controle de qualidade - Teses. 6. Automação - Teses. 7. Robôs - Sistemas de controle - Teses. 8. Visão por computador - Teses. 9. Controle (Engenharia) - Teses. 10. Processos de fabricação - Teses. I. Freitas, Gustavo Medeiros. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pela bibliotecária Ângela Cristina Silva CRB/6 2361
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"DESENVOLVIMENTO E COMPARAÇÃO DE TÉCNICAS DE CONTROLE DE UM ROBÔ MANIPULADOR PARA AUTOMATIZAÇÃO DE UMA CÉLULA DE INSPEÇÃO DE PEÇAS AUTOMOTIVAS"

LUCCA GARCIA LEÃO

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica. Aprovada em 10 de outubro de 2023. Por:

Prof. Dr. Gustavo Medeiros Freitas
DEE (UFMG) - Orientador

Prof. Dr. Bruno Nazário Coelho
DECAT (UFOP)

Dr. Luiz Fernando Etrusco Moreira
(Invent Vision)

Documento assinado eletronicamente por **Gustavo Medeiros Freitas**, **Professor do Magistério Superior**, em 10/10/2023, às 16:32, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Fernando Etrusco Moreira**, **Usuário Externo**, em 16/10/2023, às 10:10, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Bruno Nazário Coelho**, **Usuário Externo**, em 17/10/2023, às 19:17, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2696478** e o código CRC **A5CF2E7E**.

Referência: Processo nº 23072.261591/2023-46 SEI nº 2696478

À minha família.

Agradecimentos

Agradeço primeiramente a meus pais, José e Kátia, cujo apoio incondicional me proporcionou a oportunidade de realizar meu sonho de estudar em uma universidade como a UFMG. Aos meus irmãos, Julianna e Thiago, por me ajudarem a não perder o ânimo em situações difíceis.

Também gostaria de agradecer à Invent Vision, em particular ao Luiz Fernando Etrusco Moreira e ao Antônio Otávio Fernandes, pelo suporte e fornecimento de recursos imprescindíveis para a realização da pesquisa presente nesta dissertação.

Ao meu orientador, Gustavo Medeiros Freitas, que durante anos da minha graduação e mestrado, se dedicou à minha formação como pesquisador.

Aos meus amigos Luiz, Thiago, Matheus, Ítalo, Lucas, João, Daniel, e a todos que tornaram a decisão de vir para Belo Horizonte uma das melhores que já tomei em minha vida.

Por fim, gostaria de agradecer especialmente à minha companheira de vida, Carol, por estar sempre do meu lado, nos melhores e piores momentos, e por sempre enxergar o melhor em mim. Obrigado por fazer parte da minha vida.

Resumo

A quarta Revolução Industrial, ou Indústria 4.0, trouxe grandes mudanças para os sistemas de manufatura e processos de controle de qualidade por meio de novas tecnologias como a robótica e visão de máquina. A visão de máquina é uma das técnicas mais comumente empregadas para controle de qualidade de processos industriais, permitindo capturar, processar, e inspecionar imagens de peças e produtos em uma linha de produção com o intuito de identificar falhas no processo produtivo, como ausência de componentes, má fixação, manchas e trincas. Contudo, a inspeção de peças com geometria cada vez mais complexas exige que os sistemas de visão, normalmente equipados com câmeras fixas, sejam capazes de capturar imagens em diversas orientações diferentes. Assim, a combinação de manipuladores robóticos a sistemas de visão é uma solução que permite que a câmera seja reposicionada, expandindo a flexibilidade e eficácia de tais sistemas. Esta dissertação se baseia em um estudo de caso sobre a célula de inspeção de peças automotivas da empresa Invent Vision. Esta célula realiza a inspeção automática por meio de algoritmos de visão computacional para controle de qualidade das peças produzidas. A célula conta com um robô manipulador e uma câmera montada em seu efetuador para a realização de inspeções visuais de peças complexas. Ao operacionalizar a máquina, uma peça é posicionada em seu interior e o robô realiza a inspeção, que é definida como uma sequência de poses, nas quais são capturadas imagens pela câmera. Entretanto, o robô espera que a peça sempre esteja na mesma posição para que as imagens estejam na perspectiva correta e a inspeção seja bem sucedida, e portanto, problemas podem ocorrer devido ao mau posicionamento de peça dentro da célula. Nesse sentido, esta dissertação apresenta uma estratégia de replanejamento do movimento realizado pelo robô na tarefa da célula de inspeção, baseado na pose do objeto estimada a partir da identificação de um marcador fiducial. O sistema proposto é baseado em técnicas de Controle Servo Visual e tem como objetivo aprimorar o fluxo de inspeção original da célula, e contornar a imposição do posicionamento da peça, realizando o replanejamento das poses de inspeção com base na posição estimada da peça. O sistema foi inicialmente validado em ambiente de simulação. Em seguida um ambiente experimental de validação foi construído, utilizando um robô real Kuka KR4 R600 e uma câmera industrial Invent Vision idênticos aos utilizados nos sistemas em produção. Os resultados mostram que o sistema foi capaz de realizar a inspeção da peça com sucesso, mesmo com grandes variações de pose da peça dentro do espaço de trabalho. Os experimentos também demonstram que a escolha do tipo de movimento resulta em diferentes trajetórias entre os pontos de inspeção, porém sem alterar o resultado da inspeção. A metodologia proposta é versátil e pode ser utilizada em diferentes aplicações.

Palavras-chave: robótica; visão de máquina; controle de qualidade; replanejamento de tarefa; controle servo visual.

Abstract

The fourth Industrial Revolution, also called Industry 4.0, brought many changes to manufacturing systems and quality control processes through new technologies such as robotics and machine vision. Machine vision is one of the most commonly employed techniques for quality control of industrial processes, as it allows the capture, processing and inspection of products with the intent of identifying failures in the production process, such as absence of certain components, displacement, smudges and cracks. However, the inspection of objects with increasingly complex geometry demands machine vision systems, which are commonly composed of one single static camera, to be able to capture images in various different perspectives. Thus, the combination of robotic manipulators and machine vision systems can be a solution to move the camera around to desired positions, increasing the flexibility and effectiveness of said systems. This dissertation is based on the case study of the Invent Vision automated inspection cell. This cell performs automatic inspection using computer vision algorithms for quality control of the produced parts. The cell is equipped with a robotic manipulator with a camera mounted in its end-effector to perform visual inspection of complex vehicle parts. Upon operating the machine, the object is placed in its interior and the robot performs an inspection, defined by a sequence of desired poses where images are captured. However, the robot expects the inspected part to always be in the same position in order to obtain the correct image perspectives and perform a succesfull inspection, and therefore, problems could occur due to mispositioning of the object inside the cell. In this sense, this dissertation proposes a task replanning strategy for the robot movement during an inspection. The replanning is calculated with relation to the pose of the inspected component, which is estimated by identifying a fiducial marker. The proposed system is based on Visual Servo Control techniques and aims to improve the original workflow of the inspection cell by avoiding the imposition of part position, while performing the task replanning based on its estimated pose. The system was initially validated in a simulation environment. Then, an experimental setup was constructed with a real Kuka KR4 R600 and an industrial Invent Vision camera identical to the ones in real production environments. The inspections were performed using a real part, and a template matching algorithm to detect the presence of certain components. Experimental results show that the proposed system was capable of performing the inspection of a part with success, even when subject to great pose variation in the workspace. The experiments also show that the choice of movement result in different trajectories between the inspection points, but without altering the inspection result. The proposed methodology is versatile and can be also used in other applications.

Keywords: *robotics; machine vision; quality control; task replanning; visual servo control.*

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

UFMG	Universidade Federal de Minas Gerais
BH-TEC	Parque Tecnológico de Belo Horizonte
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
FIEMG	Federação das Indústrias do Estado de Minas Gerais
CBA	Congresso Brasileiro de Automática
SBAI	Simpósio Brasileiro de Automação Inteligente
ROS	*Robot Operating System*
CCD	Charge-coupled device
PAUT	*Phased Array Ultrasonic Testing*
ToF	*Time-of-Flight*
VANT	Veículo Aéreo Não Tripulado
DH	Denavit-Hartenberg
RPY	*Roll, Pitch, Yaw*
RGB	*Reg, Green, Blue*
HSV	*Hue, Saturation, Value*
IBVS	*Image Based Visual Servo*
PBVS	*Position Based Visual Servo*
CLP	Controlador Lógico Programável
PnP	*Perspective-n-Point*
TCP/IP	Transmission Control Protocol/Internet Protocol
CAD	*Computer-Aided Design*

VGA	*Video Graphics Array*
URDF	*Unified Robot Description Format*

Lista de Símbolos

$\mathbf{p_A^O}$	Vetor de posição do sistema de coordenadas A com respeito ao sistema de coordenadas O
$p_{A,x}^O$	Componente x do vetor $\mathbf{p_A^O}$
$p_{A,y}^O$	Componente y do vetor $\mathbf{p_A^O}$
$p_{A,z}^O$	Componente z do vetor $\mathbf{p_A^O}$
$\mathbf{R_A^O}$	Matriz de rotação do sistema de coordenadas A com respeito ao sistema de coordenadas O
$\varphi_\mathbf{A}^\mathbf{O}$	Representação de $roll, pitch, yaw$ do sistema de coordenadas A com respeito ao sistema de coordenadas O
ϕ_A^O	Componente $roll$ do vetor $\varphi_\mathbf{A}^\mathbf{O}$
θ_A^O	Componente $pitch$ do vetor $\varphi_\mathbf{A}^\mathbf{O}$
ψ_A^O	Componente yaw do vetor $\varphi_\mathbf{A}^\mathbf{O}$
$\mathbf{x_A^O}$	Representação da pose do sistema de coordenadas A com respeito ao sistema de coordenadas O
$\mathbf{H_A^O}$	Matriz de transformação homogênea entre o sistema de coordenadas A e o sistema de coordenadas O
$\bar{\mathbf{p}}_\mathbf{B}^\mathbf{O}$	Vetor de posição do sistema de coordenadas B com respeito ao sistema de coordenadas O
$p_{B,x}^O$	Componente x do vetor $\bar{\mathbf{p}}_\mathbf{B}^\mathbf{O}$
$p_{B,y}^O$	Componente y do vetor $\bar{\mathbf{p}}_\mathbf{B}^\mathbf{O}$
$p_{B,z}^O$	Componente z do vetor $\bar{\mathbf{p}}_\mathbf{B}^\mathbf{O}$
$\bar{\mathbf{p}}_\mathbf{B}^\mathbf{A}$	Vetor de posição do sistema de coordenadas B com respeito ao sistema de coordenadas A

$p^A_{B,x}$	Componente x do vetor $\bar{\mathbf{p}}^{\mathbf{A}}_{\mathbf{B}}$
$p^A_{B,y}$	Componente y do vetor $\bar{\mathbf{p}}^{\mathbf{A}}_{\mathbf{B}}$
$p^A_{B,z}$	Componente z do vetor $\bar{\mathbf{p}}^{\mathbf{A}}_{\mathbf{B}}$
q_i	Valor de ângulo da junta junta i
$o_i x_i y_i z_i$	Sistema de coordenadas do elo i de um manipulador robótico
$o_{i-1} x_{i-1} y_{i-1} z_{i-1}$	Sistema de coordenadas do elo $i-1$ de um manipulador robótico
$\mathbf{H}_i(q_i)$	Matriz de transformação homogênea do sistema de coordenadas do elo i de um manipulador robótico
$\mathbf{H}^{\mathbf{0}}_{\mathbf{n}}$	Matriz de transformação homogênea do sistema de coordenadas do efetuador em relação ao sistema de coordenadas inercial
$\mathbf{R}^{\mathbf{0}}_{\mathbf{n}}$	Matriz de rotação do efetuador em relação ao sistema de coordenadas inercial
$\mathbf{p}^{\mathbf{0}}_{\mathbf{n}}$	Vetor de posição do efetuador em relação ao sistema de coordenadas inercial
$\mathbf{R}^{i-1}_i$	Matriz de rotação entre o sistema de coordenadas do elo i e o sistema de coordenadas do elo $i-1$
$\mathbf{p}^{i-1}_i$	Vetor de posição entre o sistema de coordenadas do elo i e o sistema de coordenadas do elo $i-1$
$\mathbf{H}^i_j$	Matriz de transformação homogênea entre o sistema de coordenadas do elo j e o sistema de coordenadas do elo i
$\mathbf{R}^i_j$	Matriz de rotação entre o sistema de coordenadas do elo j e o sistema de coordenadas do elo i
$\mathbf{p}^i_j$	Vetor de posição entre o sistema de coordenadas do elo j e o sistema de coordenadas do elo i
n	Número de juntas de um robô manipulador
$\mathbf{x}$	Representação da pose do efetuador
$\dot{\mathbf{x}}$	Vetor de velocidade $\in \mathbb{R}^{6\times1}$ do efetuador
$\mathbf{v_e}$	Vetor de velocidade $\in \mathbb{R}^{3\times1}$ linear do efetuador
$\omega_{\mathbf{e}}$	Vetor de velocidade $\in \mathbb{R}^{3\times1}$ angular do efetuador

$\mathbf{J}(\mathbf{q})$	Matriz Jacobiana geométrica $\in \mathbb{R}^{6\times n}$ de um robô manipulador
$\mathbf{J_P}(\mathbf{q})$	Porção da velocidade linear da matriz Jacobiana $\in \mathbb{R}^{3\times n}$
$\mathbf{J_O}(\mathbf{q})$	Porção da velocidade angular da matriz Jacobiana $\in \mathbb{R}^{3\times n}$
$\dot{\mathbf{q}}$	Vetor de velocidades das juntas do robô
$\mathbf{J_{P_i}}$	Coluna i da matriz $\mathbf{J_P}(\mathbf{q})$
$\mathbf{J_{O_i}}$	Coluna i da matriz $\mathbf{J_O}(\mathbf{q})$
$\mathbf{z_{i-1}}^0$	Vetor unitário que aponta na direção do eixo $\mathbf{z}$ da junta i
p^0_{i-1}	Vetor posição do sistema de coordenadas do eixo i com relação ao sistema de coordenadas inercial
$\mathbf{u}$	Comando de velocidade das juntas
$\mathbf{x_d}$	Pose desejada
$\mathbf{e}$	Erro de pose entre $\mathbf{x_d}$ e $\mathbf{x}$
$\mathbf{e_P}$	Erro de posição $\in \mathbb{R}^{3\times 1}$ do efetuador
$\mathbf{e_O}$	Erro de orientação $\in \mathbb{R}^{3\times 1}$ do efetuador
$\dot{\mathbf{e}}$	Derivada do erro de pose do efetuador
$\dot{\mathbf{x_d}}$	Derivada da pose desejada do efetuador
$\dot{\mathbf{x}}$	Derivada da pose do efetuador
$\mathbf{p_d}(t)$	Posição desejada do efetuador do efetuador
$\dot{\mathbf{e_P}}$	Derivada do erro de posição do efetuador
$\dot{\mathbf{p_d}}(t)$	Derivada da posição desejada do efetuador
$\mathbf{R}_d$	Matriz de rotação desejada do efetuador
Q_d	Quatérnio unitário que representa a rotação desejada do efetuador
η_d	Parte vetorial de Q_d
ϵ_d	Magnitude de Q_d
Q_e	Quatérnio unitário que representa a rotação atual do efetuador
η_e	Parte vetorial de Q_e

ϵ_e	Magnitude de Q_e
ΔQ	Quatérnio que representa a diferença entre Q_d e Q_e
$\Delta\eta$	Parte vetorial de ΔQ
$\Delta\epsilon$	Magnitude de ΔQ
$[u, v]$	Coordenadas de um ponto da imagem
$[\hat{u}, \hat{v}]$	Melhor estimativa das coordenadas da localização da amostra na imagem
$\rho_{12}(u, v)$	Função de correlação normalizada entre duas imagens
$g_1(u, v)$	Valor do pixel da imagem g_1 na posição $[u, v]$
$g_2(u, v)$	Valor do pixel da imagem g_2 na posição $[u, v]$
$\sigma_{g_1 g_2}(u, v)$	Covariância entre os valores de intensidade na área de sobreposição entre as imagens g_1 e g_2 no ponto $[u, v]$
$\sigma_{g_1}(u, v)$	Desvio padrão dos valores de intensidade da imagem g_1 na área de sobreposição com a imagem g_2 no ponto $[u, v]$
σ_{g_2}	Desvio padrão dos valores de intensidade da imagem g_2
$\sigma_{g_1 g_2}(u, v)$	Covariância entre os valores de intensidade na área de sobreposição das imagens g_1 e g_2 no ponto $[u, v]$
$\mathbf{K}$	Matriz de ganho proporcional da lei de controle
$\mathbf{p_i}(t_i)$	Posição inicial da trajetória desejada
$\mathbf{p_f}(t_f)$	Posição final da trajetória desejada
q_0	Posição inicial da junta
v_0	Velocidade inicial da junta
a_0	Aceleração inicial da junta
q_f	Posição final da junta
v_f	Velocidade final da junta
a_f	Aceleração final da junta
t_0	Tempo inicial da trajetória
t_f	Tempo final da trajetória

K_c	Matriz de parâmetros intrínsecos da câmera
f_x	Distância focal no eixo x
f_y	Distância focal no eixo y
c_x	Coordenada x do centro óptico
c_y	Coordenada y do centro óptico
x_{radial}	Distorção radial no eixo x
y_{radial}	Distorção radial no eixo y
$x_{tangencial}$	Distorção tangencial no eixo x
$y_{tangencial}$	Distorção tangencial no eixo y
$\mathbf{C}$	Matriz de câmera $\in \mathbb{R}^{3\times4}$
$\mathbf{H}^{\mathbf{B}}_{\mathbf{E}}$	Matriz de transformação homogênea do efetuador do robô **E** com respeito à base **B**
$\mathbf{H}^{\mathbf{C}}_{\mathbf{M}}$	Matriz de transformação homogênea do marcador **M** com respeito à câmera **C**
$\mathbf{H}^{\mathbf{E}}_{\mathbf{C}}$	Matriz de transformação homogênea da câmera **C** com respeito ao efetuador **E**
$\mathbf{H}^{\mathbf{B}}_{\mathbf{M}}$	Matriz de transformação homogênea do marcador **M** com respeito à base **B**
$\mathbf{H}^{\mathbf{B}}_{\mathbf{E},d}$	Matriz de transformação homogênea desejada do efetuador **E** com respeito à base **B**
$\mathbf{H}^{\mathbf{C}\ -1}_{\mathbf{M},d}$	Matriz de transformação homogênea desejada do marcador **M** com respeito à câmera **C**
$\mathbf{p}^{\mathbf{C}}_{\mathbf{M},d}$	Vetor de translação desejado do marcador **M** com respeito à câmera **C**
$\mathbf{R}^{\mathbf{C}}_{\mathbf{M},d}$	Matriz de rotação desejada do marcador **M** com respeito à câmera **C**

Sumário

Capítulo 1

Introdução

Processos industriais avançados exigem desempenho cada vez mais aprimorado com uma necessidade constante de controle de qualidade durante o processo produtivo. A presença de defeitos como arranhões, manchas, e buracos na superfície de um produto podem gerar um prejuízo não apenas estético, como também de desempenho e segurança, que podem acarretar em perdas inestimáveis para o fabricante. Nesse sentido, a detecção de defeitos no processo de controle de qualidade se torna cada vez mais presente e necessária no contexto da indústria moderna nos mais diversos setores.

Nas últimas décadas, robôs manipuladores surgiram como ferramentas indispensáveis em vários setores industriais, revolucionando processos de fabricação e aumentando a produtividade, devido a sua capacidade de realizar tarefas repetitivas com precisão e agilidade, e aumentando a eficiência em linhas de montagem. Mais recentemente, com a consolidação de tecnologias da Indústria 4.0 [Lasi et al., 2014], a integração de técnicas de visão computacional fortaleceu ainda mais esses sistemas robóticos, permitindo que se adaptem a ambientes em mudança, abrindo caminho para operações industriais mais flexíveis e eficientes. Equipados com uma série de sensores, atuadores e algoritmos de controle, os manipuladores robóticos modernos possuem capacidades de precisão e repetibilidade excepcionais, permitindo-lhes lidar com tarefas complexas que eram consideradas difíceis ou até mesmo impossíveis para os métodos tradicionais ou manuais. Setores como a manufatura automotiva, montagem de eletrônicos, indústria farmacêutica e processamento de alimentos integraram manipuladores robóticos para otimizar linhas de produção, aprimorar o controle de qualidade e otimizar a utilização de recursos. À medida que avanços contínuos refinam suas capacidades, os manipuladores robóticos estão posicionados para permanecer na vanguarda da inovação industrial, impulsionando maior eficiência, precisão e versatilidade em uma infinidade de aplicações.

Com isso, robôs manipuladores têm sido utilizados também em tarefas de inspeção, aliados a sistemas de visão de máquina. Nesse tipo de aplicação, um sensor de visão (como câmeras, *lasers* e sensores infravermelho) é utilizado para capturar informações sobre

a superfície ou volume do objeto. O robô possui a função de posicionar o sensor para inspecionar diferentes partes do objeto.

A visão de máquina [Ren et al., 2022] é uma tecnologia que combina conceitos de ótica, visão computacional, e análise de imagens com o objetivo de extrair informações das imagens de forma automatizada. O tipo de informação extraída varia de acordo com o tipo de aplicação, como a detecção de defeitos em peça, análise dimensional, e identificação da posição e orientação de objetos. Por ser um método de detecção sem contato, a visão de máquina pode ser empregada em aplicações de automação inteligente de processos delicados e que exigem um controle preciso. Além disso, essa tecnologia permite a captação de uma ampla faixa espectral, podendo empregar, por exemplo, câmeras termais que capturam a radiação infravermelha emitida por objetos em ambientes com pouca ou nenhuma iluminação.

O setor automotivo foi um dos primeiros a utilizar extensivamente a visão de máquina [Shafi, 2004], sendo essa atualmente uma das técnicas mais utilizadas na indústria para soluções de inspeção, monitoramento, controle de qualidade e automação de processos devido ao alto nível de flexibilidade e repetibilidade que proporciona.

A integração de técnicas de visão computacional ao controle de robôs manipuladores fez surgir o controle *servo visual* [Hutchinson et al., 1996]. *Features* visuais das imagens podem como pontos, linhas e regiões podem ser utilizadas para, por exemplo, alinhar o efetuador de um manipulador em relação a um objeto. Assim, a visão se torna parte de uma malha de controle com *feedback* visual sobre o ambiente onde o robô está realizando a tarefa. Com isso, é possível realizar o replanejamento em tempo real de uma tarefa de um manipulador robótico, que normalmente é composta por uma série de movimentos pré-definidos executados em malha aberta, utilizando as informações visuais obtidas pelo processamento de imagens. A utilização da técnica de *Position Based Visual Servo*, ou Servo Visual Baseado em Posição, é uma solução proposta nesta dissertação para realizar o replanejamento de tarefa de um robô manipulador para inspeção visual de uma peça automotiva em uma linha de produção industrial. Devido ao grande volume de produção neste setor, a ocorrência de falhas devido ao posicionamento incorreto de uma peça pode afetar a linha de produção como um todo.

1.1 Motivação

O principal objeto de estudo desta dissertação é uma célula de inspeção de peças automotivas, desenvolvida pela empresa Invent Vision, situada no Parque Tecnológico de Belo Horizonte (BH-TEC), um condomínio de empresas e centros de pesquisa, fundado por uma parceria entre a Universidade Federal de Minas Gerais, Governo de Minas Gerais, Prefeitura Municipal de Belo Horizonte, SEBRAE e FIEMG.

Figura 1 – Célula de inspeção com robô Kuka KR4 R600.

A célula de inspeção pode ser vista na Figura 1. Ela é composta por um robô KR4 R600, da Kuka Robotics, com uma câmera industrial Invent Vision v200 de 5MP, e um conjunto de refletores montado em seu efetuador. A célula possui um sistema de automação de segurança, comandado por um CLP Siemens S7-1200, e um computador com sistema operacional Windows 10 que executa a plataforma de inspeção e controle Inspecta (Figura 3). A comunicação entre o CLP e a plataforma de inspeção é realizada via protocolo Modbus, e o CLP envia comandos para o robô por meio de uma rede Profinet.

Esta célula é aplicada no mercado automotivo, sendo totalmente integrada ao processo produtivo da linha de produção, com o intuito de garantir o controle de qualidade de toda a produção. A célula realiza a inspeção de peças de tamanhos e geometrias diferentes, desde caixas de porta luvas, até painéis e para-choques. Para realizar a inspeção, a célula conta com diferentes algoritmos para detectar ausência e presença de componentes, determinar se a cor da peça está correta, identificação de falhas na injeção do plástico, manchas e trincas.

Um sistema de visão industrial tipicamente é composto por três módulos: iluminação, aquisição de imagem, e um *software* para processamento e análise de imagens. Em primeiro lugar, é preciso realizar o projeto de uma plataforma que seja capaz de iluminar o objeto que será inspecionado, baseado nas características e requerimentos de inspeção do produto. Para a aquisição de imagem, câmeras digitais ou outro tipo de *hardware* de

aquisição de imagem são empregados para converter a luz refletida pelo objeto em imagens e transmití-las para um computador. A qualidade das imagens é crucial para o processo de detecção das falhas, portanto, é de grande importância que o sistema de iluminação forneça as condições necessárias para a detecção das falhas. Por último, um computador executa algoritmos de processamento de imagens baseado em abordagens clássicas ou de aprendizado de máquina, com o objetivo de extrair *features*, localizar, segmentar e classificar as imagens capturadas pela câmera. Por meio dessas etapas, um computador é capaz de realizar a análise automatizada das imagens em uma linha de produção de alto volume.

Além dos componentes típicos de um sistema de visão, a célula possui um berço, que é uma peça fabricada sob medida para cada máquina, a partir da geometria da peça que é inspecionada, cujo objetivo é servir de suporte para que ela seja corretamente encaixada no interior da célula na posição esperada. A fabricação do berço é um processo delicado e trabalhoso, pois ele é o responsável por garantir que o objeto inspecionado esteja sempre na posição correta durante a inspeção. A Figura 2 mostra o berço projetado para receber um para-choque dianteiro. Em uma operação típica, a peça é colocada no interior da célula e encaixada no berço, e o ciclo de inspeção é iniciado ao pressionar uma botoeira. O ciclo consiste em uma sequência de poses de inspeção pré-definidas, armazenadas na memória do robô, que são visitadas pelo robô, e fornecem uma boa visão da câmera para a captura de regiões de interesse. Cada imagem é analisada por meio de algoritmos de visão computacional, e aprovada ou reprovada com base nos resultados desta análise.

Em geral, esse tipo de sistema é satisfatório quando o objeto a ser analisado possui uma geometria simples e toda a região de inspeção pode ser capturada na cena. Por outro lado, quando são inspecionadas peças com geometria mais complexa, com concavidades e regiões de inspeção com diferentes orientações, a utilização de câmeras fixas se torna uma limitação. Uma opção que possibilita a escalabilidade de um sistema de visão de máquina para a inspeção de objetos cada vez mais complexos é a utilização de um robô manipulador, com a câmera montada em seu efetuador, permitindo o posicionamento da câmera em diferentes poses dentro do espaço de trabalho do robô. Essa configuração, denominada *eye-in-hand*, também já foi implementada com sucesso para a inspeção de objetos usando um *Scanner 3D* [Kuts et al., 2016] e na inspeção de peças de aeronaves utilizando sondas ultrassônicas [Mineo et al., 2015].

Na célula de inspeção robotizada, uma tarefa de inspeção é definida como uma sequência de configurações que deve ser realizada pelo robô. Em cada uma dessas configurações, a câmera deve capturar uma nova imagem da peça, que é enviada para a plataforma de inspeção via protocolo TCP/IP e processada em tempo real. Nessa etapa, algoritmos de visão computacional são utilizados para detectar possíveis problemas de manufatura da peça, como a ausência de parafusos, mau posicionamento de presilhas de

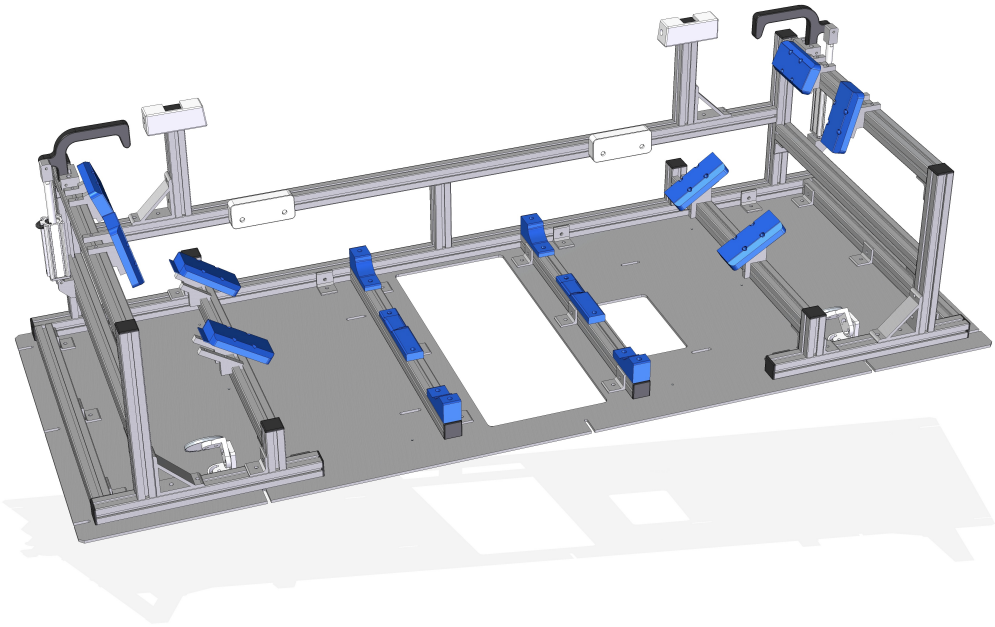


Figura 2 – Berço de uma célula de inspeção de para-choque dianteiro.

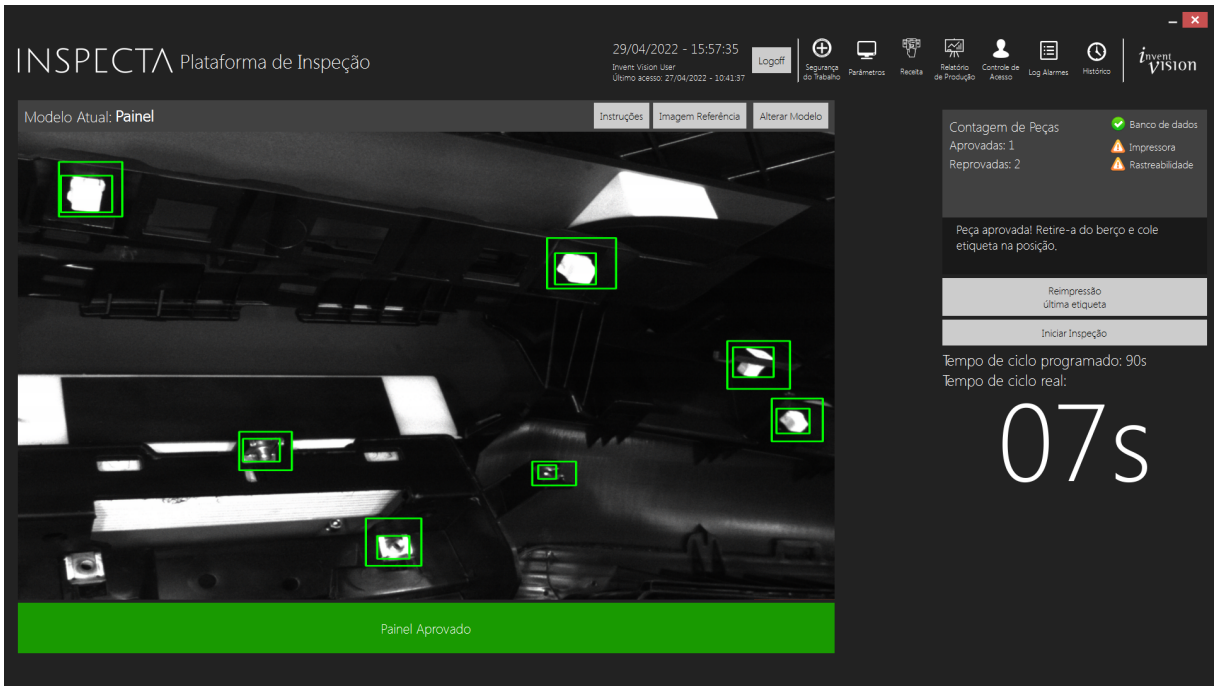


Figura 3 – Interface de inspeção da plataforma *Inspecta*.

encaixe, alterações de cor, e falhas de injeção. Tais problemas podem incorrer em uma falha crítica na peça, como por exemplo, na linha de fragilização de um *airbag*, que garante a ativação correta do dispositivo. O trabalho [Leao et al., 2022] detalha esse fluxo, e explora as diferenças dos tipos de movimento mais comuns em manipuladores robóticos, que são o movimento no espaço das juntas ($moveJ$), e o movimento linear no espaço Cartesiano ($moveL$).

Outro cenário de erro relativamente comum neste processo é a falha por mau posicionamento da peça dentro da célula. Essa situação pode ocorrer devido a desgastes no berço que impactam no encaixe da peça. Quando isso ocorre, é possível que as superfícies de interesse e componentes que devem ser inspecionados fiquem fora da área de busca dos algoritmos de inspeção, ou até mesmo não sejam capturados pela câmera, uma vez que o robô realiza uma trajetória pré-definida em malha aberta. Nesta dissertação é proposta uma melhoria neste fluxo de inspeção básico, utilizando as imagens da câmera para estimar a pose da peça e replanejar as poses de inspeção caso a peça não esteja posicionada no local adequado.

1.2 Objetivos

Dado o contexto do processo de inspeção automática de peças utilizando um manipulador robótico apresentado na Motivação, o objetivo geral desta dissertação é propor, implementar e validar um fluxo de inspeção que utilize informações visuais obtidas por uma câmera para replanejar os pontos de inspeção de uma peça. Entre os objetivos específicos desta dissertação, podem ser citados:

- Proposição de uma melhoria no fluxo da célula de inspeção robotizada;
- Implementação do fluxo de inspeção aprimorado utilizando informações visuais da peça;
- Construção e calibração de um arcabouço experimental utilizando um robô real em uma configuração *hand-in-eye* para validação do método proposto;
- Validação experimental do fluxo de inspeção proposto por meio de simulações e experimentos realizados com um robô real;
- Disponibilização dos algoritmos implementados, encapsulados em pacotes do ROS (*Robot Operating System*) [Quigley et al., 2009].

1.3 Contribuições

A principal colaboração desta dissertação é estabelecer uma melhoria de um processo industrial real utilizando técnicas de controle servo visual baseado em posição, oferecendo maior flexibilidade e repetibilidade de posicionamento.

A inspeção da peça na célula de inspeção Invent Vision está sujeita a erros oriundos do mau posicionamento da peça, que pode ocorrer devido a desgaste no berço. Foi proposto um novo fluxo de inspeção que contorna esse problema, utilizando as informações visuais da câmera para estimar a pose do objeto e então recalcular as poses de inspeção da tarefa com relação ao objeto. Dessa forma, a câmera é capaz de capturar a cena esperada pelos algoritmos de visão computacional responsáveis pela detecção das falhas. O novo fluxo de inspeção abrange o ciclo completo, assumindo que a peça pode estar em qualquer local dentro do espaço de tarefa, eliminando o problema de mau posicionamento da peça e eliminando o uso do berço.

Além disso, os resultados dos estudos e desenvolvimentos realizados durante o mestrado foram publicados em congressos:

- [Leao et al., 2022] L. G. Leao, G. M. Freitas, L. F. E. Moreira e A.O. Fernandes. *Aplicação de um manipulador robótico para automatização de uma célula de inspeção de peças automotivas. XXIV Congresso Brasileiro de Automática - CBA 2022.* Neste artigo, é apresentado o processo de automatização da célula de inspeção utilizando um robô manipulador. É feito um estudo comparativo das principais técnicas de controle de robôs manipuladores quando inseridas no contexto do processo produtivo em questão.

- (Aceito para publicação) L. G. Leao, G. M. Freitas, L. F. E. Moreira e A.O. Fernandes. *Replanejamento de Movimento Baseado em Visão da Tarefa Realizada por um Manipulador Robótico em uma Célula de Inspeção de Peças Automotivas. Simpósio Brasileiro de Automação Inteligente - SBAI 2023.* Neste artigo, é apresentado o fluxo de inspeção com replanejamento de movimento do robô manipulador na tarefa de inspeção de peças automotivas, utilizando controle servo visual baseado em posição, a partir da detecção e estimação da pose de uma peça por meio de marcadores fiduciais.

1.4 Estrutura da Dissertação

O restante desta dissertação está estruturado em mais 6 capítulos. No Capítulo 2, é feita uma revisão bibliográfica sobre a utilização de robôs manipuladores em processos industriais, visão de máquina, e utilização de técnicas de controle servo visual.

No Capítulo 3 são apresentados os fundamentos teóricos necessários para a compreensão do fluxo de inspeção proposto, como cinemática de corpos rígidos, modelagem de robôs manipuladores, formação e processamento de imagens, e controle servo visual.

O Capítulo 4 apresenta a metodologia empregada para implementar o fluxo de inspeção com replanejamento de tarefa proposto, as aplicações desenvolvidas para ROS, e descreve em detalhes todas as etapas do fluxo, bem como as etapas de calibração necessárias para a realização do mesmo.

No Capítulo 5, é apresentado o arcabouço experimental construído para a validação da metodologia proposta, incluindo ambientes de simulação e o *setup* experimental com robô real.

O Capítulo 6 apresenta os resultados obtidos nos experimentos em simulação e com robô real.

Por fim, o Capítulo 7 conclui a dissertação, e apresenta possíveis trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

O avanço tecnológico dos sistemas automotivos introduz um maior grau de complexidade na manufatura, e uma escala de produção cada vez maior, aumentando o potencial de ocorrência de erros de produção. Em um mercado competitivo, tais erros podem representar prejuízos significativos. Com isso, montadoras e seus fornecedores tem investido cada vez mais na garantia de qualidade na linha de produção, sendo os sistemas de visão de máquina uma das abordagens para a detecção de defeitos nos múltiplos processos de manufatura. Além disso, a expansão da garantia dos fabricantes exige um controle de qualidade com inspeções cada vez mais detalhadas. A quarta Revolução Industrial proporcionou diversos avanços tecnológicos para o setor de manufatura, como a automatização de processos por meio do uso de sistemas robóticos aliados a sistemas de visão de máquina inteligentes [Pérez et al., 2016].

Na indústria automotiva, sistemas de visão de máquina são empregados principalmente para realizar tarefas de inspeção, já sendo utilizada com sucesso para controle de qualidade de alimentos e produtos agrícolas [Patel et al., 2012], e para detecção de rachaduras em pontes [Oh et al., 2009]. Sensores de visão 2D ou 3D podem ser utilizados para realizar a detecção de falhas nos processos de montagem de componentes, recontrução 3D, injeção de plástico, soldagem, e pintura. Além disso, a visão de máquina pode ser empregada para o guiamento de robôs, para realizar o posicionamento de objetos (como portas e painéis) e *bin picking*.

2.1 Robótica e Visão de Máquina na Indústria

Sistemas de visão de máquina aliados a robôs manipuladores têm sido cada vez mais utilizados para a inspeção de objetos complexos.

Como exemplo, em [Mineo et al., 2015], um sistema com dois robôs de 6 graus de liberdade foi utilizado para inspecionar partes de aeronaves com superfícies complexas, utilizando sondas PAUT (*Phased Array Ultrasonic Testing*). Os dados adquiridos definem

Figura 4 – Configuração *eye-in-hand* para realização de tarefa relacionada a objeto. Fonte: Alenyà, 2014

o perfil da superfície da peça, que são analisados para detectar possíveis defeitos.

Esses sistemas também são amplamente utilizados na indústria para inspeção e controle de qualidade [Labudzki and Legutko, 2011], em aplicações que proporcionam maior segurança a trabalhadores do meio industrial, e como sistemas de percepção para posicionamento de robôs [Wöhler, 2012]. Robôs dependem desses sistemas de percepção para se mover dentro de um espaço enquanto desviam de obstáculos, obter maior precisão de posicionamento, e realizar uma tarefa de inspeção.

Dependendo do objetivo, o sistema de visão de um robô pode ser relacionado à cena ou objeto, exemplificados em [Alenyà et al., 2014]. Em tarefas relacionadas à cena, uma câmera é montada em um robô móvel para realizar mapeamento, localização, e desvio de obstáculos. Em tarefas relacionadas à objetos, uma câmera é tipicamente montada no efetuador de um robô manipulador (configuração *eye-in-hand*) para que imagens diferentes possam ser capturadas ao mudar a pose da câmera. A Figura 4 mostra uma aplicação típica de um robô nessa configuração, com uma câmera montada em seu efetuador, realizando a medição de clorofila em folhas.

A utilização de sistemas de visão para guiar robôs introduzem desafios complexos, essencialmente proporcionando olhos para uma máquina que é capaz de se mover com alta repetibilidade em ambientes industriais dinâmicos, com objetos em movimento e

humanos trabalhando. Nesse sentido o artigo [Hefele and Brenner, 2001] descreve um sistema de rastreamento *online* utilizando fotogrametria e um robô manipulador. A fotogrametria industrial cobre uma gama de desafios práticos em termos de acurácia desejada, velocidade de medição, integração de processos e custo-benefício. As principais soluções de fotogrametria estão relacionadas a medição de pontos discretos, deformações e deslocamento, determinação de parâmetros de 6 graus de liberdade, e identificação e inspeção de contornos e superfícies [Luhmann, 2010].

No artigo [Leikas, 1999], duas aplicações são escolhidas para demonstrar a acurácia, versatilidade e velocidade de sistemas robóticos guiados por visão: utilização de posicionamento 3D de objetos utilizados na identificação de pontos para a vedação precisa de um veículo, e a medição da largura de partículas em uma célula de moagem robotizada.

Sistemas de visão com câmeras estéreo também são bastante utilizados, especialmente em aplicações de *bin picking* [Rahardja and Kosaka, 1996], na qual um robô deve pegar um objeto conhecido em uma posição aleatória dentro de um recipiente. Esse tipo de aplicação possui desafios únicos. É necessário um algoritmo de identificação rápido e confiável que seja capaz de distinguir um objeto com características de cor e tamanho conhecidos dentre vários outros, independente de sua orientação e alinhamento. Além disso, essas informações devem ser obtidas e repassadas para o robô, que deve calcular trajetórias sem colisões para posicionar o efetuador e manipular o objeto identificado. O trabalho [Sturm et al., 2010] apresenta uma proposta para detectar, rastrear, e aprender modelos de articulação 3D de portas e gavetas usando um sistema de projeção de textura textura de visão estéreo. O robô usa modelos generativos para estimar o tipo de mecanismo, configuração atual e estimar a sua trajetória de abertura.

Câmeras *Time-of-Flight* (ToF) têm sido utilizadas com sucesso na reconstrução de superfícies e de objetos. Nessas aplicações, normalmente são necessárias múltiplas nuvens de pontos 3D em várias perspectivas que são combinadas. A configuração mais comum nesse tipo de sistema é a *eye-in-hand*, permitindo que o robô posicione o sensor nos locais desejados. Mesmo com pequenos deslocamentos, ainda é necessário realizar o registro das nuvens de pontos, utilizando um algoritmo como o *Iterative Closest Point* [Arun et al., 1987]. O artigo [Fuchs and May, 2008] apresenta um novo método para reconstrução de superfícies utilizando câmeras ToF, que simplifica a etapa de calibração. Com o método proposto, um objeto de referência foi reconstruído com precisão de 3mm de translação e 3º de rotação. No artigo [Maldonado et al., 2010], é apresentado um sistema autônomo utilizando um robô para fazer a manipulação de objetos não-modelados em um ambiente desconhecido, utilizando sinais de sensores na mão do robô para monitorar a tarefa. O sistema de percepção utiliza uma câmera ToF e algoritmos de otimização para determinar a pose de manipulação do objeto.

No contexto desta dissertação, sistemas de visão são tipicamente utilizados na

indústria automotiva, para a detecção de deformações na carroceria, inspeção de componentes, alinhamento de peças, e ajuste de ferramentas.

2.2 Controle Servo Visual

O controle servo visual [Corke, 1996] trata de como utilizar características visuais para guiar o movimento de um robô. Características visuais (*features*) da imagem como pontos, linhas e regiões podem ser usadas para, por exemplo, alinhar o efetuador a algum objeto da cena para realizar uma tarefa. Nesse caso, a visão é um aspecto ativo dentro da malha de controle, fornecendo informações acerca do ambiente em que o robô está. Uma das principais motivações para a incorporação de dados visuais nas malhas de controle é o aumento da versatilidade de sistemas robóticos. O trabalho [Hutchinson et al., 1996] introduz os conceitos de controle servo visual em forma de tutorial, e [Kragic et al., 2002] é uma pesquisa investigativa que busca unificar as diferentes nomenclaturas encontradas na literatura até o momento de sua publicação, além de propor uma taxonomia padrão para as estratégias de controle servo visual.

De maneira geral, o controle servo visual pode ser classificado com base em duas taxonomias principais, introduzidas em [Hutchinson et al., 1996]: **Controle Servo Visual Baseado em Posição** (*Position-Based Visual Servo - PBVS*) e **Controle Servo Visual Baseado em Imagem** (*Image-Based Visual Servo - IBVS*).

O PBVS utiliza as imagens para medir a pose de um alvo com respeito à câmera ou algum sistema de coordenadas global. Neste caso, os parâmetros da função de erro são poses, e as trajetórias do efetuador e da câmera são calculadas em coordenadas cartesianas. Entretanto, se o sistema for concebido com uma configuração *eye-in-hand*, as *features* utilizadas para estimação de pose podem sair da imagem. Isso pode ser resolvido tornando o sistema *endpoint closed loop*, no qual tanto o efetuador quanto o alvo são observados, utilizando uma câmera acoplada no efetuador do robô e uma câmera fixa. O trabalho [Allen et al., 1993], por exemplo, apresenta um sistema com uma câmera montada no efetuador e um conjunto par estéreo fixo que observa a cena. O sistema proposto pelos autores ataca três problemas comuns na robótica: cálculo rápido da pose 3D de alvos em imagens, controle preditivo de um braço robótico para rastrear um objeto em movimento, e interceptação de um alvo. Em [Wilson et al., 1996], é apresentada uma metodologia para estimar a pose de objetos no plano em uma configuração *eye-in-hand* para executar uma trajetória definida com respeito ao objeto.

Por outro lado, o IBVS utiliza as imagens para extrair um vetor de *features* $\mathbf{f}$, e o objetivo é mover a câmera de forma que esse vetor para uma pose desejada que gere um vetor de *features* desejado $\mathbf{f_d}$ [Hager et al., 1995]. Esse processo envolve o cálculo da Jacobiana de imagem, também chamada de matriz de interação, que relaciona as

velocidades da câmera com a taxa de variação das coordenadas da imagem [Hashimoto and Noritsugu, 1998]. O cálculo dessa matriz requer o conhecimento da informação de profundidade, que precisa ser atualizada constantemente enquanto a câmera se move. Essa restrição representa um dos maiores obstáculo na utilização desse método, pois a convergência não pode ser garantida caso a Jacobiana de imagem não seja atualizada. Alguns trabalhos propõem soluções para esse problema, como a estimação adaptativa de profundidade [Papanikolopoulos and Khosla, 1993]; determinação da profundidade a partir do conhecimento *a-priori* da relação entre as *features* ou utilizando uma estrutura de movimento caso seja possível medir o deslocamento da câmera [Longuet-Higgins, 1981].

Uma abordagem de controle servo visual 2 1/2D é apresentada em [Malis et al., 1998]. Esta é uma abordagem híbrida entre o PBVS e o IBVS, e evita suas respectivas desvantagens. A estimação da pose 3D de pontos no plano de imagem não é necessária, e o método garante a convergência da lei de controle em todo o espaço de tarefa. O método é demonstrado utilizando uma configuração *eye-in-hand*.

O controle servo visual também é utilizado em tarefas de *picking*, como em [che], onde um controlador servo visual é utilizado para controlar um robô manipulador para melhorar a acurácia no processo de pegar uma maçã em uma árvore. Em tarefas de *picking* como essa, a trepidação (*chattering*) é um grande problema de desempenho. Para lidar com isso, o controle de juntas do robô foi melhorado com um algoritmo de controle baseado em uma rede neural fuzzy adaptativa de modo deslizante.

A utilização dessa técnica não é limitada a robôs manipuladores, e também engloba aplicações de teleoperação, como é mostrado no trabalho [Xiao et al., 2017]. Nesse trabalho, um Veículo Aéreo Não Tripulado (VANT) substitui a função de operadores que realizam a leitura de medidores de radiação na usina nuclear de Fukushima Daiichi. A visualização nesse tipo de ambiente é limitado, aumentando o risco humano e perigo da tarefa. Para fazer as leituras dos medidores, o VANT é utilizado em modo teleoperado, de forma que os operadores podem permanecer em um ambiente seguro, e um controlador servo visual é utilizado para manter o ponto de interesse em uma pose constante dentro da cena.

2.3 *Features* visuais e Marcadores Fiduciais

Uma característica da imagem, ou *image feature*, corresponde a um valor escalar, extraído a partir de um ponto facilmente distinguível da imagem, como pontos ou linhas. Bons candidatos a *features* são aqueles que podem ser identificados inequivocamente em diferentes planos da imagem, como um furo ou um padrão artificial. A determinação da pose de um alvo na cena em relação a câmera pode ser estimada a partir das coordenadas das *features*. As *features* podem ser utilizadas dentro de uma malha de controle servo visual como o PBVS ou IBVS. Sendo assim, a escolha de *features* adequadas para o controle é

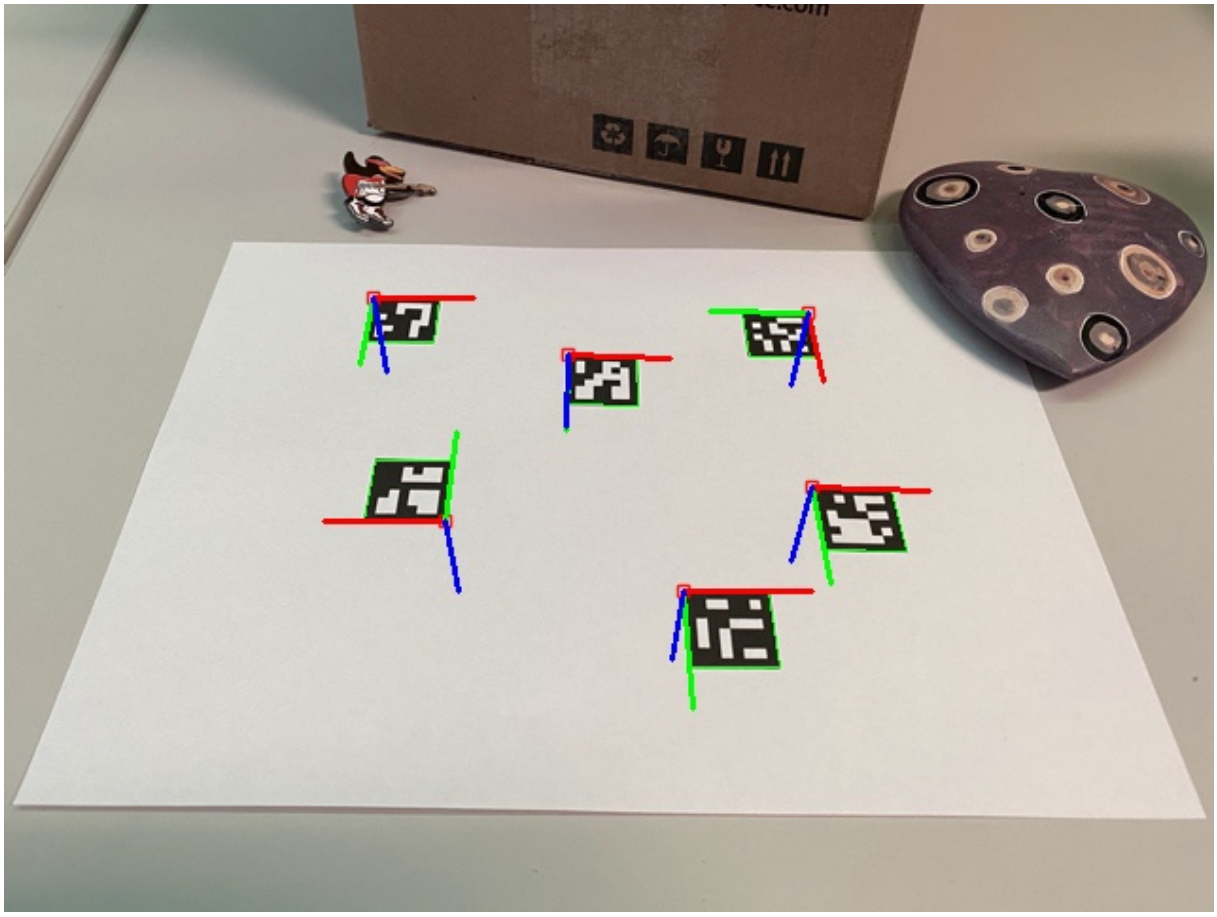


Figura 5 – Detecção e estimação de pose de marcadores ArUco. Fonte: Documentação OpenCV. Disponível em: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

importante.

Os marcadores fiduciais são padrões utilizados como sistemas de referências dentro de uma imagem. Existem vários tipos de marcadores fiduciais amplamente utilizados na visão computacional e robótica, e que oferecem diferentes tipos de codificação de dados e identificação. O ARToolkit [Kato and Billinghurst, 1999] foi um dos primeiros sistemas de marcadores fiduciais, sendo concebido para aplicações de realidade aumentada. A parte interna do marcador é quadrada com bordas pretas. O conteúdo codificado pelo ARToolkit são caracteres, como letras e números. Para decodificar esse tipo de marcador, o seu conteúdo é correlacionado com uma base de dados de marcadores conhecidos. Uma grande desvantagem desse método é o custo computacional necessário para decodificação, pois cada detecção requer um cálculo demorado de correlação.

O ARTag, apresentado por [Fiala, 2005] propõe algumas melhorias de codificação e detecção, utilizando um algoritmo baseado no gradiente da imagem, que o torna menos sensível a variações de iluminação, também sendo capaz de detectar marcadores

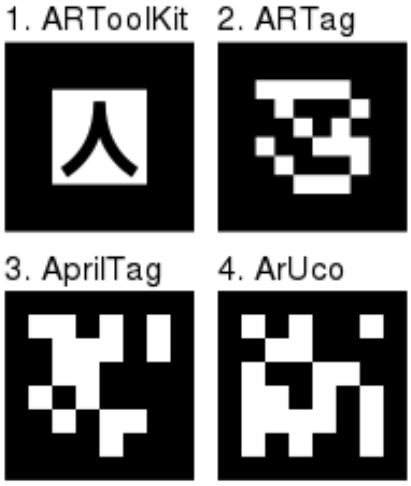


Figura 6 – Exemplos de marcadores fiduciais ARToolkit, ARTag, AprilTag e ArUco.

parcialmente ocludidos.

O trabalho de [Olson, 2011] apresenta o AprilTag, que é um sistema de marcadores fiduciais que possui algumas melhorias em relação aos sistemas anteriores, como um novo algoritmo de segmentação baseado em grafos e gradientes locais que permitem a estimação precisa de linhas. Também foram obtidos resultados melhores na acurácia da localização estimada em relação a sistemas anteriores. O sistema de codificação do AprilTag é mais robusto a rotações e falsos positivos.

O ArUco, por sua vez, é um marcador fiducial muito popular, pois a sua detecção é fácil, rápida e precisa, além de ser menos propenso a ocorrência de falsos positivos [Garrido-Jurado et al., 2014]. A Figura 5 mostra a detecção e estimação de poses a partir de um ArUco. Com a câmera montada no robô, é possível obter a pose do marcador em relação ao robô. Atualmente, há diversas bibliotecas abertas, como o OpenCV, que disponibilizam interfaces para a geração automática e detecção de ArUcos. Também existem dicionários de marcadores pré-definidos, com códigos prontos para uso.

A Figura 6 mostra exemplos dos marcadores fiduciais citados. De maneira geral, os marcadores fiduciais são capazes de fornecer um padrão de fácil detecção e uma medida altamente precisa de pose. Isso justifica a sua vasta utilização na robótica como forma de estimar a pose de locais de referência em um cenário. No caso desta dissertação, um ArUco é utilizado para estimar a pose da peça inspecionada pela célula, com o objetivo de realizar o replanejamento da tarefa a partir dessa estimativa.

Capítulo 3

Fundamentos Teóricos

Este capítulo apresenta a fundamentação teórica necessária para descrever o movimento de um robô manipulador, bem como a compreensão das técnicas de controle baseado em visão para manipuladores. A primeira seção apresenta a cinemática de corpos rígidos e como representar a pose de um corpo em relação a um sistema de coordenadas por meio de transformações homogêneas. Em seguida, a segunda seção apresenta a modelagem de robôs manipuladores seguindo a convenção de Denavit-Hatenberg, bem como define os conceitos de espaço de trabalho e de configuração, cinemática direta e diferencial. Na seção 3, é definido o erro de pose do efetuador. A seção 4 apresenta os fundamentos de formações de imagens por câmeras digitais. E por fim, a seção 5 apresenta os conceitos de controle servo visual que serão aplicados na metodologia proposta.

3.1 Corpos Rígidos e Transformações Homogêneas

Em mecánica clássica, um corpo rígido é um objeto idealizado em que a distância entre duas partículas que compõem o objeto permanece constante, mesmo com a ação de forças e torques no corpo. Ou seja, um corpo rígido é indeformável, independente das forças aplicadas sobre o mesmo. É assumido que a distribuição de massa em um corpo rígido é uniforme, portanto, é possível estudar o movimento de um corpo rígido de forma simplificada, considerando apenas o movimento de seu centro de massa, simplificando a modelagem matemática de sistemas físicos.

No contexto de manipuladores robóticos, a modelagem dos robô é feita pela união de uma cadeia de corpos rígidos, que são os elos, ou *links* do robô. As estratégias de controle e planejamento focam, particularmente, no movimento do efetuador, que é o último elo da cadeia cinemática. A representação desse movimento envolve a pose do efetuador, composta pela sua translação e rotação no espaço, bem como a sua velocidade linear e angular.

Para descrever a translação e rotação de um corpo rígido, primeiramente é neces-

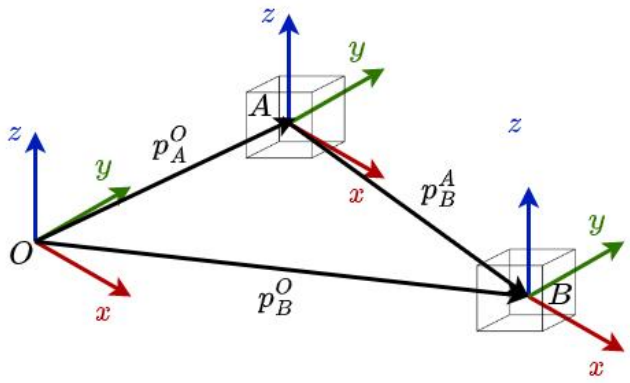


Figura 7 – Translações entre o corpo rígido A e o sistema de coordenadas O, entre o corpo rígido B e o sistema de coordenadas O, e entre o corpo rígido A e o corpo rígido B.

sário acoplar o corpo a um sistema de coordenadas local. Por ser um objeto indeformável, o corpo rígido pode ser considerado como um ponto localizado na origem do sistema de coordenadas. Dessa forma, é possível representar a translação de um corpo rígido A com relação a um referencial inercial O como um vetor $\mathbf{p_A^O} \in \mathbb{R}^3$ como na Equação 3.1:

$$\mathbf{p_A^O} = \begin{bmatrix} p_{A,x}^O \\ p_{A,y}^O \\ p_{A,z}^O \end{bmatrix}, \tag{3.1}$$

onde $p_{A,x}^O, p_{A,y}^O$ e $p_{A,z}^O$ representam as componentes da translação do corpo nos eixos x, yez com relação ao sistema de coordenadas $\mathbf{O}$. A Figura 7 mostra a translação entre diferentes corpos e um sistema de coordenadas inercial.

A orientação de um corpo rígido é definida pela rotação do corpo rígido com respeito a um sistema de coordenadas de referência. Para isso, é utilizada uma matriz de rotação que pertence ao grupo especial ortonormal de dimensão 3, ou seja, a rotação do corpo A com respeito ao sistema de coordenadas O pode ser representada por $\mathbf{R_A^O} \in SO(3)$, em que:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3x3} : \mathbf{R}\mathbf{R}^\mathbf{T} = \mathbf{I}, det(\mathbf{R}) = 1\}, \tag{3.2}$$

onde $\mathbf{I} \in \mathbb{R}^{3x3}$ é a matriz identidade.

Em robótica, existem outras formas de representação da orientação de um corpo rígido. Uma delas é a representação por ângulos de *roll, pitch* e *yaw*, que é dada por um conjunto de três ângulos de rotação em torno dos eixos x, yez do sistema de coordenadas inercial, como visto na Figura 8. A representação da orientação do sistema de coordenadas A com respeito ao sistema de coordenadas inercial O pode ser escrita como:

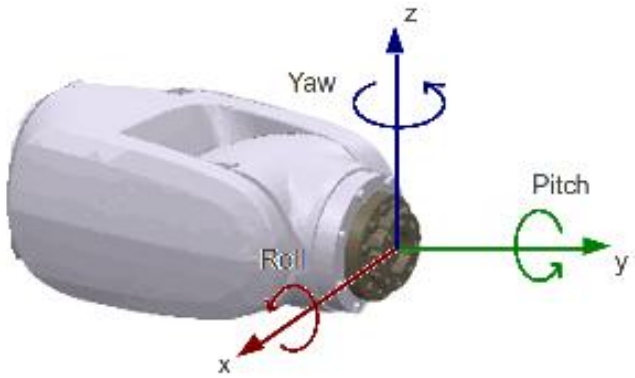


Figura 8 – Ângulos de *roll, pitch,* e *yaw* do efetuador de manipulador robótico.

$$\varphi_{\mathbf{A}}^{\mathbf{O}} = \begin{bmatrix} \phi_A^O \\ \theta_A^O \\ \psi_A^O \end{bmatrix}, \tag{3.3}$$

onde ϕ_A^O é o ângulo de *roll*, θ_A^O é o ângulo de *pitch* e ψ_A^O é o ângulo de *yaw*. A matriz de rotação $\mathbf{R}_{\mathbf{A}}^{\mathbf{O}}$ pode ser obtida pela composição das matrizes de rotação elementar $\mathbf{R}_z, \mathbf{R}_y$, e $\mathbf{R}_x$ desses ângulos:

$$\mathbf{R}_{\mathbf{A}}^{\mathbf{O}} = \mathbf{R}_z(\psi_A^O)\mathbf{R}_y(\theta_A^O)\mathbf{R}_x(\phi_A^O). \tag{3.4}$$

A representação mínima da pose de um corpo rígido pode ser escrita como:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_{\mathbf{A}}^{\mathbf{O}} \\ \varphi_{\mathbf{A}}^{\mathbf{O}} \end{bmatrix} \tag{3.5}$$

onde $\mathbf{x} \in \mathbb{R}^{6x1}$.

O quatérnio unitário é outra forma de representar a orientação de um corpo rígido. Quatérnios são extensões dos números complexos, na forma $Q = a + bi + cj + dk$. Uma rotação por um ângulo θ em torno do vetor unitário $n = (n_x, n_y, n_z)$ pode ser representada pelo quatérnio unitário $Q = (\cos\frac{\theta}{2}, n_x \sin\frac{\theta}{2}, n_y \sin\frac{\theta}{2}, n_z \sin\frac{\theta}{2})$. O quatérnio unitário pode ser escrito com a notação simplificada $Q = \{n, \theta\}$ separa a parte vetorial e a magnitude da rotação.

A transformação entre dois sistemas de coordenadas também pode ser representada por uma matriz de transformação homogênea $\mathbf{H}$. Essa matriz pertence ao Grupo Especial Euclidiano $SE(3) = \mathbb{R}^3 \times \mathrm{SO}(3)$ e condensa as informações de rotação e translação do corpo:

$$\mathbf{H} = \begin{bmatrix} \mathbf{R_{3x3}} & \mathbf{p_{3x1}} \\ \mathbf{0_{1x3}} & 1 \end{bmatrix}, \tag{3.6}$$

onde $\mathbf{0_{1x3}}$ é um vetor de zeros. Assim, é possível obter a transformação de um sistema de coordenadas B com respeito a um sistema de coordenadas O a partir da matriz de transformação homogênea de B com respeito a A:

$$\bar{\mathbf{p}}_{\mathbf{B}}^{\mathbf{O}} = \begin{bmatrix} \mathbf{R}_{\mathbf{A}}^{\mathbf{O}} & \mathbf{p}_{\mathbf{A}}^{\mathbf{O}} \\ \mathbf{0_{1x3}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{\mathbf{B}}^{\mathbf{A}} \\ 1 \end{bmatrix} = \mathbf{H}_{\mathbf{A}}^{\mathbf{O}} \bar{\mathbf{p}}_{\mathbf{B}}^{\mathbf{A}}, \tag{3.7}$$

onde $\bar{\mathbf{p}}_{\mathbf{B}}^{\mathbf{O}} = \begin{bmatrix} p_{B,x}^{O} & p_{B,y}^{O} & p_{B,z}^{O} & 1 \end{bmatrix}^T$ é o vetor de translação do sistema de coordenadas B com respeito ao sistema de coordenadas inercial, $\bar{\mathbf{p}}_{\mathbf{B}}^{\mathbf{A}} = \begin{bmatrix} p_{B,x}^{A} & p_{B,y}^{A} & p_{B,z}^{A} & 1 \end{bmatrix}^T$ é o vetor de translação do sistema de coordenadas B com respeito ao sistema de coordenadas A, e $\mathbf{H}_{\mathbf{A}}^{\mathbf{O}}$ é a matriz de transformação homogênea do sistema de coordenadas A com respeito ao sistema de coordenadas inercial.

3.2 Modelagem de um Robô Manipulador

Os manipuladores robóticos são dispositivos mecânicos reprogramáveis e multifuncionais que possuem a capacidade de mover objetos ou manipular diversas ferramentas para realizar diferentes tarefas. A modelagem desse tipo de robô é feita a partir da união de uma série de corpos rígidos, ou elos (no inglês, *links*), interconectados por juntas, que podem ser prismáticas ou rotativas.

Na indústria, existem diversos tipos de manipuladores robóticos, variando com a combinação de juntas. Alguns robôs comerciais disponíveis são apresentados na Figura 9. Dentre os mais comuns, podem ser citados os robôs com configuração cartesiana, cilíndrica, esférica e articulada. Diferentes tipos de configuração resultam em diferentes espaços de trabalho do robô, e a melhor configuração deve ser escolhida com base nos requisitos da tarefa a ser realizada. Essa escolha é feita levando em consideração o espaço de trabalho do robô e a sua interseção com o espaço em que a tarefa deve ser executada.

Uma grande parcela dos robôs na indústria possuem configuração articulada, com 6 juntas rotativas. Essa configuração permite controle de posição e orientação no espaço 3D, concedendo ao robô versatilidade para a realização de tarefas de natureza distinta, como soldagem e paletização. O escopo dessa dissertação está focado na modelagem e controle de manipuladores articulados com 6 juntas rotativas.

Figura 9 – Robôs comerciais comumente encontrados na indústria. Da esquerda para a direita: KUKA KR4 R600 (articulado), Epson Synthis T3 (SCARA), Yamaha XY-XC (cartesiano.

3.2.1 Espaço de trabalho e Espaço de Configuração

A **configuração** de um robô é a forma de descrever de forma completa a posição de cada ponto pertencente ao corpo do robô. No caso de robôs manipuladores os valores de posição de cada junta é conhecido por meio dos sistemas de medição e sensores embarcados. Assumindo que a base do robô é fixa, e os elos que conectam as juntas são corpos rígidos, é possível extrair a informação de localização de todos os pontos do robô a partir do valor de ângulo das juntas. O valor de ângulo da uma junta é dado por q_i. Para um manipulador com n juntas rotativas, sua configuração é dada por um vetor q, com os valores das juntas $q_1 \dots q_n$, onde $q_i = \theta_i$ é o valor do ângulo da junta q_i. No caso de juntas prismáticas, a posição da junta é dada por um valor de distância d_i. O conjunto de todas as configurações possíveis de um robô é chamado de **espaço de configuração**.

O **espaço de trabalho** de um robô é o volume total percorrido pelo efetuador ao executar todos os movimentos possíveis. Esse volume é determinado pela combinação das juntas, geometria do efetuador, e restrições mecânicas das juntas. Uma junta rotativa, por exemplo, pode estar limitada a posições angulares menores que 360^{o}. É assumido que cada junta possui um grau de liberdade. A Figura 10 mostra alguns tipos comuns de robôs e seus respectivos espaços de trabalho.

Em robótica, o número de graus de liberdade refere-se ao número de parâmetros independentes ou movimentos que um manipulador robótico pode fazer. Em outras palavras, é o número de maneiras que um robô pode se mover em um determinado espaço. Em geral, cada junta adiciona um grau adicional de liberdade ao movimento do manipulador. Ou seja, o número de juntas de um manipulador determina o número de graus de liberdade que ele possui. Um objeto rígido no espaço tridimensional possui 6 graus de liberdade: 3 para **posição** e 3 para **orientação**. Dessa forma, um manipulador robótico deve ter pelo

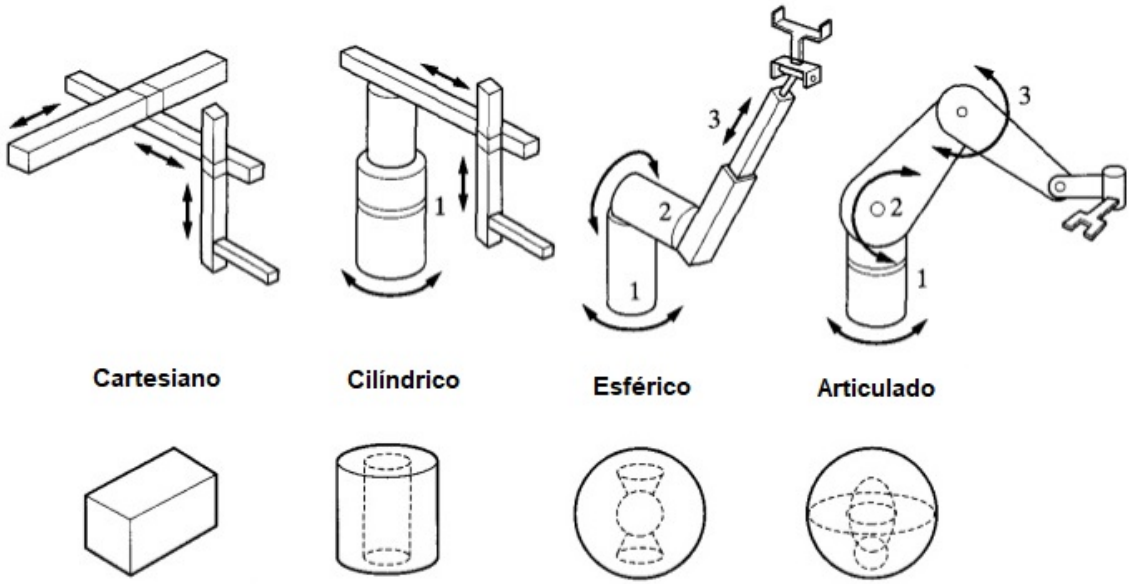


Figura 10 – Tipos de configuração de manipuladores robóticos e seus respectivos espaços de trabalho. Fonte: Adaptado de SB Niku, 2020.

menos 6 juntas independentes para ser capaz de posicionar seu efetuador em qualquer posição com orientação arbitrária dentro do espaço de tarefa, desde que os limites de juntas sejam respeitados. De forma geral, em tarefas de controle de posição e orientação em um espaço 3D, manipuladores com 6 juntas são classificados como **atuados**. Manipuladores com mais de 6 juntas são classificados como **redundantes** ou **sobre-atuado**. Por outro lado, manipuladores com menos de 6 juntas são classificados como **sub-atuados**. A Figura 11 mostra o modelo do robô manipulador articulado com 6 juntas rotativas.

3.2.2 Cinemática Direta e Convenção de Denavit-Hatenberg

A **cinemática direta** consiste na determinação da posição e orientação do efetuador a partir do valor dos ângulos das juntas. Um manipulador com n juntas possui $n+1$ elos, pois cada junta é responsável pela união de dois elos. Nessa convenção, a junta i une o elo $i-1$ com o elo i, e quando a junta i é atuada, o elo i se movimenta. Dessa forma, o elo 0 (tipicamente chamado de base) é fixo, pois não é atuado por uma junta. Cada junta q_i possui um valor de ângulo θ_i.

O primeiro passo para a análise cinemática de um manipulador consiste no acoplamento de um sistema de coordenadas $o_i x_i y_i z_i$ em cada elo i, de forma que ao sofrer uma atuação devido ao movimento do elo i, o sistema de coordenadas $o_i x_i y_i z_i$ também sofre um movimento correspondente. O sistema de coordenadas da base $o_0 x_0 y_0 z_0$, em

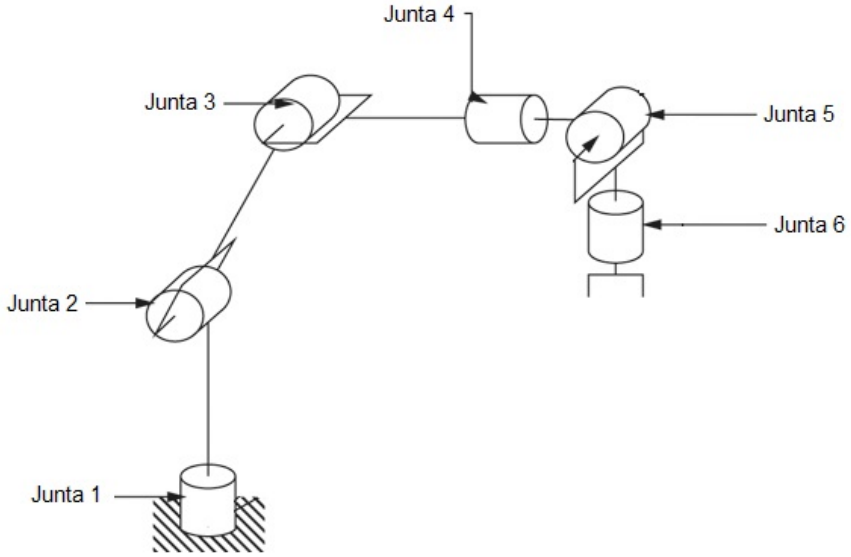


Figura 11 – Modelo do manipulador robótico articulado com 6 juntas.

especial, é chamado de sistema de coordenadas inercial. A posição e orientação do sistema de coordenadas $o_i x_i y_i z_i$ e o sistema de coordenadas $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$ pode ser representada por uma matriz de transformação homogênea $\mathbf{H_i}$. Essa matriz não é constante, e varia de acordo com a posição da junta i:

$$\mathbf{H}_i = \mathbf{H}_i(q_i). \tag{3.8}$$

A matriz de transformação homogênea $\mathbf{H_n^0}$ define a transformação de coordenadas do efetuador com respeito ao sistema de coordenadas inercial. Para obter $\mathbf{H_n^0}$, é preciso realizar a multiplicação das matrizes de transformação homogênea dos elos em sucessão:

$$\mathbf{H_n^0} = \mathbf{H_1}(q_1) \dots \mathbf{H_n}(q_n), \tag{3.9}$$

A matriz $\mathbf{H}_i$ possui forma:

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{R}_i^{i-1} & \mathbf{p}_i^{i-1} \\ 0 & 1 \end{bmatrix}, \tag{3.10}$$

portanto, a transformação entre elos intermediários do manipulador é dada por:

$$\mathbf{H}_j^i = \mathbf{H}_{i+1} \dots \mathbf{H}_j = \begin{bmatrix} \mathbf{R}_j^i & \mathbf{p}_j^i \\ 0 & 1 \end{bmatrix}, \tag{3.11}$$

onde $\mathbf{R}_j^i$ e $\mathbf{p}_j^i$ são, respectivamente, a matriz de rotação e o vetor de translação de $o_j x_j y_j z_j$ com respeito a $o_i x_i y_i z_i$.

Com isso, é possível determinar a cinemática direta de um robô maniplador conhecendo as matrizes H_i e multiplicando-as em sequência para determinar a pose do efetuador. Entretanto, determinar $H_i(q_i)$ pode ser uma tarefa complexa. Quanto mais juntas um manipulador possui, mais complexa se torna a sua análise cinemática. Por conta disso, existem convenções que simplificam e padronizam esse procedimento.

Uma das convenções mais utilizadas de representar as transformações ao longo dos elos é a convenção de Denavit-Hartenberg [Spong et al., 2006]. Seguindo essa convenção, a matriz de transformação homogênea entre os os sistemas de coordenadas do elo presente com respeito ao anterior pode ser obtida pela Equação 3.12:

$$\mathbf{H_i}(q_i) = Rot_{z_{i-1}}(\theta_i) Trans_{z_{i-1}}(d_i) Trans_{x_i}(a_i) Rot_{x_i}(\alpha_i), \tag{3.12}$$

sendo θ_i o ângulo da junta, a_i o tamanho do elo, α_i o deslocamento angular do elo, e d_i o deslocamento do elo.

A convenção de Denavit-Hartenberg estabelece um procedimento padrão para acoplar os sistemas de coordenadas a cada elo de um manipulador. O primeiro passo é escolher a direção do eixo z_i. A direção de z_i é escolhida de forma que coincida com a direção de atuação do eixo $i+1$, ou seja, a direção de z_0 é a direção de atuação da junta 1, a direção de z_1 é a direção de atuação da junta 2, e assim por diante. O próximo passo é definir o sistema de coordenadas da base $o_0 x_0 y_0 z_0$. A origem o_0 pode ser colocada em qualquer ponto ao longo do eixo z_0, e os eixos x_0 e y_0 podem ser escolhidos arbitrariamente, desde que o sistema de coordenadas siga a regra da mão direita.

Após definir o sistema de coordenadas da base, é feito um processo iterativo para cada sistema de coordenadas subsequente, utilizando o sistema $i-1$ para definir o sistema i. A convenção define duas regras que devem ser seguidas para estabelecer os sistemas de coordenadas:

1. O eixo x_i deve ser perpendicular ao eixo z_{i-1}.
2. O eixo x_i deve interceptar o eixo z_{i-1}.

Para atribuir os sistemas de coordenadas, são considerados três possíveis casos:

1. z_{i-1} **e** z_i **não são coplanares**: nesse caso, existe um único segmento de reta perpendicular a z_{i-1} e z_i que conecta os dois eixos. Essa linha define x_i e o ponto de interseção com z_i é a origem o_i. O eixo y_i deve ser escolhido de forma que o sistema de coordenadas i siga a regra da mão direita.

2. z_{i-1} **e** z_i **são paralelos**: quando z_{i-1} e z_i são paralelos, existem infinitos segmentos de reta normais que ligam os dois eixos. Nesse caso, a origem o_i pode ser escolhida em qualquer ponto ao longo de z_i. Uma prática comum é escolher o_i é escolher a normal que intercepta o_{i-1} como o eixo x_i. Assim, o_i é o ponto no qual a normal intercepta z_i. Com x_i e z_i fixados, y_i é escolhido de forma que o sistema de coordenadas siga a regra da mão direita. Nesse caso, d_i e α_i são 0.

3. z_{i-1} **intercepta** z_i: Nesse caso x_i é escolhido como a normal ao plano formado por z_{i-1} e z_i. O ponto de interseção entre z_i e z_{i-1} é escolhido como o_i. Novamente, y_i é escolhido de forma que o sistema de coordenadas obedeça a regra da mão direita.

Com os sistemas de coordenadas devidamente atribuídos a cada elo, é possível extrair todos os parâmetros necessários para construir as equações de transformação 4.31:

- a_i: distância entre z_{i-1} e z_i ao longo de x_i.
- α_i: ângulo entre z_{i-1} e z_i ao redor de x_i.
- d_i: distância entre x_{i-1} e x_i ao longo de z_{i-1}.
- θ_i: ângulo entre x_{i-1} e x_i ao redor de z_{i-1}.

É importante notar que, para o caso de um robô apenas com juntas rotativas os valores de α_i, a_i, e d_i são constantes, pois são inerentes à construção do robô, e a única variável é o ângulo das juntas θ_i.

3.2.3 Cinemática Diferencial

A cinemática direta permite obter a pose $\mathbf{x}$ do efetuador a partir da configuração $\mathbf{q}$ do robô. A **cinemática diferencial**, por sua vez, relaciona a velocidade do efetuador com a velocidade das juntas.

Definindo $\dot{\mathbf{x}} \in \mathbb{R}^{6\times 1}$ como o vetor composto pelas velocidades linear $\mathbf{v_e}$ e angular $\omega_\mathbf{e}$ do efetuador, a equação de cinemática diferencial pode ser utilizada para calcular $\dot{\mathbf{x}}$ através das velocidades das juntas $\dot{\mathbf{q}}$:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v_e} \\ \omega_\mathbf{e} \end{bmatrix} = \begin{bmatrix} \mathbf{J_P}(\mathbf{q}) \\ \mathbf{J_O}(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}}, \tag{3.13}$$

onde $\mathbf{J_P}(\mathbf{q}) \in \mathbb{R}^{3\times n}$ é a matriz Jacobiana que relaciona a contribuição das velocidades das juntas para a velocidade linear do efetuador, e $\mathbf{J_O}(\mathbf{q}) \in \mathbb{R}^{3\times n}$ é a matriz Jacobiana que relaciona a contribuição das velocidades das juntas para a velocidade angular do efetuador. Com isso, é possível construir a matriz Jacobiana geométrica $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{6\times n}$:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{J_P} \\ \mathbf{J_O} \end{bmatrix}, \tag{3.14}$$

que depende da configuração atual do robô.

Para o caso particular de um robô que possui apenas juntas rotativas, e seguindo a convenção de Denavit-Hartenberg, uma coluna i da Jacobiana é calculada por:

$$\begin{bmatrix} \mathbf{J_{P_i}} \\ \mathbf{J_{O_i}} \end{bmatrix} = \begin{bmatrix} \mathbf{z^0_{i-1}} \times (\mathbf{p^0_n} - \mathbf{p^0_{i-1}}) \\ \mathbf{z^0_{i-1}} \end{bmatrix}, \tag{3.15}$$

3.3 Definição do Erro de Pose

Assumindo que cada junta do robô possui uma malha de controle interna que recebe como referência uma velocidade angular que deve ser atingida pela junta, e computa o torque necessário para garantir o seguimento de referência. Com um ganho alto, essa malha de controle é rápida o suficiente para garantir:

$$\mathbf{u} \approx \dot{\mathbf{q}}. \tag{3.16}$$

Com isso, $\dot{\mathbf{q}}$ pode ser usado diretamente como o sinal manipulado para o controle cinemático do robô.

O erro entre a pose atual do robô pose atual $\mathbf{x}$ e a pose desejada $\mathbf{x_d}$ pode ser definido por:

$$\mathbf{e} = \mathbf{x_d} - \mathbf{x}, \tag{3.17}$$

e a sua derivada:

$$\dot{\mathbf{e}} = \dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}. \tag{3.18}$$

O vetor erro $\mathbf{e} = \begin{bmatrix} \mathbf{e_P} & \mathbf{e_O} \end{bmatrix}^T$ é composto pelo erro de posição e orientação do efetuador. Para o erro de posição, temos a expressão:

$$\mathbf{e_p} = \mathbf{p_d}(\mathbf{t}) - \mathbf{p^0_n}, \tag{3.19}$$

com derivada:

$$\dot{\mathbf{e}}_\mathbf{p} = \dot{\mathbf{p}}_\mathbf{d}(\mathbf{t}) - \mathbf{v_e}. \tag{3.20}$$

Por outro lado, a expressão do erro de orientação depende na forma escolhida para representar a orientação do efetuador (ângulos de Euler; ângulo e eixo; ou quatérnio

unitário). Nesta dissertação, foi escolhida a representação por quatérnio unitário, cuja principal vantagem é evitar a ocorrência de representações singulares.

Sendo $\mathbf{R}_d$ e $\mathbf{R}_n^0$ as matrizes de rotação que representam, respectivamente, a orientação desejada e atual do robô, e $Q_d = \{\eta_d, \epsilon_d\}$ e $Q_e = \{\eta_e, \epsilon_e\}$ os quatérnios associados a $\mathbf{R}_d$ e $\mathbf{R}_n^0$. O erro de orientação pode ser obtido pelo quatérnio $\Delta Q = \{\Delta\eta, \Delta\epsilon\}$ que é calculado como:

$$\Delta Q = Q_d * Q_e^{-1}. \tag{3.21}$$

É possível mostrar que $\Delta Q = \{1, 0\}$ quando $\mathbf{R}_d$ e $\mathbf{R}_n^0$ estiverem alinhadas. Portanto, é suficiente adotar o erro de orientação como $\mathbf{e_O} = \mathbf{\Delta}\epsilon$.

3.4 Formação e Captura de Imagens em Câmeras Digitais

O processo de formação de imagem em uma câmera digital envolve várias etapas [Szeliski, 2022] que convertem a luz visível em uma imagem digital. A câmera escura (Figura 12) é a base de construção de qualquer câmera fotográfica. Neste modelo, também chamado de câmera pinhole, a luz refletida por um objeto passa por um pequeno orifício em uma das paredes da câmera, e é projetada na parede interna oposta da câmera, chamada de anteparo. Em uma câmera digital, a luz refletida é capturada por um sensor, que gera a imagem após um certo tempo de exposição. Quanto menor o tamanho do orifício, maior é a nitidez da imagem formada pela câmera. Entretanto, a diminuição do orifício também diminui a quantidade de luz que pode passar pelo mesmo, tornando a imagem mais escura e exigindo um maior tempo de exposição. A solução para este problema é a utilização de uma lente, que permite que mais luz seja capturada pela câmera mantendo a nitidez da imagem.

Imagens digitais são organizadas como uma matriz de pixels, como pode ser visto na Figura 13, onde cada pixel representa o valor de intensidade da luz em cada ponto da imagem. Em uma imagem 8-bit, por exemplo, cada pixel pode representar um valor de intensidade entre 0 e 255. Câmeras digitais normalmente adotam um sistema de detecção de cores no qual cada cor (vermelho, verde, ou azul) é representada por um canal. Nesse caso, cada pixel possui três canais, que representam a intensidade de cada cor naquele ponto específico da imagem.

Um dos métodos mais comuns para a detecção de cores é o filtro de Bayer. O filtro de Bayer consiste em um filtro matricial de cores RGB dispostas de forma alternada, e posicionado sobre a matriz de pixels. Além do espaço de cores RGB, existem diversos outros espaços de cores que podem ser utilizados no processamento de imagens digitais,

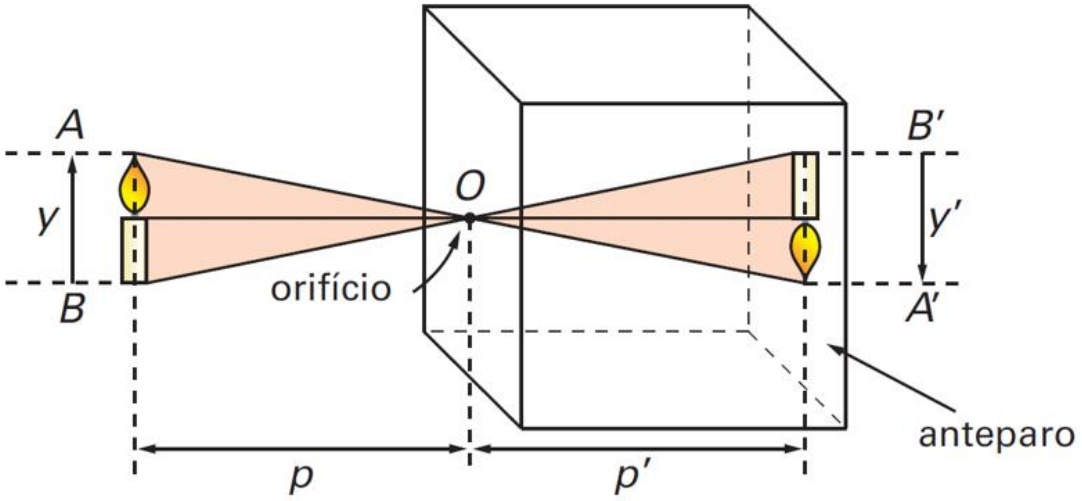


Figura 12 – Modelo de câmera escura.

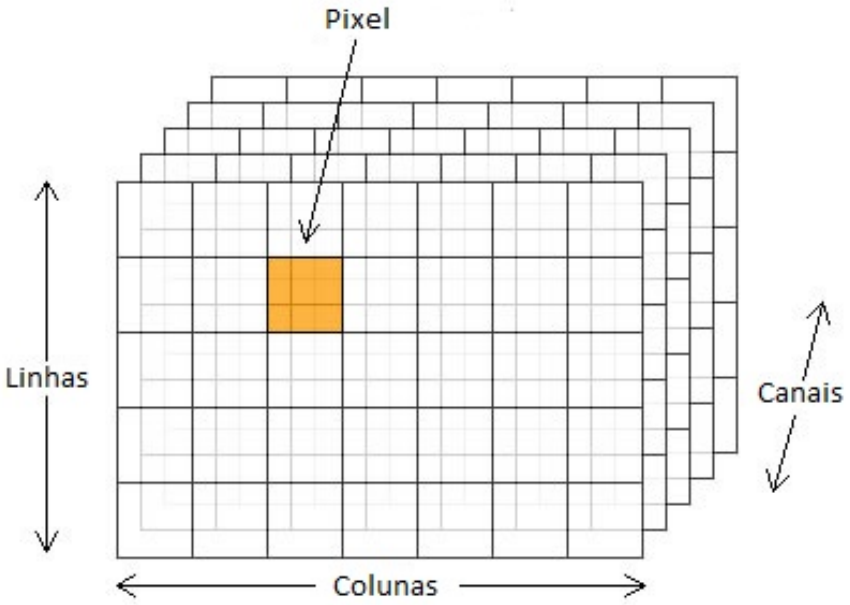


Figura 13 – Imagem digital.

como o YUV e o HSV (*hue, saturation, value*). Câmeras que possuem apenas um sensor tipicamente utilizam o filtro de Bayer para separar as cores. Por outro lado, também existem câmeras 3-CCD, que utilizam três sensores para detecção de cada canal de cor, separadas por um prisma. Essas câmeras produzem uma imagem com maior precisão na representação de cores.

O modelo da câmera pinhole possui parâmetros internos [Hartley and Zisserman,

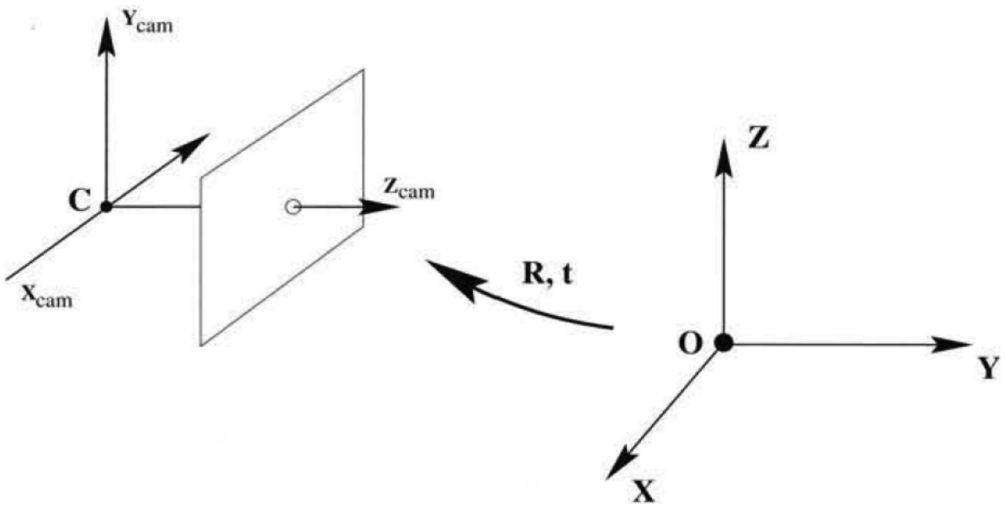


Figura 14 – Transformação entre o sistema de coordenadas da câmera e do mundo.

2003], chamados de parâmetros intrínsecos, que definem como a câmera captura imagens, como a distância focal e o centro óptico. Os parâmetros extrínsecos, por sua vez, relacionam o sistema de coordenadas da câmera com o sistema de coordenadas do mundo, descrevendo a posição e orientação da câmera no mundo 3D, como é mostrado na Figura 14.

Duas imagens da mesma superfície planar são relacionadas por uma matriz de homografia. Essa é uma matriz 3×3 que descreve a transformação entre os dois planos com um fator de escala. Dado um ponto $p_i = (x_i, y_i, z_i)$ sobre um plano π e um ponto correspondente $p'_i = (x'_i, y'_i, z'_i)$ sobre um plano π', o mapeamento entre os dois pontos pode ser dado pela matriz de homografia $\mathbf{G}$:

$$p'_i = \mathbf{G} p_i, \tag{3.22}$$

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{bmatrix}. \tag{3.23}$$

3.5 Processamento e Análise de Imagens

As imagens que são produzidas por uma câmera podem ser processadas, passando por operações de aprimoramento, transformação, redução e filtragem para extrair informações que podem ser úteis. Algoritmos de processamento possuem como entrada uma imagem, e como saída uma imagem transformada. Algoritmos de análise de imagens, por outro lado, possuem como entrada uma imagem, e como saída uma informação extraída

a partir da imagem, como as coordenadas de um ponto, vetores de descrição, regiões de interesse, entre outros.

O elemento básico de formação de uma imagem é o pixel. Cada pixel possui atributos que descrevem seu valor de intensidade, cor e localização, e representam características de objetos no mundo real. O agrupamento de pixels que representam uma informação ou parte de uma informação é chamado de *feature*, que podem ser extraídas a partir de técnicas de processamento de imagens e classificadas em relação a outras *features* com características similares (ex.: características de rostos de pessoas). Uma única *feature* possui uma infinidade de combinações que podem ser válidas dependendo de fatores como a pose do objeto, iluminação, e tempo de exposição, podendo ser descritas como um modelo matemático.

A identificação de *features* pode ser utilizada em algoritmos de inspeção, como o *template matching*, detecção de cores e detecção de linhas. O *template matching* [Brunelli, 2009] é um dos principais algoritmos utilizados na célula de inspeção da Invent Vision para controle de qualidade peças, e será o algoritmo adotado nesta dissertação. Sua utilização é simples e de implementação relativamente fácil, tornando-o um dos métodos mais populares de localização de objetos. Por outro lado, o processo de busca por amostras grandes e complexas pode ser demorado, portanto a aplicabilidade desse algoritmo está relacionada aos recursos computacionais disponíveis.

O propósito desse algoritmo é encontrar todas as localizações em uma imagem inspecionada (*input*) nas quais uma imagem de referência (amostra) pode ser encontrada. A Figura 15 mostra um exemplo que ilustra esse algoritmo.

A busca é realizada de uma maneira bem direta: a imagem de amostra é posicionada sobre a imagem *input* em cada local possível, e uma métrica de similaridade entre a amostra e a região de sobreposição da imagem de *input* é calculada em cada uma. A melhor estimativa da localização $[\hat{u}, \hat{v}]$ da amostra na imagem pode ser encontrada através de uma função correlação de covariância cruzada, maximizando o valor do coeficiente de correlação sobre todas as possíveis localizações:

$$[\hat{u}, \hat{v}] = argmax_{u,v}\rho_{12}(u, v) \tag{3.24}$$

A função $\rho_{12}(u, v)$ é uma métrica numérica de similaridade chamada de correlação normalizada entre a imagem g_1 e a amostra g_2, e é dada pela fórmula:

$$\rho_{12}(u, v) = \frac{\sigma_{g_1 g_2}(u, v)}{\sigma_{g_1}(u, v)\sigma_{g_2}}, \tag{3.25}$$

onde σ_{g_2} é o desvio padrão dos valores de intensidade de g_2:

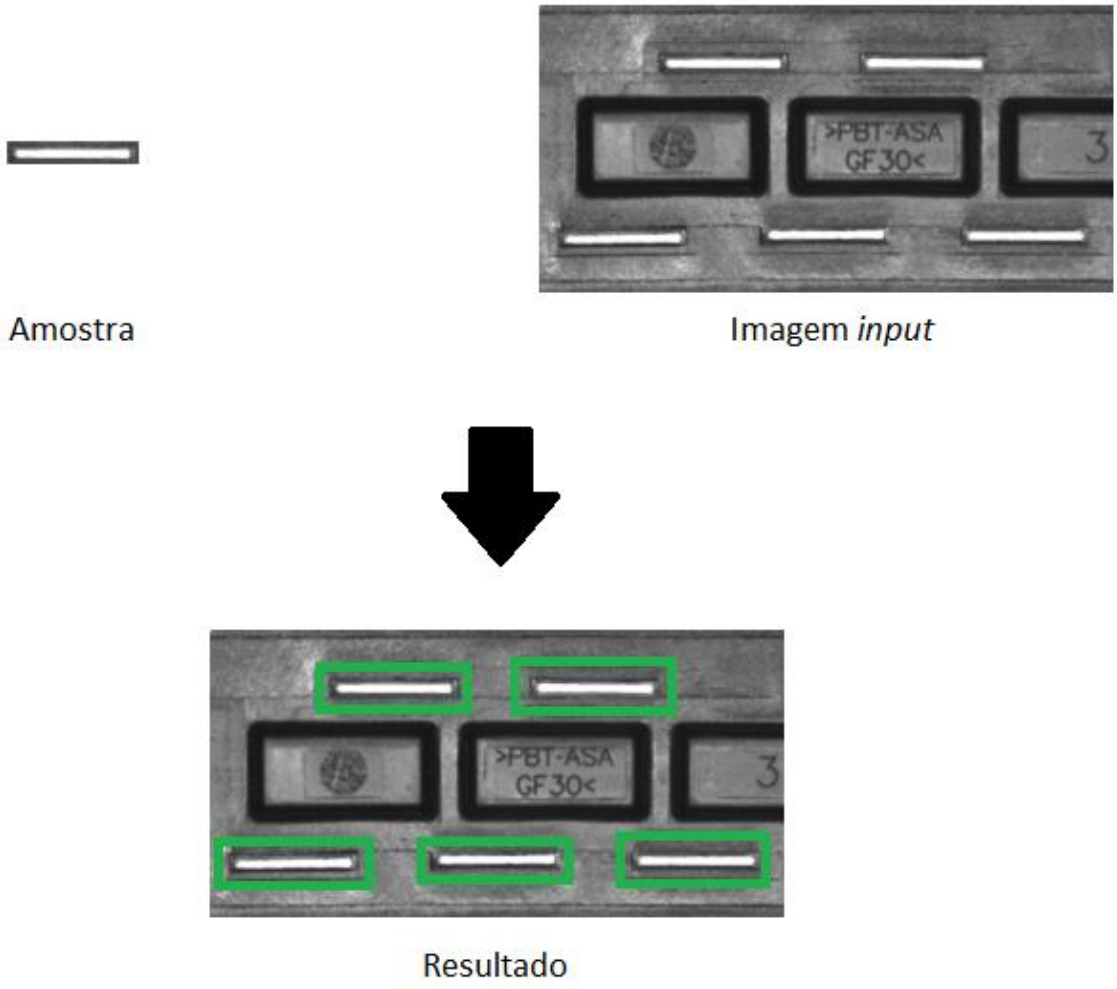


Figura 15 – Exemplo de *template matching*. Adaptado de *Adaptive Vision*, 2017

$$\sigma_{g_2}^2 = \frac{1}{M-1}\sum_{m=1}^{M}\left(g_2(i_m, j_m) - \frac{1}{M}\sum_{m=1}^{M} g_2(i_m, j_m)\right)^2, \tag{3.26}$$

onde M é o número de pixels de g_2.

O termo $\sigma_{g_1}(u, v)$ é o desvio padrão dos valores de intensidade de g_1 na área em sobreposição com g_2 na posição $[u, v]$, e é dado por:

$$\sigma_{g_1}^2 = \frac{1}{M-1}\sum_{m=1}^{M}\left(g_1(i_m - u, j_m - v) - \frac{1}{M}\sum_{m=1}^{M} g_1(i_m - u, j_m - v)\right)^2. \tag{3.27}$$

O termo $\sigma_{g_1 g_2}(u, v)$ representa a covariância entre os valores de intensidade na área de sobreposição de g_1 na posição $[u, v]$ e g_2:

$$\sigma_{g_1 g_2}(u,v) = \frac{1}{M-1}\sum_{m=1}^{M}\left[\left(g_2(i_m,j_m) - \frac{1}{M}\sum_{m=1}^{M} g_2(i_m,j_m)\right)\right. \\ \left.\cdot\left(g_1(i_m-u,j_m-v) - \frac{1}{M}\sum_{m=1}^{M} g_1(i_m-u,j_m-v)\right)\right] \tag{3.28}$$

3.6 Controle Servo Visual

O controle servo visual é um tipo de controle que utiliza informações visuais provenientes de uma câmera ou sensores de imagem como *feedback* para controlar a posição e orientação de um robô. No contexto de robôs manipuladores, é desejado controlar a pose do efetuador. Essa estratégia pode ser utilizada em diversas aplicações, como posicionar a câmera para inspeção de peças, manipular objetos em movimento numa esteira, fazer replanejamento de trajetórias em tempo real, ou manter um alvo centralizada no campo de visão da câmera.

A ideia básica por trás do controle servo visual é utilizar informações da imagem, como um conjunto de *features* detectadas para computar o erro entre a pose desejada e a pose atual do robô. Com *features* suficientes, é possível estimar a pose 3D de um objeto na cena, sendo possível posicionar o robô relativamente ao objeto para realizar uma tarefa.

A lei de controle é baseada no erro entra o vetor de *features* desejadas e o vetor de *features* extraído da imagem, sem envolver a estimação da pose do alvo. Uma grande vantagem dessa abordagem baseada em imagem é a robustez à erros de calibração da câmera, por não precisar realizar as transformações geométricas das poses dos objetos na cena. Os principais problemas do IBVS são explorados em [Chaumette, 2007]. Em particular, movimentos que envolvem grandes rotações podem gerar um efeito chamado de recuo de câmera. Além disso, a ocorrência de singularidades na matriz Jacobiana podem fazer com que o robô pare em mínimos locais. Uma abordagem híbrida proposta em [Corke and Hutchinson, 2000] busca resolver tais limitações.

No PBVS, as imagens são usadas para determinar a pose de um alvo com respeito à câmera, e uma lei de controle cartesiana é utilizada para guiar o robô até a pose desejada (17). Nesse caso, as *features* também são extraídas da imagem, e são utilizadas para realizar a estimação de pose. Por ser uma estratégia que utiliza informações 3D, está sujeita a erros de calibração do modelo da câmera, bem como erros do processo de estimação da pose do objeto.

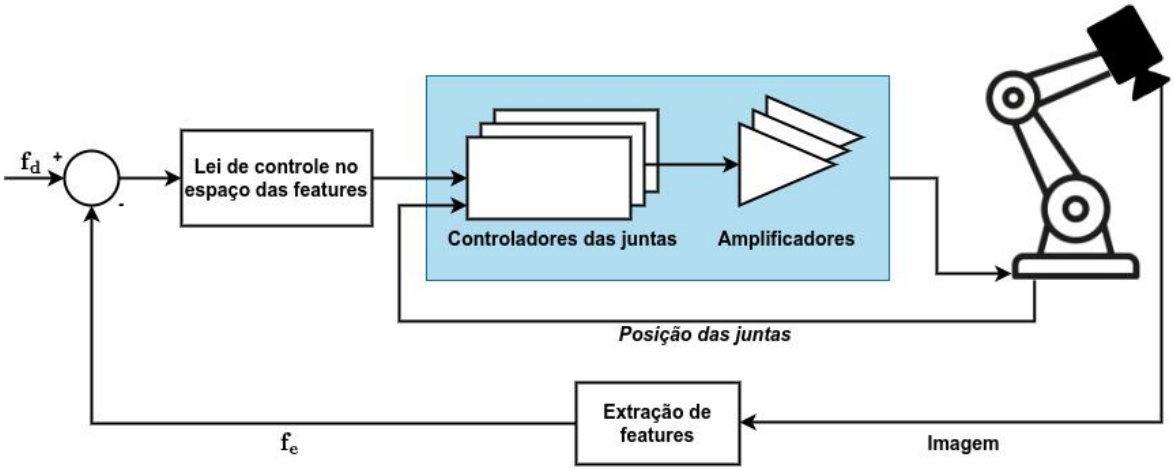


Figura 16 – Estrutura de controle baseada em imagem.

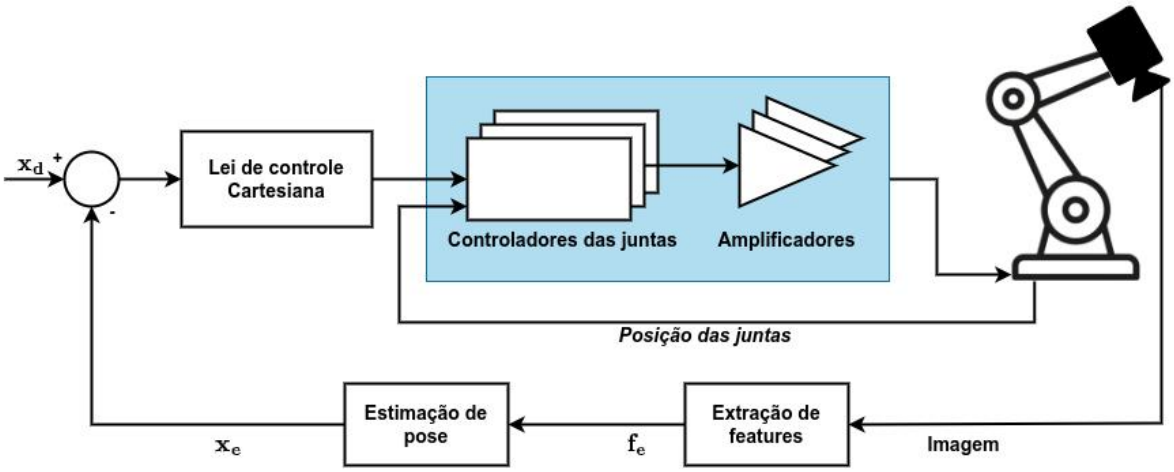


Figura 17 – Estrutura de controle baseada em posição.

Capítulo 4

Inspeção Automática de Peças por Controle Servo Visual

Este capítulo descreve a metodologia empregada para a implementação de um fluxo de inspeção automático de peças utilizando o controle servo visual baseado em posição. Inicialmente é apresentado a modelagem do robô utilizado seguindo a convenção de Denavit-Hartenberg. Em seguida, são apresentados os tipos de controle clássicos que são empregados para movimentação do robô. Também são apresentados os processos de calibração da câmera e calibração *hand-eye*. Por fim, o fluxo de inspeção automática proposto é apresentado, e cada etapa é detalhada, fazendo uma comparação com o fluxo original. São apresentadas as etapas de busca do marcador; a estimação e refinamento da pose do marcador; a definição das poses de inspeção e, por fim, a execução da inspeção.

4.1 Modelagem do Robô KUKA KR4 R600

O robô utilizado nesta dissertação é o KUKA KR4 R600. A finalidade da utilização deste robô é de manter a maior fidelidade possível com a célula de inspeção Invent Vision, que emprega o mesmo modelo. Todos os desenvolvimentos contidos neste capítulo e nos próximos são feitos levando em consideração este robô.

O KR4 é composto por seis juntas rotativas, conectados por elos de tamanho fixo, como é mostrado na Figura 18. Os parâmetros de Denavit-Hartenberg do KR4 foram extraídos a partir dos dados encontrados em seu *datasheet* [Kuka Robotics, 2021] e podem ser vistos na Tabela 1. É importante notar que três desses parâmetros, sendo inerentes à montagem do robô, são constantes, e a única variável é o ângulo da junta θ_i.

Tabela 1 – Parâmetros Denavit-Hartenberg do KR4 R600.

Junta	$\theta[rad]$	d[mm]	a[mm]	$\alpha[rad]$
1	$\theta_1 + \pi/2$	330	0	$\pi/2$
2	$\theta_2 - \pi/2$	0	290	0
3	θ_3	0	20	$\pi/2$
4	θ_4	310	0	-$\pi/2$
5	θ_5	0	0	$\pi/2$
6	θ_6	75	0	0

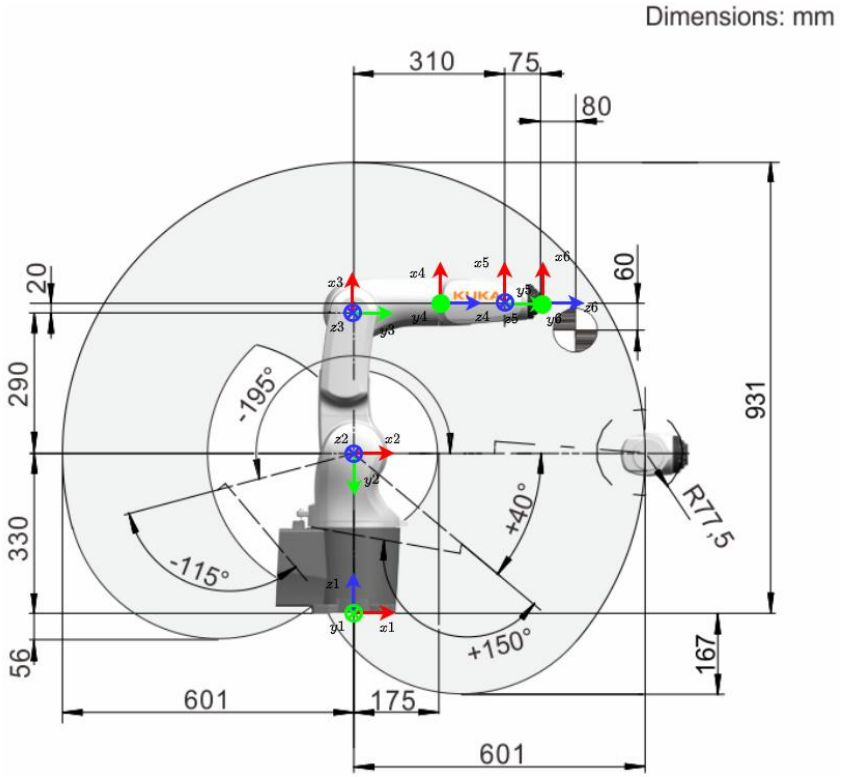


Figura 18 – Dimensões e *frames* de referência do KR4 R600. Adaptado de Kuka KR4 R600 Datasheet, 2021.

4.2 Técnicas de Controle de um Robô Manipulador

Tipicamente, robôs industriais comerciais possuem dois tipos básicos de movimento: ponto-a-ponto seguindo uma trajetória linear no espaço Cartesiano por meio do controle cinemático para seguimento de trajetória, e ponto-a-ponto no espaço das juntas. Tais movimentos são comumente conhecidos como *"moveL"* e *"moveJ"*, respectivamente.

4.2.1 Controle Cinemático para Seguimento de Trajetória

O objetivo do controle cinemático é garantir seguimento da trajetória $\mathbf{x} \longrightarrow \mathbf{x_d}(\mathbf{t})$, utilizando $\mathbf{u}$ como entrada para o sistema, seguindo o diagrama mostrado na Figura 19

[Sciavicco et al., 2011].

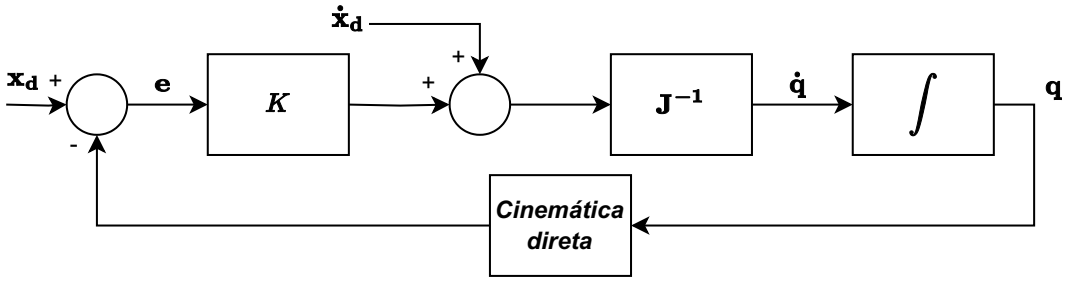


Figura 19 – Diagrama de blocos para do controle cinemático.

Substituindo a Equação 3.13 em 3.18, obtemos:

$$\dot{\mathbf{e}} = \dot{\mathbf{x}}_\mathbf{d} - \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{4.1}$$

Assumindo que $\mathbf{J}$ é quadrada e não-singular, a seguinte lei de controle pode ser escolhida:

$$\mathbf{u} = \mathbf{J}^{-1}(\mathbf{q})\bar{\mathbf{u}}, \tag{4.2}$$

que leva ao sistema linear:

$$\dot{\mathbf{e}} = \dot{\mathbf{x}_\mathbf{d}} - \bar{\mathbf{u}}. \tag{4.3}$$

Escolhendo $\bar{u}$ como:

$$\bar{\mathbf{u}} = \dot{\mathbf{x}}_\mathbf{d} + \mathbf{K}\mathbf{e}, \tag{4.4}$$

onde $\mathbf{K}$ é o termo proporcional ao erro e $\dot{\mathbf{x}}_\mathbf{d}$ corresponde a uma ação *feedforward*, resultando na lei de controle:

$$\mathbf{u} = \mathbf{J}^{-1}(\mathbf{q})(\dot{\mathbf{x}}_\mathbf{d} + \mathbf{K}\mathbf{e}). \tag{4.5}$$

A dinâmica do erro em malha fechada pode ser escrita como:

$$\dot{\mathbf{e}} + \mathbf{K}\mathbf{e} = \mathbf{0}, \tag{4.6}$$

onde $\mathbf{K} = k\mathbf{I}_{\mathbf{n}\times\mathbf{n}}$. Quando $k > 0$, o sistema é assintoticamente estável:

$$\mathbf{e}(t) = e^{-kt}\mathbf{e}(0), \tag{4.7}$$

com $\mathbf{e} \longrightarrow \mathbf{0}$ quando $t \longrightarrow \infty$.

Movimentos no espaço Cartesiano estão sujeitos ao risco de singularidades, que ocorrem quando a matriz Jacobiana tem posto menor que n. A presença de singularidades pode impactar negativamente a malha de controle, gerando sinais de controle e torques impraticáveis para as juntas, e impedindo o seguimento correto da trajetória.

A lei de controle da Equação 4.5 é capaz de fazer o efetuador do robô seguir uma trajetória $\mathbf{x_d}(t)$. É possível, portanto, especificar uma curva parametrizada no tempo que será realizada pelo robô, desde que seja contínua e diferenciável.

Em um ambiente industrial, o tempo de execução da tarefa é uma variável importante, pois impacta diretamente no restante da cadeia produtiva. Assim, técnicas que que possibilitam impor restrições de tempo para as trajetórias se tornam mais vantajosas.

O objetivo de uma tarefa é que o robô visite todas as poses alvo em sequência. Sendo assim, a trajetória realizada pelo robô entre os alvos não é crítica para a inspeção. Entretanto, considerando que o robô está inserido em uma célula cercada com paredes e obstáculos internos que limitam em grande parte o seu espaço de trabalho, é de grande valia poder checar a ocorrência de colisões dentro da trajetória entre cada pose. Por esse motivo, o seguimento de trajetórias conhecidas é muito interessante.

Definindo um ponto inicial inicial $\mathbf{p_i}(t_i)$ e posição inicial final $\mathbf{p_f}(t_f)$, sendo $t_f > t_i$, a trajetória retilínea que conecta esses dois pontos atendendo as restrições de tempo inicial e final é:

$$\mathbf{p_d}(t) = \mathbf{p_i} + \frac{(\mathbf{p_f} - \mathbf{p_i})t}{t_f}, \tag{4.8}$$

e sua derivada:

$$\dot{\mathbf{p}}_\mathbf{d}(t) = \frac{(\mathbf{p_f} - \mathbf{p_i})}{t_f}. \tag{4.9}$$

A regulação da orientação desejada pode ser feita durante o seguimento da trajetória de posição. Assim, a pose desejada pode ser definida como:

$$\mathbf{x_d}(t) = \begin{bmatrix} \mathbf{p_d}(t) \\ \varphi_\mathbf{d} \end{bmatrix}, \tag{4.10}$$

e sua derivada:

$$\dot{\mathbf{x}}_\mathbf{d}(t) = \begin{bmatrix} \dot{\mathbf{p}}_\mathbf{d}(t) \\ \mathbf{0} \end{bmatrix}. \tag{4.11}$$

Essa estratégia permite conhecer a trajetória que será realizada pelo efetuador no espaço Cartesiano, facilitando a checagem de colisões entre o efetuador e obstáculos presentes no espaço de trabalho.

A Figura 20 mostra um exemplo de movimento linear do efetuador no espaço utilizando essa estratégia. A Figura 21 mostra a evolução do erro durante o movimento, e a Figura 22 mostra os sinais de controle aplicados em cada junta para realizar essa trajetória.

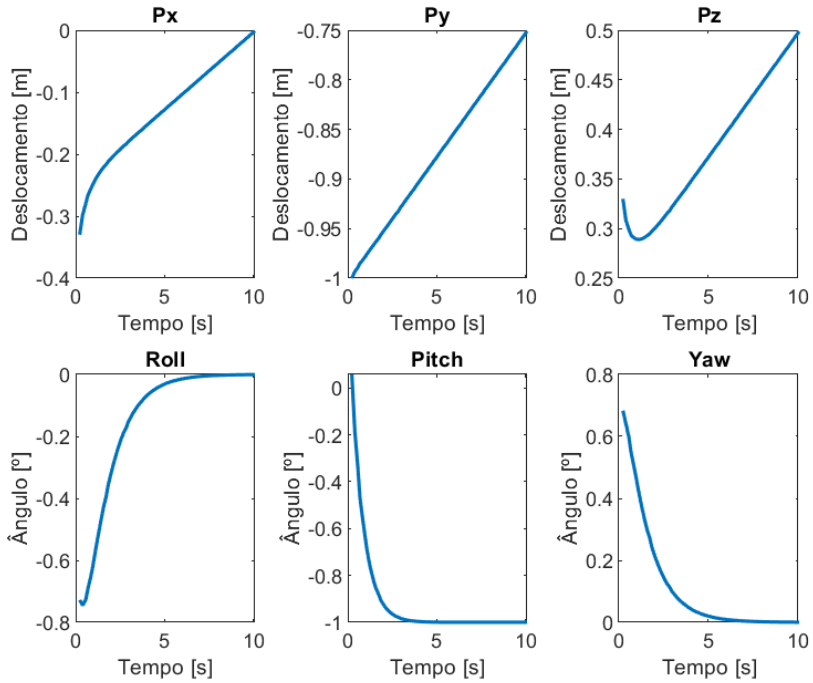


Figura 20 – Evolução da pose do efetuador com o comando de movimento no espaço Cartesiano *moveL*.

4.2.2 Movimento ponto-a-ponto no espaço das juntas

O objetivo do movimento ponto-a-ponto no espaço das juntas é levar o robô de uma configuração $\mathbf{q_i}$ até uma configuração desejada $\mathbf{q_d}$, atuando nas juntas de maneira independente. Para gerar a trajetória de cada junta, supomos que no tempo t_0 as variáveis da junta satisfazem:

$$q(t_0) = q_0, \tag{4.12}$$

$$\dot{q}(t_0) = v_0, \tag{4.13}$$

$$\ddot{q}(t_0) = \alpha_0, \tag{4.14}$$

e deseja-se alcançar os valores em t_f:

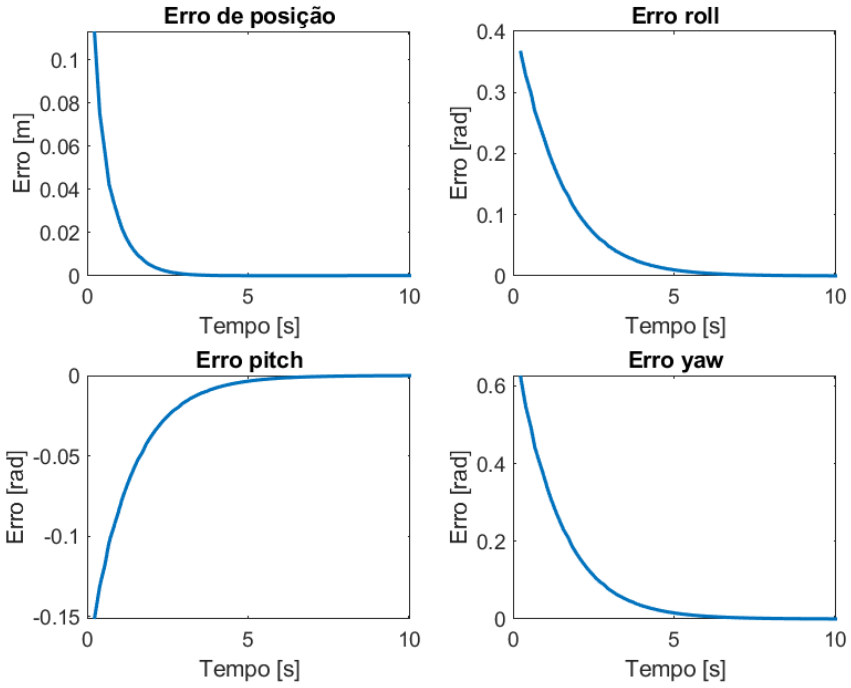


Figura 21 – Evolução do erro com o comando de movimento no espaço Cartesiano *moveL*.

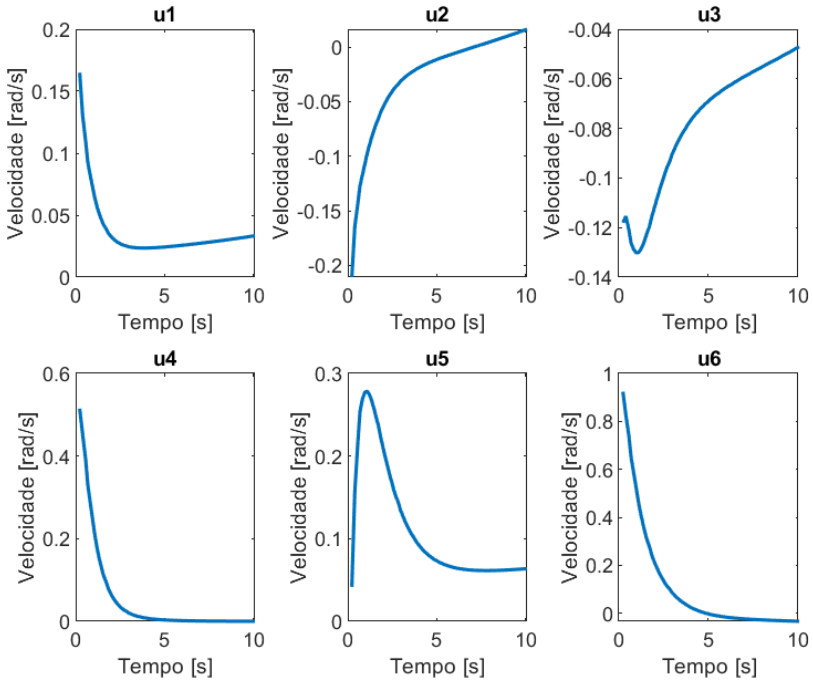


Figura 22 – Sinais de controle em cada junta com o comando de movimento no espaço Cartesiano *moveL*.

$$q(t_f) = q_f, \tag{4.15}$$
$$\dot{q}(t_f) = v_f, \tag{4.16}$$
$$\ddot{q}(t_f) = \alpha_f. \tag{4.17}$$

A trajetória deve satisfazer todas as seis restrições impostas nas variáveis das juntas. Para isso, é necessário que ela seja, no mínimo, um polinômio de quinta ordem. Assim, a velocidade e aceleração desejadas são:

$$\dot{q}(t) = a_1 + 2a_2t + 3a_3t^2 + 4a_4t^3 + 5a_5t^4, \tag{4.18}$$

$$\ddot{q}(t) = 2a_2 + 6a_3t + 12a_4t^2 + 20a_5t^3. \tag{4.19}$$

E a trajetória da junta i é dada pelo polinômio de quinta ordem:

$$q_i(t) = a_0 + a_1t + a_2t^2 + a_3t^3 + a_4t^4 + a_5t^5, \tag{4.20}$$

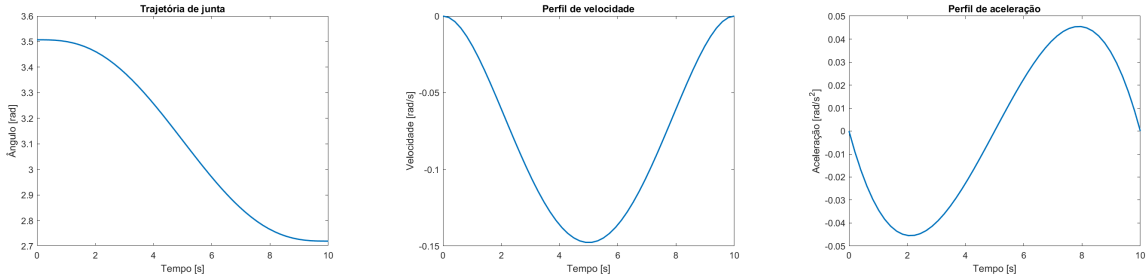
onde $i = 1, \ldots, n$. Os coeficientes $a_0, a_1, a_2, a_3, a_4, a_5$ podem ser determinados a partir dos valores de t_0 e t_f. Tomando as derivadas da Equação 4.20, as seguintes equações são obtidas:

$$\begin{aligned}
q_0 &= a_0 + a_1t_0 + a_2t_0^2 + a_3t_0^3 + a_4t_0^4 + a_5t_0^5, \\
v_0 &= a_1 + 2a_2t_0 + 3a_3t_0^2 + 4a_4t_0^3 + 5a_5t_0^4, \\
a_0 &= 2a_2 + 6a_3t_0 + 12a_4t_0^2 + 20a_5t_0^3, \\
q_f &= a_0 + a_1t_f + a_2t_f^2 + a_3t_f^3 + a_4t_f^4 + a_5t_f^5, \\
v_f &= a_1 + 2a_2t_f + 3a_3t_f^2 + 4a_4t_f^3 + 5a_5t_f^4, \\
a_f &= 2a_2 + 6a_3t_f + 12a_4t_f^2 + 20a_5t_f^3.
\end{aligned} \tag{4.21}$$

Considerando que a geração de trajetória para movimento ponto a ponto no espaço das juntas tem condições iniciais e finais nulas para a velocidade e aceleração, as Equações 4.21 podem ser escritas como:

$$\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\
0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\
0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\
1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\
0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\
0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}
=
\begin{bmatrix} q_0 \\ 0 \\ 0 \\ q_f \\ 0 \\ 0 \end{bmatrix}. \tag{4.22}$$

Os perfis de trajetória, velocidade e aceleração obtidos pela Equação 4.22 podem ser vistos na Figura 23.



(a) Trajetória da junta (b) Perfil de velocidade (c) Perfil de aceleração

Figura 23 – Perfis de trajetória, velocidade e aceleração de uma junta utilizando um polinômio de quinta ordem.

4.3 Calibração de uma Câmera

A calibração de câmera é o processo de estimar o parâmetros intrínsecos e extrínsecos de uma câmera. Os parâmetros intrínsecos estão relacionados com características internas da câmera, como distância focal, centro óptico, coeficientes de distorção da lente.

A distância focal (f_x, f_y) de uma câmera é a distância entre o sensor de imagem e o centro óptico da lente, e é tipicamente expressada em milímetros. O centro óptico (c_x, c_y), por sua vez, é o ponto onde o eixo óptico da câmera intercepta o plano de imagem, ou seja, é o ponto no sensor de imagem em que a luz converge depois de passar pela lente. A matriz de parâmetros intrínsecos câmera, apresentada na Equação 4.23, é uma matriz 3x3 que sintetiza essas informações e pode ser utilizada para imagens capturadas pela mesma câmera.

$$\mathbf{K_c} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.23}$$

Os coeficientes de distorção descrevem o quanto a imagem é distorcida. Os dois tipos de distorção mais frequentes são a radial e a tangencial. A distorção radial faz com que linhas retas pareçam curvas, e se torna mais acentuada quanto mais longe o ponto se afasta do centro da imagem. A distorção radial pode ser representada por:

$$x_{radial} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \tag{4.24}$$

$$y_{radial} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6). \tag{4.25}$$

A distorção tangencial, por sua vez, ocorre devido a imperfeições no alinhamento entre a lente e o plano de imagem, fazendo com que algumas áreas da imagem pareçam mais próximas. A distorção tangencial é dada por:

$$x_{tangencial} = x + [2p_1xy + p_2(r^2 + 2x^2)], \tag{4.26}$$

$$y_{tangencial} = y + [p_1(r^2 + 2y^2) + 2p_2xy]. \tag{4.27}$$

Portanto, os parâmetros conhecidos como coeficientes de distorção são $(k_1, k_2, p_1, p_2, k_3)$.

A matriz de câmera é uma matriz 3×4 que descreve o mapeamento dos pontos 3D do mundo para pontos 2D no plano de imagem.

$$\mathbf{p} = \mathbf{C}\mathbf{w}, \tag{4.28}$$

$$\begin{bmatrix} p_1 \\ p_2 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \\ c_5 & c_6 & c_7 & c_8 \\ c_9 & c_{10} & c_{11} & c_{12} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ 1 \end{bmatrix}. \tag{4.29}$$

Os parâmetros extrínsecos da câmera descrevem a pose 3D da câmera, como uma rotação e uma translação. A matriz $\mathbf{C}$ pode ser obtida a partir da multiplicação das matrizes de parâmetros intrínsecos e extrínsecos da câmera:

$$\mathbf{C} = \mathbf{K_c} \begin{bmatrix} \mathbf{R_c}\mathbf{t_c} \end{bmatrix}, \tag{4.30}$$

onde R_c e t_c são as matrizes de rotação e vetor de translação da câmera, respectivamente.

O processo de calibração da câmera [Zhang, 2000] consiste em capturar diversas imagens de um objeto de calibração com geometria conhecida, tipicamente um padrão *chessboard*, em diferentes pontos de vista. O objeto fornece um conjunto de pontos com coordenadas conhecidas em três dimensões, que podem ser utilizadas para estimar os parâmetros da câmera a partir de métodos computacionais. O resultado do processo de calibração completo são as matrizes de parâmetros intrínsecos extrínsecos da câmera, que podem ser utilizados para mapear pontos 3D do mundo em pontos 2D no plano de imagem. Esta é uma etapa importante no processo de detecção de *features* visuais e marcadores fiduciais, que são detectados a partir de suas características. Isso permite a estimação da pose do marcador no mundo, que é utilizada dentro da malha de controle servo visual implementada.

4.4 Detecção do Marcador

Para ser possível estimar a pose da peça inspecionada, é preciso definir um conjunto de *features*, ou características visuais que são detectadas, e posteriormente utilizadas para

determinar a pose da peça. Nesta dissertação, foram utilizados **marcadores ArUco**, por serem marcadores fiduciais com ampla utilização na robótica, com algoritmos de detecção rápidos, eficientes, e bem consolidados.

O processo de detecção, resumido pela Figura 24 e descrito no trabalho [Garrido-Jurado et al., 2014], inicia com a segmentação da imagem em escala de cinza dos contornos principais do marcador. Em seguida, os contornos são extraídos da imagem e filtrados. Isso gera um conjunto de contornos, muitos dos quais não fazem parte do marcador fiducial. Para eliminar os contornos que não fazem parte do marcador, é assumido que os marcadores são retangulares, portanto todos os contornos que não se aproximam de um polígono com quatro vértices são descartados. Na próxima etapa, o interior do marcador é analisado para a extração do código binário. A projeção de perspectiva é corrigida calculando a matriz de homografia. A imagem resultante é binarizada e dividida em uma matriz com células que são atribuídas o valor 0 ou 1, dependendo do valor da maioria dos pixels contidos nela. Por fim, é preciso determinar se o marcador pertence ao dicionário ou não. Para isso, o marcador é considerado em suas 4 possíveis rotações, e caso alguma delas pertença ao dicionário, o marcador é considerado como válido.

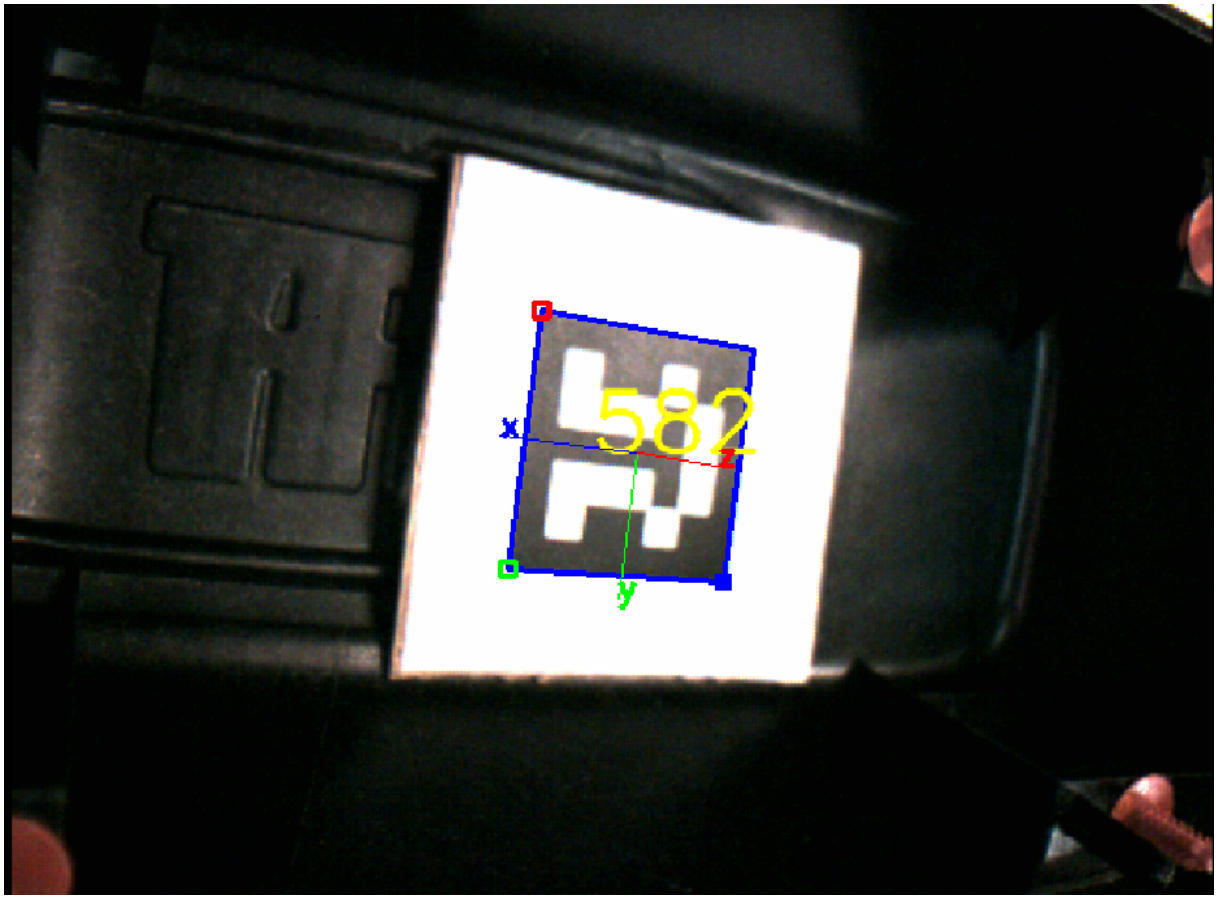


Figura 24 – Detecção de um marcador ArUco na peça.

4.5 Calibração *Hand-Eye*

Ao estimar a pose de um marcador, é possível conhecer a pose relativa do marcador em relação à câmera. Entretanto, para obter a pose do marcador em relação a um referencial inercial (como por exemplo, a base do robô), é preciso conhecer a transformação homogênea entre o efetuador e a câmera. A Figura 25 mostra como é a cadeia de transformações entre a base do robô e o marcador em uma configuração onde a câmera é montada no efetuador. A transformação $\mathbf{H}_{\mathbf{E}}^{\mathbf{B}}$ entre a base e o efetuador do robô é conhecida, podendo ser obtida pela cinemática direta. A transformação $\mathbf{H}_{\mathbf{M}}^{\mathbf{C}}$ entre a câmera e o marcador também é conhecida, através do algoritmo de estimação de pose. Portanto, a transformação $\mathbf{H}_{\mathbf{C}}^{\mathbf{E}}$ entre a câmera (*eye*) e o efetuador (*hand*) pode ser obtida através do processo chamado de calibração *hand-eye* [Daniilidis, 1999].

Uma das formas de realizar essa calibração envolve a utilização de um padrão de calibração, como um tabuleiro de marcadores ArUco (Figura 25), que deve ser posicionado no campo de visão da câmera. O próximo passo é mover o efetuador para algumas poses conhecidas, e capturar imagens do padrão através da câmera em cada uma dessas poses. Para uma única pose, existem duas matrizes de transformação desconhecidas: do marcador com respeito a base do robô, e da câmera com respeito ao efetuador. Com duas poses, sendo $\mathbf{H}_{\mathbf{M,1}}^{\mathbf{C}}$ e $\mathbf{H}_{\mathbf{M,2}}^{\mathbf{C}}$ as transformações do marcador com respeito à câmera nas poses 1 e 2, respectivamente, e $\mathbf{H}_{\mathbf{E,1}}^{\mathbf{B}}$ e $\mathbf{H}_{\mathbf{E,2}}^{\mathbf{B}}$ as transformações entre a base e o efetuador nas poses 1 e 2, é possível compor os vetores de movimento para obter a Equação formulada por [Shiu and Ahmad, 1987] e [Tsai et al., 1989]:

$$\mathbf{A}\mathbf{H}_{\mathbf{C}}^{\mathbf{E}} = \mathbf{H}_{\mathbf{C}}^{\mathbf{E}}\mathbf{B}, \tag{4.31}$$

onde $\mathbf{A} = \mathbf{H}_{\mathbf{M,1}}^{\mathbf{C}}{\mathbf{H}_{\mathbf{M,2}}^{\mathbf{C}}}^{-1}$, $\mathbf{B} = {\mathbf{H}_{\mathbf{E,1}}^{\mathbf{B}}}^{-\mathbf{1}}\mathbf{H}_{\mathbf{E,2}}^{\mathbf{B}}$ e $\mathbf{H}_{\mathbf{C}}^{\mathbf{E}}$ é a matriz de transformação desejada. Após a calibração, é possível utilizar a matriz $\mathbf{H}_{\mathbf{C}}^{\mathbf{E}}$ para determinar a pose do marcador em relação à base $\mathbf{H}_{\mathbf{M}}^{\mathbf{B}}$, permitindo posicionar o efetuador de forma que o marcador seja visto pela câmera em diversas posições e orientações diferentes.

4.6 Fluxo de Inspeção Automática

A inspeção de peças automotivas realizada pela célula de inspeção Invent Vision tem como princípio de funcionamento algoritmos de visão computacional que realizam a detecção de falhas de manufatura nas peças. Para iniciar uma inspeção, um operador posiciona a peça na máquina e aperta uma botoeira, que dá início ao ciclo de movimentação do robô. O robô visita todas as posições exigidas pelo programa, que já estão previamente registradas em sua memória, e a câmera captura imagens da peça em cada uma dessas posições. A Figura 26 apresenta o fluxo desse ciclo de inspeção.

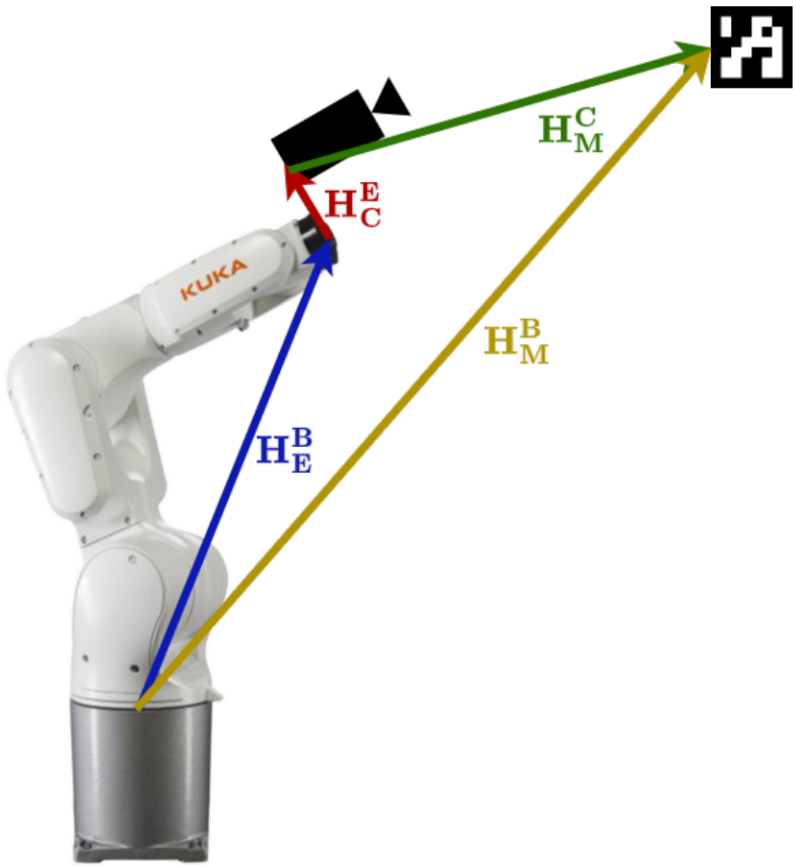


Figura 25 – Sistema de transformadas no sistema com configuração *hand-in-eye*

.

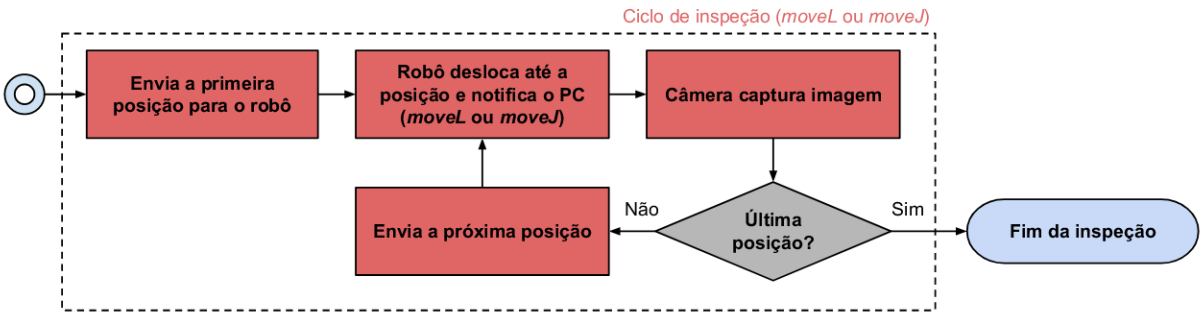


Figura 26 – Fluxo de inspeção realizado pela célula de inspeção.

Esse ciclo de inspeção, contudo, possui algumas limitações que afetam desde o projeto da célula até eventuais problemas de produção. O ciclo de inspeção é comandado pela plataforma de inspeção, que envia para o CLP um número inteiro que representa a posição para a qual o robô deve ir para capturar a próxima imagem. Cada uma dessas posições está associada a uma diferente região da peça que está sendo inspecionada. O problema ocorre quando uma peça não é encaixada corretamente na célula. O robô continuará indo para as mesmas posições que ele normalmente realiza durante um ciclo normal, porém as imagens capturadas estarão sujeitas a variabilidades devido ao posicionamento errado da peça, ocasionando em possíveis falhas de falsos negativos no resultado de inspeção.

Para tentar minimizar a ocorrência dessas falhas, a célula conta com um berço, que é projetado a partir da geometria da peça. O objetivo do berço é eliminar variabilidades no posicionamento da peça na célula. Entretanto, o projeto e fabricação desses berços não são processos triviais, pois demandam alta precisão. Além disso, o uso prolongado da máquina pode gerar desgastes e danos no berço, exigindo manutenção.

Uma maneira de contornar o problema de mau posicionamento é utilizando a própria câmera montada no efetuador. A vantagem dessa abordagem é que ela pode ser implementada sem adicionar nenhum componente extra à célula, aproveitando-se dos dispositivos que já existem na máquina. Possíveis soluções utilizando os conceitos de controle Servo Visual podem ser utilizadas para detectar o objeto inspecionado e mover o robô para poses relativas ao objeto, utilizando imagens capturadas pela câmera para determinar os comandos que serão enviados para o robô.

Esta dissertação propõe um novo fluxo de inspeção, apresentado na Figura 27. Esse fluxo é uma expansão do apresentado na Figura 26, em que foram adicionadas algumas etapas anteriores ao ciclo de inspeção, que são: rotina de busca, primeira detecção do marcador, refinamento da pose do marcador, e determinação das poses alvo. Nessas etapas, imagens da câmera são utilizadas para determinar a pose da peça com respeito a um referencial inercial, e as poses de inspeção são determinadas com relação ao objeto, em uma estrutura similar ao controle servo visual baseado em posição. As próximas subseções irão detalhar o funcionamento de cada etapa do fluxo.

4.6.1 Rotina de Busca do Marcador

A rotina de busca do marcador é a primeira etapa do novo fluxo. Nessa etapa, é assumido que a localização da peça ainda não é conhecida. Assim, o robô realiza uma rotina de busca que procura a peça em todo o espaço da tarefa, que pode ser uma trajetória pré-definida, ou uma sequência de pontos. Na aplicação dessa dissertação, o espaço de tarefa do robô está limitado nas extremidades de uma mesa, na qual o robô estava montado. A peça pode estar em qualquer posição e orientação em cima da mesa.

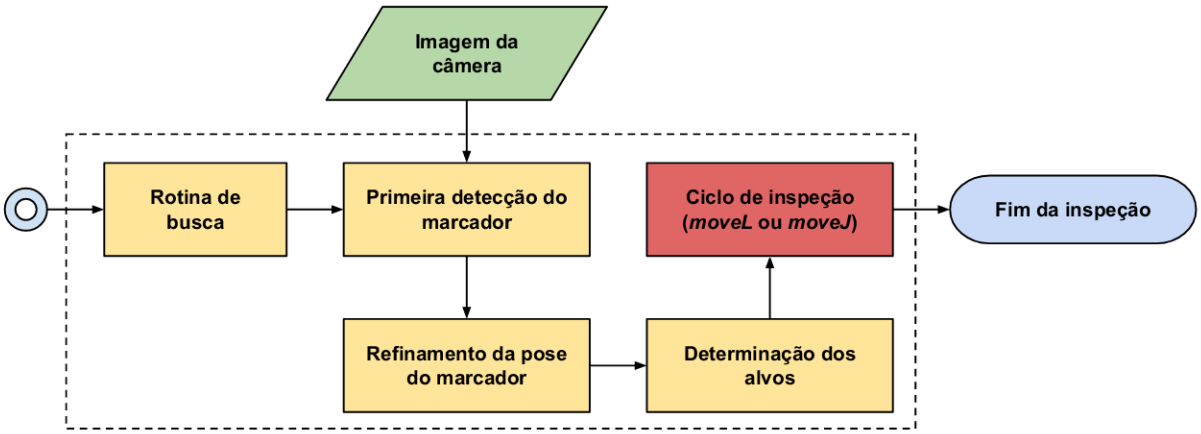


Figura 27 – Fluxo de inspeção automática com estimação de pose do marcador.

Foram definidas duas poses, uma em cada extremidade da mesa, e ao realizar uma trajetória linear (*moveL*) entre essas duas poses, mantendo a orientação constante (com a câmera "olhando" para a superfície de trabalho), é possível varrer todo o espaço de trabalho com a câmera. A Figura 28 mostra o robô nas duas poses que demarcam a rotina de busca. A rotina de busca é realizada até que o marcador da peça seja identificado.

Figura 28 – Rotina de busca do marcador.

4.6.2 Estimação e Refinamento da Pose do Marcador

Após o marcador ser identificado pela primeira vez, é possível estimar a sua pose com respeito à câmera. O problema *Perspective-n-Point*, ou PnP [Fischler and Bolles, 1981], é o problema de estimação da pose de uma câmera em relação a n pontos 3D e suas projeções 2D em uma imagem. Os cantos do marcador podem ser utilizados para encontrar a solução do problema PnP, que fornece uma matriz de transformação que determina

a translação e rotação do sistema de coordenadas do marcador em relação ao sistema de coordenadas da câmera. Para realizar a estimação, é preciso conhecer os parâmetros intrínsecos da câmera e seus coeficientes de distorção, que podem ser obtidos através do processo de calibração. Para isso, podem ser utilizados algoritmos que minimizam iterativamente o erro de reprojeção dos cantos, como o algoritmo de Levenberg-Marquadt [Marquardt, 1963].

A estimação da pose do marcador fornece a transformação do sistema de coordenadas do marcador com respeito a câmera, $\mathbf{H}^{\mathbf{C}}_{\mathbf{M}}$. Ao realizar a primeira detecção, uma primeira estimativa de $\mathbf{H}^{\mathbf{C}}_{\mathbf{M}}$, é obtida. Com essa primeira estimativa, é possível determinar uma pose do efetuador na qual a câmera enxergue o marcador de forma aproximadamente perpendicular. Supondo que $\mathbf{H}^{\mathbf{C}}_{\mathbf{M},d}$ é a matriz de transformação que representa a pose desejada do marcador com respeito a câmera, é possível obter a pose do efetuador $\mathbf{H}^{\mathbf{B}}_{\mathbf{E},d}$ para levar a câmera até a pose desejada:

$$\mathbf{H}^{\mathbf{B}}_{\mathbf{E},d} = \mathbf{H}^{\mathbf{B}}_{\mathbf{E}} \cdot \mathbf{H}^{\mathbf{E}}_{\mathbf{C}} \cdot \mathbf{H}^{\mathbf{C}}_{\mathbf{M}} \cdot {\mathbf{H}^{\mathbf{C}}_{\mathbf{M},d}}^{-1} \cdot {\mathbf{H}^{\mathbf{E}}_{\mathbf{C},d}}^{-1} \tag{4.32}$$

Ao chegar nessa pose, a câmera realiza a captura de n imagens, que fornecerão n amostras da estimativa de $\mathbf{H}^{\mathbf{C}}_{\mathbf{M}}$. Então, um filtro de média é utilizado para refinar a estimativa da pose do marcador com respeito a câmera.

4.6.3 Definição das Poses de Inspeção

Com a estimativa de $\mathbf{H}^{\mathbf{C}}_{\mathbf{M}}$, é possível utilizar a equação 4.32 para determinar poses do efetuador $\mathbf{H}^{\mathbf{B}}_{\mathbf{E}}$ que levam a câmera até uma pose relativa ao marcador. Dessa forma, é possível definir uma sequência de n poses $\mathbf{H}^{\mathbf{C}}_{\mathbf{M},1} \dots \mathbf{H}^{\mathbf{C}}_{\mathbf{M},n}$ relativas que serão utilizadas para realizar uma inspeção.

A vantagem de utilizar poses relativas é que independente da localização da peça com respeito ao referencial inercial, o robô será capaz de fazer o posicionamento da câmera para realizar a inspeção. Isso elimina o problema de variabilidade nas imagens de inspeção devido a problemas de encaixe ou mal posicionamento da peça na célula. Além disso, a escolha das poses se torna mais intuitiva ao pensar em poses relativas entre a câmera e o marcador, sendo necessário apenas definir a translação $\mathbf{p}^{\mathbf{C}}_{\mathbf{M},d}$ e a rotação $\mathbf{R}^{\mathbf{C}}_{\mathbf{M},d}$ desejadas entre os dois sistemas de coordenadas.

Uma premissa importante da aplicação nesta dissertação é de que a peça permanece parada com respeito ao referencial inercial durante todo o processo de inspeção. Isso permite que as poses do efetuador $\mathbf{H}^{\mathbf{B}}_{\mathbf{E},d}$ que levam até a câmera até a pose desejada sejam calculadas apenas uma vez (após a estimação de $\mathbf{H}^{\mathbf{C}}_{\mathbf{M}}$), sem necessidade de serem recalculadas até o fim da inspeção. Dessa forma, a captura de imagens para estimação da pose do marcador é necessária somente até o momento em que o filtro de média é aplicado. Após esse momento,

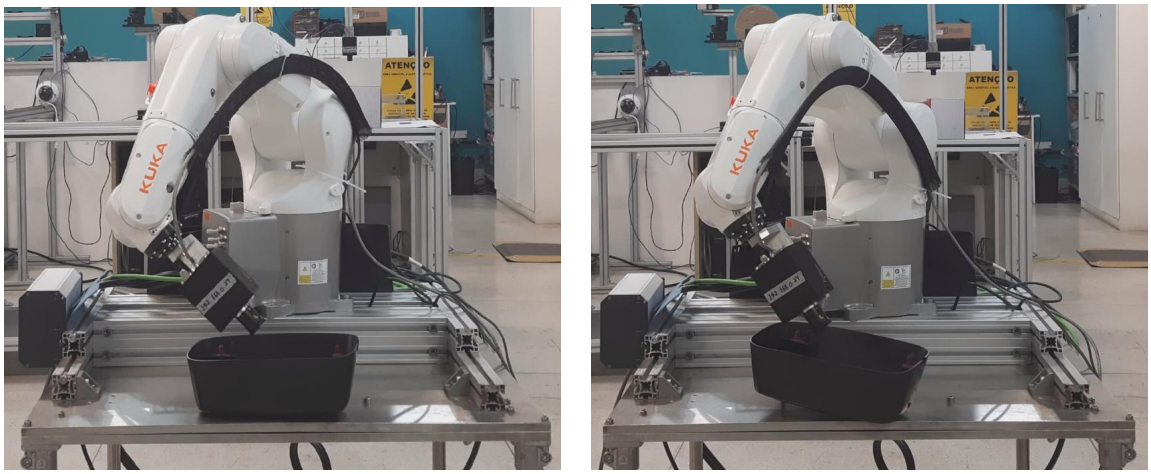
a captura de imagens é realizada apenas para a inspeção de fato, diminuindo a quantidade de processamento necessária. Além disso, também não é necessário manter o marcador em cena durante a inspeção, pois a sua pose já é conhecida.

4.6.4 Execução da Inspeção

A definição das poses de inspeção fornece uma sequência de poses $\mathbf{H}^{\mathbf{B}}_{\mathbf{E},1} \dots \mathbf{H}^{\mathbf{B}}_{\mathbf{E},n}$ que devem ser alcançadas pelo efetuador. O sistema é versátil, permitindo que o movimento entre cada pose possa ser executado tanto pelo movimento do tipo *moveL* quanto *moveJ* [Spong et al., 2006], que são nativos do robô.

No fim, o fluxo de inspeção expandido com a estimação da pose do marcador fornece ao sistema muito mais flexibilidade quanto ao posicionamento da peça. Isso tem impacto direto na ocorrência de erros de produção, além de permitir uma simplificação do projeto do berço da célula. A Figura 29 mostra como o novo fluxo permite a captura da mesma visão da peça, permitindo a detecção do componente plástico em vermelho, apesar de peça ter sido colocada em posições e orientações diferentes na mesa. Na Figura 30, é mostrada a sequência de poses durante uma inspeção da peça.

(a) Posição original (b) Posição deslocada



(c) Posição deslocada (d) Posição deslocada

Figura 29 – Inspeção da peça: (a) Na posição original vista da câmera. (b) Em uma posição com translação e orientação diferentes vista da câmera. (c) Vista da peça na mesa. (d) Vista da peça deslocada na mesa.

Figura 30 – Execução da inspeção.

Capítulo 5

Arcabouço Experimental

Neste capítulo, é apresentado o arcabouço experimental implementado para validar a metodologia empregada no fluxo de inspeção automática proposto. A primeira parte do capítulo apresenta o ambiente virtual utilizado para simulação do sistema. A modelagem do robô e a criação do cenário de simulação é apresentado, bem como a implementação dos algoritmos em MATLAB.

Em seguida, são apresentados os experimentos de validação realizados com um robô real em laboratório. São apresentados o robô e a câmera utilizados para realizar experimentos. Também são descritos os pacotes ROS e *softwares* utilizados para comunicação e controle do robô e da câmera, e os pacotes criados para implementar o fluxo de inspeção.

5.1 Experimentos em Ambiente Simulado

A simulação da célula de inspeção é uma ferramenta que se prova muito útil na etapa de modelagem da tarefa, que consiste na escolha das poses que serão visitadas pelo robô para a captura de imagens. Esse procedimento, em geral, é feito de maneira empírica, posicionando o robô manualmente em diferentes configurações até que a imagem capturada pela câmera seja adequada. Esse processo, entretanto, requer que a célula de inspeção esteja completamente montada, com o robô e a câmera instalados. Com isso, a simulação pode ser utilizada para realizar uma emulação das poses de inspeção de forma independente da célula real. Nesse contexto, o fluxo de inspeção automática com detecção e estimação da pose do marcador pode ser inteiramente validado dentro de um ambiente de simulação.

5.1.1 Modelo Virtual do Robô

As simulações foram realizadas utilizando o *software* de simulação de robótica CoppeliaSim [Rohmer et al., 2013]. Esse simulador possui ferramentas para a criação de modelos de robôs não existentes em sua biblioteca padrão a partir dos parâmetros de

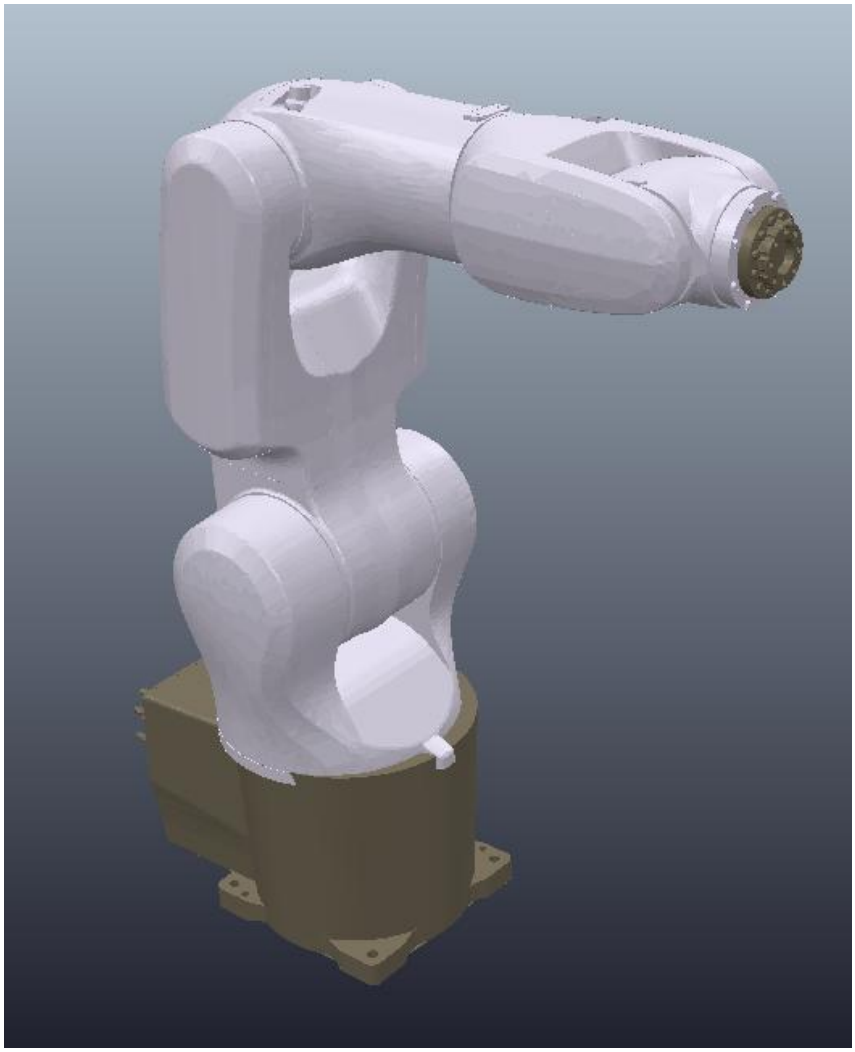


Figura 31 – Modelo do robô KUKA KR4 R600 no simulador CoppeliaSim.

Denavit-Hartenberg. A partir da tabela de parâmetros, uma cadeia de juntas em série é construída, respeitando o comprimento dos elos. Por fim, as malhas 3D (em formato *.stl*) de cada elo do robô são carregadas independentemente e posicionadas em cada elo, compondo os modelos visuais e espaços de colisão de cada elo.

5.1.2 Ambiente Virtual Utilizado nas Simulações

O modelo CAD da célula de inspeção real pode ser importado dentro do simulador, como visto na Figura 32, juntamente a uma peça fictícia que representa um *spoiler* de automóvel posicionada em seu interior. Por fim, o robô é fixado na posição correspondente à montagem real, e uma câmera fixada em seu efetuador. Essa configuração permite a prototipagem virtual de células de manufatura e inspeção, por meio da co-simulação entre o MATLAB e o CoppeliaSim, utilizando modelos CAD.

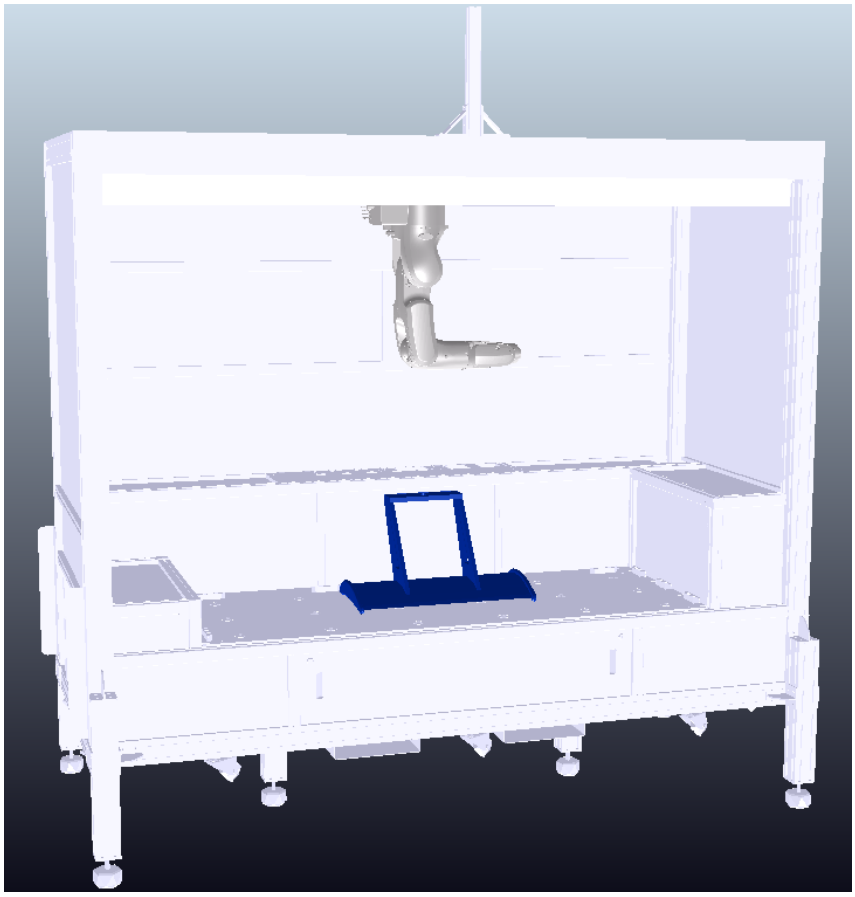


Figura 32 – Célula de inspeção no ambiente CoppeliaSim.

5.1.3 Implementação do Fluxo de Inspeção no MATLAB

Para simular a tarefa de inspeção, as estratégias de controle serão implementadas no MATLAB, com auxílio do *Robotics Toolbox* [Corke, 1996], e o CoppeliaSim, ambos com licenças educacionais. O *script* MATLAB se comunica em tempo real com o simulador, enviando comandos e recebendo dados do cenário.

A câmera virtual acoplada no efetuador do robô captura imagens em perspectiva do ambiente de simulação. Para detectar e estimar a pose de um marcador durante a simulação, foi escrito um código em linguagem C++, utilizando a biblioteca de visão computacional OpenCV. Essa biblioteca possui funções para gerar marcadores ArUco, e também possui implementações rápidas e eficientes dos algoritmos de detecção e estimação da pose do marcador.

O código compilado gera um executável, que pode ser invocado pelo MATLAB durante a execução de *scripts* a partir de um arquivo MEX. Este é um tipo de arquivo que cria uma interface entre o MATLAB e funções escritas em C ou C++. O código possui como entrada uma imagem, gerada pela câmera no CoppeliaSim, e retorna um vetor $\in \mathbb{R}^{6\times1}$ com os dados de translação e rotação (dada em ângulos RPY) do marcador

com respeito a câmera.

5.2 Experimentos em Laboratório com Robô Real

Para validar o fluxo de inspeção automática proposto, foi utilizado um robô KUKA KR4 R600 e uma câmera industrial Invent Vision V200. Este Seção entra em detalhes sobre a utilização de cada um desses dispositivos e a montagem do ambiente experimental em laboratório, bem como os *softwares* utilizados.

5.2.1 Robô KUKA KR4 R600

O robô industrial KUKA KR4 R600 [Kuka Robotics, 2021] (Figura 33(a)) é um robô compacto e rápido, com seis eixos, projetado para aplicações de manipulação de precisão. Seu design leve permite que ele se mova de forma rápida e eficiente, tornando-o uma escolha ideal para aplicações que requerem ciclos rápidos e alta produção.

(a) (b)

Figura 33 – (a) Robô KUKA KR4 R600. (b) Controlador KUKA KRC5 micro e *smartPad*.

A interação com o robô e o controlador é realizada por meio de um *smartPad* (Figura 33(b)), que permite enviar comandos de movimentação cartesiana ou movimentação de juntas, programar rotinas de movimentos, ler os valores dos sensores de juntas e pose do efetuador, configurar dos eixos e interfaces de rede, e definir o sistema de coordenadas de uma ou mais ferramentas.

Além de sua velocidade e eficiência, o KR4 R600 também possui alta precisão e repetibilidade, tornando-o adequado para tarefas que exigem alto grau de precisão e velocidade. O robô possui capacidade de carga de até 6 quilogramas e alcance de até 640

milímetros, sendo mais adequado para operações de manipulação de objetos de baixa carga com precisão e velocidade.

5.2.2 Câmera Industrial Invent Vision V200

A câmera utilizada nas células de inspeção é uma câmera IP Industrial Invent Vision V200, vista na Figura 34. Essa câmera permite a captura de imagens com resolução de 5MP ou VGA. Uma biblioteca de comunicação proprietária em C++ é utilizada para realizar a conexão e comunicação com a câmera. Com essa biblioteca, é possível estabelecer uma conexão via TCP/IP por meio da interface de rede da câmera, realizar a captura de imagens, alterar parâmetros de imagem, como a resolução, ganho, tempo de exposição, e resolução. Além disso, a câmera possui uma lente que permite realizar ajuste de foco e íris.

Para a inspeção da peça, os parâmetros da câmera foram ajustados para os seguintes valores:

- Resolução: VGA (640x480);
- Ganho: 20
- Tempo de exposição: 100ms

O ajuste desses parâmetros foi feito de forma a se adequar às condições ambientais do laboratório, que estava sujeito a grande variações de luminosidade durante o dia por não contar com um sistema de iluminação artificial. Dessa forma, a escolha de um ganho e tempo de exposição relativamente altos permite a captura de uma maior quantidade de luz pelo sensor, resultando em imagens mais claras. Além disso, a escolha da resolução VGA é justificada pelo menor tamanho da imagem, resultando em uma latência de rede menor na sua transferência, e consequentemente, uma taxa de quadros mais alta.

5.2.3 Montagem do Ambiente Experimental

O ambiente experimental foi montado em uma mesa de alumínio, onde o robô e o controlador foram instalados. A câmera foi fixada no efetuador do robô utilizando uma peça de alumínio. As Figuras 35 e 36 mostram a montagem do ambiente experimental e a instalação da câmera no efetuador do robô.

Com um paquímetro, foi medida a translação entre o efetuador e o sensor da câmera, que está localizado aproximadamente no encaixe da lente. Foram obtidos os valores $x = 0,023m, y = 0m$ e $z = 0,160m$, considerando que a medição é feita a partir da origem do sistema de coordenadas do efetuador.

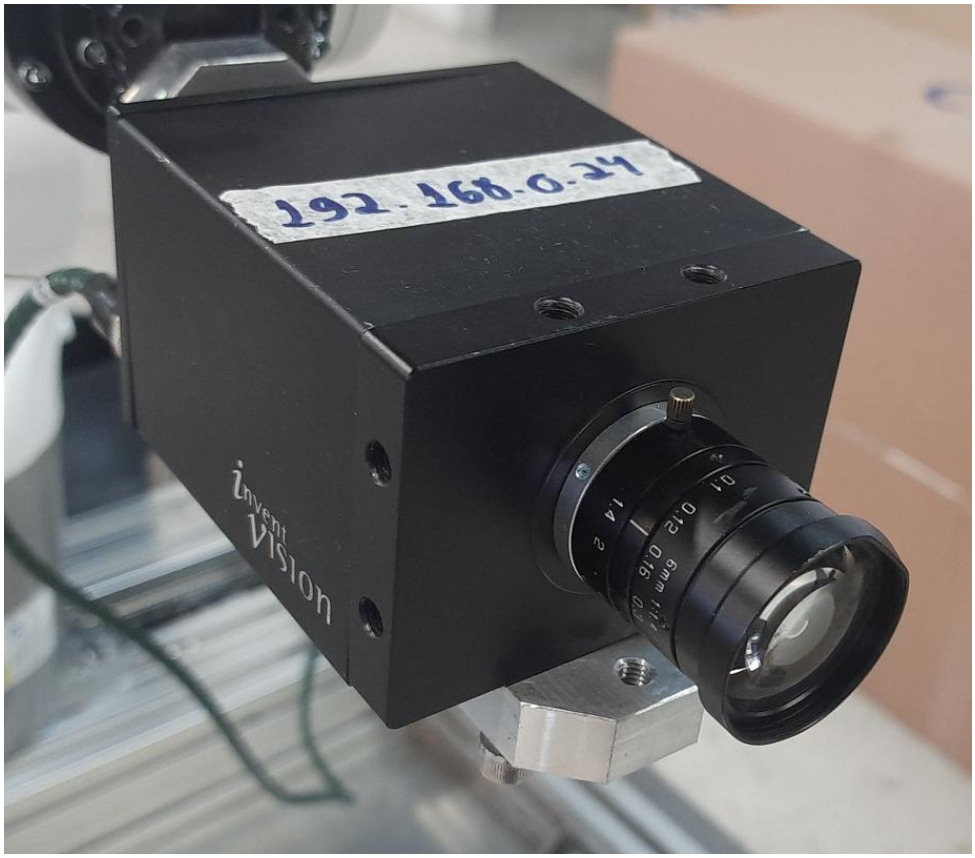


Figura 34 – Câmera Invent Vision V200 utilizada.

5.2.4 *Softwares* Utilizados na Configuração Experimental

A configuração de *software* utilizada para implementar o fluxo de inspeção é construída a partir de uma hierarquia cliente servidor TCP/IP, com componentes de *software* que são executados em um computador cliente, que é responsável pelo controle do fluxo de inspeção, processamento dos dados do robô e imagens da câmera, e envio de comandos. O servidor, executado no controlador do robô, disponibiliza variáveis de sistema que são manipuladas durante o processo. A arquitetura de comunicação cliente/servidor proposta está resumida na Figura 37:

A implementação da metodologia proposta foi feita com uso do *Robot Operating System* (ROS) [Quigley et al., 2009]. Este é um meta Sistema Operacional (SO) *open source*, que é executado sobre outro SO (normalmente Linux), facilitando o desenvolvimento de projetos de robótica, e proporcionando um arcabouço padrão de *software* que incentiva a reutilização de códigos preexistentes. O ROS também oferece uma abstração da camada de hardware, por meio de *drivers* de dispositivos que se comunicam com o ROS utilizando mensagens padrões.

Os pacotes são a principal unidade para organização de softwares no ROS. Um pacote pode conter nós (executáveis), bibliotecas e arquivos de configuração. O ROS possui

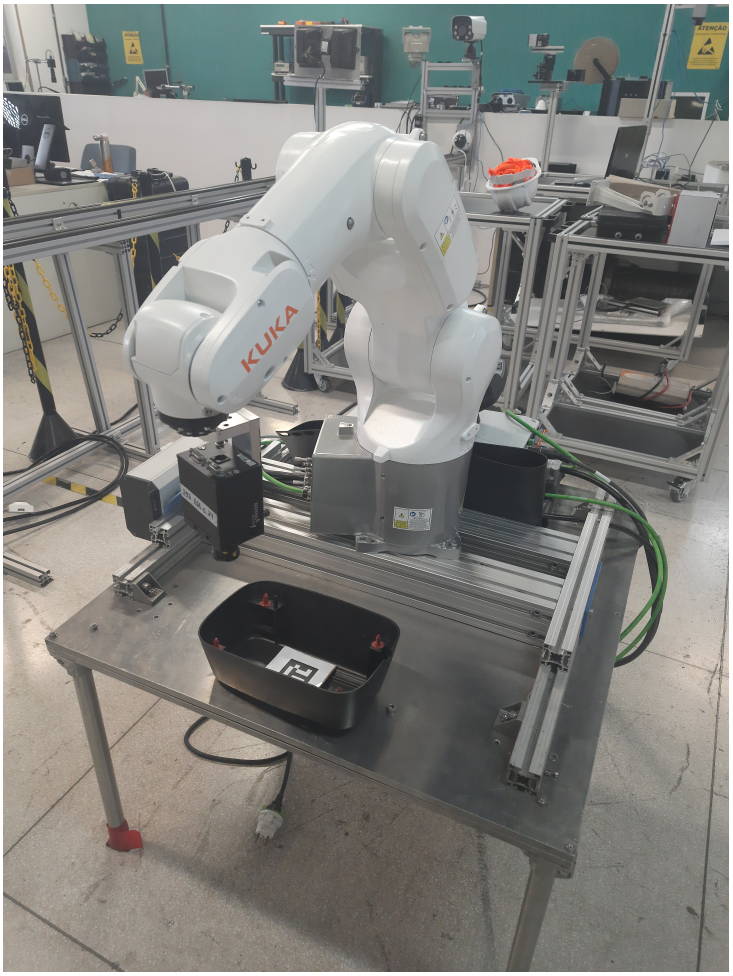


Figura 35 – Configuração do *setup* experimental.

Figura 36 – Montagem da câmera no efetuador do robô.

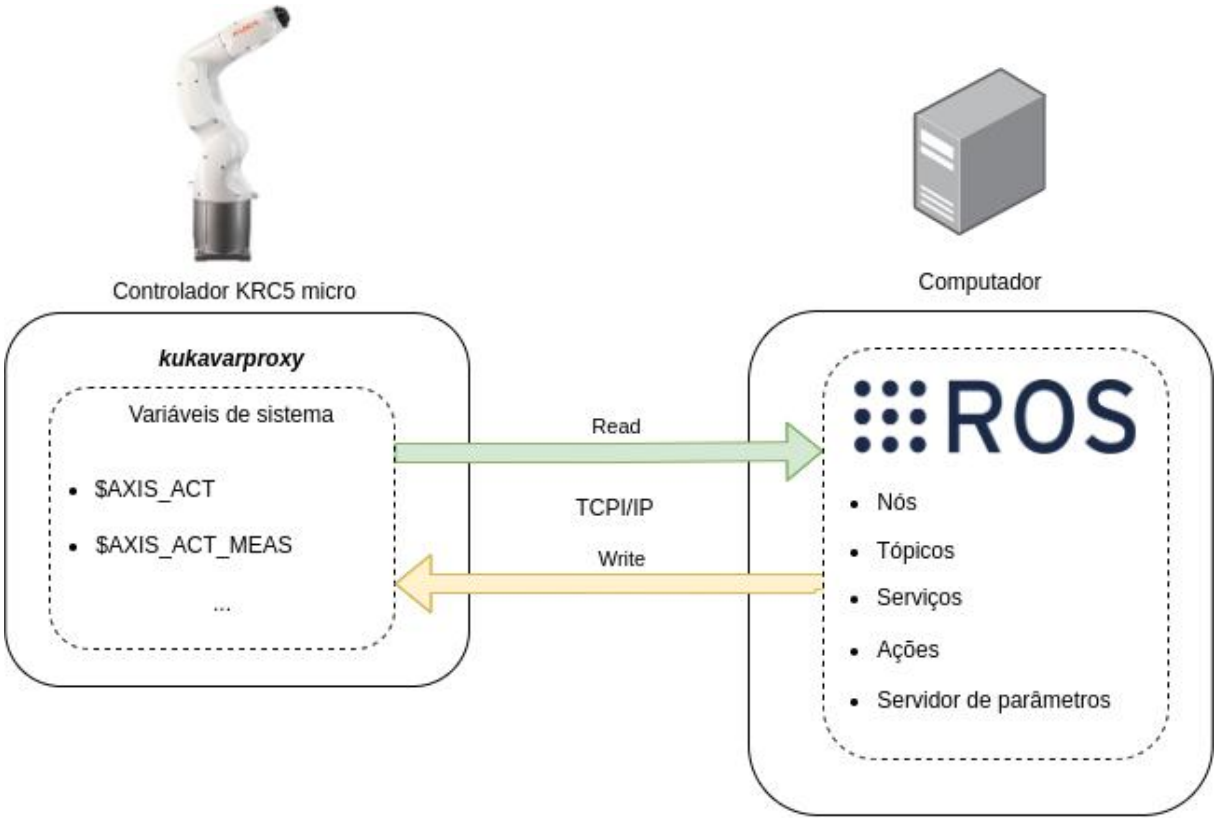


Figura 37 – Arquitetura de rede da configuração experimental.

um princípio de funcionamento modular e distribuído, onde nós são processos executados paralelamente. Nós que estão sendo executados em máquinas diferentes numa mesma rede e sob o mesmo nó mestre podem comunicar entre si da mesma maneira que nós executados numa mesma máquina. Essa modularização permite evitar que o sistema como um todo seja comprometido caso um dos nós apresente falha.

Do ponto de vista do ROS, os módulos do sistema são vistos como nós em um grafo. Um nó é um processo, como por exemplo, *drivers* de comunicação com sensores, planejadores de trajetória, processadores de imagens, dentre outras funcionalidades. A comunicação entre nós é dada pela troca de mensagens via tópicos e ações, que são as arestas do grafo. Um nó pode publicar mensagens em um tópico, e outros nós podem se inscrever nesse tópico e ler as mensagens publicadas. Os pacotes específicos utilizados no sistema são abordados em detalhe na Seção 5.2.4.2.

Para gerenciar os sistemas de coordenadas existentes em um sistema robótico, é utilizado o pacote *tf*, que é um pacote padrão do ROS. O *tf* mantém os sistemas de coordenadas do sistema em uma estrutura em forma de árvore que descreve as transformações entre cada sistema de coordenadas, permitindo demandar informações sobre os diferentes componentes na árvore e também a criação de transformações estáticas. A Figura 38

mostra a árvore de transformadas completa do sistema. Nessa árvore, as transformações da cadeia cinemática $base\_link \longrightarrow link\_6$ representam a cinemática direta do robô. A transformação $link\_6 \longrightarrow aruco\_frame$ é obtida pela estimação e refinamento da pose do marcador, e a transformação $aruco\_frame \longrightarrow first\_goal$ é de inspeção relativa ao sistema de coordenadas do marcador.

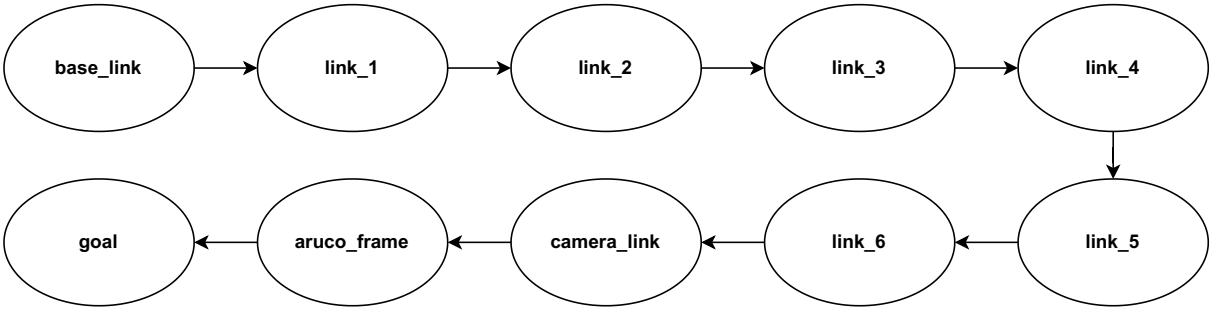


Figura 38 – Árvore de transformadas do sistema.

5.2.4.1 Servidor *kukavarproxy*

Tradicionalmente, a programação de robôs industriais é feita por meio de seu controlador específico, utilizando as linguagens proprietárias disponibilizadas pelo fabricante e com configurações de juntas pré-definidas. Entretanto, para realizar a rotina de inspeção com estimação de pose da peça, é preciso que um computador envie as poses alvo para o robô em tempo real em cada execução. O controlador KRC5 micro utilizado nos experimentos possui interfaces Ethernet que permitem a comunicação com dispositivos por meio da rede.

Para manipular variáveis internas do robô, foi utilizado o *kukavarproxy*, que é um servidor TCP/IP *open source*, executado no controlador do robô que permite ler e escrever em variáveis de sistema do robô. O robô possui uma série de variáveis globais e internas, que podem ser acessadas por meio do servidor. Em particular, duas variáveis são necessárias para o fluxo de inspeção completo: a posição atual das juntas do robô, que pode ser obtida pela leitura da variável **$AXIS_ACT_MEAS**, e *setpoint* de juntas, que podem ser escritos pela variável **$AXIS_ACT**.

5.2.4.2 Pacotes ROS Utilizados

A seguir são apresentados todos os pacotes de ROS utilizados na implementação do fluxo de inspeção automática proposto, bem como os tópicos subscritos e publicados pelos mesmos, e sua função desempenhada no sistema de exploração. A Figura 39 mostra as conexões entre os tópicos e nós do sistema.

Pacotes criados para a implementação do sistema:

- **pbvs**: este pacote contém um nó que controla todo o fluxo de inspeção com replanejamento de tarefa, com a rotina de inspeção, estimação de pose do marcador, e execução da inspeção. Os comandos de movimento são publicados no tópico */position_trajectory_controller/command*.

- **ivsn_camera**: este pacote é um *driver* de ROS para comunicação com câmeras de espectro visível Invent Vision. Internamente, a biblioteca padrão de comunicação com câmeras Invent Vision é utilizada para estabelecer a conexão com a câmera e definir os parâmetros de resolução, ganho e tempo de exposição. As imagens são publicadas no tópico */camera_publisher/camera*, e os parâmetros intrínsecos e extrínsecos da câmera são publicados no tópico */camera_publisher/camera_info*.

- **kuka_kr4_moveit**: este pacote fornece suporte para a utilização do modelo do robô no *software* MoveIt, que permite o planejamento e execução de trajetórias de robôs manipuladores integrado ao ROS.

Pacotes de código aberto utilizados:

- **kuka_experimental**: este é um pacote de suporte para utilização do robô KUKA KR4 R600 com ROS. Contém arquivos de descrição do modelo do robô no formato URDF (*Unified Robot Description Format*), *meshes* visuais e de colisão, e arquivos *launch* para carregar os modelos. Os parâmetros são carregados no servidor de parâmetros do ROS e ficam disponíveis para uso por outros nós.

- **kuka_kvp_hw_interface**: este pacote possui nós para controle de robôs KUKA por meio da manipulação de variáveis de sistema por meio do servidor *kukavarproxy*. Publica o estado atual das juntas no tópico */joint_states*. Este pacote também permite a escrita de comandos de posição de juntas no tópico */position_trajectory_controller/command*.

- **moveit_calibration**: este é um pacote que contém um *plugin* do MoveIt que fornece uma interface para a realização da calibração *hand-eye* seguindo os passos descritos na Seção 4.5.

- **aruco_ros**: este pacote contém nós para geração, detecção, e estimação da pose de marcadores ArUco. Utiliza os dados de câmera e imagem publicados pelo pacote **ivsn_camera**, e publica a pose do marcador com respeito ao sistema de coordenadas da câmera no tópico */aruco_single/pose*.

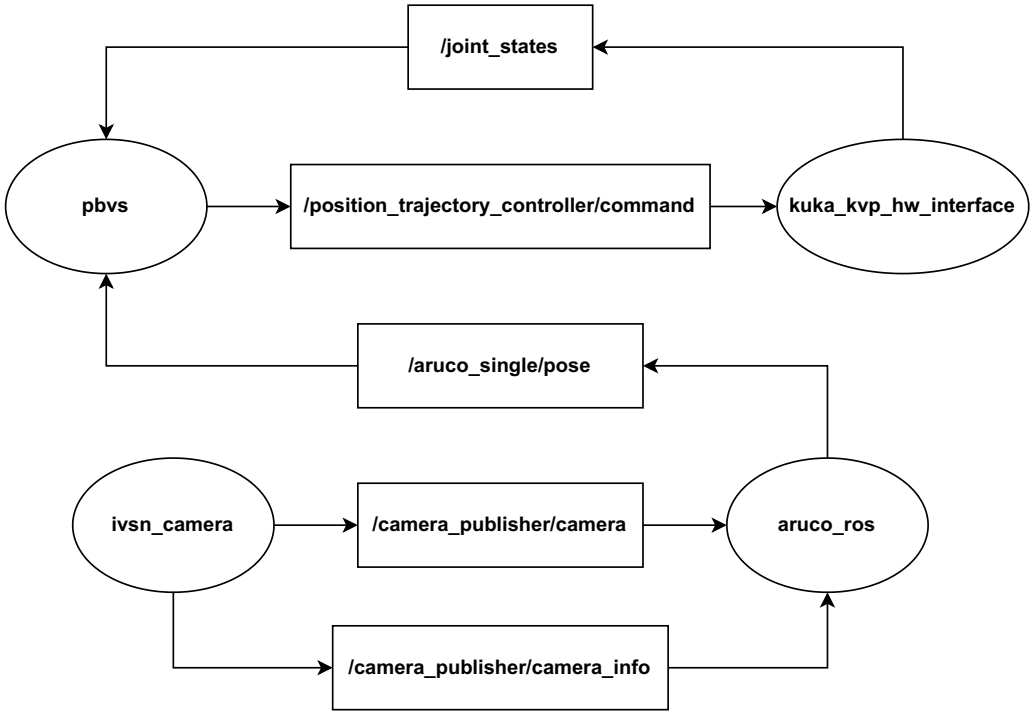


Figura 39 – Interconexões entre os nós e tópicos do sistema.

Capítulo 6

Experimentos e Resultados

Neste Capítulo, são apresentados os resultados obtidos pelos experimentos realizados em ambiente de simulação e em laboratório utilizando o arcabouço experimental. Primeiramente, são apresentados os experimentos realizados no ambiente de simulação e os seus resultados. Em seguida, são apresentados os experimentos realizados em laboratório, com uma peça automotiva real. As poses de inspeção são definidas de forma a identificar componentes que devem estar presentes na peça. Os resultados de inspeção a partir das imagens capturadas nos experimentos são analisados, assim como a trajetória realizada pelo robô em cada experimento, com os comandos de movimento *moveJ* e *moveL*.

Para que o replanejamento da tarefa de inspeção seja eficaz, a cena obtida pelas imagens capturadas durante a inspeção precisa conter as regiões da peça que são inspecionadas. Para simular a realização de inspeções assim como é feito em produção, foi utilizado o *software IvsnDesigner*, que permite a modelagem de algoritmos que são utilizados pela plataforma de inspeção das células. Para as tarefas de inspeção em ambiente de simulação e nos experimentos reais, foi modelado um algoritmo de *template matching* para realizar uma busca por uma amostra pré-definida em cada imagem. Caso a amostra seja encontrada na imagem (isto é, se o coeficiente de correlação ρ_{12} for superior a um certo limiar), é possível concluir que o replanejamento foi bem sucedido.

6.1 Resultados dos Experimentos em Simulação

Com o ambiente de simulação implementado, foram realizados experimentos para a análise e comparação das técnicas de controle de movimento *moveL* e *moveJ*. Em seguida, foram realizadas simulações para validar o fluxo de inspeção com replanejamento de tarefa proposto.

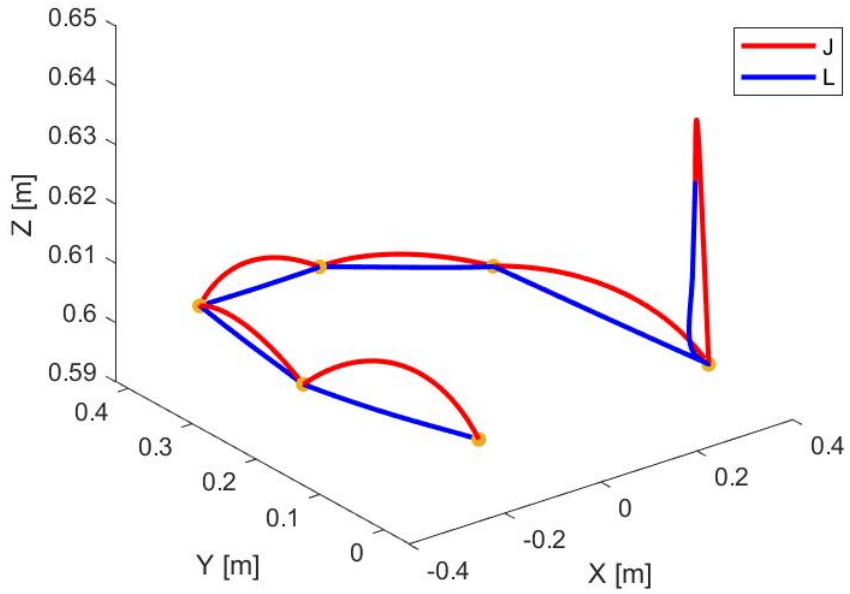


Figura 40 – Trajetórias realizadas pelo robô utilizando os comandos de movimento *moveL* e *moveJ*.

6.1.1 Comparação Entre as Técnicas de Controle de Movimento *moveL* e *moveJ*

As diferenças nas malhas de controle de cada tipo de movimento gera comportamentos diferentes durante a execução de uma tarefa. O movimento no espaço das juntas gera trajetórias individuais para cada junta. Nesse caso, a trajetória do efetuador é imprevisível, pois é apenas uma resultante da combinação das trajetórias das juntas. Por outro lado, o movimento linear no espaço Cartesiano possibilita gerar trajetórias lineares previsíveis para o efetuador.

Essas diferenças podem ser observadas na Figura 40, que mostra a trajetória realizada pelo efetuador entre diversos pontos no espaço com os movimentos *moveL* e *moveJ*.

A Figura 41 evidencia uma situação onde a imprevisibilidade do movimento no espaço das juntas resulta em uma colisão entre uma e o robô. A trajetória em amarelo mostra o movimento linear no espaço Cartesiano, e a trajetória em vermelho mostra o movimento realizado no espaço das juntas.

A Figura 42 mostra os sinais de controle que foram aplicados em cada junta durante a inspeção com vários pontos de inspeção. É possível notar que no momento em que ocorre um chaveamento de *setpoint*, há um pico no sinal de controle do movimento linear. Dessa forma, o movimento no espaço das juntas resultou em sinais de controle significativamente mais suaves para o robô quando comparados ao movimento linear.

Considerando as características de cada movimento, é importante realizar a escolha

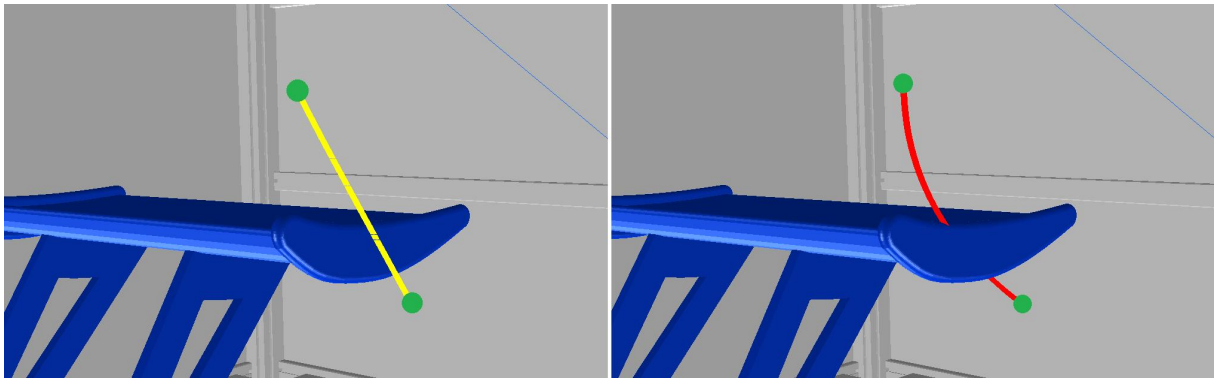


Figura 41 – Colisão gerada pelo movimento no espaço das juntas.

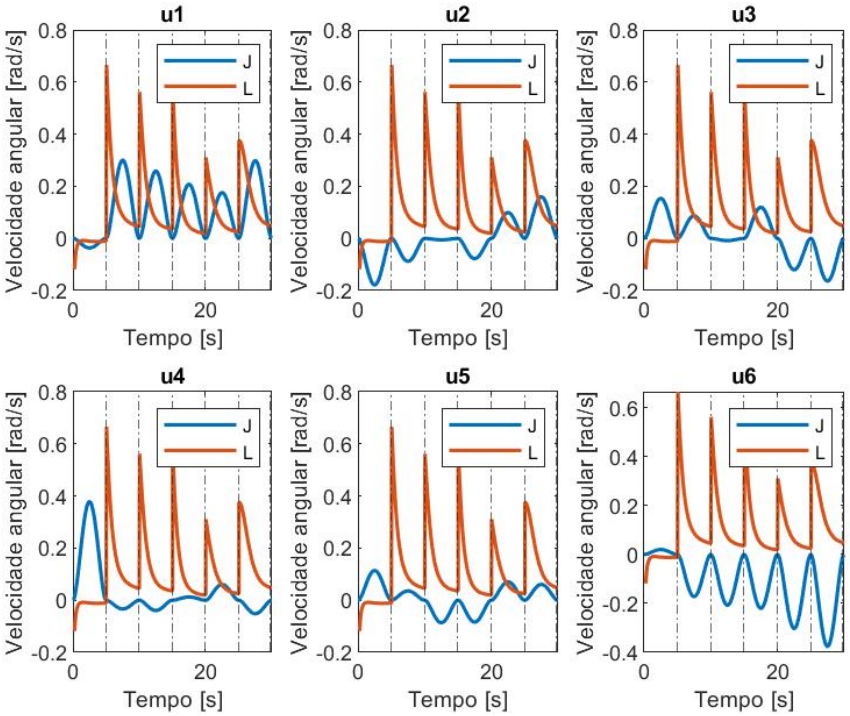
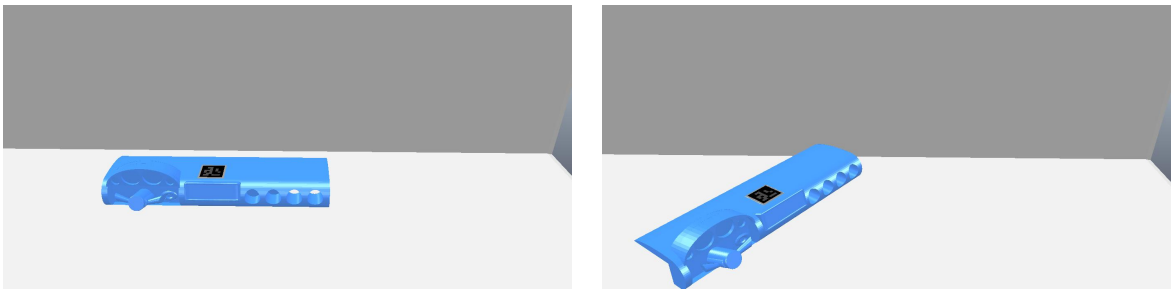


Figura 42 – Sinais de controle aplicados nas juntas utilizando os comandos de movimento *moveL* e *moveJ*.

de qual é mais adequado de acordo com a aplicação. O sistema proposto nesta dissertação é versátil e permite a escolha de qual movimento será realizado em cada inspeção.

6.1.2 Simulação de Inspeção com Replanejamento de Tarefa

Para realizar os experimentos no ambiente de simulação, foi utilizado o modelo de um painel de carro, com um marcador ArUco acoplado em sua superfície. Foram definidas quatro poses dadas com respeito ao sistema de coordenadas do marcador, que fornecem perspectivas de diferentes regiões da peça. Na Figura 43, é possível ver a peça dentro da célula de inspeção. Inicialmente, a peça foi posicionada perpendicularmente às paredes laterais da célula, como pode ser visto na Figura 43(a) (chamada de pose "original"). A simulação foi realizada por meio do código de MATLAB que implementa o replanejamento de tarefa baseada na pose estimada do marcador, utilizando a API do CoppeliaSim para salvar as imagens da câmera acoplada no efetuador do robô.



(a) Pose original (b) Pose deslocada

Figura 43 – Peça na pose original e pose deslocada.

A Figura 44 mostra a pose do robô com respeito ao sistema de coordenadas do marcador durante a execução das simulações de uma inspeção com a peça na pose original e na pose deslocada, utilizando comando de movimento no espaço Cartesiano *moveL*. As linhas vermelhas verticais indicam o momento de chaveamento de referência. É possível notar que durante toda a inspeção, o robô foi capaz de manter a pose relativa em relação ao marcador e alcançar as poses desejadas.

A Figura 45 mostra a pose absoluta do robô durante a simulação. É possível perceber que apesar do robô manter a pose relativa ao marcador, as poses absolutas são diferentes, evidenciando o replanejamento da tarefa realizado.

Para a execução do algoritmo de *template matching*, foi definida uma amostra para cada pose de inspeção. A Figura 46 mostra as amostras das regiões que são inspecionadas em cada pose.

O algoritmo de *template matching* retorna a região mais provável onde a amostra pode ser encontrada na imagem, bem como o coeficiente de correlação calculado para essa região. A inspeção foi feita inicialmente com a peça na pose original.

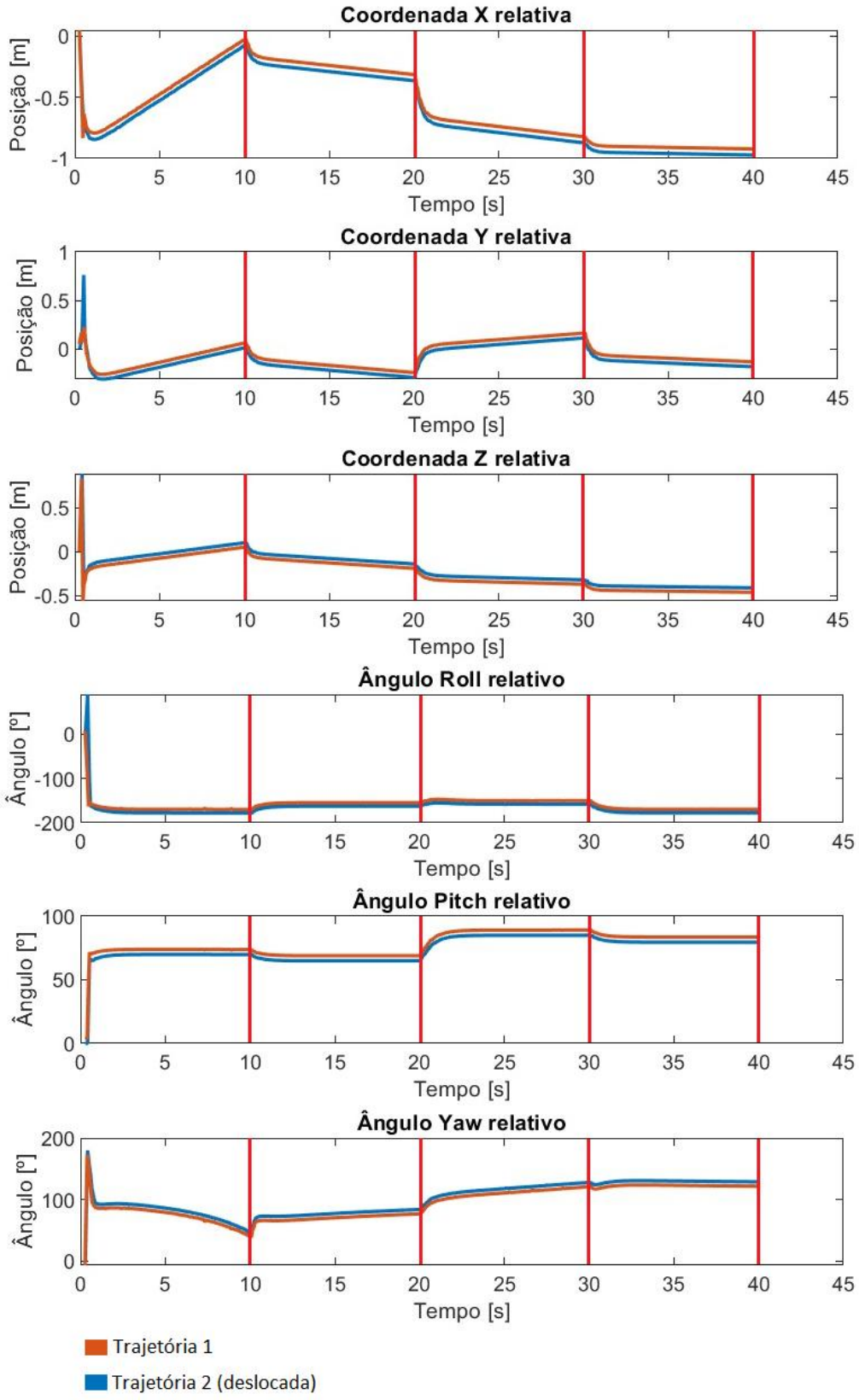


Figura 44 – Pose relativa da câmera com respeito ao marcador durante a simulação com comando de movimento linear *moveL*.

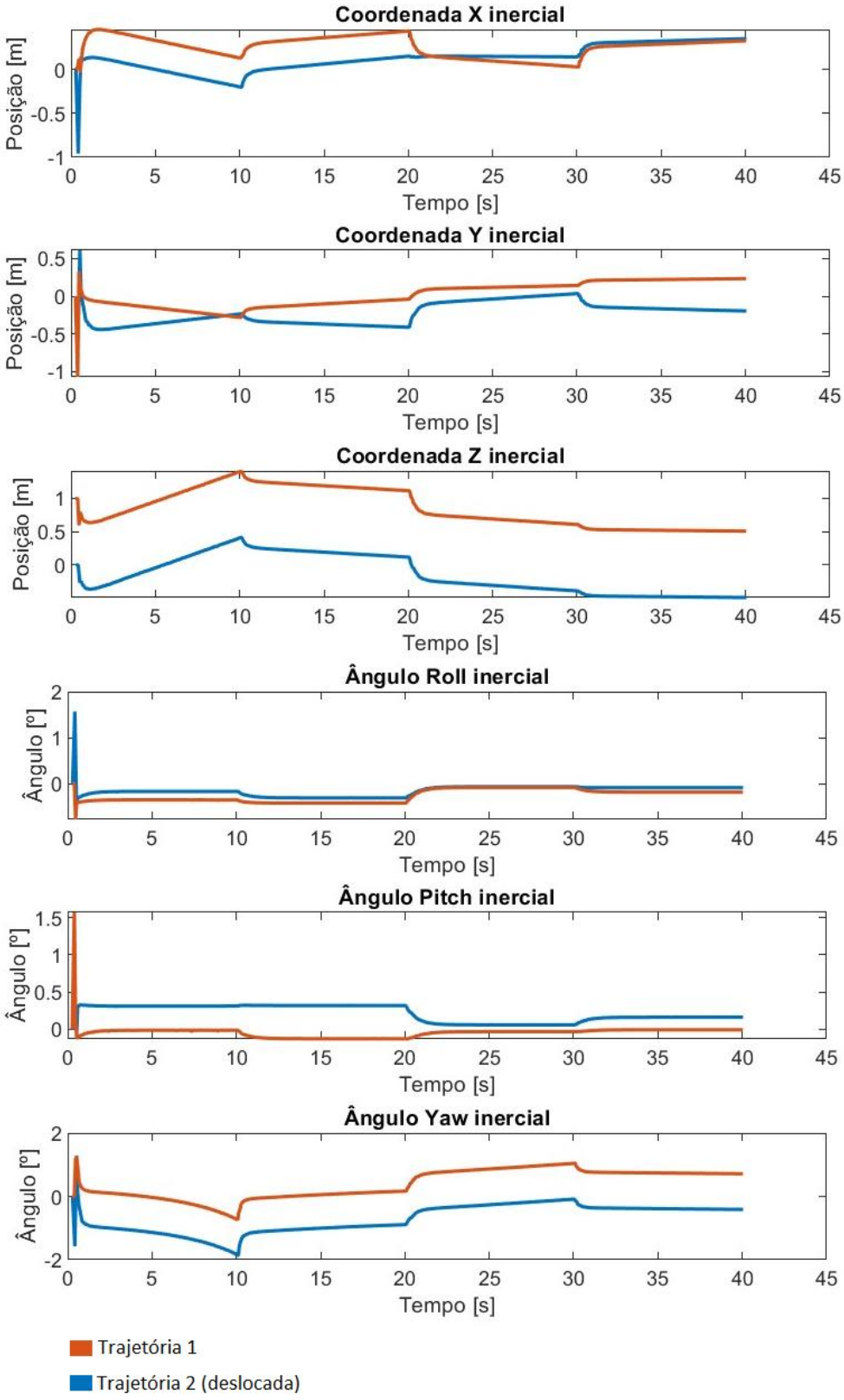
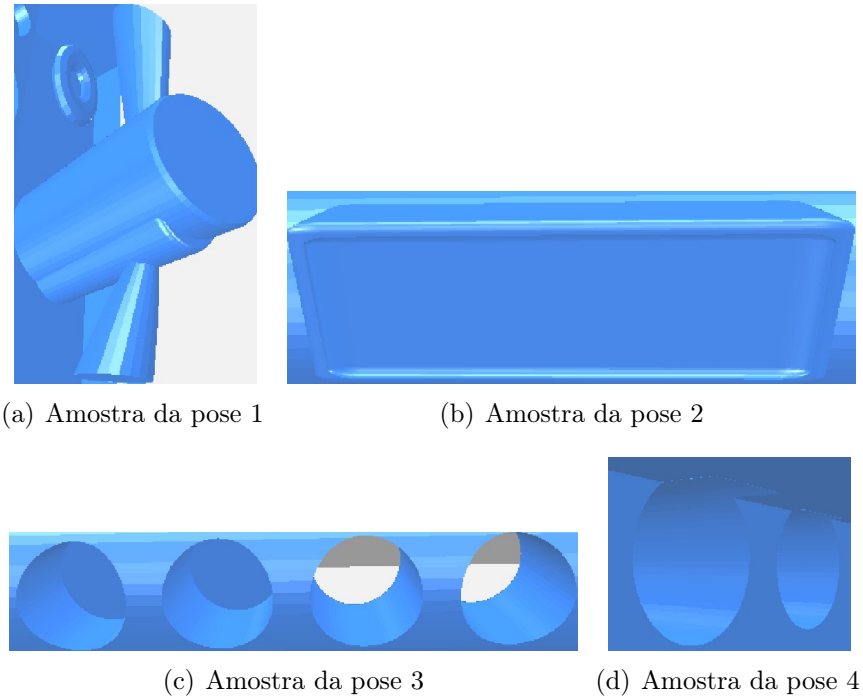


Figura 45 – Pose absoluta da câmera durante a inspeção com o comando de movimento linear *moveL*.



(a) Amostra da pose 1 (b) Amostra da pose 2

(c) Amostra da pose 3 (d) Amostra da pose 4

Figura 46 – Amostras criadas para cada pose de inspeção no ambiente simulado.

Em seguida, a peça foi colocada em uma pose diferente, vista na Figura 43(b) (chamada de pose "deslocada"). A simulação foi realizada da mesma maneira, capturando as imagens da peça durante a inspeção. A Figura 47 mostra as imagens capturadas durante essas simulações, com as regiões mais prováveis de serem encontradas as amostras em verde. Na coluna esquerda encontram-se as imagens capturadas na pose original, e na coluna da direita as imagens capturadas na pose deslocada.

A Tabela 2 apresenta os coeficientes de correlação máximo obtidos em cada pose. Quanto mais próximo de 1, maior a correlação entre a região da imagem e a amostra.

A partir da análise das imagens e das regiões das amostras encontradas, é possível concluir que o sistema foi capaz de realizar o replanejamento das poses de inspeção com sucesso, sendo capaz de realizar a inspeção da mesma região da peça mesmo após uma drástica mudança na pose da mesma. Além disso, a comparação entre os coeficientes de correlação obtidos em cada pose indicam uma alta correlação ($\rho_{12} > 0,99$) mesmo após a peça ser deslocada.

Tabela 2 – Coeficientes de correlação obtidos em cada pose de inspeção em ambiente de simulação.

Pose	ρ_{12} - Pose Original	ρ_{12} - Pose Deslocada
1	0,99972	0,99301
2	0,99998	0,99825
3	0,99995	0,99088
4	0,99995	0,99143

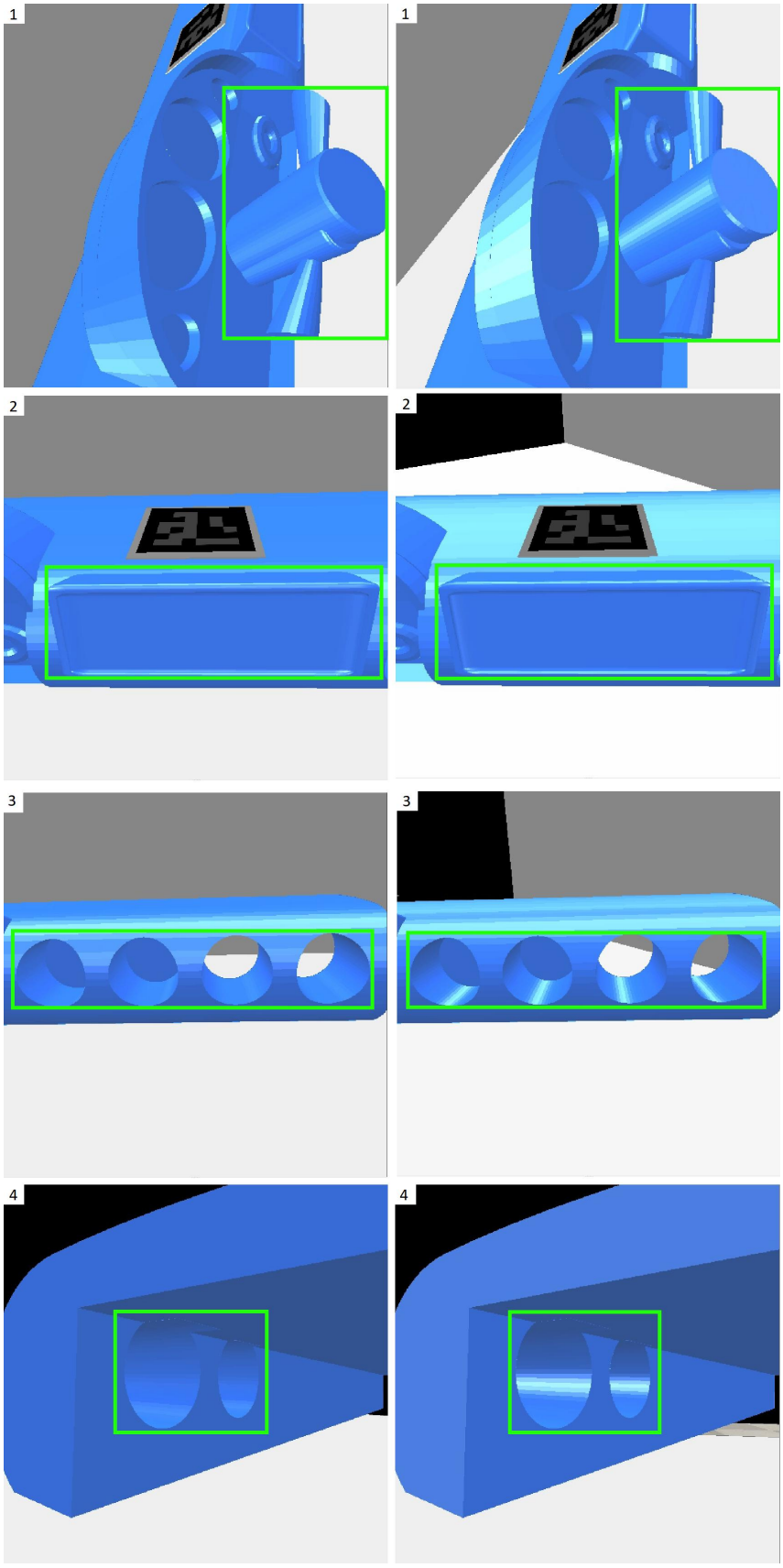


Figura 47 – Imagens capturadas e resultados obtidos durante as inspeções no ambiente simulado.

6.2 Resultados dos Experimentos em Laboratório

Para validar o fluxo de inspeção com o replanejamento baseado em imagem em laboratório, a peça com o marcador foi colocada em diversas posições e orientações diferentes na mesa de trabalho. A peça possui quatro presilhas vermelhas que devem ser fixadas durante a montagem, portanto foram definidas quatro poses de inspeção diferentes para detectar a presença das presilhas. Nos experimentos realizados, a peça foi colocada em cinco poses diferentes. Uma *playlist* contendo os vídeos dos experimentos pode ser acessada por meio do link: https://www.youtube.com/watch?v=2-JvX60Pwhc&list=PL-PUVPVhJCxrssCAlXmzPopyXWMFN3YOn.

Assim como nos experimentos em ambiente de simulação, o algoritmo de *template matching* foi utilizado para validar o replanejamento de poses de inspeção. A área de busca do algoritmo é delimitada como um retângulo na imagem onde se espera encontrar a presilha inspecionada. No sistema original, problemas de mau posicionamento da peça fazem com que os componentes inspecionados fiquem fora do retângulo de busca, gerando falhas. A Figura 48 mostra as amostras criadas para a presilha inspecionada em cada pose. Apesar da detecção de presença da presilha ser uma tarefa relativamente fácil devido ao seu alto contraste, as poses de inspeção foram escolhidas para que o corpo inteiro da presilha esteja completamente visível na imagem.

(a) Amostra da pose 1 (b) Amostra da pose 2 (c) Amostra da pose 3 (d) Amostra da pose 4

Figura 48 – Amostras das presilhas em cada pose de inspeção.

6.2.1 Resultado de Inspeção com o Comando de Movimento *moveJ*

A Figura 49 mostra a pose do marcador com respeito a câmera durante dois experimentos onde a peça estava em diferentes posições e orientações, utilizando movimentos *moveJ*. A Figura 50, por sua vez, mostra a pose da câmera com respeito ao referencial inercial. A partir da análise desses gráficos, é possível perceber que o fluxo proposto foi capaz de alcançar as mesmas poses relativas entre a câmera e o marcador durante a inspeção, apesar da peça estar em diferentes posições. As diferenças nos gráficos da pose absoluta do robô demonstram o deslocamento da peça nos dois experimentos.

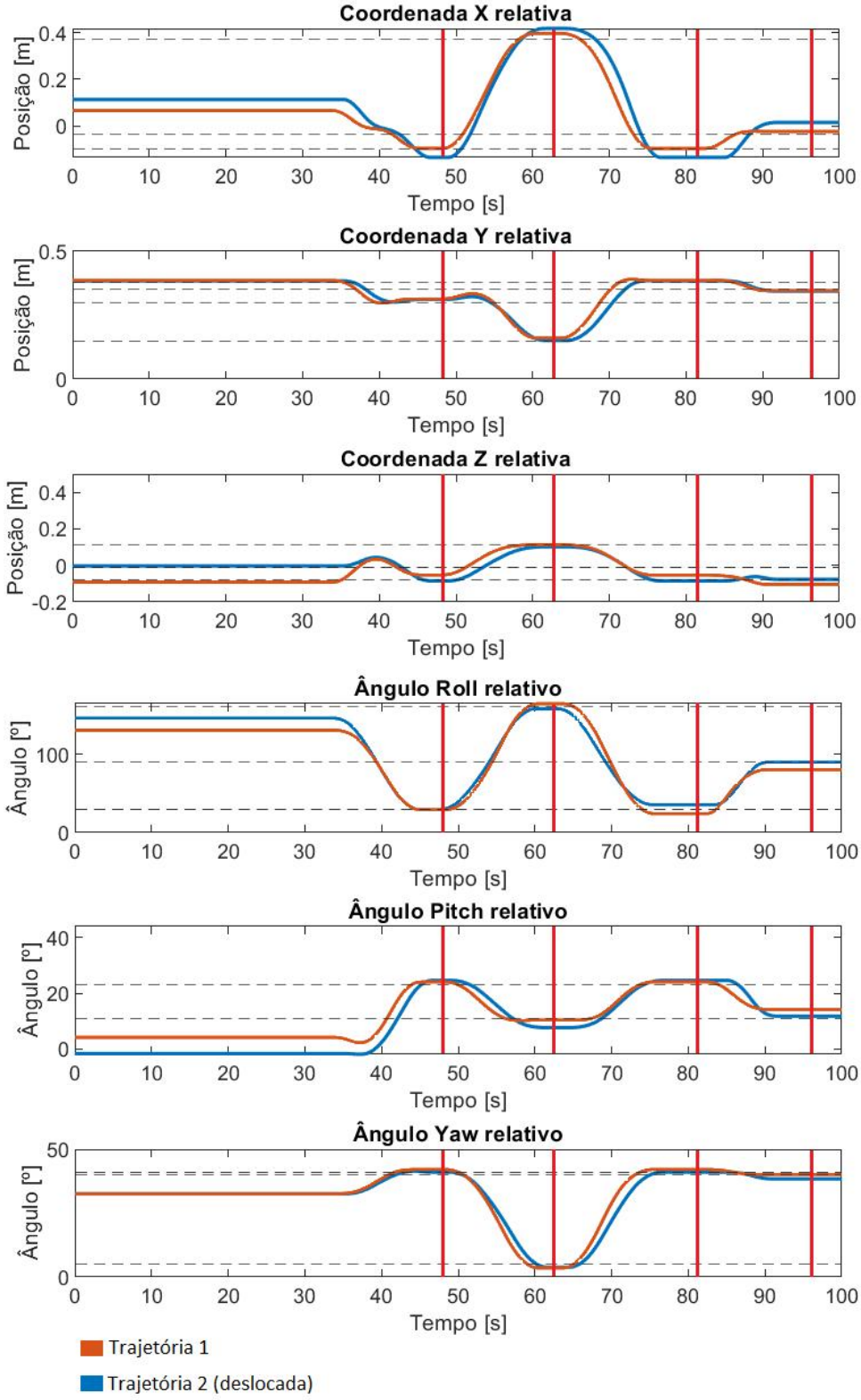


Figura 49 – Pose relativa da câmera com respeito ao marcador durante a inspeção com o comando de movimento no espaço das juntas *moveJ*.

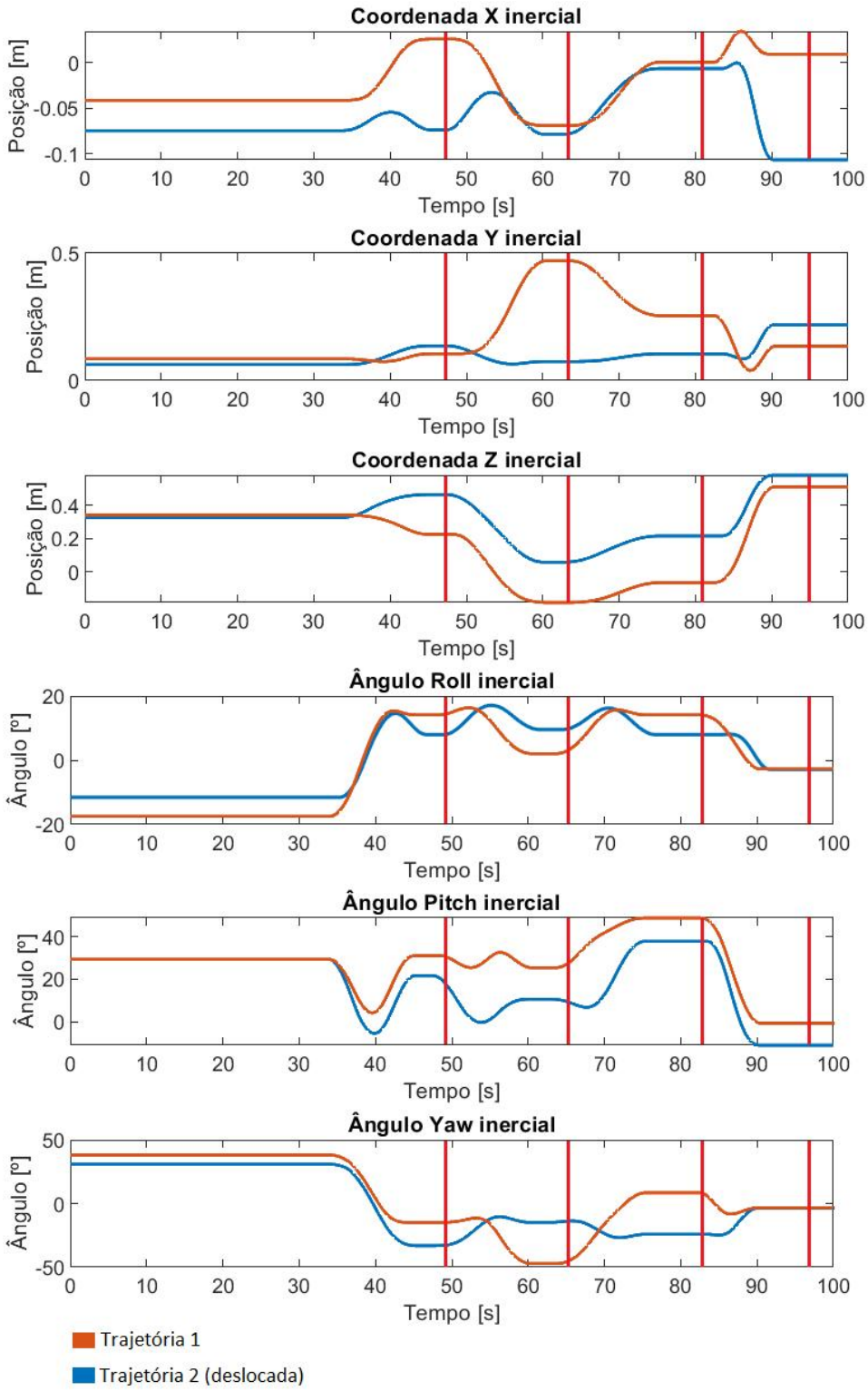


Figura 50 – Pose absoluta da câmera durante a inspeção com o comando de movimento no espaço das juntas *moveJ*.

A Figura 51 mostra os resultados dos algoritmos executados com as imagens capturadas durante a inspeção da peça com *moveJ*, tanto na pose original (coluna esquerda) quanto na pose deslocada pose deslocada (coluna direita). Nas duas poses, os algoritmos detectaram a presilha dentro da área de busca, demonstrando que o sistema de inspeção foi capaz de reposicionar a câmera para capturar a imagem esperada da peça em ambos os casos. A Tabela 3 mostra os coeficientes de correlação obtidos em cada pose durante os experimentos com *moveJ*. Os valores dos coeficientes indicam uma alta correlação mesmo após o deslocamento da peça.

Tabela 3 – Coeficientes de correlação obtidos em cada pose de inspeção em experimentos em laboratório com o comando de movimento *moveJ*.

Pose	ρ_{12} - Pose Original	ρ_{12} - Pose Deslocada
1	0.95739	0.95028
2	0.97064	0.96188
3	0.97505	0.96757
4	0.96331	0.95927

6.2.2 Resultado de Inspeção com o Comando de Movimento *moveL*

Utilizando movimentos do tipo *moveL*, o sistema foi capaz de realizar a inspeção da peça em diferentes poses por meio do replanejamento de tarefa, assim como nos experimentos com o *moveJ*. A Figura 52 mostra as trajetórias realizadas pelo robô com respeito ao marcador durante a inspeção com *moveL*, e a Figura 53 mostra as trajetórias realizadas pelo robô com respeito ao sistema de coordenadas inercial. O formato das trajetórias difere quando comparados ao movimento realizado pelo *moveJ*, porém o robô foi capaz de alcançar as mesmas poses relativas ao marcador para realizar as inspeções.

O robô foi capaz de posicionar a câmera em relação a peça de forma que as presilhas sempre fiquem dentro da área de busca do algoritmo. Dessa forma, em todas as imagens coletadas pelos experimentos, o algoritmo de *template matching* detectou todas as presilhas corretamente. A Figura 54 mostra em verde a região de busca e a presilha que foi detectada pelo algoritmo com a peça nas poses original (coluna esquerda) e deslocada (coluna direita). A Tabela 6.2.2 mostra os coeficientes de correlação obtidos nesses experimentos. Assim como nos experimentos realizados com o comando *moveJ*, os coeficientes indicam uma alta correlação mesmo após o deslocamento da peça.

Os resultados obtidos demonstram que o fluxo de inspeção com replanejamento de tarefa proposto é capaz de realizar a inspeção de peças em diferentes poses dentro do espaço de tarefa. A escolha do comando de movimento *moveL* ou *moveJ* não causou um impacto no resultado dos algoritmos de inspeção, pois a presilha foi encontrada em todas

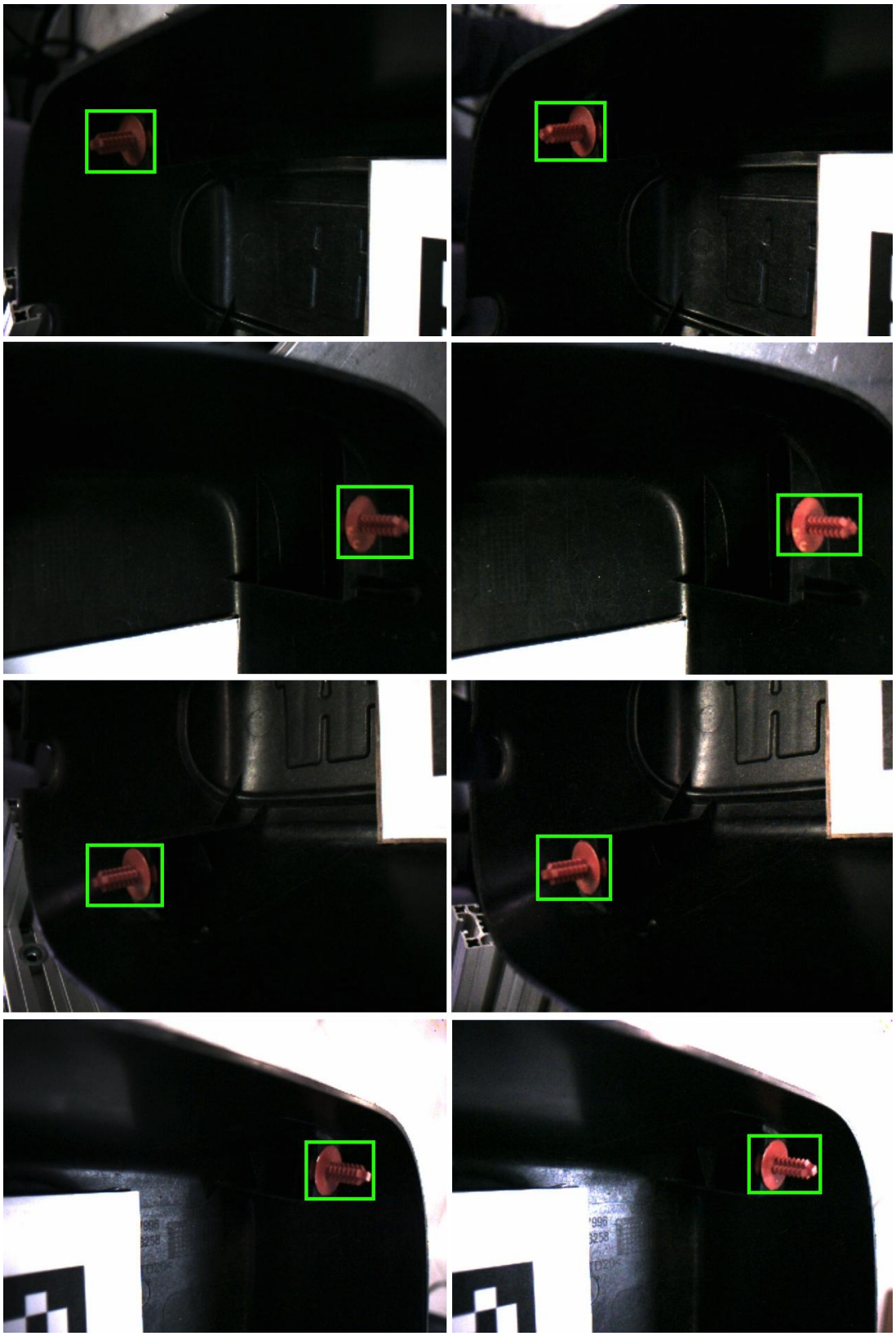


Figura 51 – Resultado do algoritmo de detecção da peça nas poses original e deslocada usando o comando de movimento no espaço das juntas *moveJ*.

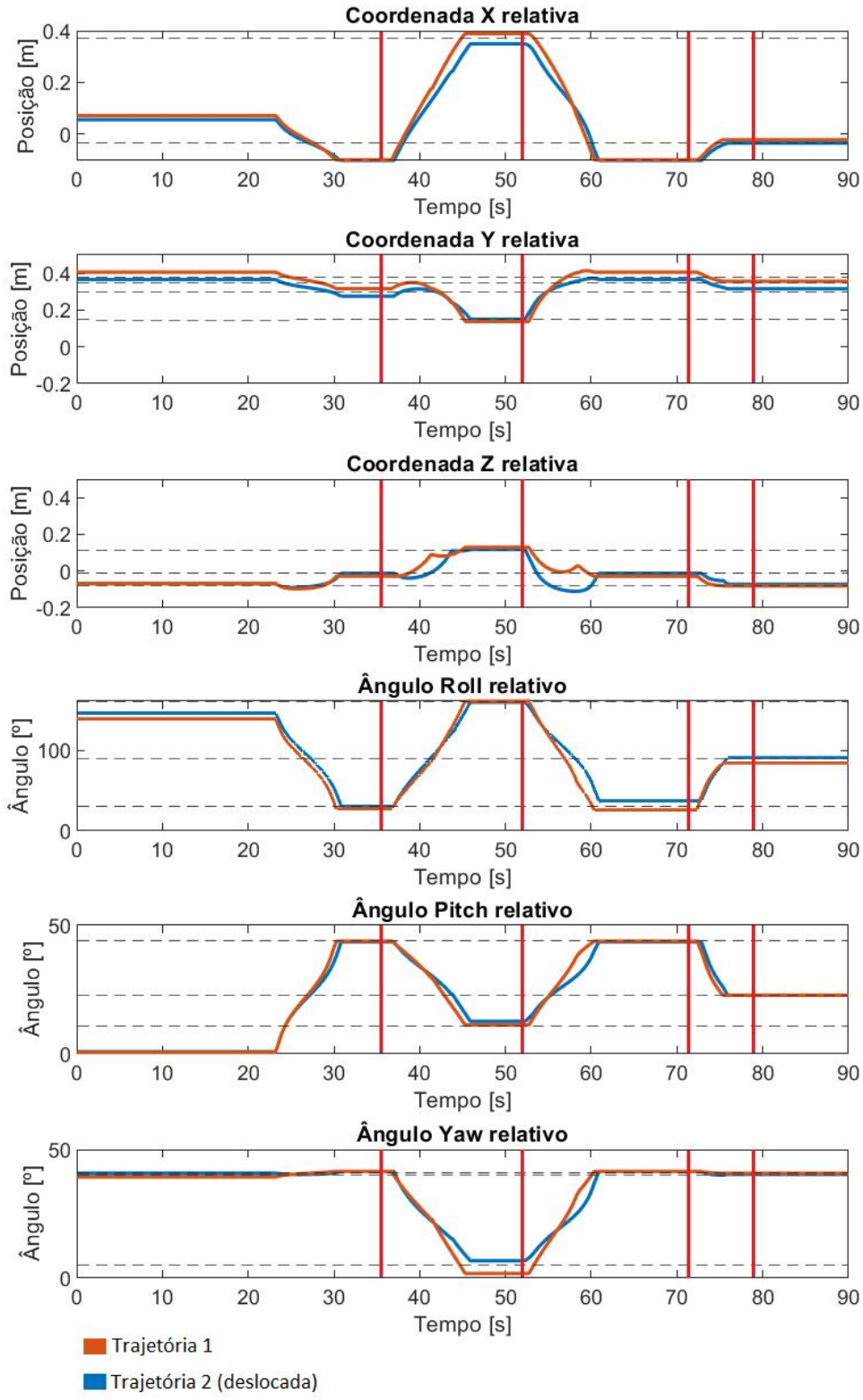


Figura 52 – Pose relativa da câmera com respeito ao marcador durante a inspeção com o comando de movimento Cartesiano *moveL*.

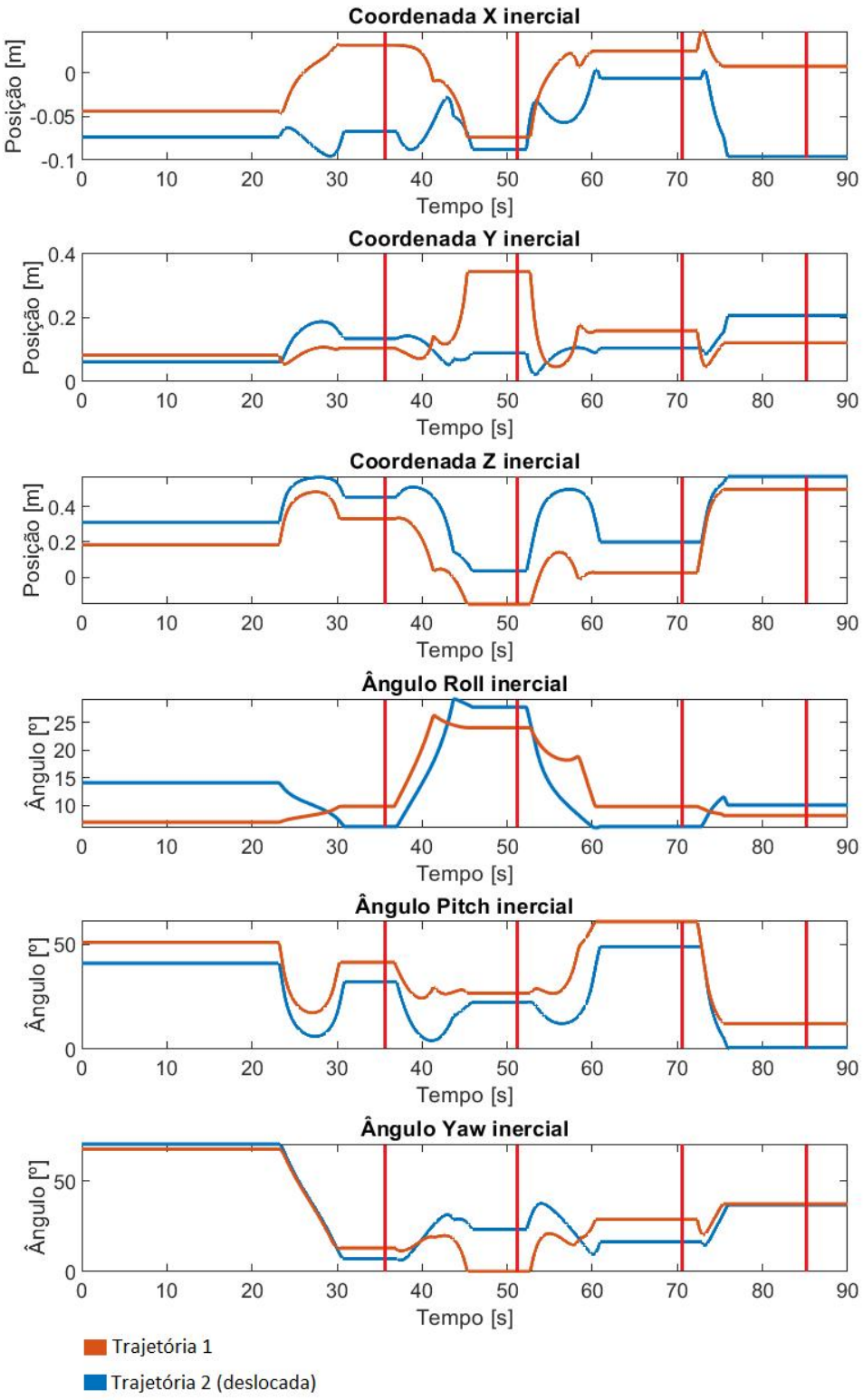


Figura 53 – Pose absoluta da câmera durante a inspeção com o comando de movimento Cartesiano *moveL*.

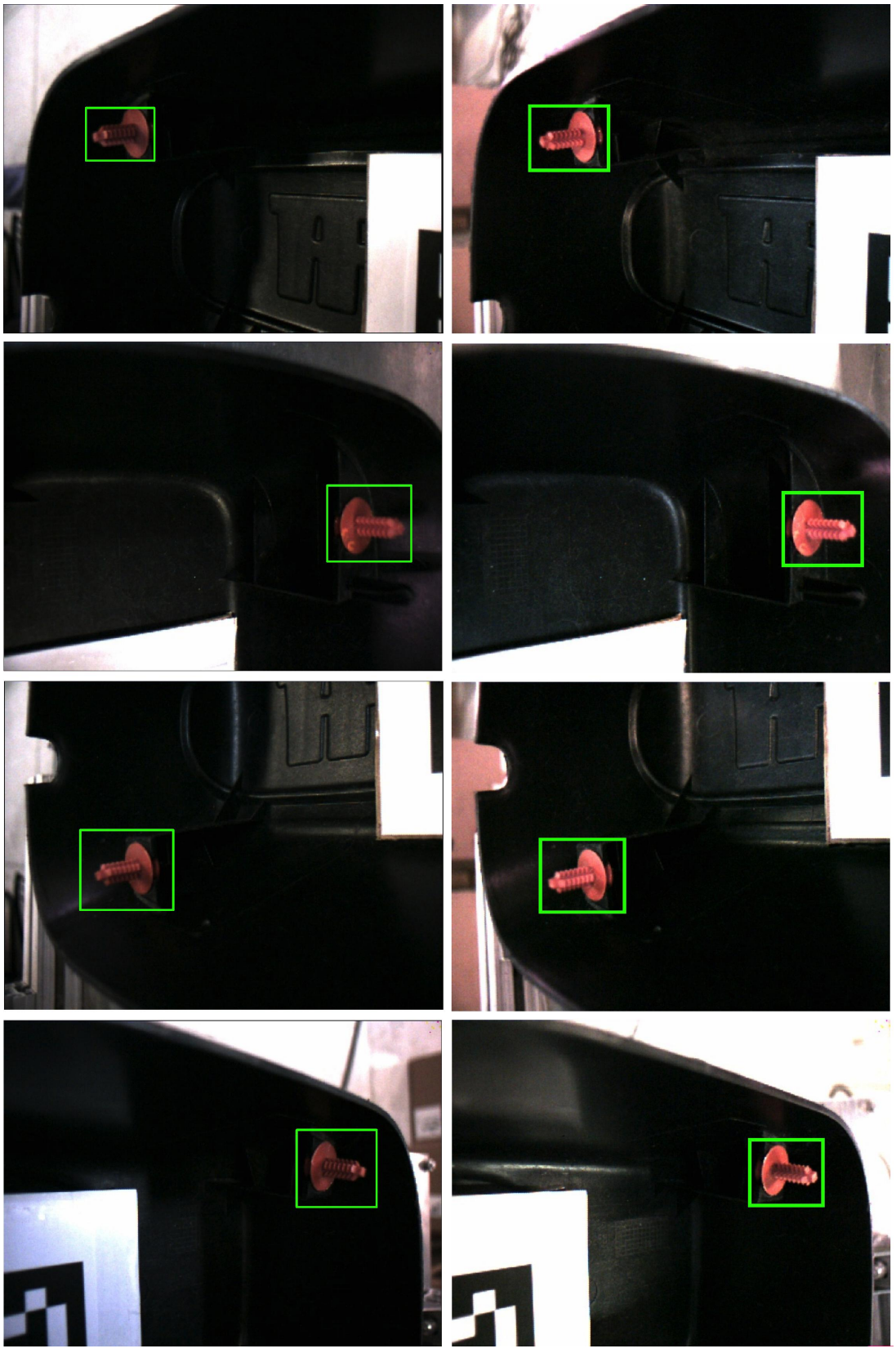


Figura 54 – Resultado do algoritmo de detecção da peça nas poses original e deslocada com o comando de movimento *moveL*.

Tabela 4 – Coeficientes de correlação obtidos em cada pose de inspeção em experimentos em laboratório com o comando de movimento *moveL*.

Pose	ρ_{12} - Pose Original	ρ_{12} - Pose Deslocada
1	0.96159	0.94109
2	0.96597	0.96972
3	0.97779	0.97287
4	0.95509	0.94685

as poses, independente movimento realizado. A análise dos gráficos das Figuras 49 e 52 permite observar que o movimento no espaço das juntas gera uma trajetória mais suave para o efetuador, enquanto o movimento linear pode gerar paradas bruscas, que podem ser mais desgastantes para as juntas, apesar de ter maior previsibilidade nas trajetórias intermediárias. A escolha do movimento utilizado, portanto, deve levar em consideração as vantagens e desvantagens de cada um, bem como as características específicas da tarefa e limitações físicas do ambiente.

Capítulo 7

Conclusões e Trabalhos Futuros

Esta dissertação apresentou um sistema que permite o replanejamento da tarefa de inspeção automática de peças automotivas baseado em visão computacional. Foi apresentada a célula de inspeção industrial Invent Vision, que realiza a inspeção automatizada de peças de uma linha de montagem do setor automotivo utilizando um robô manipulador KUKA KR4 R600 com uma câmera montada em seu efetuador. O robô é utilizado para posicionar a câmera em diversas poses diferentes, de forma a capturar imagens da peça em perspectivas que permitem a detecção de ausência ou presença de componentes, detecção de cores e outros tipos de falhas na produção da peça. O robô possui uma série de poses armazenadas em sua memória e realiza uma rotina que visita essas poses em sequência para realizar a inspeção. A plataforma de inspeção Inspecta é responsável pela execução dos algoritmos de visão de máquina em tempo real.

Por não possuir um *feedback* visual sobre o ambiente da tarefa, um dos problemas recorrentes na linha de produção são atrasos na linha ocasionados por falhas de inspeção decorrentes do mau posicionamento da peça no berço da célula. Senso assim, com o intuito de proporcionar mais versatilidade e assertividade ao sistema, foi proposta uma melhoria no fluxo de inspeção, utilizando uma técnica de Controle Servo Visual Baseado em Posição, para replanejar as poses de inspeção do robô que são dadas com relação à peça. Uma rotina de busca da peça foi implementada, na qual o robô percorre o espaço de tarefa até que um marcador ArUco seja detectado, e sua pose em relação à câmera é estimada e refinada resolvendo o problema de PnP e um filtro de média. Nesse ponto, o robô executa uma série de movimentos que levam a câmera para uma sequência de poses desejadas que são dadas relativamente entre a câmera e a peça, eliminando variabilidades decorrentes do mau posicionamento da peça.

A construção dos cenários em ambiente de simulação é uma ferramenta que permite a realização de um estudo prévio acerca da viabilidade da inspeção de uma peça. O modelo da peça pode ser importado dentro da cena e diversas câmeras podem ser acopladas à célula (câmeras fixas) e ao robô (câmeras móveis), capturando imagens da

peça nas perspectivas desejadas. Isso acelera o projeto de uma nova célula e também permite o conhecimento prévio das poses de inspeção do robô mesmo que o robô ainda não esteja fisicamente disponível. Além disso, a implementação do fluxo de inspeção proposto em ambiente de simulação também permite avaliar o comportamento do sistema em caso de mau posicionamento da peça.

Os experimentos realizados demonstram o sucesso da solução proposta, sendo capaz de realizar a inspeção de todos os componentes de uma peça com diversas posições e orientações diferentes na superfície de trabalho. Ou seja, o robô foi capaz de posicionar a câmera na mesma pose relativa à peça em todos os cenários experimentais. As imagens capturadas durante os experimentos foram testadas utilizando os mesmos algoritmos de detecção utilizados nos sistemas em produção, e os resultados demonstraram que os componentes inspecionados estavam presentes nas regiões esperadas dentro das imagens capturadas pela câmera com o fluxo proposto. Com isso, o sistema se tornou mais confiável contra falhas devido ao posicionamento da peça.

Uma vantagem da solução proposta é que ela não utiliza nenhum tipo de componente além dos que já existem nas células, e também não necessita de nenhuma mudança estrutural na montagem da máquina, sendo uma solução de baixo custo financeiro que poderá ser incluída tanto nas células em produção como nas que ainda serão construídas. Além disso, a utilização do ROS, que possui como linguagens nativas C++ e Python, permite a fácil integração com a plataforma de inspeção Inspecta, que é escrita em linguagem C++. O sistema proposto foi encapsulado em um pacote ROS e disponibilizado online pelo link: `https://github.com/lucca-leao/pbvs`.

Trabalhos Futuros

A estratégia proposta nesta dissertação para a expansão e melhoria do fluxo de inspeção automático de uma peça utilizando informações visuais do objeto tem como base a estimação da pose do objeto em relação à câmera a partir da detecção de marcadores fiduciais ArUco acoplados na peça. Um trabalho futuro que visa a melhoria desse processo consiste na substituição do algoritmo de detecção do ArUco por um algoritmo mais customizado capaz de detectar e estimar a pose do objeto a partir de uma etiqueta identificadora da peça, que já é utilizada dentro da linha para rastreamento da peça, sendo composto por um simples QR Code. Para esse fim, um algoritmo promissor é proposto em [Cao et al., 2019].

O arcabouço experimental construído para a validação da solução proposta integra o ROS com diversos *drivers* que permitem que o robô realize uma rotina visitando todas as poses de inspeção que são corrigidas a partir da pose estimada da peça. A validação dos algoritmos de inspeção com o sistema foi feita *offline* em uma etapa posterior à realização

da rotina. Com isso, um trabalho futuro que integra o sistema proposto à plataforma de inspeção que executa os algoritmos em tempo real é necessário para que experimentos em tempo real sejam realizados e para que o sistema seja homologado em produção.

Uma premissa adotada nesta dissertação é que a peça permanece estática durante toda a inspeção. Isso permite que um filtro muito simples, como o filtro de média, seja utilizado na etapa de estimação da pose da peça. Portanto, um trabalho futuro que permitirá a quebra dessa premissa é a utilização de um filtro de estados como um Filtro de Kalman para estimar a pose do objeto em relação à câmera.

Referências

G. Alenyà, S. Foix, and C. Torras. Tof cameras for active vision in robotics. *Sensors and Actuators A: Physical*, 218:10–22, 2014.

P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, 1993.

K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

R. Brunelli. *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

X. Cao, Y. Yang, T. Lu, L. Fang, and J. Zhang. A fast pose estimation method based on new qr code for location of indoor mobile robot. In *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 257–262. IEEE, 2019.

F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The confluence of vision and control*, pages 66–78. Springer, 2007.

P. I. Corke. A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1): 24–32, 1996.

P. I. Corke and S. A. Hutchinson. A new hybrid image-based visual servo control scheme. In *Proceedings of the 39th IEEE conference on decision and control (Cat. No. 00CH37187)*, volume 3, pages 2521–2526. IEEE, 2000.

K. Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research*, 18(3):286–298, 1999.

M. Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596. IEEE, 2005.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

S. Fuchs and S. May. Calibration and registration for precise surface reconstruction with time-of-flight cameras. *International Journal of Intelligent Systems Technologies and Applications*, 5(3-4):274–284, 2008.

S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

G. D. Hager, W.-C. Chang, and A. S. Morse. Robot hand-eye coordination based on stereo vision. *IEEE Control Systems Magazine*, 15(1):30–39, 1995.

R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

K. Hashimoto and T. Noritsugu. Performance and sensitivity in visual servoing. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 3, pages 2321–2326. IEEE, 1998.

J. Hefele and C. Brenner. Robot pose correction using photogrammetric tracking. In *Machine Vision and Three-Dimensional Imaging Systems for Inspection and Metrology*, volume 4189, pages 170–178. SPIE, 2001.

S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*, pages 85–94. IEEE, 1999.

D. Kragic, H. I. Christensen, et al. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, 15:2002, 2002.

Kuka Robotics. KUKA KR4 R600 Datasheet. Number 0000-356-043, 2021.

V. Kuts, T. Tahemaa, T. Otto, M. Sarkans, and H. Lend. Robot manipulator usage for measurement in production areas. *Journal of Machine Engineering*, 16, 2016.

R. Labudzki and S. Legutko. Applications of machine vision. *Manufact Ind Eng*, 2:27–29, 2011.

H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann. Industry 4.0. *Business & information systems engineering*, 6(4):239–242, 2014.

L. G. Leao, G. M. Freitas, L. F. E. Moreira, and A. O. Fernandes. Aplicaçao de um manipulador robótico para automatizaçao de uma célula de inspeçao de peças automotivas. *XXIV Congresso Brasileiro de Automática - CBA 2022*, 2022.

E. Leikas. Robot guidance with a photogrammetric 3-d measuring system. *Industrial Robot: An International Journal*, 26(2):105–108, 1999.

H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

T. Luhmann. Close range photogrammetry for industrial applications. *ISPRS journal of photogrammetry and remote sensing*, 65(6):558–569, 2010.

A. Maldonado, U. Klank, and M. Beetz. Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2586–2591. IEEE, 2010.

E. Malis, F. Chaumette, and S. Boudet. Positioning a coarse-calibrated camera with respect to an unknown object by 2d 1/2 visual servoing. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, volume 2, pages 1352–1359. IEEE, 1998.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

C. Mineo, S. Pierce, B. Wright, I. Cooper, and P. Nicholson. Paut inspection of complex-shaped composite materials through six dofs robotic manipulators. *Insight-Non-Destructive Testing and Condition Monitoring*, 57(3):161–166, 2015.

J.-K. Oh, G. Jang, S. Oh, J. H. Lee, B.-J. Yi, Y. S. Moon, J. S. Lee, and Y. Choi. Bridge inspection robot system with machine vision. *Automation in Construction*, 18 (7):929–941, 2009.

E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

N. P. Papanikolopoulos and P. K. Khosla. Adaptive robotic visual tracking: Theory and experiments. *IEEE Transactions on Automatic Control*, 38(3):429–445, 1993.

K. K. Patel, A. Kar, S. Jha, and M. Khan. Machine vision system: a tool for quality inspection of food and agricultural products. *Journal of food science and technology*, 49 (2):123–141, 2012.

L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. F. García. Robot guidance using machine vision techniques in industrial environments: A comparative review. *Sensors*, 16(3):335, 2016.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

K. Rahardja and A. Kosaka. Vision-based bin-picking: Recognition and localization of multiple complex objects using simple visual cues. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, volume 3, pages 1448–1457. IEEE, 1996.

Z. Ren, F. Fang, N. Yan, and Y. Wu. State of the art in defect detection based on machine vision. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 9(2):661–691, 2022.

E. Rohmer, S. P. Singh, and M. Freese. V-REP: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE, 2013.

L. Sciavicco, B. Siciliano, L. Villani, and G. Oriolo. Robotics: Modelling, planning and control, ser. advanced textbooks in control and signal processing, 2011.

A. Shafi. Machine vision in automotive manufacturing. *Sensor Review*, 2004.

Y. C. Shiu and S. Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form ax= xb. 1987.

M. W. Spong, S. Hutchinson, M. Vidyasagar, et al. *Robot modeling and control*, volume 3. Wiley New York, 2006.

J. Sturm, K. Konolige, C. Stachniss, and W. Burgard. 3d pose estimation, tracking and model learning of articulated objects from dense depth video using projected texture stereo. In *RGB-D: Advanced Reasoning with Depth Cameras Workshop, RSS*, 2010.

R. Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

R. Y. Tsai, R. K. Lenz, et al. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3): 345–358, 1989.

W. J. Wilson, C. W. Hulls, and G. S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5): 684–696, 1996.

C. Wöhler. *3D computer vision: efficient methods and applications*. Springer Science & Business Media, 2012.

X. Xiao, J. Dufek, and R. Murphy. Visual servoing for teleoperation using a tethered uav. In *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pages 147–152. IEEE, 2017.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.