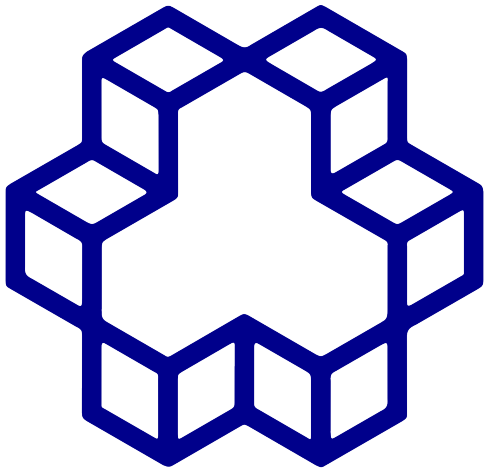


دانشگاه صنعتی خواجه نصیرالدین طوسی

یادگیری دنبال کردن مسیر در ربات‌های موازی صفحه‌ای مبتنی بر کابل با استفاده از یادگیری تقویتی

دانشجو:

دانبال عبداللهی نژاد

۴۰۱۰۹۱۶۴

استاد درس:

دکتر علیاری

تابستان ۱۴۰۳

چکیده

این پروژه به بررسی و پیاده‌سازی کنترل ربات‌های موازی کابلی با استفاده از الگوریتم‌های یادگیری تقویتی می‌پردازد. در ابتدا، مقدمات ربات‌های سری و موازی و به‌ویژه ربات‌های موازی کابلی مورد بحث قرار گرفته و مزایا و محدودیت‌های آنها بررسی می‌شود. سپس، محیط شبیه‌ساز دینامیکی MuJoCo برای پیاده‌سازی ربات کابلی کمال الملک مورد استفاده قرار می‌گیرد. برای کنترل ربات، از الگوریتم Deep Deterministic Policy Gradient (DDPG) استفاده شده است که ترکیبی از شبکه‌های عصبی عمیق و یادگیری تقویتی است. این الگوریتم برای مدیریت سیستم‌های پیچیده و غیرخطی مانند ربات‌های موازی کابلی بسیار موثر است. در بخش پیاده‌سازی، مراحل مختلف شامل تعریف اجزای ربات در MuJoCo، افزودن کابل‌ها و محرک‌ها، و همچنین استفاده از کتابخانه stable-baselines3 برای پیاده‌سازی الگوریتم DDPG توضیح داده شده است. همچنین، مسئله ردیابی مسیر دایره‌ای به عنوان یکی از کاربردهای مهم کنترل ربات بررسی و فرموله شده است. نتایج نشان می‌دهند که الگوریتم DDPG توانسته است با دقت خوبی مسیر دایره‌ای را ردیابی کند و خطاهای سرعت و موقعیت را به حداقل برساند. همچنین، مزایای استفاده از یادگیری تقویتی برای کنترل ربات‌های موازی کابلی به وضوح نشان داده شده است. در نهایت، نتیجه‌گیری‌هایی در مورد کارایی و پایداری روش‌های یادگیری تقویتی برای کنترل ربات‌های پیچیده ارائه شده و بهبودهای ممکن در آینده مورد بحث قرار گرفته است.

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

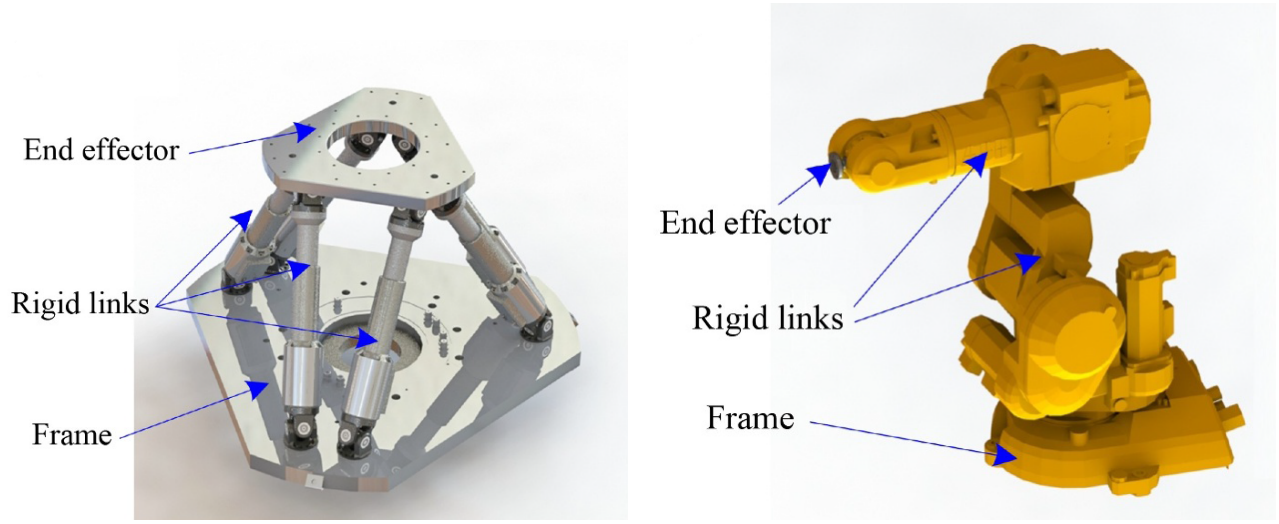
ربات یک ماشین چند منظوره با قابلیت برنامه‌ریزی است که می‌توان از آن‌ها برای طیف گسترده‌ای از کاربردها استفاده کرد. ربات‌ها را می‌توان از نقطه نظرهای گوناگون مانند کاربرد، درجات آزادی ، ساختار سینماتیکی و غیره دسته‌بندی کرد. با دسته‌بندی ربات‌ها از نظر زنجیره سینماتیکی، انواع ربات‌ها در دو شاخه سری و موازی دسته‌بندی می‌شوند.

۱.۱ ربات‌های سری و موازی

ربات‌های سری به خانواده‌ای از ربات‌ها گفته می‌شود که در آن مجری نهایی ربات به واسطه تنها یک زنجیره سینماتیکی کنترل شده و به نسبت ساختار ساده‌تری دارند. در این ربات‌ها معمولا محرک‌ها در محل اتصال بازوها قرار گرفته و یک درجه آزادی در آن نقطه ایجاد می‌کنند. محاسبه وضعیت مجری نهایی[۱] با داشتن اطلاعات بازوها ساده می‌باشد، درحالی که مسئله سینماتیک وارون آن پیچیده است.

مکانیسم‌های موازی دارای دو یا بیشتر زنجیره‌های سینماتیکی هستند که به طور همزمان پایه و مجری نهایی را به یکدیگر متصل می‌کنند. این مکانیسم یک مکانیسم حلقه بسته با مفصل‌های غیرفعال است. ربات‌های موازی می‌توانند

[1] end effector



(ب) ربات موازی (آ) ربات سری

شکل ۱.۱: یک نمونه از ربات‌های سری و موازی

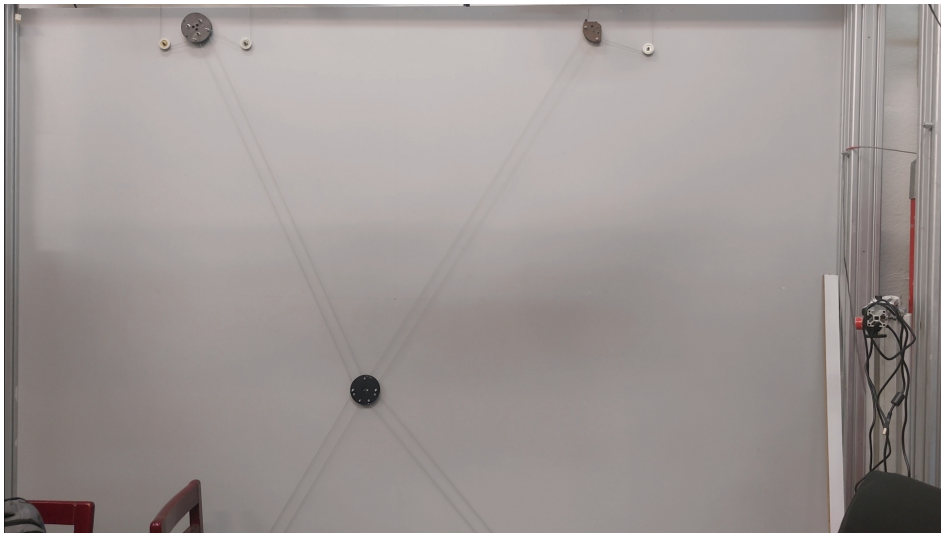
به‌عنوان چندین ربات سری با هم حمل کننده‌ی مجری نهایی در نظر گرفته شوند. در شکل ۱.۱ یک نمونه از ربات‌های سری و موازی مشاهده می‌شود.

۲.۱ ربات موازی کابلی

ربات کابلی از خانواده ربات‌های موازی است. یکی از بزرگ‌ترین نقاط ضعف ربات‌های موازی، فضای کاری محدود آن‌ها است، که این محدودیت به دلیل وجود بازوهای موازی و متصل به مجری نهایی ایجاد می‌شود. به منظور رفع مشکل فضای کاری ربات‌های موازی، می‌توان از کابل به جای بازوی صلب استفاده کرد؛ در این صورت محل قرارگیری عملگرها با محل مجری نهایی می‌تواند فاصله‌ی زیادی داشته باشد و در نتیجه فضای کاری ربات افزایش پیدا کند. از سوی دیگر، کابل‌ها ممان اینرسی کوچک‌تری دارند و در نتیجه ربات می‌تواند سرعت‌های بالاتری داشته باشد. نکته حائز اهمیت درباره این ربات آن است که این کابل‌ها توانایی اعمال نیروی فشاری را ندارند، پس کابل‌های متصل به ربات باید همیشه در حال کشش باشند. این مساله محدودیتی جدی در مورد این خانواده از ربات‌ها به حساب می‌آید و برای کنترل این ربات دقت به این نکته حائز اهمیت است.

۳.۱ ربات کابلی کمال الملک

ربات کابلی کمال الملک یکی از ربات‌های طراحی شده در آزمایشگاه رباتیک ارس است که به طور ویژه برای طراحی و نقاشی ساخته شده است (شکل ۲.۱). این ربات به عنوان یک سیستم دو درجه آزادی صفحه‌ای طراحی شده که توانایی طراحی بر روی یک تخته عمودی را دارد. همچنین این ربات از افزونگی سینماتیکی برخوردار است، به این معنا که با وجود داشتن دو درجه آزادی، از سه محرک برای کنترل استفاده می‌کند. مجری نهایی این ربات از طریق سه کابل که هر یک به یک موتور سروو متصل هستند، به حرکت در می‌آید. این کابل‌ها با چرخش موتور حول یک درام پیچیده می‌شوند و با تنظیم دقیق طول و کشش کابل‌ها، مجری نهایی ربات به طور دقیق در مسیرهای تعیین شده حرکت می‌کند. هر یک از موتورهای سروو به صورت مستقل کنترل می‌شوند و از طریق ترکیب حرکات آنها، ربات قادر است به دقت و با انعطاف‌پذیری بالا، نقاشی‌ها و طرح‌های مختلفی را بر روی تخته عمودی ایجاد کند.



شکل ۲.۱: ربات کابلی کمال الملک آزمایشگاه رباتیک ارس

۴.۱ یادگیری تقویتی در رباتیک

استفاده از Reinforcement Learning (RL) در رباتیک به دلیل توانایی آن در یادگیری و بهینه‌سازی از طریق تعامل با محیط، به طور گسترده‌ای مورد توجه قرار گرفته است. در این روش، عامل (ربات) با محیط تعامل می‌کند و از طریق دریافت پاداش یا تنبیه، سیاست‌های بهینه را برای انجام وظایف مختلف یاد می‌گیرد. این روش به ویژه در سیستم‌های پیچیده و غیرخطی که مدل‌سازی دقیق آن‌ها دشوار است، بسیار مفید است.

یکی از مثال‌های موفق استفاده از RL در رباتیک، کنترل ربات‌های تعاملی است. به عنوان مثال، در تحقیقی توسط Levine et al.، از RL برای کنترل یک ربات تعاملی استفاده شده است که قادر است وظایف پیچیده‌ای مانند برداشتن و جابجایی اشیاء را انجام دهد [۱]. این ربات با استفاده از دوربین و حسگرهای خود، محیط را مشاهده کرده و از طریق یادگیری تقویتی، سیاست‌های بهینه را برای انجام وظایف مختلف یاد می‌گیرد.

در مثالی دیگر، Kober et al. از RL برای کنترل ربات‌های پروازی استفاده کرده‌اند. در این تحقیق، ربات‌های پروازی با استفاده از RL قادر به یادگیری پرواز پایدار و انجام مانورهای پیچیده شدند [۲]. این روش به ربات‌ها اجازه می‌دهد تا از طریق تجربه و تعامل با محیط، بهینه‌سازی کنند و عملکرد خود را بهبود بخشند.

همچنین، در تحقیق دیگری توسط Lillicrap et al.، از الگوریتم DDPG برای کنترل یک ربات بازویی استفاده شده است. این ربات با استفاده از RL قادر به یادگیری و اجرای حرکات دقیق و پیچیده شده است [۳]. این تحقیق نشان می‌دهد که الگوریتم‌های RL مانند DDPG می‌توانند در کنترل ربات‌های پیچیده و غیرخطی بسیار مؤثر باشند. در زمینه کنترل ربات‌های کابلی موازی، (CDPR) روش‌های مختلفی از RL مورد بررسی قرار گرفته‌اند. به عنوان مثال Li, S et al. رویکردی نوآورانه برای کنترل ربات‌های کابلی موازی مسطح با استفاده از یادگیری تقویتی ارائه داد شده‌است. نویسندگان با بهره‌گیری از RL به چالش‌های کنترل دقیق موقعیت در هاCDPR پرداخته‌اند و با تمرکز بر پیچیدگی‌ها و غیرخطی‌های ذاتی این سیستم‌ها، نشان داده‌اند که استفاده از الگوریتم‌های RL می‌تواند بهبودهای قابل توجهی در توانایی ربات در دستیابی به کنترل دقیق موقعیت فراهم آورد. این مقاله تحلیل جامع از استراتژی کنترلی مبتنی بر RL ارائه می‌دهد و اثربخشی آن را در مقایسه با روش‌های کنترلی سنتی برجسته می‌کند. این کار نه تنها پیشرفت قابل توجهی در زمینه کنترل هاCDPR به شمار می‌آید، بلکه نشان‌دهنده پتانسیل RL در بهبود عملکرد سیستم‌های رباتیک پیچیده است [۴].

شکل ۳.۱: نمونه‌ای از ربات‌های تعاملی با محیط

به طور خلاصه، استفاده از RL در رباتیک می‌تواند به بهبود عملکرد و دقت ربات‌ها در انجام وظایف مختلف کمک کند. این روش‌ها به ربات‌ها اجازه می‌دهند تا از طریق تعامل با محیط و یادگیری از تجربه، سیاست‌های بهینه را برای انجام وظایف مختلف بیاموزند و بهبود یابند. با توجه به پیچیدگی‌ها و چالش‌های موجود در کنترل ربات‌ها، استفاده از RL می‌تواند راه‌حل‌های نوآورانه و مؤثری را ارائه دهد.

فصل ۲

برنامه شبیه‌ساز دینامیکی MuJoCo

MuJoCo مخفف "Multi-Joint dynamics with Contact" است. این یک موتور فیزیک عمومی است که هدف آن تسهیل پژوهش و توسعه در رباتیک، بیومکانیک، گرافیک و انیمیشن، یادگیری ماشین و سایر زمینه‌هایی است که نیاز به شبیه‌سازی سریع و دقیق ساختارهای مفصلی در تعامل با محیط دارند. این موتور ابتدا توسط شرکت Roboti LLC توسعه داده شد و در اکتبر 2021 توسط DeepMind خریداری و به صورت رایگان در دسترس قرار گرفت و در ماه مه 2022 به صورت متن‌باز منتشر شد. کدبیس MuJoCo در مخزن deepmind/mujoco در GitHub موجود است [۵].

MuJoCo یک کتابخانه C/C++ با API زبان C است که برای پژوهشگران و توسعه‌دهندگان طراحی شده است. ماژول شبیه‌سازی در زمان اجرا به منظور حداکثر کردن عملکرد تنظیم شده و بر روی ساختارهای داده سطح پایین عمل می‌کند که توسط مفسر و کامپایلر داخلی XML پیش تخصیص داده شده‌اند. کاربر مدل‌ها را در زبان توصیف صحنه بومی MJCF تعریف می‌کند — یک فرمت فایل XML که به گونه‌ای طراحی شده است که تا حد ممکن خوانا و قابل ویرایش توسط انسان باشد. فایل‌های مدل URDF نیز قابل بارگذاری هستند. این کتابخانه شامل تصویری‌سازی تعاملی با یک رابط کاربری گرافیکی بومی است که در OpenGL رندر می‌شود. MuJoCo همچنین تعداد زیادی از توابع کمکی برای محاسبه مقادیر مرتبط با فیزیک ارائه می‌دهد.

MuJoCo می‌تواند برای اجرای محاسبات مبتنی بر مدل مانند سنتز کنترل، تخمین حالت، شناسایی سیستم، طراحی

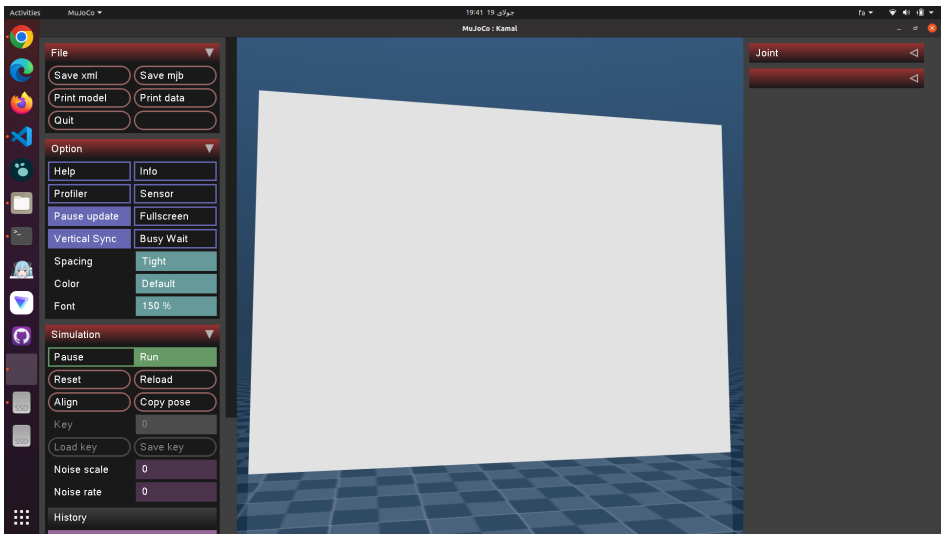
مکانیزم، تحلیل داده از طریق دینامیک معکوس، و نمونه‌برداری موازی برای کاربردهای یادگیری ماشین استفاده شود. همچنین می‌تواند به عنوان یک شبیه‌ساز سنتی‌تر، از جمله برای بازی و محیط‌های مجازی تعاملی، مورد استفاده قرار گیرد.

۱.۲ پیاده‌سازی ربات کمال الملک در MuJoCo

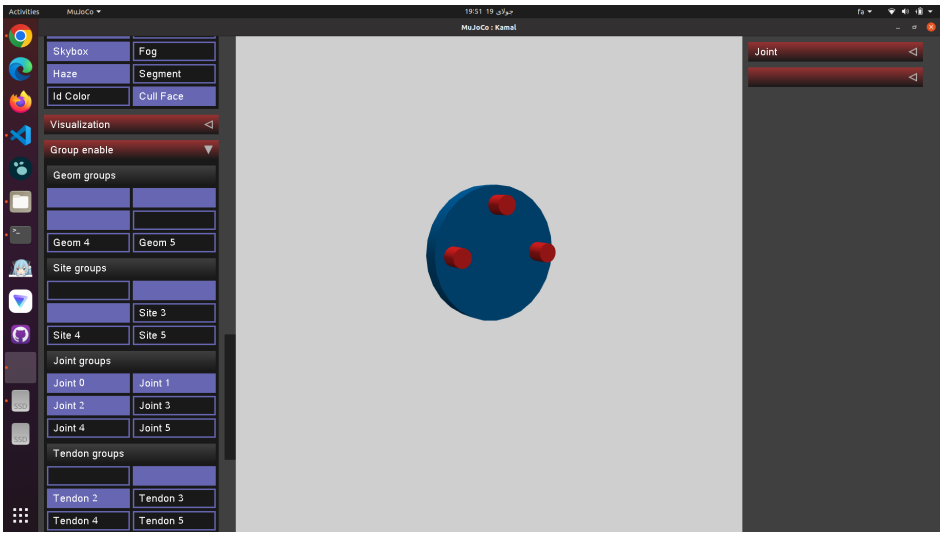
برای پیاده‌سازی ربات کمال الملک در MuJoCo، باید اجزای اصلی ربات را با دقت تعریف کنید. در اینجا یک توضیح مرحله‌به‌مرحله از ایجاد این اجزا در MuJoCo آورده شده است:

۱.۱.۲ تعریف تخته

ابتدا باید یک تخته اصلی که بقیه اجزاء به آن متصل می‌شوند، ایجاد کنید. این تخته به عنوان پایه‌ای ثابت برای ربات عمل خواهد کرد و باید در محیط XML تعریف شود. تخته تعریف شده در شکل ۱.۲ نشان داده شده‌است.



شکل ۱.۲: تعریف تخته ربات



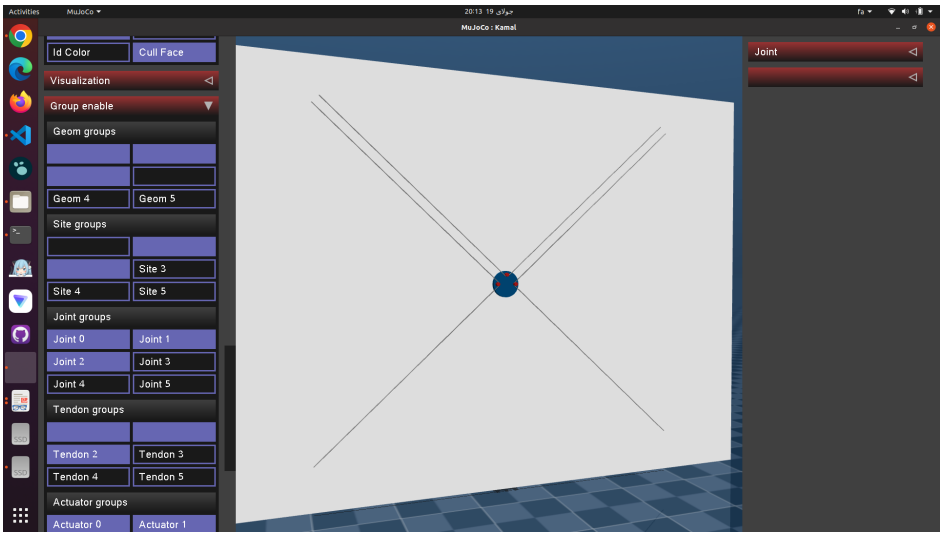
شکل ۲.۲: مجری نهایی ربات

۲.۱.۲ تعریف مجری نهایی

در تصویر ۲.۲، مجری نهایی ربات کمال الملک نمایش داده شده است. این مجری نهایی شامل یک صفحه دایره‌ای به عنوان پایه است که سه عدد پولی روی آن نصب شده‌است. پولی‌ها در موقعیت‌های استراتژیک روی سطح دایره‌ای قرار گرفته‌اند تا امکان حرکت و انتقال نیرو به طور موثر فراهم شود. این پولی‌ها برای هدایت کابل‌ها و انتقال نیرو از محرک‌ها به مجری نهایی استفاده می‌شوند.

۳.۱.۲ افزودن کابل‌ها

کابل‌ها بخش حیاتی ربات کمال الملک هستند و به عنوان عناصر انتقال نیرو از محرک‌ها به مجری نهایی عمل می‌کنند. در MuJoCo، کابل‌ها مسیری با حداقل طول را با عبور از نقاط مشخص شده و پیچیدن به دور اجسام مشخص ایجاد می‌کند و می‌توانند به صورت عناصر کشسان تعریف شوند. هر کابل باید دارای خصوصیاتی مانند قطر، محدودیت طولی، کشسانیت و مقاومت در برابر فشار و کشش باشد.



شکل ۳.۲: کابل‌های ربات

۴.۱.۲ اضافه کردن محرک‌ها

محرک‌ها نیروی لازم برای حرکت دادن کابل‌ها و در نتیجه حرکت دادن مجری نهایی را فراهم می‌کنند. در MuJoCo، محرک‌ها به صورت موتورهای الکتریکی که نیرو یا سرعت زاویه‌ای را کنترل می‌کنند، تعریف شده‌اند. تعریف محرک‌ها شامل تنظیماتی مانند نوع محرک، قدرت و مد رفتاری آن‌ها است. این محرک‌ها مسئولیت تبدیل سیگنال‌های کنترلی به نیرو یا گشتاور را بر عهده دارند که به اجزای متحرک ربات اعمال می‌شوند تا حرکت مورد نظر را ایجاد کنند.

۲.۲ نتیجه‌گیری

در این فصل به بررسی شبیه‌ساز دینامیکی MuJoCo و کاربرد آن در پیاده‌سازی و کنترل ربات‌های کابلی پرداخته شد. MuJoCo به عنوان یک ابزار قدرتمند در شبیه‌سازی دینامیک سیستم‌های چند مفصلی، توانایی ارائه شبیه‌سازی‌های دقیق و سریع را دارا می‌باشد که این ویژگی‌ها آن را به یک انتخاب مناسب برای تحقیق و توسعه در حوزه رباتیک تبدیل کرده است.

با استفاده از MuJoCo، امکان مدل‌سازی دقیق اجزای ربات، شامل تخته پایه، مجری نهایی، کابل‌ها و محرک‌ها،

فراهم شد. این شبیه‌ساز به ما اجازه می‌دهد تا به صورت واقع‌گرایانه رفتار و عملکرد ربات‌های کابلی را بررسی کنیم و از این طریق نقاط قوت و ضعف طراحی‌های مختلف را شناسایی و بهبود بخشیم.

MuJoCo همچنین با پشتیبانی از زبان‌های برنامه‌نویسی مانند C و Python و فراهم کردن یک رابط کاربری گرافیکی کاربرپسند، فرآیند توسعه و آزمایش الگوریتم‌های کنترل ربات را تسهیل می‌کند. این ویژگی‌ها، در کنار قابلیت تعامل با محیط‌های مجازی و بازی، نشان‌دهنده‌ی انعطاف‌پذیری و کارآمدی این شبیه‌ساز در کاربردهای مختلف است.

در نهایت، نتایج به دست آمده از شبیه‌سازی‌ها با استفاده از MuJoCo نشان می‌دهد که این ابزار می‌تواند به طور مؤثری در بهینه‌سازی و ارزیابی سیستم‌های رباتیکی مورد استفاده قرار گیرد و به محققان در دستیابی به اهداف پژوهشی خود کمک شایانی کند.

فصل ۳

یادگیری تقویتی

۱.۳ الگوریتم Deep Deterministic Policy Gradient (DDPG)

۱.۱.۳ مقدمه

الگوریتم Deep Deterministic Policy Gradient (DDPG) یک روش پیشرفته در یادگیری تقویتی (RL) است که به‌طور خاص برای فضاهای عمل‌های پیوسته و ابعاد بالا طراحی شده است. این الگوریتم به ویژه برای کنترل سیستم‌های پیچیده مانند ربات‌های موازی کابل‌محور (CDPR) بسیار مناسب است، زیرا این ربات‌ها نیاز به استراتژی‌های کنترل دقیق و تطبیق‌پذیر دارند به دلیل سینماتیک پیچیده و محیط‌های پویای خود. DDPG با ترکیب مزایای شبکه‌های عصبی عمیق (DQN) و گرادیان سیاست قطعی (DPG)، چارچوبی قدرتمند برای یادگیری سیاست‌های کنترل بهینه فراهم می‌آورد و به‌طور موثر در حل مسائل پیچیده کنترل به کار می‌رود [۶–۸].

۲.۱.۳ Markov Decision Process (MDP) و فرمول‌بندی آن

فرآیند تصمیم‌گیری مارکوف (MDP) یک چارچوب ریاضی برای مدل‌سازی تصمیم‌گیری در شرایط عدم قطعیت است. MDP شامل موارد زیر است [۹]:

- مجموعه حالات $\mathcal{S}$: همه حالات ممکن.
- مجموعه اعمال $\mathcal{A}$: همه اعمال ممکن.
- تابع انتقال $P(s'|s,a)$: احتمال انتقال از حالت s به حالت s' با انجام عمل a.
- تابع پاداش $R(s,a)$: پاداش دریافتی پس از انجام عمل a در حالت s.
- عامل تخفیف γ: تعیین اهمیت پاداش‌های آینده.

هدف یافتن سیاست (Policy) π است که به ما می‌گوید در هر حالت کدام عمل را انتخاب کنیم تا مجموع پاداش‌های تخفیف‌یافته بیشینه شود.

$$V^{\pi}(s) = E\left[\sum_{t=۰}^{\infty} \gamma^t R(s_t, a_t) \mid s_۰ = s, \pi\right]$$

Deep Q-Network (DQN) ۳.۱.۳

DQN یک الگوریتم یادگیری تقویتی است که از شبکه‌های عصبی برای تقریب تابع ارزش استفاده می‌کند. هدف DQN یادگیری سیاستی است که به حداکثر پاداش مورد انتظار دست یابد. DQN از یک شبکه عصبی برای تقریب تابع Q استفاده می‌کند:

$$Q(s,a|\theta) \approx Q^*(s,a)$$

جایی که θ وزن‌های شبکه عصبی هستند. به‌روزرسانی وزن‌ها بر اساس معادله زیر انجام می‌شود:

$$L(\theta) = E\left[\left(R + \gamma \max_{a'} Q(s', a'|\theta^-) - Q(s,a|\theta)\right)^۲\right]$$

۴.۱.۳ Deterministic Policy Gradient (DPG)

DPG یک الگوریتم یادگیری تقویتی است که برای فضای عمل‌های پیوسته طراحی شده است. DPG از یک سیاست قطعی [۱] $\mu(s|\theta^{\mu})$ استفاده می‌کند و از تابع ارزش Q برای به‌روزرسانی سیاست بهره می‌برد. به‌روزرسانی سیاست بر اساس گرادیان سیاست قطعی انجام می‌شود [۸]:

$$\nabla_{\theta^{\mu}} J = E_{s\sim\rho^{\beta}}\left[\nabla_a Q(s,a|\theta^Q)\Big|_{a=\mu(s|\theta^{\mu})} \nabla_{\theta^{\mu}}\mu(s|\theta^{\mu})\right]$$

۵.۱.۳ سیاست‌های یادگیری تقویتی مختلف

در یادگیری تقویتی، الگوریتم‌ها به دو دسته اصلی On-Policy و Off-Policy تقسیم می‌شوند.

On-Policy

در الگوریتم‌های On-Policy، سیاستی که برای تولید داده‌های آموزشی استفاده می‌شود همان سیاستی است که بهینه‌سازی می‌شود. یک مثال معروف از الگوریتم On-Policy، Policy Gradient است که به صورت مستقیم گرادیان تابع پاداش را به‌روزرسانی می‌کند تا سیاست بهینه را پیدا کند [۹].

Off-Policy

در الگوریتم‌های Off-Policy، سیاستی که برای تولید داده‌های آموزشی استفاده می‌شود ممکن است متفاوت از سیاستی باشد که بهینه‌سازی می‌شود. Q-Learning یک مثال کلاسیک از الگوریتم Off-Policy است. در این الگوریتم، هدف یافتن تابع Q بهینه است که مقدار مورد انتظار پاداش آینده را با توجه به یک عمل خاص در یک حالت خاص برآورد می‌کند [۱۰].

[۱] Deterministic

از **DQN** تا **DPG**

DQN الگوریتمی است که با استفاده از شبکه‌های عصبی، مقدار توابع Q را برای هر جفت حالت_عمل تقریب می‌زند. این روش در فضای حالت‌های پیوسته بسیار موفق بوده است، اما محدودیت‌هایی نیز دارد. یکی از این محدودیت‌ها این است که DQN برای فضای عمل‌های گسسته طراحی شده است. برای غلبه بر این محدودیت، DDPG طراحی شد تا بتواند در فضای عمل‌های پیوسته کار کند [۶].

چالش‌های **DQN** در فضای عمل‌های پیوسته

در DQN، تابع Q مقدار مورد انتظار پاداش را برای هر جفت حالت_عمل برآورد می‌کند. این کار برای فضای عمل‌های گسسته به خوبی انجام می‌شود زیرا می‌توان برای هر عمل ممکن مقدار Q را محاسبه و مقایسه کرد. اما در فضای عمل‌های پیوسته، تعداد عمل‌های ممکن بی‌نهایت است و نمی‌توان برای هر عمل مقدار Q را محاسبه کرد. این مشکل باعث شد تا نیاز به یک الگوریتم جدید برای فضای عمل‌های پیوسته احساس شود.

برای حل این مشکل، محققان ابتدا تلاش کردند تا Q-Learning را با روش‌های تقریب تابع ترکیب کنند. اما این روش‌ها نیز به دلیل پیچیدگی فضای عمل‌های پیوسته و نیاز به محاسبه گرادیان‌های دقیق، کارایی لازم را نداشتند.

الگوریتم DDPG به عنوان یک ترکیب از DQN و DPG متولد شد. این الگوریتم از شبکه‌های عصبی عمیق برای تقریب توابع سیاست و ارزش استفاده می‌کند و از گرادیان‌های قطعی برای به‌روزرسانی سیاست بهره می‌برد. DDPG از دو شبکه عصبی اصلی به نام‌های شبکه بازیگر [۱] و شبکه منتقد [۲] استفاده می‌کند. شبکه بازیگر وظیفه تولید عمل‌های بهینه برای هر حالت را دارد و شبکه منتقد مقدار Q مربوط به هر جفت حالت_عمل را تخمین می‌زند.

فرمول‌بندی **DDPG** به عنوان **MDP**

الگوریتم DDPG را می‌توان به صورت یک فرآیند تصمیم‌گیری مارکوف (MDP) فرمول‌بندی کرد. در اینجا، اجزای مختلف MDP را در زمینه DDPG بررسی می‌کنیم:

[۱] Actor
[۲] Critic

- مجموعه حالات $\mathcal{S}$: مجموعه حالات شامل همه حالات ممکن سیستم است که معمولاً توسط یک بردار ویژگی نمایش داده می‌شود. برای مثال، در یک سیستم رباتیکی، حالت‌ها می‌توانند موقعیت و سرعت اتصالات ربات باشند.

- مجموعه اعمال $\mathcal{A}$: مجموعه اعمال شامل همه اعمال ممکن است که سیستم می‌تواند انجام دهد. در DDPG، این اعمال پیوسته هستند. برای مثال، در یک سیستم رباتیکی، اعمال می‌توانند گشتاورهای اعمال‌شده به اتصالات ربات باشند.

- تابع انتقال $P(s'|s,a)$: تابع انتقال احتمال انتقال سیستم از حالت s به حالت s' با انجام عمل a را مشخص می‌کند. در عمل، این تابع معمولاً به صورت ضمنی در محیط مدل‌سازی می‌شود.

- تابع پاداش $R(s,a)$: تابع پاداش میزان پاداش دریافتی پس از انجام عمل a در حالت s را تعیین می‌کند. هدف DDPG حداکثر کردن پاداش تجمعی تخفیف‌یافته است.

- عامل تخفیف γ: عامل تخفیف میزان اهمیت پاداش‌های آینده نسبت به پاداش‌های کنونی را تعیین می‌کند. این عامل بین ۰ و ۱ قرار دارد.

اجزای کلیدی **DDPG**

۱. شبکه بازیگر:

- وظیفه: نگاشت حالات به عمل‌ها و تعیین عمل بهینه برای هر حالت.
- ساختار: یک شبکه عصبی کاملاً متصل که خروجی‌های پیوسته را برای اعمال فراهم می‌کند.

۲. شبکه منتقد:

- وظیفه: ارزیابی عملکرد بازیگر از طریق تخمین مقدار Q، که نمایانگر پاداش تجمعی مورد انتظار است.
- ساختار: یک شبکه عصبی کاملاً متصل که هم حالت و هم عمل را به عنوان ورودی دریافت کرده و یک مقدار اسکالر (مقدار Q) را خروجی می‌دهد.

۳. شبکه‌های هدف:

- هدف: فراهم کردن اهداف پایدار برای آموزش شبکه‌های بازیگر و منتقد.
- روش به‌روزرسانی: این شبکه‌ها نسخه‌هایی از شبکه‌های بازیگر و منتقد هستند که به آرامی به‌روزرسانی می‌شوند تا ثبات آموزش حفظ شود.

۴. بافر بازپخش تجربه:

- هدف: ذخیره تجربیات گذشته شامل حالات، اعمال، پاداش‌ها، و حالات بعدی برای کاهش همبستگی بین به‌روزرسانی‌های متوالی و بهبود کارایی یادگیری.
- نحوه استفاده: مینی بچ‌های تصادفی از این بافر استخراج شده و برای آموزش استفاده می‌شوند.

فرایند **DDPG**

Algorithm 1 DDPG Process

Initialize actor network $\mu(s|\theta^{\mu})$ and critic network $Q(s,a|\theta^{Q})$ with random weights

Initialize target networks μ' and Q' with weights $\theta^{\mu'} \leftarrow \theta^{\mu}$ and $\theta^{Q'} \leftarrow \theta^{Q}$

Initialize experience replay buffer D

Set soft update parameter τ

for each episode **do**

  Initialize a random process $\mathcal{N}$ for action exploration

  Receive initial state s_0

  for each time step t **do**

    Select action $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$

    Execute action a_t and observe reward r_t and next state s_{t+1}

    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer D

    if replay buffer D has enough samples **then**

      Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from D

      Compute target Q-value:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$$

      Update critic network by minimizing the loss:

$$L(\theta^{Q}) = \frac{1}{N}\sum_{i}\left(Q(s_i, a_i|\theta^{Q}) - y_i\right)^2$$

      Update actor network using the sampled policy gradient:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{i} \nabla_a Q(s,a|\theta^{Q})\Big|_{a=\mu(s|\theta^{\mu})} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})$$

      Perform soft updates for target networks:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

    end if

  end for

end for

فرمول‌بندی ریاضی

۱. به‌روزرسانی شبکه بازیگر: شبکه بازیگر $\mu(s|\theta^\mu)$ با حداکثر کردن بازگشت مورد انتظار با استفاده از گرادیان سیاست به‌روزرسانی می‌شود:

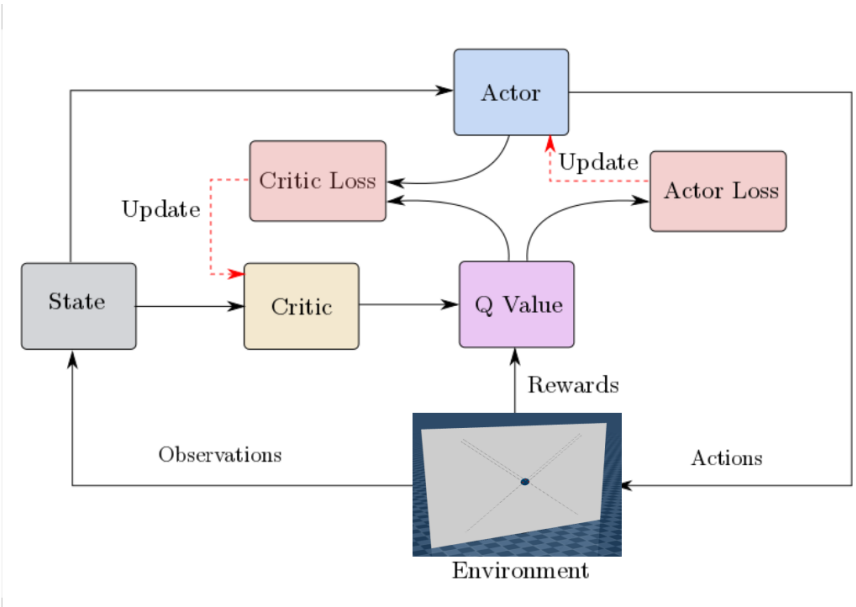
$$\theta^\mu \leftarrow \theta^\mu + \alpha \nabla_{\theta^\mu} J(\theta^\mu)$$

۲. به‌روزرسانی شبکه منتقد: شبکه منتقد $Q(s,a|\theta^Q)$ با به حداقل رساندن تابع ضرر به‌روزرسانی می‌شود:

$$L(\theta^Q) = E_{s_t,a_t,r_t,s_{t+1}\sim D}\left[\left(Q(s_t,a_t|\theta^Q) - y_t\right)^2\right]$$

مقدار Q هدف y_t به صورت زیر محاسبه می‌شود:

$$y_t = r_t + \gamma \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}|\theta^{Q'})$$



شکل ۱.۳: دیاگرام الگوریتم DDPG

جزئیات پیاده‌سازی

معماری‌های شبکه:

شبکه بازیگر: به طور معمول شامل چندین لایه کاملاً متصل با توابع فعال‌سازی غیرخطی (مثلاً ReLU) است. لایه خروجی از یک تابع فعال‌سازی مانند تانژانت هایپربولیک (tanh) استفاده می‌کند تا عمل در محدوده معتبر باشد.

شبکه منتقد: به طور مشابه، شامل لایه‌های کاملاً متصل با توابع فعال‌سازی غیرخطی است. لایه خروجی یک مقدار اسکالر منفرد را که نمایانگر مقدار Q است، خروجی می‌دهد. مثال شبکه بازیگر:

$$a_t = \tanh(W_2 \cdot \mathrm{ReLU}(W_1 \cdot s_t + b_1) + b_2)$$

مثال شبکه منتقد:

$$Q(s_t, a_t) = W_2 \cdot \mathrm{ReLU}(W_1 \cdot [s_t, a_t] + b_1) + b_2$$

فراپارامترها:

نرخ‌های یادگیری: نرخ‌های یادگیری جداگانه برای شبکه‌های بازیگر و منتقد.

عامل تخفیف γ: اهمیت پاداش‌های آینده را تعیین می‌کند.

پارامتر به‌روزرسانی نرم τ: سرعت به‌روزرسانی شبکه‌های هدف را کنترل می‌کند.

اندازه دسته: تعداد نمونه‌های استفاده‌شده برای هر به‌روزرسانی[1].

نویز اکتشافی:

افزودن نویز به فرایند انتخاب عمل، اکتشاف فضای حالت–عمل را تشویق می‌کند. یک انتخاب رایج فرایند اورنشتاین–اولن‌بک است که نویز همبسته زمانی اضافه می‌کند.

[1] size Batch

حلقه آموزشی:

به طور مداوم با محیط تعامل کنید، تجربیات را ذخیره کنید و شبکه‌ها را با استفاده از مینی بچ‌های نمونه‌برداری‌شده به‌روزرسانی کنید. به‌روزرسانی‌های نرم شبکه‌های هدف را به‌طور دوره‌ای انجام دهید.

مزایای **DDPG**

چالش‌ها و جهت‌گیری‌های آینده

کارایی نمونه: آموزش DDPG نیاز به تعداد زیادی تعامل با محیط دارد. بهبود کارایی نمونه از طریق تکنیک‌هایی مانند بازپخش تجربه اولویت‌بندی‌شده و RL مبتنی بر مدل یک حوزه تحقیق فعال است.

حساسیت فراپارامترها: عملکرد DDPG به انتخاب فراپارامترها حساس است. روش‌های تنظیم خودکار فراپارامتر می‌تواند به بهینه‌سازی عملکرد کمک کند.

ایمنی و استحکام: اطمینان از عملکرد ایمن و مستحکم در برنامه‌های دنیای واقعی حیاتی است. ترکیب محدودیت‌های ایمنی و روش‌های آموزشی مستحکم می‌تواند قابلیت اطمینان سیستم‌های کنترل‌شده توسط DDPG را بهبود بخشد.

با پرداختن به این چالش‌ها، DDPG می‌تواند برای کنترل CDPRها بیشتر بهینه شود و به سیستم‌های رباتیکی کارآمدتر و مؤثرتری منجر شود.

فصل ۴

کنترل ربات کابلی با استفاده از یادگیری تقویتی

۱.۴ استفاده از RL برای کنترل CDPR

استفاده از Reinforcement Learning (RL) برای کنترل Cable-Driven Parallel Robot (CDPR) به دلیل توانایی آن در مدیریت سیستم‌های پیچیده و غیرخطی، روشی بسیار مؤثر است. RL یک روش مبتنی بر یادگیری است که به عامل (ربات) اجازه می‌دهد تا از طریق تعامل با محیط، سیاست بهینه را برای رسیدن به اهداف مشخص یاد بگیرد. این روش بدون نیاز به مدل دقیق سیستم، با استفاده از داده‌های تجربی بهینه‌سازی می‌شود. از دلایل استفاده از RL برای کنترل CDPR می‌توان به مدیریت پیچیدگی و غیرخطی بودن، عدم نیاز به مدل دقیق، توانایی انطباق با تغییرات محیطی و کاربردهای متعدد اشاره کرد. تحقیقات متعددی در این زمینه انجام شده است. به عنوان مثال، در تحقیقی توسط Zhao et al. از Deep Q-Network (DQN) برای کنترل یک CDPR استفاده شده است که نتایج نشان می‌دهد روش DQN توانسته است مسیرهای پیچیده را با دقت بالا دنبال کند [۱۱]. همچنین، Yang et al. استفاده از Proximal Policy Optimization (PPO) را برای کنترل ربات‌های CDPR بررسی کرده‌اند که نتایج نشان می‌دهد PPO عملکرد بهتری نسبت به روش‌های سنتی داشته است [۱۲]. در تحقیقی دیگر، Ma et al. از ترکیب روش‌های RL و کنترل کلاسیک برای بهبود عملکرد CDPR استفاده کرده‌اند که این ترکیب توانسته است مزایای هر دو روش را به کار بگیرد و عملکرد سیستم را بهبود بخشد [۱۳]. این تحقیقات نشان می‌دهند که استفاده از RL برای

کنترل ربات‌های CDPR می‌تواند عملکرد بهتری نسبت به روش‌های کنترلی سنتی داشته باشد و قادر است با پیچیدگی‌ها و تغییرات محیطی بهتر سازگاری پیدا کند.

استفاده از الگوریتم DDPG برای کنترل CDPR به دلیل توانایی آن در مدیریت فضای عمل پیوسته، روشی بسیار مؤثر است. DDPG یک الگوریتم actor-critic است که به صورت خارج از سیاست عمل می‌کند و بدون نیاز به مدل سیستم، به یادگیری می‌پردازد. این ویژگی‌ها، آن را به گزینه‌ای مناسب برای کنترل ربات‌های پیچیده مانند CDPR تبدیل می‌کند.

در این قسمت به بررسی دو وظیفه کنترلی می‌پردازیم و سپس مراحل پیاده‌سازی را با استفاده از روش‌های یادگیری تقویتی توضیح خواهیم داد.

۲.۴ کنترل نقطه به نقطه

کنترل نقطه به نقطه (Point-to-Point Control) یکی از وظایف اساسی در رباتیک است که شامل حرکت دادن مجری نهایی ربات از یک نقطه شروع به یک نقطه هدف می‌باشد. هدف از کنترل نقطه به نقطه این است که ربات بتواند با دقت بالا و در کوتاه‌ترین زمان ممکن، از نقطه شروع به نقطه هدف برسد. این مسئله نیازمند برنامه‌ریزی دقیق مسیر و اعمال کنترل‌های مناسب بر روی موتورها و کابل‌های ربات است تا حرکت به نرمی و بدون خطا انجام شود.

۳.۴ ردیابی مسیر مشخص

ردیابی مسیر مشخص یکی از وظایف مهم در سیستم‌های رباتیک است که شامل دنبال کردن یک مسیر پیش‌تعیین شده توسط مجری نهایی ربات می‌باشد. هدف از ردیابی مسیر این است که ربات بتواند با دقت و صحت بالا، مسیر مشخص شده را طی کند و به نقاط تعیین شده در زمان‌های مشخص برسد. این مسئله نیازمند کنترل دقیق و هماهنگی بین اجزای مختلف ربات است تا خطاهای موقعیت و سرعت به حداقل برسد. استفاده از الگوریتم‌های یادگیری تقویتی در این زمینه می‌تواند به بهبود عملکرد ربات در ردیابی مسیر کمک کند، زیرا این الگوریتم‌ها قادرند به طور خودکار سیاست‌های کنترلی بهینه را یاد بگیرند و به تغییرات محیط و دینامیک سیستم پاسخ دهند. ردیابی مسیر مشخص به ویژه

در کاربردهای صنعتی، پزشکی و خدماتی که نیاز به دقت بالا دارند، اهمیت ویژه‌ای دارد.

۱.۳.۴ توضیح مسئله (ردیابی مسیر دایره‌ای) و چگونگی ارائه این مسیر

مسئله‌ی ردیابی مسیر برای CDPR شامل دنبال کردن یک مسیر دایره‌ای از پیش تعیین شده توسط مجری نهایی[۱] ربات است. این مسیر به صورت نقاط مختلف در فضای دوبعدی تعریف می‌شود که مجری نهایی باید آنها را دنبال کند. هدف این است که ربات به طور دقیق و با حداقل خطا این مسیر را طی کند. مسیر دایره‌ای با استفاده از فرمول‌های ریاضی زیر تعریف می‌شود:

$$x(t) = R\cos(\omega t)$$

$$y(t) = y_0$$

$$z(t) = R\sin(\omega t)$$

در این معادلات، R شعاع دایره، ω فرکانس زاویه‌ای، و y_0 ثابت است. این مسیر دایره‌ای در طول زمان به صورت پیوسته تولید می‌شود و ربات باید آن را دنبال کند.

سرعت مسیر

علاوه بر موقعیت، سرعت مجری نهایی ربات نیز برای ردیابی دقیق مسیر اهمیت دارد. سرعت مسیر دایره‌ای می‌تواند از مشتق معادلات موقعیت نسبت به زمان به دست آید. معادلات سرعت به صورت زیر تعریف می‌شوند:

$$\dot{x}(t) = -R\omega\sin(\omega t)$$

$$\dot{y}(t) = 0$$

$$\dot{z}(t) = R\omega\cos(\omega t)$$

[۱] end-effector

در این معادلات، $\dot{x}(t)$ و $\dot{z}(t)$ سرعت‌های مؤلفه‌های x و z هستند و $\dot{y}(t)$ به دلیل ثابت بودن مؤلفه y برابر صفر است.

برای ربات، کنترل دقیق سرعت به همان اندازه کنترل موقعیت اهمیت دارد. بنابراین، هدف نهایی این است که نه تنها مجری نهایی ربات موقعیت‌های تعریف شده را دنبال کند، بلکه با سرعت مناسب نیز حرکت کند تا پایداری و دقت ردیابی مسیر بهبود یابد.

طراحی تابع پاداش

تابع پاداش در الگوریتم RL باید به گونه‌ای طراحی شود که خطاهای موقعیت و سرعت را جریمه کند تا مجری نهایی ربات بتواند مسیر دایره‌ای را به خوبی دنبال کند. تابع پاداش پیشنهادی به صورت زیر است:

$$r(s,a) = -\alpha \cdot \|d(s,s^*)\|_۲ - \beta \cdot |V(s)\|_۲$$

در این معادله:

- $\|d(s,s^*)\|_۲$ فاصله بین حالت فعلی و حالت هدف است که به صورت نُرم دوم محاسبه می‌شود.
- $\|V(s)\|_۲$ خطای سرعت است که به صورت نُرم دوم محاسبه می‌شود.
- α و β ضرایب وزن‌دهی هستند که اهمیت نسبی هر جزء خطا را تعیین می‌کنند.

این فرمول‌بندی تضمین می‌کند که ربات نه تنها به موقعیت هدف برسد، بلکه با سرعت مناسب و کمترین تنش کابل‌ها این کار را انجام دهد.

۴.۴ فرموله کردن مسئله در زمینه RL

مسئله‌ی ردیابی مسیر دایره‌ای به عنوان یک Markov Decision Process (MDP) تعریف می‌شود که شامل حالت‌ها، اعمال، پاداش‌ها و احتمالات انتقال است.

- حالت‌ها: شامل موقعیت و سرعت مجری نهایی، خطای موقعیت، خطای سرعت و طول کابل‌ها می‌باشد. حالت‌ها به صورت بردارهای ویژگی توصیف می‌شوند که وضعیت فعلی سیستم را نشان می‌دهند.

$$\mathbf{s} = \begin{bmatrix} \mathbf{p}_{\text{eff}} & \mathbf{v}_{\text{eff}} & \mathbf{e}_{\text{pos}} & \mathbf{e}_{\text{vel}} & \mathbf{l}_{\text{cable}} \end{bmatrix}^\top$$

  - $\mathbf{p}_{\text{eff}} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}$: موقعیت مجری نهایی در ابعاد x، y، و z.
  - $\mathbf{v}_{\text{eff}} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$: سرعت مجری نهایی در ابعاد x، y، و z.
  - $\mathbf{e}_{\text{pos}} = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}$: خطای موقعیت در ابعاد x، y، و z.
  - $\mathbf{e}_{\text{vel}} = \begin{bmatrix} \dot{e}_x & \dot{e}_y & \dot{e}_z \end{bmatrix}$: خطای سرعت در ابعاد x، y، و z.
  - $\mathbf{l}_{\text{cable}} = \begin{bmatrix} l_۱ & l_۲ & l_۳ \end{bmatrix}$: طول کابل‌ها.

- اعمال: ورودی‌های کنترلی به موتورها برای تغییر طول کابل‌ها. اعمال به صورت بردارهایی از ورودی‌های کنترلی تعریف می‌شوند که سیاست بهینه باید آنها را تولید کند.

$$\mathbf{a} = \begin{bmatrix} \Delta l_۱ & \Delta l_۲ & \Delta l_۳ \end{bmatrix}^\top$$

  - $\Delta l_۱, \Delta l_۲, \Delta l_۳$: تغییرات طول کابل‌ها که به عنوان ورودی‌های کنترلی برای موتورها استفاده می‌شوند.

- پاداش‌ها: تابع پاداش برای هدایت فرایند یادگیری، متعادل‌سازی بین ردیابی مسیر و کمینه کردن خطای سرعت برای هموار کردن آن. تابع پاداش به گونه‌ای طراحی می‌شود که خطای ردیابی و سرعت بیش از حد را جریمه کند.

- انتقالات: تغییرات در حالت‌ها بر اساس دینامیک سیستم و اعمال انجام شده. انتقالات نشان‌دهنده‌ی تغییرات وضعیت سیستم در پاسخ به اعمال انجام شده توسط سیاست هستند.

۱.۴.۴ مراحل پیاده‌سازی

۱. راه‌اندازی محیط

- محیط MuJoCo برای شبیه‌سازی CDPR تنظیم می‌شود.

۲. تعریف معماری شبکه

- شبکه Actor: یک شبکه عصبی که اعمال پیوسته را برای یک حالت مشخص تولید می‌کند.
- شبکه Critic: یک شبکه عصبی که مقدار Q-value را برای جفت حالت_عمل مشخص ارزیابی می‌کند.

۳. فرایند آموزش

- شبکه‌های actor و critic به همراه شبکه‌های هدف مقداردهی اولیه می‌شوند.
- برای هر اپیزود:
  - محیط بازنشانی شده و حالت اولیه مشاهده می‌شود.
  - برای هر مرحله در اپیزود:
    * یک عمل با استفاده از شبکه actor انتخاب می‌شود و نویز اکتشافی اضافه می‌شود.
    * عمل در محیط اجرا شده و حالت بعدی و پاداش مشاهده می‌شود.
    * انتقال در replay buffer ذخیره می‌شود.
    * یک مینی‌بچ از انتقال‌ها از replay buffer نمونه‌گیری می‌شود.
    * شبکه‌ی critic با کمینه کردن تفاوت بین Q-value پیش‌بینی شده و Q-value هدف به‌روزرسانی می‌شود.
    * شبکه‌ی actor با استفاده از شیب سیاست نمونه‌گیری شده به‌روزرسانی می‌شود.
    * شبکه‌های هدف به‌روزرسانی می‌شوند.

۴. آزمایش و ارزیابی

- پس از آموزش، مدل در محیط شبیه‌سازی آزمایش می‌شود تا عملکرد آن ارزیابی شود.
- ردیابی مسیر و خطاهای موقعیت و سرعت اندازه‌گیری می‌شوند.
- نتایج با معیارهای از پیش تعیین شده مقایسه می‌شوند.

۲.۴.۴ اجزای DDPG

- شبکه‌ی سیاست: این شبکه وظیفه تولید اعمال بهینه بر اساس حالت فعلی را دارد.
- شبکه‌ی Q: این شبکه ارزش اعمال تولید شده توسط شبکه‌ی سیاست را ارزیابی می‌کند.
- ذخیره‌سازی تجارب: تجارب جمع‌آوری شده در طول فرایند یادگیری در یک حافظه بازپخش ذخیره می‌شوند و برای به‌روزرسانی شبکه‌ها استفاده می‌شوند.
- به‌روزرسانی شبکه‌ها: به‌روزرسانی پارامترهای شبکه‌ها با استفاده از الگوریتم Backpropagation و بهینه‌سازی Gradient Descent.

۳.۴.۴ به‌روزرسانی شبکه‌ها

شبکه‌ی سیاست با استفاده از بهینه‌سازی Policy Gradient به‌روزرسانی می‌شود. هدف این بهینه‌سازی کمینه کردن تابع خطا است که به صورت زیر تعریف می‌شود:

$$L(\theta^{\mu}) = E_{s\sim\mathcal{D}}\left[-Q(s,\mu(s|\theta^{\mu}))\right]$$

شبکه‌ی Q با استفاده از بهینه‌سازی Q-Learning به‌روزرسانی می‌شود. هدف این بهینه‌سازی کمینه کردن تابع خطا است که به صورت زیر تعریف می‌شود:

$$L(\theta^{Q}) = E_{(s,a,r,s')\sim\mathcal{D}}\left[(Q(s,a|\theta^{Q}) - y)^{2}\right]$$

که در آن $y = r + \gamma Q'(s', \mu'(s'|\theta^{\mu'})|\theta^{Q'})$ و γ ضریب تخفیف است.

لایه	نوع	جزئیات
۰	Linear	bias=True ،output=400 ،input=11
۱	ReLU	-
۲	Linear	bias=True ،output=300 ،input=400
۳	ReLU	-
۴	Linear	bias=True ،output=3 ،input=300
۵	Tanh	-

جدول ۱.۴: لایه‌های شبکه Actor

لایه	نوع	جزئیات
۰	Linear	bias=True ،output=400 ،input=14
۱	ReLU	-
۲	Linear	bias=True ،output=300 ،input=400
۳	ReLU	-
۴	Linear	bias=True ،output=1 ،input=300

جدول ۲.۴: لایه‌های شبکه Critic

ساختار شبکه و پارامترها

- اندازه دسته: این پارامتر تعیین می‌کند که چند نمونه در هر دوره آموزشی استفاده می‌شود. اندازه دسته 256 به تعادل بین پایداری فرایند آموزش و کارایی محاسباتی کمک می‌کند.
- گاما (γ): ضریب تخفیف برای پاداش‌های آینده، که به 0.99 تنظیم شده است. این مقدار بالا تضمین می‌کند که عامل پاداش‌های بلندمدت را در نظر بگیرد و عملکرد پایدار در طولانی‌مدت داشته باشد.

پارامتر	مقدار
اندازه دسته	256
گاما	0.99
تاو	0.005
اندازه بافر	1000000
نرخ یادگیری بازیگر	0.001
نرخ یادگیری منتقد	0.001

جدول ۳.۴: پارامترهای مدل DDPG

- تاو (τ): پارامتر برای به‌روزرسانی نرم شبکه هدف، که به 0.005 تنظیم شده است. این مقدار کوچک به به‌روزرسانی‌های پایدار و تدریجی کمک می‌کند و از تغییرات ناگهانی در شبکه هدف جلوگیری می‌کند که می‌تواند آموزش را ناپایدار کند.

- اندازه بافر: اندازه بافر بازپخش، که به 1000000 تنظیم شده است. اندازه بزرگ بافر به عامل امکان می‌دهد که از مجموعه متنوعی از تجربیات یاد بگیرد و مقاومت و تعمیم‌پذیری سیاست را بهبود بخشد.

- نرخ یادگیری بازیگر: نرخ یادگیری برای شبکه بازیگر، که به 0.001 تنظیم شده است. این نرخ تعیین می‌کند که پارامترهای شبکه با چه سرعتی به‌روزرسانی شوند، اطمینان از پیشرفت پایدار به سمت سیاست بهینه بدون پرش بیش از حد.

- نرخ یادگیری منتقد: نرخ یادگیری برای شبکه منتقد، که همچنین به 0.001 تنظیم شده است. مشابه نرخ یادگیری بازیگر، این نرخ تضمین می‌کند که تقریب‌های تابع ارزش به طور پیوسته و پایدار بهبود یابد.

انتخاب این پارامترها بر اساس تجربیات پیشین و نیازهای خاص کنترل ربات CDPR انجام شده است. در شبکه actor، لایه‌های Linear با تعداد زیادی از ویژگی‌ها به کار گرفته شده‌اند تا نمایش‌های پیچیده‌ای از حالت را به دست آورند. استفاده از توابع ReLU به دلیل غیرخطی بودن آنهاست که اجازه می‌دهد شبکه توابع پیچیده‌ای را یاد بگیرد.

همچنین، استفاده از تابع Tanh در خروجی به دلیل مقیاس‌بندی خروجی‌ها بین ۱- و ۱ است که برای فضای عمل پیوسته مناسب است.

در شبکه critic، ساختار مشابهی با شبکه actor استفاده شده است، اما با ورودی‌هایی که شامل جفت‌های حالت_عمل است. هدف این شبکه ارزیابی کیفیت اعمال انجام شده توسط شبکه actor است.

نرخ یادگیری 0.001 انتخاب شده است تا همگرایی پایدار بدون فراجهش را تضمین کند. این نرخ یادگیری متعادل به شبکه اجازه می‌دهد تا به آرامی به سمت یک راه‌حل بهینه همگرا شود.

۵.۴ پیاده‌سازی با استفاده از Stable Baselines3

Stable Baselines3 یک کتابخانه‌ی Python است که پیاده‌سازی‌های الگوریتم‌های یادگیری تقویتی مدرن را فراهم می‌کند. این کتابخانه برای آموزش و ارزیابی مدل‌های یادگیری تقویتی بسیار مناسب است.

برای پیاده‌سازی DDPG از این کتابخانه، ابتدا باید محیط MuJoCo را تعریف کرده و سپس الگوریتم DDPG را با تنظیمات مناسب راه‌اندازی کنیم. در نهایت، مدل آموزش داده شده و سیاست بهینه برای کنترل CDPR تولید می‌شود.

در طول فرایند آموزش، مدل با استفاده از تجارب جمع‌آوری شده، پارامترهای خود را به‌روزرسانی می‌کند تا سیاست بهینه برای ردیابی مسیر دایره‌ای تولید شود. این فرایند تا زمانی ادامه می‌یابد که مدل به عملکرد مطلوب برسد.

۱.۵.۴ استفاده از محیط MuJoCo برای پیاده‌سازی

محیط MuJoCo یک شبیه‌ساز دینامیک چندبدنه‌ای با دقت بالا است که برای شبیه‌سازی و کنترل سیستم‌های رباتیک پیچیده استفاده می‌شود. در این بخش، ما از محیط MuJoCo برای شبیه‌سازی و کنترل یک ربات CDPR استفاده می‌کنیم. هدف از این شبیه‌سازی، پیاده‌سازی یک کنترل‌کننده مبتنی بر RL است که بتواند مسیر دایره‌ای مشخصی را دنبال کند.

۲.۵.۴ تعریف کلاس CableControlEnv

کلاس CableControlEnv یک محیط شبیه‌سازی شده در MuJoCo است که برای کنترل ربات CDPR طراحی شده است. این کلاس از MujocoEnv و EzPickle ارث‌بری می‌کند که به ترتیب برای تعامل با محیط MuJoCo و ذخیره‌سازی محیط استفاده می‌شوند.

این محیط شامل پارامترهایی مانند تعداد نقاط مسیر دایره‌ای، تعداد فریم‌های شبیه‌سازی و مشخصات محیط شبیه‌سازی است. همچنین، فضاهای مشاهده و اعمال تعریف شده‌اند که شامل مقادیر موقعیت و سرعت مجری نهایی، طول کابل‌ها و تنش‌های کابل‌ها می‌باشد.

۳.۵.۴ تولید مسیر هدف

در این محیط، تابعی برای تولید موقعیت‌های هدف در طول مسیر دایره‌ای تعریف شده است. این تابع با استفاده از معادلات دایره‌ای، موقعیت‌های مختلف هدف را تولید می‌کند. مسیر دایره‌ای به صورت پیوسته تولید می‌شود و شامل پارامترهایی مانند شعاع دایره، مرکز دایره و زاویه حرکت می‌باشد.

۴.۵.۴ تولید سرعت مسیر هدف

تابعی نیز برای محاسبه سرعت‌های هدف در طول مسیر دایره‌ای تعریف شده است. این تابع مشتقات معادلات موقعیت را برای محاسبه سرعت در هر لحظه استفاده می‌کند. این سرعت‌ها برای محاسبه خطای سرعت و بهینه‌سازی عملکرد ربات استفاده می‌شوند.

۵.۵.۴ اجرای قدم‌های شبیه‌سازی

در هر قدم شبیه‌سازی، ابتدا عمل شبیه‌سازی اجرا شده و سپس موقعیت و سرعت مجری نهایی ربات به روزرسانی می‌شود. همچنین، اگر مجری نهایی ربات به هدف نهایی رسیده باشد، مسیر دایره‌ای به پایان می‌رسد و مجری نهایی در نقطه پایانی ثابت می‌ماند. در این حالت، موقعیت و سرعت مجری نهایی به روزرسانی می‌شود و اگر شرایط پایان شبیه‌سازی برآورده شود، شبیه‌سازی به پایان می‌رسد.

۶.۵.۴ بازنشانی مدل

برای بازنشانی وضعیت ربات به حالت اولیه، تابعی تعریف شده است که مجری نهایی ربات را به موقعیت اولیه بازمی‌گرداند و پارامترهای مسیر دایره‌ای را نیز به حالت اولیه تنظیم می‌کند. این تابع تضمین می‌کند که ربات از وضعیت اولیه شروع به حرکت کرده و مسیر دایره‌ای را دنبال می‌کند.

۷.۵.۴ محاسبه مشاهده‌ها

در این محیط، مشاهده‌ها شامل موقعیت و سرعت مجری نهایی ربات، طول کابل‌ها و خطاهای موقعیت و سرعت است. این مشاهده‌ها در هر قدم شبیه‌سازی جمع‌آوری شده و برای به روزرسانی مدل و محاسبه پاداش‌ها استفاده می‌شوند. مشاهده‌ها به صورت بردارهایی از ویژگی‌ها تعریف می‌شوند که وضعیت فعلی سیستم را نشان می‌دهند.

۸.۵.۴ محاسبه پاداش

تابعی برای محاسبه پاداش هر قدم شبیه‌سازی تعریف شده است. این پاداش بر اساس خطاهای موقعیت و سرعت محاسبه می‌شود تا ربات تشویق شود که به مسیر هدف نزدیک شود و با سرعت مناسب حرکت کند. تابع پاداش به گونه‌ای طراحی شده است که سیستم را به سمت کمینه کردن خطای ردیابی و کاهش تنش‌های کابل هدایت کند.

۹.۵.۴ شرایط پایان

برای بررسی شرایط پایان شبیه‌سازی، تابعی تعریف شده است که بررسی می‌کند آیا مجری نهایی ربات به موقعیت هدف رسیده است یا خیر. اگر مجری نهایی به موقعیت هدف برسد و خطای سرعت نیز کمتر از حد مشخصی باشد، شبیه‌سازی پایان می‌یابد. این شرایط تضمین می‌کند که ربات به طور دقیق و با حداقل خطا مسیر دایره‌ای را دنبال کند.

۱۰.۵.۴ شرایط قطع شبیه‌سازی

برای بررسی شرایط قطع شبیه‌سازی، تابعی تعریف شده است که بررسی می‌کند آیا تعداد قدم‌های شبیه‌سازی از حداکثر مقدار مشخص شده بیشتر شده است یا خیر. اگر این شرایط برآورده شود، شبیه‌سازی قطع می‌شود. این شرایط

تضمین می‌کند که شبیه‌سازی در صورت عدم رسیدن به هدف در مدت زمان معقولی قطع شود.

پیاده‌سازی DDPG برای کنترل CDPR نشان‌دهنده‌ی توانایی الگوریتم‌های RL در مدیریت سیستم‌های دینامیکی پیچیده است. استفاده از محیط MuJoCo و کتابخانه‌ی stable-baselines3 فرایند پیاده‌سازی را تسهیل می‌کند و امکان آزمایش و ارزیابی دقیق را فراهم می‌آورد.

فصل ۵

نتایج و بحث

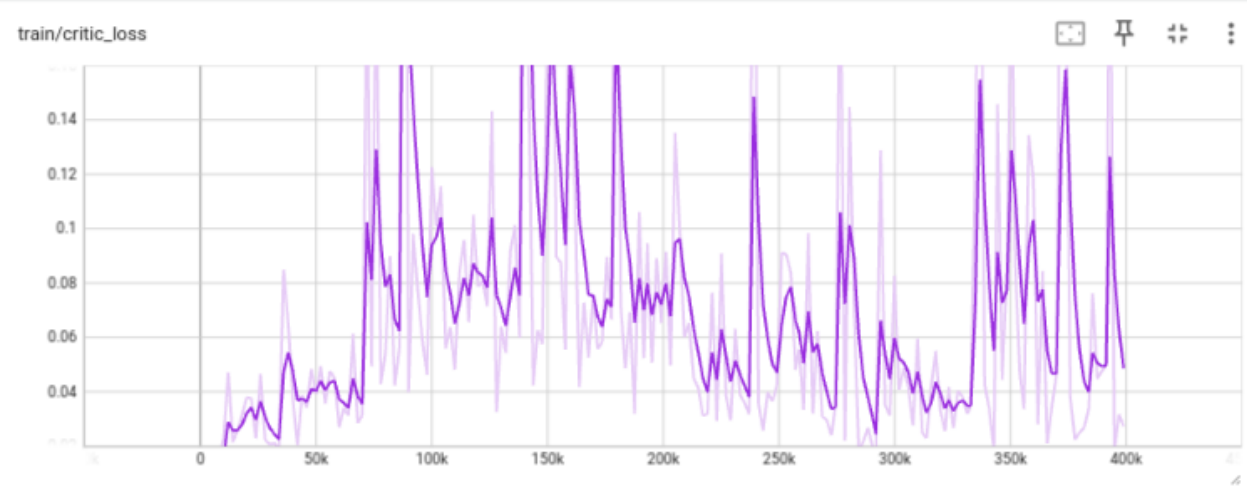
در این فصل، نتایج به دست آمده از پیاده‌سازی کنترل‌کننده‌های یادگیری تقویتی برای ربات CDPR بررسی و مورد بحث قرار می‌گیرند. این نتایج شامل ارزیابی عملکرد ربات در دو وظیفه مختلف، ردیابی مسیر دایره‌ای و کنترل نقطه به نقطه، می‌باشند. همچنین تحلیل‌های آماری خطاهای موقعیت و سرعت نیز ارائه شده‌اند.

۱.۵ نتایج

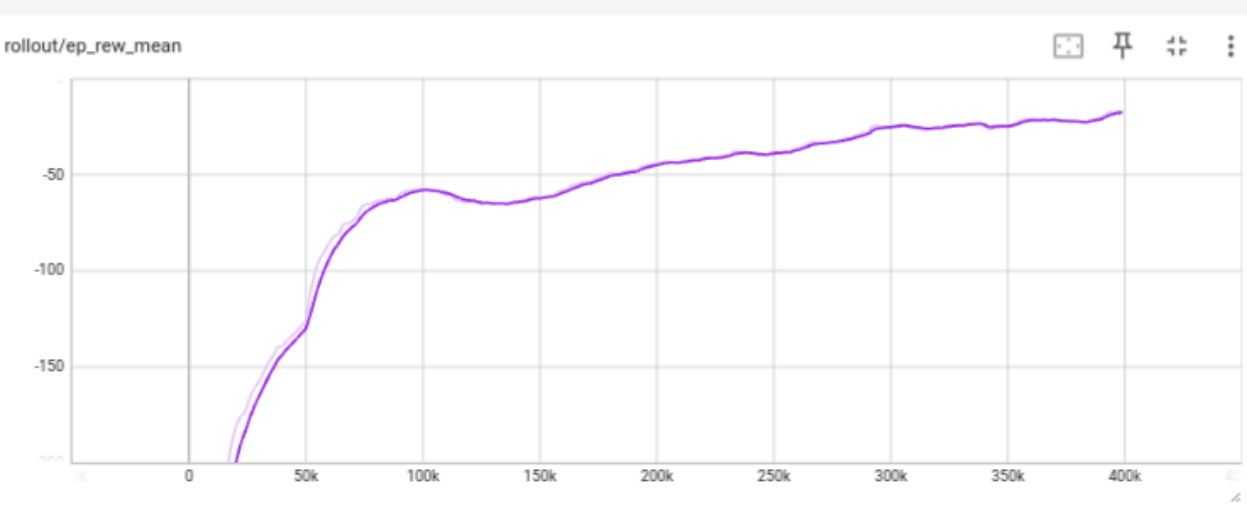
۱.۱.۵ نمودارهای آموزش کنترل نقطه به نقطه

نمودارهای زیر نشان‌دهنده روند کاهش خطای شبکه Actor و شبکه Critic و همچنین میانگین پاداش اپیزودیک در طول آموزش مدل DDPG برای کنترل نقطه به نقطه هستند.

شکل ۱.۵: نمودار خطای شبکه Actor در طول آموزش برای کنترل نقطه به نقطه



شکل ۲.۵: نمودار خطای شبکه Critic در طول آموزش برای کنترل نقطه به نقطه



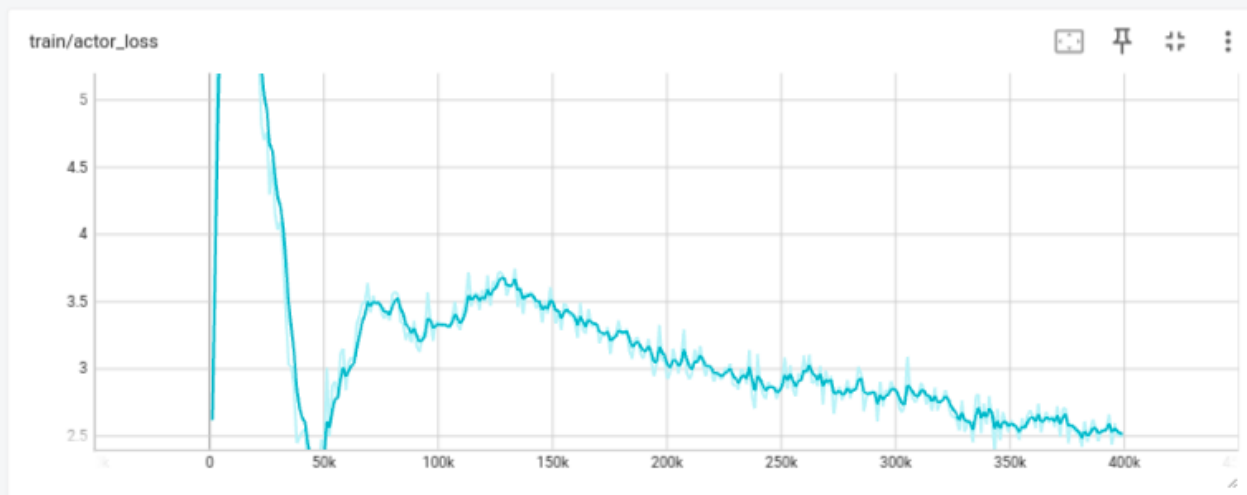
شکل ۳.۵: نمودار میانگین پاداش اپیزودیک در طول آموزش برای کنترل نقطه به نقطه

بر اساس این نمودارها، روند کاهش خطای شبکه‌های Actor و Critic نشان‌دهنده یادگیری موثر و بهبود عملکرد مدل است. همچنین، افزایش میانگین پاداش اپیزودیک نشان‌دهنده بهبود توانایی مدل در انجام وظایف مورد نظر است.

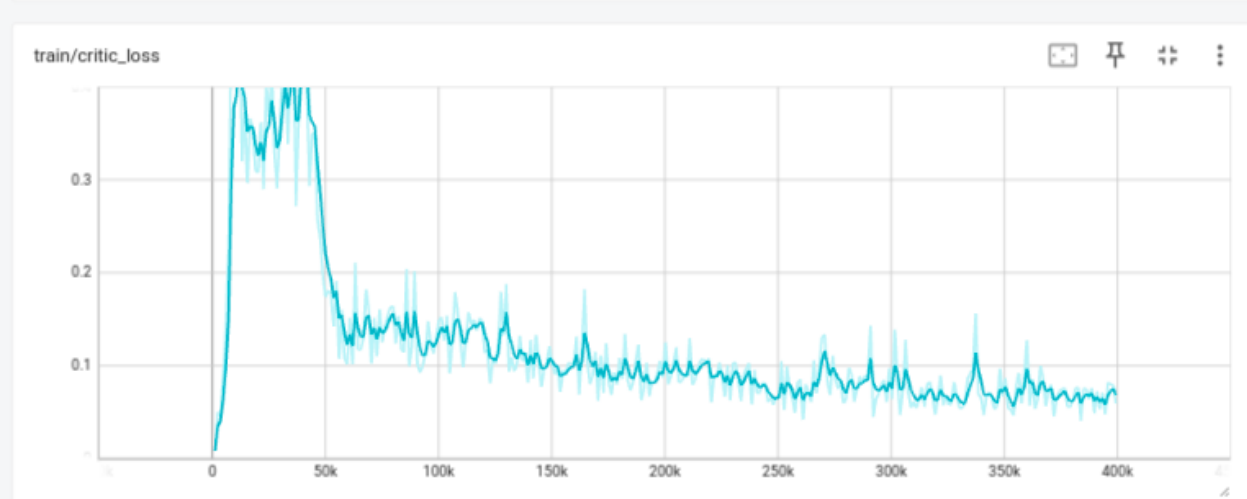
نوسان موجود در این نمودارها به علت مواجه شدن ربات با نقاط جدید است که بعضا باعث ایجاد چالش در یادگیری شده است.

۲.۱.۵ نمودار آموزش ردیابی مسیر دایره‌ای

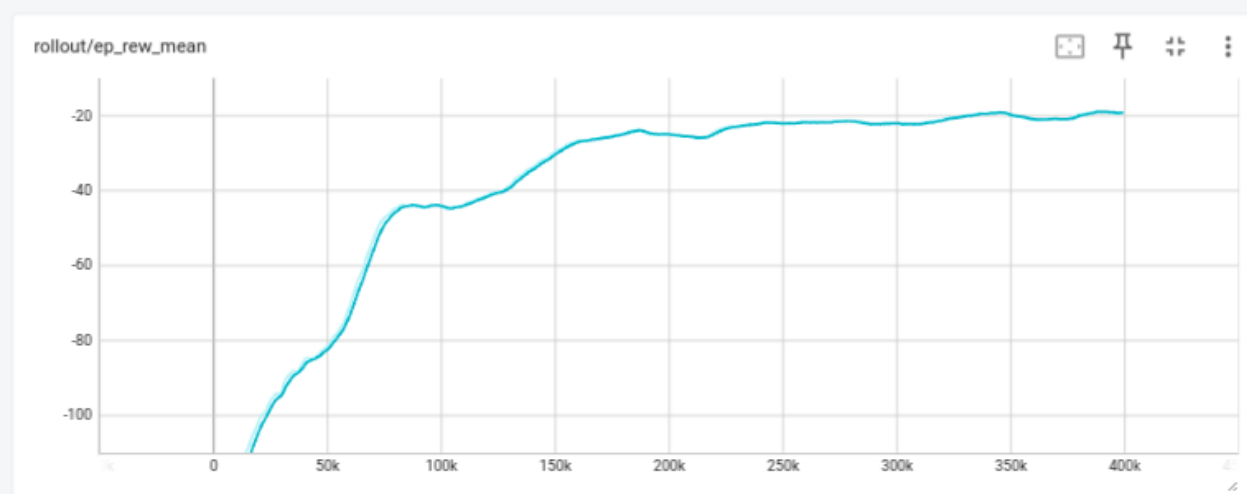
در این بخش، نمودارهای خطای شبکه Actor و شبکه Critic و همچنین میانگین پاداش اپیزودیک در طول آموزش مدل DDPG برای ردیابی مسیر دایره‌ای ارائه شده‌اند.



شکل ۴.۵: نمودار خطای شبکه Actor در طول آموزش برای ردیابی مسیر دایره‌ای



شکل ۵.۵: نمودار خطای شبکه Critic در طول آموزش برای ردیابی مسیر دایره‌ای

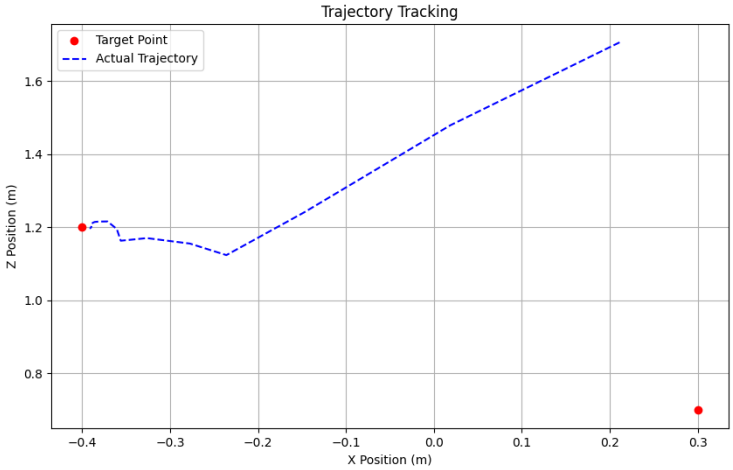


شکل ۶.۵: نمودار میانگین پاداش اپیزودیک در طول آموزش برای ردیابی مسیر دایره‌ای

نمودارها نشان می‌دهند که مدل DDPG توانسته است خطای شبکه‌های Actor و Critic را به تدریج کاهش داده و میانگین پاداش اپیزودیک را افزایش دهد، که این نشان‌دهنده یادگیری موفق مدل و بهبود عملکرد آن در ردیابی مسیر دایره‌ای است.

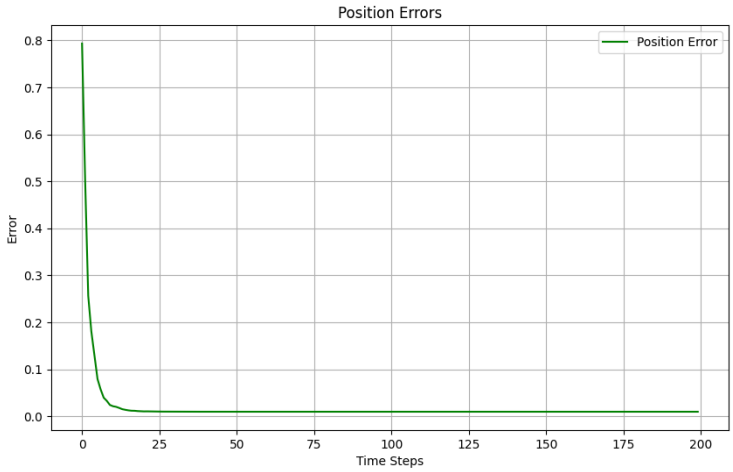
۳.۱.۵ نتایج کنترل نقطه به نقطه

در این قسمت به بررسی نتایج وظیفه کنترلی نقطه به نقطه می‌پردازیم. ربات به گونه‌ای آموزش دیده است که با تعیین یک نقطه هدف به سمت آن حرکت می‌کند و روی آن نقطه ثابت می‌ماند. مکان اولیه ربات نیز به صورت رندوم در فضای کاری آن خواهد بود.



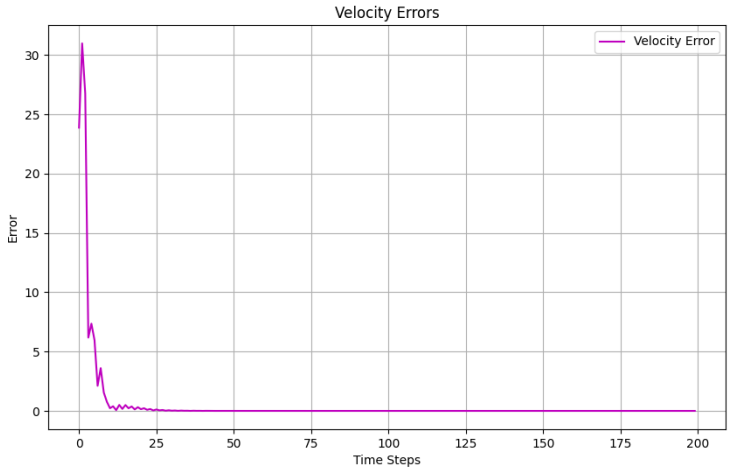
شکل ۷.۵: کنترل نقطه به نقطه

شکل ۷.۵ کنترل نقطه به نقطه را نشان می‌دهد. نقطه قرمز، نقطه هدف است و مسیر واقعی توسط ربات با خط چین آبی مشخص شده است. همانطور که مشخص است ربات تقریبا مسیر بهینه را برای رسیدن به نقطه هدف انتخاب کرده است(مسیر بهینه خط راست مستقیم است). در نهایت نیز تقریبا روی نقطه هدف به صورت ثابت قرار گرفته است.



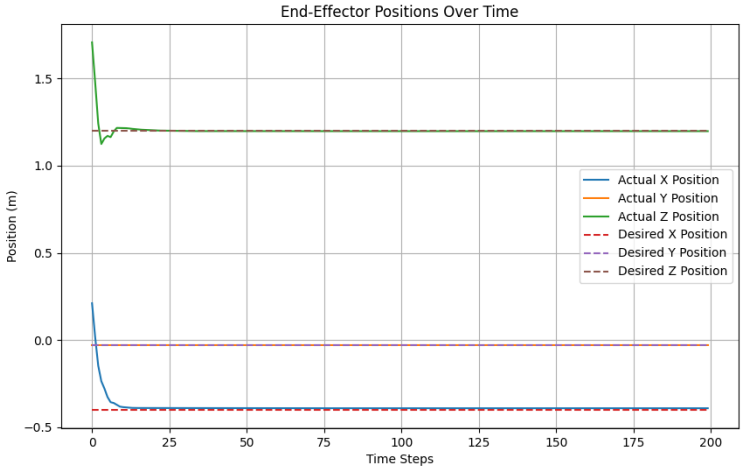
شکل ۸.۵: خطای موقعیت در طول زمان برای وظیفه کنترل نقطه به نقطه

شکل ۸.۵ خطای موقعیت در طول زمان را نشان می‌دهد. خطای موقعیت به سرعت کاهش یافته و به مقدار نزدیکی به صفر می‌رسند.



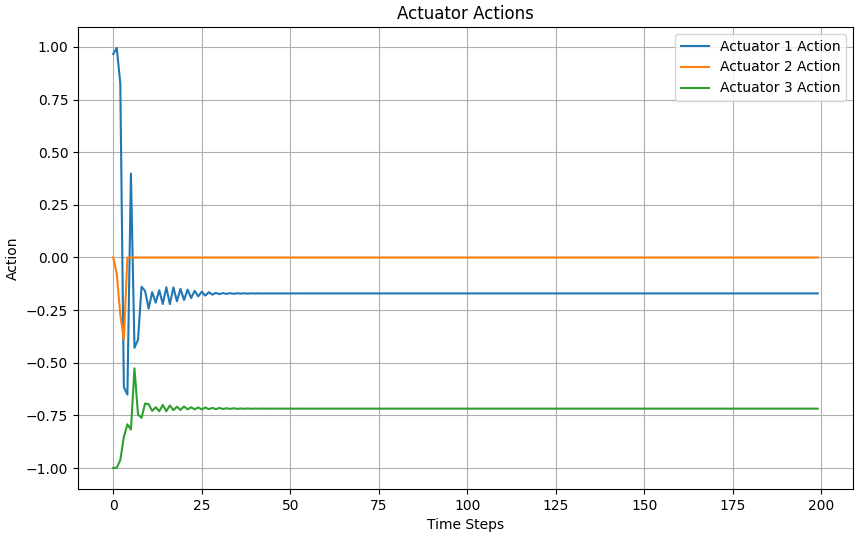
شکل ۹.۵: خطای سرعت در طول زمان برای وظیفه کنترل نقطه به نقطه.

شکل ۹.۵ خطای سرعت در طول زمان را نشان می‌دهد. خطاهای سرعت نیز به تدریج کاهش یافته و به مقدار نزدیکی به صفر می‌رسند.



شکل ۱۰.۵: موقعیت مجری نهایی در طول زمان برای وظیفه کنترل نقطه به نقطه

شکل ۱۰.۵ موقعیت مجری نهایی در طول زمان را نشان می‌دهد. این نمودار نشان می‌دهد که مجری نهایی به خوبی مسیر مورد نظر را دنبال کرده است.



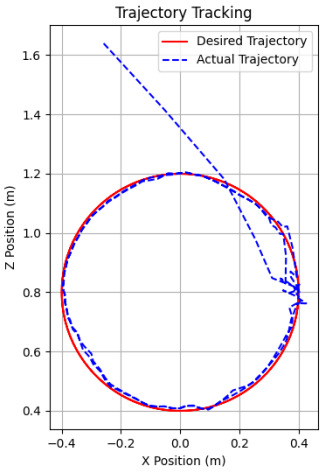
شکل ۱۱.۵: اعمال موتور در طول زمان برای وظیفه کنترل نقطه به نقطه

شکل ۱۱.۵ اعمال موتور در طول زمان را نشان می‌دهد. اعمال موتور به تدریج پایدار شده و به مقادیر بهینه‌ای می‌رسند.

این نتایج نشان می‌دهند که مدل DDPG به خوبی توانسته است وظیفه کنترل نقطه به نقطه را انجام دهد و مجری نهایی را به دقت به نقطه هدف برساند.

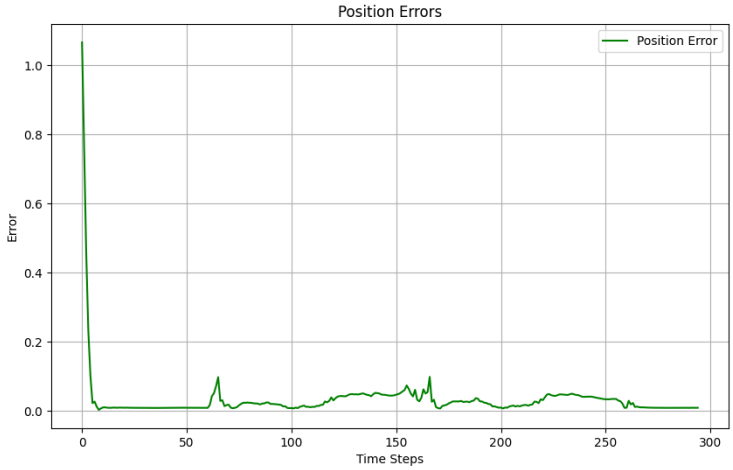
۴.۱.۵ نتایج ردیابی مسیر دایره‌ای

در این قسمت به بررسی نتایج وظیفه ردیابی مسیر ربات می‌پردازیم که در آن ربات به صورت رندوم از یک نقطه در فضای کاری شروع به حرکت می‌کند. در ابتدا ربات روی نقطه ابتدایی مسیر دایروی قرار می‌گیرد و مدت زمانی تعیین شده‌ای را در آن نقطه می‌ماند. سپس مسیر دایروی مشخص که دوبار چرخش حول نقطه مرکزی است را طی می‌کند و در نهایت روی نقطه انتهایی ثابت می‌ماند.



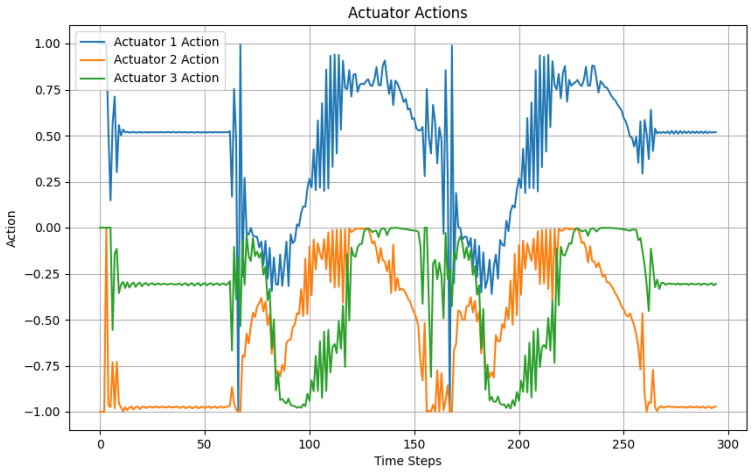
شکل ۱۲.۵ : ردیابی مسیر دایره‌ای

شکل ۱۲.۵ ردیابی مسیر دایره‌ای را نشان می‌دهد. در این نمودار، مسیر واقعی ربات و مسیر هدف دایره‌ای با هم مقایسه شده‌اند. دقت بالای ردیابی نشان می‌دهد که ربات توانسته است با موفقیت مسیر دایره‌ای را دنبال کند و انحراف از مسیر هدف بسیار کم بوده است.



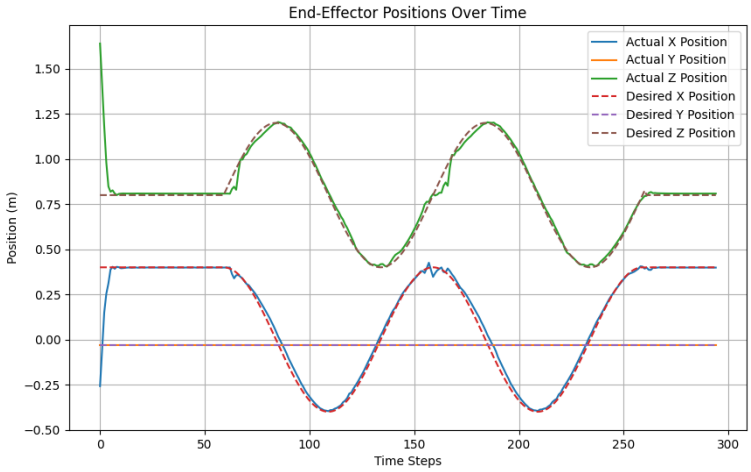
شکل ۱۳.۵: خطاهای موقعیت در طول زمان برای ردیابی مسیر دایره‌ای

شکل ۱۳.۵ خطاهای موقعیت را در طول زمان نشان می‌دهد. این نمودار نشان می‌دهد که چگونه خطای موقعیت (تفاوت بین موقعیت واقعی و هدف) در طول زمان کاهش می‌یابد. کاهش خطاهای موقعیت نشان‌دهنده بهبود دقت و پایداری سیستم کنترلی در ردیابی مسیر دایره‌ای است.



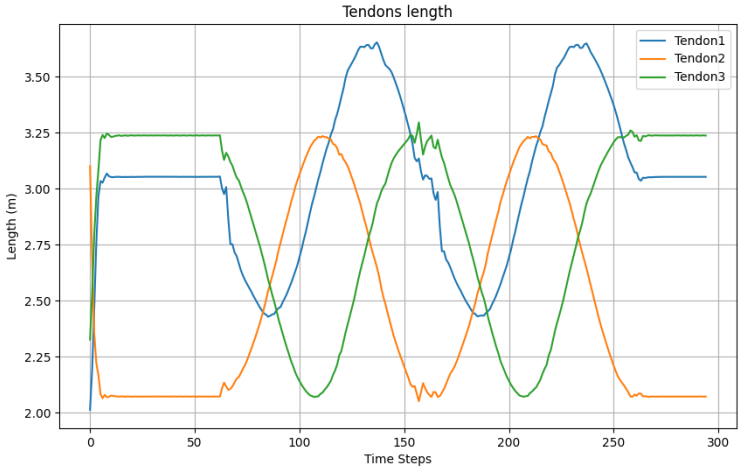
شکل ۱۴.۵: اعمال محرک‌ها در طول زمان برای ردیابی مسیر دایره‌ای

شکل ۱۴.۵ اعمال کنترلی محرک‌ها را در طول زمان نشان می‌دهد. همان‌طور که مشاهده می‌شود، اعمال محرک‌ها برای هر سه کابل به صورت متناوب تغییر می‌کنند تا ربات بتواند مسیر دایره‌ای را دنبال کند. تغییرات در اعمال کنترلی نشان‌دهنده تنظیمات دقیق و پیوسته سیستم کنترلی برای حفظ تعادل و حرکت دقیق در مسیر دایره‌ای است.



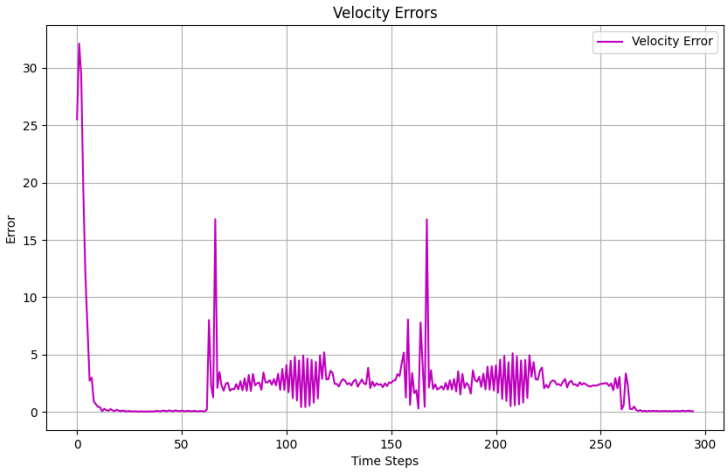
شکل ۱۵.۵ : موقعیت‌های مجری نهایی در طول زمان برای ردیابی مسیر دایره‌ای

شکل ۱۵.۵ موقعیت‌های مجری نهایی را در طول زمان نشان می‌دهد. این نمودار نشان می‌دهد که چگونه موقعیت‌های مجری نهایی در ابعاد مختلف (X Y Z) تغییر می‌کند. مقایسه موقعیت‌های واقعی با موقعیت‌های هدف نشان می‌دهد که ربات با دقت بالایی مسیر دایره‌ای را دنبال می‌کند و انحراف از مسیر هدف به حداقل رسیده است.



شکل ۱۶.۵: طول کابل‌ها در طول زمان برای ردیابی مسیر دایره‌ای

شکل ۱۶.۵ طول کابل‌ها را در طول زمان نشان می‌دهد. این نمودار نشان می‌دهد که چگونه طول هر یک از سه کابل به طور مداوم تنظیم می‌شود تا ربات بتواند مسیر دایره‌ای را دنبال کند. تغییرات طول کابل‌ها نشان‌دهنده هماهنگی دقیق بین محرک‌ها و سیستم کنترلی برای دستیابی به حرکت دقیق و پایدار است.



شکل ۱۷.۵: خطاهای سرعت در طول زمان برای ردیابی مسیر دایره‌ای

شکل ۱۷.۵ خطاهای سرعت را در طول زمان نشان می‌دهد. این نمودار نشان می‌دهد که چگونه خطای سرعت (تفاوت بین سرعت واقعی و هدف) در طول زمان کاهش می‌یابد. کاهش خطاهای سرعت نشان‌دهنده بهبود دقت و پایداری سیستم کنترلی در حفظ سرعت مناسب برای ردیابی مسیر دایره‌ای است.

۵.۱.۵ تحلیل آماری

نتایج حاصل از کنترل نقطه به نقطه در جدول ۱.۵ آورده شده است. این نتایج نشان‌دهنده دقت و پایداری سیستم در رسیدن به نقاط هدف می‌باشند.

جدول ۱.۵: نتایج کنترل نقطه به نقطه

شاخص	خطای موقعیت
RMSE	0.04
میانگین	0.02
واریانس	0.004
انحراف معیار	0.068

در این بخش، شاخص‌های آماری شامل میانگین، واریانس و انحراف معیار برای خطای ردیابی و خطای سرعت در وظیفه ردیابی مسیر محاسبه شده‌اند.

میانگین خطای ما در حالت کنترل نقطه به نقطه حدود ۲ سانتی متر و در حالت ردیابی مسیر برابر ۴ سانتی متر است. باید توجه داشت که این خطا در ابتدا مقدار زیادی است که دلیل آن نقطه شروع به کار رندوم ربات است و این خطا در طول دنبال کردن مسیر مقدار کوچکتری است.

جدول ۲.۵: تحلیل آماری خطای ردیابی و خطای سرعت

شاخص	خطای ردیابی	خطای سرعت
RMSE	0.023	1.63
میانگین	0.04	3.3
واریانس	0.008	13.64
انحراف معیار	0.09	3.69

۶.۱.۵ تحلیل خطاها و عملکرد

در هر دو وظیفه کنترل نقطه به نقطه و ردیابی مسیر دایره‌ای، خطاهای موقعیت و سرعت به تدریج کاهش یافته‌اند که نشان‌دهنده بهبود دقت و پایداری مدل‌ها است. کاهش این خطاها نشان می‌دهد که مدل‌ها توانسته‌اند به خوبی به اهداف تعیین شده برسند و مسیرهای مورد نظر را دنبال کنند.

علاوه بر این، میانگین پاداش اپیزود در طول فرآیند آموزش افزایش یافته که نشان‌دهنده بهبود عملکرد کلی مدل‌ها است. اعمال محرک‌ها و تغییرات طول کابل‌ها نیز نشان‌دهنده تنظیمات مناسب برای رسیدن به اهداف تعیین شده هستند.

این نتایج نشان می‌دهند که استفاده از روش‌های یادگیری تقویتی برای کنترل ربات‌های موازی کابلی می‌تواند منجر به بهبود دقت و پایداری در انجام وظایف مختلف شود. مدل‌ها توانسته‌اند به خوبی با محیط‌های پیچیده سازگاری پیدا کنند و اهداف تعیین شده را با دقت بالا دنبال کنند.

۷.۱.۵ بحث

در این تحلیل آماری، شاخص‌های مختلفی برای بررسی عملکرد سیستم در ردیابی مسیر مورد استفاده قرار گرفت. خطای میانگین نشان می‌دهد که به طور متوسط، پنجه ربات از مسیر هدف چقدر فاصله دارد. واریانس نشان‌دهنده پراکندگی خطاها نسبت به میانگین است و انحراف معیار میزان تغییرات خطاها را نشان می‌دهد.

شاخص RMSE (ریشه میانگین مربع خطاها) نیز نشان‌دهنده مقدار خطای کلی در ردیابی مسیر است و مقدار

بالاتر آن نشان‌دهنده دقت کمتر در ردیابی مسیر می‌باشد.

به طور کلی، نتایج نشان می‌دهد که روش به کار گرفته شده توانسته است با دقت قابل قبولی مسیر مورد نظر را دنبال کند، هرچند که همچنان نیاز به بهبود در کاهش خطاهای سرعت و ردیابی وجود دارد.

۸.۱.۵ بحث و بررسی

نتایج به دست آمده از پیاده‌سازی کنترل‌کننده یادگیری تقویتی DDPG بر روی ربات CDPR نشان‌دهنده توانایی بالای این الگوریتم در انجام دو وظیفه متفاوت، یعنی کنترل نقطه به نقطه و ردیابی مسیر دایره‌ای، با دقت و پایداری مناسب می‌باشد. در ادامه، به بررسی جزئیات نتایج و تحلیل عملکرد ربات در هر دو وظیفه پرداخته می‌شود.

کنترل نقطه به نقطه

در آزمایش‌های کنترل نقطه به نقطه، ربات از یک نقطه شروع به یک نقطه هدف حرکت داده شد. نمودارهای مربوط به موقعیت و سرعت مجری نهایی نشان می‌دهند که ربات توانسته است به طور دقیق به نقطه هدف برسد و پس از رسیدن، بدون نوسانات زیاد در نقطه مورد نظر باقی بماند. این نتایج نشان‌دهنده پایداری و دقت بالای کنترل‌کننده یادگیری تقویتی DDPG در انجام این وظیفه است. همچنین، تحلیل‌های آماری خطای موقعیت و سرعت نشان می‌دهند که میانگین خطاها کم و انحراف معیار آنها نیز کوچک است که بیانگر عملکرد مطلوب کنترل‌کننده می‌باشد.

ردیابی مسیر دایره‌ای

در وظیفه ردیابی مسیر دایره‌ای، کنترل‌کننده یادگیری تقویتی DDPG توانسته است مسیر دایره‌ای تعیین شده را با دقت بالایی دنبال کند. نمودارهای مربوط به مسیر هدف و مسیر واقعی طی شده توسط مجری نهایی ربات نشان می‌دهند که ربات با خطای کمی توانسته است مسیر دایره‌ای را دنبال کند. تحلیل‌های آماری خطاهای ردیابی نیز نشان‌دهنده میانگین خطاهای کم و انحراف معیار کوچک می‌باشند که این موضوع نشانگر عملکرد مناسب کنترل‌کننده در ردیابی مسیر می‌باشد.

تحلیل خطاها و عملکرد

در هر دو وظیفه کنترل نقطه به نقطه و ردیابی مسیر دایره‌ای، نتایج نشان می‌دهند که خطاهای موقعیت و سرعت به مرور زمان کاهش یافته و به مقادیر کوچکی می‌رسند. این موضوع نشان‌دهنده یادگیری مناسب کنترل‌کننده و توانایی آن در کاهش خطاهای ردیابی می‌باشد. به طور کلی، عملکرد ربات در هر دو وظیفه رضایت‌بخش بوده و نشان‌دهنده کارایی بالای الگوریتم DDPG در کنترل ربات CDPR است.

۹.۱.۵ نتیجه‌گیری و کارهای آینده

به طور کلی، استفاده از RL به دلیل بی‌نیاز کردن از مدل‌سازی دقیق سیستم و قابلیت مدیریت پیچیدگی‌ها و نامعینی‌های ذاتی، یک روش مؤثر برای کنترل CDPR است. این روش توانسته است با بهره‌گیری از داده‌های تجربی و تعامل مستمر با محیط، سیاست‌های کنترلی بهینه‌ای را یاد بگیرد و پیاده‌سازی کند. با این حال، همچنان نیاز به بررسی و تحقیقات بیشتری به‌خصوص در زمینه‌های پایداری و امنیت وجود دارد.

در این پروژه، با ایجاد ساده‌سازی‌های ممکن سعی شد تا به عملکرد مطلوبی دست یابیم. هرچند در رابطه با خطاهای موجود، نیاز به ایجاد قیود پیچیده‌تر و بهینه‌سازی‌های بیشتری داریم. نتایج حاصل نشان‌دهنده‌ی توانایی روش‌های یادگیری تقویتی در بهبود عملکرد ربات‌های CDPR است، اما همچنان جای پیشرفت و بهبود وجود دارد.

اهداف آینده

برای بهبود و توسعه‌ی بیشتر این پژوهش، اهداف زیر پیشنهاد می‌شود:

- ایجاد شبکه عصبی برای تنظیم وزن‌های پارامترهای خطا: یکی از اهداف مهم آینده، استفاده از یک شبکه عصبی تحت عنوان Meta Learner برای تنظیم خودکار وزن‌های پارامترهای خطا در تابع پاداش است. این شبکه می‌تواند به صورت پویا و با توجه به شرایط فعلی سیستم، وزن‌های بهینه را تعیین کند و بهبود عملکرد کلی سیستم را تضمین نماید.
- اضافه کردن قیود مربوط به شتاب: در مرحله‌ی بعدی، می‌توان قیود مربوط به شتاب را به مدل اضافه کرد. این

کار می‌تواند به ایجاد یک عامل با توانایی ردیابی مسیر بهینه و هموار کمک کند و عملکرد سیستم را در شرایط دینامیکی بهبود بخشد.

- استفاده از مدل‌های متنوع و بررسی عملکرد آن‌ها: یکی دیگر از اهداف مهم آینده، استفاده از مدل‌های متنوع یادگیری تقویتی و بررسی عملکرد آن‌ها در کنترل CDPR است. با مقایسه‌ی مدل‌های مختلف می‌توان به انتخاب بهترین مدل برای شرایط خاص سیستم دست یافت و عملکرد کلی را بهبود داد.

- بررسی پایداری و امنیت: پایداری و امنیت از مهم‌ترین مباحث در پیاده‌سازی سیستم‌های کنترل رباتیک است. در آینده می‌توان به بررسی عمیق‌تر این مسائل پرداخت و راهکارهای مناسبی برای تضمین پایداری و امنیت سیستم ارائه داد.

- یکپارچه‌سازی با روش‌های دیگر: در نهایت، می‌توان روش‌های یادگیری تقویتی را با سایر روش‌های کنترلی و بهینه‌سازی ترکیب کرد تا به عملکرد بهتری دست یافت. به عنوان مثال، ترکیب RL با کنترل کلاسیک می‌تواند مزایای هر دو روش را به کار بگیرد و سیستم را بهینه‌تر و کارآمدتر کند.

در مجموع، با انجام این تحقیقات و بهبودهای پیشنهادی، می‌توان به کاربردهای گسترده‌تری از یادگیری تقویتی در کنترل ربات‌های CDPR دست یافت و عملکرد این سیستم‌ها را به طور قابل توجهی ارتقا داد.

کتاب‌نامه

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol.17, no.1, pp.1334–1373, 2016. Accessed: 2024-07-19.

[2] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol.32, no.11, pp.1238–1274, 2013. Accessed: 2024-07-19.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015. Accessed: 2024-07-19.

[4] S. Li, Y. Ma, and B. Hu, "Position control of a planar cable-driven parallel robot using reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp.1148–1154, 2020. Accessed: 2024-07-19.

[5] Google DeepMind, "Mujoco: A physics engine," https://github.com/google-deepmind/mujoco. Accessed: 2024-07-19.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," 2015.

[8] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," 2014.

[9] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," 1998.

[10] C. J. Watkins, "Learning from delayed rewards," 1989.

[11] X. Zhao, L. Yang, and D. Luo, "Efficient deep q-network for cdpr control," *Journal of Robotics and Automation*, vol.34, no.2, pp.123–134, 2018. Accessed: 2024-07-19.

[12] J. Yang, H. Wu, and Y. Zhang, "Reinforcement learning control of cdpr using ppo," *IEEE Transactions on Robotics*, vol.35, no.7, pp.672–685, 2019. Accessed: 2024-07-19.

[13] Y. Ma, S. Li, and B. Hu, "Combining rl and classical control for enhanced cdpr performance," in *International Conference on Robotics and Automation*, pp.1123–1129, 2020. Accessed: 2024-07-19.