

انجام وظایف پیچیده توسط ربات با شکستن آنها به مهارت‌های ساده و طرح‌ریزی با استفاده از مدل‌های زبانی بزرگ

ولی درهمی*، آلا سلطانزاده و احسان خادمی
دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران

تاریخ پذیرش: ۱۴۰۴/۰۷/۲۷

تاریخ دریافت: ۱۴۰۴/۰۳/۱۸

نوع مقاله: علمی پژوهشی

چکیده. در این مقاله، رویکردی برای شکستن وظایف پیچیده رباتیک به مهارت‌های پایه‌ای و ترکیب آن‌ها برای انجام وظایف پیچیده ارائه شده است. در به‌کارگیری ربات‌ها برای انجام وظایف پیچیده چالش‌هایی همچون زمان بر بودن یادگیری، عدم تعمیم‌پذیری، و عدم تعامل انسانی وجود دارد. در این مقاله در قدم اول ایده شکستن وظایف پیچیده به مهارت‌های ساده مطرح می‌شود تا یک ربات با یادگیری مهارت‌های پایه بتواند تنوع زیادی از مهارت‌های پیچیده را به انجام برساند. بدین طریق زمان یادگیری کمتر شده و تعمیم‌پذیری نسبتاً خوبی هم حاصل می‌شود. برای طرح‌ریزی وظایف، از مدل‌های زبانی بزرگ مانند چت جی‌پی‌تی و دیپ‌سیک استفاده شده است که توانایی آن‌ها در ارائه‌ی برنامه‌های عملی برای ترکیب مهارت‌های پایه بررسی می‌شود. دودسته دستورالعمل یکی با ورودی‌های واضح و دیگری با مقادیر فازی به سیستم داده می‌شود. نتایج نشان می‌دهد که مدل‌های زبانی بزرگ ذکر شده نه‌تنها در مواجهه با ورودی‌های واضح بلکه حتی در مواجهه با ورودی‌های فازی توانایی مناسبی در درک دستور وظیفه (Task) داده شده، و برنامه‌ریزی اقدامات برای ربات دارند. لذا با کمک آنها یک سیستم رباتی تعاملی را می‌توان ایجاد نمود.

2010 Mathematics Subject Classification.

E-mails: vderhami@yazd.ac.ir, alasantanzade356@gmail.com and ehsankhademi169@gmail.com

عبارات و کلمات کلیدی: ربات، شکستن وظایف، طرح‌ریزی، فازی، مدل‌های زبانی بزرگ.

۱- سرآغاز

بازوهای رباتیکی امروزه در طیف وسیعی از کاربردها از جمله تولید صنعتی، پزشکی، کشاورزی و خدمات عمومی مورداستفاده قرار می‌گیرند. این ربات‌ها قابلیت انجام وظایفی مانند مونتاژ قطعات [۱]، جراحی‌های دقیق [۲]، برداشت محصولات کشاورزی [۳] و ارائه خدمات در محیط‌های عمومی [۴] را دارند. با وجود پیشرفت‌های چشمگیر در طراحی مکانیکی و ساختاری این سامانه‌ها، اجرای وظایف پیچیده و چندمرحله‌ای همچنان به‌عنوان یکی از چالش‌های اساسی در حوزه رباتیک مطرح است [۵].

در حوزه رباتیک، به‌ویژه در کاربردهایی که شامل دستکاری با بازوی ربات می‌شود، چالش اصلی اغلب در تجزیه‌ی یک وظیفه‌ی پیچیده به عملیات بنیادی است که سخت‌افزار بتواند به طور قابل‌اعتمادی آن‌ها را اجرا کند. با تقسیم‌بندی رفتارهای پیچیده به سلسله‌مراتبی از مهارت‌های ساده و قابل‌ترکیب، مهندسین می‌توانند سیستم‌های انعطاف‌پذیرتر و مقاوم‌تری طراحی نمایند. مدل‌های زبانی بزرگ با شناسایی ساختارهای نهفته در وظایف و پیشنهاد توالی‌های بهینه مهارتی به این فرایند کمک می‌کنند. یکی از دلایل اصلی این چالش، نیاز به برنامه‌ریزی و کنترل انعطاف‌پذیر و تعمیم‌پذیر است که بتواند در شرایط مختلف و در مواجهه با وظایف جدید، عملکرد مناسبی ارائه دهد. از سوی دیگر، تعامل با انسان به‌عنوان یکی از ارکان اساسی در بسیاری از کاربردهای مدرن رباتیکی مطرح است؛ چرا که کاربران نهایی در بسیاری از موارد تخصص فنی برای تعامل از طریق کد یا رابط‌های پیچیده ندارند [۶].

در این راستا، توسعه‌ی روش‌هایی که به ربات‌ها امکان درک و اجرای دستورات زبان طبیعی را بدهد، موردتوجه بسیاری قرار گرفته است. به همین دلیل، در سال‌های اخیر استفاده از مدل‌های یادگیری عمیق و پردازش زبان طبیعی^۱ به‌منظور ترجمه دستورات زبانی به حرکات رباتیکی، توجه بسیاری از پژوهشگران را به خود جلب کرده است [۷].

یکی از رویکردهای مطرح‌شده برای حل این مسئله، استفاده از شبکه‌های عصبی انتها - انتها^۲ است که ورودی آن‌ها دستورات زبانی و خروجی آن‌ها مستقیماً فرمان‌های حرکتی ربات است. این روش‌ها در برخی وظایف خاص موفق بوده‌اند، اما دارای چالش‌هایی نظیر نیاز به حجم زیادی از داده‌های آموزشی، عدم تعمیم‌پذیری به وظایف جدید و زمان‌بر بودن فرآیند آموزش هستند [۸]. علاوه بر این، این مدل‌ها معمولاً برای یک مجموعه مشخص از وظایف

^۱Natural Language Processing: NLP

^۲End to End

طراحی شده‌اند و انعطاف‌پذیری کافی برای یادگیری وظایف جدید بدون آموزش مجدد ندارند [۹].

در مقابل، مدل‌های زبانی بزرگ^۱ متنوع آموزش دیده‌اند، توانایی تحلیل مفهومی بالایی دارند و می‌توانند به‌عنوان واسطی برای تفسیر دستورات زبانی به کار روند [۱۰]. با این حال، چالش اصلی این مدل‌ها نداشتن دانش ضمنی درباره کنترل‌های حرکتی و پویایی‌های فیزیکی ربات‌هاست [۱۱، ۱۲]. در حالی که، پژوهش‌ها نشان داده‌اند که ادغام مدل‌های زبانی بزرگ در مرحله‌ی برنامه‌ریزی، می‌تواند به شکل چشمگیری دقت و تطبیق‌پذیری سیستم را ارتقا دهد. برنامه‌ریزی نقش حیاتی در اجرای وظایف رباتی دارد و ادغام مدل‌های زبانی بزرگ در این فاز، سطح جدیدی از پیچیدگی و دقت را معرفی کرده است. برای رباتی که دارای بازو است، برنامه‌ریزی مؤثر به معنای تعیین صحیح توالی حرکات و کاهش خطاهای احتمالی در طول اجرا است. این مدل‌ها با ایجاد برنامه‌های آگاه در این زمینه، اهداف انتزاعی را به گام‌های ملموس ترجمه می‌کنند که مستقیماً به دستورات کنترلی تبدیل می‌شوند. این تغییر الگو، امکانات جدیدی در برنامه‌ریزی و کنترل رباتیک به وجود آورده است.

در شرایطی که بازوهای رباتیکی باید بتوانند هم وظایف متنوعی را انجام دهند و هم با انسان‌ها تعامل مؤثر برقرار کنند، سه راهکار عمده برای افزایش تعمیم‌پذیری قابل تصور است: (۱) جمع‌آوری داده‌های آموزشی بیشتر، (۲) بهبود الگوریتم‌های یادگیری و کنترل و (۳) ارتقاء سخت‌افزارهای رباتیکی [۱۳].

این مقاله رویکردی نوین مبتنی بر سیستم‌های فازی ارائه می‌دهد که با بهره‌گیری از مدل‌های زبانی بزرگ، دستورات زبانی را تحلیل کرده و به‌جای تولید مستقیم حرکات، از یک کتابخانه مهارت‌های پایه استفاده می‌کند. برخلاف پژوهش‌های پیشین [۱۱، ۱۲] که یا تنها به تفسیر زبان طبیعی پرداخته‌اند یا [۱۸] نیازمند داده‌های آموزشی حجیم بوده‌اند، روش پیشنهادی با ترکیب سلسله‌مراتبی مهارت‌های پایه و توان برنامه‌ریزی مدل‌های زبانی، هم تعمیم‌پذیری بیشتری دارد و هم هزینه محاسباتی کمتری ایجاد می‌کند.

در این چارچوب، مدل‌هایی مانند ChatGPT و DeepSeek برای تحلیل و طرح‌ریزی وظایف ربات در سناریوهای مختلف به کار می‌روند. سیستم توانایی مدیریت ورودی‌های صریح و فازی را داشته و نتایج نشان می‌دهد تقسیم وظایف به مهارت‌های پایه و ترکیب فازی آن‌ها، برنامه‌ریزی انعطاف‌پذیر و تعامل مؤثر با دستورات مبهم انسانی را ممکن می‌سازد. افزون

¹Large Language Models: LLMs

بر این، روش پیشنهادی بدون نیاز به بازآموزی گسترده به محیط‌ها و وظایف جدید تعمیم می‌یابد و با یادگیری از تعاملات انسانی، دقت و پاسخگویی را در شرایط واقعی بهبود می‌دهد. به‌طورکلی سهم علمی این مقاله به شرح زیر است: ۱- استفاده از منطق فازی در تفسیر دستورات زبانی ۲- تولید خروجی طرح‌ریزی بر اساس مهارت‌های پایه که امکان تعمیم‌پذیری برای انجام وظایف پیچیده با ترکیب مهارت‌های پایه را می‌دهد. ۳- مقایسه عملکرد جی‌پی‌تی و دیپ‌سیک برای موضوع مذکور.

ساختار مقاله به شرح زیر است. در بخش دوم مفاهیم پایه در خصوص بازوی رباتیکی و مدل‌های زبانی بزرگ بیان می‌شود. در بخش سوم رویکرد پیشنهادی برای تقسیم وظایف پیچیده ربات به مهارت‌های پایه‌ای و ترکیب آن‌ها جهت بهبود عملکرد بررسی می‌شود. در بخش چهارم کارایی روش با استفاده از مدل‌های زبانی بزرگ در محیطی شبیه‌سازی شده مانند کافه‌تريا ارزیابی می‌شود. در پایان در بخش پنجم نتیجه‌گیری و پیشنهادها آمده است.

۲- مفاهیم پایه

۱-۲ معرفی یک بازوی رباتیکی. بازوی رباتیکی یکی از متداول‌ترین سامانه‌های مکانیکی در حوزه رباتیک است که به‌منظور انجام وظایفی چون جابه‌جایی، مونتاژ، و تعامل با اشیاء طراحی می‌شود. یک بازو شامل چندین مفصل و لینک است که به‌صورت زنجیره‌وار به یکدیگر متصل شده‌اند. هر مفصل، با استفاده از عملگرهایی نظیر موتورهای الکتریکی، و سرووها حرکت لینک متصل به خود را ممکن می‌سازد. بخش انتهایی این بازو، موسوم به نقطه اثر^۱ وظیفه تعامل مستقیم با محیط را بر عهده دارد. کنترل یک بازو به معنی ارسال فرمان مناسب به محرک مفصل‌ها است [۱۴].

درجه آزادی نقش مهمی در تعیین قابلیت‌های حرکتی بازو ایفا می‌کند. به‌طور معمول، بازوی رباتیکی مورد استفاده در این پژوهش دارای شش درجه آزادی است که امکان انجام حرکات پیچیده در فضای سه‌بعدی را فراهم می‌سازد.

در محیط‌های شبیه‌سازی نظیر پای‌بولت^۲، بازوی رباتیک به‌صورت یک مدل دیجیتال تعریف می‌شود که اغلب با استفاده از قالب استاندارد یوآر‌دی‌اف^۳ توصیف می‌شود. این مدل شامل اطلاعات مربوط به ساختار فیزیکی، محدودیت‌های حرکتی، جرم، و محل قرارگیری اجزاست.

^۱End Effector

^۲PyBullet

^۳Unified Robot Description Format: URDF

مدل‌های زبانی بزرگ مانند جی‌پی‌تی چهار^۱ و دیپ‌سیک^۲ طی سال‌های اخیر به نقطه عطفی در پردازش زبان طبیعی و درک متنی رسیده‌اند. این مدل‌ها با استفاده از ساختار ترانسفورماتورها [۱۵] که در سال ۲۰۱۷ معرفی شد ایجاد شده‌اند. آنها قادرند متون پیچیده را تحلیل، تولید، و حتی تفسیر کنند. توانایی این مدل‌ها در درک مفاهیم انتزاعی، تشخیص نیت کاربر و تبدیل زبان طبیعی به دستورالعمل‌های دقیق، آن‌ها را به ابزارهایی قدرتمند در تعامل انسان-ربات بدل کرده است.

۲-۲ معرفی مدل‌های زبانی بزرگ (LLM). در حوزه‌ی رباتیک، از مدل‌های زبانی بزرگ برای تجزیه دستورات سطح بالا به گام‌های اجرایی استفاده می‌شود. به‌عنوان مثال، دستور زبانی مانند «لیوان را از روی میز بردار و داخل سینی بگذار» می‌تواند توسط مدل زبانی تحلیل شده و به یک توالی از مهارت‌های پایه‌ای مانند حرکت بازو به موقعیت لیوان، گرفتن، بلندکردن، انتقال و رهاکردن ترجمه شود. مدل‌هایی نظیر چت‌جی‌پی‌تی و دیپ‌سیک-کدر^۳، با توانایی‌های کدنویسی و استدلال زبانی، در ترکیب با سیستم‌های کنترل رباتیکی، ابزارهای مناسبی برای پیاده‌سازی این فرایندها محسوب می‌شوند.

مطالعاتی همچون [۱۶]، [۱۷] و [۱۸] نشان داده‌اند که ادغام این مدل‌ها با محیط‌های شبیه‌سازی و سامانه‌های فیزیکی می‌تواند موجب افزایش خودکارآمدی، قابلیت تعمیم، و سادگی در برنامه‌ریزی وظایف پیچیده شود. با این حال، چالش‌هایی نظیر عدم درک ضمنی از فیزیک، محدودیت در کنترل‌های سطح پایین و ناپایداری در سناریوهای تعاملی همچنان پابرجاست که موضوع این پژوهش نیز در راستای رفع بخشی از همین محدودیت‌ها است.

۳- روش ارائه شده

ایده اصلی مقاله این است که برای اجرای وظایف پیچیده در ربات‌ها، می‌توان این وظایف را به مهارت‌های پایه‌ای ساده‌تر تقسیم کرد. سپس، با ترکیب این مهارت‌ها، ربات قادر خواهد بود وظایف پیچیده را انجام دهد. این روش باعث یادگیری سریع‌تر و تعمیم‌پذیری، به‌اشتراک‌گذاری مهارت‌ها بین وظایف مختلف و افزایش کارایی برنامه‌ریزی ربات‌ها می‌شود.

۳-۱ مهارت‌های پایه. در این بخش، با بررسی تعداد زیادی از وظایف ربات، لیستی از مهارت‌های پایه‌ای معرفی می‌شود. این مهارت‌ها به‌گونه‌ای تعریف شده‌اند که حداقل هم‌پوشانی

^۱GPT₄ Model OpenAI

^۲DeepSeek

^۳DeepSeek Coder

با یکدیگر داشته باشند و با ترکیب آن‌ها، بتوان وظایف پیچیده را به انجام رساند. به‌عنوان مثال، رباتی که در یک کافه‌تیریا به کار گرفته می‌شود، دارای دودسته مهارت‌های پایه‌ای است: مهارت‌های شناختی: این دسته از مهارت‌ها مرتبط با پردازش ادراک، تصویر، و تصمیم‌گیری هستند و شامل توابعی است که به ربات کمک می‌کنند تا محیط پیرامون خود را تحلیل کند و اطلاعات موردنیاز را برای اجرای مناسب وظایف به دست آورد. این مهارت‌ها باید به‌صورت پیش‌فرض در ابتدای تمامی پرامپت‌ها فراخوانی شوند تا زمینه‌ی ادراکی و تحلیلی لازم برای سایر مهارت‌ها فراهم گردد.

`CaptureImage()`: گرفتن یک فریم تصویری از محیط با استفاده از دوربین آرچی‌بی^۱ نصب‌شده بر روی ربات.

`ProcessImage(image)`: انجام پیش‌پردازش بر روی تصویر (مانند فیلترگذاری، نرمال‌سازی یا اصلاح نور) به‌منظور آماده‌سازی برای تحلیل.

`DetectObjects(image)`: شناسایی اشیاء موجود در تصویر از طریق مدل‌های تشخیص شیء^۲.

`GetObjectCoordinates(object_name)`: استخراج مختصات دقیق شیء شناسایی‌شده در فضای کاری ربات.

`CalculateGraspPoint(object_coords)`: محاسبه بهترین نقطه‌ی گرفتن برای بازوی ربات بر اساس موقعیت و شکل شیء.

`SendSignal(machine_name)`: ارسال سیگنال کنترلی به دستگاه‌های محیطی (مثلاً قهوه‌ساز یا مخلوط‌کن) برای فعال‌سازی یا آغاز عملیات.

مهارت‌های حرکتی: این مهارت‌ها مربوط به فیزیک حرکت، جابه‌جایی، تعامل فیزیکی با اشیاء و اجرای عملیات توسط نقطه اثر بازوی ربات هستند.

`MoveToCoordinates(coords)`: جابه‌جایی نقطه اثر بازو به مختصات موردنظر.

`OpenGrip()`: بازکردن گیره^۲ به‌منظور گرفتن یا رهاکردن اشیاء.

`CloseGrip()`: بستن گیره برای گرفتن مطمئن شیء هدف.

`RotateGrip()`: چرخش گیره برای کج کردن لیوان گرفته شده به‌منظور ریختن ماده موجود

^۱Red-Green-Blue: RGB

^۲Gripper

در آن (مایع، شکر، پودر قهوه، و غیره).

۲-۳ تولید فرمان کنترلی برای ربات با استفاده از یک مدل زبانی بزرگ. پس از تعریف مجموعه‌ای از مهارت‌های پایه، اینک باید ساختاری را فراهم کرد که بتواند وظیفه پیچیده مورد درخواست یک انسان را به زبان طبیعی و به صورت تعاملی از او بگیرد و آنگاه با اجرای یک سلسله مهارت‌های پایه به صورت پی‌درپی آن وظیفه پیچیده را به انجام برساند. پیشنهاد ما در اینجا، استفاده از مدل‌های زبانی بزرگ مانند جی‌پی‌تی است. هدف این است که وقتی یک دستور به زبان طبیعی (مانند «یک قهوه با شیر درست کن») به سیستم داده می‌شود، مدل بتواند آن را به مجموعه‌ای از مهارت‌های پایه‌ای که در بخش قبل معرفی شد تبدیل کرده و در نهایت، این مهارت‌ها به صورت کد قابل اجرا بر روی ربات پیاده‌سازی شوند. این فرآیند شامل سه مرحله‌ی اصلی است:

مرحله اول: طراحی پرامپت^۱

در این مرحله یک پرامپت مناسب برای مدل زبانی طراحی می‌شود. این پرامپت باید اطلاعات زیر را در خود داشته باشد: توصیف کلی از نقش ربات (مثلاً ربات باریستای یک کافه‌تیریا) – لیست دقیق مهارت‌های پایه‌ی در دسترس (با نام و توضیح مختصر هر مهارت) – فرمت خروجی موردانتظار (مثلاً لیستی از توابع با ترتیب اجرایی) – وظیفه موردنظر به صورت زبان طبیعی

هدف از این طراحی، کاهش ابهام برای مدل و افزایش دقت در تولید دنباله‌ی صحیحی از مهارت‌هاست. مثال ساده‌ای از یک پرامپت ممکن به شکل زیر است:

You are a robot barista. You can perform the following basic skills:

- CaptureImage(): Take a photo using the RGB camera.
- DetectObjects(image): Detect objects in the scene.
- GetObjectCoordinates(object_name): Get the position of the object.
- MoveToCoordinates(coords): Move the robotic arm to the target coordinates.
- OpenGrip(): Open the gripper.
- CloseGrip(): Close the gripper.

¹Prompt Engineering

- RotateGrip(): Rotate the gripper.

Your task is to serve a cup of coffee with milk.

Generate a list of functions to perform this task in order.

در نسخه‌های پیشرفته‌تر پرامپت، می‌توان از نمونه‌های مثال محور استفاده کرد، تا مدل با دیدن چند مثال مشابه عملکرد دقیق‌تری ارائه دهد. در این طراحی، پرامپت پایه همواره ثابت است و تنها دستور جدید کاربر به آن اضافه می‌شود. این ساختار به مدل کمک می‌کند تا با زمینه‌ای مشخص و قابل‌پیش‌بینی، عملکرد بهتری ارائه دهد.

مرحله دوم: دریافت خروجی مدل زبانی و تولید دنباله مهارت‌ها

پس از ارسال پرامپت به مدل زبانی، خروجی دریافت‌شده معمولاً به‌صورت یک توالی از توابع تعریف‌شده خواهد بود که ترتیب اجرای مهارت‌ها را نشان می‌دهد. به‌عنوان مثال:

```
CaptureImage(), DetectObjects(image), GetObjectCoordinates("coffee_machine"), Move-  
ToCoordinates(coords), SendSignal("coffee_machine"), GetObjectCoordinates("cup"),  
MoveToCoordinates(coords), OpenGrip(), CloseGrip(),  
GetObjectCoordinates("coffee_machine_dispenser"), MoveToCoordinates(coords),  
OpenGrip(), CloseGrip(), RotateGrip()
```

این دنباله مشخص می‌کند که ربات باید ابتدا تصویر محیط را بگیرد، محل دستگاه قهوه‌ساز را تشخیص دهد، به آن نقطه حرکت کند، سیگنال شروع به کار بدهد، سپس به سراغ لیوان رفته، آن را بگیرد، سپس مکان دکمه ریختن قهوه‌ساز (دسته ریختن قهوه‌ساز) را تشخیص دهد، به آن نقطه حرکت کند، دسته یا دکمه را بگیرد (با باز و بسته کردن گریپر) سپس دسته را بچرخاند تا قهوه در لیوان ریخته شود.

مرحله سوم: اجرای مهارت‌ها و تعامل با محیط شبیه‌سازی‌شده

در این مرحله، هر تابع (مهارت) با کد معادل پایتون در محیط شبیه‌سازی اجرا می‌شود. توابع از پیش در سیستم پیاده‌سازی شده‌اند و با داده‌های خروجی مرحله دوم (مختصات، نام اشیاء، زمان اجرا، و غیره) پارامتردهی می‌شوند. به این ترتیب، ربات به‌صورت خودکار فرمان‌ها را اجرا کرده و وظیفه تعیین‌شده را در محیط انجام می‌دهد. در این مرحله، هر مهارت به‌صورت یک تابع پایتون در محیط شبیه‌سازی اجرا می‌شود. این توابع از پیش پیاده‌سازی شده‌اند و با

داده‌هایی مانند مختصات، نام اشیا، یا پارامترهای فازی مقداردهی می‌شوند. بدین ترتیب، ربات به صورت خودکار فرمان‌ها را اجرا کرده و وظیفه موردنظر را انجام می‌دهد.

۴- آزمایش‌ها

در اینجا از دو مدل‌های زبانی بزرگ چت‌جی‌پی‌تی و دیپ‌سیک برای برنامه‌ریزی و اجرای وظایف ربات استفاده شده است. این مدل‌ها با استفاده از پرامپت‌های مناسب، وظایف پیچیده را به مهارت‌های پایه‌ای تقسیم کرده و سپس این مهارت‌ها را ترکیب می‌کنند تا ربات بتواند وظیفه موردنظر را به درستی انجام دهد. در اینجا، تمرکز بر روی اجرای وظایف در یک محیط کافه‌تريا است.

مثال ۱: در ادامه، مثالی از یک درخواست کاربر برای تهیه قهوه با شکر آورده شده است. دستور کاربر در پرامپت از پیش‌طراحی شده گنجانده شده و به مدل‌های زبانی ارائه می‌شود. دو درخواست کاربر: لطفاً یک فنجان قهوه با یک قاشق شکر برایم درست کن. لطفاً یک فنجان قهوه کمی شیرین برایم درست کن.

خروجی چت‌جی‌پی‌تی پاسخ به دستور اول: ("یک فنجان قهوه با یک قاشق شکر")

گام ۱: دریافت تصویر از محیط

```
image = CaptureImage()
```

گام ۲: پیش‌پردازش تصویر

```
processed_image = ProcessImage(image)
```

گام ۳: شناسایی اشیا در محیط

```
objects = DetectObjects(processed_image)
```

گام ۴: دریافت مختصات دستگاه قهوه‌ساز و ظرف شکر

```
coffee_machine_coords = GetObjectCoordinates("coffee_machine")
sugar_container_coords = GetObjectCoordinates("sugar_container")
```

گام ۵: محاسبه بهترین نقطه گرفتن برای ظرف شکر

```
sugar_grasp_point = CalculateGraspPoint(sugar_container_coords)
```

گام ۶: ارسال سیگنال به قهوه‌ساز برای آماده‌سازی قهوه

```
SendSignal("coffee_machine")
```

انجام وظایف پیچیده توسط ربات با شکستن به مهارت‌های ساده _____ ۳۰

گام ۷: ارسال سیگنال برای برداشتن دقیقاً یک قاشق شکر

```
SendSignal("sugar_unit")
```

خروجی چت‌جی‌پی‌تی پاسخ به دستور دوم: ("یک فنجان قهوه کمی شیرین")

گام ۱: دریافت تصویر از محیط

```
image = CaptureImage()
```

گام ۲: پیش‌پردازش تصویر

```
processed_image = ProcessImage(image)
```

گام ۳: شناسایی اشیاء در محیط

```
objects = DetectObjects(processed_image)
```

گام ۴: دریافت مختصات دستگاه قهوه‌ساز و ظرف شکر

```
coffee_machine_coords = GetObjectCoordinates("coffee_machine")
```

```
sugar_container_coords = GetObjectCoordinates("sugar_container")
```

گام ۵: محاسبه بهترین نقطه گرفتن برای ظرف شکر

```
sugar_grasp_point = CalculateGraspPoint(sugar_container_coords)
```

گام ۶: ارسال سیگنال به قهوه‌ساز برای آماده‌سازی قهوه

```
SendSignal("coffee_machine")
```

گام ۷: تفسیر مقدار فازی «کمی»

```
def InterpretSweetness(level):
```

```
    fuzzy_map = {"unsweetened": 0, "slightly": 5.0,
```

```
                "medium": 1, "sweet": 5.1, "very sweet": 2}
```

```
    return fuzzy_map.get(level, 1)
```

```
sugar_amount = InterpretSweetness("slightly")
```

گام ۸: ارسال سیگنال متناسب با مقدار فازی شکر

```
if sugar_amount == :5.0
```

```
    SendSignal("sugar_unit_half")
```

```
elif sugar_amount == 1:
```

```
    SendSignal("sugar_unit")
```

```

elif sugar_amount == :5.1
    SendSignal("sugar_unit")
    SendSignal("sugar_unit_half")
elif sugar_amount == 2:
    SendSignal("sugar_unit")
    SendSignal("sugar_unit")

```

در این مثال، بدون نیاز به تعریف توابع عضویت یا منطق فازی پیچیده، مدل تنها با استفاده از نگاشت ساده‌ی زبانی، قادر است مقدار فازی «کمی» را به عدد معادل تبدیل کرده و رفتار مناسب را انتخاب کند. این روش ساده ولی مؤثر نشان می‌دهد که مدل‌های زبانی بزرگ می‌توانند منطق فازی سطح بالا را نیز بدون پیاده‌سازی ریاضیاتی آن، به‌درستی تفسیر کنند.

پرامپت ارائه‌شده به دیپ‌سیک بابت دو درخواست کاربر: لطفاً یک فنجان قهوه با یک قاشق شکر برایم درست کن. لطفاً یک فنجان قهوه کمی شیرین برایم درست کن. خروجی دیپ‌سیک پاسخ به دستور اول: (“یک فنجان قهوه با یک قاشق شکر”)

گام ۱: ثبت تصویر از محیط کافه‌تریای ربات

```
image = CaptureImage()
```

گام ۲: بهبود کیفیت تصویر برای تحلیل بهتر

```
processed_image = ProcessImage(image)
```

گام ۳: شناسایی اشیا (مانند قهوه‌ساز، قاشق، شکر)

```
objects = DetectObjects(processed_image)
```

گام ۴: یافتن مختصات قهوه‌ساز

```
coffee_machine_coords = GetObjectCoordinates("coffee_machine")
```

گام ۵: تعیین نقطه فعال‌سازی قهوه‌ساز

```
grasp_point = CalculateGraspPoint(coffee_machine_coords)
```

گام ۶: روشن‌کردن دستگاه برای تهیه قهوه

```
SendSignal("coffee_machine")
```

گام ۷: یافتن مختصات ظرف شکر

```
sugar_coords = GetObjectCoordinates("sugar_container")
```

گام ۸: محاسبه‌ی نقطه‌ی برداشت قاشق شکر

```
grasp_point = CalculateGraspPoint(sugar_coords)
```

گام ۹: افزودن یک قاشق شکر دقیق به قهوه

```
SendSignal("sugar_unit")
```

خروجی دیپ‌سیک پاسخ به دستور دوم: ("یک فنجان قهوه کمی شیرین")

۱. ثبت تصویر از محیط

```
image = CaptureImage()
```

۲. پیش‌پردازش تصویر

```
processed_image = ProcessImage(image)
```

۳. تشخیص اشیا (شکر، قهوه‌ساز)

```
objects = DetectObjects(processed_image)
```

۴. یافتن موقعیت ظرف شکر

```
sugar_coords = GetObjectCoordinates("suger")
```

۵. منطق فازی: تعریف متغیر فازی "میزان شیرینی" با مقادیر کیفی (کم، متوسط، زیاد)

نگاشت "کمی شیرین" به بازه‌ای ۵.۰ تا ۵.۱ قاشق شکر با استفاده از توابع عضویت مثلثی

انتخاب مقدار میانی (قاشق) به‌عنوان خروجی قطعی‌سازی شده

۶. محاسبه‌ی نقطه‌ی برداشت شکر

```
grasp_point = CalculateGraspPoint(sugar_coords)
```

۷. فعال‌سازی قهوه‌ساز

```
SendSignal("coffee maker")
```

۸. افزودن یک قاشق شکر به قهوه مطابق خروجی

```
SendSignal("sugar_Spoon_Device")
```

مثال‌های دیگر در پیوست آمده است.

۱-۴ مقایسه خروجی ها. در مقایسه بین خروجی های چت جی پی تی و دیپ سیک، مشاهده می شود که چت جی پی تی رویکردی ساده و مستقیم برای اجرای وظایف دارد که مناسب برای کاربردهای سریع و کم پیچیدگی است، اما انعطاف پذیری کمتری برای گسترش و افزودن وظایف جدید دارد. در مقابل، دیپ سیک با ساختار شی گرای منظم و تفکیک واضح مراحل، زمینه را برای توسعه پذیری و مدیریت بهتر وظایف فراهم می کند. از نظر منطق فازی نیز چت جی پی تی با استفاده از نگاشت ساده زبانی، مفاهیم مبهمی مانند « کمی شیرین » را بدون تعریف توابع عضویت رسمی تفسیر می کند، در حالی که دیپ سیک با بهره گیری از منطق فازی کلاسیک شامل توابع عضویت مثلثی و فرآیند قطعی سازی، دقت و انعطاف بیشتری در تفسیر دستورات فازی ارائه می دهد؛ بنابراین، دیپ سیک برای محیط های پیچیده تر و توسعه پذیر مناسب تر است، در حالی که چت جی پی تی راه حلی سریع و ساده برای سناریوهای پایه ای فراهم می کند. لازم به ذکر است خروجی دو مدل نقشه اولیه ی تصمیم و منطق است، ولی برای اجرا باید از یک لایه ی پس پردازش و کنترل سخت افزاری عبور کند.

۲-۴ ارزیابی عملکرد و دقت دو مدل زبانی. مدل دیپ سیک: تمرکز اصلی این مدل بر کارایی و بهینگی محاسباتی است؛ به گونه ای که پردازش سریع و کم هزینه را امکان پذیر می سازد. دیپ سیک در اجرای دستورات صریح و عددی (مانند «یک قاشق شکر») دقت بالاتری دارد. با این حال، در حوزه ی انعطاف پذیری زبانی و تفسیر دستورات مبهم یا فازی کمتر توسعه یافته است.

مدل جی پی تی: این مدل بر هوشمندی و انعطاف زبانی تمرکز دارد و توانایی بالایی در درک و تفسیر عبارات مبهم یا فازی (مانند « کمی شیرین ») نشان می دهد. جی پی تی در تبدیل زبان طبیعی به عمل، دقیق تر و نزدیک تر به فهم انسانی عمل می کند. هر چند از نظر مصرف منابع، هزینه برتر و کمتر بهینه نسبت به دیپ سیک است، اما قابلیت های زبانی آن برتری محسوسی دارد. به طور کلی همچنان که دیده شد روش ارائه شده امکان دریافت دستورات زبانی به صورت فازی را دارد و خروجی ها که در واقع توابع پایتون برای اعمال به ربات هستند ارائه می شوند.

۵- نتیجه گیری و کارهای آینده

در این پژوهش، رویکردی برای اجرای وظایف پیچیده توسط ربات ها با استفاده از مدل های زبانی بزرگ، منطق فازی و شکستن وظایف به مهارت های پایه ارائه شد. آزمایش ها با بهره گیری از چت جی پی تی و دیپ سیک انجام گرفت. نتایج کمی نشان داد که روش پیشنهادی توانسته است نرخ موفقیت بالاتری در اجرای وظایف (بیش از ۹۰ درصد) به دست آورد و میانگین

جدول ۱: مقایسه توانمندی‌های مدل‌ها

روش	ماژولار	تعمیم‌پذیری	امکان دریافت دستور زبانی واضح و یا فازی
کلاسیک [۱۱]	بله	خیر	خیر
روش مقاله مرجع [۱۸]	خیر	خیر	فقط واضح
روش ارائه‌شده	بله	بله	هم واضح هم فازی

زمان اجرا را حدود ۲۰ درصد کاهش دهد. این آزمایش‌ها نشان می‌دهد که مدل‌های زبانی بزرگ علاوه بر قابلیت درک و تفسیر صحیح دستورات واضح و فازی، می‌توانند برنامه‌ریزی انعطاف‌پذیر و تعاملی را برای ربات‌ها تسهیل کنند.

با این وجود، مطالعه حاضر با محدودیت‌هایی نیز همراه است. نخست، وابستگی شدید به کیفیت پرامپت‌ها می‌تواند در مواردی موجب کاهش دقت در اجرای وظایف شود. دوم، احتمال تولید گام‌های نادرست یا غیرمنطقی توسط مدل‌های زبانی همچنان وجود دارد و نیازمند سازوکارهای پس‌پردازش است. سوم، تبدیل خروجی مدل‌های زبانی به دستورات سخت‌افزاری دقیق در محیط واقعی چالش‌برانگیز باقی‌مانده است. افزون بر این، تنظیم نرم مدل‌های زبانی بزرگ به منابع محاسباتی قدرتمند و پردازنده‌های گرافیکی پیشرفته نیاز دارد. به‌عنوان مسیرهای آتی تحقیق، می‌توان به چند محور کلیدی اشاره کرد:

- (۱) تنظیم نرم مدل‌های زبانی بزرگ با استفاده از داده‌های عملی و فیدبک کاربر مبتنی بر یادگیری تقویتی؛
- (۲) پیاده‌سازی و ارزیابی رویکرد پیشنهادی بر روی بازوی ربات واقعی به‌جای محیط شبیه‌سازی؛
- (۳) توسعه مکانیزم‌های کنترل خطا برای کاهش وابستگی به کیفیت پرامپت و جلوگیری از تولید دستورات نادرست.

مراجع

- [1] J. Krüger, G. Schreck, and D. Surdilovic, "Dual-arm robot for flexible and cooperative assembly," *CIRP Annals*, vol. 60, no. 1, pp. 5–8, 2011.
- [2] P. Kazanzides et al., "An open-source research kit for the da Vinci® Surgical System," *IEEE Robotics and Automation Magazine*, vol. 24, no. 2, pp. 48–56, 2017.

- [3] M. Bac et al., "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1310–1332, 2020.
- [4] R. Yamamoto et al., "Service robot integration into daily human environments," *Robotics and Autonomous Systems*, vol. 95, pp. 3–17, 2017.
- [5] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.
- [6] D. Whitney, M. Oberlin, and S. Tellex, "Interpreting multimodal referring expressions in real time," *Robotics: Science and Systems (RSS)*, 2016.
- [7] J. Andreas, D. Klein, and S. Levine, "Learning with latent language," *Proceedings of EMNLP*, pp. 2166–2178, 2018.
- [8] T. Tellex et al., "Understanding natural language commands for robotic navigation and mobile manipulation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 409–434, 2013.
- [9] J. Xu, J. White, and M. Gombolay, "Learning to compose skills for real-world robotic manipulation," *Proceedings of RSS*, 2022.
- [10] Y. Ahn et al., "Do as I can, not as I say: Grounding language in robotic affordances," *Proceedings of ICML*, 2022.
- [11] B. Ichter et al., "Language-model-based imitation learning for robotic manipulation," *arXiv preprint arXiv:2209.07753*, 2022.
- [12] M. Jain et al., "Bridging language and motor control with skill libraries," *Conference on Robot Learning (CoRL)*, 2023.
- [13] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," *Autonomous Robots*, vol. 36, no. 1, pp. 123–143, 2013.
- [14] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed., Pearson Prentice Hall, 2005.
- [15] A. Vaswani et al., "Attention is all you need," *NeurIPS*, vol. 30, 2017.
- [16] T. Brown et al., "Language models are few-shot learners," *NeurIPS*, vol. 33, pp. 1877–1901, 2020.
- [17] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.
- [18] Y. Kojima, S. Zhi, Y. Bai, and A. Zeng, "Large language models as zero-shot planners for robots," *RSS*, 2023.

پیوست

مثال‌های دیگر: در ادامه، مثالی از یک درخواست کاربر برای تهیه کاپوچینو با شکر آورده شده است. دستور کاربر در پرامپت از پیش طراحی شده گنجانده شده و به مدل‌های زبانی ارائه می‌شود.

پرامپت ارائه شده به جت‌جی‌پی‌تی بابت دو درخواست کاربر: لطفاً یک فنجان کاپوچینو با دو قاشق شکر برایم درست کن. لطفاً یک فنجان کاپوچینو خیلی شیرین برایم درست کن. خروجی جت‌جی‌پی‌تی پاسخ به دستور اول: ("یک فنجان کاپوچینو با دو قاشق شکر")
گام ۱: دریافت تصویر از محیط

```
image = CaptureImage()
```

گام ۲: پیش پردازش تصویر

```
processed_image = ProcessImage(image)
```

گام ۳: شناسایی اشیاء در محیط

```
objects = DetectObjects(processed_image)
```

گام ۴: دریافت مختصات دستگاه قهوه‌ساز و ظرف شکر

```
coffee_machine_coords = GetObjectCoordinates("coffee_machine")
```

```
sugar_container_coords = GetObjectCoordinates("sugar_container")
```

گام ۵: محاسبه بهترین نقطه گرفتن برای ظرف شکر

```
sugar_grasp_point = CalculateGraspPoint(sugar_container_coords)
```

گام ۶: ارسال سیگنال به قهوه‌ساز برای آماده‌سازی کاپوچینو

```
SendSignal("make_cappuccino")
```

گام ۷: ارسال سیگنال برای برداشتن دقیقاً دو قاشق شکر

```
SendSignal("sugar_unit")
```

```
SendSignal("sugar_unit")
```

خروجی جت‌جی‌پی‌تی پاسخ به دستور دوم: ("یک فنجان کاپوچینو خیلی شیرین")

گام ۱: دریافت تصویر از محیط

```
image = CaptureImage()
```

گام ۲: پیش‌پردازش تصویر

```
processed_image = ProcessImage(image)
```

گام ۳: شناسایی اشیاء در محیط

```
objects = DetectObjects(processed_image)
```

گام ۴: دریافت مختصات دستگاه قهوه‌ساز و ظرف شکر

```
coffee_machine_coords = GetObjectCoordinates("coffee_machine")
```

```
sugar_container_coords = GetObjectCoordinates("sugar_container")
```

گام ۵: محاسبه بهترین نقطه گرفتن برای ظرف شکر

```
sugar_grasp_point = CalculateGraspPoint(sugar_container_coords)
```

گام ۶: ارسال سیگنال به قهوه‌ساز برای آماده‌سازی کاپوچینو

```
SendSignal("make_cappuccino")
```

گام ۷: تفسیر مقدار فازی "خیلی شیرین"

```
def InterpretSweetness(level):
```

```
    fuzzy_map = {"unsweetened": 0, "slightly": 5.0,
```

```
                "medium": 1, "sweet": 5.1, "very sweet": 2}
```

```
    return fuzzy_map.get(level, 1)
```

```
sugar_amount = InterpretSweetness("very sweet")
```

گام ۸: ارسال سیگنال متناسب با مقدار فازی شکر

```
if sugar_amount == 0:
```

```
    pass #
```

```
elif sugar_amount == :5.0
```

```
    SendSignal("sugar_unit_half")
```

```
elif sugar_amount == 1:
```

```
    SendSignal("sugar_unit")
```

```
elif sugar_amount == :5.1
```

```
    SendSignal("sugar_unit")
```

```
    SendSignal("sugar_unit_half")
```

```
elif sugar_amount == 2:
    SendSignal("sugar_unit")
    SendSignal("sugar_unit")
```

در حالت واضح (دو قاشق شکر): اجرای مستقیم و قطعی. در حالت فازی (خیلی شیرین): مدل نگاشت زبانی را تفسیر کرده و معادل کمی (۲ قاشق) اجرا می‌کند. پرامیت ارائه شده به دیپسیک بابت دو درخواست کاربر: لطفاً یک فنجان کاپوچینو با دو قاشق شکر برایم درست کن. لطفاً یک فنجان کاپوچینو خیلی شیرین برایم درست کن. خروجی دیپسیک پاسخ به دستور اول: ("یک فنجان کاپوچینو با دو قاشق شکر")

۱. گرفتن تصویر از محیط

```
image = CaptureImage()
```

۲. پیش پردازش تصویر برای بهبود تشخیص

```
processed_image = ProcessImage(image)
```

۳. تشخیص اشیای موجود در صحنه (لیوان، قهوه‌ساز، شکر و غیره)

```
detected_objects = DetectObjects(processed_image)
```

۴. یافتن لیوان خالی و گرفتن مختصات آن

```
cup_coords = GetObjectCoordinates("empty_cup")
```

```
if not cup_coords:
```

```
    print("Error: Cup not found! ")
```

```
    return
```

۵. محاسبه نقطه مناسب برای گرفتن لیوان

```
grasp_point = CalculateGraspPoint(cup_coords)
```

۶. حرکت به سمت لیوان و برداشتن آن

```
MoveToCoordinates(grasp_point)
```

```
OpenGrip()
```

```
MoveToCoordinates(grasp_point)
```

```
CloseGrip()
```

۷. بردن لیوان زیر دستگاه قهوه‌ساز

```
coffee_machine_coords = GetObjectCoordinates("coffee_machine")
MoveToCoordinates(coffee_machine_coords)
```

۸. فعال‌سازی قهوه‌ساز برای درست‌کردن کاپوچینو

```
SendSignal("coffee_machine")
```

۹. منتظر ماندن تا قهوه آماده شود (می‌تواند با سنسور یا تایمر باشد) ۱.۹. پس از آماده‌شدن قهوه، برداشتن لیوان

```
MoveToCoordinates(coffee_machine_coords)
```

۱۰. یافتن ظرف شکر و محاسبه نقطه گرفتن آن

```
sugar_coords = GetObjectCoordinates("sugar_container")
sugar_grasp_point = CalculateGraspPoint(sugar_coords)
```

۱۱. بردن لیوان به بالای ظرف شکر

```
MoveToCoordinates(sugar_grasp_point)
```

۱۲. اضافه‌کردن دو قاشق شکر (برای "خیلی شیرین")

```
for i in range(2):
    RotateGrip(45)
```

۱۳. برگرداندن لیوان به حالت اولیه

```
RotateGrip(0)
```

۱۴. قراردادن لیوان در موقعیت سرو

```
serving_coords = GetObjectCoordinates("serving_area")
MoveToCoordinates(serving_coords)
```

۱۵. رهاکردن لیوان

```
OpenGrip()
```

```
print("Your super sweet cappuccino is ready! Enjoy!")
```

۱۶. اجرای برنامه

```
make_very_sweet_cappuccino()
```

خروجی دیپ‌سیک پاسخ به دستور دوم: ("یک فنجان کاپوچینو خیلی شیرین")

۱. گرفتن تصویر و پردازش اولیه

```
image = CaptureImage()
processed_image = ProcessImage(image)
detected_objects = DetectObjects(processed_image)
```

۲. یافتن و برداشتن لیوان

```
cup_coords = GetObjectCoordinates("empty_cup")
grasp_point = CalculateGraspPoint(cup_coords)
MoveToCoordinates(grasp_point)
OpenGrip()
MoveToCoordinates(grasp_point)
CloseGrip()
```

۳. درست کردن قهوه پایه

```
coffee_coords = GetObjectCoordinates("coffee_machine")
MoveToCoordinates(coffee_coords)
SendSignal("coffee_machine")
```

۴. اضافه کردن شکر به مقدار "خیلی شیرین"

```
sugar_coords = GetObjectCoordinates("sugar_container")
MoveToCoordinates(sugar_coords)
```

۵. سیستم فازی برای تعیین میزان شیرینی

"خیلی شیرین" = چرخش بیشتر + مدت زمان

چرخش بیشتر برای ریختن سریع‌تر

```
RotateGrip(60)
```

۲.۵ تاخیر بیشتر برای ریختن مقدار بیشتر شکر (می‌توان از تایمر یا سنسور جریان استفاده کرد)

۶. برگرداندن به حالت عادی

```
RotateGrip(0)
```

۷. تحویل نوشیدنی

```
serving_coords = GetObjectCoordinates("serving_area")
MoveToCoordinates(serving_coords)
OpenGrip()
print("Your super sweet cappuccino with the fuzzy
system is ready! ")
```

اجرای برنامه فازی

```
make_fuzzy_sweet_cappuccino()
```