UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
CURSO DE ENGENHARIA DE AUTOMAÇÃO

Projeto de Graduação em Engenharia de Automação

Gerador de *Datasets* para Aprendizado de Manipulação Robótica Orientado por *Affordances*

Gabriel Amaral Dorneles

Projeto de Graduação apresentado ao Curso de Engenharia de Automação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Engenheiro de Automação

Orientador: Prof. Dr. Rodrigo da Silva Guerra
Co-orientador: Eng. João Francisco de Souza Santos Lemos

Rio Grande, 2025

Dados de catalogação na fonte:
colocar NOME DO BIBLIOTECÁRIO – CRB-colocar número do crb do bibliotecário
Biblioteca Central – FURG

A999a

Dorneles, Gabriel Amaral

Gerador de *Datasets* para Aprendizado de Manipulação Robótica Orientado por *Affordances* / Gabriel Amaral Dorneles. – Rio Grande, 2025. – 70 f: gráf. – Projeto de Graduação – Engenharia de Automação. Universidade Federal do Rio Grande - FURG. Centro de Ciências Computacionais. Rio Grande, 2025. – Orientador Rodrigo da Silva Guerra; Co-orientador João Francisco de Souza Santos Lemos.

1. Manipulação robótica. 2. Affordance. 3. Gerador de dataset. 4. Aprendizado de máquina. I. Guerra, Rodrigo da Silva. II. Lemos, João Francisco de Souza Santos. III. Título.

CDD: 999.9

Projeto de Graduação em Engenharia de Automação

Gerador de *Datasets* para Aprendizado de Manipulação Robótica Orientado por *Affordances*

Gabriel Amaral Dorneles

Banca examinadora:

Prof. Dr. Marcelo Rita Pias

Prof. Dr. Paulo Lilles Jorge Drews Junior

Dedico. . . aos meus pais, Cláudio e Susana.
Pelo amor, paciência e incentivo constantes.

AGRADECIMENTOS

A realização deste trabalho de conclusão de curso só foi possível graças ao apoio, orientação e incentivo de muitas pessoas, às quais gostaria de expressar minha gratidão.

Agradeço, primeiramente, à minha família e, em especial, aos meus pais, Cláudio e Susana, pelo amor incondicional, paciência e suporte ao longo de toda a minha trajetória acadêmica. Vocês sempre acreditaram no poder transformador da educação e me proporcionaram oportunidades que, apesar de não terem experienciado, sempre desejaram para mim. Às minhas irmãs, Luciele e Simone, ao meu sobrinho Cícero e ao meu primo Leonardo, por me incentivarem em todos os momentos e por serem uma fonte de inspiração e motivação. Além disso, dedico em memória as minhas avós Flora e Marina, que sempre me apoiaram incondicionalmente.

Agradeço aos professores e colegas que estiveram presentes ao longo de toda a minha trajetória acadêmica, desde a formação de base na Escola Adventista de Rio Grande, passando pelo ensino médio no Colégio Marista São Francisco, até minha graduação na FURG. Agradeço ao meu orientador, Prof. Dr. Rodrigo Guerra, por guiar o desenvolvimento deste trabalho de conclusão de curso. Expresso também minha gratidão ao Eng. João Francisco Lemos, meu coorientador, por sua constante disponibilidade e apoio durante todo o processo de elaboração deste trabalho.

Aos meus colegas de curso e amigos, em especial Adir Pedroso, Andressa Cavalcante, Lucas Corrêa, Marina Zanotta e Murilo Castanheira, meu sincero agradecimento por estarem ao meu lado ao longo de toda a jornada acadêmica, compartilhando desafios, conquistas e momentos de alegria. Um agradecimento especial aos membros da equipe de robótica FBOT, e aos líderes Jardel Dyonisio e João Francisco Lemos, que foram fundamentais para minha formação como engenheiro e o desenvolvimento desse trabalho.

Muito Obrigado.

The most important step a man can take.
It's not the first one, is it?
It's the next one.
Always the next step.
— BRANDON SANDERSON

RESUMO

DORNELES, Gabriel Amaral. **Gerador de *Datasets* para Aprendizado de Manipulação Robótica Orientado por *Affordances***. 2025. 70 f. Projeto de Graduação – Engenharia de Automação. Universidade Federal do Rio Grande - FURG, Rio Grande.

A manipulação robótica desempenha um papel fundamental no avanço da automação, permitindo que robôs interajam com o ambiente e realizem tarefas que vão desde montagem industrial até serviços domésticos. Apesar do progresso significativo, ensinar robôs a manipular objetos com formas, propriedades e funções diversas em ambientes não estruturados continua sendo um desafio considerável.

Este trabalho propõe uma ferramenta capaz de gerar automaticamente *datasets* em larga escala, voltados a ensinar robôs a manipular objetos nos locais corretos por meio de *affordances* baseadas em tarefas. O *pipeline* desenvolvido parte de um modelo 3D anotado com *affordances* e uma tarefa específica, gerando demonstrações automáticas e expandindo-as com o MimicGen, resultando em um conjunto de dados significativamente maior. Por uma série de experimentos, este estudo avalia a eficácia dos *datasets* gerados através do treinamento de múltiplas redes neurais. Os resultados evidenciam a capacidade das redes de se adaptarem a novos modelos de objetos e variações no ambiente, como mudanças de textura, ao mesmo tempo em que revelam as limitações das arquiteturas atuais em lidar com cenários altamente variáveis.

Palavras-chave: Manipulação robótica, affordance, gerador de dataset, aprendizado de máquina.

ABSTRACT

DORNELES, Gabriel Amaral. **Dataset Generator for Affordance-driven Robotic Manipulation Learning**. 2025. 70 f. Projeto de Graduação – Engenharia de Automação. Universidade Federal do Rio Grande - FURG, Rio Grande.

Robotic manipulation plays a crucial role in advancing automation, enabling robots to interact with their environment and perform tasks ranging from industrial assembly to home chores. Despite significant progress, teaching robots to manipulate objects with diverse shapes, properties, and functions in unstructured environments remains a substantial challenge.

This work proposes a tool capable of automatically generating large-scale datasets, focused on teaching robots to manipulate objects in appropriate ways through task-oriented affordances. The developed pipeline begins with a 3D object model annotated with its respective affordances and a specific task, generating automatic demonstrations and expanding them using MimicGen, resulting in a significantly larger dataset. Through a series of experiments, this study evaluates the effectiveness of the generated datasets by training multiple neural networks. The results demonstrate the networks' ability to adapt to new object models and environmental variations, such as texture changes, while revealing the limitations of current architectures in handling highly variable scenarios.

Keywords: Robotic manipulation, affordance, dataset generator, machine learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BORIS	Do inglês *Brazilian Open Robot for Indoor Service*
CBR	Competição Brasileira de Robótica
RNA	Redes Neurais Artificiais
MDP	Do inglês *Markov Decision Process*
RL	Do inglês *Reinforcement Learning*
IL	Do inglês *Imitation Learning*
BC	Do inglês *Behavior Cloning*
IRL	Do inglês *Inverse Reinforcement Learning*
NASA	Do inglês *National Aeronautics and Space Administration*
DoF	Do inglês *Degrees of Freedom*
LiDAR	Do inglês *Light Detection and Ranging*
DRL	Do inglês *Deep Reinforcement Learning*
VR	Do inglês *Virtual Reality*
AR	Do inglês *Augmented Reality*
RNN	Do inglês *Recurrent Neural Network*
ROS	Do inglês *Robot Operating System*
IEEE	Do inglês *Institute of Electrical and Electronics Engineers*
XML	Do inglês *Extensible Markup Language*
API	Do inglês *Application Programming Interface*
OSC	Do inglês *Operational Space Control*
HDF5	Do inglês *Hierarchical Data Format Version 5*

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Os robôs desenvolvidos ao longo das últimas décadas têm demonstrado eficácia na tarefa de pegar e manipular objetos em ambientes repetitivos e familiares, como nas linhas de montagem industrial, onde a geometria, as propriedades do material e o peso são geralmente conhecidos. Contudo, a crescente adoção de robôs em contextos que vão além dos ambientes estruturados e controlados das fábricas, especialmente no setor de serviços, tem introduzido novos desafios que exigem uma habilidade de compreender e interagir com o mundo maior do que a necessária anteriormente [1].

Diante disso, a manipulação robótica emerge como uma das habilidades mais fundamentais. Ambientes como hospitais, restaurantes e residências necessitam que robôs sejam capazes de interagir com o meio e manipular objetos em cenários desconhecidos e não planejados, exigindo competências de percepção e manipulação mais sofisticadas [2]. Apesar dos grandes avanços em design mecânico, percepção e controle direcionados à manipulação, a capacidade dos robôs de se adaptarem a objetos desconhecidos, com diferentes propriedades e finalidades, ainda permanece bastante rudimentar [3]. Além disso, grande parte das decisões ainda depende da intervenção humana, evidenciando a necessidade de evoluir os sistemas para maior independência na tomada de decisão.

Um conceito fundamental para aprimorar a interação de robôs com o mundo é o de *affordance* [4]. Esse conceito se refere às possibilidades de interação que um objeto oferece a um agente, estabelecendo uma associação entre a forma, o material e a funcionalidade dos objetos com ações específicas que podem ser realizadas, como agarrar, empurrar, rolar ou servir. Os *affordances* de diferentes objetos desempenham um papel importante para o desenvolvimento de robôs autônomos capazes de interagir com ambientes complexos, permitindo que, ao manipular um objeto, o robô seja capaz de utilizá-lo de forma adequada para alcançar o objetivo desejado.

1.2 Justificativa

A Universidade Federal do Rio Grande promove o projeto FBOT [1], uma iniciativa voltada à pesquisa e à participação em competições na área de robótica. Entre as diversas categorias contempladas pelo projeto, destaca-se a Robocup@Home [2], que busca estabelecer um padrão de referência para a comparação de robôs de serviço em ambientes domésticos.

A plataforma utilizada pela equipe na categoria é chamada de *Brazilian Open Robot for Indoor Service* (BORIS), e é equipada com o manipulador robótico Interbotix WidowX-200 [3]. Através desse manipulador o robô deve, de forma autônoma, executar diferentes tarefas que envolvem interagir com o ambiente, como abrir portas, organizar itens em prateleiras, retirar o lixo da casa e servir café da manhã.

Figura 1: Robô BORIS manipulando objetos durante a Competição Brasileira de Robótica (CBR 2024). Fonte: Equipe de Robótica FBOT.

A pesquisa atual busca contribuir para o aprimoramento do robô BORIS, capacitando-o a realizar tarefas de manipulação de maneira mais inteligente, permitindo que ele adapte a forma de manipular os objetos de acordo com suas propriedades e da tarefa que está sendo executada. Além disso, busca-se contribuir para o avanço da robótica de serviço, ajudando a tornar a integração de robôs em atividades cotidianas uma realidade cada vez

[1] Site do projeto: `https://fbot.vercel.app`

[2] Site da Robocup@Home: `https://athome.robocup.org`

[3] Documentação da Interbotix: `https://docs.trossenrobotics.com/interbotix_xsarms_docs/`

mais próxima e consolidando o papel da robótica como uma ferramenta essencial para melhorar a qualidade de vida no ambiente doméstico.

1.3 Objetivos

O objetivo principal deste trabalho é desenvolver um método para a geração de *datasets* que capacitem robôs a manipular objetos com base em suas funcionalidades, considerando a intenção por trás da execução de uma ação. Em outras palavras, busca-se automatizar a criação de *datasets* capazes de integrar a relação entre diferentes tarefas e as formas específicas de manipulação requeridas para um objeto, ensinando essas formas ao modelo de maneira simultânea ao aprendizado da tarefa.

Para alcançar esse objetivo, foram estabelecidos os seguintes objetivos específicos:

1. Definir um ambiente de simulação que garanta a compatibilidade com diversas tarefas de manipulação robótica;
2. Desenvolver um método para a geração de *datasets* de manipulação, orientado pelo conceito de *affordances*;
3. Gerar *datasets* para diferentes tarefas de manipulação robótica;
4. Validar os *datasets* gerados utilizando algoritmos de aprendizado de máquina, avaliando seu desempenho em tarefas e na manipulação baseada em *affordances*.

1.4 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O Capítulo 1 introduziu a proposta do trabalho, assim como sua motivação e justificativa. No Capítulo 2, são discutidos os conceitos fundamentais necessários para o entendimento do tema, enquanto o Capítulo 3 explora trabalhos relacionados ao estudo em questão. A metodologia empregada, as ferramentas utilizadas e os experimentos realizados são detalhados no Capítulo 4. O Capítulo 5 apresenta os resultados obtidos, e o Capítulo 6 encerra o trabalho, sugerindo direções para trabalhos futuros. Por fim, os Apêndices contêm informações adicionais relevantes sobre o desenvolvimento do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os principais conceitos, teorias e estudos que sustentam a pesquisa sobre o tema. Embora o foco principal deste trabalho seja a criação de *datasets*, a Seção 2.1 discutirá conceitos de aprendizado de máquina, visando contextualizar as aplicações dos conjuntos de dados desenvolvidos. A Seção 2.2 explorará o papel dos *datasets* no aprendizado de máquina, destacando as diferentes formas de obtê-los. Por fim, a Seção 2.3 abordará conceitos gerais sobre manipulação robótica.

2.1 Aprendizado de Máquina

Em problemas mais complexos da robótica, métodos clássicos, como os baseados em cinemática, física e princípios geométricos, não são suficientes para representar ambientes complexos em sua totalidade [5]. Nesses casos, o uso de aprendizado de máquina tem se destacado como estado da arte, pois permite que o sistema extraia padrões diretamente a partir dos dados, sem a necessidade de modelagem explícita.

2.1.1 Conceitos Gerais

Pavone [5] descreve o aprendizado de máquina como dividido em duas categorias principais: aprendizado não supervisionado e aprendizado supervisionado.

Aprendizado Não Supervisionado. Dado um conjunto de dados $D = \{x_1, \ldots, x_n\}$ o problema de aprendizado não supervisionado consiste em encontrar padrões nos dados, como agrupamentos (*clusters*) ou associações.

Aprendizado Supervisionado. Dado um conjunto de dados rotulados $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, onde cada par (x_i, y_i) representa uma entrada x_i e sua respectiva saída y_i, o o problema de aprendizado supervisionado é encontrar uma função $f(x)$ que possa mapear novas entradas x para respectivas saídas y, generalizando o comportamento aprendido nos dados de treinamento para novos exemplos.

O aprendizado supervisionado é o mais utilizado em aplicações robóticas, como no caso de classificação de objetos, nas quais o robô deve identificar categorias específicas (por exemplo, reconhecer objetos em uma cena), e em problemas de regressão, nos quais o

objetivo é entender a relação entre uma variável dependente e uma ou mais variáveis independentes para prever um resultado específico (por exemplo, prever a posição e orientação futuras de um robô com base em seus dados de movimento anteriores) [5].

Em problemas de aprendizado supervisionado, há parâmetros que precisam ser ajustados para minimizar uma função conhecida como função de custo ou função de perda, que avalia o desempenho de modelos candidatos $f(x)$ na tarefa de ajuste aos dados. Esse processo de otimizar os parâmetros é conhecido como treinamento de modelo.

Um dos modelos mais comuns utilizado em aprendizado de máquina são as redes neurais artificiais (RNAs). Inspiradas pela estrutura do cérebro humano, as RNAs são compostas por múltiplas camadas de neurônios artificiais que processam e transformam dados de entrada. Um modelo com múltiplas camadas ocultas é conhecido como uma rede neural profunda, e é frequentemente treinado para realizar tarefas de alta complexidade, como reconhecimento de imagens, tradução automática e processamento de linguagem natural [5].

2.1.2 Processo de Decisão de Markov

O Processo de Decisão de Markov (*Markov Decision Process* - MDP) [6] é uma das abordagens mais comuns para modelar problemas na robótica, principalmente envolvendo aprendizado por reforço, pois oferece uma representação formal para a tomada de decisão sequencial que inclui incertezas [7]. Um MDP pode ser descrito por meio de uma tupla:

$$(S, A, R, T, \gamma) \tag{1}$$

Nesta, $S \subseteq \mathbb{R}^n$ representa o conjunto de estados possíveis do sistema, onde cada estado é descrito por um vetor de n dimensões contínuas. Por exemplo, um estado pode incluir a posição e orientação do manipulador robótico, assim como características do objeto sendo manipulado. $A \subseteq \mathbb{R}^m$ é o conjunto de ações possíveis que o robô pode realizar, como se mover em um eixo ou ajustes de força. Em um sistema Markoviano, o resultado de cada ação depende exclusivamente do estado atual, e não na sequência de eventos que o precederam.

A função de recompensa $R(s, a, s')$ atribui um valor à execução da ação $a \in A$ quando o sistema está no estado $s \in S$ e transita para o estado $s' \in S$, incentivando o robô a escolher ações que maximizem o retorno acumulado ao longo do tempo. Já a função de transição $T(s'|s, a)$ descreve a probabilidade do sistema passar para o estado s' dado que ele estava anteriormente no estado s e a ação a foi executada. Por fim, $\gamma \in [0, 1]$ é o fator de desconto que ajusta a importância de recompensas futuras, permitindo que o modelo balanceie entre ações que resultem em ganhos imediatos e aquelas que proporcionem benefícios de longo prazo [7].

Supondo que tem-se uma função objetivo $G : S \rightarrow \{\text{true}, \text{false}\}$, a tarefa de pla-

nejamento se resume a calcular uma sequência de ações de modo que, aplicando essa sequência a partir de um estado inicial, se alcance um estado $s \in G$. Por exemplo, a função de recompensa pode ser configurada para fornecer um reforço positivo sempre que um estado-alvo for atingido, e um reforço zero nos demais casos [8]:

$$R(s, a, s') = \begin{cases} 1, & \text{if } s \notin G \text{ and } s' \in G \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

O objetivo de um MDP, portanto, é encontrar uma política de controle π que mapeie estados para ações, de forma a maximizar o retorno $\sum_{i=0}^{\infty} \gamma^i r_i$ para todos os estados [8].

2.1.3 Aprendizado por Reforço

O aprendizado por reforço (*Reinforcement Learning* - RL) é uma área de aprendizado de máquina na qual um agente aprende a tomar decisões em um ambiente com o objetivo de maximizar uma recompensa acumulada, ou seja, minimizar uma função de custo [9].

A figura 2 ilustra o cenário clássico de RL baseado em MDPs, destacando a interação entre o agente e o ambiente. O agente é o sistema que toma decisões, responsável por observar as informações provenientes do ambiente. O ambiente, por sua vez, é um sistema com o qual o agente interage, abrangendo tudo o que está além do seu controle direto [10].

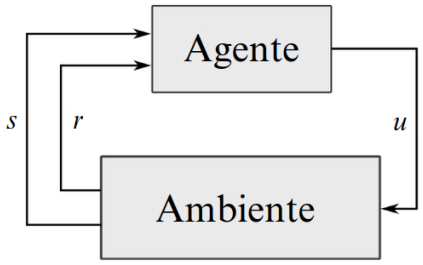


Figura 2: Em problemas de RL, o agente observa o estado s e a recompensa r, executa uma ação u e recebe do ambiente um novo estado e recompensa, aprendendo iterativamente com esse ciclo. Fonte: Universidade de Stanford [9]

O agente tem duas capacidades principais: interagir com o ambiente e receber um sinal de recompensa proveniente dessa interação. Não se sabe previamente como as ações do agente irão influenciar a evolução futura do estado ou as recompensas que serão obtidas. Dessa forma, não é necessário que o modelo do sistema seja conhecido, cabendo ao agente aprender por meio de suas próprias experiências [9].

O objetivo em RL é encontrar uma política π^* que maximize o retorno esperado, ou seja, a soma das recompensas futuras descontadas que um agente pode obter [9]. Matematicamente, isso é expresso como:

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t))\right] \tag{3}$$

No qual E representa o valor esperado, considerando todas as possíveis trajetórias de estados e ações, $\gamma \in [0, 1]$ é o fator de desconto e $R(x_t, \pi(x_t))$ é a recompensa imediata obtida ao tomar a ação $\pi(x_t)$ no estado x_t.

O aprendizado por reforço tradicional, conhecido como aprendizado por reforço *online*, envolve um agente que aprende interagindo diretamente com o ambiente, seja ele real ou simulado. Nesse processo, o agente alterna entre a exploração de novas ações (*explore*) e a exploração de ações conhecidas (*exploit*) para coletar dados e aprimorar suas políticas [9]. No entanto, essa abordagem pode ser cara ou inviável, especialmente em algumas aplicações do mundo real, como na robótica, onde a exploração pode levar a resultados indesejáveis [11].

Por outro lado, o aprendizado por reforço *offline*, também chamado de aprendizado por reforço em lotes, utiliza conjuntos de dados previamente coletados, sem exigir interação contínua com o ambiente. Essa abordagem permite focar no aprendizado de políticas orientado por dados, aproveitando extensos *datasets* coletados por outras políticas, sem a necessidade de exploração adicional [12].

Entretanto, quando um agente é treinado com um conjunto de dados originado de uma política específica, os estados e ações observados durante o treinamento podem não refletir aqueles que ele encontrará ao implementar a política aprendida [12]. Essa discrepância, conhecida como *distribution shift*, representa um dos principais desafios no aprendizado por reforço *offline*. Diversos estudos têm buscado desenvolver algoritmos capazes de mitigar os efeitos dessa questão, com o principal método sendo restringir certos aspectos do processo de aprendizado de modo a limitar o quanto a política aprendida π difere da política comportamental [13, 14].

2.1.4 Aprendizado por Imitação

Muitos dos desafios atuais em tecnologia dependem de inteligências artificiais capazes de replicar o comportamento humano ao serem expostas a situações similares, como no caso de robôs assistivos e veículos autônomos. Nessas aplicações, a quantidade de cenários possíveis é grande demais para ser abordada por programação explícita, exigindo que o agente deva ser capaz de lidar com cenários inéditos [15].

Assim, técnicas de aprendizado por imitação se destacam em casos nos quais é mais simples para um especialista demonstrar o comportamento desejado do que definir uma função de recompensa que gere esse mesmo comportamento. Em IL, em vez de aprender com recompensas esparsas ou especificar manualmente uma função de recompensa, um especialista fornece um conjunto de demonstrações. Com base nessas demonstrações, o agente busca aprender a política ideal π imitando as decisões do especialista e reprodu-

zindo o comportamento desejado.

De maneira formal, considerando que o sistema é um MDP, uma demonstração ξ é composta por um par (x, u), no qual $x \in X$ é um vetor de características que descreve o estado no instante observado, e $u \in U$ representa a ação de controle realizada pelo demonstrador [16]. A principal diferença em relação aos problemas de RL é que, em vez de usar uma função de recompensa explícita $r_t = R(x_t, u_t)$, assume-se que um conjunto de demonstrações de um especialista é fornecido. O problema de aprendizado por imitação, portanto, é utilizar um conjunto de demonstrações $\Xi = (\xi_1, \ldots, \xi_D)$ derivado de uma política especialista π^* para encontrar a política π tal que:

$$u_t = \pi(x_t) \tag{4}$$

De modo geral, existem dois tipos de abordagens para aprendizado por imitação: o primeiro é aprender diretamente a imitar a política do especialista, com o *Behavior Cloning* (BC) sendo um exemplo clássico comum; e o segundo é imitar indiretamente essa política, aprendendo a sua função de recompensa, um método conhecido como *Inverse Reinforcement Learning* [16]. A seguir, essas abordagens serão discutidas brevemente.

Behavior Cloning (BC) [17, 18, 19]. Introduzido por Pomerleau [17] no desenvolvimento do sistema *Autonomous Land Vehicle In a Neural Network* (ALVINN), Figura 3, o *Behavior Cloning* é um método clássico de aprendizado por imitação. Nesse trabalho, o agente é treinado através de uma rede neural para prever as ações corretas com base nas entradas dos sensores, mapeando-as diretamente em ângulos de direção para manter um veículo na pista sob diversas condições

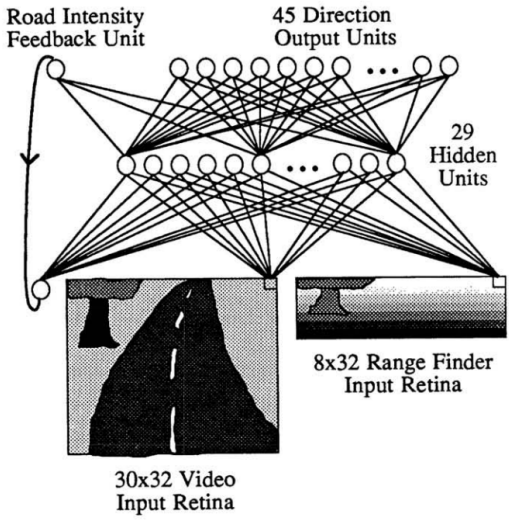


Figura 3: Arquitetura do sistema ALVINN. A entrada do sistema consiste de uma imagem de vídeo da estrada, a proximidade usando um sensor *laser* e uma unidade de *feedback* de intensidade. Essas entradas se conectam a uma camada oculta que gera comandos de navegação na camada de saída. Fonte: Pomerleau [17].

BC utiliza de um conjunto de demonstrações $\xi \in \Xi$ para minimizar a diferença entre a política π e a política especialista π^*. Especificamente, o objetivo é resolver o seguinte problema de otimização:

$$\hat{\pi}^* = \arg\min_{\pi} \sum_{\xi \in \Xi} \sum_{x \in \xi} L(\pi(x), \pi^*(x)) \tag{5}$$

No qual a função minimiza a perda L entre a política aprendida π e a política especialista π^* em todas as demonstrações ξ no conjunto Ξ.

Embora seja uma simples e eficaz em algumas aplicações, essa abordagem apresenta um desempenho insatisfatório na maioria dos casos. Isso se deve ao fato do processo de aprendizado ser baseado apenas em um conjunto de amostras fornecidas pelo especialista. Em muitos casos, essas demonstrações não estão distribuídas uniformemente por todo o espaço de estados, o que torna provável que a política aprendida funcione de forma limitada ao se afastar dos estados observados em ξ, resultando no problema de *distribution shift* [16]. Isso implica que os erros cometidos em diferentes estados se acumulam, de modo que uma falha do agente pode levá-lo a um estado que o especialista nunca visitou e no qual o agente nunca foi treinado, resultando em possíveis falhas, como mostrado na figura 4.

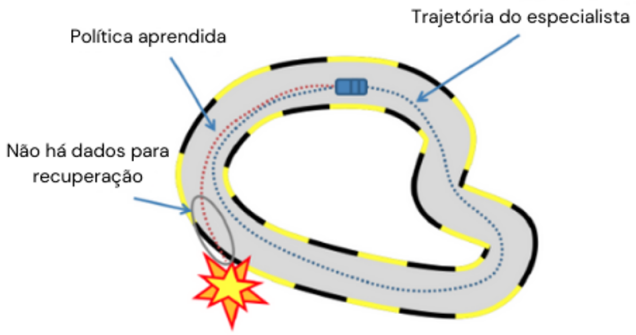


Figura 4: BC pode falhar se o agente cometer um erro que o desvia da trajetória do especialista. Fonte: Brunskill [20]

Inverse Reinforcement Learning (IRL) [21]. O BC não possui nenhuma maneira de compreender a intenção do especialista que o agente está tentando imitar, o que pode trazer limitações em tarefas mais complexas, especialmente quando a adaptação dinâmica a novos cenários é necessária. Dessa forma, o aprendizado por reforço inverso visa aprender uma representação da função de recompensa subjacente R que o especialista utilizou para gerar suas ações [16].

Ng e Russell [21] aponta que a função de recompensa é mais transferível em comparação com a política observada do agente. A recompensa aprendida pode ser aplicada diretamente se o agente receptor compartilhar o mesmo ambiente e objetivos do

agente original; caso contrário, ela ainda fornece uma base útil quando as especificações do agente diferem, como no caso em que o domínio do problema do agente receptor apresenta estados adicionais [22]. Dessa forma, o IRL possui maior capacidade de generalização quando comparado à replicação de políticas.

Algumas aplicações de IRL incluem o aprendizado do controle de voo de um helicóptero a partir das preferências de um operador especialista sobre 24 características [23] e a modelagem do comportamento de pedestres para permitir uma navegação robótica socialmente adaptativa, voltada a evitar colisões com pessoas [24].

2.2 Datasets para Aprendizado de Máquina

Métodos de aprendizado como IL e RL *offline* dependem de conjuntos de dados extensos, que podem ser obtidos de diferentes maneiras. A qualidade, o tamanho e a diversidade desses conjuntos de dados influenciam diretamente a robustez e a capacidade de generalização desses modelos [25]. Um conjunto de dados grande e diverso faz com que os modelos aprendam com uma extensa variedade de cenários, o que é particularmente importante em aplicações como saúde, robótica e carros autônomos, nos quais a variabilidade é inerente.

Em alguns casos, os dados são coletados de forma passiva, como por meio de interações de usuários na internet ou da gravação de informações de um carro para condução autônoma. Embora esses dados possam não estar necessariamente direcionados à tarefa específica a ser realizada, partes das sequências em um conjunto de dados como esse ainda podem fornecer informações úteis para a política a ser desenvolvida [25].

Para tarefas com objetivos claros, como a manipulação robótica e a navegação autônoma, os dados podem ser coletados em um ambiente controlado, onde as variáveis ambientais são cuidadosamente gerenciadas para garantir que os dados sejam relevantes e de alta qualidade. Nesses cenários, é possível utilizar tanto demonstrações realizadas por humanos quanto aquelas geradas por máquinas. De acordo com Mandlekar et al. [26], demonstrações humanas diferem dos conjuntos gerados por máquina devido a um processo de decisão não markoviano, já que humanos não agem exclusivamente com base na observação atual. Além disso, a qualidade dos dados e a estratégia de execução variam se os dados forem coletados por múltiplos humanos [27], o que não acontece para conjuntos gerados por máquina.

Uma das formas de lidar com à escassez de dados em certas áreas é o uso de dados sintéticos para o treinamento de modelos. Esses dados podem ser gerados automaticamente a partir de simulações, o que permite a criação de conjuntos de dados sem a necessidade de coleta no mundo real, como na geração de cenas de tráfego para o treinamento de carros autônomos [28, 29].

No entanto, a geração de dados sintéticos enfrenta um desafio técnico fundamental co-

nhecido como o *reality gap* ou *sim-to-real gap* [30], que refere-se à discrepância entre os dados gerados sinteticamente e a complexidade existente no mundo real. Uma abordagem para lidar com essa lacuna é a técnica de *domain randomization*, cujo objetivo é treinar os modelos em uma grande variedade de ambientes com características aleatoriamente modificadas. Isso permite que o modelo aprenda características invariantes ao domínio (real ou simulado), resultando em modelos mais transferíveis [29].

Outra forma de enfrentar esse problema é, além de superar a lacuna de domínio, abordar a lacuna de conteúdo [29]. Ou seja, é necessário lidar com o fato de que o conteúdo sintético replica um conjunto limitado de cenas, sem, necessariamente, refletir a diversidade e a distribuição dos objetos encontrados no mundo real. Reduzir essa lacuna tem sido um tema de grande interesse na robótica, pois oferece o potencial de aplicar algoritmos que até agora foram restritos a domínios simulados [30].

2.3 Conceitos Gerais de Manipulação Robótica

Um roteiro da NASA define manipulação na robótica como "[...] fazer uma mudança intencional no ambiente ou nos objetos que estão sendo manipulados." [1] Embora existam diversas definições desse conceito, este trabalho se concentra especificamente nesta, com ênfase em ações que favoreçam tarefas comumente realizadas na indústria, logística, saúde e, especialmente, no ambiente doméstico.

Algumas das principais tarefas realizadas por manipuladores robóticos incluem pegar, empilhar, cortar, abrir, fechar, empurrar, servir ou até escrever. Embora essas operações sejam complexas por natureza, raramente representam a totalidade das atividades que o manipulador precisa executar. Na prática, esses movimentos são componentes de tarefas mais amplas, como montar produtos em uma linha de produção, preparar alimentos ou limpar um cômodo.

No topo da hierarquia está uma única tarefa principal que define o propósito fundamental do robô autônomo, como, por exemplo, a manutenção de uma casa. Essa tarefa pode ser dividida em subtarefas, como lavar a louça, arrumar a cama e retirar o lixo. Essas subtarefas podem ser ainda mais divididas em suas próprias subtarefas. Por exemplo, a tarefa de retirar o lixo pode envolver abrir a tampa da lixeira, pegar o saco de lixo, levá-lo para fora da casa e descartá-lo adequadamente. Mesmo habilidades básicas, como pegar objetos, podem ser subdivididas em várias fases de ação orientadas por objetivos [7].

Mason [31] classifica as tarefas de manipulação robótica nas seguintes categorias: movimentos programados, movimentos complacentes, manipulação *pick-and-place* estruturada e não estruturada, mecânica de tarefas, manipulação dentro da mão (que consiste em ajustar a pose de um objeto após pegá-lo), manipulação não preênsil e manipulação com

[1] Tradução Livre: Manipulation pertains to making an intentional change in the environment or to objects that are being manipulated [31].

o corpo inteiro. Dentre essas, um dos desafios mais abordados em trabalhos atuais é a manipulação *pick-and-place* não estruturada, que envolve tarefas em que o robô precisa mover objetos cuja posição e forma apresentam incertezas.

Diferentemente do *pick-and-place* estruturado, a manipulação em ambientes não estruturados exige a consideração de fatores adicionais, como o planejamento de trajetórias, para evitar colisões e respeitar as limitações do manipulador, garras multi propósito capazes de agarrar todos os tipos de objetos, e o planejamento tanto das poses de *grasp* quanto das de soltura do objeto [31].

3 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados e discutidos os principais trabalhos relacionados ao tema deste estudo. A Seção 3.1 aborda as principais abordagens de aprendizado em manipulação robótica. A Seção 3.2 explora os métodos empregados para a coleta de *datasets* para tarefas de manipulação. A Seção 3.3 discute como o conceito de *affordance* tem sido tratado em pesquisas recentes. Por fim, a Seção 3.4 apresenta alguns dos simuladores mais utilizados na robótica atualmente.

3.1 Métodos de Aprendizado para Manipulação

Para manipular objetos, uma garra robótica precisa compreender sua relação com o objeto, incluindo a distância até ele e sua pose definida por seis graus de liberdade (6-DoF), englobando três eixos de translação e três de rotação. Isso geralmente é feito utilizando LiDARs, que utilizam pulsos de laser para calcular distâncias, ou câmeras RGB-D, que incluem informação de profundidade junto a imagem, transformando os dados em uma representação 3D como nuvens de pontos ou malhas [32]. Segundo Zhang et al. [33], os dois principais métodos utilizados para o controle de manipulação de robôs são o aprendizado por reforço (*Reinforcement Learning* - RL) e o aprendizado por imitação (*Imitation Learning* - IL).

Apesar de o aprendizado por reforço tradicional ter obtido sucesso em diversas aplicações no passado, as abordagens eram intrinsecamente limitadas a problemas de baixa dimensionalidade [34]. Isso se demonstra inviável para desafios mais sofisticados, como os encontrados em problemas reais de manipulação robótica. Nesse sentido, o aprendizado por reforço profundo (*Deep Reinforcement Learning*), que combina técnicas de RL com redes neurais artificias, tem se mostrado eficaz [33], permitindo que robôs aprendam comportamentos diretamente a partir de sinais de entrada de alta dimensionalidade, como imagens [35, 36], ou nuvens de pontos [37, 38, 39].

Diante disso, uma abordagem existente é a utilização de nuvens de pontos para determinar pontos de contato em objetos. Uma metodologia proeminente nessa área é a de Mousavian et al. [40], que emprega a arquitetura PointNet++ [41] junto a um *Variational*

Auto-Encoder [42] para extrair características tridimensionais de cenas e, assim, gerar poses de *grasp* com 6 graus de liberdade, exemplificadas na Figura 5. O treinamento do modelo é realizado utilizando dados sintéticos gerados por um simulador, e, posteriormente, testado em ambientes reais. Além de gerar diversas poses para um único objeto, o método se destaca por introduzir uma rede avaliadora, que verifica a qualidade do *grasp* e a refina de forma iterativa.

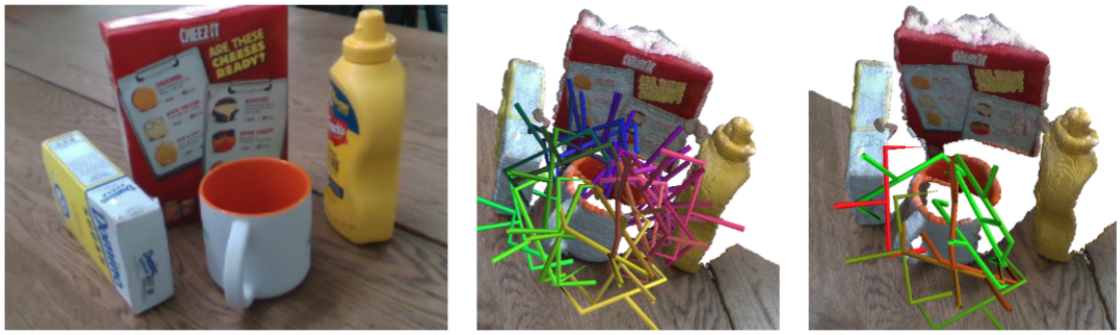


Figura 5: Visualização das poses previstas para uma caneca. Cada pose é definida por 3 valores de posição e 3 de orientação, com a cor indicando a qualidade estimada do *grasp*. Fonte: Mousavian et al. [40]

Por outro lado, uma abordagem muito relevante para o presente trabalho é o ensino de tarefas através de IL ou RL *offline*, pois, ao contrário de métodos baseados exclusivamente em pontos de contato, possibilita a transmissão de tarefas complexas com conhecimento especializado mínimo, sem a necessidade de programação explícita [43]. De acordo com Mandlekar et al. [26], métodos de IL *offline* são em grande parte variações de BC, em que uma política é treinada para gerar as mesmas ações que as realizadas pelo demonstrador em cada estado.

Trabalhos anteriores em aprendizado por imitação têm demonstrado a capacidade de generalização *one-shot* [44, 45] ou *zero-shot* [46, 47] para novos objetos, ou seja, os modelos são capazes de generalizar para um novo objeto com nenhuma demonstração prévia (*zero-shot*), ou apenas uma demonstração (*one-shot*). No entanto, a generalização *zero-shot* para novas tarefas ainda é um desafio, especialmente ao considerar tarefas de manipulação baseadas em visão que envolvem uma variedade de habilidades com diferentes objetos. Uma das formas de alcançar esse tipo de generalização depende de superar desafios relacionados à ampliação da coleta de dados [47], um dos principais objetivos do presente trabalho.

3.2 Métodos de Coleta de *Datasets* para Manipulação

Uma das limitações de IL e RL *offline* é a dificuldade de obter conjuntos de dados suficientemente grandes e diversificados para treinar redes neurais capazes de generalizar em uma variedade ampla de tarefas [48]. Alguns trabalhos anteriores se limitam a ambi-

entes 2D simples [49] ou políticas codificadas manualmente [50, 51], mas a aplicação em tarefas mais complexas pode ser limitada.

Para contornar essa limitação, diversas abordagens exploram vídeos não estruturados de pessoas realizando tarefas de manipulação como forma de ensinar robôs a reproduzir esses movimentos [52, 53], mas apresentam uma significativa lacuna de incorporação em relação aos robôs, dificultando a sua aplicação direta [48]. Outras estratégias empregam operadores humanos teleoperando braços robóticos através de diferentes interfaces de controle como *spacemouses* 3D [54], controladores de VR ou AR [47, 55], *smartphones* [56] e dispositivos hápticos [57]. Apesar de mostrar bons resultados, essa estratégia é onerosa e demorada, pois depende da participação de operadores humanos qualificados e equipamentos especializados. Por esse motivo, o presente trabalho opta por utilizar trajetórias geradas automaticamente, ou seja, soluções que permitem a geração de trajetórias sem a necessidade de intervenção direta de um operador.

Outro fator dificultante é treinar modelos em conjuntos de dados muito variados, onde a posição da câmera e o tipo de robô não são padronizados. Diferentemente de áreas como visão computacional e processamento de linguagem natural, onde os formatos de dados são bem definidos, a robótica ainda carece de uma uniformidade tanto em termos de configurações de *hardware*, como câmeras e sensores, quanto de robôs [46]. Dentre os métodos para lidar com esse problema que mais se destacam na literatura estão:

OXE [58]. O *Open X-Embodiment* é o maior *dataset* de manipulação robótica, com mais de 1 milhão de trajetórias e 22 configurações de corpo do robô. Esse *dataset* foi criado ao combinar 60 conjuntos de dados robóticos provenientes de 34 laboratórios de pesquisa ao redor do mundo, unificando-os em um formato consistente para facilitar o uso. O principal objetivo do OXE é viabilizar a transferência de aprendizado entre diferentes robôs por meio de um *dataset* diverso.

No entanto, um *dataset* como o OXE exige que dados sejam coletados ao longo de um período prolongado em diversos ambientes e configurações. Essa dimensão de trabalho torna o processo de coleta e padronização de dados um desafio logístico e técnico significativo. Além disso, a implantação desses modelos em novos ambientes ainda exige a coleta de dados para ajuste fino.

Dessa forma, uma alternativa que tem sido explorada é o uso de garras manuais equipadas com sensores como interface de coleta de dados [46, 48, 59], o que reduz a lacuna entre os dados coletados e o mundo real, além de facilitar o processo de coleta.

UMI [48]. O *Universal Manipulation Interface*, Figura 6, é uma plataforma projetada para transferir demonstrações humanas coletadas em ambientes reais para políticas de controle robótico. Comparado aos outros métodos, os dados coletados com o UMI apresentam uma lacuna mínima de incorporação nos espaços de ação e de observação, eliminando a necessidade de robôs físicos ou simulados durante a coleta de dados e fornecendo dados e políticas que são transferíveis para diferentes configurações de robôs.

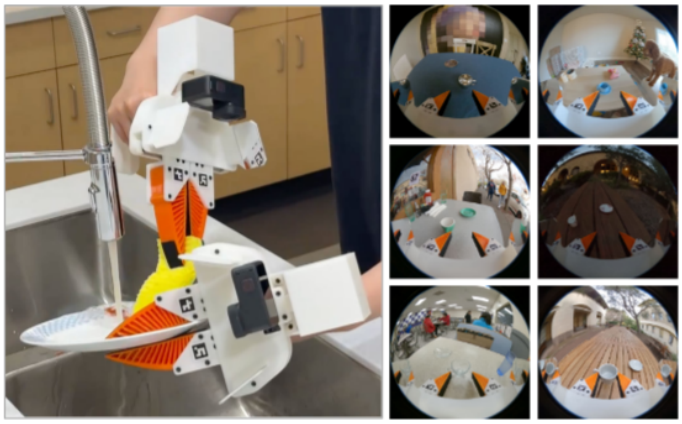


Figura 6: Operador coletando um conjunto de dados com a plataforma UMI. Fonte: Chi et al. [48]

Embora seja uma alternativa viável para a coleta de dados em larga escala, métodos como o UMI apresentam algumas desvantagens que limitam sua aplicabilidade. Primeiramente, é necessário o uso de sensores específicos para a captura das demonstrações, além do sistema ainda depender de operadores humanos para executá-las. Além disso, a escalabilidade do UMI para diferentes configurações de robôs é limitada, uma vez que a garra do robô precisa ser compatível com as configurações utilizadas no conjunto de dados capturado [48].

MimicGen [60]. Diante desses desafios, o presente trabalho utiliza o MimicGen, um método que permite criar grandes conjuntos de dados a partir de um número limitado de demonstrações, adaptando-as para novas configurações de robôs e ambientes. Através de um conjunto de dados original $\mathcal{D}_{\text{src}}$, composto por um pequeno número de demonstrações humanas em uma tarefa t, o MimicGen é capaz de expandi-lo para criar um conjunto de dados maior, $\mathcal{D}$, que inclui tanto a tarefa original quanto variações dela, permitindo alterações na distribuição do estado inicial, nos objetos envolvidos ou no braço robótico.

O processo para gerar uma nova demonstração tem as seguintes etapas: (1) um estado inicial é selecionado a partir da tarefa para a qual se deseja gerar dados, (2) uma demonstração $\tau \in \mathcal{D}_{\text{src}}$ é escolhida e adaptada para produzir uma nova trajetória de robô τ', (3) o robô executa a trajetória τ' na cena atual, e, se a tarefa for concluída com sucesso, a sequência de estados e ações é adicionada ao conjunto de dados $\mathcal{D}$ [60].

A partir de 10 demonstrações, o método é capaz de gerar um conjunto de dados com mais de 1000 demonstrações. Ao comparar o desempenho de agentes treinados em $\mathcal{D}_{\text{src}}$ com aqueles treinados em $\mathcal{D}$, foi observada uma melhoria consistente em todas as tarefas, com um aumento de até 80% na taxa de sucesso em tarefas específicas [60].

Além disso, o desempenho dos agentes treinados no conjunto $\mathcal{D}$ foi comparado ao de agentes treinados em conjuntos de dados compostos inteiramente por demonstrações humanas. Em muitos casos, os agentes apresentaram desempenho similar, evidenciando que as demonstrações geradas pelo MimicGen podem ser tão efetivas quanto dados coletados

manualmente. Isso sugere que, com o MimicGen, é possível alcançar resultados relevantes sem a necessidade de extensa coleta manual, economizando tempo e recursos [60].

3.3 Métodos de Manipulação Baseados em *Affordance*

Na área de manipulação robótica, o conceito de *affordance* refere-se à manipulação de objetos com base na ação específica que se deseja realizar. Ele representa um avanço importante para tarefas de longo alcance, onde o foco se estende além de simplesmente pegar ou manusear um objeto, considerando como ele será utilizado em ações subsequentes (por exemplo, um pano pode ser agarrado de forma diferente dependendo se o objetivo é dobrá-lo ou limpar uma superfície) [61]. Embora o presente trabalho não tenha como objetivo inferir a *affordance* dos objetos, mas sim manipulá-los com base nesse conceito, esta seção visa contextualizar como a *affordance* tem sido abordada no campo da manipulação robótica.

Diversos estudos empregam redes neurais profundas para segmentar partes de um objeto com base em suas respectivas *affordances*, seguido pelo uso de métodos analíticos para encontrar pontos de contato na área segmentada [62]. Ju et al. [63] utiliza vídeos de ações realizadas por humanos para criar uma memória de *affordance* através das interações entre humanos e objetos, encontrando o ponto ideal de contato e generalizando para objetos ainda não vistos através de modelos de correspondência semântica, Figura 7. Após identificar o ponto de contato no objeto, o método AnyGrasp [37] é utilizado para encontrar possíveis pegadas, e a mais próxima do ponto determinado pelo método é selecionada. Embora atinja uma taxa de sucesso de 85,7% em tarefas de manipulação com 7 categorias de objetos, a *affordance* considerada está restrita a uma única maneira correta de segurar cada objeto, o que limita sua aplicação em diferentes tipos de tarefas.

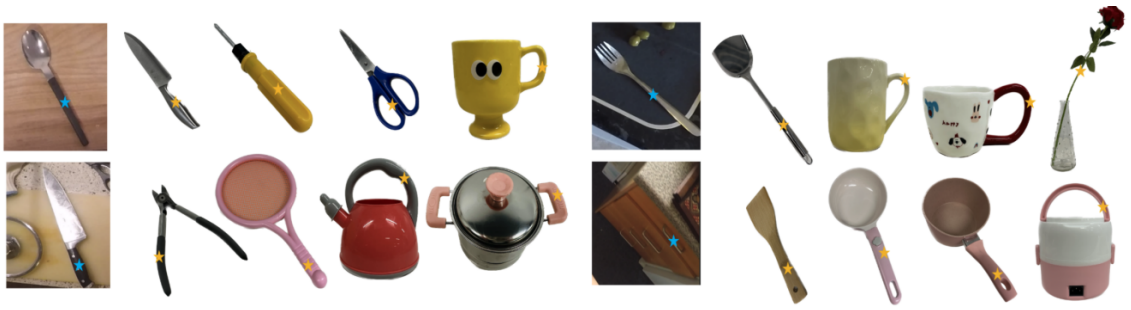


Figura 7: Ponto de *affordance* generalizado para diferentes objetos. As estrelas azuis representam o ponto extraído de vídeos e as amarelas os pontos inferidos pelo método [63].

Ardón et al. [64] se destaca por considerar múltiplos usos para um único objeto, desenvolvendo uma representação de grafo por meio de uma rede lógica de Markov para obter as probabilidades de diferentes *affordances* no manuseio do objeto. Liu et al. [65] propõe o *Context-Aware Grasping Engine* (CAGE), um método que considera tanto as

restrições do objeto quanto as do objetivo da tarefa, permitindo seleções de *grasp* mais adequadas com base na tarefa específica de manipulação. Além disso, o trabalho apresenta um conjunto de dados contendo objetos segmentados com base em suas *affordances* e associados a sete diferentes tipos de tarefas.

Os conjuntos de dados de *affordances* concentram-se principalmente em objetos, seja por meio da anotação manual de pontos [64, 66] ou pela inferência através de vídeos ou imagens [63, 67, 68]. No entanto, essas abordagens apresentam limitações, como a dependência de anotações manuais e a dificuldade de capturar contextos mais amplos relacionados às tarefas.

A Tabela 1 apresenta uma comparação de trabalhos que utilizam o conceito de *affordance*, abrangendo estudos publicados entre 2016 e 2024. Os trabalhos listados variam em tipo, englobando tanto abordagens focadas exclusivamente em inferência quanto aquelas que combinam inferência com a criação de *datasets*. Além disso, a coluna "Multiuso" indica se o método considera apenas uma *affordance* por objeto ou se explora múltiplas possibilidades de interação para os mesmos.

Trabalho	**Ano**	**Tipo**	**Multiuso**	**Entrada**	**Ambiente**
[69]	2016	Inferência	✗	RGB-D	Real
[70]	2017	Inferência	✓	RGB	Real
[71]	2018	Inf + *Dataset*	✓	RGB-D	Sim + Real
[64]	2019	Inf + *Dataset*	✓	RGB + 3D	Sim + Real
[65]	2019	Inf + *Dataset*	✓	*Point Cloud*	Real
[68]	2019	Inf + *Dataset*	✓	RGB-D + Termal	Real
[72]	2020	Inf + *Dataset*	✓	RGB-D	Real
[66]	2020	Inferência	✗	*Point Cloud*	Sim
[67]	2023	Inferência	✗	Vídeos	Sim + Real
[73]	2023	Inferência	✓	Texto + *Point Cloud*	Real
[74]	2023	Inf + *Dataset*	✓	Texto + RGB-D	Real
[63]	2024	Inferência	✗	Vídeos	Real

Tabela 1: Comparação de trabalhos com *affordance*.

O conceito de *affordance* assume diferentes nomenclaturas ao longo da literatura, sendo frequentemente apresentado sob o termo *Task-Oriented Grasping* [71, 72, 73, 74]. Essa perspectiva ressalta a conexão entre o ato de agarrar o objeto e a realização de uma tarefa subsequente, alinhando-se indiretamente aos princípios de *affordance*.

Nesse sentido, Fang et al. [71] demonstra que a compreensão do modelo sobre *affordance* de objetos emerge de forma natural ao treinar simultaneamente as ações de *grasping* e manipulação. No entanto, a abordagem apresentada por esse trabalho é limitada a ferramentas simples, um conjunto de dados restrito com apenas duas tarefas e um manipulador com apenas 4 graus de liberdade, o que simplifica significativamente o problema.

3.4 Simuladores para a Robótica Doméstica

Os avanços recentes em IA têm sido impulsionados pelo treinamento de modelos de redes neurais com conjuntos de dados acumulados ao longo de décadas de informações coletadas por humanos. No entanto, ao contrário da visão computacional e linguagem natural, nos quais dados visuais e textuais em grande quantidade estão amplamente disponíveis a partir de fontes online, os dados para robótica são relativamente escassos [48].

Esse contexto, somado ao elevado custo de robôs, torna a simulação indispensável para o teste e desenvolvimento na robótica. Em ambientes simulados, é possível coletar dados sintéticos de forma rápida e em grandes quantidades, enquanto os custos e riscos associados à testes são praticamente inexistentes. Falhas podem ser corrigidas e iteradas rapidamente, possibilitando a realização de novos testes de forma imediata [75].

A solução padrão atual para simulação de robôs baseados em ROS [75] é o Gazebo [76], uma solução de código aberto em desenvolvimento desde 2001. Sua popularidade na ciência e no desenvolvimento de software é evidenciado pelo número de publicações que utilizam o termo "Gazebo ROS" na base de dados do IEEE. Devido à sua longa presença na área, o Gazebo possui uma base de usuários científicos significativamente maior em comparação à outras alternativas.

No entanto, para tarefas mais especializadas na robótica, é comum o uso de simuladores que, apesar de não possuírem a amplitude de aplicação do Gazebo, são apropriadas para cenários específicos. Como exemplo, simuladores como AI2-THOR [77], Robosuite [78] e Habitat 2.0 [79] são projetados para fornecer simulações realistas de ambientes internos, com foco em tarefas como manipulação de objetos e execução de tarefas domésticas. Entre os principais diferenciais desses simuladores estão a capacidade de gerar ambientes fotorrealistas [77, 78, 79, 80], a disponibilização de cenas em escala, objetos e tarefas pré-definidas [78, 79, 80, 81] e até mesmo a inclusão de tarefas e objetos gerados por IA, garantindo diversidade praticamente ilimitada [80].

Robosuite [78]. O robosuite é uma estrutura modular de simulação e *benchmark* para aprendizado de robôs. Baseado no mecanismo de física MuJoCo [82], seu principal objetivo é apoiar a pesquisa e o desenvolvimento de algoritmos e técnicas robóticas orientados por dados. As principais vantagens do robosuite em comparação com outros simuladores incluem seu design modular, que oferece maior flexibilidade na criação de novos ambientes de simulação, a disponibilização de controladores robóticos e algoritmos de aprendizado já implementados, além de um conjunto de tarefas de *benchmark* voltado para a avaliação e o desenvolvimento de algoritmos [78].

A versão 1.0 do robosuite inclui sete modelos de robôs, oito modelos de garras, seis modos de controladores e nove tarefas padronizadas [78]. Diversos trabalhos referências em manipulação robótica utilizam o robosuite como um *benchmark* de algoritmos [26, 27, 54] ou para a criação de *datasets* [26, 60, 80].

4 METODOLOGIA E FERRAMENTAS UTILIZADAS

Esse capítulo tem como objetivo apresentar as metodologias empregadas e as ferramentas utilizadas no desenvolvimento do presente trabalho. Inicialmente, a Seção 4.1 descreve o uso do simulador escolhido, destacando as adaptações realizadas para alinhar a ferramenta ao objetivo proposto. Em seguida, as Seções 4.2 e 4.3 detalham o desenvolvimento do gerador de *datasets*, que constitui a principal parte do trabalho. Por fim, a Seção 4.4 apresenta os métodos de validação dos *datasets* gerados, enquanto a Seção 4.5 descreve os experimentos realizados.

4.1 Simulação com robosuite

O simulador utilizado nesse trabalho é o robosuite [78], escolhido devido à sua modularidade e a ampla adoção em pesquisas relacionadas à manipulação robótica. O robosuite oferece suporte a diversos modelos de robôs, garras e tarefas padronizadas, simplificando a criação de ambientes de simulação. Sua integração com o MuJoCo, que executa simulações física de dinâmicas de contato de forma rápida [82], o torna particularmente adequado para estudos envolvendo aprendizado e controle robótico [78]. Essas características fazem dele uma ferramenta ideal para atender aos objetivos do presente trabalho, proporcionando um ambiente confiável e configurável para a criação de *datasets* e *benchmark* de algoritmos.

4.1.1 *Environments*

Os ambientes (*environments*) são os principais objetos do robosuite com os quais o código externo irá interagir. Cada ambiente representa uma tarefa de manipulação e integra os três elementos fundamentais de uma simulação robótica: modelos de robôs e garras, carregados a partir de arquivos XML; modelos de objetos, que podem ser definidos como modelos 3D ou programados diretamente; e uma arena, que define o espaço de trabalho do robô [78].

Esses ambientes são criados por meio de uma função que aceita diversos parâmetros configuráveis, incluindo o nome da tarefa, o modelo do braço robótico a ser utilizado,

como Sawyer [1] ou Panda [2], as configurações dos controladores, o modelo de recompensa da tarefa, entre outros aspectos.

As tarefas são instâncias da classe `Task` que utilizam um inicializador de posicionamento como entrada, responsável por determinar a distribuição do estado inicial do ambiente. Esse inicializador amostra posicionamentos válidos e sem colisões para todos os objetos na cena no início de cada episódio, permitindo não apenas a variação dos tipos de objetos, mas também de suas posições e orientações [78].

4.1.2 Desenvolvimento do Ambiente

Para o desenvolvimento do presente trabalho, é necessário a criação de um ambiente no robosuite que atenda às especificações do objetivo proposto. Çalli et al. [83] destaca que, em tarefas de manipulação, um *dataset* deve incluir objetos que apresentem uma grande variedade de formas, tamanhos, pesos, rigidez e texturas, além de abrangerem diversas aplicações e desafios de manipulação.

Para a criação de ambientes mais variados, foram utilizados objetos provenientes de três fontes principais. A primeira é o Objaverse [84], um extenso conjunto de dados que reúne mais de 800 mil objetos 3D, projetado para facilitar aplicações em aprendizado de máquina e robótica. A segunda é o ShapeNet [85], um repositório de modelos 3D anotados, com mais de 3 milhões de formas categorizadas em 3.135 classes. Além dessas bases de dados, também foram incorporados objetos nativos do simulador robosuite, que oferece modelos especialmente customizáveis para simulação de tarefas robóticas.

Utilizando a API do robosuite, é possível modificar versões existentes de uma `Task`. Através disso, foram desenvolvidas funções capazes de randomizar elementos na cena, aumentando a complexidade das tarefas. Por exemplo, a posição dos objetos pode ser ajustada a cada nova instância por meio do inicializador de posicionamento, a garra do robô pode ser substituída e até a aparência da mesa onde os objetos estão dispostos pode ser alterada. A Figura 8 ilustra como essas variações podem ser aplicadas em uma tarefa de manipulação, destacando as mudanças nos componentes da cena a cada nova inicialização.

4.1.3 Criação da Tarefa

Uma tarefa no robosuite envolve um robô com uma garra como seu atuador, uma arena (área de trabalho) e objetos com os quais o robô interage. As tarefas de manipulação no Robosuite são comumente desenvolvidas a partir do ambiente *ManipulationEnv*, que serve como uma implementação base para criar ambientes de manipulação [78]. No entanto, para criar uma tarefa personalizada, é necessário modificar algumas das funções pré-implementadas na classe base, permitindo a adaptação do ambiente às necessidades

[1] *Rethink Robotics*: `https://rethinkrobotics.com/`
[2] *Franka Robotics*: `https://franka.de/`

Figura 8: Diferentes instâncias de uma tarefa de manipulação no robosuite. Em cada imagem, o modelo de garra, o modelo e número de canecas, a textura da mesa, as posições e orientações dos objetos são randomizadas. Fonte: Elaborada pelo Autor.

específicas da nova tarefa.

Entre os elementos mais relevantes estão os sensores, que podem incluir diferentes tipos de câmeras, como as acopladas ao robô, e câmeras de visão superior ou externas, proporcionando múltiplas perspectivas do ambiente. Sensores adicionais, como os de força e proximidade, devem ser integrados mediante a implementação de funções complementares, de acordo com os requisitos da tarefa. Por exemplo, em tarefas de manipulação, a função *check_grasp* permite verificar se o *gripper* está segurando o objeto especificado no ambiente. Por padrão, essa função retornará *True* se pelo menos uma das geometrias nos dedos da garra estiver em contato com qualquer geometria associada ao objeto especificado.

Cada tarefa no robosuite possui objetivos bem definidos, como agarrar, levantar, empurrar ou organizar objetos. O sucesso da execução é avaliado por meio da função *check_success*, que considera critérios específicos, como se um objeto foi colocado na área correta ou se o robô conseguiu agarrar um objeto corretamente. Para algoritmos de aprendizado por reforço, é possível configurar uma função de recompensa que especifica as recompensas atribuídas a cada etapa da execução da tarefa, orientando o aprendizado do modelo.

4.2 Gerador de *Datasets*

O processo de geração de *datasets*, ilustrado na Figura 9, é composto por cinco estágios. No primeiro estágio, a região de *affordance* do objeto de interesse é definida, baseada nos rótulos propostos por Ardón et al. [64]. Os estágios subsequentes envolvem o desenvolvimento de um *script* projetado para automatizar o processo de coleta de dados.

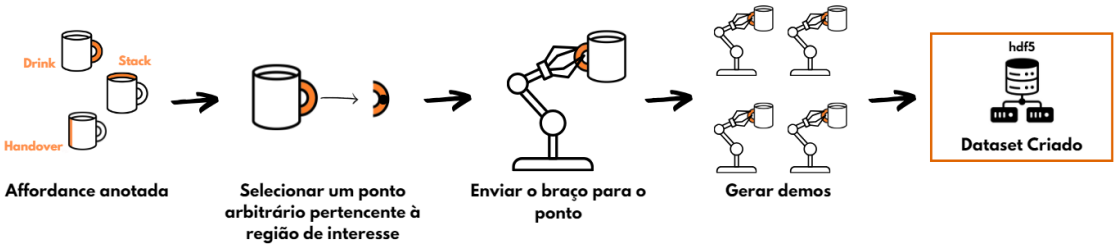


Figura 9: *Pipeline* de criação do *Dataset*. Fonte: Elaborada pelo Autor.

Esse *script*, descrito no Algoritmo 1, é responsável por selecionar de forma aleatória um ponto dentro da região de *affordance* definida, mover o braço robótico até esse ponto, agarrar o objeto, e executar as ações necessárias para gerar demonstrações. Para implementar esse processo, foi utilizado como base o método *collect_human_demonstrations* do robosuite, originalmente projetado para a coleta de lotes de demonstrações controladas por humanos, como mencionado na Seção 3.2.

Devido à necessidade de equipamentos especializados e operadores humanos proficientes para realizar demonstrações teleoperadas, optou-se por uma abordagem automatizada para a geração das demonstrações. Assim, o *script* foi ajustado para substituir os dados de entrada provenientes de dispositivos de entrada/saída (*I/O Devices*), como um *spacemouse* 3D, por comandos programados diretamente no ambiente de simulação.

Com esse objetivo, foi utilizado o *Operational Space Control* (OSC) [86] para transformar as ações de alto nível em comandos virtuais de baixo nível que acionam os motores dos robôs. Nesse controlador, o valor desejado é definido como uma pose 6D, composta por posição e orientação, associada a um *frame* de controle definido no modelo da garra, posicionado entre os dedos. O OSC calcula os torques articulares necessários para minimizar o erro entre a pose desejada e a atual da garra, utilizando a mínima energia cinemática necessária.

A pose desejada consiste em uma posição arbitrária dentro da região de interesse do objeto a ser agarrado e uma orientação, determinada através do rótulo de *affordance*. Com base nessa pose, o controlador direciona o braço robótico até o ponto especificado no objeto e executa o fechamento da garra. Após a verificação do método *check_grasp* confirmar que o objeto foi corretamente agarrado, o estágio de *grasp* é concluído, iniciando o estágio de manipulação, no qual o braço é movimentado para atender aos requisitos específicos da tarefa.

Todas as tarefas possuem o método *check_success*, que tem a função de verificar se os estágios de *grasp* e manipulação foram concluídos de forma adequada. Utilizando esse método, o *script* desenvolvido armazena apenas as demonstrações bem-sucedidas no formato *Hierarchical Data Format Version 5* (HDF5) [87], descartando aquelas em que a tarefa não foi realizada com sucesso.

O formato HDF5 foi escolhido por sua estrutura hierárquica, semelhante a um sistema

de arquivos, com grupos que podem armazenar subgrupos ou *datasets*. No contexto deste trabalho, cada arquivo gerado pelo método descrito contém subgrupos que representam cada uma das demonstrações capturadas. A organização e o conteúdo desses subgrupos estão detalhados na Tabela 2.

Grupo	Dataset/Subgrupo
data	date
	time
	environment
	↓ **demo_0**
	actions
	dones
	↓ **obs**
	agentview_image
	object
	robot0_eef_pos
	robot0_eef_quat
	robot0_eef_vel_ang
	robot0_eef_vel_lin
	robot0_eye_in_hand_image
	robot0_gripper_qpos
	robot0_gripper_qvel
	robot0_joint_pos
	robot0_joint_pos_cos
	robot0_joint_pos_sin
	robot0_joint_vel
	rewards
	states
	→ **demo_1**
	→ **demo_2**
	⋮

Tabela 2: Estrutura hierárquica de uma demonstração armazenada no arquivo hdf5. Cada demonstração contém *datasets* que registram ações, sinais de término da tarefa, recompensas, estados, além de um subgrupo dedicado às observações.

4.3 Ampliação do *Dataset*

Grande parte dos *datasets*, sejam eles coletados por humanos ou gerados automaticamente, frequentemente contém habilidades de manipulação semelhantes aplicadas em diferentes contextos ou situações. Um exemplo disso é que operadores humanos tendem a demonstrar trajetórias robóticas quase idênticas para agarrar uma caneca, independentemente de sua localização em diferentes superfícies [60].

Com base nesse princípio, este trabalho emprega a ferramenta MimicGen, detalhada na Seção 3.2, para replicar essas trajetórias em novos contextos, permitindo a geração de dados diversificados com um esforço humano reduzido. O MimicGen foi escolhido devido a sua fácil integração com o robosuite e comprovado bom desempenho em tarefas de longo horizonte e alta precisão que exigem diferentes habilidades de manipulação, como pegar e posicionar (*pick-and-place*) [60].

O primeiro passo para utilizar o MimicGen consistiu da criação de uma subclasse da classe base de ambiente. No caso do Robosuite, essa classe base é a `RobosuiteInterface`. Para adaptar essa interface às necessidades do projeto, foi necessário implementar dois métodos: *get_object_poses* e *get_subtask_term_signals*.

O MimicGen gera dados a partir da composição de segmentos de subtarefas [60]. Durante esse processo, é importante que o MimicGen tenha uma forma de capturar a pose do objeto relevante no início de cada subtarefa. Para isso, utiliza-se o método *get_object_poses*, que deve retornar um dicionário associando o nome do objeto à sua matriz de pose direto do robosuite.

Em seguida, foi implementado o método *get_subtask_term_signals*. Esta função tem o propósito de fornecer sinais binários que indicam a conclusão de cada subtarefa em cada passo de tempo das demonstrações de origem. Esses sinais segmentam cada demonstração em partes contínuas, permitindo que o MimicGen trate cada subtarefa de forma independente durante a geração de dados [60]. Cada subtarefa corresponde a um sinal de terminação específico, exceto a subtarefa final, que se estende até o término da demonstração original.

Subsequentemente, foi gerado um arquivo de configuração para a tarefa no MimicGen. Esses arquivos têm o objetivo de especificar todas as características da geração de dados, incluindo o ambiente de simulação, os robôs e garras utilizadas, o número de demonstrações a serem coletadas, as estratégias de seleção de ações, os ruídos aplicados, os parâmetros de interpolação, as câmeras para coleta de observações e as especificações de cada subtarefa envolvida na tarefa principal.

Após a conclusão desse processo, o MimicGen está apto para ser utilizado no ambiente customizado desenvolvido pelo autor. Com a configuração adequada, é possível gerar dados sintéticos em volumes muito superiores aos do conjunto de dados original, permitindo o treinamento de modelos de aprendizado de máquina sem a necessidade de coleta extensiva de dados.

Assim, um conjunto de 15 demonstrações originais foram processadas para gerar *datasets* com 200 demonstrações, com variações que incluem diferenças na posição e orientação inicial dos objetos, assim como a trajetória da garra. A escolha do número de demonstrações foi fundamentada em Mandlekar et al. [60], que evidencia um salto significativo no desempenho em 200 demonstrações.

4.4 Validação

A validação de um conjunto de dados é uma etapa importante para garantir a confiabilidade de qualquer análise ou modelo desenvolvido a partir dele. Existem diversos métodos para validar conjuntos de dados, cada um adequado a diferentes contextos e tipos de dados. Neste trabalho, ele será avaliado através de diferentes algoritmos que devem aprender a realizar a tarefa especificada com o *dataset* desenvolvido.

A distribuição dos dados é uniforme, com 80% destinados ao conjunto de treinamento, 10% ao conjunto de validação e 10% ao conjunto de teste. Para avaliar o desempenho do conjunto de dados, foram utilizados dois métodos:

BC-RNN [26]. O BC-RNN é uma variante do Behavioral Cloning (BC), Seção 2.1.4, que utiliza uma Rede Neural Recorrente (RNN) como política. Isso permite que a política modele dependências temporais nos dados por meio de um estado oculto que retém informações de etapas anteriores. Conforme demonstrado por Mandlekar et al. [26], o BC-RNN supera o BC em todas as tarefas de manipulação testadas, especialmente em conjuntos de dados subótimos, que contêm demonstrações coletadas por humanos não especializados.

IRIS [27]. O IRIS é uma abordagem de RL *offline*, Seção 2.1.3, desenvolvida especificamente para tarefas de manipulação robótica. Seu funcionamento baseia-se em um componente de alto nível definindo estados-alvo que um controlador de baixo nível deve alcançar, alternando o controle entre esses níveis. Essa estrutura permite que o IRIS aproveite demonstrações de tarefas em larga escala, mesmo quando apresentam desempenho subótimo e diversidade nos cenários, tornando-o ideal para o presente trabalho.

Seguindo o padrão estabelecido por Mandlekar et al. [26], foram utilizados dois espaços de observação: *low-dim* e *image*. Ambas as modalidades incluem as informações descritas na Tabela 2, diferenciando-se apenas no uso do *ground truth* da posição do objeto (*low-dim*) ou na substituição dessas informações pelas observações disponíveis da câmera (*image*). O algoritmo IRIS foi excluído do treinamento baseado em imagens, pois sua dependência das reconstruções de subobjetivos torna-o problemático para imagens de alta dimensionalidade [26]. Para cada modalidade, o agente foi treinado por $\mathcal{N}$ épocas, sendo avaliado periodicamente a cada $\mathcal{E}$ épocas. Detalhes adicionais sobre os hiperparâmetros utilizados encontram-se no Apêndice C.

Para avaliar os resultados das redes, utilizou-se o *checkpoint* com a maior taxa de sucesso durante o treinamento, realizando três execuções com 50 tentativas cada para a tarefa. A média da taxa de sucesso e o desvio padrão das execuções foram calculados e organizados em uma tabela para cada experimento desenvolvido.

4.5 Experimentos

Esta seção apresenta os experimentos conduzidos para avaliar a proposta deste trabalho, com o objetivo de responder às seguintes questões: **Q1.** É viável gerar um *dataset* de *affordance* diversificado, contendo múltiplas variáveis, utilizando o robosuite? **Q2.** Quais variáveis causam maior impacto no desempenho dos modelos? **Q3.** Os dados gerados permitem o aprendizado implícito de *affordances* por meio de técnicas de aprendizado de máquina? **Q4.** Utilizando os conjuntos de dados gerados, os modelos atuais são capazes de generalizar a execução da tarefa para composições inéditas?

Para todos os experimentos, foi desenvolvido um ambiente utilizando o robosuite. Esses ambientes incluem a versão base do braço robótico Panda Research 3 [3], uma mesa, uma câmera acoplada à extremidade do braço robótico e uma câmera de visão superior. Informações detalhadas sobre os experimentos realizados estão disponíveis no Apêndice B.

4.5.1 Experimento 1: Prova de Conceito

O primeiro experimento realizado consistiu em uma prova de conceito do método proposto. Este experimento buscou avaliar se diferentes métodos de aprendizado de máquina são capazes de manipular o objeto de acordo com a *affordance* definida.

A tarefa desenvolvida foi baseada na tarefa *Lift*, disponível no robosuite. Originalmente, essa tarefa envolve o uso de um braço robótico para localizar, agarrar e levantar um cubo presente na cena. Na versão adaptada para este estudo, o cubo foi substituído pelo objeto *PotWithHandles*, também fornecido pelo robosuite, que é colocado na cena com posição e orientação aleatórias. Este objeto foi segmentado em duas regiões de *affordance*, como ilustrado na Figura 10. O objeto não possui funcionalidades específicas atribuídas a cada *affordance*, apenas regiões distintas que o robô deve agarrar, com o objetivo de provar o conceito de aprendizado da tarefa.

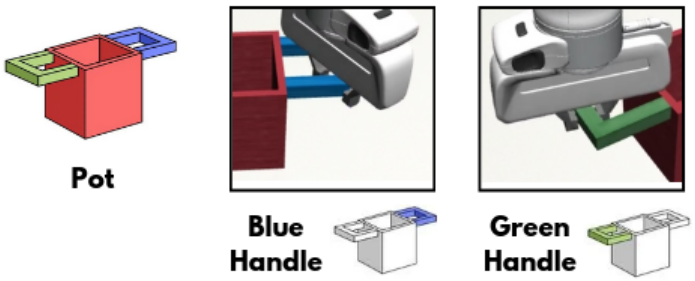


Figura 10: Divisão do objeto *PotWithHandles* em duas áreas: alça azul e alça verde. Fonte: Elaborado pelo Autor.

[3] Website: https://franka.de/products

A tarefa funciona da seguinte forma: 1º) o objeto é posicionado de forma aleatória dentro de uma região previamente definida na mesa, com orientação também aleatória; 2º) o braço robótico deve agarrar o objeto pela alça especificada e levantá-lo. O sucesso da tarefa é avaliado pelo método *check_success*, que avalia a altura do objeto em relação à superfície da mesa e utiliza a verificação *check_grasp* para confirmar se o objeto foi corretamente agarrado.

4.5.2 Experimento 2: *Affordance* de Uma Caneca

O segundo experimento buscou explorar o uso da *affordance* para aplicações práticas do cotidiano, além de criar um conjunto de dados diversificado que englobe múltiplas variáveis. Para alcançar esse objetivo, tornou-se necessário desenvolver um ambiente mais complexo no robosuite, capaz de simular situações mais semelhantes às encontradas em ambientes domésticos.

A tarefa escolhida para esse experimento foi a tarefa de *pick-and-place*, uma das tarefas mais importantes em manipulação robótica [3]. O objetivo da tarefa é agarrar uma caneca e posicioná-la em uma região específica da mesa, denominada zona de entrega, identificada por uma base vermelha e ilustrada no Apêndice B. A Figura 11 ilustra as *affordances* escolhidas para a caneca, baseadas no trabalho de Ardón et al. [64].

O método *check_success* foi desenvolvido de forma à verificar se a garra está em contato com o objeto, através de *check_grasp*, e se o objeto foi posicionado dentro da área designada para entrega. Dessa forma, a tarefa é considerada bem-sucedida quando o braço robótico deposita a caneca na região de entrega, sem a necessidade de soltá-la.

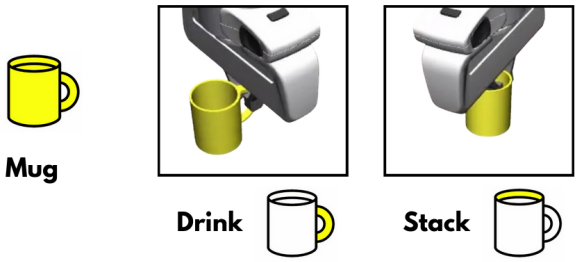


Figura 11: Divisão da caneca em duas áreas: Beber e Empilhar. Fonte: Elaborado pelo Autor.

Diferentemente do experimento anterior, que contava com uma única tarefa, este experimento foi concebido para incluir múltiplas versões de uma tarefa, com complexidade progressiva, apresentadas na Tabela 3.

Na versão mais simples (V_0), há apenas uma caneca posicionada aleatoriamente sobre a mesa em uma área pequena (0.2m x 0.1m), enquanto a região de entrega permanece em uma posição fixa. Na próxima versão (V_1), foi introduzida uma caneca adicional como objeto de distração, e o modelo das canecas passou a ser selecionado aleatoriamente entre diferentes opções.

Tarefa	Quantidade	Área	Modelo	Zona de Entrega	Textura da Mesa
V_0	1	$\odot$	$\mathcal{F}$	$\mathcal{F}$	$\mathcal{F}$
V_1	2	$\odot$	$\mathcal{R}(3)$	$\mathcal{F}$	$\mathcal{F}$
V_2	2	$\odot$	$\mathcal{R}(3)$	$\odot$	$\mathcal{F}$
V_3	2	$\bigodot$	$\mathcal{R}(6)$	$\odot$	$\mathcal{R}(4)$

Tabela 3: Comparação entre as diferentes versões desenvolvidas para a tarefa. Nesta tabela, $\odot$ representa a randomização de posição e orientação em uma área menor, $\bigodot$ em uma área maior, $\mathcal{F}$ simboliza uma característica fixa e $\mathcal{R}$ uma característica randomizada.

A versão V_2 acrescenta complexidade ao variar aleatoriamente a posição da zona de entrega, em uma área pequena (0.3m x 0.1m), assim como sua orientação (0 à $\pi/2$). Por fim, a versão final (V_3), introduz novos modelos de canecas distribuídas em uma área maior (0.3m x 0.2m) e 4 texturas diferentes para a mesa. É importante destacar que, embora o gerador de *dataset* desenvolvido permita a variação de robôs e garras, essa funcionalidade não foi incluída neste teste, uma vez que é incomum treinar um único modelo para diferentes tipos de robôs simultaneamente [60].

O desenvolvimento dessas diferentes versões para a tarefa buscou responder as questões propostas na Seção 4.5. Ao introduzir variações parametrizadas, como a posição, quantidade de objetos e a cor da mesa, é possível avaliar a viabilidade de gerar um conjunto de dados diversificado utilizando o robosuite (**Q1**). Adicionalmente, ao identificar quais variáveis foram modificadas em cada versão e analisar seu impacto na qualidade do conjunto de dados, buscava-se identificar aquelas às quais os modelos demonstram maior sensibilidade (**Q2**).

4.5.3 Experimento 3: Teste de Generalização

O terceiro experimento avaliou se os modelos treinados com dados diversificados são capazes de desempenhar suas funções de maneira eficaz em cenários inéditos (**Q4**) e de manipular um objeto novo, porém semelhante aos utilizados durante o treinamento, pela região de *affordance* escolhida (**Q3**). Para isso, foram utilizados os modelos treinados no Experimento 2, especificamente a Versão 1, que já incorporava três tipos diferentes de canecas.

Dois cenários foram analisados, envolvendo duas canecas distintas. No primeiro cenário, foi utilizada uma caneca semelhante a uma das presentes na tarefa V_1, diferenciando-se apenas no tamanho e na cor. No segundo cenário, a caneca teve um formato inédito não visto durante o treinamento do modelo. As canecas utilizadas, juntamente com outros detalhes das tarefas, podem ser visualizadas no Apêndice B.

5 RESULTADOS

Neste capítulo, são apresentados os resultados dos experimentos descritos na Seção 4.5, acompanhados de uma discussão dos resultados obtidos.

5.1 Experimento 1

Os resultados do primeiro experimento, apresentados na Tabela 4, demonstram a viabilidade de treinar as redes previamente descritas com o *dataset* gerado. Os valores apresentadas na tabela representam a taxa de sucesso total do experimento, ou seja, é considerado sucesso quando o objeto é pego e levantado. Vale destacar que, embora a tarefa não tenha sido concluída em todos os casos devido a falhas no *grasp* ou ao limite de tempo, o modelo foi capaz de identificar a área correta na maioria das tentativas. Grande parte dos erros observados decorreu de falhas na execução do *grasp* da alça, com raros casos em que o braço robótico desviou-se da trajetória correta e nenhum caso em que ele agarrou uma área incorreta do objeto.

Esse desempenho manteve-se consistente mesmo com a rotação aleatória do objeto, tanto na modalidade *low-dim* quanto na modalidade *image*. Isso é especialmente relevante, considerando que o objeto pode rotacionar em até 180º, criando cenários nos quais as alças ocupam posições completamente opostas em diferentes instâncias da tarefa.

Tarefa	**Low Dim (%)**		**Image (%)**
	BC-RNN	**IRIS**	**BC-RNN**
Alça Azul	**87,3 ± 4,1**	80,0 ± 3,3	78,3 ± 3,9
Alça Verde	**92,7 ± 4,7**	89,3 ± 6,1	82,0 ± 1,6

Tabela 4: Resultados comparativos para observações de baixa dimensão e imagens. As taxas de sucesso apresentadas são médias calculadas a partir de 3 execuções, cada uma com 50 tentativas, para cada método.

5.2 Experimento 2

Os resultados do segundo experimento serão apresentados em duas partes: o experimento padrão e o experimento com o *dataset* ampliado.

5.2.1 Experimento padrão

Os resultados do segundo experimento, apresentados na Tabela 5, demonstraram desempenho acima da média no treinamento de diferentes redes utilizando os conjuntos de dados gerados pelo método proposto. Todas as versões da tarefa apresentaram uma taxa de sucesso acima de 45% em pelo menos uma de suas modalidades, com destaque para o desempenho máximo de 82,6% na tarefa de pegar a caneca pela alça, treinada utilizando o modelo BC-RNN em V_0. As curvas de treinamento podem ser encontradas no Apêndice D.

Além disso, os modelos demonstraram a capacidade de aprender corretamente a *affordance* das canecas, agarrando-as nos locais designados em todos os casos em que a tarefa foi realizada com sucesso. No geral, as redes treinadas para a tarefa de empilhar apresentaram desempenho inferior àquelas treinadas para a tarefa de beber. Com base nas demonstrações geradas, pressupõe-se que a principal razão para essa diferença está na maior probabilidade de colisões ao agarrar o objeto pela borda, o que dificulta a execução da etapa de *grasp*. Estudos futuros podem investigar métricas que diferenciem a taxa de acerto na predição do ponto de contato da taxa de acerto de realizar a tarefa.

Tarefa	**Low Dim (%)**		**Image (%)**
	BC-RNN	**IRIS**	**BC-RNN**
Empilhar V_0	61,3 ± 2,5	**64,0 ± 1,6**	62,0 ± 1,6
Beber V_0	**82,6 ± 3,4**	80,7 ± 6,8	60,6 ± 4,1
Empilhar V_1	43,0 ± 2,9	**52,0 ± 3,3**	23,3 ± 1,9
Beber V_1	**56,6 ± 0,9**	48,0 ± 2,8	19,3 ± 3,4
Empilhar V_2	**36,0 ± 0,0**	34,0 ± 2,8	16,7 ± 0,9
Beber V_2	**46,0 ± 4,3**	37,3 ± 1,9	15,3 ± 0,9
Empilhar V_3	36,0 ± 4,3	**36,7 ± 2,5**	16,7 ± 5,3
Beber V_3	**48,0 ± 3,3**	40,7 ± 0,9	12,7 ± 1,6

Tabela 5: Resultados comparativos para observações de baixa dimensão e imagens. As taxas de sucesso apresentadas são médias calculadas a partir de 3 execuções, cada uma com 50 tentativas, para cada método.

A modalidade *low-dim* apresentou desempenho superior à modalidade *image*, principalmente para versões mais complexas da tarefa proposta. Esse resultado pode ser atribuído ao fato de que a *low-dim* utiliza o *ground truth* da posição do objeto, eliminando a necessidade de inferir sua posição e orientação a partir da imagem RGB.

Outro fator que contribui para essa discrepância é que a maioria das alterações realiza-

das entre as diferentes versões das tarefas impactam a modalidade *image* de maneira mais pronunciada. Por exemplo, canecas com formatos similares, mas visualmente distintas, devem ser tratadas da mesma forma em relação à posição e orientação. No entanto, ao trabalhar apenas com a imagem RGB do ambiente, exige-se um grau maior de inferência para identificar corretamente as diferentes características do objeto. Esse aspecto torna-se ainda mais relevante em V_3, na qual as diferentes texturas da mesa aproximam o ambiente de um cenário real, aumentando a complexidade e dificultando a tarefa apenas para o modelo treinado com imagens.

Observa-se, também, que BC-RNN apresentou um desempenho consideravelmente inferior na modalidade *image* a partir de V_1, nos quais o modelo das canecas são randomizados. Isso evidencia que, embora as redes atuais sejam capazes de resolver tarefas complexas de manipulação utilizando *datasets* que exemplifiquem essas tarefas, elas ainda enfrentam desafios consideráveis ao lidar com a transição de um ambiente simulado controlado para cenários mais próximos do mundo real, caracterizados por uma maior diversidade de variáveis no ambiente.

5.2.2 Experimento com *dataset* ampliado

De forma complementar, foi realizado um experimento na versão V_3, utilizando o dobro de demonstrações em comparação com o teste anterior. Este teste foi realizado devido ao fato de V_3 apresentar um número significativamente maior de variações possíveis, com 144 combinações de modelo de caneca e textura de mesa. Dessa forma, garantiu-se pelo menos duas demonstrações para cada combinação de variáveis.

Os resultados, organizados na Tabela 6, demonstram que ampliar o conjunto de dados gerou um impacto positivo em todos os casos, com destaque para um aumento de 11,3 pontos percentuais na taxa de sucesso da tarefa de empilhar na modalidade *image*. Isso demonstra que, para *datasets* mais diversos, o número de amostras geradas é um fator importante, mesmo quando estas são derivadas de apenas 15 amostras originais.

Tarefa	**Low Dim (%)**		**Image (%)**
	BC-RNN	**IRIS**	**BC-RNN**
Empilhar V_3	40,0 $\pm$ 3,3	**43,0 $\pm$ 6,6**	28,0 $\pm$ 4,3
Beber V_3	**51,3 $\pm$ 4,7**	46,0 $\pm$ 3,2	13,3 $\pm$ 4,7

Tabela 6: Resultados comparativos para um *dataset* com o dobro de amostras na Versão 3.

Este tema também foi explorado por Mandlekar et al. [60], que demonstrou que aumentos de até 1000 demonstrações melhoram os resultados das redes, com retornos decrescentes à medida que o número de amostras aumenta. Um estudo mais aprofundado sobre esse assunto, no contexto deste trabalho, está fora do escopo e será abordado em pesquisas futuras.

5.3 Experimento 3

O último experimento teve como objetivo avaliar a capacidade de generalização das redes treinadas, especificamente em V_1, para novos objetos não apresentados durante o treinamento. Assim como nos demais experimentos, foram realizadas três execuções, cada uma com 50 tentativas, cujos resultados estão apresentados na Tabela 7.

Os resultados indicam que as redes são, de fato, capazes de generalizar para novos objetos, apresentando taxas de sucesso satisfatórias e agarrando o objeto pelas regiões corretas. Como esperado, a caneca com formato semelhante às utilizadas no treinamento alcançou uma taxa de sucesso superior em comparação à caneca de formato distinto, especialmente na modalidade *low-dim*. Na modalidade *image*, essa diferença foi menos acentuada, o que pode ser atribuído à maior capacidade da rede de se adaptar a diferentes formatos utilizando informações provenientes da imagem RGB.

Tarefa	**Low Dim (%)**		**Image (%)**
	BC-RNN	**IRIS**	**BC-RNN**
Empilhar (Similar)	38,0 $\pm$ 4,9	**50,7 $\pm$ 4,1**	8,7 $\pm$ 2,5
Beber (Similar)	55,3 $\pm$ 7,5	**59,3 $\pm$ 3,8**	16,7 $\pm$ 0,9
Empilhar (Diferente)	**24,7 $\pm$ 5,0**	19,3 $\pm$ 1,8	19,3 $\pm$ 4,7
Beber (Diferente)	**26,7 $\pm$ 2,5**	23,3 $\pm$ 1,9	14,7 $\pm$ 3,8

Tabela 7: Resultados comparativos para observações de baixa dimensão e imagens. As taxas de sucesso apresentadas são médias calculadas a partir de 3 execuções, cada uma com 50 tentativas, para cada método.

5.4 Discussão

- **Q1.** É viável gerar um *dataset* de *affordance* diversificado, contendo múltiplas variáveis, utilizando o robosuite?

Através dos experimentos detalhados na Seção 4.5, é possível observar que, utilizando o robosuite, é viável criar um *dataset* de *affordance* baseado em tarefas, incorporando variáveis relacionadas aos objetos, câmeras e texturas do ambiente. As ferramentas fornecidas pelo robosuite, combinadas com as funcionalidades desenvolvidas neste trabalho, viabilizaram a construção de um gerador de *datasets* baseado na geração de demonstrações automáticas e no uso do MimicGen.

Os resultados apresentados nas Tabelas 4 e 5 indicam que os *datasets* gerados por essa ferramenta possibilitaram o treinamento de redes, tanto de IL quanto de RL, com desempenho comparável aos *datasets* inteiramente criados por humanos, como os descritos em Mandlekar et al. [26], nas tarefas de *Lift* e *Pick-and-Place*. Além disso, essas redes demonstraram a capacidade de aprender os locais adequados para segurar o objeto, utili-

zando as informações de *affordance* anotadas durante o processo de geração do conjunto de dados.

- **Q2.** Quais variáveis causam maior impacto no desempenho dos modelos?

Os experimentos realizados com diferentes versões da mesma tarefa, Seção 5.2, possibilitaram uma análise do impacto que as variações no ambiente causam na taxa de sucesso da rede. Mandlekar et al. [26] obtêm sucesso considerável ao utilizar BC-RNN e outras redes em tarefas mais complexas do que as realizadas neste experimento. No entanto, os ambientes utilizados no estudo são consideravelmente mais simplificados, restringindo-se à randomização de posições e orientações de alguns objetos, com variações inferiores a 45º em determinados casos. Em contraste, os experimentos conduzidos neste trabalho ampliam o escopo de randomização, incluindo posições e orientações em intervalos maiores, além de adicionar novos modelos de canecas e texturas para a mesa.

De maneira geral, observa-se que, à medida que o ambiente simulado se afasta de um cenário totalmente controlado e incorpora variações comumente encontradas no mundo real, o desempenho das redes sofre uma queda significativa. A introdução de novos modelos de caneca foi o fator que causou a maior queda no desempenho das redes, resultando em uma redução de até 41 pontos percentuais na taxa de sucesso no caso mais extremo. Em contraste, a randomização da posição e orientação da zona de entrega teve um impacto muito menor, com uma queda máxima de 18 pontos percentuais no pior cenário e uma redução de apenas 2 pontos percentuais no melhor caso na modalidade *image*.

De maneira notável, expandir o número de modelos de canecas de 3 para 6 não resultou em um impacto significativo na taxa de sucesso em nenhum dos cenários testados. Esse resultado sugere que, embora a adição de novos modelos afete o desempenho das redes de forma considerável, a introdução de mais canecas com características semelhantes às já presentes no conjunto pode não impactar tão fortemente a performance.

- **Q3.** Os dados gerados permitem o aprendizado implícito de *affordances* por meio de técnicas de aprendizado de máquina?

Os experimentos realizados demonstram que os conjuntos de dados gerados pela ferramenta desenvolvida neste estudo são eficazes em ensinar a manipular um único objeto por diferentes *affordances*. Essa capacidade foi observada em ambas as modalidades analisadas e abrangeu seis modelos de canecas, além de um modelo de pote.

O aprendizado de *affordances* por meio dos métodos utilizados neste estudo evidencia a capacidade das técnicas de aprendizado de máquina de ensinar o modelo de forma implícita. Essa estratégia se diferencia da maioria dos métodos apresentados na literatura (Seção 3.3) por adotar uma abordagem orientada a tarefas, em contraste apenas a inferência de *grasp* sem considerar os passos subsequentes.

- **Q4.** Utilizando os conjuntos de dados gerados, os modelos atuais são capazes de generalizar a execução da tarefa para composições inéditas?

O Experimentou 3 demonstrou que o modelo foi capaz de generalizar para canecas com formatos e cores não vistos durante o treinamento. Além da taxa de sucesso, foi possível observar que os modelos conseguiram transferir o conhecimento de *affordance*, identificando corretamente as regiões adequadas para agarrar os objetos. Em todas as tentativas bem-sucedidas, as novas canecas foram manipuladas corretamente pelos locais apropriados.

Em trabalhos futuros, pretende-se explorar a possibilidade de transferir o *affordance* aprendido para diferentes tipos de objetos. Embora o Experimento 3 tenha utilizado diferentes modelos de caneca, eles representam variações de um mesmo tipo de objeto. Um avanço relevante seria investigar se as redes atuais são capazes de identificar, por exemplo, alças em uma jarra após terem sido treinadas exclusivamente com canecas. Essa abordagem permitiria avaliar a capacidade das redes de generalizar conceitos aprendidos para cenários mais amplos.

6 CONCLUSÃO

Neste trabalho, foi desenvolvida uma ferramenta geradora de *datasets* capaz de ensinar manipulação robótica por diferentes regiões de *affordances* de objetos, explorando a relação entre as propriedades dos objetos e as ações necessárias para manipulá-los de maneira funcional. O *pipeline* desenvolvido parte de um modelo de objeto 3D, anotado com suas *affordances*, e de uma tarefa específica a ser executada. A partir dessas informações, a ferramenta gera demonstrações automaticamente e amplia o conjunto de dados utilizando a ferramenta MimicGen, garantindo um *dataset* diversificado para o treinamento de redes neurais.

Os experimentos realizados evidenciaram que as redes treinadas foram capazes de aprender as *affordances* dos objetos, manipulando-os de forma correta em diferentes cenários, incluindo objetos que não foram observados durante o treinamento. Além disso, os resultados mostraram que a randomização de variáveis, como a introdução de novos modelos de objetos e a variação de texturas, impacta diretamente o desempenho das redes, especialmente em ambientes mais próximos de cenários reais, destacando a dificuldade que as redes atuais tem para lidar com cenários completamente variáveis, e reforçando o *reality gap* como um desafio a ser superado para viabilizar a aplicação prática em robótica.

Este estudo apresentou contribuições relevantes para o campo da manipulação robótica. Ao propor uma solução automatizada para a geração de *datasets* em larga escala, buscou-se reduzir a lacuna de conteúdo, descrita na Seção 2.2, através da utilização de texturas e modelos 3D distribuídos publicamente [78, 80, 84, 85] e das trajetórias geradas pelo MimicGen. Além disso, abordando o problema de *affordances* através de uma abordagem orientada a tarefas, foi demonstrado que o aprendizado de *affordances* pode ser realizado de forma implícita, através do treinamento de redes neurais, e transferido para diferentes cenários e objetos.

Para trabalhos futuros, propõe-se expandir o escopo da ferramenta e dos experimentos para testar a transferência de *affordances* aprendidas entre objetos funcionalmente similares, mas estruturalmente diferentes, como alças de jarras ou copos a partir do treinamento com canecas. Adicionalmente, pretende-se investigar como a qualidade e diversidade dos *datasets* gerados impactam o desempenho de redes treinadas em tarefas mais complexas.

REFERÊNCIAS

[1] Cristian Mejía and Yuya Kajikawa. Bibliometric analysis of social robotics research: Identifying research trends and knowledgebase. *Applied Sciences*, 7:1316, 2017.

[2] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms, 2020. URL https://arxiv.org/abs/1907.03146.

[3] A. Billard and D. Kragic. Trends and challenges in robot manipulation. *Science*, 364, 2019. doi: 10.1126/science.aat8414.

[4] James Jerry Gibson. The theory of affordances. 1977.

[5] Marco Pavone. Cs237b: Principles of robot autonomy ii - lecture 1: Introduction. https://web.stanford.edu/class/cs237b/pdfs/lecture/lecture_1.pdf, 2023. Accessed: 2024-11-03.

[6] Martin L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. 1994.

[7] Oliver Kroemer, Scott Niekum, and George Dimitri Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *ArXiv*, abs/1907.03146, 2019.

[8] Martijn van Otterlo and Marco Wiering. *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_1. URL https://doi.org/10.1007/978-3-642-27645-3_1.

[9] Stanford University. Cs237b: Principles of robot autonomy ii - lecture 4: Reinforcement learning, 2024. URL https://web.stanford.edu/class/cs237b/pdfs/lecture/cs237b_lecture_4.pdf. Accessed: 2024-12-01.

[10] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9, 1998.

[11] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. 2020. doi: 10.48550/arxiv.2006.04779.

[12] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning. *ArXiv*, abs/2005.05951, 2020.

[13] John Schulman, Sergey Levine, P. Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. *ArXiv*, abs/1502.05477, 2015.

[14] C. Gelada and M. G. Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3647–3655, 2019. doi: 10.1609/aaai.v33i01.33013647.

[15] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: a survey of learning methods. *ACM computing surveys*, 50, 2017. ISSN 0360-0300. doi: 10.1145/3054912.

[16] Stanford University. Lecture 10, 11, 12, 13: Imitation learning. `https://web.stanford.edu/class/cs237b/pdfs/lecture/lecture_10111213.pdf`, n.d. CS237B: Principles of Robot Autonomy II.

[17] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Proceedings of (NeurIPS) Neural Information Processing Systems*, pages 305 – 313. Morgan Kaufmann, December 1989.

[18] Michael Bain and Claude Sommut. A framework for behavioural cloning. *Machine Intelligence 15*, 15:103, 1999.

[19] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[20] Emma Brunskill. Cs234: Reinforcement learning - lecture 7: Inverse reinforcement learning. `https://web.stanford.edu/class/cs234/slides/lecture7.pdf`, 2023. Accessed: 2024-11-03.

[21] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

[22] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress, 2020. URL https://arxiv.org/abs/1806.06877.

[23] P. Abbeel, Adam Coates, Morgan Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Neural Information Processing Systems*, 2006.

[24] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35:1289 – 1307, 2016. URL https://api.semanticscholar.org/CorpusID:14275694.

[25] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.

[26] Ajay Mandlekar, Danfei Xu, J. Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Mart'in-Mart'in. What matters in learning from offline human demonstrations for robot manipulation. *ArXiv*, abs/2108.03298, 2021.

[27] Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4414–4420, 2019.

[28] Shuhan Tan, K. Wong, Shenlong Wang, Sivabalan Manivasagam, Mengye Ren, and Raquel Urtasun. Scenegen: Learning to generate realistic traffic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 892–901, 2021.

[29] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4550–4559, 2019.

[30] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2017.

[31] Matthew T. Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

[32] Yuhang Ming, Xingrui Yang, Weihan Wang, Zheng Chen, Jinglun Feng, Yifan Xing, and Guofeng Zhang. Benchmarking neural radiance fields for autonomous robots: An overview. *Engineering Applications of Artificial Intelligence*, 140: 109685, 2025. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2024.109685. URL https://www.sciencedirect.com/science/article/pii/S0952197624018438.

[33] Haoxu Zhang, Parham Mohsenzadeh Kebria, Shady M. K. Mohamed, Samson Yu, and Saeid Nahavandi. A review on robot manipulation methods in human-robot interactions. *ArXiv*, abs/2309.04687, 2023.

[34] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, November 2017. ISSN 1053-5888. doi: 10.1109/msp.2017.2743240. URL http://dx.doi.org/10.1109/MSP.2017.2743240.

[35] Hu Cao, Guang Chen, Zhijun Li, Qian Feng, Jianjie Lin, and Alois Knoll. Efficient grasp detection network with gaussian-based grasp representation for robotic manipulation. *IEEE/ASME Transactions on Mechatronics*, 28(3):1384–1394, 2023. doi: 10.1109/TMECH.2022.3224314.

[36] Yifei Shi, Zixin Tang, Xiangting Cai, Hongjia Zhang, Dewen Hu, and Xin Xu. Symmetrygrasp: Symmetry-aware antipodal grasp detection from single-view rgb-d images. *IEEE Robotics and Automation Letters*, 7:12235–12242, 2022.

[37] Haoshu Fang, Chenxi Wang, Hongjie Fang, Minghao Gou, Jirong Liu, Hengxu Yan, Wenhai Liu, Yichen Xie, and Cewu Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 39:3929–3945, 2022.

[38] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13438–13444, 2021.

[39] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert W. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36:1455 – 1473, 2017.

[40] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2901–2910, 2019. URL https://arxiv.org/abs/1905.10520.

[41] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017.

[42] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[43] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL https://doi.org/10.1145/3054912.

[44] Chelsea Finn, Tianhe Yu, Tianhao Zhang, P. Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *ArXiv*, abs/1709.04905, 2017.

[45] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, P. Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *ArXiv*, abs/1802.01557, 2018.

[46] Haritheja Etukuru, Norihito Naka, Zijin Hu, Seungjae Lee, Julian Mehu, Aaron Edsinger, Chris Paxton, Soumith Chintala, Lerrel Pinto, and Nur Muhammad Mahi Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. *ArXiv*, abs/2409.05865, 2024.

[47] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. *ArXiv*, abs/2202.02005, 2022.

[48] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric A. Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *ArXiv*, abs/2402.10329, 2024.

[49] Sam Toyer, Rohin Shah, Andrew Critch, and Stuart J. Russell. The magical benchmark for robust imitation. *ArXiv*, abs/2011.00401, 2020.

[50] Stephen James, Z. Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5:3019–3026, 2019.

[51] Andy Zeng, Peter R. Florence, Jonathan Tompson, Stefan Welker, Jonathan M. Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, 2020.

[52] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Learning reward functions for robotic manipulation by observing humans. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5006–5012, 2022.

[53] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *ArXiv*, abs/2308.10901, 2023.

[54] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric A. Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *ArXiv*, abs/2303.04137, 2023.

[55] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IEEE International Conference on Robotics and Automation*, 2017.

[56] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, Silvio Savarese, and Li Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

[57] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. *ArXiv*, abs/2212.04498, 2022.

[58] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Haoshu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeff Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishna Parasuram Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama,

Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Manfred Otto Heess, Nikhil J. Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R. Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Ho Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan C. Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open x-embodiment: Robotic learning datasets and rt-x models. *ArXiv*, abs/2310.08864, 2023.

[59] Shuran Song, Andy Zeng, Johnny Lee, and Thomas A. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5:4978–4985, 2019.

[60] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj S. Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning*, 2023.

[61] Huaqing Min, Chang'an Yi, Ronghua Luo, Jin-Hui Zhu, and Sheng Bi. Affordance research in developmental robotics: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 8:237–255, 2016.

[62] Rhys Newbury, Morris Gu, Lachlan Chumbley, Arsalan Mousavian, Clemens Eppner, J. Leitner, Jeannette Bohg, Antonio Morales, Tamim Asfour, Danica Kragic, Dieter Fox, and Akansel Cosgun. Deep learning approaches to grasp synthesis: A review. *IEEE Transactions on Robotics*, 39:3994–4015, 2022.

[63] Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation. *ArXiv*, abs/2401.07487, 2024.

[64] Paola Ardón, Éric Pairet, Ronald P. A. Petrick, Subramanian Ramamoorthy, and Ka-

trin Solveig Lohan. Learning grasp affordance reasoning through semantic relations. *IEEE Robotics and Automation Letters*, 4:4571–4578, 2019.

[65] Weiyu Liu, Angel Andres Daruna, and S. Chernova. Cage: Context-aware grasping engine. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2550–2556, 2019. URL `https://api.semanticscholar.org/CorpusID:202750339`.

[66] Yikun Li, Lambertus Schomaker, and Seyed Hamidreza Mohades Kasaei. Learning to grasp 3d objects using deep residual u-nets. *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 781–787, 2020.

[67] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 01–13, 2023.

[68] Samarth Brahmbhatt, Cusuh Ham, Charles C. Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8701–8711, 2019.

[69] Anh Nguyen, D. Kanoulas, Darwin Gordon Caldwell, and Nikolaos G. Tsagarakis. Detecting object affordances with convolutional neural networks. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2765–2770, 2016.

[70] Thanh-Toan Do, Anh Viet Nguyen, Ian D. Reid, Darwin Gordon Caldwell, and Nikos G. Tsagarakis. Affordancenet: An end-to-end deep learning approach for object affordance detection. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–5, 2017. URL `https://api.semanticscholar.org/CorpusID:3703397`.

[71] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39:202 – 216, 2018.

[72] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, S. Chernova, and Abhinav Kumar Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In *Conference on Robot Learning*, 2020.

[73] Adam Rashid, Satvik Sharma, Chung Min Kim, Justin Kerr, Lawrence Yunliang Chen, Angjoo Kanazawa, and Ken Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *Conference on Robot Learning*, 2023.

[74] Chao Tang, Dehao Huang, Wenqiang Ge, Weiyu Liu, and Hong Zhang. Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping. *IEEE Robotics and Automation Letters*, 8:7551–7558, 2023.

[75] Steve Macenski, Tully Foote, Brian P. Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7, 2022.

[76] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

[77] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Kumar Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *ArXiv*, abs/1712.05474, 2017.

[78] Yuke Zhu, J. Wong, Ajay Mandlekar, and Roberto Mart'in-Mart'in. robosuite: A modular simulation framework and benchmark for robot learning. *ArXiv*, abs/2009.12293, 2020.

[79] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimír Vondru, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *ArXiv*, abs/2106.14405, 2021.

[80] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *ArXiv*, abs/2406.02523, 2024.

[81] Chengshu Li, Fei Xia, Roberto Mart'in-Mart'in, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, Andrey Kurenkov, Karen Liu, Hyowon Gweon, Jiajun Wu, Li Fei-Fei, and Silvio Savarese. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *ArXiv*, abs/2108.03272, 2021.

[82] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[83] Berk Çalli, Arjun Singh, Aaron Walsman, Siddhartha S. Srinivasa, P. Abbeel, and Aaron M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. *2015 International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015.

[84] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2022.

[85] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015.

[86] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3 (1):43–53, 1987. doi: 10.1109/JRA.1987.1087068.

[87] The HDF Group. Hierarchical Data Format, version 5. URL `https://github.com/HDFGroup/hdf5`.

A ALGORITMOS DESENVOLVIDOS

Algorithm 1: *Script* para coletar demonstrações

Input:

- **task_name:** tarefa a ser realizada.
- **robots:** escolha de robô.
- **camera:** câmeras a serem utilizadas.
- **demos_num:** quantidade de demonstrações bem-sucedidas a serem salvas.
- **controller:** controlador a ser utilizado para o braço robótico.

```
1  begin
2      env ← robosuite.make(*args);
3      while number of demos < demos_num do
4          affordance_region ← findAffordanceRegion(env);
5          6D_pose ← calculatePose(affordance_region, env);
6          actions, dones, obs, rewards, states ← collectDemonstration(6D_pose);
7          demo ← actions, dones, obs, rewards, states;
8          demo_success ← checkSuccess(demo);
9          if demo_success is True then
10             add demo to demos;
11         else
12             discard demo;
13     saveDemonstrationsAsHdf5(demos);
```

B TAREFAS DESENVOLVIDAS

Este apêndice apresenta uma descrição de cada tarefa desenvolvida no contexto deste trabalho.

B.1 Pot With Handles

Parâmetro	**Configuração**
Posição do Objeto	(0.3m x 0.3m)
Rotação do Objeto	(90º, 270º)
Textura da Mesa	Cerâmica
Garra	PandaGripper
Robô	Panda

Tabela 8: Configurações do ambiente Pot With Handles.

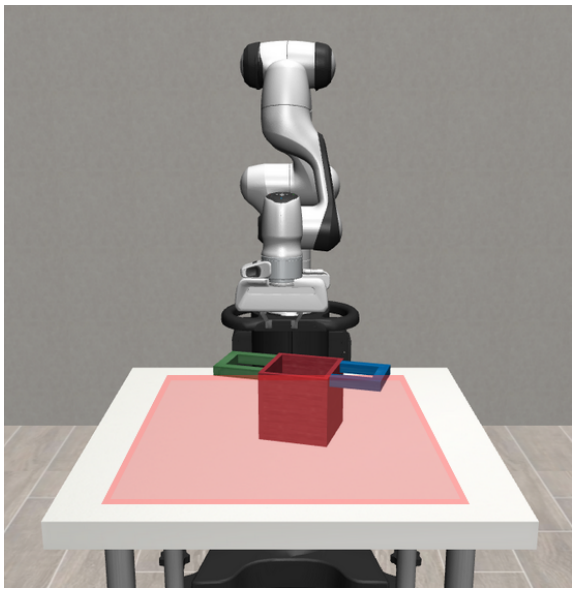
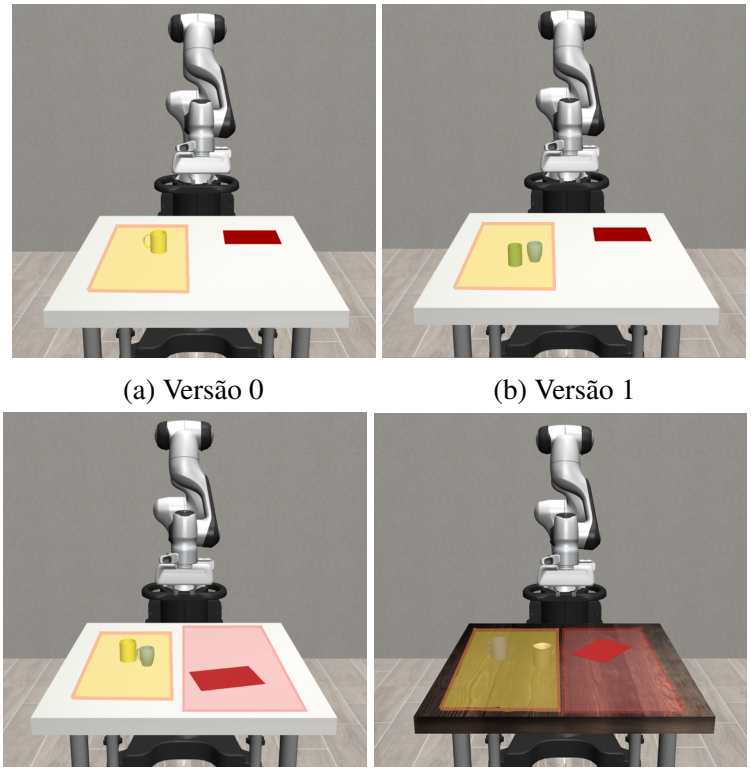


Figura 12: Ambiente da tarefa Pot With Handles. A região vermelha mostra os possíveis locais de posicionamento do objeto de forma aproximada. Fonte: Elaborado pelo Autor.

B.2 Mug Pick And Place

Parâmetro	Versão 0	Versão 1	Versão 2	Versão 3
Modelos	1	3	3	6
Quantidade	1	2	2	2
Área de Posição	(0.2m x 0.1m)	-	-	(0.3m x 0.2m)
Rotação	(0º, 180º)	-	-	-
Área de Entrega	Fixa	-	(0.3m x 0.1m)	(0.3m x 0.1m)
Rotação de Entrega	Fixa	-	(0º, 90º)	(0º, 90º)
Textura da Mesa	Cerâmica	-	-	Cerâmica, Madeira (3)
Garra	PandaGripper	-	-	-
Robô	Panda	-	-	-

Tabela 9: Configurações do ambiente Mug Pick and Place. Fonte: Elaborado pelo Autor.



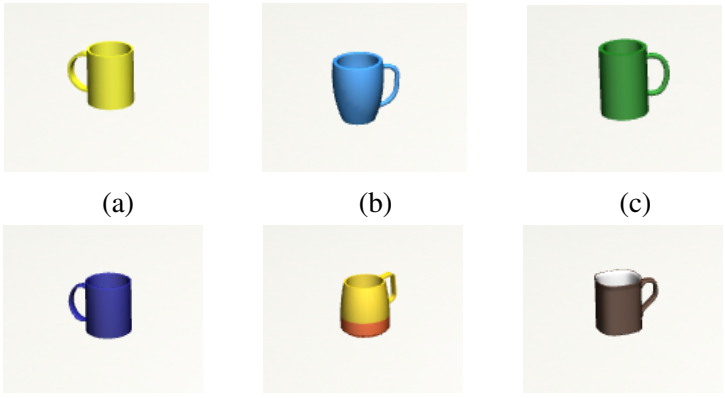
(a) Versão 0 (b) Versão 1

(c) Versão 2 (d) Versão 3

Figura 13: Visualização das diferentes versões do ambiente Mug Pick and Place. Fonte: Elaborado pelo Autor.

(a) Cerâmica (b) *Lightwood* (c) *Darkwood* (d) *Plywood*

Figura 14: Texturas utilizadas para a mesa. Fonte: Zhu et al. [78].



(a) (b) (c)

(d) (e) (f)

Figura 15: Canecas utilizadas. V_0 utiliza a caneca (a), V_1 e V_2 utilizam as canecas (a), (b) e (c), enquanto V_3 utiliza todas. O Experimento 3 utiliza as canecas (d) e (e). Fonte: Elaborado pelo Autor.

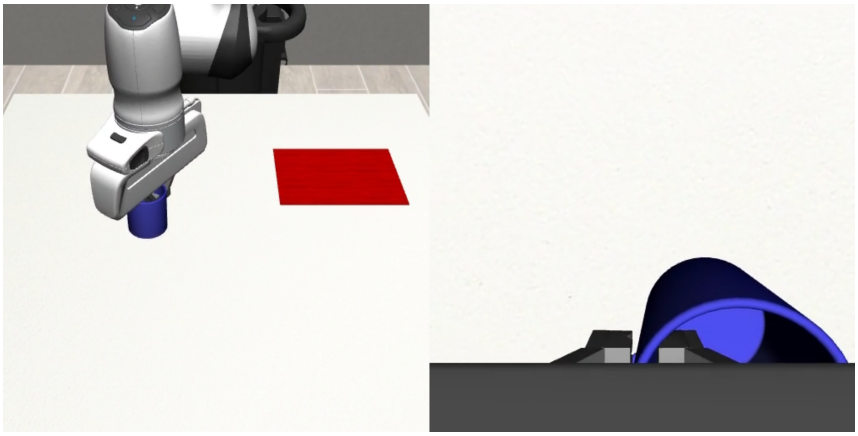


Figura 16: Câmeras utilizadas. Fonte: Elaborada pelo Autor.

C HIPERPARÂMETROS DE TREINAMENTO

Este apêndice apresenta os hiperparâmetros utilizados em cada método, definidos com base nos experimentos descritos no trabalho de Mandlekar et al. [26].

Hiperparâmetro	**Valor**
Epochs ($\mathcal{N}$)	2000
Evaluation ($\mathcal{E}$)	100
Gradient Steps	100
Learning Rate	$1e{-3}$
Actor MLP Dims	[]
RNN Hidden Dim	400
RNN Seq Len	10
GMM Num Modes	5

Tabela 10: Valores dos hiperparâmetros para **BC-RNN - Low Dim**.

Hiperparâmetro	**Valor**
Epochs ($\mathcal{N}$)	600
Evaluation ($\mathcal{E}$)	50
Gradient Steps	500
Learning Rate	$1e{-4}$
Actor MLP Dims	[]
RNN Hidden Dim	1000
RNN Seq Len	10
GMM Num Modes	5
Image Encoder	ResNet-18
SpatialSoftmax (num-KP)	64

Tabela 11: Valores dos hiperparâmetros para **BC-RNN - Image**.

Hiperparâmetro	**Valor**
Epochs ($\mathcal{N}$)	2000
Evaluation ($\mathcal{E}$)	100
Gradient Steps	100
Planner Learning Rate	$1e{-3}$
Planner VAE KL	$5e{-4}$
Planner VAE GMM Prior	True
Planner VAE GMM Latent Dim	16
Planner VAE MLP Dims	[1024, 1024]
Actor Learning Rate	$1e{-3}$
Actor RNN Hidden Dim	400
Actor MLP Dims	[]
Value Learning Rate	$1e{-3}$
Value KL	0.5

Tabela 12: Valores dos hiperparâmetros para **IRIS - Low Dim**.

D CURVAS DE APRENDIZADO

Neste apêndice, são apresentados curvas de aprendizado que mostram a taxa de sucesso em relação à época para o BC-RNN e IRIS nos conjuntos de dados desenvolvidos. Observa-se que, assim como demonstrado por Mandlekar et al. [26], o desempenho de época para época pode variar drasticamente, embora o número de tentativas por avaliação seja alto (50).

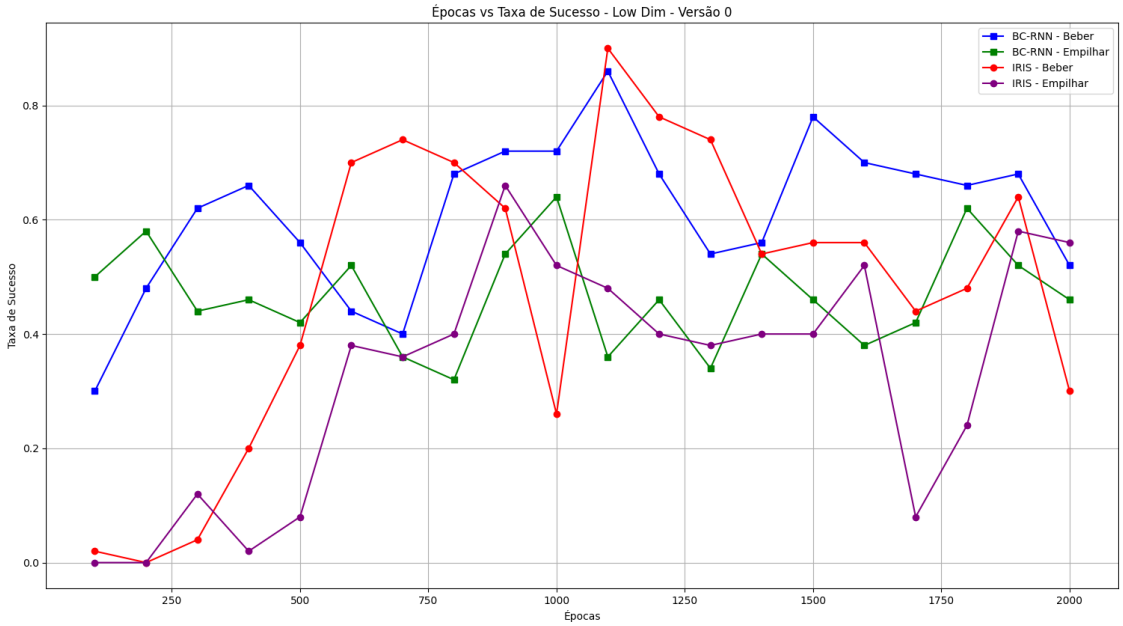


Figura 17: Épocas vs Taxa de Sucesso - **Low Dim - Versão 0**. Fonte: Elaborado pelo Autor

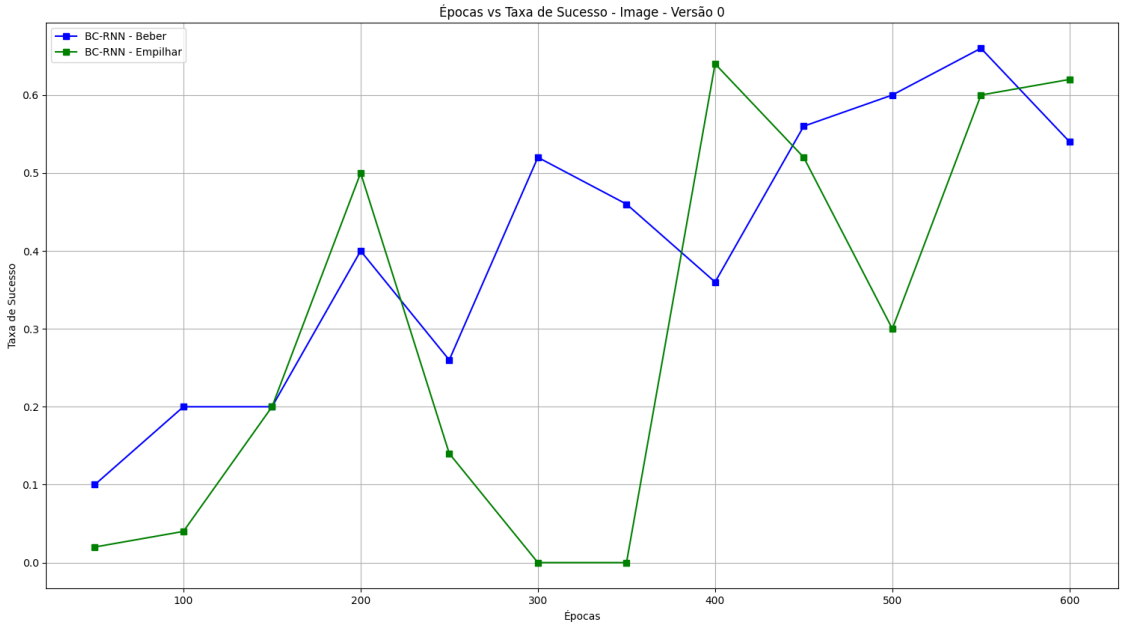


Figura 18: Épocas vs Taxa de Sucesso - **Image - Versão 0**. Fonte: Elaborado pelo Autor

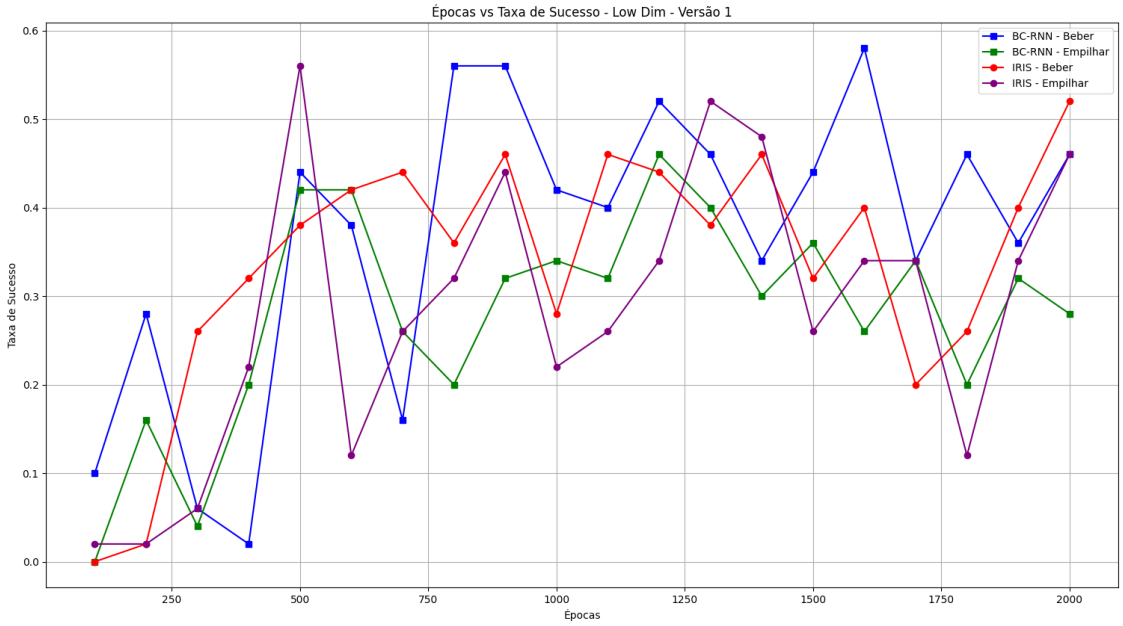


Figura 19: Épocas vs Taxa de Sucesso - **Low Dim - Versão 1**. Fonte: Elaborado pelo Autor

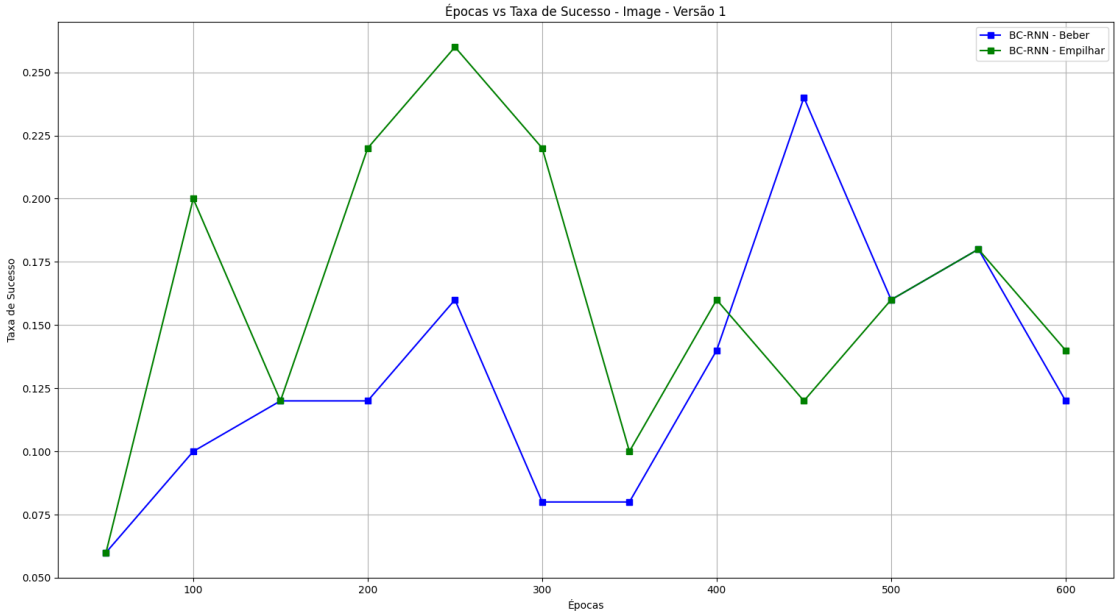


Figura 20: Épocas vs Taxa de Sucesso - **Image - Versão 1**. Fonte: Elaborado pelo Autor

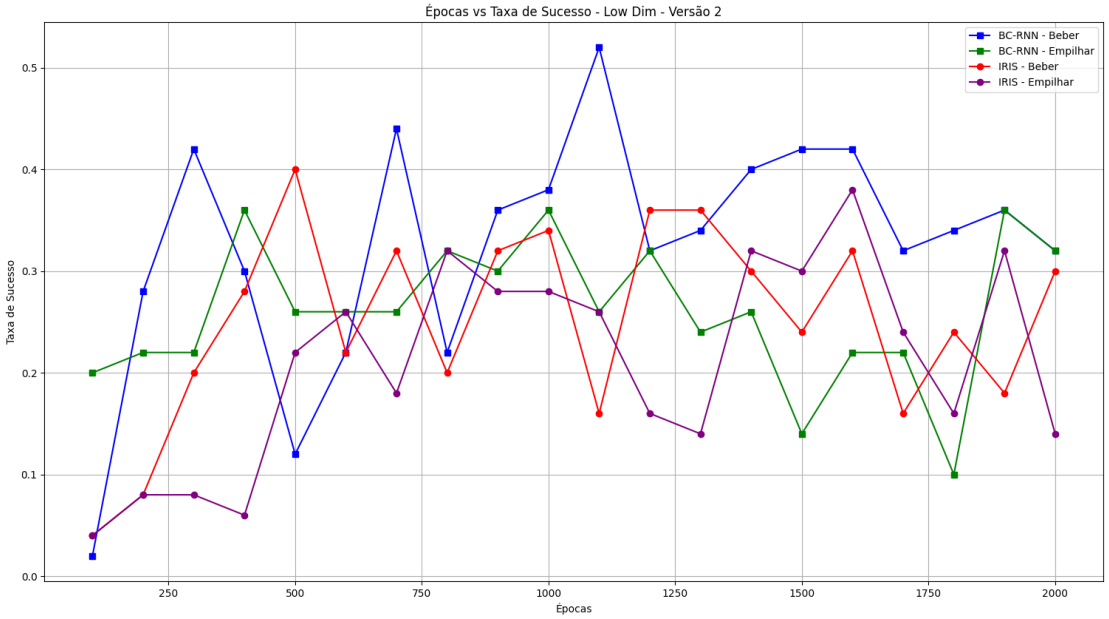


Figura 21: Épocas vs Taxa de Sucesso - **Low Dim - Versão 2**. Fonte: Elaborado pelo Autor

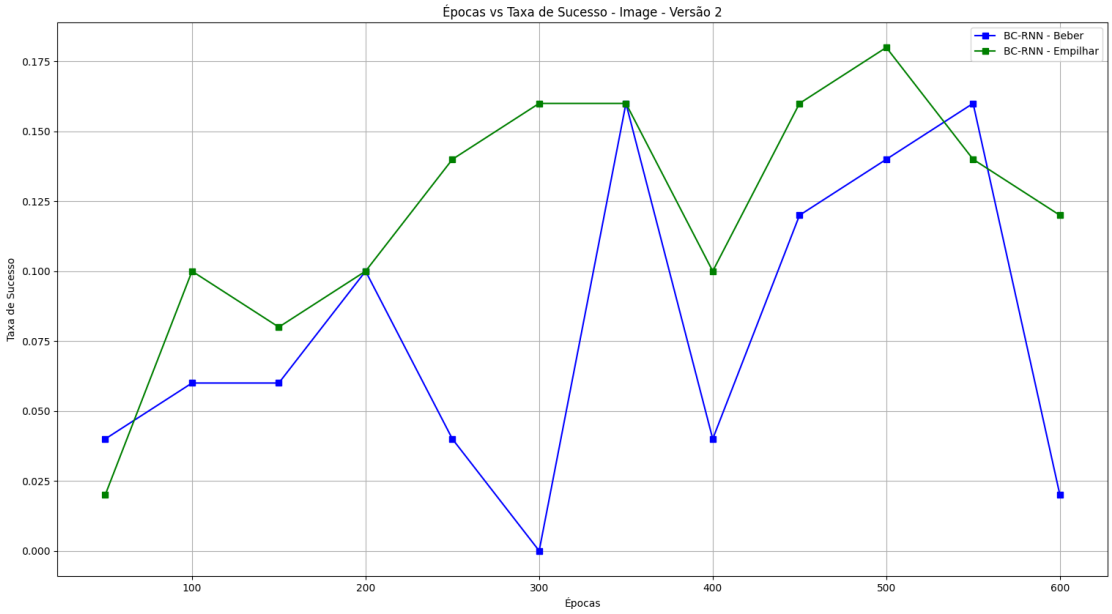


Figura 22: Épocas vs Taxa de Sucesso - **Image - Versão 2**. Fonte: Elaborado pelo Autor

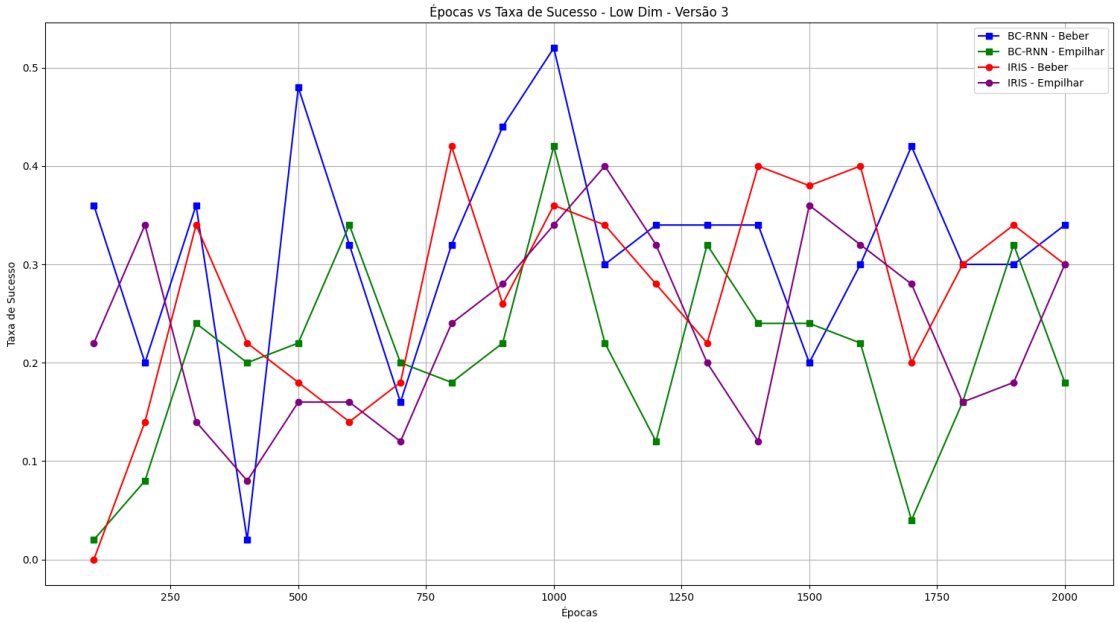


Figura 23: Épocas vs Taxa de Sucesso - **Low Dim - Versão 3**. Fonte: Elaborado pelo Autor

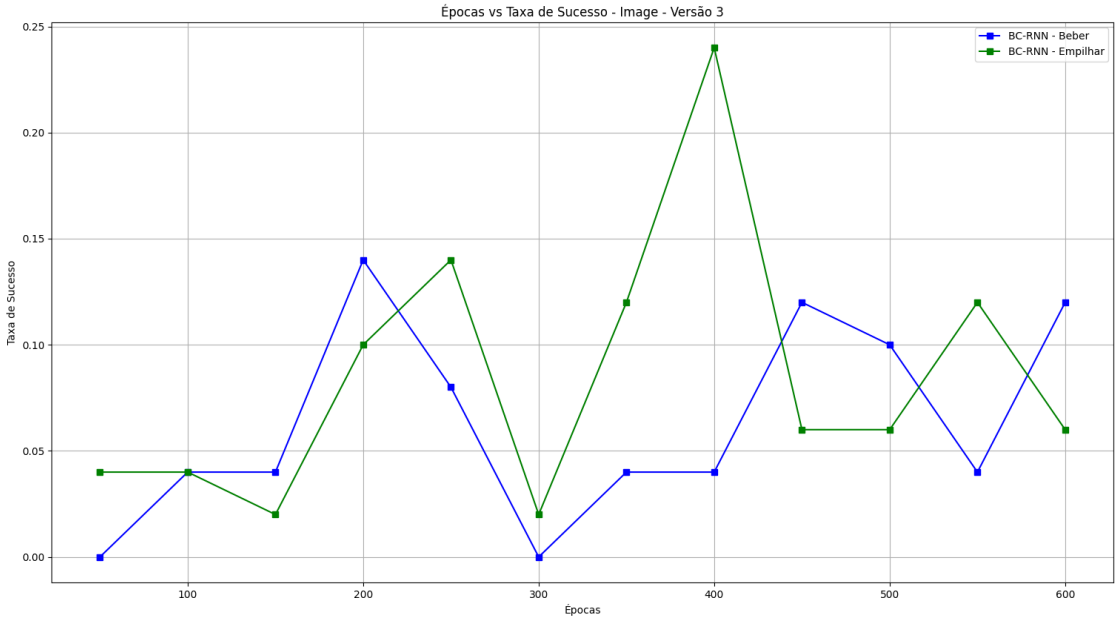


Figura 24: Épocas vs Taxa de Sucesso - **Image - Versão 3**. Fonte: Elaborado pelo Autor