دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی برق

پروژه درس مقدمه ای بر هوش محاسباتی

کنترل بازوی رباتیک با استفاده از منطق فازی

نگارش
ستایش خاصه تراش
مریم یحیی پور
آرمیتا سید محسنی
آروین طالوئی

استاد
دکتر فرزانه عبداللهی

دی ۱۴۰۳

چکیده

یکی از موضوعات چالش‌برانگیز و پیشرفته در دنیای فناوری، کنترل بازوی رباتیک به صورت هوشمند و خودکار، بدون نیاز به دخالت مستقیم انسان است. در این پروژه، حسگرهای موقعیت و سرعت استفاده می‌شود تا بازو بتواند به طور دقیق محیط را درک کرده و وظایف پیچیده‌ای نظیر رسیدن به هدف، برخورد نکردن به موانع ثابت و متحرک می باشد.

هدف این پروژه، طراحی و پیاده‌سازی یک کنترل‌کننده فازی برای هدایت بازوی رباتیک در محیط‌های مختلف با استفاده از کتابخانه skfuzzy است. در فاز اول، بازو باید به یک نقطه ثابت در فضای کاری هدایت شود. ورودی‌های سیستم شامل زوایای مفاصل، سرعت مفاصل و فاصله بازو تا هدف بوده و خروجی‌ها دستورات حرکتی به مفاصل هستند و از قوانین فازی ساده‌ای استفاده می‌شوند. در فاز دوم، هدف متحرک شده و موانع ثابت به محیط افزوده می‌شوند. ورودی‌ها شامل فاصله و سرعت هدف متحرک، فاصله تا موانع، زوایای مفاصل و سرعت آنها است و خروجی‌ها دستورات حرکتی برای دنبال کردن هدف و اجتناب از موانع خواهند بود. برای مثال: "اگر مانع خیلی نزدیک است، سرعت را کاهش بده و مسیر را تغییر بده". در فاز سوم، محیط پیچیده‌تر شده و موانع متحرک و محدودیت‌هایی مانند محدودیت زاویه مفاصل افزوده می‌شود. ورودی‌ها شامل موقعیت و سرعت موانع متحرک، سرعت هدف و محدودیت‌های محیطی است و خروجی‌ها دستورات تطبیقی برای حفظ دقت و ایمنی حرکت بازو در شرایط دینامیک هستند.

فهرست مطالب

صفحه

فهرست علائم

علائم لاتین

IK	سینماتیک معکوس
FK	سینماتیک مستقیم

فصل اول

مقدمه (دستور العمل)

مقدمه

با پیشرفت‌های چشمگیر در زمینه فناوری و هوش مصنوعی، کنترل هوشمند و خودکار بازوهای رباتیک تبدیل به یکی از مباحث کلیدی و چالش‌برانگیز در علوم مهندسی شده است. بازوهای رباتیک به عنوان یکی از ابزارهای مهم در صنایع مختلف، نقش حیاتی در انجام وظایف پیچیده و دقیق ایفا می‌کنند. از جمله این وظایف می‌توان به جابجایی اشیاء، مونتاژ قطعات، و تعامل با محیط اطراف اشاره کرد.

برای دستیابی به کنترل دقیق و کارآمد این بازوها، استفاده از حسگرهای پیشرفته مانند حسگرهای نیرو/گشتاور، دوربین‌های D-RGB و حسگرهای موقعیت‌یاب (Encoders) ضروری است. این حسگرها اطلاعات دقیقی از محیط فراهم می‌کنند که به بازو کمک می‌کند تا به صورت هوشمندانه تصمیم‌گیری کرده و وظایف خود را با دقت انجام دهد.

هدف از این پروژه، طراحی و پیاده‌سازی یک کنترل‌کننده فازی برای هدایت بازوی رباتیک در محیط‌های مختلف با استفاده از کتابخانه Skfuzzy است. کنترل‌کننده فازی با بهره‌گیری از قوانین و توابع عضویت فازی، به بازو اجازه می‌دهد تا در مواجهه با شرایط مختلف محیطی، به‌صورت تطبیقی و هوشمند عمل کند. در این پروژه، سه فاز مختلف در نظر گرفته شده است که هر کدام شامل چالش‌ها و موانع خاص خود است و هدف نهایی، هدایت دقیق و ایمن بازوی رباتیک در محیط‌های پیچیده و دینامیک می‌باشد.

باید توجه داشت که فهرست ایجاد شده برای زبان فارسی مناسب نیست، کافی است تا کل فهرست را انتخاب کرده و سپس در تب Home بر روی گزینه‌ی Right-to-Left کلیک کنید تا فهرست از راست به چپ قرار گیرد.

فصل دوم

انتخاب محیط و ربات

انتخاب محیط و ربات

در فرآیند طراحی و پیاده‌سازی این پروژه، یکی از مهم‌ترین و پیچیده‌ترین مراحل، انتخاب محیط شبیه‌سازی مناسب و انتخاب ربات مناسب بود. این انتخاب‌ها تاثیر مستقیم و بسزایی بر کیفیت و کارایی پروژه نهایی دارند. هر یک از محیط های در نظر گرفته شده دارای مزایا و معایب خود بودند. در ادامه به به بررسی هر یک از محیط ها و ربات های می پردازیم.

2-1- شبیه ساز Coppeliasim

این شبیه ساز یکی از شبیه ساز های قدرتمند و متنوع در زمینه رباتیک می باشد. زبان مورد استفاده برای ربات های مد نظر پروژه (بازوی متحرک چند مفصله) Lua بود که شباهت زیادی به پایتون دارد اما به قدرتمندی آن نیست و فاقد کتابخانه پردازش فازی می باشد. این شبیه ساز برای ربات های در نظر گرفته شده IK و FK به صورت پیش فرض دارا می باشند. همچنین ربات ها قابلیت اضافه شدن سنسور و موانع ثابت و متحرک را نیز دارند.

2-2- شبیه ساز Gymnasium

شبیه سازی قدرتمند می باشد که به زبان پایتون متصل است و در نتیجه آن به کتابخانه فازی دسترسی دارد. اما ایراد اصلی این شبیه ساز غیر محبوب بودن آن است که باعث شده تعداد کمی از آن استفاده کنند و منابع مناسب برای یادگیری وجود نداشته باشد.

2-3- محیط شبیه سازی Webots

این محیط نیز شبیه سازی قدرتمند می باشد. زبان مورد استفاده در محیط می تواند پایتون و در نتیجه قابلیت اتصال به کتابخانه فازی را دارد. خواصی که برای هر ربات ارائه می دهد متفاوت می باشد و بسته

به کاربرد باید انتخاب شوند. با توجه به اتصال به زبان پایتون و تسلط به نرم افزار (در پروژه مربوط به درس کنترل دیجیتال) تصمیم به انتخاب این شبیه ساز گرفته شد. با توجه به ویژگی هایی که هر ربات ارائه می دهند سه ربات به عنوان گزینه های انتخابی در نظر گرفته شدند که در ادامه به آن ها پرداخته می شود.

2-3-1- ربات Ipr

شامل سنسور های فاصله بر روی بدنه می باشد. فاقد کتابخانه و منابع برای محاسبه IK و FK چه به صورت آماده و چه به صورت دستی.

2-3-2- ربات IRB

شامل کتابخانه IK می باشد ولی قابلیت اتصال سنسور به آن وجود ندارد.

2-3-3- ربات UR5

شامل کتابخانه FK و منابع کافی برای پیدا کردن مستقیم کد IK. دارای سنسور فاصله بر روی سر ربات.

در نهایت ربات IRB و UR5 به عنوان بهترین گزینه ها انتخاب شدند و برای فاز ۱ و ۲ از IRB و برای فاز ۳ از ربات UR5 استفاده شد.

فصل سوم

فاز اول

فاز اول

در این فاز، هدف طراحی و پیاده سازی یک کنترل کننده فازی ساده برای هدایت بازوی رباتیک به یک نقطه ثابت در فضای کاری است که ورودی های آن شامل زوایای مفاصل، سرعت مفاصل و فاصله بازو تا هدف می باشد. این طراحی طی در چند مرحله به شرح زیر، انجام می پذیرد:

- فازی سازی ورودی خروجی ها
- انتخاب موتور استنتاج
- غیر فازی سازی و تحلیل خروجی

3-1- فازی سازی ورودی خروجی ها

برای طراحی فازی ساز و رسیدن به مرحله نهایی با سناریو های متفاوت جلو رفته و در نهایت به یک جمع بندی رسیده شد که شرح این مراحل بدین صورت می باشد:

- در ابتدا به علت فهم اشتباه پروژه کنترل کننده به صورت PID طراحی شد. در این طراحی به فازی سازی ورودی ها (خطا و مشتق آن برای هر مفصل) و فازی سازی خروجی(ضرایب این کنترل کننده) پرداخته شد.
- با توجه به شباهت قوانین طرح شده برای ضریب Kp و تنظیم سرعت از همان قوانین استفاده کرده و بازی بندی و نام گذاری Kp را با سرعت وفق داده شد.

A \ B	NB	NM	NS	ZO	PS	PM	PB
PB	ZO	ZO	NS	NM	NM	NB	NB
PM	PS	ZO	NS	NM	NM	NM	NB
PS	PS	PS	ZO	NS	NM	NM	NM
ZO	PM	PM	PS	ZO	NS	NS	NM
NS	PB	PM	PM	PS	ZO	NS	NS
NM	PB	PM	PM	PS	PS	ZO	NS
NB	PB	PB	PM	PM	PS	ZO	ZO

شکل ۱-0: قانون گذاری سرعت (سمت راست بر حسب خطا و پایین بر حسب نرخ خطا می باشد)

3-2- انتخاب موتور استنتاج

طبق مطالعه مقالات مرجع و همچنین پیشفرض زبان پایتون موتور استنتاج ممدانی انتخاب شد. دلیل انتخاب این موتور این است که

3-3- غیرفازی سازی

برای غیرفازی سازی از پیشفرض پایتون استفاده شد که همان روش *مرکز ثقل* می باشد.

در نهایت با توجه به تست های گرفته شده با استفاده از کنترل کننده فازی نتیجه تا حدودی رضایت بخش بود فقط سرعت کمی داشت که با کاهش قوانین و membership_functions سرعت بالا رفته شد ولی دقت کاسته شد.

فصل چهارم فاز دوم

فاز دوم

در این فاز، هدف بهبود عملکرد کنترل کننده فازی است تا بازوی رباتیک علاوه بر دنبال کردن هدف، بتواند از داده های حسگر فاصله برای تشخیص موانع استفاده کرده و به صورت پویا سرعت خود را تنظیم کند. در این فاز کنترل کننده بخش قبلی گسترش یافت و با استفاده را ورودی های جدید قوانین فازی مربوط به اجتناب از موانع تعریف شد. مراحل این فاز در ادامه توضیح داده خواهد شد.

4-1-اضافه کردن حسگر فاصله و تعریف قوانین جدید

در این قسمت برای ایجاد حسگر فاصله به چالش های زیادی برخورد شد که در زیر سناریو ها و کنترل کننده های طراحی شده مربوط به هر یک شرح داده خواهند.

- سناریو اول: اضافه کردن سنسور فاصله به تمامی مفاصل برای سنجش فاصله مفصل مربوطه تا مانع و ایجاد قوانین مربوطه برای حرکت ربات

```
class JointFuzzyController:
    def __init__(self, joint_name):
        self.e = ctrl.Antecedent(np.arange(-1.5, 1.5, 0.1), f'{joint_name}_e')
        self.ec = ctrl.Antecedent(np.arange(-1.5, 1.5, 0.1), f'{joint_name}_ec')
        self.abs_e = ctrl.Antecedent(np.arange(0, 1.5, 0.1), f'{joint_name}_abce')  # Take the absolute value of the error
        self.velocity = ctrl.Consequent(np.arange(0, 2, 0.1), f'{joint_name}_velocity')
        self.velocityob = ctrl.Consequent(np.arange(0, 2, 0.1), f'{joint_name}_velocity')
        self.distance_obstacle = ctrl.Antecedent(np.arange(-1.5, 1.5, 0.1), f'{joint_name}_distance_obstacle')#fix this------
```

شکل ۱-0: فازی سازی ورودی ها و بازه بندی آن ها

در اینجا مانند بخش قبل بازه اصلی برای فازی کردن در نظر گرفته شده با این تفاوت که ورودی abs_e که نشان دهنده قدر مطلق خطای مفصل مربوطه برای رسیدن به زاویه مورد تا به هدف می باشد؛ velocity و velocityob برای سرعت مفصل می باشند و distance_obstacle نشان دهنده فاصله مفصل تا مانع می باشد.

```
self.distance_obstacle['VN'] = fuzz.trimf(self.distance_obstacle.universe, [-1.5, -1.5, -0.75])
self.distance_obstacle['N'] = fuzz.trimf(self.distance_obstacle.universe, [-0.75, 0, 0.75])
self.distance_obstacle['F'] = fuzz.trimf(self.distance_obstacle.universe, [-1.5, 0, 1.5])
self.distance_obstacle['VF'] = fuzz.trimf(self.distance_obstacle.universe, [0.75, 1.5, 1.5])
# Define fuzzy membership functions for absolute error
self.abs_e['VS']= fuzz.trimf(self.abs_e, [0, 0, 0.375])
self.abs_e['S'] = fuzz.trimf(self.abs_e, [0, 0.375, 0.75])
self.abs_e['B'] = fuzz.trimf(self.abs_e, [0.375, 0.75, 1.125])
self.abs_e['VB'] = fuzz.trimf(self.abs_e, [0.75, 1.5, 1.5])
#Define velocity with obstacle
# Define fuzzy membership functions for velocity
self.velocityob['VS'] = fuzz.trimf(self.velocityob.universe, [0, 0, 0.5])
self.velocityob['S'] = fuzz.trimf(self.velocityob.universe, [0, 0.5, 1])
self.velocityob['M'] = fuzz.trimf(self.velocityob.universe, [0.5, 1, 1.5])
self.velocityob['F'] = fuzz.trimf(self.velocityob.universe, [1, 1.5, 2])
self.velocityob['MF'] = fuzz.trimf(self.velocityob.universe, [1.5, 2, 2])
self.velocityob['VF'] = fuzz.trimf(self.velocityob.universe, [1.75, 2, 2])
```

شکل ۲-0: بازه بندی های ورودی ها و خروجی ها

در این قسمت فاصله تا مانع به صورت پیشفرض بازه بندی شده است. خطای مطلق نیز بر اساس اندازه آن به ۴ دسته تخصیص یافته است. سرعت برای مانع نیز به ۷ دسته تقسیم شده است. تقسیم بندی ها همانطور که در کد مشخص است مانند فاز ۱ انجام شده است.

قانون گذاری نیز به صورت شکل زیر می باشد:

هدف / مانع	خیلی نزدیک	نزدیک	دور	خیلی دور
خیلی نزدیک	کاهش سرعت	کمی کاهش سرعت	ادامه با سرعت قبلی	ادامه با سرعت قبلی
نزدیک	تغییر مسیر	کاهش سرعت	کمی کاهش سرعت	ادامه با سرعت قبلی
دور	تغییر مسیر	تغییر مسیر	ادامه با سرعت قبلی	ادامه با سرعت قبلی
خیلی دور	تغییر مسیر	تغییر مسیر	ادامه با سرعت قبلی	ادامه با سرعت قبلی

شکل ۳-۴: جدول قانون گذاری

در این جدول ۴ حالت وجود دارد :

- کاهش سرعت : سرعتی که مفصل در حال حاضر دارد بررسی می شود و به اندازه دو پله در صورت امکان کاهش یافته می شود(برای مثال سرعت کند به سرعت خیلی کند کاهش یافته می شود و سرعت تقریبا کند به سرعت خیلی کند کاهش یافته می شود چرا که که خیلی کند آخرین پله می باشد.
- کمی کاهش سرعت: سرعتی که مفصل دارد به اندازه یک پله کاهش یافته می شود (برای مثال بررسی می شود سرعت اگر متوسط است کند شود و یا اگر بسیار تند است تقریبا تند شود)
- ادامه با سرعت قبلی: در اینجا از همان کد های فاز ۱ استفاده شده و با بررسی میزان خطا و نرخ تغییرات آن سرعت محاسبه می شود.
- تغییر مسیر: در اینصورت ربات باید مسیر خود را تغییر دهد تا به مانع برخوردی نکند که در آینده دو روش پیشنهاد داده خواهد شد.

❖ **ایراد این سناریو:** در صورت اتصال سنسور به مفاصل از کتابخانه های IK دیگر نمی توان استفاده کرد چرا که ربات را دیگر به عنوان جسمی جدید در نظر میگیرد. برای اینکه IK دستی صورت بگیرد باید از بدنه ربات اطلاعات دقیق می بود که دیتاشیت کاملی از ربات ها وجود نداشت.

- سناریو دوم: سنسور فقط بر روی گریپر. در این صورت کد سناریو قبلی قابل استفاده است.

❖ **ایراد این سناریو:** هم این که سنسور در دهنه آن قرار دارد و باید ربات به صورت خاصی مسیر رسیدن به هدف را دنبال کند و هم اینکه بعد از چند بار امتحان دست یافتیم که سنسور ربات قابلیت تشخیص موانع را ندارد.

- سناریو سوم: سنسور بر روی مانع

❖ **ایراد این سناریو:** مانند سناریو اول جسم جدید تشخیص داده نشده و سنسور کار نمی کند. در غیر اینصورت کد قبلی باز هم پاسخگو می باشد.

- سناریو چهارم: دانستن فاصله تا مانع

این سناریو به دو قسمت تقسیم می شود : برای ربات IRB و برای ربات UR5

- IRB: همانطور که در قسمت های قبلی گفته شد این ربات فاقد FK هست در نتیجه نمیتوان فاصله سر ربات تا مانع و هدف را به دست آورد. رویکرد در نظر گرفته شده برای این ربات جمع قدر مطلق اختلاف فاصله زوایا تا زوایای مورد نیاز می باشد. طبق این رویکرد در بیشترین فاصله فرضی عدد ۲۱ (۷ مفصل که از ۱.۵- تا ۱.۵ رادیان چرخش میکنند) و کمترین ۰ است که یعنی سر ربات به هدف رسیده است طبق این فاصله های در نظر گرفته در قسمت قبلی آپدیت شدند.
- UR5: در این ربات با استفاده از سینماتیک مستقیم می توان به موقعیت سر ربات دست یافت و از طریق آن اختلاف فاصله ها را بررسی کرد که در این حالت با آزمون و خطا و در گرفتن فضای کاری ربات از ۰ تا ۱.۷ در نظر گرفته شده و طبق آن بازه بندی های آن ها مشخص شد.

❖ **ایراد این سناریو:** در این سناریو تنها ایراد مسئله در این است که فقط برای برخورد سر با مانع شرایط لحاظ می شود و ممکن است بدنه باز هم به مانع برخورد کند.

در بین سناریو های گفته شده سناریو آخر بهترین گزینه می باشد.

4-2-شرط تغییر مسیر

برای تغییر مسیر نیز دو رویکرد وجود دارد:

- میدان های پتانسیل مصنوعی: طبق این رویکرد فرض می شود که هدف به عنوان نقطه ای جاذب نیرویی کششی به ربات وارد می کند و مانع به صورت نقطه ای دافع نیرویی رانشی وارد میکند و ربات را از خود دور می سازد.

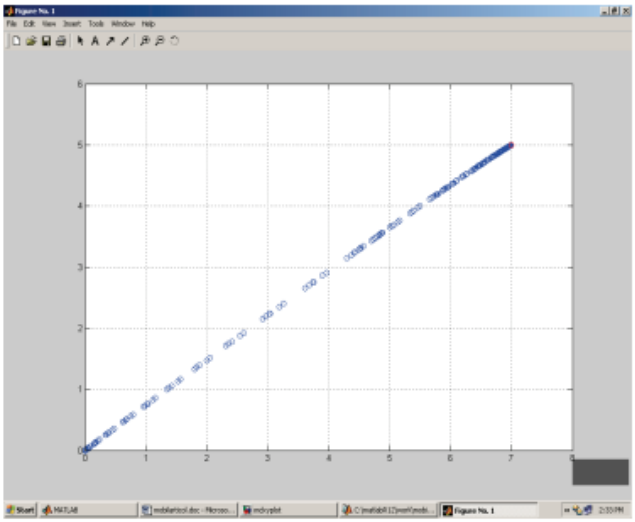

Fig. 3. 4. The robot trajectory without obstacles

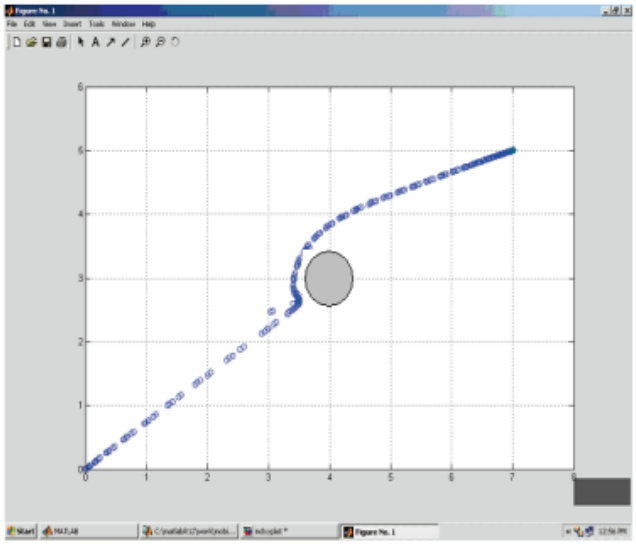

Fig. 3.5. The constrained robot trajectory by one obstacle

شکل ۴-0: میدان های پتانسیلی مصنوعی

- رویکرد دوم بدین صورت است که در ابتدا سنجیده می شود که مانع در کدام سمت هدف قرار دارد و با توجه به جایگاه مانع حرکت متفاوتی در پیش میگیرد. (برای مثال اگر مانع در سمت چپ ربات قرار داشت ابتدا به اندازه فاصله مرکز ثقل مانع تا هدف به سمت چپ مانع میرود به همان میزان از همان نقطه بالا میرود سپس به نقطه بالای هدف میرود و در نهایت نقطه هدف را پیش میگیرد.

در تمامی این حالات فرض بر نقطه ای بودن مانع بود. در اخرین ورژن آپدیت شده از کد بر اساس حجم ربات حد ایمنی را به کلاس فازی تحویل میدهد تا در بازه بندی های (خیلی نردیک) و (نزدیک) فاصله تا مانع لحاظ کند.

فصل پنجم

فاز سوم

فاز سوم

در این فاز، هدف توسعه یک کنترل کننده فازی پیشرفته است که بتواند در یک محیط پویا با موانع متحرک، محدودیت های محیطی و هدف متحرک عمل کند. کنترل کننده طراحی شده قابلیت تطبیق با شرایط مختلف محیطی را داشته باشد و قوانین فازی باید به گونه ای طراحی شوند که بازو بتواند با رعایت محدودیت ها و اجتناب از موانع، هدف را به دقت دنبال کند. با توجه به محدودیت های داشته شوربختانه کنترل کننده این فاز از پروژه نتوانست به درستی تبیین شود. با توجه به تلاش هایی که در این راستا صورت گرفت توضیحات آن خالی از لطف نیست:

5-1- هدف متحرک

کنترل کننده طراحی شده در قسمت های قبل برای رسیدن ربات به یک نقطه بود. به جای هدف متحرک در این محیط محدود قرار بر این گذاشته شد که این بار ربات یک مسیر را دنبال کند که نمایانگر نقاط هدف می باشد که در حال تغییر است. کنترل کننده بخش قبل برای این قسمت و حتی زمانی که مانع وجود داشته باشد پاسخگو می باشد. نقطه تفاوت این بخش با فاز های قبلی در تغییر مسیر می باشد که در ادامه به توضیحات آن پرداخته خواهد شد.

5-2- بررسی ساختار کلی

ربات‌های سری UR شامل مدل‌های UR3، UR5 و UR10 به دلیل ساختار ماژولار، سادگی رابط‌های کنترلی و کتابخانه‌های گسترده، از محبوب‌ترین بازوهای رباتیک در حوزه‌های صنعتی و پژوهشی برای شبیه‌سازی محسوب می‌شوند. در شبیه‌ساز Webots نیز برای این سه مدل، پیکربندی‌های پیش‌فرضی ارائه شده که کاربران می‌توانند به سادگی از آن‌ها در کاربردهای شبیه‌سازی مانند تست‌های اولیه، عیب‌یابی و آموزش استفاده کنند. با این حال، محیط نمونه پیش‌فرض در Webots عمدتاً یک فضای اتوماسیون صنعتی و همکاری بین ربات‌ها است که ممکن است برای پروژه‌های خاص، مناسب یا کاربردی نباشد.

در جریان فاز سوم از پروژه، نیاز به پیاده‌سازی سریع و مؤثر یک ربات UR5 احساس شد. به جای ایجاد یک دنیای کاملاً جدید، تصمیم به استفاده از مدل‌ها و کتابخانه‌های آماده گرفته شد؛ به‌خصوص مدل‌هایی که علاوه بر فایل شبیه‌سازی، توابع کینماتیک مستقیم و معکوس را نیز در دسترس قرار می‌دهند. این کار باعث کاهش زمان توسعه و بهره‌مندی از ساختارها و توابع آزموده‌شده گردید.

در طی فرآیند تحقیق و جست‌وجو در منابع آنلاین، پروژه‌ای به نام «ur5-pick-and-place-webots» که توسط آقای آلن آل‌میییدا (Allan Almeida) انجام شده است، یافت شد. این پروژه مدل نسبتاً کاملی از ربات UR5 را در یک سناریوی کاربردی ارائه می‌دهد. در سناریوی اصلی این پروژه، مراحل زیر پیاده‌سازی و شبیه‌سازی شده است:

1. تشخیص یک بطری از طریق پردازش تصویر و تفکیک آن از پس‌زمینه
2. محاسبات کینماتیک معکوس (Inverse Kinematics) برای دسترسی به بطری با دقت بالا
3. گرفتن بطری توسط گِریپر (Gripper) و اطمینان از پایدار ماندن آن در حین جابه‌جایی
4. خم کردن بطری جلوی دهان یک انسان (که در کنار میز است) برای شبیه‌سازی فرآیند نوشیدن
5. برگرداندن بطری به جای اولیه و بازگشت به وضعیت مبنا (Home Position) برای ربات

از آنجا که در این پروژه نیازی به بخش پردازش تصویر یا حضور انسان در محیط نبود، بخش‌های مربوط به آن سناریو حذف گردید و تمرکز اصلی بر کینماتیک ربات قرار گرفت. بنابراین، توابع مربوط به ماتریس‌های انتقال مفاصل (Joint Transformation Matrices) و محاسبات کینماتیک مستقیم و معکوس در این کد آماده، پاسخگوی دقیق نیازها بود. این امر به شکل قابل‌توجهی در زمان توسعه صرفه‌جویی کرد و هم‌زمان باعث استفاده از کتابخانه‌های تست‌شده و مطمئن در محاسبات کینماتیکی گردید.

در فرآیند پیاده‌سازی فاز سوم پروژه، به‌منظور شناسایی و اجتناب از برخورد با موانع موجود در مسیر ربات، در ابتدا تصمیم بر آن شد که از سنسورهای فاصله‌سنج بر روی مفاصل و چنگک (گریپر) ربات، مطابق با خواست دستورکار پروژه استفاده شود. هدف از این اقدام، فراهم آوردن امکان اندازه‌گیری پیوسته فاصله‌ی موانع و اصلاح مسیر حرکتی ربات به‌صورت بلادرنگ بود. با این حال، افزودن این سنسورها به ساختار فیزیکی ربات، منجر به تغییر در مدل کینماتیک آن شد. این تغییرات به‌گونه‌ای بود که کتابخانه‌ی کنترلی مورد استفاده، که مبتنی بر فرض ثابت بودن ساختار ربات توسعه یافته بود، کارایی خود را از دست داد و عملاً قابلیت استفاده از آن برای محاسبات سینماتیکی ممکن نشد. افزون بر این، عملکرد سنسور فاصله‌سنج نصب‌شده روی چنگک نیز دچار عدم دقت و اطمینان‌پذیری بود. این سنسور، بدون در نظر گرفتن موقعیت واقعی مانع، در تمامی شرایط—از فاصله‌ی بسیار دور گرفته تا تماس مستقیم با مانع—مقدار ثابتی معادل "1000.00" را بازمی‌گرداند. این رفتار نامطلوب، استفاده از این سنسور را برای تشخیص موانع غیرممکن ساخت. بررسی‌های انجام‌شده نشان داد که یا سنسور نیاز به کالیبراسیون تخصصی‌تر داشته یا با معماری کنترلی ربات سازگاری کامل نداشته است؛ امری که در محدوده‌ی زمانی و منابع پروژه، امکان پیگیری دقیق آن وجود نداشت.

پس از این مرحله، به‌عنوان جایگزین، استفاده از دوربین تعبیه‌شده روی چنگک ربات به‌منظور شناسایی موانع از طریق پردازش تصویر مورد بررسی قرار گرفت. این روش شامل پردازش لحظه‌ای تصاویر و شناسایی موانع موجود در میدان دید دوربین بود. با این حال، به دلیل پیچیدگی‌های فنی در پیاده‌سازی پردازش تصویر، بهینه‌سازی الگوریتم‌ها و همچنین محدودیت زمانی پروژه، این روش نیز به عنوان یک راهکار عملی کنار گذاشته شد.

با ارزیابی مجدد شرایط و محدودیت‌های موجود، تصمیم بر آن شد که به‌جای استفاده از سنسورها و پردازش تصویر، یک ساده‌سازی منطقی و کارآمد اعمال شود. در این راهبرد جدید، فرض بر آن گذاشته شد که موقعیت مرکز جرم مانع از پیش مشخص و ثابت است. این رویکرد امکان تعریف مسیر ربات به‌صورت برنامه‌ریزی‌شده و قطعی را بدون نیاز به سنجش لحظه‌ای فاصله فراهم کرد. برای ساده‌سازی بیشتر و به‌منظور حذف نیاز به محاسبه‌ی دقیق ابعاد و شکل هندسی مانع، یک حجم ایمن و حداکثری

به عنوان مدل مانع در نظر گرفته شد. این حجم ایمن به‌صورت یک کره فرضی با مرکزیت مرکز جرم مانع تعریف گردید. در این مدل‌سازی، فرض بر این است که در صورت اجتناب از برخورد با این کره، مانع به‌صورت کامل دور زده می‌شود و از هرگونه برخورد با جسم واقعی جلوگیری خواهد شد. برای عبور از مانع نیز رویکردی ساده اتخاذ شد. فرض بر این گذاشته شد که مانع بر روی سطح میز، یعنی همان سطحی که پایه ربات UR5 بر روی آن نصب شده است، قرار دارد. بنابراین، جهت دور زدن مانع، تنها کافی است که ارتفاع چنگک ربات افزایش یابد تا از بالای این کره‌ی فرضی عبور کرده و سپس ربات به مسیر حرکتی اولیه‌ی خود بازگردد.

در نهایت مجموع پیش‌فرض‌ها و ساده‌سازی‌های اتخاذشده در این فاز از پروژه به صورت زیر می‌باشند:

- حذف سنسورهای فاصله و دوربین پردازش تصویر به دلیل محدودیت‌های عملکردی و زمانی.
- فرض دانستن موقعیت مرکز جرم مانع به‌صورت پیش‌فرض و عدم نیاز به اندازه‌گیری لحظه‌ای.
- تعریف یک حجم ایمن و حداکثری به شکل یک کره برای مدل‌سازی موانع.
- ساده‌سازی مسیر حرکتی ربات با افزایش ارتفاع چنگک برای عبور از بالای کره‌ی ایمن.
- حفظ ساختار کینماتیکی اولیه ربات جهت استفاده‌ی مجدد از کتابخانه‌های آماده.

این مجموعه تصمیمات، اگرچه ممکن است دقت بالایی برای شرایط واقعی نداشته باشد، اما در چارچوب زمانی پروژه، منجر به ساده‌سازی فرایند کنترل ربات شد. این راهکار با حداقل تغییر در مدل اولیه، امکان استفاده‌ی مجدد از کتابخانه‌های آماده و پیشبرد موفقیت‌آمیز فاز سوم پروژه را فراهم ساخت.

با اعمال این مجموعه از ساده‌سازی‌ها و پیش‌فرض‌ها، ربات توانست مسیر تعریف‌شده را با دقت بالا و بدون برخورد با مانع طی نماید. استفاده از توقف‌های موقت و افزایش تعداد نقاط هدف، اگرچه منجر به کاهش سرعت حرکت شد، اما دقت و ایمنی کلی سیستم را بهبود بخشید. علاوه بر این، لحاظ کردن محدودیت‌های حرکتی مفاصل، حرکت ربات را به واقعیت نزدیک‌تر ساخته و امکان ارزیابی دقیق‌تر کنترلر طراحی‌شده را فراهم ساخت.

با این حال، با احتمال بالایی اگر از الگوریتم‌هایی مبتنی بر استلزام لوکازویج یا مدل‌های مشابه آن استفاده می‌شد، ربات قادر بود حرکت پیوسته‌تر و سریع‌تری داشته باشد. علت این برتری به ماهیت پیش‌بینانه و حافظه‌دار این مدل‌ها بازمی‌گردد؛ به این معنا که چنین الگوریتم‌هایی، با حفظ اطلاعات وضعیت‌های قبلی ربات، مسیر را به‌صورت هموار و مداوم پیش‌بینی کرده و نیاز به توقف‌های مکرر برای اندازه‌گیری مجدد را کاهش می‌دهند. همچنین، این مدل‌ها قابلیت مدیریت بهینه‌تر قیود حرکتی را دارند و می‌توانند با لحاظ کردن محدودیت‌های مفصلی، مسیر را با کمترین تغییرات ناگهانی طی کنند.

فصل ششم

جمع‌بندی و نتیجه‌گیری و پیشنهادات

جمع‌بندی و نتیجه‌گیری

با توجه به بررسی عملکرد کنترل کننده فازی در شرایط مختلف نتایجی حاصل شد که برخی از مزایای آن و برخی دیگر از عیب های آن می باشد.

- بر خلاف سایر کنترل کننده های آموخته شده در رباتیک این کنترل کننده نیازی به محاسبات سنگین و حتی گاه غیر خطی نداشت و میتوانست با دقت قابل قبولی به نتیجه مطلوب دست یابد.
- طراحی آن ساده بود و به جای ریاضیات بیشتر از خلاقیت نشات می گرفت.
- انعطاف پذیر بود و برای محیط های مختلف آمادگی داشت تا عملکرد ربات را بهبود بخشد.
  - سرعت سیستم را به شدت پایین می آورد.
  - در صورت کاهش قوانین برای بهبود سرعت دقت کاهش می یابد.

پیشنهادات

با توجه به محدودیت زمانی که برای پروژه وجود داشت ایده های زمانبری که برای بهبود عملکرد پیشنهاد داده شده ولی اجرا نشدند در زیر امده است:

- ترکیب با شبکه عصبی : برای پیدا کردن مسیر بهینه در محیط(استفاده از شبکه عصبی RL)؛ استفاده از شبکه عصبی برای تعیین بازه ها در membership functions و پیدا کردن بهترین حالت
- کاهش قوانین یا به بیانی بهینه کردن آن: بعضی از قوانین عملکرد مشابه دارند و می توان با کاهش بازه بندی ورودی ها قوانین را کاهش و در عین حال سرعت افزایش یابد بدون آنکه دقت تغییر محسوسی بکند.

- ترکیب کنترل کننده های کلاسیک با فازی: مانند فـاز ۱ کـه در ابتـدا کنتـرل کننـده PID بـا ضرایب فازی تعیین شدند می توان سایر کنترل کننده ها را نیز با قوانین گذاری درسـت از ایـن روش استفاده کرد.

منابع و مراجع

[1] Ying Liu , Du Jiang, Juntong Yun, Ying Sun, Cuiqiao Li, Guozhang Jiang , Jianyi Kong, Bo Taol and Zifan Fang; "Self-Tuning Control of Manipulator Positioning Based on Fuzzy PID and PSO Algorithm", *frontiers*, Feb 2022.

[2] Ying Liu , Du Jiang, Juntong Yun, Ying Sun, Cuiqiao Li, Guozhang Jiang , Jianyi Kong, Bo Taol and Zifan Fang; "Robot Control by Fuzzy Logic", *frontiers*, Feb 2022.

[3] Viorel Stoian, Mircea Ivanescu; "Type-3 Fuzzy Control of Robotic Manipulators", *ResearchGate*, Oct 2008.

پیوست‌ها

Abstract

One of the challenging and advanced topics in the world of technology is the intelligent and automated control of robotic arms without the need for direct human intervention. In this project, position and speed sensors are utilized to enable the arm to accurately perceive its environment and perform complex tasks such as reaching a target and avoiding both static and dynamic obstacles. The goal of this project is to design and implement a fuzzy controller to guide the robotic arm in various environments using the skfuzzy library.

In the first phase, the arm is directed to a fixed point within the workspace. The system inputs include joint angles, joint speeds, and the distance from the arm to the target, while the outputs are motion commands to the joints, based on simple fuzzy rules. In the second phase, the target becomes mobile, and static obstacles are added to the environment. Inputs include the distance and speed of the moving target, the distance to obstacles, joint angles, and their speeds, with outputs being motion commands for tracking the target and avoiding obstacles. For example: "If the obstacle is too close, reduce speed and change direction." In the third phase, the environment becomes more complex with the addition of moving obstacles and constraints such as joint angle limits. Inputs include the position and speed of moving obstacles, the speed of the target, and environmental constraints, while outputs are adaptive commands to maintain accuracy and safety of the arm's movement under dynamic conditions.

Key Words: fuzzy, workspace, adaptive

Abstract

Amirkabir University of Technology
(Tehran Polytechnic)

Electrical Engineering Department

Manipulator Control Using Fuzzy Logic

By
Setayesh Khasehtarash
Maryam Yahyapoor
Armita Seyed Mohseni
Arvin Taloui

Supervisor
Dr.Abdollahi

Jan 2025