Pontifícia Universidade Católica
do Rio de Janeiro

Robótica Industrial

ENG1458

Mapeamento Visão-Ação em Robótica: Da Seleção de Pixel 2D ao Posicionamento 3D de um Manipulador

Aluno(a) Paloma Fernanda Loureiro Sette - 1621595
Professor(a) Wouter Caarls

Rio de Janeiro, 15 de Julho de 2025

Sumário

Lista de Figuras

Resumo

Este relatório detalha o projeto, desenvolvimento e implementação de um sistema integrado de robótica e visão computacional para o controle de um manipulador robótico Pincher de 4 graus de liberdade. O objetivo central foi criar uma plataforma *eye-in-hand* capaz de traduzir a seleção de um ponto em uma imagem bidimensional para um movimento preciso no espaço tridimensional. A metodologia empregou uma câmera de profundidade Luxonis OAK-D [4] para a aquisição de dados visuais e de profundidade, e uma placa de controle Arbotix-M [5] para a interface com os servos Dynamixel [6] do robô. O pipeline de *software*, desenvolvido em Python com as bibliotecas DepthAI [7], OpenCV [8] e NumPy [9], executa a cadeia completa de transformações: a partir de um pixel selecionado pelo usuário, o sistema calcula as coordenadas 3D no referencial da câmera (`p_cam`) utilizando a matriz de parâmetros intrínsecos. Subsequentemente, estas coordenadas são transformadas para o referencial do robô (`p_robot`) através de uma matriz de transformação extrínseca estática, que foi definida para a configuração física do sistema. A cinemática inversa do manipulador é então resolvida para determinar os ângulos das juntas (q), que são enviados aos atuadores para executar o movimento. Os resultados demonstram o sucesso funcional do sistema em atingir o posicionamento ponto-a-ponto, validando a correção matemática das transformações e a integração hardware-software. Limitações, como a ausência de controle de orientação do efetuador com base na normal da superfície do alvo, são discutidas como decisões pragmáticas de engenharia. Por fim, são delineados trabalhos futuros, incluindo a implementação de uma cinemática inversa mais robusta e a aplicação do sistema em tarefas de inspeção acústica de superfícies.

1 Introdução

1.1 Contextualização

A intersecção entre a robótica manipuladora e a visão computacional representa uma das fronteiras mais dinâmicas da automação moderna. Em ambientes não estruturados, onde a posição de objetos e superfícies não é previamente conhecida, a capacidade de um robô "ver"e interpretar seu entorno é fundamental para a execução de tarefas complexas. Sistemas que integram percepção visual 3D permitem que manipuladores robóticos transcendam a simples repetição de trajetórias pré-programadas, adaptando-se dinamicamente a variações no ambiente. Esta capacidade é um pilar para aplicações na Indústria 4.0, como montagem flexível, pick-and-place inteligente, inspeção de qualidade e interação segura com humanos. O presente trabalho se insere neste contexto, explorando os desafios e soluções na criação de um sistema onde a visão guia a ação de forma direta e intuitiva.

1.2 Definição do Problema

O desafio central deste projeto reside em desenvolver um sistema de controle de malha aberta (do ponto de vista da visão) onde um operador humano possa designar um alvo no campo de visão de uma câmera e o braço robótico alcance este alvo no espaço físico. Isso requer a solução de um problema de correspondência fundamental: a conversão de coordenadas de um espaço 2D (a imagem da câmera) para um espaço 3D (o mundo do robô). A solução envolve uma cadeia de transformações geométricas e cinemáticas, desde a projeção de um pixel para um raio no espaço, passando pela medição de profundidade para obter um ponto tridimensional, até a tradução deste ponto entre diferentes sistemas de coordenadas e, finalmente, o cálculo dos comandos motores para o robô. A precisão de todo o sistema é dependente da exatidão de cada etapa desta cadeia de transformação.

2 Objetivo

O desenvolvimento deste projeto foi norteado pelos seguintes objetivos:

- Objetivo Principal: Implementar um sistema integrado e funcional, demonstrando o ciclo completo de percepção-ação, capaz de:

    1. Adquirir e alinhar imagens RGB e mapas de profundidade de uma cena em tempo real utilizando a câmera OAK-D.
    2. Desenvolver uma interface gráfica que permita a um usuário selecionar um Ponto de Interesse (PdI) através de um clique do mouse na imagem visualizada.
    3. Calcular as coordenadas 3D do PdI no referencial do robô (p_robot) a partir das coordenadas de pixel e do valor de profundidade medido.
    4. Comandar o braço robótico Pincher para mover seu efetuador final até as coordenadas 3D calculadas, validando a correção do pipeline de transformação.

- Objetivos Secundários: Investigar teoricamente os requisitos para o controle de orientação do efetuador final, especificamente o alinhamento perpendicular à superfície do alvo, analisando as limitações do modelo cinemático existente e as modificações necessárias para tal implementação.

3 Estrutura do Documento

Este relatório está organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica sobre robótica manipuladora e visão computacional que embasa o projeto. A Seção 3 detalha a metodologia, descrevendo o hardware utilizado e a arquitetura de *software*, com ênfase no pipeline de transformação de coordenadas. A Seção 4 apresenta os resultados obtidos, incluindo a validação funcional do sistema. A Seção 5 realiza uma discussão crítica dos resultados, analisando os desafios de implementação e as decisões de engenharia tomadas. Finalmente, a Seção 6 conclui o trabalho e propõe direções para desenvolvimentos futuros.

4 Fundamentação Teórica

Para a completa compreensão do sistema desenvolvido, esta seção apresenta os conceitos fundamentais que governam a operação de manipuladores robóticos e a extração de informação tridimensional a partir de sistemas de visão. São abordados os modelos cinemáticos que descrevem o movimento do robô e os princípios de calibração de câmeras que permitem a fusão de dados entre os domínios visual e físico.

4.1 Robótica Manipuladora

Um manipulador robótico é formalmente definido como uma cadeia cinemática aberta, composta por uma série de corpos rígidos, denominados **elos** (*links*), conectados por articulações motorizadas, conhecidas como **juntas** (*joints*). As juntas podem ser de dois tipos principais: de revolução (rotacionais) ou prismáticas (lineares). O manipulador Pincher [6], utilizado neste trabalho, é composto exclusivamente por juntas de revolução. O número de juntas independentes define os **graus de liberdade** (*Degrees of Freedom - DoF*) do robô, que correspondem ao número de parâmetros necessários para descrever sua configuração de forma única.
É crucial distinguir entre dois espaços de representação [10]:

- **Espaço de Juntas:** Um espaço n-dimensional, onde 'n' é o número de graus de liberdade, definido pelo vetor de variáveis de junta, $\mathbf{q} = [q_1, q_2, ..., q_n]^T$. É o espaço nativo no qual os comandos de controle dos motores são executados.
- **Espaço Cartesiano (ou Espaço de Tarefa):** O espaço tridimensional euclidiano (geralmente $\mathbb{R}^3$ para posição e $SO(3)$ para orientação) no qual o efetuador final do robô opera para executar uma tarefa.

A principal problemática da cinemática de manipuladores reside em estabelecer o mapeamento matemático entre estes dois espaços.

4.1.1 Estrutura e Componentes

Um manipulador robótico é formalmente definido como uma cadeia cinemática aberta, composta por uma série de corpos rígidos, denominados **elos** (*links*), conectados por articulações motorizadas, conhecidas como **juntas** (*joints*). As juntas permitem o movimento relativo entre os elos e são, em sua maioria, de dois tipos: de revolução (que produzem rotação) ou prismáticas (que produzem translação). O manipulador Pincher, objeto deste trabalho, é um exemplo de robô serial composto exclusivamente por juntas de revolução. O número total de variáveis independentes necessárias para definir completamente a posição e a orientação de todas as partes do manipulador é conhecido como seus **graus de**

liberdade (*Degrees of Freedom - DoF*). Em um manipulador de cadeia aberta simples, o número de graus de liberdade é igual ao número de juntas.
A descrição da configuração do manipulador pode ser realizada em dois domínios distintos e fundamentais:

- **Espaço de Juntas:** É o espaço n-dimensional, onde 'n' é o número de graus de liberdade do robô. Uma configuração neste espaço é representada por um vetor $\mathbf{q} = [q_1, q_2, ..., q_n]^T$, cujos componentes são os valores angulares (para juntas de revolução) ou lineares (para juntas prismáticas) de cada junta. Este é o espaço de configuração nativo do robô.
- **Espaço Cartesiano:** Também conhecido como Espaço de Tarefa, é o espaço tridimensional euclidiano no qual o órgão terminal do robô, ou **efetuador final** (*end-effector*), executa suas tarefas. A configuração do efetuador neste espaço é descrita por sua **pose**, que consiste em sua posição (x, y, z) e orientação (geralmente representada por três ângulos, como roll, pitch, yaw).

A análise cinemática de um manipulador, que será detalhada a seguir, tem como objetivo central estabelecer as relações matemáticas para o mapeamento entre o espaço de juntas e o espaço cartesiano.

4.1.2 Cinemática Direta (*Forward Kinematics*)

O problema da cinemática direta (CD) consiste em determinar a posição e orientação (denominadas conjuntamente como **pose**) do efetuador final do robô, dado um vetor de configuração no espaço de juntas, $\mathbf{q}$. Este é um problema com solução única e bem definida. A ferramenta matemática padrão para esta tarefa é o uso de **matrizes de transformação homogênea**.
Uma transformação homogênea $\mathbf{T}$ é uma matriz 4×4 que representa a pose de um sistema de coordenadas em relação a outro. Ela é composta por uma submatriz de rotação $\mathbf{R}$ (3×3) e um vetor de translação $\mathbf{p}$ (3×1):

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$$

Utilizando uma metodologia sistemática, como a convenção de Denavit-Hartenberg (DH), atribui-se um sistema de coordenadas a cada elo do robô. A transformação total da base até o efetuador final é então obtida pelo produto sequencial das matrizes de transformação entre os elos adjacentes:

$$\mathbf{T}_n^0(\mathbf{q}) = \mathbf{T}_1^0(q_1)\mathbf{T}_2^1(q_2)\cdots\mathbf{T}_n^{n-1}(q_n)$$

Onde $\mathbf{T}_n^0$ é a pose do efetuador final (elo n) em relação à base (elo 0). Este é o princípio que fundamenta a função `fk(q)` no modelo de software do robô.

4.1.3 Cinemática Inversa (*Inverse Kinematics*)

O problema da cinemática inversa (CI) é consideravelmente mais complexo e de maior importância prática para o controle. Ele busca responder à pergunta: dada uma pose desejada para o efetuador final no espaço cartesiano, $\mathbf{T}_d$, qual é o vetor de ângulos de junta, $\mathbf{q}$, necessário para alcançá-la?
Ao contrário da CD, o problema da CI pode não ter solução (se o alvo estiver fora do alcance), ter um número finito de soluções ou até mesmo um número infinito de soluções (em caso de robôs redundantes). Existem duas abordagens principais para resolver a CI:

- **Soluções Analíticas (ou Algébricas):** Para robôs com poucos graus de liberdade e geometria simples, como o Pincher, é frequentemente possível derivar um conjunto de equações em forma fechada que mapeiam diretamente a pose cartesiana para os ângulos das juntas. Esta abordagem é computacionalmente muito eficiente e, quando disponível, é preferível. O método `ik(p_target)` implementado no software deste projeto utiliza esta abordagem.

- **Soluções Numéricas (ou Iterativas):** Para robôs mais complexos, uma solução analítica é inviável. Nesses casos, métodos numéricos são empregados, os quais partem de uma configuração inicial e iterativamente ajustam os ângulos das juntas para minimizar o erro entre a pose atual e a pose desejada. A ferramenta fundamental para estes métodos é a matriz **Jacobiana** ($\mathbf{J}(\mathbf{q})$), que relaciona as velocidades no espaço de juntas com as velocidades no espaço cartesiano ($\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$). Embora não implementada neste trabalho, esta abordagem é o padrão da indústria para robôs de alta complexidade.

4.2 Visão Computacional para Robótica

A visão computacional atua como o principal sistema sensorial para manipuladores robóticos autônomos, provendo a informação necessária para a percepção do ambiente. A capacidade de extrair dados tridimensionais de uma cena permite que o robô localize alvos e interaja com objetos cujas posições não são previamente conhecidas, habilitando tarefas em ambientes dinâmicos e não estruturados.

4.2.1 Percepção de Profundidade com Visão Estéreo

A estereoscopia computacional é uma técnica de percepção de profundidade que emula a visão binocular humana. O princípio se baseia no uso de duas ou mais câmeras posicionadas a uma distância conhecida uma da outra — a **linha de base** (*baseline*) — para capturar imagens simultâneas de uma mesma cena a partir de pontos de vista ligeiramente distintos.
Para um ponto qualquer no espaço 3D que é visível em ambas as imagens, a diferença em suas coordenadas horizontais nas duas imagens é denominada **disparidade**. A relação fundamental da visão estéreo é que a disparidade de um ponto é inversamente proporcional à sua distância (profundidade) da câmera. Pontos mais próximos exibem uma disparidade maior, enquanto pontos mais distantes apresentam uma disparidade menor.
O processo de obtenção de profundidade consiste em duas etapas principais:

1. **Correspondência Estéreo:** Um algoritmo computacional busca por correspondências entre as imagens esquerda e direita para encontrar os pixels que representam o mesmo ponto 3D. O resultado deste processo é um **mapa de disparidade**.

2. **Triangulação:** Utilizando a geometria do sistema estéreo (distância focal e linha de base), o mapa de disparidade é convertido em um **mapa de profundidade**, onde o valor de cada pixel representa a distância do objeto correspondente até o plano da câmera.

Este é o princípio de funcionamento empregado pela câmera Luxonis OAK-D [4] para fornecer dados de profundidade em tempo real.

4.2.2 Calibração de Câmeras

A calibração de uma câmera é o processo de estimar os parâmetros que definem o seu modelo matemático. O objetivo é estabelecer uma relação precisa entre as coordenadas 3D de um ponto no mundo e as coordenadas 2D (em pixels) de sua projeção na imagem. Estes parâmetros são divididos em duas categorias: intrínsecos e extrínsecos.

Parâmetros Intrínsecos Os parâmetros intrínsecos modelam as características ópticas, geométricas e de fabricação internas da câmera. São eles: a **distância focal** em pixels (f_x, f_y) e o **ponto principal** (c_x, c_y), que é a coordenada do pixel onde o eixo óptico da câmera intercepta o plano da imagem. Estes parâmetros são agrupados em uma matriz 3×3 denominada **matriz de calibração intrínseca**, $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

A matriz $\mathbf{K}$ permite a projeção de pontos 3D do referencial da câmera para o plano da imagem 2D. Inversamente, sua inversa, $\mathbf{K}^{-1}$, é utilizada no processo de retroprojeção: a conversão de uma coordenada de pixel para um raio no espaço 3D, uma operação fundamental para este projeto.
No contexto deste projeto, o sistema de coordenadas do mundo é definido como o sistema de coordenadas da base do robô. Portanto, o conjunto de parâmetros extrínsecos de maior criticidade é aquele que define a transformação do referencial da câmera para o referencial do robô, aqui denotado como $\mathbf{T}_{\text{cam}\to\text{robô}}$. Esta matriz é o elo que permite que um ponto medido no espaço da câmera, $\mathbf{p}_{\text{cam}}$, seja corretamente expresso como um ponto alvo no espaço do robô, $\mathbf{p}_{\text{robô}}$.
A Figura 1 ilustra de forma esquemática a relação entre os diferentes sistemas de coordenadas abordados e as transformações matemáticas que os conectam.

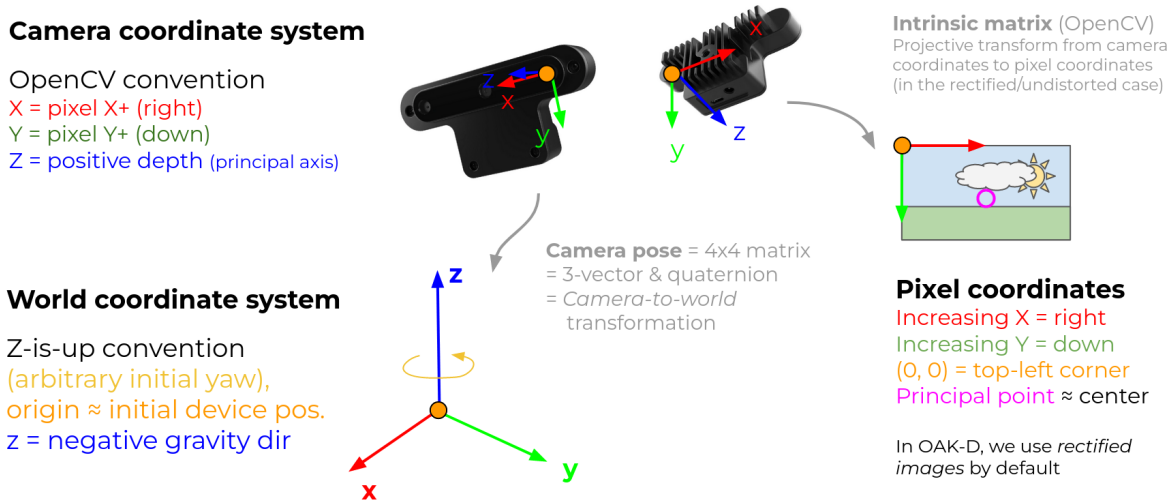


Figura 1: Representação dos principais sistemas de coordenadas em um sistema de visão robótica. A imagem ilustra o referencial de pixels (origem no canto superior esquerdo), o referencial da câmera (convenção OpenCV, com Z para frente) e um referencial de mundo (com Z para cima). Também são mostradas as transformações que os relacionam: a matriz intrínseca (parâmetros intrínsecos) e a matriz de pose da câmera (parâmetros extrínsecos). Fonte: Adaptado de [1].

Parâmetros Extrínsecos Os parâmetros extrínsecos descrevem a pose (posição e orientação) do sistema de coordenadas da câmera em relação a um sistema de coordenadas de referência externo, frequentemente chamado de sistema de coordenadas do mundo. Estes parâmetros são representados por uma matriz de transformação homogênea 4×4, $\mathbf{T}$, que contém uma submatriz de rotação $\mathbf{R}$ (3×3) e um vetor de translação $\mathbf{p}$ (3×1). No contexto deste projeto, o sistema de coordenadas do mundo é definido como o sistema de coordenadas da base do robô. Portanto, o conjunto de parâmetros extrínsecos de maior criticidade é aquele que define a transformação do referencial da câmera para o referencial do robô, aqui denotado como $\mathbf{T}_{\text{cam}\to\text{robô}}$. Esta matriz é o elo que permite que um ponto medido no espaço da câmera, $\mathbf{p}_{\text{cam}}$, seja corretamente expresso como um ponto alvo no espaço do robô, $\mathbf{p}_{\text{robô}}$, através da relação:

$$\mathbf{p}_{\text{robô}} = \mathbf{T}_{\text{cam}\to\text{robô}} \cdot \mathbf{p}_{\text{cam}}$$

5 Metodologia e Desenvolvimento

Esta seção descreve em detalhes os componentes de hardware, a arquitetura de software e o pipeline de processamento de dados que foram implementados para atingir os objetivos do projeto. A metodologia abrange desde a aquisição de dados do ambiente até a atuação física do manipulador robótico.

5.1 Hardware Utilizado

O sistema proposto é uma integração de três subsistemas de hardware principais: um sistema de percepção (câmera), um sistema de atuação (braço robótico) e uma interface de controle (placa controladora). A figura a seguir (inserir figura do seu setup aqui) ilustra a montagem física dos componentes.

5.1.1 Câmera de Profundidade Luxonis OAK-D

O principal sensor de percepção do sistema é a câmera de inteligência artificial espacial Luxonis OAK-D [4]. Este dispositivo é composto por uma câmera central de resolução 4K para captura de imagens coloridas (RGB) e um par de câmeras monocromáticas globais para visão estéreo. A característica distintiva da OAK-D é sua Unidade de Processamento de Visão (VPU) Myriad X embarcada, que permite o processamento de algoritmos de visão computacional diretamente no dispositivo. Para este projeto, a VPU é utilizada para executar o algoritmo de correspondência estéreo, gerar o mapa de profundidade e alinhar espacialmente este mapa com a imagem RGB, reduzindo significativamente a carga computacional no computador hospedeiro.

5.1.2 Braço Robótico Pincher e Servos Dynamixel

O sistema de atuação é um braço robótico comercial do modelo Pincher, ilustrado na Figura 3. Trata-se de um manipulador serial de 4 graus de liberdade (DoF), com todas as suas juntas sendo do tipo revolução.
Cada junta do manipulador é atuada por um servomotor inteligente do modelo Dynamixel AX-12A [6], ilustrado na figura 2. Estes atuadores são notáveis por sua arquitetura de comunicação em barramento (*daisy-chain*) e por uma tabela de controle interna que permite o gerenciamento digital de múltiplos parâmetros, como posição angular alvo, velocidade de movimento e limites de torque (compliance), através de um protocolo de comunicação baseado em pacotes.

Figura 2: O servomotor inteligente Dynamixel AX-12A, utilizado em cada uma das juntas do manipulador robótico.

O diagrama cinemático na Figura 3 detalha o modelo matemático utilizado para o robô, incluindo a atribuição dos sistemas de coordenadas para cada um dos elos e as dimensões utilizadas nos cálculos da cinemática direta e inversa.

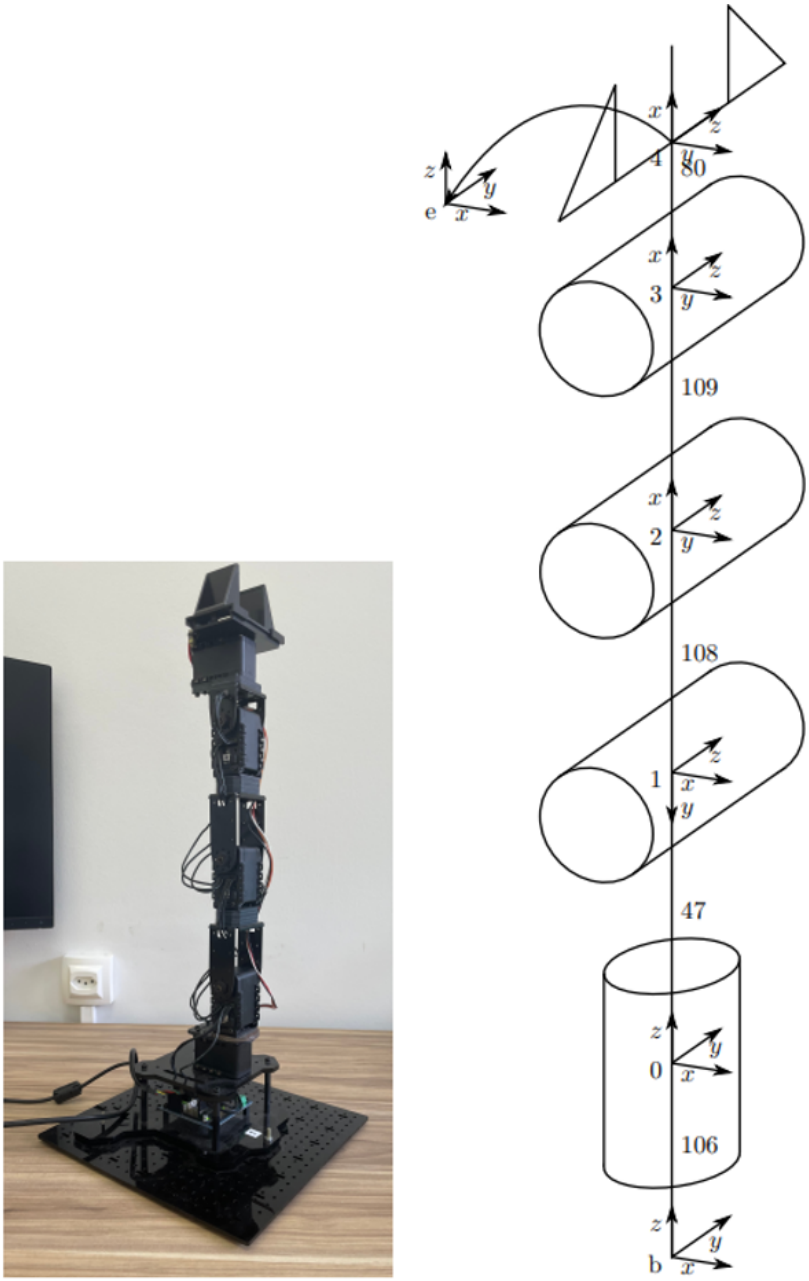


Figura 3: À esquerda, o manipulador robótico Pincher utilizado no projeto. À direita, o diagrama cinemático correspondente, exibindo a atribuição dos sistemas de coordenadas para a base (b), as juntas (0 a 3) e o efetuador final (e), juntamente com as dimensões dos elos em milímetros.

5.1.3 Placa de Controle Arbotix-M

A interface entre o computador hospedeiro e o barramento de servos Dynamixel é realizada pela placa controladora Arbotix-M [5]. Baseada em um microcontrolador ATmega644P, esta placa se conecta ao computador via USB (emulando uma porta serial) e é responsável por traduzir os comandos de alto nível enviados pelo software em Python para o protocolo de comunicação de baixo nível exigido pelos servos Dynamixel.

5.2 Arquitetura de Software

5.2.1 Ambiente e Bibliotecas

O software de controle e processamento foi desenvolvido integralmente na linguagem Python 3.9 [11]. O ambiente de desenvolvimento se apoiou em um conjunto de bibliotecas de código aberto que são padrão na indústria de robótica e visão computacional:

- **NumPy [9]:** Utilizada para todas as operações numéricas, especialmente a álgebra linear envolvida nas transformações de coordenadas e cálculos cinemáticos.
- **OpenCV [8]:** Empregada para a criação da interface gráfica do usuário (GUI), a exibição dos streams de vídeo e a captura de eventos de mouse e teclado.
- **DepthAI [7]:** A API oficial da Luxonis, utilizada para configurar o pipeline da câmera OAK-D, iniciar os streams de dados (RGB e profundidade) e recuperar os frames para processamento.

5.2.2 Estrutura do Código

Para este protótipo funcional, foi adotada uma estrutura de código monolítica. O script principal é responsável por inicializar os dispositivos de hardware (câmera e robô), configurar a GUI e entrar em um laço de execução principal. Dentro deste laço, o programa continuamente adquire novos frames, atualiza a tela, e processa eventos assíncronos do usuário (inputs do teclado e cliques do mouse) para disparar as rotinas de cálculo e movimentação do robô. Ao final da execução, uma rotina de finalização garante a desabilitação segura dos motores e o fechamento das conexões de hardware.

5.3 Pipeline de Percepção e Controle

O coração do sistema é o pipeline de processamento que converte um clique de mouse em um movimento de robô. A lógica fundamental deste pipeline foi primeiramente desenvolvida e validada em um ambiente de simulação construído com a biblioteca Matplotlib, conforme ilustrado na Figura 4. Este simulador permitiu depurar a cadeia de transformações matemáticas de forma isolada, antes da integração com o hardware. O processo, tanto na simulação quanto no sistema final, é executado na seguinte sequência:

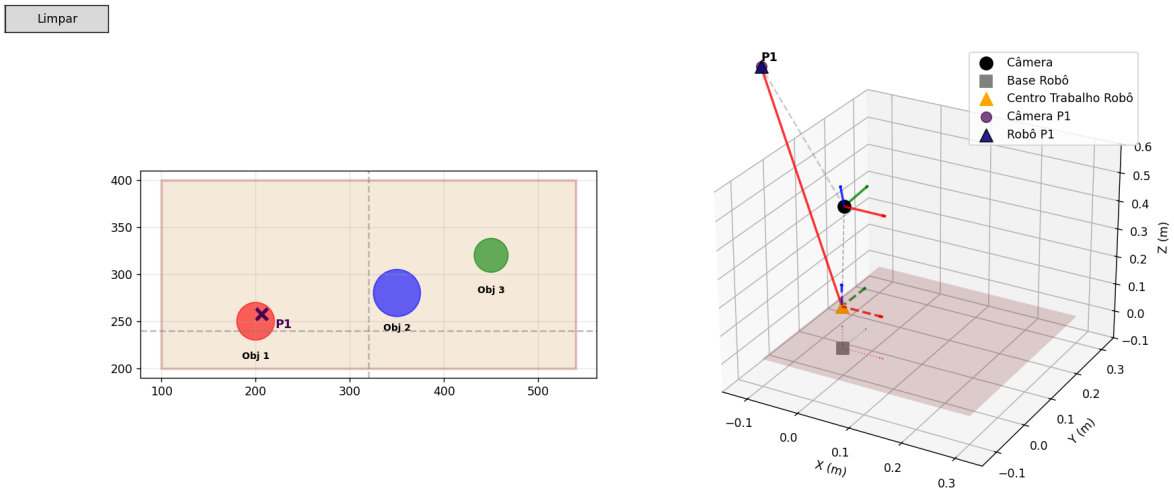
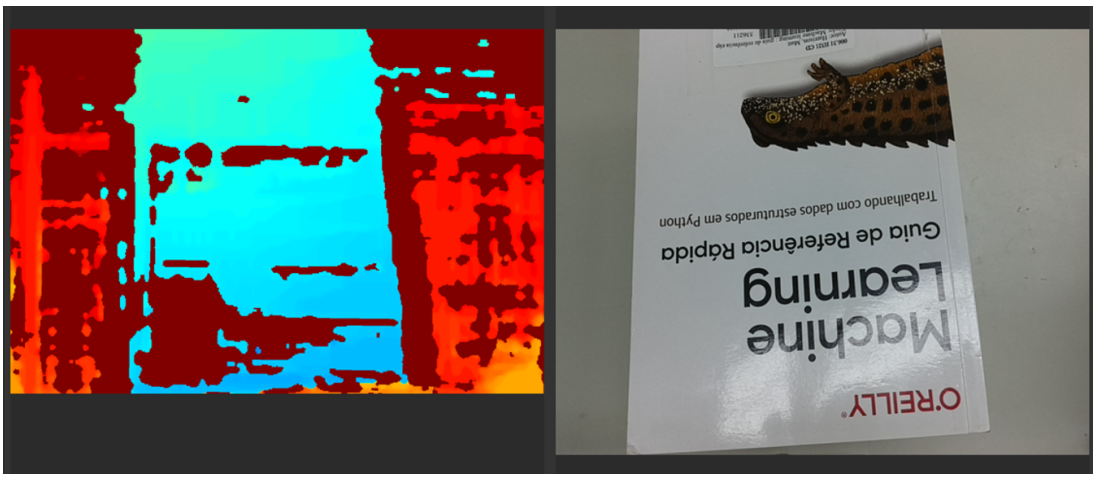


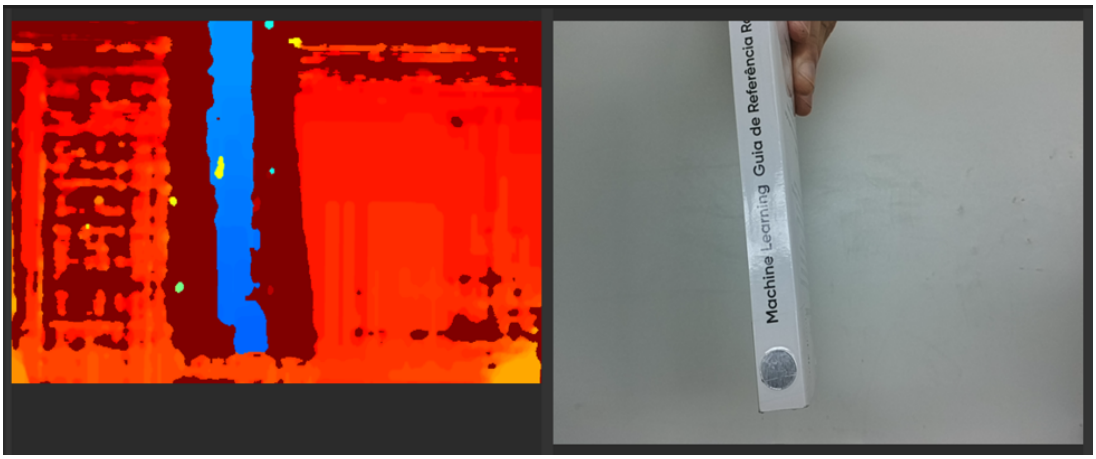
Figura 4: Simulador desenvolvido [2] para validar a lógica do pipeline de transformação de coordenadas. O painel esquerdo representa a visão da câmera virtual, onde um alvo (P1) é selecionado. O painel direito visualiza a transformação deste ponto do referencial da câmera para o referencial do robô. **Nota:** Este simulador foi uma ferramenta de desenvolvimento inicial; os sistemas de coordenadas aqui representados são conceituais e podem diferir da configuração final do hardware.

5.3.1 Aquisição e Alinhamento de Dados

O primeiro passo do pipeline de percepção e controle é a aquisição dos dados sensoriais do ambiente. A API DepthAI [7] é configurada para que a câmera OAK-D forneça dois *streams* de dados já sincronizados e espacialmente alinhados: um frame de imagem RGB e um mapa de profundidade. O alinhamento garante que cada pixel (u, v) na imagem RGB corresponda ao mesmo pixel (u, v) no mapa de profundidade.
A Figura 5 ilustra a qualidade e a natureza dos dados adquiridos. As imagens demonstram como diferentes posições e orientações de um objeto no espaço físico são representadas no mapa de profundidade, onde cores mais frias (azul) indicam maior proximidade com a câmera e cores mais quentes (vermelho) indicam maior distância.



(a) Objeto (livro) em posição horizontal. O mapa de profundidade (esquerda) representa a superfície plana com uma cor (profundidade) consistente e distinta do plano de fundo.



(b) Objeto (livro) em posição vertical. O mapa de profundidade mostra a lombada do livro como uma região mais próxima (azul) e claramente segmentada do fundo mais distante.

Figura 5: Exemplos de dados de percepção capturados pelo sistema OAK-D. Cada exemplo mostra a imagem RGB (direita) e o mapa de profundidade alinhado correspondente (esquerda), demonstrando a capacidade do sistema de distinguir a profundidade de objetos em diferentes orientações.

5.3.2 Transformação Pixel → Coordenadas da Câmera

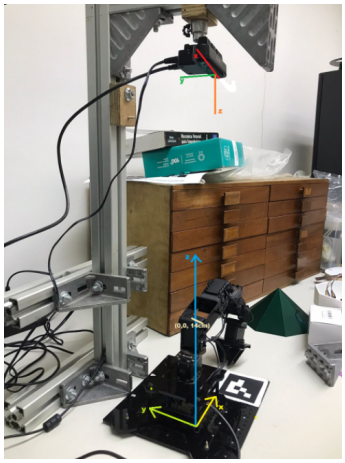
Quando um clique do mouse ocorre nas coordenadas de pixel (u, v), o valor de profundidade d (em milímetros) é lido no mapa de profundidade na mesma localização. Com estes três valores e a matriz intrínseca da câmera $\mathbf{K}$ (obtida da calibração do dispositivo), as coordenadas 3D do ponto no referencial da câmera, $\mathbf{p}_{\text{cam}}$, são calculadas através da equação de retroprojeção:

$$\mathbf{p}_{\text{cam}} = d \cdot \mathbf{K}^{-1} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

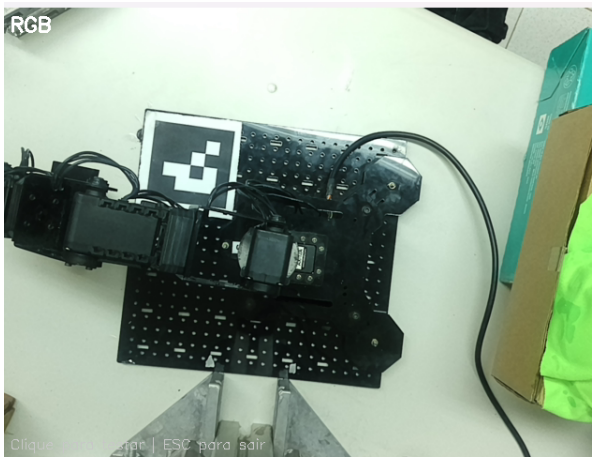
5.3.3 Transformação Câmera → Coordenadas do Robô

Uma vez que o ponto é localizado em 3D no referencial da câmera ($\mathbf{p}_{\text{cam}}$), a próxima etapa crucial é transformar estas coordenadas para o sistema de referência da base do robô, resultando em $\mathbf{p}_{\text{robô}}$. Esta operação é o que permite ao robô compreender a localização

do alvo em seu próprio mundo. A relação espacial entre os dois sistemas de coordenadas está ilustrada na Figura 6.



(a) Configuração física e sistemas de coordenadas.



(b) Visão da câmera (RGB).

Figura 6: Ilustração da configuração espacial do sistema. (a) Disposição física da câmera sobre o robô, com a representação dos seus respectivos sistemas de coordenadas. O referencial do robô (em amarelo/azul) tem origem na base, enquanto o da câmera (vermelho/verde/laranja) está montado acima. (b) Imagem da cena capturada pelo sensor RGB da câmera, mostrando o ponto de vista superior do sistema.

A transformação é realizada através da multiplicação pela matriz de transformação extrínseca $\mathbf{T}_{\text{cam}\to\text{robô}}$. Conforme a Figura 6a, esta matriz encapsula a relação geométrica entre os dois sistemas, que corresponde a uma rotação de 180 graus (π radianos) em torno do eixo Y e uma translação de 550 mm ao longo do eixo Z. A matriz exata, definida como uma constante no software do sistema, é:

$$\mathbf{T}_{\text{cam}\to\text{robô}} = \begin{bmatrix} -1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & 550.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

A implementação desta matriz no código-fonte é mostrada na Figura 7.

```python
# Matriz de transformação câmera->robô (FIXA baseada na calibração)
T_CAM_TO_ROBOT = np.array([
    [-1.,  0.,  0.,   0.],   # Rotação em Y (π) + Translação Z (500mm)
    [0.,  1.,  0.,   0.],
    [0.,  0., -1., 550.],
    [0.,  0.,  0.,   1.]
])
```

Figura 7: Trecho do código-fonte [3] em Python exibindo a definição da matriz de transformação homogênea $\mathbf{T}_{\text{cam}\to\text{robô}}$.

A aplicação desta matriz converte o ponto $\mathbf{p}_{\text{cam}}$ para o sistema de coordenadas do robô, resultando no vetor alvo final $\mathbf{p}_{\text{robô}}$, através da equação:

$$\begin{bmatrix} \mathbf{p}_{\text{robô}} \\ 1 \end{bmatrix} = \mathbf{T}_{\text{cam}\to\text{robô}} \cdot \begin{bmatrix} \mathbf{p}_{\text{cam}} \\ 1 \end{bmatrix}$$

5.3.4 Geração do Comando e Cinemática Inversa

As coordenadas de $\mathbf{p}_{\text{robô}}$, originalmente em milímetros, são convertidas para metros. Um vetor de pose alvo 4D, $\mathbf{p}_{\text{target}}$, é então construído. As três primeiras componentes correspondem à posição desejada, e a quarta componente corresponde a um ângulo de orientação do efetuador final, ϕ, que foi mantido fixo em $\pi/4$ radianos nesta implementação. Este vetor $\mathbf{p}_{\text{target}}$ é então fornecido à função de cinemática inversa analítica do robô, `ik()`, que calcula o vetor de ângulos de junta $\mathbf{q}$ necessário para atingir a pose alvo.

5.3.5 Validação da Solução e Atuação

Antes de executar o movimento, a solução $\mathbf{q}$ é passada por uma função de validação, `admissible()`, que verifica se os ângulos calculados estão dentro dos limites operacionais de cada junta e se a configuração resultante não causa colisões ou violações de espaço. Se a solução for considerada admissível, o vetor $\mathbf{q}$ é enviado, via placa Arbotix-M, para os servos Dynamixel, que então movem o braço para a configuração final.

6 Resultados

Nesta seção, são apresentados os resultados práticos obtidos com a implementação do sistema descrito. A avaliação foca na funcionalidade da interface de operação e na validação quantitativa da cadeia de transformação de coordenadas, que constitui o núcleo do sistema de percepção.

6.1 Interface de Operação

Para a interação do operador e a visualização de dados, foi desenvolvida uma interface gráfica de usuário (GUI) utilizando a biblioteca OpenCV [8]. A Figura 8 exibe a tela principal do sistema em funcionamento.

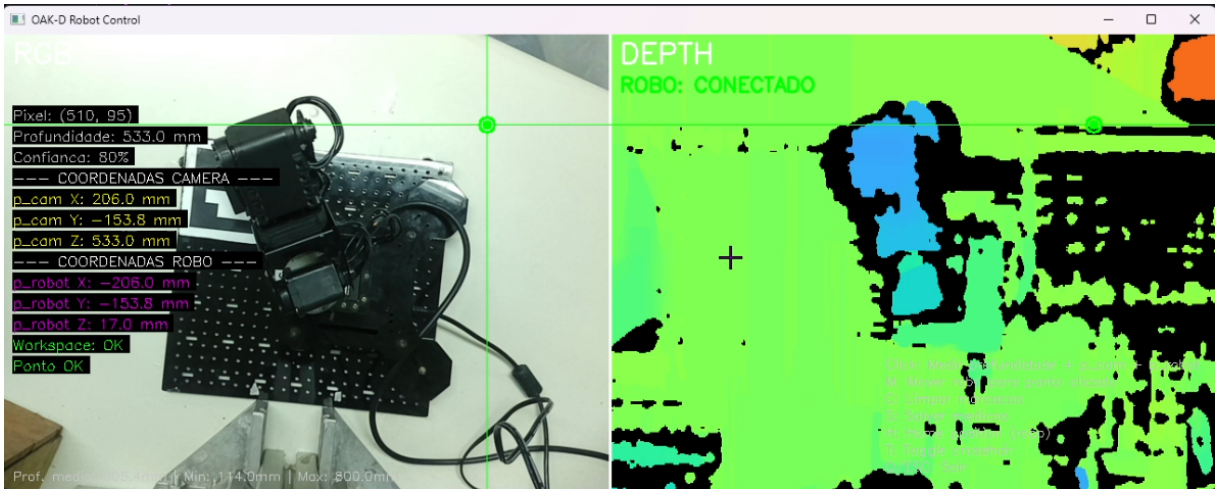


Figura 8: Interface de operação do sistema, exibindo o painel de visão RGB (esquerda) e o mapa de profundidade (direita). As sobreposições de texto mostram os dados calculados em tempo real para o ponto selecionado (marcado com um círculo verde).

A interface é dividida em dois painéis principais:

- **Painel Esquerdo (Visão RGB):** Exibe o *stream* de vídeo colorido da câmera central. É neste painel que o operador interage com o sistema, selecionando pontos

de interesse através de um clique do mouse. Um cursor em forma de cruz auxilia na mira precisa do ponto.

- **Painel Direito (Visão de Profundidade):** Mostra o mapa de profundidade gerado pelo sistema estéreo, com as distâncias codificadas em um mapa de cores para facilitar a visualização. Pontos mais próximos aparecem em tons mais quentes (vermelho/laranja) e pontos mais distantes em tons mais frios (azul/verde).

Sobre o painel esquerdo, um painel de status textual é renderizado para fornecer um diagnóstico completo e em tempo real a cada clique. Ele exibe as coordenadas do pixel selecionado, a profundidade medida naquele ponto, as coordenadas 3D resultantes no referencial da câmera (`p_cam`) e, mais importante, as coordenadas 3D finais no referencial do robô (`p_robot`). Adicionalmente, o status da conexão com o robô e o resultado da verificação do espaço de trabalho são exibidos, tornando a interface uma poderosa ferramenta de depuração e validação.

6.2 Validação da Transformação de Coordenadas

A validação da correção de todo o pipeline de transformação matemática foi realizada através de um caso de teste, cujos resultados quantitativos são apresentados na Figura 8. Para este teste, um ponto sobre a própria estrutura do braço robótico foi deliberadamente selecionado. A lógica é que, sendo a origem do referencial do robô localizada em sua base, um ponto em sua estrutura deve resultar em coordenadas `p_robot` com valores plausíveis e, notavelmente, um valor Z pequeno.
Os dados registrados para o ponto de teste foram os seguintes:

- **Entrada (Seleção do Usuário):**
  - Coordenadas de Pixel: $(u, v) = (510, 95)$
  - Profundidade Medida: $d = 533.0$ mm
- **Cálculo Intermediário (Referencial da Câmera):**
  - `p_cam` $= (206.0, -153.8, 533.0)$ mm
- **Resultado Final (Referencial do Robô):**
  - `p_robot` $= (-206.0, -153.8, 17.0)$ mm

A análise do vetor de coordenadas final, `p_robot`, serve como uma forte evidência da validade do sistema. Um valor de $Z = 17.0$ mm (ou 1.7 cm) é fisicamente coerente com a altura do ponto selecionado em relação à base do robô, que define o plano $Z = 0$. Este resultado confirma que a matriz de calibração intrínseca $\mathbf{K}$, a matriz de transformação extrínseca $\mathbf{T}_{\text{cam}\to\text{robô}}$, e as implementações das transformações matemáticas foram realizadas corretamente, produzindo um mapeamento preciso entre o espaço da imagem e o espaço de trabalho do robô.

6.3 Demonstração Funcional em Vídeo

Para complementar a análise estática, foram produzidos vídeos que demonstram a operação completa e dinâmica do sistema. Os vídeos registram o processo desde a seleção do ponto na interface gráfica até a conclusão do movimento do braço robótico.
Dois vídeos principais foram carregados na plataforma YouTube e podem ser acessados online:

- **Vídeo 1: Demonstração Geral do Sistema.** Apresenta uma visão geral da interface e um movimento de alcance para um objeto na mesa de trabalho.[1]
- **Vídeo 2: Teste de Precisão.** Demonstra múltiplos movimentos sequenciais para pontos distintos, evidenciando a repetibilidade do sistema.[2]

A análise destes vídeos confirma a funcionalidade do pipeline de percepção e controle, com o robô atingindo os alvos designados de forma suave e precisa, conforme esperado pela validação das coordenadas.

7 Discussão e Análise Crítica

Esta seção se propõe a interpretar os resultados apresentados, contextualizando-os dentro dos objetivos do projeto. Serão discutidos os principais desafios de implementação, as decisões de engenharia que moldaram a solução final e uma análise transparente das limitações inerentes ao sistema desenvolvido.

7.1 Análise dos Resultados

Os resultados quantitativos e qualitativos demonstram que o objetivo primário do projeto foi alcançado com sucesso. O sistema implementado é capaz de realizar o ciclo completo de percepção-ação: um ponto selecionado em um domínio de imagem 2D é corretamente mapeado para um alvo no espaço cartesiano 3D do robô, e o movimento para alcançar tal alvo é executado de forma precisa.
A validação da transformação de coordenadas, em particular, confirma a robustez do modelo matemático adotado. A obtenção de um vetor de posição final, `p_robot`, que é fisicamente coerente com a cena real, valida não apenas a implementação do código, mas toda a cadeia teórica subjacente — desde o modelo de câmera pinhole e a retroprojeção até a aplicação das transformações homogêneas. O sistema, portanto, constitui uma prova de conceito bem-sucedida para a tarefa de controle robótico guiado por visão.

7.2 Desafios de Implementação e Decisões de Engenharia

O desenvolvimento de um sistema que integra múltiplos domínios de hardware e software apresenta desafios significativos. A seguir, são discutidos os dois obstáculos mais relevantes e as decisões de engenharia tomadas para superá-los.

A Matriz de Calibração Extrínseca O desafio mais fundamental no desenvolvimento inicial foi a correta definição da relação espacial entre a câmera e o robô. Este é um problema clássico de calibração *eye-in-hand*. Tentativas iniciais de definir a matriz de transformação extrínseca, $\mathbf{T}_{\text{cam}\to\text{robô}}$, baseadas em estimativas diretas resultaram em movimentos erráticos do robô, pois pequenos erros na matriz de rotação são amplificados em grandes desvios de posição no espaço de trabalho.
A solução foi obtida através de uma definição rigorosa da matriz a partir de princípios básicos da geometria de transformações. Foi estabelecido que a montagem física da câmera correspondia, com boa aproximação, a uma rotação pura de 180 graus em torno do eixo Y do robô, seguida por uma translação ao longo do seu eixo Z. A codificação explícita desta matriz de rotação e translação no software resolveu as imprecisões e viabilizou o

[1] Disponível em: `https://www.youtube.com/watch?v=tvQ-3jA6I2M`
[2] Disponível em: `https://www.youtube.com/watch?v=tBhzTV4Ekug`

mapeamento correto. Esta experiência ressalta que a precisão de um sistema de percepção robótica é, em primeiro lugar, dependente da fidelidade do seu modelo de mundo, do qual a calibração extrínseca é o pilar.

O Problema da Orientação e a Solução Pragmática Um objetivo secundário do projeto era investigar o controle da orientação do efetuador final para que este pudesse abordar uma superfície de forma perpendicular ao seu vetor normal. Tal capacidade é um pré-requisito para tarefas avançadas como inspeção por contato ou montagem de precisão. O obstáculo técnico para esta implementação residiu na natureza da função de cinemática inversa disponível para o robô Pincher. A função `ik()` implementada é de natureza analítica e foi projetada para aceitar como entrada um vetor de pose 4D, onde o quarto elemento representa um único ângulo de orientação do efetuador (ϕ). Ela não é capaz de processar uma matriz de rotação 3×3 completa como alvo.
Diante desta limitação de software, foi tomada a **decisão de engenharia** de priorizar o objetivo principal (posicionamento correto) em detrimento do objetivo secundário (orientação controlada). A orientação do efetuador final foi fixada em um valor constante ($\phi = \pi/4$). Esta simplificação estratégica permitiu a utilização da cinemática inversa existente, garantindo a funcionalidade do sistema dentro do escopo viável. Esta abordagem exemplifica um *trade-off* comum em engenharia: decompor um problema complexo e resolver sua parte mais crítica, deixando extensões mais avançadas para iterações futuras baseadas em ferramentas mais capazes.

7.3 Limitações do Sistema

Uma análise crítica honesta deve reconhecer as limitações da solução implementada. As principais são:

- **Controle de Orientação Fixo:** Como discutido, o sistema não possui controle dinâmico da orientação do efetuador, o que restringe sua aplicação a tarefas simples de alcance de ponto.
- **Calibração Estática:** A matriz $\mathbf{T}_{\text{cam}\to\text{robô}}$ é definida estaticamente no código. Qualquer perturbação na posição ou orientação física da câmera invalidaria a calibração e exigiria uma atualização manual do código.
- **Controle em Malha Aberta:** O sistema opera em malha aberta do ponto de vista visual. Ele calcula um alvo e comanda o movimento, mas não utiliza a visão para verificar se o alvo foi alcançado com sucesso ou para corrigir erros dinamicamente (i.e., não há servo-controle visual).
- **Falta de Tratamento de Oclusão e Singularidades:** A implementação atual não trata casos em que o ponto de interesse está ocluído na visão da câmera ou situações em que a cinemática inversa pode falhar devido a uma configuração singular do robô.

8 Trabalhos Relacionados e Trabalhos Futuros

Para contextualizar a presente implementação e delinear seus possíveis desdobramentos, esta seção revisa brevemente áreas de pesquisa correlatas e propõe uma série de extensões e melhorias para o sistema.

8.1 Trabalhos Relacionados

O problema de controle de robôs guiado por visão é um campo vasto e bem estabelecido na literatura. O trabalho aqui desenvolvido se relaciona diretamente com três áreas principais:

- **Calibração *Eye-in-Hand*:** A determinação da matriz de transformação $\mathbf{T}_{\text{cam}\to\text{robô}}$ é um problema clássico conhecido como calibração *eye-in-hand*. Soluções canônicas para este problema, como as propostas por Tsai e Lenz, ou por Shiu e Ahmad, envolvem o robô mover um padrão de calibração para múltiplas posições e orientações, resolvendo a equação matricial $\mathbf{AX} = \mathbf{XB}$, onde $\mathbf{X}$ é a transformação desconhecida. Embora uma abordagem manual tenha sido adotada neste projeto, o arcabouço teórico para uma calibração automática é robusto e amplamente documentado.

- **Servo-Controle Visual (*Visual Servoing*):** Diferente da abordagem em malha aberta aqui implementada ("olhar e mover"), o servo-controle visual utiliza o feedback da câmera em tempo real para guiar o movimento do robô. As duas principais arquiteturas são o *Position-Based Visual Servoing* (PBVS), que reconstrói a pose 3D do alvo para calcular o erro no espaço cartesiano, e o *Image-Based Visual Servoing* (IBVS), que calcula o erro diretamente no espaço de pixels da imagem. A implementação de qualquer uma dessas arquiteturas representaria a transição para um sistema de controle em malha fechada.

- **Percepção 3D para Robótica:** Enquanto este trabalho utilizou visão estéreo passiva, outras modalidades de sensoriamento 3D, como *Time-of-Flight* (ToF) e luz estruturada, também são amplamente empregadas para a reconstrução de cenas e localização de objetos em aplicações robóticas, cada qual com seus próprios *trade-offs* de precisão, alcance e robustez a condições de iluminação.

8.2 Trabalhos Futuros

Com base nas limitações identificadas e no potencial da plataforma desenvolvida, uma série de trabalhos futuros pode ser proposta para expandir as capacidades do sistema. Notavelmente, o desenvolvimento de algumas dessas funcionalidades já foi iniciado em módulos experimentais, aguardando integração.

- **Implementação do Controle de Orientação Normal:** A extensão mais direta é a implementação do controle de orientação perpendicular. O trabalho fundamental para esta tarefa já foi realizado através de um módulo experimental de análise de superfície, cuja interface é apresentada na Figura 9. Este módulo já é capaz de extrair um perfil de profundidade de um objeto e calcular seu gradiente. O próximo passo lógico consiste em: (1) utilizar estes dados de gradiente para estimar o vetor normal à superfície no ponto de interesse; (2) construir uma matriz de rotação alvo a partir deste vetor; e (3) integrar esta matriz a um solver de cinemática inversa numérico, baseado na Jacobiana, que substitua ou complemente o solver analítico atual.

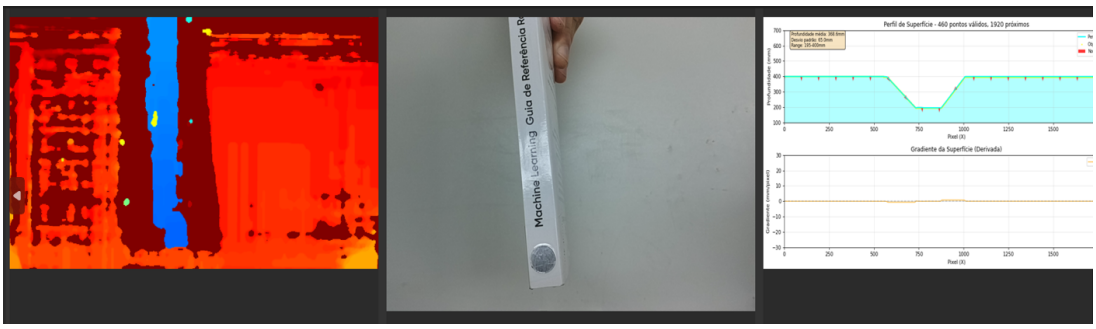


Figura 9: Interface do módulo experimental de análise de perfil de superfície. Da esquerda para a direita: mapa de profundidade, imagem RGB correspondente e os gráficos de diagnóstico. O gráfico superior exibe o perfil de profundidade ao longo de uma linha, enquanto o inferior mostra o gradiente (derivada) deste perfil, dados que são a base para a estimativa do vetor normal.

- **Rotina de Calibração Extrínseca Automática:** Para aumentar a flexibilidade e robustez do sistema, uma rotina de calibração automática pode ser desenvolvida. Utilizando um marcador fiducial, como um padrão de xadrez ou um marcador ArUco, o robô poderia autonomamente se mover para visualizar o marcador de diferentes perspectivas, permitindo o cálculo automático da matriz $\mathbf{T}_{\text{cam}\to\text{robô}}$.
- **Aplicação em Inspeção Acústica de Superfícies:** A motivação final do projeto pode ser realizada como um trabalho futuro. O sistema seria estendido para guiar um sensor acústico (e.g., um transdutor de ultrassom) até um ponto de contato. O controle de orientação normal, desenvolvido a partir do módulo da Figura 9, seria crucial nesta fase para garantir um acoplamento acústico de alta qualidade com a superfície inspecionada, permitindo a detecção de defeitos sub-superficiais.
- **Refatoração do Software para Arquitetura Modular:** Para facilitar a manutenção e futuras extensões, o código monolítico poderia ser refatorado para uma arquitetura modular baseada em classes ou, para maior escalabilidade, migrado para um framework de robótica como o ROS (*Robot Operating System*).

9 Conclusões

Este projeto abordou o desenvolvimento de um sistema de controle robótico guiado por visão, com o objetivo de mapear um ponto selecionado por um usuário em uma imagem para uma ação de posicionamento no espaço físico. Foi implementado com sucesso um protótipo funcional que integra uma câmera de profundidade OAK-D e um manipulador Pincher de 4-DoF, validando a cadeia completa de transformações desde o pixel na imagem até as coordenadas no espaço de trabalho do robô.

O principal êxito do trabalho foi a implementação correta e a validação de um pipeline matemático e de software que traduz de forma robusta a percepção visual em um comando motor. A análise dos resultados demonstrou que a definição precisa da matriz de transformação extrínseca entre a câmera e o robô é o fator mais crítico para o sucesso de tais sistemas. As decisões de engenharia, como a fixação da orientação do efetuador final, foram justificadas como uma abordagem pragmática para garantir a funcionalidade do sistema dentro das limitações das ferramentas disponíveis, constituindo uma base sólida

para futuras expansões.

Apesar das limitações reconhecidas, como a operação em malha aberta e a calibração estática, o sistema desenvolvido representa uma prova de conceito bem-sucedida e uma valiosa plataforma experimental. Ele não apenas atinge seu objetivo primário, mas também estabelece a fundação necessária para a implementação de funcionalidades mais avançadas, como o controle de orientação e aplicações de inspeção por contato. O trabalho, portanto, contribui como um estudo de caso prático e detalhado sobre os desafios e soluções na integração de sistemas de percepção e atuação robótica.

Referências

[1] Spectacular AI: *SDK Documentation - Coordinate Systems*, 2024. https://spectacular.ai/docs/sdk/coordinate-systems, Documentação técnica sobre sistemas de coordenadas. Acessado em: 15 de julho de 2025.

[2] Sete, Paloma F. L.: *RAISE-BOT: Simulador de Validação de Pipeline de Coordenadas*, 2025. https://github.com/palomaflsette/raise-bot/blob/development/simulator.py, Repositório do código-fonte. Acessado em: 15 de julho de 2025.

[3] Sete, Paloma F. L.: *RAISE-BOT: Software de Controle Integrado e Percepção 3D*, 2025. https://github.com/palomaflsette/raise-bot/blob/development/monolito.py, Repositório do código-fonte. Acessado em: 15 de julho de 2025.

[4] Luxonis: *OAK-D: Spatial AI Camera and DepthAI API*, 2024. https://docs.luxonis.com/, Hardware and software documentation for the OpenCV AI Kit, DepthAI. Acessado em: 15 de julho de 2025.

[5] Interbotix Labs: *ArbotiX-M Robocontroller*, 2024. https://www.interbotix.com/arbotix-robocontroller, Documentação técnica e especificações do produto. Acessado em: 15 de julho de 2025.

[6] ROBOTIS Co., Ltd.: *e-Manual for Dynamixel AX-12A*, 2024. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/, Official technical documentation, including control table and communication protocol. Acessado em: 15 de julho de 2025.

[7] Luxonis: *DepthAI: The Python API for OAK Cameras*, 2024. https://github.com/luxonis/depthai-python, Acessado em: 15 de julho de 2025.

[8] Bradski, Gary: *The OpenCV Library*. Dr. Dobb's Journal of Software Tools, 25(11):120–125, 2000.

[9] Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke e Travis E. Oliphant: *Array programming with NumPy*. Nature, 585(7825):357–362, setembro 2020. https://doi.org/10.1038/s41586-020-2649-2.

[10] Siciliano, Bruno, Lorenzo Sciavicco, Luigi Villani e Giuseppe Oriolo: *Robotics: Modelling, Planning and Control*. Springer, 2010.

[11] Python Software Foundation: *Python Language Reference, version 3.10*, 2024. https://docs.python.org/3/, Disponível em: https://www.python.org. Acessado em: 15 de julho de 2025.