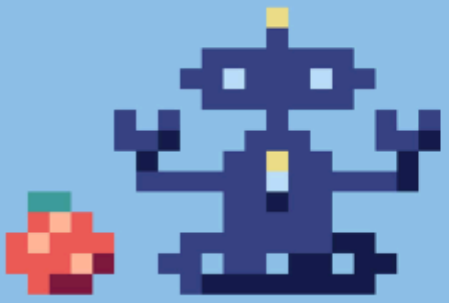
UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Modelos de Visão-Linguagem-Ação para Robótica

Desenvolvimento de Ambiente de Controle via Código

Letícia Lima Mendes

UFG
UNIVERSIDADE
FEDERAL DE GOIÁS



UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

LETÍCIA LIMA MENDES

Modelos de Visão-Linguagem-Ação para Robótica

Desenvolvimento de Ambiente de Controle via Código

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): LETÍCIA LIMA MENDES

Título do trabalho: Modelos de Visão-Linguagem-Ação para Robótica

Desenvolvimento de Ambiente de Controle via Código

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Letícia Lima Mendes**, **Discente**, em 05/02/2026, às 07:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:36, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

LETÍCIA LIMA MENDES

Modelos de Visão-Linguagem-Ação para Robótica

Desenvolvimento de Ambiente de Controle via Código

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

MENDES, LETÍCIA LIMA
Modelos de Visão-Linguagem-Ação para Robótica [manuscrito]: Desenvolvimento de Ambiente de Controle via Código / LETÍCIA LIMA MENDES. - 2025.
55 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Visão Linguagem e Ação. 3. Robótica.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

LETÍCIA LIMA MENDES

Modelos de Visão-Linguagem-Ação para Robótica

Desenvolvimento de Ambiente de Controle via Código

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Profa. Dra. Telma Woerle de Lima Soares
(INF-UFG)

LETÍCIA LIMA MENDES

Modelos de Visão-Linguagem-Ação para Robótica

Desenvolvimento de Ambiente de Controle via Código

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Modelos de Visão Linguagem Ação (VLAs)**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: Inteligência artificial; Visão linguagem e ação; Robótica.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Vision Language Action (VLA) models**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: Artificial intelligence; Vision, language, and action; Robotics.

Goiânia

2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Minha Jornada

Letícia Lima Mendes

Especialista em: Modelos de Visão Linguagem Ação (VLAs)

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

MINHA JORNADA

Nome: Letícia Lima Mendes

Especialidade: Modelos de Visão Linguagem Ação (VLAs)

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

O início da minha trajetória ocorreu na **Semana 1**, quando estabeleci o primeiro contato com os fundamentos que guiaram todo o processo da Residência em IA. Após analisar as áreas propostas pelo CSCI2025 e revisitar os conteúdos que já haviam moldado minha formação, percebi que meu interesse gravitava em torno de sistemas capazes de integrar percepção, linguagem e execução. Os primeiros estudos sobre Modelos de Visão Linguagem e Ação (VLAs) abriram esse caminho, principalmente com a leitura inicial do survey que descrevia as diferentes abordagens de action tokenization. Esse conjunto inicial de leituras e reflexões, detalhado no **Apêndice 1**, serviu como base para eu compreender a diversidade de representações possíveis, bem como para distinguir quais abordagens me pareceriam mais promissoras.

Na **Semana 2**, a compreensão do campo se aprofundou consideravelmente. Após concluir o survey e consolidar meus próprios resumos, identifiquei que a abordagem de

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

geração de tokens de ação por código era a que mais alinhava a intersecção entre linguagem, robótica e geração automatizada de políticas. Essa constatação orientou minhas leituras posteriores, guiando-me para ProgPrompt e Code as Policies, artigos considerados inaugurais dentro da subárea. Ambos os trabalhos, discutidos no **Apêndice 2**, revelaram contrastes relevantes na forma como modelos de linguagem estruturam comandos robóticos e também expuseram desafios técnicos recorrentes, como restrição de contexto, estrutura de loops e definição clara de APIs de ação. Essa etapa foi fundamental para minha decisão de seguir por um caminho que combinasse geração de código com robôs manipuladores.

A consolidação do tema continuou pela **Semana 3**, quando finalizei a leitura dos artigos pioneiros e ampliei meu repertório sobre as diferentes metodologias para transformar comandos naturais em sequências codificadas. A comparação entre abordagens mais restritivas, como ProgPrompt, e outras mais flexíveis, como Code as Policies, tornou evidente que a qualidade final da política gerada depende fortemente da maneira como as APIs são apresentadas ao modelo. Essa discussão, aprofundada no **Apêndice 2**, reforçou minha percepção de que a maior parte dos problemas de execução em VLAs não está na arquitetura do LLM, mas na estrutura de comunicação com o robô. Essa percepção se tornaria essencial para as Semanas seguintes, quando iniciei meus experimentos práticos.

A transição para o trabalho prático ocorreu na **Semana 4**, quando dei início ao processo de trabalhar diretamente com o repositório do RoboCodeX, um artigo e código-base publicados recentemente e que sintetizam muito bem os princípios que eu vinha estudando. No entanto, o ambiente utilizado pelo projeto original requeria ferramentas que não faziam parte do meu repertório inicial: ROS Noetic, Mamba e versões antigas do Gazebo. Ao tentar replicar esse ambiente, enfrentei uma série de incompatibilidades e instabilidades. Por essa razão, decidi adaptar a estrutura para Docker, tornando a execução mais previsível e alinhada às minhas competências. O relato completo dessa reestruturação encontra-se registrado no **Apêndice 4**, incluindo links dos repositórios utilizados.

A **Semana 5** representou uma virada importante na compreensão sistêmica dos VLAs. Elaborei um fluxograma sobre o pipeline completo de geração de código para controle robótico, descrito no **Apêndice 3**, que me ajudou a refletir sobre o que significaria trabalhar o sistema como um todo versus dedicar-me a uma parte específica. Essa visualização

tornou claro que a qualidade da política gerada depende diretamente de como as APIs são organizadas no ROS2 e, principalmente, de sua modularidade. Passei então a formular a hipótese de que muitos erros recorrentes em VLAs não surgem da geração do código em si, mas de como o robô expõe suas ações ao modelo. Essa reflexão guiou as tomadas de decisão das semanas seguintes.

Na **Semana 6**, somando o estudo prático ao teórico, mergulhei no campo mais amplo de geração de código por LLMs, com o objetivo de compreender métricas, metodologias de avaliação e padrões de engenharia de prompt que pudessem fortalecer minha hipótese. Identifiquei temas como análise de vulnerabilidades, uso de persona para controle comportamental e abordagens de fine tuning como PEFT. Embora esse estudo não tenha sido focado especificamente em VLAs, ele serviu como referência para criar paralelos técnicos úteis, registrados no **Apêndice 5**, reforçando novamente a importância da qualidade das interfaces de ação.

Esse acúmulo de conhecimento me preparou para a **Semana 7**, quando avancei significativamente na construção de um ambiente próprio, inspirado pelo RoboCodeX, porém totalmente compatível com ROS2 Humble. O foco desta etapa foi reorganizar as APIs de ação existentes, revisitar o prompt utilizado pelo LLM e transformar a antiga Task Library em um Action Server, permitindo comunicação mais robusta e adequada ao tipo de pipeline exigido por VLAs. As listas completas de APIs, bem como os links dos repositórios, encontram-se no **Apêndice 4**.

Na **Semana 8**, enfrentei o desafio de implementar o braço Panda no simulador Gazebo Ignition e garantir que as posições das articulações pudessem ser controladas diretamente por código gerado por modelos de linguagem. Essa etapa exigiu a criação de pontes entre ROS2 e o simulador, além de um pacote dedicado ao controle das joints. Foi também a primeira Semana em que a integração completa entre modelo de linguagem e robô começou a se tornar funcional. As modificações realizadas, bem como os registros e demonstrações, estão documentados no **Apêndice 4**.

A **Semana 9** foi marcada pelo esforço de adicionar objetos manipuláveis ao ambiente

simulado e de permitir que o próprio LLM fosse capaz de gerar comandos referentes ao spawn desses objetos. Um obstáculo significativo dessa etapa envolveu a divergência entre o mundo simulado no Gazebo e a representação de TFs no RViz, exigindo ajustes manuais. Embora a geração de código ainda estivesse em estágio inicial, essa Semana permitiu organizar melhor o repositório e preparar o terreno para análises mais robustas no futuro. Os vídeos, códigos e links pertinentes estão relacionados no **Apêndice 4**.

Finalmente, a **Semana 10** consolidou os avanços obtidos ao longo de toda a Jornada. Ampliei o conjunto de políticas de ação, refinei o prompt do modelo e finalizei a transição completa da Task Library para um Action Server funcional. Com uma estrutura mais estável, novos experimentos tornaram-se viáveis, aproximando meu ambiente de estudo de um pipeline realista de VLA inspirado no RoboCodeX. A documentação desta etapa também pode ser encontrada no **Apêndice 4**.

Ao observar todo o percurso, percebo que minha Jornada não foi apenas técnica, mas também conceitual. A cada **Semana**, reposicionei minha compreensão sobre o que significa gerar ações para robôs a partir de comandos em linguagem natural. Hoje, vejo que a integração eficiente entre modelos de linguagem e sistemas robóticos depende menos da complexidade dos modelos e mais da clareza, modularidade e consistência com que oferecemos ao modelo as capacidades do robô. Esse insight sintetiza não apenas o que aprendi, mas também o que pretendo continuar desenvolvendo após o encerramento desta Residência.

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 3 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

LETÍCIA LIMA MENDES

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Para a primeira Semana do processo da disciplina Residência em IA foi feito:

- Estudo de interesse sobre as áreas de aplicação segundo CSCI'2025:
  - Automated problem solving;
  - Integration of AI with other technologies;
  - Evaluation of AI tools;
  - Modeling Human Brain processing systems.
- Definição final do tema a ser trabalhado durante o Processo da Disciplina:
  - Visual Language Action (VLA)
- Pesquisa de artigos, trabalhos científicos e surveys sobre o assunto (VLAs)
  - Modelos e Artigos - Inaugurais
  - A Survey on Vision-Language-Action Models: An Action Tokenization Perspective
- Leitura inicial do Survey sobre VLAs
  - Leitura e Resumo Survey VLA

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Finalização da leitura sobre o Survey de VLA;
Definição dos conceitos mais interessantes para mim, para serem tratados com mais detalhes.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

[Resumo: Leitura e Resumo Survey VLA.doc citado no Termo de Aceite de Entrega de 03 de setembro e 10 de setembro]

Resumo do Survey: A Survey on Vision-Language-Action Models: An Action Tokenization Perspective

Modelos de Vision Language Action são sistemas que integram entradas visuais e linguagem natural a fim de gerar ações no mundo real, por meio de um corpo, entrando na área de Embodied AI. A área apresenta diversas implementações diferentes, dado a abrangência do tema, todavia, segundo o survey, todas as implementações possuem um termo em comum "Tokens de Ação" que são uma estrutura capaz de produzir uma ação executável.

O objetivo do artigo é então categorizar modelos de VLA em tipos e analisar as perspectivas trazidas por cada implementação dos Tokens de Ação.

A área é recente, inicia com artigos de 2022, isso dado o crescimento das áreas de Modelos de Linguagem, Visão e, enfim, Visão Linguagem (VLMs) que integram as duas modalidades permitindo compreensão mais integrada. Então, a partir dessa evolução no campo digital, a próxima fronteira da IA, segundo o artigo, é movê-la para o mundo físico, obviamente apresentando desafios significativamente maiores.

Módulos VLA

O artigo define "Módulos VLA" como os blocos de construção de um modelo VLA, que podem ser tanto redes neurais quanto unidades funcionais (como um planejador de movimento). As saídas geradas por esses módulos são os "tokens de ação". A ideia principal é que os tokens de ação nos modelos VLA são uma contraparte generalizada dos tokens de linguagem nos LLMs, eles carregam informações que guiam a ação física.

Taxonomia dos Tokens de Ação: O artigo separa seus capítulos de maneira a trabalhar a taxonomia dos modelos tanto aos tokens de ação:

1. **Descrição em Linguagem** (Capítulo 4)
2. **Código** (Capítulo 5)
3. **Affordance** (pistas visuais sobre como interagir com um objeto) (Capítulo 6)
4. **Trajetória** (Capítulo 7)
5. **Goal** (uma imagem ou vídeo do resultado desejado) (Capítulo 8)

6. **Representação Latente** (vetores abstratos que codificam a ação) (Capítulo 9)
7. **Ação Bruta** (comandos diretos de controle do robô) (Capítulo 10)
8. **Reasoning** (processo de pensamento explícito para gerar outro token de ação) (Capítulo 11)

Descrição em Linguagem

Esse tópico analisa a primeira e mais intuitiva categoria de tokens de ação: o uso de linguagem natural. A principal motivação é aproveitar diretamente as capacidades de compreensão, planejamento e raciocínio dos grandes modelos de linguagem (LLMs) e de visão-linguagem (VLMs). Essa abordagem se alinha à forma como os humanos planejam tarefas complexas.
Níveis de abstração definem como que comandos ou instruções podem influenciar os tokens de ação, “Pegue uma maçã” define um objetivo enquanto “Mova o braço para a direita” implica uma ação.
Os primeiros trabalhos com esse tipo de abordagem foram, **SayCan**, **PaLM-E**, **Hi Robot**, oferecem diversas vantagens como a fácil integração com outros modelos, assim como a facilidade em interpretação humano, por estar no mesmo nível de linguagem e capacidade de desintegrar tasks em tasks menores e executar ao longo prazo. Porém, nem tudo são rosas, esses modelos são mais pesados e aumentam o tempo de latência, e com o aspecto de tempo real pode não ser a melhor escolha.

Código

Apesar de também utilizar de VLMs, ou LLMs, a reprodução de tokens de ação em código, se refere a traduzir um comando como “Segure a xícara” em “move.arm(coordinates) …” diferente do capítulo anterior que transformaria o comando em “Mova o braço para a esquerda e depois abaixe o braço”. Essa metodologia permite uma solução de problema mais robusta e eficiente, porém, se limita muito em termos de generalização, já que depende da capacidade de gerar os códigos para as ações específicas.
Pesquisas como **Code as Policies** e **ProgPrompt** foram pioneiras em transformar a linguagem natural em código robótico funcional.

Affordance

O termo “affordance” se refere diretamente a interação com objetos, assim como uma maçaneta pode indicar abrir ou empurrar, uma porta lisa sem maçaneta indica apenas empurrar. Esse método preza exatamente essa interação com objetos e pode ser implementada de diferentes maneiras:

- Keypoints
- Bounding boxes
- Segmentação
- Mapas de Calor

Essa abordagem acaba sendo muito interessante para manipuladores, principalmente por sua generalização em pegar objetos, todavia, os treinamentos, por consequência dos datasets, são muitas vezes 2D, e isso limita interpretações em um ambiente 3D com mudança temporal.

Trajetória

A trajetória pode ser descrita como uma sequência de passos (estados) ao longo de um período de tempo, parecido com o método gerando “Goals”, é uma previsão da possível trajetória necessária para performar o comando, podendo ser **Point Trajectory**, uma sequência de coordenadas, **Visual Trajectory**, desenho sobre a imagem, ou **Optical Flow**, o movimento de cada pixel na cena.
O interessante dessa metodologia é a capacidade de utilizar vídeos de humanos para treinamento, possuindo então uma ótima generalização, mas novamente sofre com o fato de ser treinado explicitamente em 2D, tendo uma péssima aplicação para cenários 3D.

Goal

Goal ou objetivo, implementa assim como o raciocínio humano, a capacidade de imaginar o resultado final antes de agir, assim, o modelo tem como token de ação uma representação visual do objetivo do commando, isso permite lidar com os problemas dos outros métodos em ambientes 3D, porém sofre com a execução, que, por si, é muito custoso gerar imagens dos estados alvos de alta qualidade, além de ser lento.

Representação Latente

Esse capítulo explora o uso de tokens de ação latentes, vetores compactos e abstratos que não são diretamente legíveis por humanos. A principal motivação é, novamente, superar a escassez de dados. Visando aprender um "vocabulário" de comportamentos básicos de forma não supervisionada a partir de diversas fontes (vídeos de humanos, dados de outros robôs), os modelos podem criar uma representação unificada ao robô. **Genie**, **LAPA** e **GO-1** são exemplos que constroem um espaço de ação latente e o utilizam para treinar políticas de forma mais eficiente.

Reasoning

Reasoning, apesar de parecer semelhante com a primeira abordagem, representação por linguagem, tem como fim ser o mais descritivo possível, e por isso, é capaz de se auto questionar enquanto gera o token de ação, isso permite, que as políticas de mais baixo níveis possam ser corrigidas em caso de algo errado acontecer durante o processo de execução de comando. Nesse caso, o survey chama-se de “meta-token” pois há a geração inicial do reasoning e após a geração de um token de ação por meio de linguagem.

Conclusão

Com base no survey, é possível criar um entendimento básico sobre a área de Visão-Linguagem-Ação e como é abordado diferentes metodologias na mesma área, e mostra um caminho muito rico para desenvolvimento em cima de Embodied AI. A utilização da definição de "Tokens de Ação" ajuda a entender a área de uma maneira mais única, diferente de apenas módulos únicos e de VLMs no geral.

As diferentes metodologias demonstram pontos fortes e positivos mostrando muito espaço para pesquisa dentro da área, é possível que futuramente haja conexões ou até já tenha entre mais de uma abordagem, mostrando o potencial da segmentação feita pelo Survey.

[Tabela: Modelos e Artigos - Inaugurais.xlsx citado no Termo de Aceite de Entrega de 03 de setembro]

Modelos e Artigos Inaugurais

Nome do Modelo	Ano	Instituição(ões) Chave	Contribuição Principal
CLIPort	2021	Univ. of Washington	Fundiu a compreensão semântica (CLIP) com o raciocínio espacial (TransporterNets) numa arquitetura de dois fluxos para manipulação eficiente em dados.
SayCan	2022	Google Research	Ancorou o planeamento de tarefas de LLM na realidade física, combinando a utilidade semântica ("Say") com as "affordances" robóticas ("Can").
Gato	2022	DeepMind	Demonstrou um único agente generalista para controlo multimodal, multitarefa e multi corporificado através de tokenização universal.
RT-1	2022	Google Research	Apresentou o controlo em escala no mundo real com uma política Transformer, provando a eficácia da recolha de dados diversificados e em grande escala do mundo real.
PaLM-E	2023	Google Research	Introduziu "modelos de linguagem corporificados", injetando dados de sensores contínuos diretamente no espaço de incorporação de um LLM, permitindo a transferência de conhecimento positiva.
VIMA	2023	NVIDIA, Caltech, etc.	Propôs uma estrutura de manipulação geral baseada em "instruções multimodais", convertendo diversas tarefas num problema uniforme de modelação de sequências.
OpenVLA	2024	Stanford, Google, etc.	Um modelo VLA de código aberto e alto desempenho, treinado no massivo conjunto de dados Open X-Embodiment, democratizando a investigação no campo.

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 10 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Letícia Lima Mendes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Meu objetivo durante a segunda Semana do processo da Residência em IA foi ler um survey sobre o meu tema Modelos de Visão Linguagem Ação (VLAs).

O Survey que encontrei como meu objetivo, trabalha com a definição de **Tokens de Ação**, que demonstrou o mais compreensível para mim sobre a área. E, o survey, tem como proposta diferenciar as diferentes abordagens dentro de Modelos de VLA:

- Linguagem
- Código
- Affordance
- Goal
- Latência
- Trajetória
- Reasoning
- Raw Action

A imagem a seguir descreve de maneira visual as principais diferenças entre as abordagens:
A Survey on Vision-Language-Action Models An Action Tokenization Perspective.pdf

O link abaixo é um resumo feito por mim sobre o Survey lido:
Leitura e Resumo Survey VLA

Por fim, eu decidi trabalhar com a abordagem de geração de tokens de ação em código.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Ler os 2 principais artigos da área de VLAs com representação em código **Code as Policies** e **ProgPrompt**, definidas pelo survey como pioneiras.

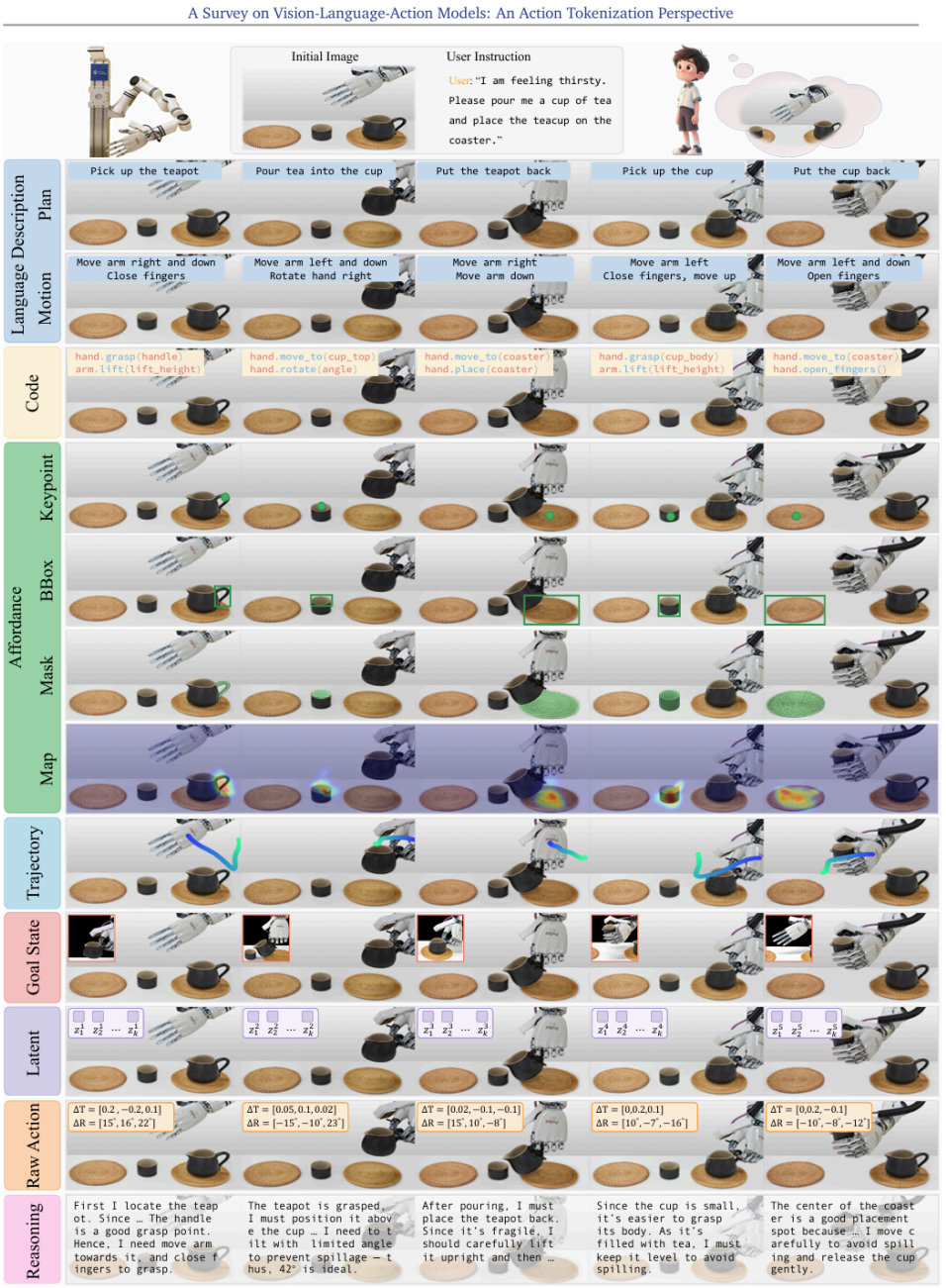
Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

[Imagem: A Survey on Vision-Language-Action Models An Action Tokenization Perspective.pdf citado no Termo de Aceite de Entrega de 10 de setembro]

A Survey on Vision-Language-Action Models An Action Tokenization Perspective



APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 17 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

LETÍCIA LIMA MENDES

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Após decidir trabalhar com o tema de Visual Language Action (VLAs), meu segundo passo foi afunilar a área, assim, descobri sobre a geração de Tokens de Ação com representação em código.

Para entender mais sobre esse tópico, minha atividade da Semana foi ler os artigos pioneiros, segundo o survey, da área de VLAs com geração de código para políticas de ação.

- **ProgPrompt** - Trata Large Language Models (LLMs) como "Task Planners", onde a estrutura do código gerado é quase linear, apenas chamando APIs para funções no corpo. O artigo propõe Prompts mais robustos e explícitos, com informações sobre objetos disponíveis e comandos, assim o LLM é mais restrito, porém mais específico.
- **Code as Policies** - O artigo traz uma geração de código mais flexível e generalista, possuindo loops e lógicas utilizando if/else, while e for. Nessa implementação, o LLM é mais "livre" no sentido de só ser oferecido o necessário e a partir disso a geração de código é mais abrangente.

Resumos dos artigos

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Ler o artigo RoboCodeX para comparar com os artigos ProgPrompt e Code as Policies em relação à evolução ao decorrer dos anos da área.
Entender os datasets utilizados na área e métricas.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

[Resumo: Resumo - Artigos Pioneiros.docs citado no Termo de Aceite de Entrega de 17 de setembro]

Resumo do Artigo: ProgPrompt

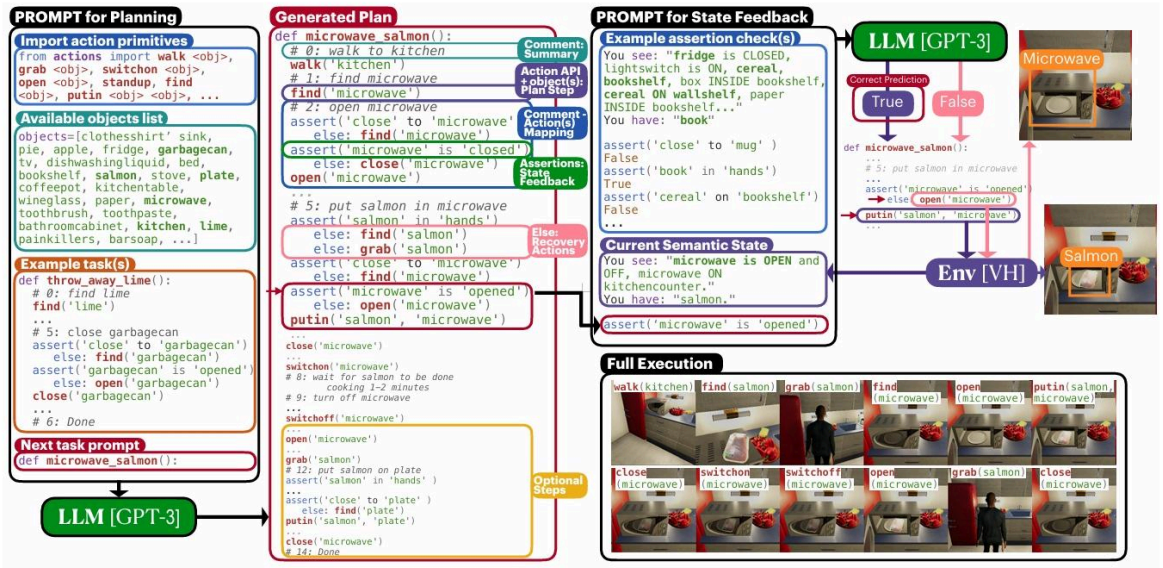


Fig. 2: Our PROGPROMPTs include `import` statement, object list, and example tasks (*PROMPT for Planning*). The *Generated Plan* is for `microwave salmon`. We highlight prompt comments, actions as imported function calls with objects as arguments, and `assertions` with recovery steps. *PROMPT for State Feedback* represents example assertion checks. We further show execution of the program. We illustrate a scenario where an assertion succeeds or fails, and how the generated plan corrects the error before executing the next step. *Full Execution* of the program is shown in bottom-right.

O artigo propõe uma metodologia pela instrução (prompt) do modelo de linguagem (LLM) ser estruturado mais como um programa de computador, contendo:

- **Importações** - Funções (e APIs) disponíveis para geração do código
- **Objetos disponíveis** - Não é dito no artigo se esses objetos são pegos por um modelo de visão ou já são pré definidos pelo ambiente.
- **Função exemplo** - Exemplo de uma tarefa feita para referência.

A principal inovação do artigo é seu mecanismo de feedback e recuperação durante a geração de código, o processo envolve definir “asserts”, que verifica se uma pré-condição é verdadeira, como por exemplo “close to mug” e a partir desse assert e criado uma condição else que corrige caso False.

Para metrificação, o artigo traz três métricas, taxa de sucesso, goal conditions recall e executability, o interessante é que o próprio artigo, taz 10 diferentes tasks e entre essas 10, 5 possuem uma taxa de sucesso correspondente a 0, implicando que não foram concluídas com sucesso, mas nas outras métricas apresentam resultado maior que zero.

Resumo do Artigo: Code as Policies

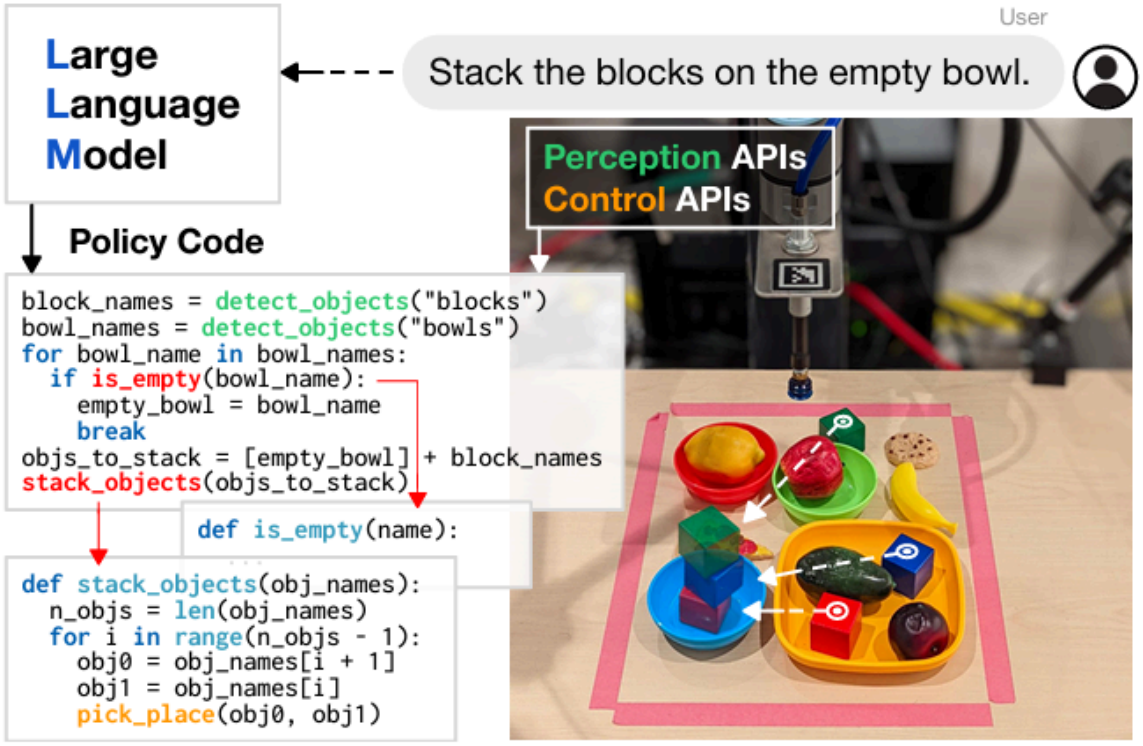


Fig. 1: Given examples (via few-shot prompting), robots can use code-writing large language models (LLMs) to translate natural language commands into robot policy code which process perception outputs, parameterize control primitives, recursively generate code for undefined functions, and generalize to new tasks.

Este artigo apresenta uma abordagem diferente, onde o LLM não é um planejador em alto nível, mas sim um gerador de políticas de controle, ao invés de utilizar de funções já prontas em sequência linear, o próprio código incluindo as funções são gerados pelo LLM que então reagem como percepções e controlam diretamente o os motores.
Em vez de gerar uma lista de etapas discretas (1. Pegue o bloco, 2. Mova para a tigela), o CaP faz com que o LLM escreva um script Python que executa a tarefa do início ao fim. Esse script pode conter lógica complexa, como laços e condicionais, tornando o comportamento do robô muito mais dinâmico e inteligente do que um plano estático.
O artigo propõe prompts “few-shot” com poucos exemplos do que deve ser feito, permitindo que o resultado seja mais generalista para um ambiente não controlado, por exemplo, o LLM pode usufruir de funções como “encontrar_objetos(“objeto”)” e isso permite obter informações sobre o ambiente ao seu redor, além de poder controlar diretamente o robô com APIs como “robô.velocidade(x=0.1, y=02)”.

Por fim, a maior inovação do artigo é trazer um sistema de inovação hierárquico, que permite, por exemplo, ao chamar uma função que não existe, o LLM pode por si, criar essa função.

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 25 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Letícia Lima Mendes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Para minha quarta Semana no processo da Residência em IA, trabalhando com Visual Language Action Models, em específico com a parte de geração de código para generalização de comandos para tradução em políticas de ação do robô.
Li o artigo **RoboCodeX** que trabalha exatamente com a geração de código para tradução em ações, em específicos para braços robóticos. O artigo além de propor uma metodologia generalista para diversos robôs, ainda trás a criação de um dataset sintético que permite essa generalização.

Além da leitura do artigo, fiz uma pesquisa sobre os dataset, métricas e metodologias compartilhadas entre a área, e o que eu encontrei:

- **Benchmarks**: A maior parte dos benchmarks via analisar a capacidade de generalização de comandos.
- **Datasets**: Apesar de diversos datasets, a maioria não se aplica a geração de tokens de ação de código, porém é possível gerar datasets sintéticos a partir deles, assim como RoboCodeX fez.

Documentos relacionados a Semana:
Datasets & Benchmarks
RoboCodeX

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Pretendo preparar um ambiente para simulação no meu computador, se possível instalar e deixar preparado para testar com algum modelo VLA estudado até agora.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

[Resumo: RoboCodeX.docs citado no Termo de Aceite de Entrega de 25 de setembro]

Resumo do Artigo: RoboCodeX: Multimodal Code Generation for Robotic Behavior Synthesis

O RoboCodeX é uma estrutura de geração de código multimodal com uma arquitetura em árvore, projetada para a síntese de comportamento robótico generalizado. Essencialmente, ele atua como uma interface que traduz o entendimento semântico de alto nível do ambiente e das instruções em comportamentos robóticos específicos e adaptados.

Principais Inovações:

- **Geração de Código como Ponte:** O grande diferencial do RoboCodeX é o uso de código como uma "ponte simbólica" entre o conhecimento conceitual dos Modelos de Linguagem Multimodais (MLLMs) e os comportamentos de baixo nível dos robôs. Isso permite que os planos e preferências sejam compartilhados e transferidos entre robôs com morfologias distintas.
- **Estrutura de Pensamento em Árvore (Tree-of-Thought):** O RoboCodeX utiliza uma estrutura de pensamento em árvore para decompor instruções humanas complexas em unidades de manipulação menores e focadas em objetos.
- **Dataset Sintético e Metodologia de Auto-Atualização:** Para aprimorar a capacidade de mapear o entendimento conceitual e perceptual em comandos de controle, foi coletado um dataset de raciocínio multimodal para pré-treinamento. Além disso, uma metodologia iterativa de auto-atualização foi introduzida para o fine tuning supervisionado.

Como o RoboCodeX Funciona:

O processo começa com a captura de observações do ambiente, que consistem em três quadros RGBD de diferentes pontos de vista, para evitar problemas de oclusão e fornecer informações 3D abrangentes.

O RoboCodeX decompõe as instruções humanas de alto nível em múltiplas unidades de manipulação centradas no objeto. Cada uma dessas unidades é então detalhada em componentes chave:

1. **Geração de Propostas de Posição Alvo:** Através de uma análise espacial 3D detalhada dos objetos em seus ambientes.
2. **Previsão das Características Físicas do Objeto:** Utilizando inspiração visual e informações de geometria 3D, como restrições de eixo de translação ou rotação em objetos articulados.

3. **Previsão de Preferências:** Com a compreensão multimodal dos objetivos ambientais e da tarefa, selecionando as posições de agarre e as direções de abordagem ideais.
4. **Geração de Trajetória:** Aproveitando o algoritmo de planejamento para produzir planos de movimento seguros e otimizados, que respeitam as restrições de colisão e físicas.

Resultados e Desempenho:

Experimentos extensivos demonstraram que o RoboCodeX alcança um desempenho de ponta tanto em simuladores quanto com robôs reais em quatro tipos diferentes de tarefas de manipulação e uma tarefa de navegação. O modelo mostrou uma melhoria de 17% na taxa de sucesso em comparação com o GPT-4V.

Em resumo, o RoboCodeX se destaca por sua capacidade de gerar código que não apenas planeja ações, mas também incorpora um profundo entendimento das propriedades físicas dos objetos e das restrições do ambiente. Isso o torna uma ferramenta poderosa para criar comportamentos robóticos mais adaptáveis e generalizáveis.

[Tabela: Datasets & Benchmarks.xlsx citado no Termo de Aceite de Entrega de 25 de setembro]

Benchmark / Environment	Platform Type	Core Focus / Tasks Tested	Key Evaluation	VLA Models Evaluated
CALVIN	Simulation	Long-horizon, language-conditioned manipulation.	Zero-shot transfer to unseen environments (e.g., ABC→D protocol).	GR-1, GR-2, RoboFlamingo, 3D-VLA, UniPi
RLBench	Simulation	Diverse, vision-guided manipulation (100 unique tasks).	Imitation learning, few-shot learning, multi-task learning.	3D-VLA, HybridVLA
Meta-World	Simulation	Multi-task and meta-reinforcement learning (50 tasks).	Few-shot adaptation to new tasks and goal variations.	EmbodiedGPT, QueST, HIRT
LIBERO	Simulation	Lifelong learning; long-horizon manipulation.	Knowledge transfer and retention across a sequence of tasks.	ATM, FLIP, OpenVLA, UniAct
VIMA-Bench	Simulation	General manipulation with multimodal prompts.	Following complex instructions that mix text and visual examples.	CoTDiffusion
Habitat / iTHOR	Simulation	Embodied navigation and instruction following.	Vision-and-language navigation in realistic indoor environments.	LEO, AVDC
Google Robot Evaluations	Real-World	Mobile manipulation in office kitchen environments.	Generalization to unseen tasks and robustness to distractors.	RT-1, RT-2, OpenVLA
Franka-Tabletop / DROID	Real-World	Stationary tabletop manipulation.	Data-efficient fine-tuning and adaptation to new robot setups.	OpenVLA, π_0-FAST

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 8 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

LETÍCIA LIMA MENDES

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Trabalhando com **Visual Language Action Models**, em específico para a parte de utilizar **código como action tokens**.

Desenvolvi esse fluxograma: Fluxograma - VLA (Code Generation) com a intenção de visualizar as etapas do processo de um modelo VLA, e entender se é interessante trabalhar com o sistema como um todo, ou focar em pequenos pedaços.
O que eu entendi foi:

- Trabalhar como um todo, envolve entender as partes e otimizar a conexão entre elas, e não otimizar a qualidade de todas.
- Trabalhar com uma parte envolve otimizar a qualidade dessa como uma só.

O que eu acho interessante:

- Como otimizar a modularidade dos padrões de comunicação (nesse caso com ROS2) para que a geração de código possa ser mais direta e proporcionar menos erros. Ou seja, **como os padrões de comunicação de ROS2 (Tópicos, nós, ações*, …) de cada ação do robô podem ser definidos de forma a proporcionar uma geração de código (por LLMs) mais eficiente**.
- Explicação mais detalhada do que eu pretendo fazer: Projeto - Residência

Além disso, como comentado Semana passada, eu iniciei o processo de passar o ambiente do RoboCodeX para ROS2, por enquanto configurei o docker, launcher e descrição do environment.

Segue aqui o repositório onde estou colocando o código: VLA Lab

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana planejo entender melhor sobre como LLMs geram códigos (métodos, dificuldades, métricas) e como assimilar isso com padrões de comunicação em ROS2.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Como vou participar da Robótica 2025 durante 11/10-20/10 escolhi um planejamento reduzido para

evitar não entregar atividades.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

[Documento: Projeto - Residência.docs citado no Termo de Aceite de Entrega de 08 de outubro]

Nível de Abstração e Especificidade: A Estratégia de Duas Camadas

Uma abordagem seria criar uma hierarquia com (pelo menos) duas camadas de abstração. Isso torna o trabalho do LLM mais simples e o sistema muito mais robusto e reutilizável.

Camada 1: Habilidades de Alto Nível (A API para o LLM)

Esta é a camada que o LLM vai "enxergar" e utilizar diretamente. As ações aqui devem ser:

- **Orientadas à tarefa:** Descrevem "o quê" fazer, não "como" fazer.
- **Humanamente compreensíveis:** Seus nomes e parâmetros devem ser intuitivos.
- **Robustas:** Cada habilidade deve ter lógica interna para lidar com falhas comuns.

Exemplos de Habilidades de Alto Nível:

- **Navegação:** `MapsTo(destination_name)` em vez de `GoToCoordinates(x, y, z, qx, qy, qz, qw)`. O LLM pode pensar em "cozinha", e o robô traduz isso para coordenadas.
- **Manipulação:** `PickUpObject(object_name, location_hint)` em vez de uma sequência de `MoveArm()`, `OpenGripper()`, `MoveArm()`, `CloseGripper()`. A habilidade `PickUpObject` cuida de toda essa orquestração complexa.
- **Visão:** `FindPerson(person_name)` ou `DetectObject(object_class)`. O resultado seria a localização da pessoa/objeto ou a confirmação de sua presença.
- **Comunicação:** `AskUser(question_text)` que internamente usa TTS para fazer a pergunta e STT/RAG para entender a resposta.

> **Regra geral para esta camada:** Define a ação no nível de abstração de um comando que você daria a uma pessoa. "Vá para a cozinha", "Pegue a garrafa de água", "Me avise quando vir o João".

Camada 2: Primitivas de Baixo Nível (A Implementação Interna)

Esta camada contém os componentes que implementam as Habilidades de Alto Nível. São os blocos de construção fundamentais. O LLM geralmente **não** interage com esta camada.

Exemplos de Primitivas de Baixo Nível:

- **Navegação:** Um nó que expõe a capacidade de ir para coordenadas `(x, y, θ)`.

- **Manipulação:** Um nó que controla o braço para ir a uma pose específica (`move_to_pose`), e outro para controlar a garra (`set_gripper(state)`).
- **Visão:** Um nó que publica continuamente as detecções de objetos (`/detected_objects`) e outro que realiza o reconhecimento facial.
- **Comunicação:** Nós individuais para TTS, STT, etc.

Como Definir/Adaptar Essas Ações Usando Padrões de Comunicação ROS2

Tarefas como "navegar" ou "pegar um objeto" não são instantâneas. Elas levam tempo, podem falhar no meio do caminho e é útil saber o progresso. É necessário documentar, comentar, e mastigar o conteúdo para que um LLM seja capaz de entender o processo, como cancelar, e como entender que foi concluído.

Comando do Usuário: "Robô, pegue a maçã na mesa e traga para mim."
LLM "Pensa":

- "O usuário quer a maçã. A maçã está na mesa. Eu estou em outro lugar."
- "Preciso dividir isso em passos usando as habilidades que conheço."
- **Passo 1:** Usar a habilidade `MapsTo`. Meta: `destination_name: "mesa"`.
- **Passo 2:** Após sucesso, usar a habilidade `PickUpObject`. Meta: `object_name: "maçã"`, `location_hint: "mesa"`.
- **Passo 3:** Após sucesso, usar a habilidade `MapsTo`. Meta: `destination_name: "usuário"`.

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 1 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Letícia Lima Mendes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Para essa Semana, venho trabalhado com os passos iniciais de aplicar **Modelos de Visão Linguagem Ação** (VLA).

Decidi trabalhar em cima do repositório RoboCodeX dado que o artigo aborda a mesma área de interesse VLAs por Geração de Código. O repositório utiliza de:

- ROS Noetic
- Mamba (Package Manager)
- Gazebo (Old Version)

Que foi um desafio para mim pois tenho experiência apenas com Conda, ROS2 Humble/Foxy e a nova versão do Gazebo.
Infelizmente não consegui fazer funcionar com o Mamba e adaptei o repositório para utilizar Docker, que é algo que tenho mais experiência e também confio mais que Conda/Mamba para aplicações com ROS/ROS2.
Após isso, tive alguns erros com a câmera então tive que alterar algumas configurações no repositório original, fazendo um Fork apenas para mim (mendesLet/RoboCodeX)

Por fim, coloquei todas minhas modificações no seguinte repositório para ir trabalhando em cima:
VLA Lab

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Entender melhor como o repositório do RoboCodeX funciona, estudar se é possível alterar para ROS2 (apenas as partes que irei utilizar) pois ROS foi descontinuado. Se não for possível tentar iniciar uma arquitetura inspirada pelo RoboCodeX.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 22 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Leticia Lima Mendes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Modelos de **Visão Linguagem e Ação** (VLAs) são modelos que tem por si uma entrada de um comando, seja por linguagem natural ou imagem, e que, transformam essa entrada em uma estrutura de dados que pode ser traduzida em ações em um robô, esses são chamados de **Action Token**.
Tokens de ação podem ser implementados de diversos meios diferentes, entre eles por **Código**, em resumo, um modelo de linguagem (LLM/MLLM) recebe o comando e gera, por código, uma sequência de ações utilizando APIs (ações do robô).
RoboCodeX, um artigo publicado no final de 2024 trás uma metodologia de geração de código para controle de um braço robótico, dado isso, meu atual processo é reproduzir o repositório em Ros2 Humble como metodologia de aprendizado do assunto.
Durante essa Semana, precisei implementar o braço Panda no gazebo ignite de uma maneira que seria possível controlar a posição das articulações. Por isso, precisei apenas criar uma ponte entre o gazebo e ros2, assim, criei também um pacote para controlar as "joints" do braço, com as APIs de ação, e por consequência ser capaz de controlar utilizando o código gerado pelo modelo de linguagem (LLM).
Segue aqui o repositório: https://github.com/mendesLet/vla-lab/tree/main

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Finalizar a implementação do LLM que gera código e controla o braço.
Adicionar mesa e objetos para controle no Gazebo.
Deixar o repositório mais organizado.
Analisar o código gerado (não só para VLAs).

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 5 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

LETÍCIA LIMA MENDES

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Modelos de **Visão Linguagem e Ação** (VLAs) são modelos que tem por si uma entrada de um comando, seja por linguagem natural ou imagem, e que, transformam essa entrada em uma estrutura de dados que pode ser traduzida em ações em um robô, esses são chamados de **Action Token**.
Tokens de ação podem ser implementados de diversos meios diferentes, entre eles por **Código**, em resumo, um modelo de linguagem (LLM/MLLM) recebe o comando e gera, por código, uma sequência de ações utilizando APIs (ações do robô).

Na última Semana, meu objetivo foi:

- Adicionar objetos maleáveis para o braço
- Adicionar geração de comandos com LLM
- Organizar o repositório
- Analisar o código gerado pelo LLM com métricas comuns da área de geração de códigos com LLM

Para adicionar os objetos, percebi que não era uma tarefa muito fácil, pois uma coisa é adicionar esses objetos no gazebo, outra coisa é adicionar as TFs do objeto no RViz.

- **Gazebo**: Simulador de mundo (Onde os objetos, braço, … estão presentes)
- **Rviz**: O que o braço "vê" e compreende do mundo.

A geração de código (utilizando LLM) ainda é muito inicial e não parecida com RoboCodeX, principalmente pois minhas políticas são muito poucas, isso também impactou com a análise do código, pois não tem muito o que analisar.
O repositório está mais organizado, com instruções de como rodar, o que pode rodar, …

Segue aqui os links correspondentes:
Repositório com arquitetura: https://github.com/mendesLet/vla-lab
Repositório só do braço: https://github.com/mendesLet/panda_ros2_gazebo

Vídeo do braço sofrendo para colocar um socket em uma bandeja:
Screencast from 03-11-2025 09:58:58.webm

*Observação sobre aplicação:
Estou fazendo tudo por texto para evitar aumentar a complexidade de lidar com Áudios (Dado que num cenário real dificilmente você escreveria o comando)
Estou dando a oportunidade do LLM “spawn” o objeto para também reduzir a complexidade de aplicar um modelo de visão para pegar as coordenadas do objeto.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Tornar o pacote de Task Library em uma action (permitindo adicionar feedback)
Criar mais políticas de ação (APIs de ação) além das existentes
Melhorar o prompt do LLM (ficar mais parecido com o RoboCodeX)

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 11 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

LETÍCIA LIMA MENDES

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Modelos de **Visão Linguagem e Ação** (VLAs) são modelos que tem por si uma entrada de um comando, seja por linguagem natural ou imagem, e que, transformam essa entrada em uma estrutura de dados que pode ser traduzida em ações em um robô, esses são chamados de **Action Token**.
Tokens de ação podem ser implementados de diversos meios diferentes, entre eles por **Código**, em resumo, um modelo de linguagem (LLM/MLLM) recebe o comando e gera, por código, uma sequência de ações utilizando APIs (ações do robô).

Meu objetivo desde a Semana 7 tem sido criar um ambiente parecido com RoboCodeX, utilizando ferramentas mais atualizadas como por exemplo **ROS2**.
E venho atualizado no seguinte repositório: https://github.com/mendesLet/vla-lab

Os objetivos dessa Semana foram:

- Refatorar a **"Task Library"** para ser um Action Server ao invés de apenas um Server.
- Melhorar o prompt
- Aumentar as APIs de ação disponíveis.

O código refatorado pode ser encontrado aqui: Panda Task Library
O novo prompt pode ser encontrado aqui: NL Command Bridge - Prompt
E as APIs de ação totalizam em:

- **home**: Retorna o braço para a posição inicial.
- **open_gripper**: Abre a garra.
- **close_gripper**: Fecha a garra.
- **go_to_position**: Move o braço para uma posição específica definida por coordenadas e ângulos.
- **pick_object**: Pega um objeto em uma posição e tamanho determinados.
- **place_object**: Posiciona um objeto em uma posição e tamanho determinados.
- **spawn_box**: Cria uma caixa (placeholder).
- **default_box_size**: Retorna o tamanho padrão da caixa (placeholder).
- **wait**: Aguarda por um tempo determinado.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 15 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Letícia Lima Mendes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Modelos de **Visão Linguagem e Ação** (VLAs) são modelos que tem por si uma entrada de um comando, seja por linguagem natural ou imagem, e que, transformam essa entrada em uma estrutura de dados que pode ser traduzida em ações em um robô, esses são chamados de **Action Token**.
Tokens de ação podem ser implementados de diversos meios diferentes, entre eles por **Código**, em resumo, um LLM/MLLM recebe o comando e gera, por código, uma sequência de ações utilizando APIs (ações do robô).
Essa metodologia ataca exatamente nos desafios de generalização da área, todavia acaba sofrendo de outro desafio, a taxa de sucesso. Apesar de sozinhas, as ações rodarem e funcionarem, em conjunto, principalmente em sequência ou dependentes, qualquer erro pode causar consequências na capacidade de completar perfeitamente o comando. O ato do robô se posicionar em uma localização levemente distante de onde deveria pode atrapalhar todos os próximos possíveis comandos.
Muitos estudos tentam tratar esse desafio utilizando métodos tanto na geração do código, como assert em "ProgPrompt", mas mesmo assim ainda enfrentam o desafio, por isso, enquanto estudava e praticava a área, me questionei se às vezes o problema não está na geração do código, mas na maneiras que as APIs de ação são apresentadas para o LLM que gera o código do comando.
Todavia, para qualquer avanço em minha teoria, preciso ter um ambiente controlado e pré estabelecido, que é minha principal atividade atualmente, inspirado pelo artigo do RoboCodeX assim como o repositório.
Além disso, essa Semana, com objetivo de entender melhor sobre geração de código, para então posicionar minha suposição com auxílio da área de geração de código por LLM de maneira generalista e não aplicada apenas a VLAs. Esse estudo não tem objetivo de "afunilar" o conhecimento, apenas acrescentar metodologias, principalmente metrificações e desafios da área, e se correlacionam com a geração de código aplicada a VLAs.
Geração de Código por LLM
Com esse breve estudo, pude compreender metodologias de maneira mais direta a geração de LLM, todavia, alguns tópicos parecem encaixar perfeitamente com minha área de atuação durante o Processo da Residência em IA, como Contagem de Vulnerabilidades em métricas, utilização de Persona, em engenharia de prompt, e PEFT, como metodologia de ajuste fino para a área de VLA.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Continuar implementando ambiente simulado.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

[Documento: Geração de Código por LLM.docs citado no Termo de Aceite de Entrega de 08 de outubro]

Estudo sobre Geração de Código com LLMs

A geração automática de código com Modelos de Linguagem de Grande Escala (LLMs — Large Language Models) tornou-se uma das áreas mais promissoras da inteligência artificial aplicada à engenharia de software [1]. Modelos como Codex, StarCoder, CodeLlama e GPT-Code demonstram capacidade de compreender instruções em linguagem natural e produzir trechos de código coerentes e funcionais. Este estudo aborda os principais métodos de adaptação de modelos (fine tuning), engenharia de prompt e métricas utilizadas para avaliar a qualidade do código gerado

Métodos de fine tuning

O fine tuning consiste em ajustar um LLM pré-treinado para tarefas específicas, como geração de código em uma linguagem ou domínio particular.

Convencional

O finetuning convencional envolve o reajuste completo dos parâmetros do modelo base. Requer grandes quantidades de dados e alto custo computacional, mas proporciona melhor adaptação às especificidades do conjunto de treinamento [2].

Instrução

Neste método, o modelo é ajustado com exemplos do formato entrada → instrução → resposta, ensinando-o a seguir comandos explícitos. Essa técnica melhora a usabilidade e a compreensão de instruções em linguagem natural [3]

PEFT

O PEFT busca eficiência modificando apenas um subconjunto de parâmetros. Métodos como LoRA, Prefix Tuning e Adapter Tuning reduzem custos de memória e preservam o conhecimento geral do modelo [4].

Métodos de engenharia de prompt

A prompt engineering consiste em formular instruções que guiam o comportamento do modelo. A qualidade e clareza do prompt influenciam fortemente o resultado da geração

Instrução direta

Consiste em fornecer um comando explícito. É o método mais simples e direto, adequado para tarefas bem definidas

Few Shot (Exemplos)

Inclui exemplos de entrada e saída no prompt, melhorando a consistência e o controle sobre o formato da resposta [5]

Baseado em Consulta

O modelo consulta informações contextuais, como documentação ou código existente, para melhorar a coerência e o contexto [6]

Refinamento iterativo

Gera um rascunho inicial e solicita melhorias sucessivas, simulando o ciclo humano de revisão de código

Decomposição

Divide uma tarefa complexa em subtarefas menores, aumentando a precisão e reduzindo erros de lógica

Chain of Thought

Instrui o modelo a raciocinar passo a passo antes de gerar o código final [7]

Persona

Define uma identidade para o modelo, ajustando o estilo e o nível de detalhe

Template

Utiliza estruturas pré-definidas com campos fixos e variáveis, padronizando a geração em larga escala

Métricas

pass@k

Mede a probabilidade de que pelo menos uma das k amostras geradas passe em todos os testes. É usada em benchmarks como HumanEval [8]

CodeBLEU

Extensão da métrica BLEU tradicional, que incorpora aspectos sintáticos e semânticos específicos do código

Complexidade Ciclomática

Avalia a complexidade lógica do código gerado, contabilizando o número de caminhos independentes no fluxo de controle [9]

Uso de memória

Mede a eficiência do código em termos de consumo de memória durante a execução.

Contagem de Vulnerabilidades

Quantifica vulnerabilidades detectadas, garantindo segurança e robustez no código gerado [10]

Referências

[1] Vaswani et al., “Attention is All You Need,” 2017.
[2] Brown et al., “Language Models are Few-Shot Learners,” 2020.
[3] Wei et al., “Finetuned Language Models are Instruction Followers,” 2022.
[4] Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” 2021.
[5] Schick & Schütze, “Few-Shot Text Generation with Pattern-Exploiting Training,” 2021.
[6] Chen et al., “Evaluating Large Language Models Trained on Code,” 2021.
[7] Kojima et al., “Large Language Models are Zero-Shot Reasoners,” 2022.
[8] Chen et al., “Evaluating Code Generation Models with HumanEval,” 2021.
[9] McCabe, “A Complexity Measure,” 1976.
[10] Pearce et al., “Examining the Security Risks of LLM-Generated Code,” 2023.

* Eu não li essas referências, foram colocadas para caso eu tenha interesse em aprofundar no assunto poder direcionar após minha residência ou durante para complementar.