دانشکده مهندسی مکانیک و مکاترونیک

پایان‌نامه کارشناسی ارشد مهندسی مکاترونیک

طراحی و کنترل ربات پرنده جهت بازرسی خطوط فشار قوی به وسیله پردازش تصویر

نگارنده: مجید مرآتی

اساتید راهنما

دکتر علیرضا احمدی‌فرد

دکتر سعید شیری قیداری

استاد مشاور

دکتر مهدی بامداد

فروردین ۱۳۹۶


بسم الله الرحمن الرحیم

شماره:

باسمه تعالی

تاریخ:

مدیریت تحصیلات تکمیلی

ویرایش:

دانشکده : مهندسی مکانیک و مکاترونیک

گروه : مکاترونیک

پایان نامه کارشناسی ارشد آقای مجید مرآتی به شماره دانشجویی ۹۳۰۲۳۱۰۵ تحت عنوان "طراحی و کنترل ربات پرنده جهت بازرسی خطوط فشار قوی به وسیله پردازش تصویر"

در تاریخ ۱۳۹۶/۱/۲۱ توسط کمیته تخصصی زیر جهت اخذ مدرک کارشناسی ارشد مورد ارزیابی و با درجه ..... مورد پذیرش قرار گرفت.

اساتید راهنما	امضاء	اساتید مشاور	امضاء
دکتر علیرضا احمدی‌فرد		دکتر مهدی بامداد	
دکتر سعید شیری قیداری			

اساتید داور	امضاء	نماینده تحصیلات تکمیلی	امضاء
دکتر حسین خسروی		نام و نام خانوادگی :	
دکتر مصطفی نظری			

تعهد نامه

اینجانب **مجید مرآتی** دانشجوی دوره کارشناسی ارشد رشته مکاترونیک دانشکده مهندسی مکانیک و مکاترونیک دانشگاه صنعتی شاهرود نویسنده پایان نامه **"طراحی و کنترل ربات پرنده جهت بازرسی خطوط فشار قوی به وسیله پردازش تصویر"** تحت راهنمائی دکتر علیرضا احمدی‌فر و دکتر سعید شیری قیداری متعهد می شوم.

- تحقیقات در این پایان نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می باشد و مقالات مستخرج با نام « دانشگاه صنعتی شاهرود » و یا « Shahrood University of Technology» به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می‌گردد.
- در کلیه مراحل انجام این پایان نامه ، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری ، ضوابط و اصول اخلاق انسانی رعایت شده است .

تاریخ

امضای دانشجو

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه های رایانه ای، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه صنعتی شاهرود می باشد. این مطلب باید به نحو مقتضی در

تقدیر و تشکر

من لم یشکر المخلوق، لم یشکر الخالق

حمد و سپاس یکتای بی‌همتا را که لطفش بر ما عیان است، ادای شکرش را هیچ زبان و دریای فضلش را هیچ کران نیست و اگر در این وادی هستیم، همه محبت اوست.

الهی ای مهربان‌تر از ما به ما، از تو می‌خواهم همه کسانی را که حتی ذره‌ای در انجام این امر مرا یاری نموده‌اند، در سایه لطف و محبت بی‌کرانت، سلامت، شاد و موفق بداری.

با تشکر و سپاس فراوان از:

اساتید محترم راهنما آقایان دکتر **علیرضا احمدی‌فرد** و دکتر **سعید شیری قیداری** و استاد مشاور آقای دکتر **مهدی بامداد** که با راهنمایی‌های ارزنده، بی‌دریغ و بی‌منت خود اینجانب را در انجام این پایان‌نامه یاری نمودند.

چکیده

امروزه به کارگیری ربات‌هایی که قابلیت نفوذ به نقاط با شرایط محیطی سخت را دارند، در صنایع مختلف از جمله صنعت برق مورد توجه صنعتگران قرار گرفته است. در این صنعت، برای بازرسی و یا تعمیر خطوط انتقال از نیروی انسانی مجرب استفاده می‌شود. اما به خاطر شرایط خطرناک محیط کاری، نیروی انسانی شاغل در این بخش از صنعت برق با خطراتی مواجه می‌باشند. لذا برای حل این مشکل، ربات‌های گوناگونی ساخته شده است که بعضی از آنها صرفاً پرنده‌اند و بعضی دیگر صرفاً برروی خطوط سوار شده و بر روی آن حرکت می‌کنند که هرکدام از این ربات‌ها شامل ایرادهایی هستند.

هدف ما در این پایان نامه، طراحی و کنترل ربات پرنده‌ای با قابلیت حرکت روی خطوط انتقال برق می‌باشد. ربات در حین حرکت روی خطوط انتقال، به محض رسیدن به موانع، از خط جدا شده و بعد از عبور از مانع مجدداً بر روی خط سوار شده و به مسیر خود ادامه می‌دهد. این حرکت تا رسیدن به برج‌های دارای مقره‌های کششی ادامه پیدا می‌کند.

فرض بر این است که ربات ابتدا به صورت دستی راه اندازی شده و توسط ریموت کنترل تا نزدیکی خطوط انتقال هدایت می‌شود. سپس به صورت کاملاً مختار، با استفاده از گیره نگهدارنده طراحی شده بر روی کابل محافظ خطوط انتقال سوار شده و شروع به حرکت می‌کند.

ابتدا، برای قرار گرفتن کوادکوپتر بر روی کابل محافظ خطوط انتقال (شیلد وایر)، گیره نگهدارنده طراحی شد. در ادامه، در طراحی فیزیکی کوادکوپتر بازرسی کننده از خطوط انتقال فشار قوی و الگوریتم‌های مورد استفاده در آن، نیازها و وظایف این ربات به صورت کلی بیان گردید و برای هریک از زیر بخش‌های آن همانند گیره نگهدارنده، کنترل کننده مرکزی و بدنه کوادکوپتر، طراحی مهندسی انجام شد. از اطلاعات خواص فیزیکی و دینامیکی به دست آمده از محیط نرم افزاری، برای بیان نیازهای کنترل کننده استفاده شد و کنترل کننده تطبیقی LQR+PID طراحی شده، و شبیه‌سازی

آن صورت گرفت. نتایج شبیه سازی انجام شده نشان داد که عملکرد الگوریتم طراحی شده دارای قابلیت اطمینان می‌باشد.

در بخش بینایی ماشین و پردازش تصویر، برای توقف خودکار عملکرد کوادکوپتر در شرایط جوی نامساعد، الگوریتم تشخیص وضعیت جوی مورد بحث و طراحی قرار گرفت و با انجام آزمایشی در محیط Matlab، دقت ۹۱ درصدی در تشخیص وضعیت جوی حاصل گردید. این نتیجه با توجه به تحقیقات صورت گرفته تاکنون از بهترین نتایج در این زمینه می‌باشد.

لازم به ذکر است در این پایان نامه، از آنجا که بحث بازرسی خطوط انتقال با استفاده از دوربین‌های کرونا و مادون قرمز، مساله حل شده هستند، به آنها پرداخته نخواهد شد.

واژه‌های کلیدی:

ربات پرنده، کوادکوپتر، نیمه مختار، خطوط فشار قوی انتقال برق، پردازش تصویر

فهرست عناوین صفحه

فهرست اشکال

صفحه

فهرست جداول

صفحه

فصل اول

مقدمه

درحال حاضر جامعه جهانی به نیروی برق بعنوان یک منبع انرژی مطمئن بسیار وابسته شده است. از آنجا که قطع شدن نیروی برق باعث وارد آمدن خساراتی در بخش‌های مختلف از جمله مصرف کننده‌های صنعتی، تجاری و خانگی می‌شود، تلاش در این زمینه این است که مجموع زمان قطعی برق به حداقل برسد. برای این منظور، در قسمت‌های مختلف صنعت برق بازرسی‌هایی صورت می-پذیرد و در صورت مشاهده معایب، جهت رفع آن اقدام می‌شود. یکی از قسمت‌هایی که نیاز به بازرسی پیوسته دارند، خطوط انتقال و توزیع نیروی برق هستند. از آنجا که به مرور زمان، خطوط انتقال برق فرسوده می‌شوند، لازم است عمل بازرسی، تعمیر و یا تعویض قطعات معیوب به صورت مداوم و پیوسته انجام شود.

۱-۱ نگهداری از خطوط انتقال برق

لوازم و تجهیزات موجود بر روی خطوط فشار قوی برق، نیاز به بازرسی دوره‌ای، تعمیر و یا تعویض دارند. اپراتورهایی که انجام این کار را برعهده دارند، بیشتر به تعویض قطعات معیوب تمایل دارند[۱]. تعویض در اینجا به معنی این است که بخشی از خط و یا قسمتی از آن از شبکه جدا می‌شود و کابل جدید جایگزین می‌شود. همچنین نگهداری از آن نیز به این معنی است که خطوط در شرایطی مناسب قرار گیرد و طول عمر تجهیزات موجود بر روی آن افزایش پیدا کند[۱]. عملیات نگهداری که برروی خطوط انتقال برق ممکن است انجام شود شامل نگهداری برج و تیر برق، نگهداری سیم و کابل برق و کنترل رویش و رشد گیاه بر روی دکل می‌باشد.

از بین موارد فوق، کنترل رویش و رشد گیاه بر روی دکل به خصوص در مناطق گرمسیر و مرطوب، از بیشترین سهم در بازرسی برخوردار است. حتی در مناطقی که در آن گیاه دارای سرعت رشد پایین هستند، می‌تواند نیاز به بازرسی هر ۳ سال یکبار را داشته باشند[۲].

در شکل ۱-۱ نمونه‌ای از رشد و نزدیک شدن درختان به خطوط فشار قوی نمایش داده شده است.

شکل (۱-۱) نمایی از نزدیک شدن درخت به خطوط انتقال برق[۴]

در مورد نگهداری از سیم و کابل برق هوایی نیز می‌توان گفت، در صورت مشاهده عیوب مثل زدگی و پارگی، کابل باید تعویض شود. یکی از موارد مهم در بازرسی از خطوط انتقال برق، بازرسی از زنجیره مقره و به طور کلی، عایق‌ها است. در صورتی که سطوح عایق‌ها آلوده به مواد با قدرت عایقی کمتر از خود شوند، با ایجاد جریان خزشی و سطحی بر روی آن، باعث افزایش تلفات می‌شوند و در مواقعی که میزان آلودگی بیشتر از حد مجاز باشد، موجب ایجاد جرقه بین ناحیه برق‌دار با ناحیه خنثی شده و باعث قطعی برق نیز می‌شود[۲]. همچنین در صورتی که زنجیره مقره با شکستگی همراه باشد، لازم است جهت جلوگیری از قطعی برق و یا سقوط کابل بر روی سایر فازها، قسمت معیوب جایگزین شوند.

۱-۲ بازرسی از خطوط انتقال برق

برای اجرای عملیات تعمیر و یا نگهداری از خطوط انتقال برق، نیاز است که شرایط آن قسمت از خط مشخص شده باشد. این موضوع در مورد شبکه‌های برق هوایی نیز صدق می‌کند. افراد متخصص، قبل

از اینکه به محل اعزام شوند، لازم دارند در مورد نیاز و یا عدم نیاز به تعمیر یا تعویض قطعات را مطالعه نمایند[۳].

بازرسی از خطوط برق دارای مشکلات عدیده‌ای است. چیزی که در اینجا نیاز است، اطلاعات کافی در مورد شبکه است که بر اساس آن بتوان تصمیم قاطعانه‌ای گرفت که آیا نقطه مورد نظر نیاز به تعمیر یا تعویض دارد یا خیر[۳]. به دست آوردن این دسته اطلاعات از محیط‌های خشن و پرخطری چون خطوط شبکه فشار قوی و فوق فشار قوی کار ساده‌ای نمی‌باشد و نیاز به هزینه و وقت بسیار دارد[۳]. خطوط انتقال برق، در مناطق و مسیرهای متنوعی ساخته شده‌اند. هرکدام از آنها دارای ولتاژهای مختلف بر اساس استانداردهای مختلف می‌باشند. بنابراین حتی ممکن است بهره‌برداران یک کشور، دارای تجهیزات متفاوتی باشند. بر همین اساس تفاوت بین کشورها در این زمینه بسیار زیاد می‌باشد[۳].

۱-۳ روش‌های متداول بازرسی خطوط انتقال برق

به طور کلی، روش‌های موجود بازرسی خطوط انتقال فشار قوی را در سه قسمت می‌توان شرح داد. بازرسی زمینی، هوایی و بازرسی خودکار. در اینجا به اختصار درمورد دو مورد اول بحث می‌شود و در مورد بازرسی خودکار در فصول بعد به طور مفصل پرداخته خواهد شد.

۱-۳-۱ بازرسی زمینی

در بازرسی زمینی که قدیمی‌ترین نوع روش بازرسی از خطوط هوایی انتقال برق نیز می‌باشد، بازرسین به منطقه اعزام می‌شوند و ابتدا به صورت چشمی و یا با استفاده از دوربین‌های مخصوص از جمله دوربین ماورای بفنش، عمل بازرسی از تجهیزات و سازه‌ها را انجام می‌دهند[۲ و ۳]. این کار برای بسیاری از معایب احتمالی موجود در برج‌ها، عایق‌ها و هادی‌ها موثر می‌باشد. همچنین بازرسان می-

توانند با استفاده از آنتن و گوش دادن به صدای اطراف، در مورد وجود پدیده کرونا تحقیق کنند[۲]. در مواقعی که بازرسی کامل به روش زمینی برای خطوط امکان پذیر نباشد، با صعود به برج‌ها عمل بازرسی به صورت هوایی صورت می‌پذیرد.

۱-۳-۲ بازرسی هوایی

در بازرسی زمینی خطوط توزیع هوایی با توجه به اینکه ولتاژها در محدوده فشار متوسط (۱۱ ، ۲۰ ، ۳۳ و ۶۲ کیلوولت) و فشار ضعیف (۱۱۰ و ۲۲۰) هستند، و با توجه به اینکه خطرات ناشی از این مقدار ولتاژ را می‌توان با رعایت نکات ایمنی و با استفاده از وسایلی مانند لباس ایمنی، کفش عایق، دستکش، کمربند و سایر تجهیزات ایمنی رفع نمود، کار بازرسی از این نوع خطوط در حال حاضر نیز توسط نیروهای شرکت توزیع و برق منطقه‌ای صورت می‌گیرد[۴]. اما در مورد خطوط هوایی فشار قوی و فوق فشار قوی که بین ۱۳۲ الی ۷۲۰ (۷۴۰) کیلوولت هستند، موضوع تفاوت پیدا می‌کند. در این نوع خطوط با ولتاژهای بسیار بالا، خطراتی که نیروی انسانی را تهدید می‌کند بسیار بیشتر و شدیدتر است. برای کار در محدوده این خطوط، به ابزار و لباس‌های پیشرفته‌تری مانند برهند[1] نیاز است و از آنجا که اغلب این نوع خطوط در مناطق بین‌شهری و صعب‌العبور واقع‌اند، در مواقعی لازم است از بالگرد جهت رسیدن به خطوط استفاده شود که هزینه را افزایش می‌دهد[۲]. بنابراین، بازرسی از این نوع خطوط دارای هزینه بالا می‌باشد و برای بازرسی از آن لازم است وقت زیادی صرف شود. همچنین این کار برای نیروهای متخصص دارای خطرات جانی جدی می‌باشد.

به طور کلی، بازرسی هوایی در شبکه انتقال فشار قوی و فوق فشار قوی، با استفاده از هواپیما و بالگرد انجام می‌شود. در بازرسی با استفاده از هواپیما، پرواز در نزدیکی خطوط انجام می‌شود و در اکثر اوقات چند متخصص به طور همزمان نقاط مختلف خط را مورد بازرسی قرار می‌دهد. عمل بازرسی با استفاده

[1] barehand

از بالگرد همانند روش هواپیما است. با این تفاوت که استفاده از بالگرد فقط در مواقعی صرفه اقتصادی خواهد داشت که منطقه شهری باشد و یا طول کوتاهی از خط برای بازرسی مد نظر باشد.

در شکل ۱-۲ نمونه‌ای از روند بازرسی خطوط انتقال فشار قوی با سوار شدن افراد بر روی خط نمایش داده شده است و شکل ۱-۳ نیز استفاده از بالگرد جهت بازرسی از خطول انتقال فشار قوی برق را نشان می‌دهند.

شکل (۱-۲) بازرسی خطوط انتقال فشار قوی با سوار شدن بر خط

شکل (۱-۳) بازرسی از خطوط انتقال فشار قوی با استفاده از بالگرد

تا اینجا، روند بازرسی سنتی و متداول از خطوط انتقال فشار قوی مورد بحث و بررسی قرار گرفت. در ادامه، روش مدرن در بازرسی از خطوط ارائه می‌شود و به طور مختصر در مورد آن توضیح داده می-شود.

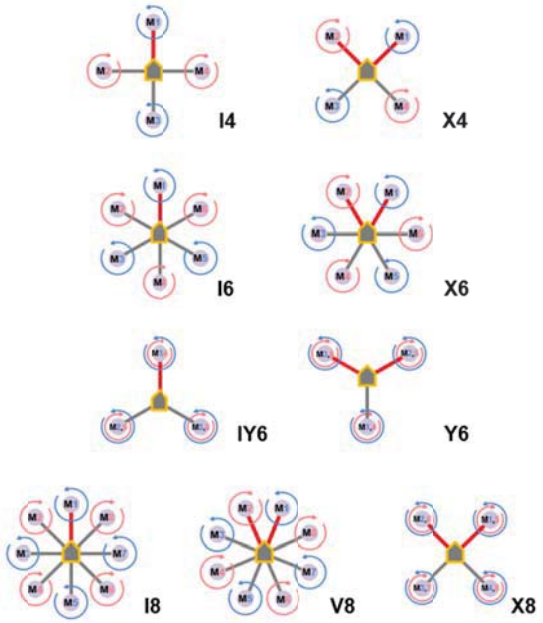
۱-۴ بازرسی با استفاده از هواپیمای بدون سرنشین (پهپاد)[1]

هواپیمای بدون سرنشین یا پهپاد به رباتی گفته می‌شود که انسان در آن قرار ندارد و کنترل آن یا به صورت خودکار و با استفاده از برنامه از پیش تعریف شده روی کنترل کننده آن و یا به صورت بی سیم توسط انسان صورت می‌گیرد[۲]. در سیستم نیروی محرکه هواپیماهای بدون سرنشین و ساختمان پرواز آنها تنوع زیادی وجود دارد. اما به صورت کلی به دو کلاس هواپیماهای بدون سرنشین و عمود-پروازهای بدون سرنشین تقسیم می‌شوند[۳]. پهپادهایی که از تعدادی روتور برای پرواز و کنترل آن استفاده می‌شود، در مقایسه با هواپیماهای با بال‌های ثابت مزیت‌هایی دارند. اولاً این نوع ربات‌ها قادرند از حالت سکون و درجا به صورت عمودی ارتفاع بگیرند، ثانیاً این قابلیت را دارند که در سرعت‌های پایین نیز به شکل پایدار و بدون مشکل پرواز نمایند. در صورتی که هواپیماهای با بال‌های ثابت فقط قادرند در سرعت بالا پرواز کنند.[۲] همچنین اینگونه پهپادهای چند روتوره، به باند و فضای بزرگ برای نشست و برخاست نیازی ندارد. اما هواپیماهای بدون سرنشین، به اینگونه فضاها نیاز دارد تا فرصت کافی جهت افزایش و یا کاهش سرعت خود را داشته باشد[۳].

رایج‌ترین و شناخته‌شده‌ترین عمودپروازها دارای یک روتور هستند، درحالی‌که مولتی‌روتورها نوع خاصی از عمودپروازها هستند که از دو روتور یا روتورهای بیشتری برای پرواز استفاده می‌کنند (شکل ۱-۴). چندروتورها اغلب از زاویه پیچ ثابت استفاده می‌کنند لذا کنترل آن‌ها از طریق تغییر دور

[1] Unmanned Aerial Vehicle (UAV)

روتورها که منجر به تغییر اندازه نیروی بالا برنده می‌شود، صورت می‌گیرد. این مولتی‌روتورها بر اساس تعداد روتورها و پیکربندی آن، دسته‌بندی می‌شوند.



شکل(۱-۴) رایج ترین مولتی روتورها[۲۱]

نوع خاصی از ربات‌های چندروتور، کوادکوپتر (کوادروتور) نام دارد که همانند بالگرد کار می‌کند ولی به جای یک ملخ، از ۴ ملخ برای پرواز و کنترل در آن استفاده می‌شود (شکل ۱-۵)[۲]. کنترل جهت در آن با ایجاد اختلاف بین سرعت بال‌ها انجام می‌شود. این نوع ربات پرنده، برخلاف سایر هواپیماهای بدون سرنشین، اغلب در کاربردهای غیر نظامی استفاده می‌شوند در صورتی که هواپیماهای بدون سرنشین اغلب در کاربردهای نظامی مورد استفاده قرار می‌گیرند.

شکل (۱-۵) نمونه‌ای از کوادکوپتر

۱-۵ مقایسه پیکربندی‌های موجود عمودپروازها

جدول ۱-۱ مقایسه‌ای کلی و گذرا نسبت به پیکربندی‌های موجود و عمودپروازها را ارائه می‌دهد. در این جدول می‌توان مشاهده کرد که با در نظر گرفتن سادگی و عملکرد، کوادکوپتر و هلیکوپتر هم‌محوره در زمره‌ی بهترین پیکربندی‌ها هستند. لازم به ذکر است که پیکربندی‌های مختلف فهرست شده در جدول مذکور، تمام پیکربندی‌های مناسب برای سیستم‌های عمودپروازهای بدون سرنشین نیستند؛ اما چند تفاوت کلیدی بین این دو پیکربندی وجود دارد. کوادکوپترها دینامیک ناپایدار دارند درحالی‌که که هم‌محورها ذاتاً پایدار هستند. این ناپایداری کوادکوپتر باعث مانور پذیری و طراحی کنترل کننده مطلوب می‌شود.

جدول (۱-۱) مقایسه کلی پیکربندی‌های موجود در عمودپروازها

پیکربندی	مزایا	معایب
Fixed-wing	مکانیک ساده، عملکرد آرام	عدم توانایی در شناور ماندن
Single Rotor	کنترل‌پذیری خوب، مانور پذیری خوب	مکانیک پیچیده، روتور بزرگ
Axial Rotor	مکانیک ساده، جمع‌وجور	آیرودینامیک پیچیده
Coaxial Rotor Heli	مکانیک ساده، جمع‌وجور	آیرودینامیک پیچیده
Tendem Rotor Heli	کنترل‌پذیری خوب، آیرودینامیک ساده	مکانیک پیچیده، سایز بزرگ
Quadrotor	مانور پذیری خوب، مکانیک ساده،	مصرف انرژی زیاد، سایز بزرگ

	بارگذاری زیاد	
Blimp	قدرت کم، مداومت پروازی زیاد	سایز بزرگ، مانور پذیری ضعیف
Bird-Like (Ornithopter)	مانور پذیری خوب، جمع‌وجور، مصرف انرژی کم	کنترل پیچیده، مکانیک پیچیده، عدم توانایی در شناور ماندن
Insect-Like (Flapping Wing)	مانور پذیری خوب، جمع‌وجور	کنترل پیچیده، مکانیک پیچیده

۱-۶ اهداف اصلی پایان نامه

هدف در این پروژه، طراحی و کنترل ربات پرنده چهار ملخه‌ای است که جهت اتوماسیون عمل بازرسی از خطوط انتقال فشار قوی برق، بتواند بر روی خط سوار شود و بر روی آن حرکت کند. این ربات ابتدا به صورت دستی راه‌اندازی شده و توسط ریموت کنترل تا نزدیکی خطوط انتقال هدایت می‌شود و سپس به صورت کاملا مختار، بر روی کابل محافظ هوایی (شیلد وایر) سوار شده و شروع به حرکت می‌کند. ربات به محض رسیدن به موانع، از خط جدا شده و بعد از عبور از مانع مجدداً بر روی خط سوار شده و به راه خود ادامه می‌دهد. این حرکت تا رسیدن به برج های دارای مقره‌های کششی ادامه پیدا می‌کند.

استفاده از چنین ربات پرنده‌ای مزایای زیر را به همراه خواهد داشت:

۱- نیاز حضور اپراتور در مدت زمان بازرسی از خطوط انتقال را از بین می‌برد.

۲- سرعت کار آن در مقایسه با ربات‌هایی که روی خطوط حرکت می‌کنند بیشتر است.

۳- کیفیت بازرسی و تصاویر گرفته شده توسط آن، از ربات صرفاً پرنده بهتر است.

۴- با توجه به اینکه با سوار شدن کوادکوپتر بر روی خط، دیگر لازم نیست تمامی ۴ موتور پرمصرف آن روشن باقی بماند و تنها از ۲ موتور کم مصرف و سرعت پایین، جهت حرکت آن بر روی خط استفاده می‌شود، از نظر مسافت قابل پیمایش، با نصب پنل خورشیدی بر روی کوادکوپتر، می‌توان از آن برای بازرسی‌های طولانی مدت یا حتی دایمی استفاده کرد.

از مشکلات موجود در اینجا، می‌توان به پیچیدگی کنترل ربات برای موقعیت‌یابی در اطراف شبکه انتقال قدرت اشاره کرد. با توجه به موقعیت حساس و ولتاژ بالای خطوط انتقال برق، جهت موقعیت‌یابی ربات پرنده، دقت بالایی مورد نیاز می‌باشد. دومین مشکل، وجود وزش باد شدید در ارتفاعات در بازرسی از خطوط انتقال است. حفظ پایداری در اینگونه شرایط برای یک ربات پرنده کار مشکلی است. به خصوص اینکه اینگونه ربات پرنده وزن نسبتاً کمی دارند و وزش باد، آن را از وضعیت عادی خود منحرف می‌کند. بنابراین، ربات باید قادر به تشخیص چنین انحرافات ناخواسته شود. حتی در شرایطی که ربات بر روی خط قرار گرفته باشد، ربات باید بتواند با قابلیت اطمینان بالا پایدار باقی بماند.

۱-۷ مطالب فصل‌های بعدی

در ادامه پروژه و در فصول بعد، موضوعات زیر مورد بررسی قرار می‌گیرند.

در فصل ۲، پژوهش‌های انجام شده در زمینه کوادکوپتر و سوابق کارهای تحقیقاتی در خصوص بخش-های مهم طراحی و پیاده‌سازی یک کوادکوپتر مانند بینایی ماشین، طراحی کوادکوپتر، مدلسازی کوادکوپتر و کنترل آن شرح داده می‌شود.

در فصل ۳، روش‌های انجام کار در این تحقیق، در بخش‌های مختلف نظیر طراحی گیره نگهدارنده کواکوپتر بر روی خط انتقال برق، و الگوریتم‌های کنترل و بینایی ماشین بطور مفصل توضیح داده می‌شود. و در هر بخش، نتایج حاصل از طراحی فیزیکی یا شبیه سازی مربوطه، بیان و در مورد آن به صورت مفصل بحث می‌شود.

در فصل ۴، جمع بندی نتایج بدست آمده صورت گرفته و نتیجه‌گیری نهایی انجام می‌شود. در ادامه این فصل، پیشنهادات جهت انجام کارهای آینده نیز ارائه می‌گردد.

فصل دوم

مطالعات انجام شده

در این فصل در زمینه پژوهش‌ها و تحقیقات درباره طراحی و کنترل ربات پرنده کوادکوپتر جهت بازرسی خطوط فشار قوی بحث می‌شود. به طور عمده، بخش‌های مهم از طراحی و پیاده‌سازی یک کوادکوپتر عبارت از "پردازش تصویر و بینایی ماشین"، "طراحی" و "مدل‌سازی و کنترل" می‌باشد.

۲-۱ پردازش تصویر و بینایی ماشین

پردازش تصاویر دیجیتال شاخه‌ای از دانش رایانه است که دارای دو شاخه عمده بهبود تصاویر و بینایی ماشین است. بهبود تصاویر دربرگیرنده روش‌هایی چون استفاده از فیلتر محوکننده و افزایش تضاد برای بهتر کردن کیفیت دیداری تصاویر و اطمینان از نمایش درست آنها در محیط مقصد (مانند چاپگر یا نمایشگر رایانه) است، در حالی که بینایی ماشین به روش‌هایی می‌پردازد که به کمک آنها می‌توان معنی و محتوای تصاویر را درک کرد تا از آنها در کارهایی چون رباتیک استفاده شود.

در معنای خاص آن پردازش تصویر عبارتست از هر نوع پردازش سیگنال که ورودی یک تصویر است مثل عکس یا صحنه‌ای از یک فیلم و خروجی پردازشگر تصویر می‌تواند یک تصویر یا یک مجموعه از نشان‌های ویژه یا متغیرهای مربوط به تصویر باشد. اغلب تکنیک‌های پردازش تصویر شامل برخورد با تصویر به عنوان یک سیگنال دو بعدی و به کار بستن تکنیک‌های استاندارد پردازش سیگنال روی آنها می‌شود. پردازش تصویر اغلب به پردازش دیجیتالی تصویر اشاره می‌کند ولی پردازش آنالوگ تصویر هم وجود دارند.

از آنجا که تصویر از یک محیط می‌تواند اطلاعات بسیار زیادی را شامل شود، معمولاً در ربات از تمامی اطلاعاتی که یک تصویر به یک انسان معمولی می‌دهد استفاده نمی‌شود و یک مدل ساده‌سازی شده از آن برای کارهای خاصی که برای ربات مطرح شده‌اند به کار برده می‌شوند. بنابراین دید و برداشت یک ربات از محیط اطراف و جزئیات آن، در مقایسه با برداشت یک انسان -به دلیل محدودیت در سرعت پردازش اطلاعات تصویری- بسیار ناچیز خواهد بود.

به طور کلی، در این پایان نامه، از دوربین در ربات به منظور مکان‌یابی، شناسائی، رهگیری، تشخیص آب و هوا استفاده می‌شود.

۲-۱-۱ مکان‌یابی توسط ربات پرنده

به عنوان مثال، برای ربات پرنده‌ای[۱] که هدف آن فرود بر روی یک سطح شناور دریایی است، هدف از نصب دوربین بر روی ربات، مکان‌یابی دقیق آن در محیط است. از آنجا که سنسورهای GPS امروزی خطایی در حدود چند متر را دارند، استفاده از آن به تنهایی نمی‌تواند برای ربات اطمینان حاصل کند که آیا نقطه فرود دقیقاً بر روی شناور است؟ بنابراین از یک دوربین استفاده شده و با پردازش اطلاعات حاصل از تصویر آن و تنظیم موقعیت و کنترل ربات به وسیله ترکیب بهینه GPS و دوربین، این کار صورت می‌گیرد. در اینجا، دوربین در محیط فقط به دنبال علائمی است که توسط کاربر تعریف شده است. ممکن است این علائم به صورت یک رنگ یا شکل خاصی باشد که برای علامت در نظر گرفته شده باشد. بنابراین ربات با باقی نقاط روی تصویر کاری ندارد و بدین ترتیب سرعت کار خود را بالا می‌برد.

در مواردی که نیاز به محاسبه موقعیت دقیق ربات وجود دارد، بحث ادومتری تصویری[۱] یا نقشه برداری و مکان یابی همزمان[۲] مطرح می‌گردد.

به دست آوردن مدل فیزیکی و تهیه نقشه محیط اطراف یکی از مسائل مهم و اساسی در رباتیک است و کاربردهای بسیاری از قبیل هدایت خودکار ربات، ردیابی و تشخیص اجسام و اشخاص و انجام عملیات جستجو و نجات دارد. طی ده سال گذشته پیشرفت چشمگیری در زمینه هدایت خودکار ربات و به خصوص مکان‌یابی و نقشه‌برداری همزمان از محیط صورت گرفته است.

[1] Visual Odometry
[2] Simultaneous Localization And Mapping (SLAM)

در علم رباتیک، SLAM به مساله‌ای ریاضی گفته می‌شود که در آن، همزمان با مکان‌یابی ربات در محیط ناشناخته، به تهیه نقشه از محیط نیز پرداخته می‌شود. الگوریتم‌های مختلف احتمالاتی با به کارگیری از یک، دو یا چند عدد دوربین RGB، یا با استفاده از دوربین‌های RGB-d مانند Xbox Kinect در این زمینه وجود دارد.

در SLAM تصویری، ما به دنبال تخمین تراژکتوری حرکت ربات و محیط در حین بازسازی نقشه آن محیط هستیم[۴]. هر الگوریتم SLAM تصویری، وظایفی شامل رهگیری، نقشه برداری از محیط، موقعیت‌یابی درون نقشه و بستن حلقه[1] را برعهده دارند.

انواع مختلفی از الگوریتم‌های SLAM در زمینه روش‌های مکان‌یابی احتمالاتی وجود دارد که از آن جمله می‌توان به فیلتر کالمن[2]، مونت کارلو، فیلتر ذره و مدل مارکوف اشاره کرد.

در سال ۱۹۸۵ یکی از اولین تلاش‌ها برای انجام مکان‌یابی و نقشه‌برداری به صورت همزمان با استفاده حسگر پویشگر لیزری و شفت انکودر و با توجه به عدم قطعیت‌های موجود در مسئله توسط چتیلا و همکارانش صورت گرفت[۵]. در سال ۱۹۹۰ اسمیت و همکارانش برای اولین بار با استفاده از مشخصه‌های محیط به نقشه‌برداری پرداختند و از فیلتر کالمن برای حل مسئله استفاده کردند[۶].

در دهه گذشته تمرکز بسیاری از محققان بر روی یافتن راه‌حل‌های مناسب برای مکان‌یابی و نقشه‌برداری هم‌زمان به صورت بلادرنگ[3] بوده است. در این میان، محبوب‌ترین روش‌ها برای مسئله مکان‌یابی و نقشه‌برداری همزمان فیلتر کالمن توسعه‌یافته[4][۷] و فیلتر ذره رائو-بلک‌ولایزد[5] بوده‌اند[۸]. فیلتر کالمن تعمیم‌یافته از تقریب خطی استفاده می‌کند و فرض می‌کند که تابع مسیر و سرعت (همچنین شتاب) حرکت ربات به صورت خطی تعریف شده اند. نقشه به‌دست آمده به این روش به دلیل خطای ناشی از خطی‌سازی چندان دقیق نیست و همچنین پیچیدگی محاسباتی این روش زیاد

[1] Loop closing
[2] Kalman Filter (KF)
[3] Real Time
[4] Extended Kalman Filter (EKF)
[5] Rao-Blackwellized Particle Filter (RBPF)

است و در زمان واقعی قابل اجرا نیست. تحقیقات بسیاری برای برطرف کردن این مشکلات صورت گرفته است.

در تلاش برای افزایش دقت مکان‌یابی و نقشه‌برداری هم‌زمان، فیلتر کالمن جدیدی تحت عنوان کالمن بی بو[1] معرفی شد که از تقریب خطی استفاده نمی‌کرد[۹]. فیلتر کالمن بی بو از یک روش قطعی نمونه‌برداری استفاده می‌کند که متوسط و کواریانس تخمین‌ها را به دست می‌آورد. فیلتر کالمن بی بو به جای تابع غیرخطی سیستم، تابع چگالی احتمال را تخمین می‌زند و هر چقدر میزان غیرخطی بودن سیستم بیشتر باشد تخمین قابل اطمینان‌تری نسبت به فیلتر کالمن توسعه‌یافته ارائه می‌دهد. اما پیچیدگی محاسباتی آن مشابه فیلتر کالمن توسعه یافته است. از طرف دیگر الگوریتم مکان‌یابی و نقشه‌برداری هم‌زمان سریع یا نقشه‌برداری سریع[2]، با بکارگیری هم‌زمان فیلتر ذره و فیلتر کالمن توسعه یافته، حجم محاسبات را به طور قابل توجهی نسبت به فیلتر کالمن توسعه یافته و فیلتر کالمن بی بو کاهش می‌دهد[۱۰]. در نهایت الگوریتم مکان‌یابی و نقشه‌برداری هم‌زمان سریع بر مبنای فیلتر کالمن بی بو تحت عنوان مکان‌یابی و نقشه‌برداری همزمان سریع بدون ردیابی یا نقشه‌برداری سریع بدون ردیابی[3] به عنوان یکی از جدیدترین روش‌های مطرح شده در زمینه مکان‌یابی و نقشه‌برداری هم‌زمان با بهره‌گیری از تخمین قابل اطمینان فیلتر کالمن بی بو ، دقت را افزایش و حجم محاسبات را کاهش می‌دهد[۱۱]. نشان داده شده است که نقشه‌برداری سریع بدون ردیابی در مقایسه با روش‌های نقشه‌برداری سریع بر مبنای فیلتر کالمن توسعه یافته و نقشه‌برداری سریع-۲[4] از نظر دقت و کیفیت نقشه‌برداری عملکرد مطلوب‌تری دارد[۱۲]. با وجود این برتری، هنوز ظرفیت‌های بررسی نشده زیادی در رابطه با الگوریتم نقشه‌برداری سریع بدون ردیابی وجود دارد.

مکان‌یابی و نقشه‌برداری کاملا به هم وابسته هستند. به این معنا که برای یافتن مکان دقیق ربات در یک محیط، نقشه صحیحی از محیط مورد نیاز است. نقشه مجموعه‌ای از مشخصه‌های محیط است که

[1] Unscented Kalman Filter (UKF)
[2] FastSLAM (FS)
[3] Unscented FastSLAM (UFS)
[4] FastSLAM2

مختصات این مشخصه‌ها نسبت به ربات اندازه‌گیری شده است و در نتیجه دقت نقشه به دقت مکان‌یابی ربات بستگی دارد و افزایش دقت تخمین موضع، نقش بسزایی در بهبود دقت و کیفیت نقشه دارد.

جدول (۲-۱) مقایسه خطای موقعیت یابی تصویری با استفاده از معیار آزمون TUM RGB-D (X به معنی از دست رفتن موقعیت ربات در حین آزمایش است)[۱۴]

	میانگین مربعات خطا (cm)			
نام پایگاه داده	ORB-SLAM	PTAM	LSD-SLAM	RGBD-SLAM
fr1_xyz	۰.۹	۱.۱۵	۹.۰۰	۱.۳۴ (۱.۳۴)
fr2_xyz	۰.۳	۰.۲	۲.۱۵	۱.۴۲(۲.۶۱)
fr1_floor	۲.۹۹	X	۳۸.۰۷	۳.۵۱(۳.۵۱)
fr1_desk	۱.۶۹	X	۱۰.۶۵	۲.۵۲(۲.۵۸)
fr2_360_kidnap	۳.۸۱	۲.۶۳	X	۱۰۰.۵ (۳۹۳.۳)
fr2_desk	۰.۸۸	X	۴.۵۷	۳.۹۴ (۹.۵)
fr3_long_office	۳.۴۵	X	۳۸.۵۳	_
fr3_nstr_tex_far	-	۳۴.۷۴/۴.۹۲	۱۸.۳۱	_
fr3_nstr_tex_near	۱.۳۹	۲.۷۴	۷.۵۴	_
fr3_str_tex_far	۰.۷۷	۰.۹۳	۷.۹۵	_
fr3_str_tex_near	۱.۵۸	۱.۰۴	X	_
fr2_desk_person	۰.۶۳	X	۳۱.۷۳	)۲.۰۰(۶.۹۷
fr3_sit_xyz	۰.۷۹	۰.۸۳	۷.۷۳	_
fr3_sit_halfsph	۱.۳۴	X	۵.۸۷	_
fr3_walk_xyz	۱.۲۴	X	۱۲.۴۴	_
fr3_walk_halfsph	۱.۷۴	X	X	_

در الگوریتم‌های مکان‌یابی و نقشه‌برداری هم‌زمان موضع لحظه‌ای ربات با استفاده از مدل حرکت پیش‌بینی شده و سپس با استفاده از تفاوت ویژگی‌های ثبت شده در نقشه و ویژگی‌هایی که به تازگی مشاهده شده‌اند، اصلاح می‌شود. در جدول (۲-۱)، نتایج مقایسه بین دقت الگوریتم‌های مختلف مکان

یابی[1] آورده شده است. در این جدول، مشاهده می‌شود از بین الگوریتم‌های اشاره شده، Orb-SLAM کمترین میانگین مربعات خطا را در همه شرایط داشته است.

لازم به ذکر است از آنجا که ما در این پروژه، صرفاً قصد استفاده از بخش ادومتری SLAM از پیش آماده را داریم، به جزئیات این مبحث پرداخته نخواهد شد.

۲-۱-۲ شناسایی اشیا توسط ربات پرنده

با استفاده از تصاویر ثبت شده توسط دوربین و اعمال پردازش و الگوریتم‌های مورد نظر، می‌توان از آن برای شناسایی و تشخیص اشیا هدف استفاده نمود.

به عنوان مثال، در مرجع [۳] محققان جهت بازرسی از خطوط فشار قوی و تشخیص معایب و آسیب‌دیدگی در آن با استفاده از ربات پرنده، از پایگاه داده ۷۰۰ تایی از مجموعه‌ای از تصاویر آموزشی RGB از شبکه برق با رزولوشن ۱۳ مگاپیکسل (که توسط یک کوادکوپتر گرفته شده است) استفاده کرده و تلاش جهت شناسایی تیر برق و کراس آرم[2] در تصاویر گرفته شده توسط ربات بازرس (تصویر آزمون) نمودند.

الگوریتم استفاده شده در تحقیق اشاره شده، از مجموعه ویژگی‌های تصاویر آموزشی ثبت شده –که همه آنها با یکدیگر تفاوت زیادی دارند- جهت شناسایی شی هدف در تصویر آزمون استفاده می-کند[۳]. در اینجا، منظور از تفاوت زیاد بین تصاویر، وجود اشیایی در تصویر است که در شناسایی شی هدف مزاحمت ایجاد می‌کنند (مانند خودرو، خیابان، خطوط عابر پیاده، انسان و ...) و یا تفاوت در شرایط آب و هوایی منطقه در زمان عکس‌برداری می‌باشد (مانند وجود و یا عدم وجود سایه، درخشندگی هادی، مه و ... در تصویر)[۳]. این محققان برای تشخیص شی هدف در تصاویر آموزشی

[1] Localization
[2] Crossarm

جهت آموزش دسته بند[1] و یافتن آن در تصویر آزمون از خصوصیات رنگی به عنوان ویژگی اصلی به همراه پردازش‌های دیگر (همانند فیلترینگ، تطبیق کلیشه و آستانه‌سازی سطوح خاکستری) استفاده کردند که نتیجه کار، صحت ۷۰ درصدی الگوریتم در تشخیص تیر برق و صحت ۷۲ درصدی در تشخیص کراس آرم بوده است[۳].

شکل (۲-۱) عملکرد صحیح بینایی ربات پرنده در تشخیص تیر برق و کراس آرم [۳]

شکل (۲-۲) عملکرد ناصحیح بینایی ربات پرنده در تشخیص کراس آرم در تصویر [۳]

شکل ۲-۱ نمونه‌ای از تصویری است که عملکرد صحیح الگوریتم را نمایش می‌دهد. همچنین شکل ۲-۲ نیز نمونه‌ای از عملکرد نادرست آن را نمایش می‌دهد.

جهت شناسایی اشیا در تصویر RGB، روش‌های مختلفی مطرح هستند که به طور کلی به صورت زیر دسته بندی می‌شوند:

۱. روش‌های مبتنی بر هندسه و ظاهر اجسام

۲. روش مبتنی بر استخراج ویژگی از تصویر

[1] classifier

۲-۱-۲-۱ روش‌های مبتنی بر هندسه و ظاهر اجسام

در این روش، با استخراج و توصیف شکل هندسی و ظاهر اجسام مورد علاقه، به دنبال این ویژگی‌ها در تصویر، جستجو صورت می‌گیرد. در صورتی که شکل محیطی شی مورد نظر، شکل هندسی داشته باشد، از تبدیل‌هایی همانند تبدیل هاف[۱] و هاف تعمیم یافته[۲] می‌توان جهت پیدا کردن خطوط، دوایر و سایر اشکال هندسی اشیا مورد نظر استفاده کرد. اما اگر شی مورد نظر، شکل هندسی قابل تعریف به صورت معادلات ریاضی را نداشته باشند (یا معادلات پیچیده‌ای داشته باشند)، می‌توان از روش دیگر تحت عنوان تطبیق الگو[۳] استفاده کرد.

در صورتی که تصویر شی مورد نظر دارای نقاط شاخص بارزی نباشد یا در مواقعی که بخش عمده تصویر شی مورد نظر در تصویر هدف نیز پیداست، می‌توان از این روش استفاده کرد. در این روش، از آنجا که تعداد نقاط مورد بررسی برای انطباق در تصاویر بزرگ، بسیار افزایش پیدا می‌کند، این امکان وجود دارد که ابتدا تصاویر شی و تصویر مورد نظر را به یک مقیاس کوچک‌سازی کرده و ابتدا در این تصویر، شی مورد نظر را جستجو کرد. بعد از یافتن مکان حدودی شی مورد نظر در تصویر کوچک‌سازی شده، در تصویر اصلی و در همان محدوده، مجدداً شی مورد نظر جستجو می‌شود تا مکان و وضعیت دقیق آن در تصویر اصلی به دست آید. این کار باعث کاهش بار محاسباتی شده و سرعت عملیات جستجوی الگو را افزایش می‌دهد.

۲-۱-۲-۲ روش مبتنی بر استخراج ویژگی از تصویر

در صورتی که تصویر شی مورد نظر، دارای نقاط شاخص و واضحی باشد، می‌توان از این روش برای جستجوی تصویر شی در تصویر دیگر بهره جست. از آنجا که در این روش از الگوریتم‌های استخراج

[1] Hough Algorithm
[2] Generalised Hough
[3] Template Matching

ویژگی‌ای استفاده می‌شود که نسبت به پارامترهای مختلف هندسی مانند اندازه و چرخش مقاوم هستند، کار تطبیق الگو راحت‌تر انجام می‌شود.

جهت استخراج نقاط شاخص و استخراج ویژگی‌های این نقاط، الگوریتم‌های مختلفی در دو دهه گذشته مطرح شدند. اما با توجه به اینکه از این الگوریتم‌ها در این پروژه استفاده نمی‌شود، در مورد آنها بحث نخواهد شد.

۲-۱-۳ رهگیری اشیا در تصویر

یکی از کاربردهای بسیار مهم و ارزشمند بینایی ماشین، عملیات رهگیری شی و یا مجموعه‌ای از اشیا در تصویر است. هدف اصلی در اینجا، تشخیص و تعقیب اشیائی مشخص و تعیین شده در فریم‌های یک فیلم ویدئویی می‌باشد که ممکن است این ویدئو به صورت زنده و توسط یک دوربین به سیستم ارسال شده باشد. در این زمینه در دهه اخیر پژوهش‌های بسیار زیادی صورت گرفته است و الگوریتم-های مختلفی برای رهگیری طراحی و توسعه داده شده است که در ساده‌ترین آنها بعد از تشخیص شی مورد علاقه در یک فریم، برای آن یک مدل جایگزین مانند اشکال هندسی و یا مدل استخوانی استخراج می‌شود و در فریم‌های بعدی به دنبال این مدل جستجو می‌شود[۱ و ۳]. اما در رهگیری این‌گونه مدل‌ها مشکلات فراوانی وجود دارد. از جمله این مشکلات می‌توان به تغییر شکل ظاهری شی مورد علاقه اشاره کرد[۳]. این تغییر شکل بدان معنی نمی‌باشد که جسم تحت تاثیر نیروهایی شکل اصل خودش را از دست بدهد. بلکه این تغییرات شکل شی ناشی از اثراتی مانند شدت نور محیط، حرکت دوربین و موقعیت آن، جهت قرارگیری شی و قرار گرفتن بخشی از شی در پشت اشیای دیگر می‌باشد که در دید دوربین، به نظر می‌رسد ظاهر شی دچار تغییر شده است[۱ و ۳]. به دنبال تغییر شکل شی در تصویر، تشخیص آن در دنباله تصاویر ویدئویی دچار مشکل می‌شود[۱].

از راهکارهای دیگر برای رهگیری اشیا در تصاویر ویدئویی، می‌توان به استفاده از فیلتر کالمن[1] و فیلتر کالمن تعمیم یافته[2] اشاره کرد.

در ادامه این بخش ابتدا فیلتر کالمن به طور خلاصه تعریف می‌شود و نحوه کاربرد آن در رهگیری اشیا متحرک در تصویر ویدئویی توضیح داده می‌شود.

۲-۱-۳-۱ فیلترکالمن

فیلتر کالمن مشهور، ریشه در فرمولاسیون فضای حالت یا سیستم‌های دینامیکی دارد که راه حل بازگشتی برای مسئله فیلترینگ خطی بهینه فراهم می‌کند[۱۳]. این راه حل از آنجایی بازگشتی است که هرکدام از پیش بینی‌های حالت، از روی پیش‌بینی گذشته و ورودی جدید صورت می‌گیرد. بنابراین فقط برای پیش‌بینی پیشین نیاز به حافظه خواهیم داشت. علاوه بر از بین رفتن نیاز به نگهداری تمامی اطلاعات مشاهده شده پیشین، فیلتر کالمن از نظر بار محاسباتی نیز از تخمین حالت به صورت مستقیم با استفاده از تمامی اطلاعات پیشین سبک‌تر می‌باشد[۱۳].

در این بخش، مطالبی از فیلتر کالمن جهت هموار نمودن استفاده از آن در کاربرد بینایی و پردازش تصویر بیان می‌شود.

یک سیستم دینامیکی خطی و گسسته را که در بلوک دیاگرام شکل ۲-۳ نمایش داده شده است، در نظر بگیرید. مفهوم حالت به طور اساسی به این شرح است:

بردار حالت یا حالت که با X_k نمایش داده شده است، مجموعه حداقلی از داده‌ها است که برای توصیف رفتار دینامیکی سیستم خاص کفایت کند، اندیس k زمان گسسته را بیان می‌دارد[۱۳]. به عبارتی دیگر، حالت عبارتست از مجموعه حداقلی از اطلاعات رفتار پیشین سیستم که برای پیش‌بینی رفتار آینده آن مورد نیاز باشد. به طور کلی، متغیر حالت X_k مجهول است[۱۳]. برای تخمین آن ما از دسته اطلاعات مشاهده شده (اندازه گیری) که با بردار y_k نمایش داده می‌شود، استفاده می‌کنیم.

[1] Kalman Filter (KF)
[2] Extended Kalman Filter (EKF)

به عبارت ریاضی، از بلوک دیاگرام شکل (۲-۳) دو تابع زیر حاصل می‌شود[۱۳]:

۱. معادله فرایند

$$X_k = F_{k+1,k}Xk + w_k \qquad (۱-۲)$$

که در آن:

$F_{k+1,k}$: ماتریس انتقال که حالت X_k را از زمان k تا k+1 می‌گیرد.

w_k : نویز در اینجا، از نوع گاوسی با میانگین صفر و با کواریانس تعریف شده به صورت زیر است:

$$\mathrm{E}[w_n\, w_k^T] = \begin{cases} Q_k \; for \; n = k \\ 0 \; for \; n \neq k \end{cases} \qquad (۲-۲)$$

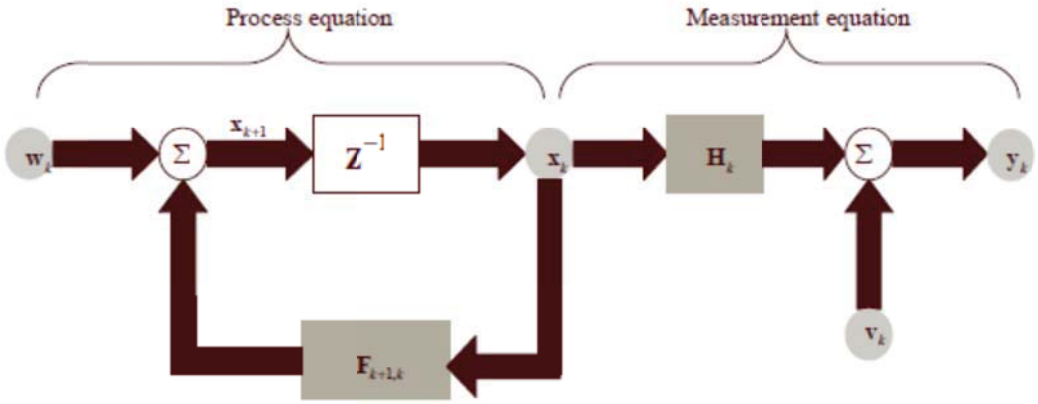
که در آن منظور از بالانویس T، ماتریس ترانسپوز (ترانهاده) می‌باشد. اندازه فضای حالت با M نشان داده شده است.

۲. تابع اندازه گیری

$$\mathrm{y}_k = H_k x_k + v_k \qquad (۳-۲)$$

که در آن، y_k در زمان k قابل مشاهده است و H_k ماتریس اندازه گیری است. [۱۴] نویز اندازه گیری که با v_k نمایش داده شده است، افزودنی، گاوسی سفید با میانگین صفر و کواریانس به صورت زیر است:

$$\mathrm{E}[v_n\, v_k^T] = \begin{cases} R_k \; for \; n = k \\ 0 \; for \; n \neq k \end{cases} \qquad (۴-۲)$$



شکل (۲-۳) گراف شار سیگنال از سیستم دینامیکی خطی و گسسته[۱۳]

علاوه بر این، نویز اندازه گیری v_k، ناهمبسته از نویز فرایند w_k می‌باشد. بعد فضای اندازه گیری با N نمایش داده شده است.

مسئله فیلترینگ کالمن، همانطور که از نامش پیداست، مسئله حل همزمان معادلات فرایند و اندازه‌گیری برای حالت ناشناخته به شیوه‌ای مطلوب به صورت زیر شرح داده می‌شود:

استفاده از تمامی اطلاعات مشاهده شده، شامل بردارهای y1، y2،....yn برای پیدا کردن k بزرگتر از ۱ که حداقل مربعات میانگین خطا را برای x_k به ارمغان آورد.

مسئله، فیلتر کردن در صورت i=k ، پیش بینی در صورت i>k و هموارسازی در صورت 1<i<k می-باشد.

پیش‌بینی بهینه

قبل از رسیدن به کاربرد فیلتر کالمن، آن را برای شرح برخی مفاهیم اساسی برای پیش‌بینی بهینه مفید یافتیم. برای مسائل ساده، این خلاصه در چارچوب متغیر تصادفی اسکالر نمایش داده شده است[۱۳]. تعمیم تئوری به بردار متغیرهای تصادفی یک موضوع ساده است. فرض کنید متغیر اندازه‌گیری قابل مشاهده به ما داده شده است. آنگاه

$$y_k = x_k + v_k \qquad (۵-۲)$$

که x_k سیگنال مجهول است و v_k نویز افزودنی است. فرض کنید x_k یک برآورد پسینی از سیگنال x_k با توجه به مشاهدات y1، y2، ... yk باشد. به طور کلی، برآورد $\widehat{x_k}$ با سیگنال ناشناخته x_k متفاوت است. برای استخراج این برآورد به طور مطلوب، ما به تابع هزینه برای تخمین‌های نادرست احتیاج داریم. تابع هزینه باید دو نیاز را برطرف سازد:

- تابع هزینه نباید منفی باشد.
- تابع هزینه یک رابطه صعودی از خطای پیش بینی x_k است که به صورت زیر تعریف می‌شود.

$$\tilde{x}_k = x_k - \hat{x}_k \qquad (۶-۲)$$

این دو نیاز، با تابع مربع میانگین خطاها که به صورت زیر تعریف می‌شود برآورده می‌شود.

$$J_k = E[(x_k - \hat{x}_k)^2] \qquad (۷-۲)$$

$$J_k = E[(\tilde{x}_k)^2] \qquad (۸-۲)$$

که در آن E اپراتور انتظار و توقع[1] یا امید ریاضی می‌باشد که برای حالت گسسته به صورت زیر تعریف می‌شود.

$$E(x) = \sum_{i=1}^{N} x_i p(x_i) \qquad (۹-۲)$$

وابستگی تابع هزینه Jk در زمان k، بر طبیعت غیر ایستا بودن فرایند پیش بینی بازگشتی تاکید دارد. برای به دست آوردن مقدار بهینه برای پیش بینی $\hat{x}_k$، ما دو تئوری برگرفته از تئوری فرایند تصادفی را در نظر می‌گیریم[۱۳]:

تئوری ۱. برآورد کننده میانگین شرطی، اگر فرایندهای تصادفی $\{x_k\}$ و $\{y_k\}$ به طور مشترک گاوسی باشند، آنگاه براورد بهینه $\hat{x}_k$ که میانگین مربعات خطا J_k را حداقل می‌کند، برآورد کننده میانگین شرطی می‌باشد.

$$x_k = E[x_k | y_1, y_2, \dots, y_k] \qquad (۱۰-۲)$$

[1] Expectation operator

تئوری ۲. اصل تعامد[1]. در صورتی که میانگین فرایندهای تصادفی $\{x_k\}$ و $\{y_k\}$ صفر باشند آنگاه:

$$E[x_k] = E[y_k] = 0 \qquad (۱۱-۲)$$

به ازائ همه مقادیر k

آنگاه:

(۱) فرایندهای تصادفی $\{x_k\}$ و $\{y_k\}$ به طور مشترک گاوسین هستند یا

(۲) اگر برآورد بهینه $\hat{x}_k$ به یک تابع خطی از مشاهدات محدود باشد و تابع هزینه میانگین مربع خطا باشد،

(۳) آنگاه براورد بهینه $\hat{x}_k$ که مشاهدات y_1، y_2، ...، y_k را می‌دهد، نمای متعامد از x_k در فضای محدود شده توسط این مشاهدات است.

۲-۱-۳-۲ فیلتر کالمن تعمیم یافته

تا به اینجا در مورد استفاده از فیلتر کالمن برای تخمین بردار حالت یک سیستم دینامیکی خطی بحث شد. در صورتی که مدل سیستم غیر خطی باشد، فیلتر کالمن را با اضافه نمودن فرایند خطی سازی توسعه می‌دهیم. فیلتر منتجه را فیلتر کالمن تعمیم یافته گویند. (EKF)[۱۳]. چنین توسعه‌ای به واسطه این واقعیت که فیلتر کالمن در سیستم‌های زمان گسسته به صورت معادلات دیفرانسیلی هستند، قابل دسترسی می‌باشد.

برای شروع توسعه یک فیلتر کالمن تعمیم یافته، سیستم دینامیک غیر خطی را که با مدل فضای حالت به صورت زیر نمایش داده شده است در نظر بگیرید:

$$x_{k+1} = f(k, x_k) + w_k \qquad (۱۲-۲)$$

$$y_k = h(k, x_k) + v_k \qquad (۱۳-۲)$$

[1] Principle of orthogonality

که در آن، همانگونه که در قسمت گذشته نیز اشاره شد، w_k و v_k نویزهای سفید گاوسی با میانگین صفر هستند و R_k و Q_k به ترتیب ماتریس های کواریانس هرکدام از آنها هستند. در اینجا، تابع $f(k, x_k)$ نشان دهنده یک تابع ماتریس انتقال غیر خطی است که احتمالا از نوع متغیر با زمان نیز می‌باشد. به همین ترتیب، تابع $h(k, x_k)$ نشان دهنده یک ماتریس اندازه گیری غیر خطی است که این نیز ممکن است متغیر با زمان باشد.

ایده اصلی فیلتر کالمن تعمیم یافته، خطی سازی مدل فضای حالت روابط (۲-۱۲) و (۲-۱۳) در هر لحظه از زمان در حوالی جدیدترین تخمین حالت است که بسته به اینکه کدامیک از توابع خاص مد نظر قرار می‌گیرد، ممکن است یکی از $\hat{x}_k$ و یا $\hat{x}_k^-$ باشد. بعد از اینکه رابطه خطی‌سازی شده به دست آیند، روابط فیلتر کالمن استاندارد در ادامه اعمال می‌شود.

به عبارتی دیگر، تقریب خطی سازی در دو مرحله صورت می‌گیرد [۱۳].

مرحله اول: دو ماتریس زیر تشکیل می‌شوند:

$$F_{k,k+1} = \frac{\partial f(k,x_k)}{\partial x} \quad , x = \hat{x}_k \qquad (۲-۱۴)$$

$$H_k = \frac{\partial h(k,x_k)}{\partial x} \quad , x = \hat{x}_k \qquad (۲-۱۵)$$

که بیان کننده این است که درایه (i,j) ام $F_{k,k+1}$ برابر است با مشتق جزئی iامین عضو F(k,x) نسبت به yامین عضو از x. به همین ترتیب، عضو (i,j) ام ماتریس H_k برابر است با مشتق جزئی iامین عضو H(k,x) نسبت به jامین عضو از x. در حالت ابتدایی، مشتقات جزئی در $\hat{x}_k$ مد نظر است و در حالت بعدی، مشتقات جزئی در نقطه $\hat{x}_k^-$ محاسبه و ارزیابی می‌شوند.

درایه‌های ماتریس‌های $F_{k,k+1}$ و H_k، با داشتن $\hat{x}_k^-$ و $\hat{x}_k$ در زمان kام، همگی قابل محاسبه هستند.

مرحله دوم: بعد از اینکه ماتریس‌های $F_{k,k+1}$ و H_k به دست آمدند، در تقریب مرتبه اول تیلور توابع غیر خطی F(k,x) و H(k,x) حول $\hat{x}_k^-$ و $\hat{x}_k$ به ترتیب قرار داده می‌شوند. در این حالت، F(k,x) و H(k,x) به صورت زیر تقریب زده می‌شوند:

$$\mathrm{F}(\mathrm{k}, \mathrm{x}_k) \approx F_{k+1,k}(x, \hat{x}_k) + F(x, \hat{x}_k) \qquad (۱۶-۲)$$

$$\mathrm{H}(\mathrm{k}, x_k) \approx H_{k+1,k}(x, \hat{x}_k^-) + H(x, \hat{x}_k^-) \qquad (۱۷-۲)$$

با داشتن عبارات تقریب فوق، حال به خطی سازی و تقریب روابط فضای حالت غیر خطی (۴۱) و (۴۲) می‌پردازیم.

$$\mathrm{x}_{k+1} \approx F_{k+1,k} x_k + w_k + d_k \qquad (۱۸-۲)$$

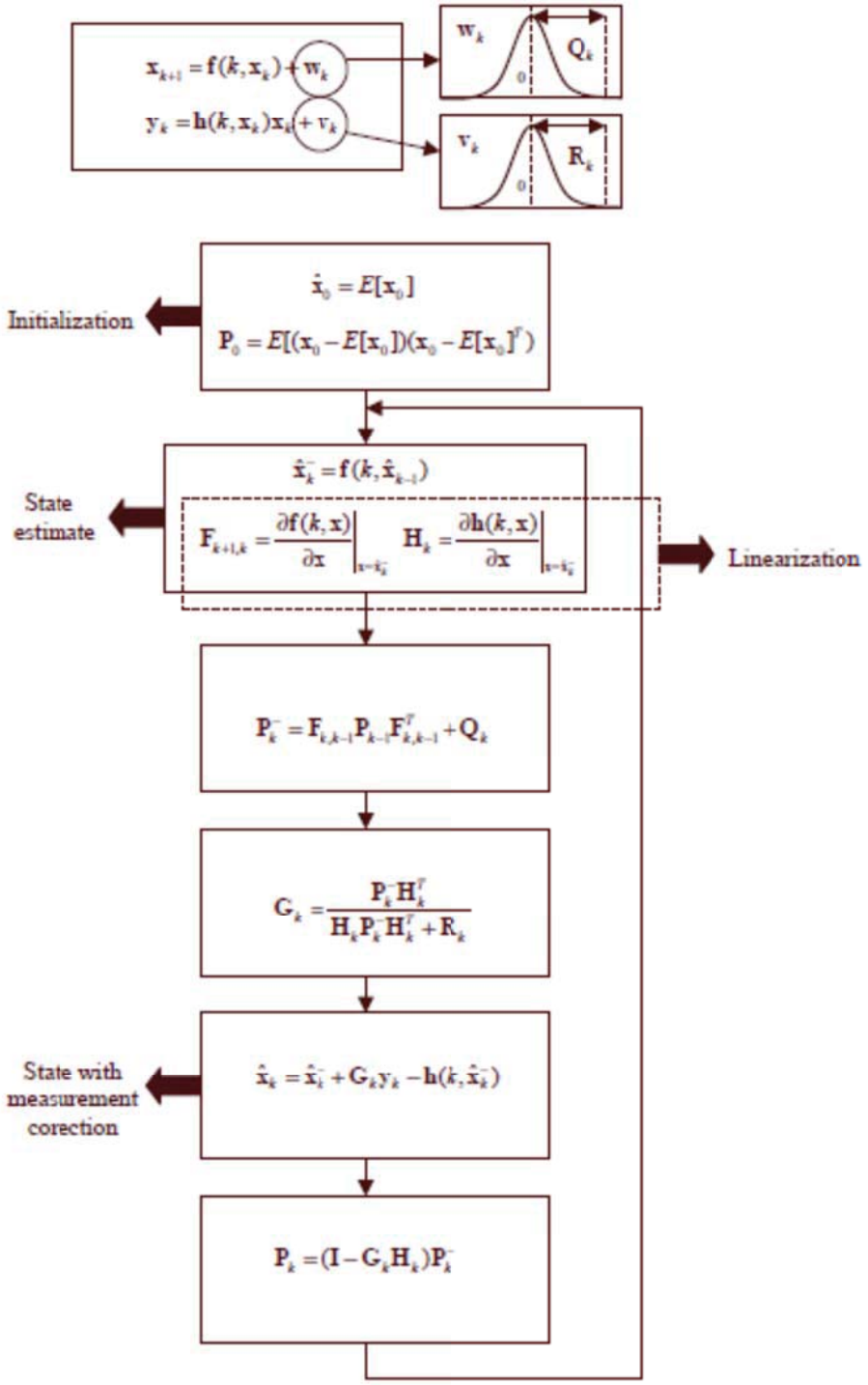
$$\bar{\mathrm{y}}_k \approx H_k x_k + v_k \qquad (۱۹-۲)$$

که در آن:

$$\bar{y}_k = y_k - \{h(x, \hat{x}_k^-) - H_k \hat{x}_k^-\} \qquad (۲۰-۲)$$

$$d_k = \mathrm{f}(\mathrm{x}, \hat{x}_k) - F_{k+1,k} \hat{x}_k \qquad (۲۱-۲)$$

تمامی درایه‌های $\bar{y}_k$ در زمان kام معلوم هستند و بنابراین، $\bar{y}_k$ می‌تواند به عنوان یک بردار رویت در زمان n در نظر گرفته شود. به همین ترتیب، تمامی درایه های d_k در زمان kام معلوم هستند.



شکل (۲-۴) خلاصه الگوریتم کالمن تعمیم یافته[۱۳]

با به دست آمدن روابط فضای حالت خطی‌سازی شده از روابط (۲-۱۹) و (۲-۲۰)، حال همان الگوریتم فیلتر کالمنی را که در مورد آن بحث شد، روی آنها اعمال می‌کنیم. شکل (۲-۴) خلاصه‌ای از الگوریتم حلقه بسته فیلتر کالمن تعمیم یافته را نمایش می‌دهد[۱۳].

۲-۱-۴ تشخیص آب و هوا

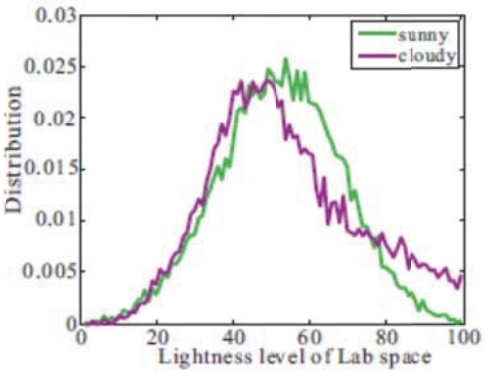
برای ایجاد قانون و محدودیت در عملکرد ربات‌ها در آب و هواهای مختلف، نیاز به تشخیص آب و هوا وجود دارد. درحال حاضر، تشخیص دقیق وضعیت آب و هوا با استفاده از سنسورها و سیستم‌های با هزینه‌های زیاد صورت می‌گیرد. از طرفی، تشخیص آب و هوا قرن‌هاست که متکی به بینایی انسان می‌باشد. (و انسان‌های مختلف دیدهای مختلفی دارند). بنابراین اگر بتوانیم از دوربین‌های عادی که درهمه جا یافت می‌شود بهره‌برداری نمائیم، امکان این وجود دارد که رصد و تشخیص وضعیت آب و هوا به یک کاربرد به صرفه و قدرتمند بینایی کامپیوتر تبدیل شود[۱۴].

با وجود ارزش قابل توجه آن، درک و تشخیص آب و هوا از یک تصویر واحد به طور کامل مطالعه نشده است. در منابع شماره[۱۵] و [۱۶] شناسایی وضعیت آب و هوا در خودروهای هوشمند به کار گرفته شده است.

روش‌های شناسایی صحنه بر روی اطلاعات ساختاری برای دسته‌بندی کردن صحنه‌ها به طبقات مختلف متکی است. اطلاعات ساختار یافته برپایه ویژگی‌های نور نامتغیر مانند SIFT و HOG هستند. نشانه‌های آب و هوا پیچیده‌تر هستند و صحنه‌های خاص نیستند، بنابراین روش‌های طبقه‌بندی صحنه‌های معمولی برای تشخیص آن قابل اجرا نیستند.

تصویربرداری در طول یک روز، ممکن است توسط دوربین‌های مختلف، وضعیت هوایی مختلف و در زمان‌های مختلف انجام شود. همانطور که در شکل (۲-۵) و (۲-۶) نمایش داده شده است، دسته‌بندی با استفاده از به ترتیب شدت روشنایی و یا رنگ و شدت رنگ در این زمینه، با شکست روبرو می‌شود.

شکل (۲-۵) a) تصویر روز آفتابی با نور متوسط b) تصویر روز ابری با نور متوسط[۱۴]



شکل (۲-۶) توزیع چگالی رنگ در تصاویر روز آفتابی و روز ابری (از روی این شکل مشخص است که تشخیص بین دو وضعیت آب و هوا از روی هیستوگرام غیر مکمن است) [۱۴]

بنابراین، جهت تشخصی صحیح وضعیت آب و هوا، نیاز به تعریف شاخص‌های هواشناسی وجود دارد. این شاخص‌ها که همان ویژگی‌های آب و هوایی هستند و انسان از طریق تشخصی آنها، وضعیت آب و هوا را معین می‌کند، می‌تواند پارامترهای زیر باشد:

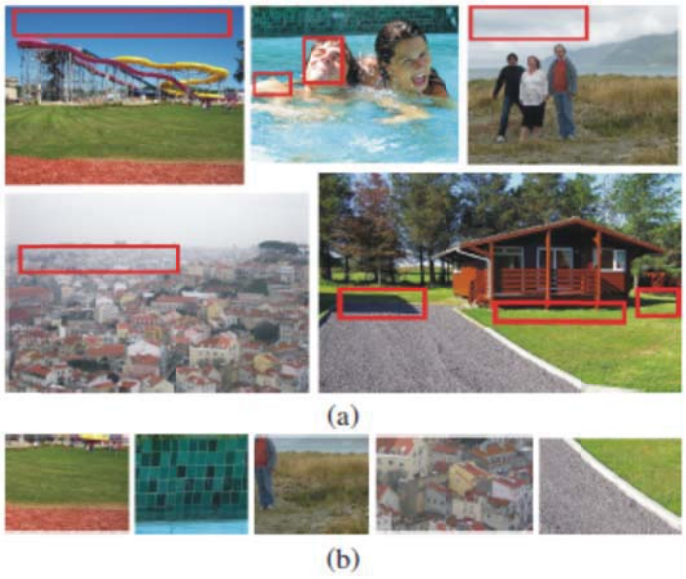
۱- آسمان: تشخیص آسمان در تصویر و تعیین وضعیت آن

۲- سایه: شدت کنتراست بین نقطه روشن و سایه آن در تصویر

۳- انعکاس: شدت انعکاس نور توسط سطوح در تصویر

۴- کنتراست

۵- مه: وجود و یا عدم وجود مه آلودگی در تصویر



شکل (۲-۷) نشانه‌های آب و هوایی a) نشانه‌های آب و هوایی عمومی در داخل مستطیل‌های قرمز b) نواحی مختلف از شکل (a) که در آن نشانه‌های آب و هوایی وجود ندارند [۱۴]

انسان هنوز با استفاده از این شاخص‌ها برای مشاهده و تشخیص آب و هوا استفاده می‌کند. به عنوان مثال آسمان مه آلود و یا مایل به خاکستری یک روز ابری را نشان می‌دهد، سایه شفاف و سخت بر روی زمین، یک روز آفتابی را نشان می‌دهد. و برعکس، همانطور که در شکل (۲-۷) نمایش داده شده است، در مواقعی که هیچگونه نشانه‌ای از وضعیت آب و هوا نداریم، حتی ما انسان‌ها نیز اعتماد به خودمان را در رابطه با نظر در مورد وضعیت آب و هوا از دست می‌دهیم.

با داشتن ویژگی‌های آب و هوایی، حال مشکل این است که چگونه یک دسته‌بند را آموزش دهیم. مشکل اساسی اینجاست که ممکن است همه‌ی ویژگی‌های هواشناسی را نتوان در همه تصاویر یافت

(به عنوان مثال، همه تصاویر بیرونی، دارای تصویری از آسمان نمی‌باشند) که این موضوع برای یادگیری دسته‌بندهای سنتی مانند ماشین‌های بردار پشتیبان [1]مشکل‌ساز می‌باشد. برای حل این مشکل، می‌توان از طریق چارچوب یادگیری ترکیبی با استفاده از رای دهندگان، تصاویر بیرونی را در داخل خوشه‌هایی که تصاویر در داخل آن از نظر نشانه‌های آب و هوا مشابه همدیگر هستند قرار دهیم. این کار به لطف همگنی در هر خوشه، ما را قادر می‌سازد، دسته‌بندهایی را به روش متعارف بسازیم. نتیجه نهایی برچسب گذاری، رای وزن‌دار منتجه از خروجی‌های دسته‌بند خوشه است. خوشه‌ای که به تصویر تستی نزدیک‌تر است، وزن بزرگتری می‌گیرد. همانطور که در بندهای فوق توضیح داده شد، رای دهندگان همگن با هم در همکاری با یکدیگر تحت یک چارچوب بهینه‌سازی واحد آموزش دیدند. با وجود نبود کار نمونه در برچسب‌گذاری آب و هوا از طریق تصویر، مقایسات کیفی بین روش پیشنهادی و روش‌های پیشین مثل SVM، Adaboost و غیره انجام شده است. برای این مقایسه، یک مجموعه از تعداد زیادی از تصاویر مربوط به وضعیت آب و هوا به طور کامل و دقیق در دو دسته جداسازی شده‌اند تا برای آموزش دسته‌بند و برچسب‌گذاری استفاده شود.

اولین سیستم پایه، اجرای دسته‌بند SVM به طور مستقیم بر روی ویژگی ۶۲۱ بعدی است. جفت دسته‌بند خطی و غیر خطی با هسته‌های مختلف تست شده است و بهترین نتایج در اینجا شرح داده شده است. دومین سیستم پایه، Adaboost سنتی است که ترکیبی از دسته‌بندهای متعدد برای ساخت دسته بند قوی‌تر می‌باشد. دو سیستم دیگر از دسته‌بندها که بر پایه یادگیری دیکشنری [۱۷] هستند، به نام های LLC [۱۸] و ScSPM [۱۹] هستند که نمونه‌ای از روش‌های دسته‌بندی تصاویر هستند. جدول (۲-۲) نتیجه دسته‌بندی را نمایش می‌دهد.

[1] Support Vector Machine (SVM)

جدول (۲-۲) نتایج دسته بندی با استفاده از روش های دسته بندی مختلف[۱۴]

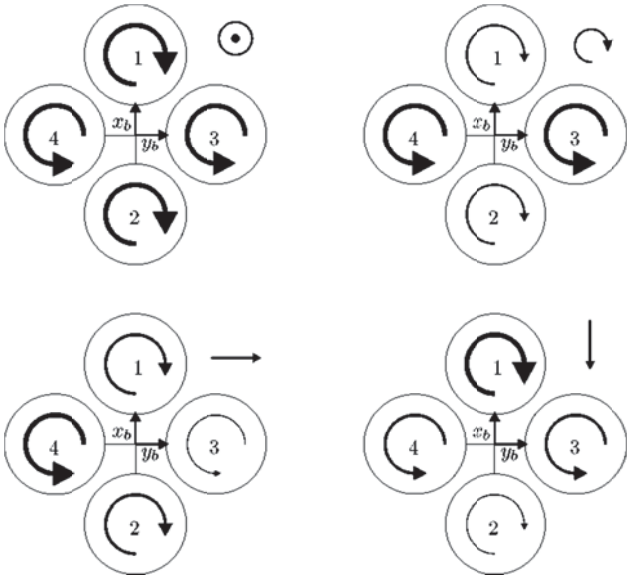
	SVM	Adaboost	LLC	ScSPM	الگوریتم جدید
دقت نرمالیزه شده	۴۱.۲±۲.۲	۳۶.۴±۲.۳	۰.۳±۰.۱	۰.۲±۰.۱	۵۳.۱±۲.۲

همانطور که از جدول (۲-۲) استنتاج می‌شود، الگوریتم معرفی شده، در مقایسه با سایر الگوریتم‌های دسته‌بندی ویژگی‌ها، از عملکرد بسیار بهتری برخوردار است. به گونه‌ای که دقت نرمالیزه شده این الگوریتم، از الگوریتم SVM با تنظیمات مذکور که بهترین نتیجه را پس از این الگوریتم در بین سایر الگوریتم‌ها داشته است، حدود ۱۲ درصد بهتر عمل کرده است.

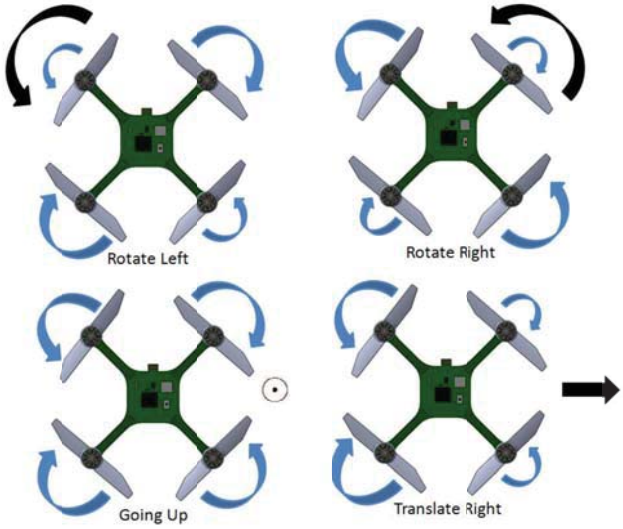
۲-۲ ساختار کوادکوپترها

در این بخش، ساختار فیزیکی کوادکوپتر که یکی از انواع پهپادهای عمودپروازها می‌باشد، بیان می-شود.

کوادکوپتر یک پهپاد عمودپرواز بدون سرنشین است که دارای چهار عدد ملخ بوده و سازه آن به شکل علامت جمع "+" (شکل ۲-۸) یا علامت ضربدر "X" (شکل ۲-۹) می‌باشد و در انتهای هر یک از بخش‌های سازه یک سیستم پیشران قرار گرفته است که به وسیله تغییر دور موتورها، مانور می‌دهد. تفاوت کنترلی در این دو نوع پیکربندی این است که در طرح ضربدری برای چرخش حول هرکدام از محورهای X و یا Y هر چهار موتور درگیر می‌شوند. این نحوه به‌کارگیری موتورها باعث می‌شود که نیروی موردنیاز برای چرخش حول محور X و Y و در نتیجه حرکت در جهت‌های مختلف بین چهار موتور تقسیم شود و درنتیجه موتورها از سطح اشباع خود دورتر می‌شوند. درحالی‌که در ترکیب بعلاوه، همان مقدار نیرو بین دو موتور تقسیم می‌شود. البته باید توجه داشت که در پیاده‌سازی کنترل قطعاً هماهنگ کردن دو موتور، ساده‌تر از هماهنگ کردن چهار موتور می‌باشد[۲۰].



شکل (۲-۸) ترکیب بعلاوه کوادکوپتر [۲۱]



شکل (۲-۹) ترکیب ضربدری کوادکوپتر[۲۲]

نیروی تراست هر یک از موتورها به‌طور مجزا قابل‌کنترل است و دور موتورها بر اساس برنامه هدایت و کنترل تعیین می‌شود. نیروی تراست موردنیاز از طریق انتقال نیروی موتور به ملخ‌ها تولید می‌شود[۲۱].

کوادورتور به عنوان یک بسته پروازی کامل طراحی‌شده است. بخش کنترلی که در وسط پرنده قرار دارد، درواقع مغز و سیستم پردازش و کنترل کوادکوپتر است. سیستم کنترل از راه دور، سیستم پایدارسازی و سیستم نیرومحرکه در این صفحه قرار داده‌شده‌اند، سه بخش مهم کوادکوپتر هستند.

دو نسل طراحی برای کوادروتوها موجود است[۲۱]:

- نسل اولیه تولید کوادکوپتر که برای حمل یک یا دو مسافر طراحی‌شده بود. با این وجود الگوهای اولیه از ضعف عملکرد رنج می‌برد که به علت عدم کنترل مناسب و مصرف سوخت بالا عملیاتی نشدند. به همین جهت این طرح‌ها جای خود را به طرح‌های کوچک‌تر امروزی دادند.
- نسل طراحی امروزی کوادورتور بدون سرنشین می‌باشد که از یک سیستم کنترلی الکترونیکی و سنسورهای الکترونیکی برای پایدارسازی استفاده می‌کنند. اندازه کوچک آن‌ها و مانور پذیری بالای آن‌ها موجب پرواز این وسیله در محیط‌های بسته و بازشده است.

کوادکوپتر ابتدا در سال ۱۹۰۷ توسط دو برادر به نام‌های لوئیس برگویت[۱] و جاکوئیس برگویت[۲] و تحت راهنمایی‌های پروفسور چارلز ریچارت[۳] ساخته شد و به‌عنوان اولین کوادروتور، به نام جایروپلن[۴] نام‌گذاری شد، اما در آن زمان پرواز با موفقیت همراه نشد[۲۱].

امروزه با توسعه علم و فنّاوری و مینیاتوری شدن و سبک شدن سیستم‌های مختلف پروازی امکان ساخت ریزپرنده‌ها در مقیاس کوچک امکان‌پذیر شده است. کوادکوپتر نیز به دلیل قابلیت مانور پذیری و نیز قابلیت پرواز در محیط‌های بسته، موردتوجه قرار گرفته است.

[1] Leis Breguet
[2] Jacques Bregoet
[3] Churls Richart
[4] Gyroplane

تحقیقات علمی و تکنیکی کوادکوپترها امروزه بیشتر حول افزایش مانور پذیری، اکتشاف، افزایش زمان پروازی، افزایش بارگذاری بر روی کوادکوپتر، جابجایی اجسام بزرگ و روبات‌های همکار بررسی می-شود.

مزایا و معایب استفاده از کوادکوپتر

مزایا:

۱- قابلیت مانورپذیری بسیار بالا

۲- سادگی ساخت و هزینه کمتر تعمیر و نگهداری

۳- انجام پرواز به‌صورت عمودپرواز

۴- انجام مأموریت با حداقل یک نفر در هر جا بدون محدودیت

۵- یادگیری آسان پرواز

۶- سهولت انتقال اطلاعات به‌صورت تصویری

۷- انجام پرواز ناوبری و حمل تجهیزات شامل دوربین و یا تجهیزات دیگر اضافی

۸- امکان فیلم برداری در مکان‌هایی که امکان تردد نیروی انسانی وجود ندارد.

۹- صدای کم پرنده نسبت به پرنده‌های بدون سرنشین

۱۰- ارتباط رادیویی و کنترل وسیله توسط ایستگاه زمینی با استفاده از فرکانس اچ‌اف[1] به‌صورت کاملاً واضح و شفاف.

۱۱- توانایی و امکان انجام شناسایی محیط پیرامون و انتقال مستقیم اطلاعات تصویری از رویدادها در زمان امداد و نجات زمینی و هوایی.

۱۲- توانایی شناسایی و تصویربرداری به‌صورت زنده از سطح زمین و یا دیگر وسایل پرنده در حین پرواز.

[1] HF

معایب:

۱- مداومت پروازی پایین. از آنجا که باتری نصب‌شده‌ای که بر روی آن قرار گرفته است، در زمان پرواز هر چهار موتور را تغذیه می‌نماید و به همین دلیل مدت پرواز آن محدود می‌باشد. امروزه برای برطرف نمودن این عیب در طراحی‌های جدید، از یک سری باتری مخصوص استفاده می‌کنند. این باتری‌ها با استفاده از ژنراتورهای مجزا روی هر موتور کوادکوپترها، در حین پرواز شارژ می‌گردند، لذا این محدودیت درحال حاضر قابل برطرف کردن است[۲۲].

۲- با از کار افتادن یکی از موتورها یا شکستن یکی از ملخ‌ها، پایداری کوادکوپتر با مشکل روبرو می‌شود. درصورتی که هگزاروتورها و اکتاروتورها این مشکل را تا حد زیادی برطرف کرده‌اند.

۲-۳ مدلسازی و کنترل کوادکوپتر

در اینجا ابتدا به ساختار کلی کوادکوپتر پرداخته و سپس نحوه کنترل پرواز و پایداری آن مورد بحث قرار می‌گیرد.

در یک سیستم همانند بالگرد، نیروی بالا برنده توسط ملخ اصلی تولید می‌شود. به همراه آن گشتاوری حول محور عمودی تولید می‌شود که منجر به چرخش آن حول آن می‌شود. جهت جلوگیری از این امر، روتور انتهایی بالگرد به چرخش در می‌آید و گشتاوری در خلاف گشتاور روتور اصلی ایجاد می‌کند و آن را خنثی می‌کند[۲۳].

برای یک ربات پرنده که با چهار ملخ کار می‌کند، چهار ورودی برای کنترل روتور وجود دارد. نیروهای محرکه و گشتاوری که به ربات اعمال می‌شود، از برآیند نیروهای ناشی از اختلاف سرعت چرخش روتورها محاسبه می‌شود. همانطور که در شکل (۲-۹) نمایش داده شده است، در اکثر کوادکوپترها روتورهای ۱ و ۲ با ۳ و ۴ در جهت مخالف یکدیگر چرخش می‌کنند تا گشتاورهای حاصل همدیگر را بتوانند در سرعت یکسان خنثی نمایند. بنابراین به غیر از حرکات در جهت افقی، چهار حالت برای

حرکت چرخشی ربات ممکن است[۲۱]. کاهش و یا افزایش سرعت چرخش همه روتورها به یک میزان ثابت، باعث حرکت در محور عمودی می‌شود (شکل ۲-۹ بالا، چپ). تغییرات نسبی سرعت روتورهای ۱ و ۲ باعث ایجاد چرخش حول محور yها می‌شود[1] که منجر به انحراف ربات به سمت جلو و یا عقب می‌گردد. البته حرکت در صفحه افقی نخواهد بود و ارتفاع ربات نیز تغییر خواهد کرد (شکل ۲-۹ پایین، راست). تغییرات نسبی سرعت روتورهای ۳ و ۴ باعث ایجاد چرخش حول محور xها می‌شود[2] که منجر به انحراف ربات به سمت چپ و یا راست می‌شود. در اینجا نیز حرکت در صفحه افقی نخواهد بود و ارتفاع ربات نیز تغییر خواهد کرد (شکل ۲-۹ پایین، چپ). تغییرات نسبی سرعت روتورهای ۳ و ۴ در مقابل روتورهای ۱ و ۲ باعث ایجاد غیر صفر شدن گشتاور برآیند شده و منجر به چرخش ربات حول محور zها می‌شود[3] (شکل ۲-۹ بالا، راست).

جهت تسهیل در طراحی کنترلر کوادکوپتر، مدلسازی آن امری اجتناب ناپذیر است. در ادامه به مدلسازی سینماتیکی کوادکوپتر پرداخته می‌شود.

۲-۳-۱ مدلسازی کوادکوپتر

قبل از شروع مدلسازی ربات، باید برای مدل دینامیکی آن فرضیاتی در نظر گرفته شوند. این فرضیات بر اساس طبیعت پرواز و ساختار ربات پرنده کوادکوپتر مطرح می‌شوند. مندس[4] در پایان نامه کارشناسی ارشد خود فرضیات ذیل را برای مدلسازی کوادکوپتر در نظر گرفته است[۱]:

- کوادکوپتر دارای ساختار صلب است و بدنه آن به اندازه کافی مستحکم می‌باشد.
- ساختار کوادکوپتر کاملا متقارن است. بنابراین درایه‌های غیر قطری ماتریس اینرسی آن همگی صفر هستند.

[1] pitch
[2] roll
[3] yaw
[4] Mendes

- به دلیل پایین گرفته شدن سرعت چرخش ملخ‌ها، از پدیده فلپینگ ملخ‌های ربات صرف نظر می‌شود.
- نیروهایی که ملخ‌ها جهت پرواز ایجاد می‌کنند کاملاً عمودی است.
- کوادکوپتر دارای وزن ثابتی است.
- در پرواز، کره زمین ثابت و مسطح در نظر گرفته می‌شود. (به دلیل کم بودن مدت زمان پرواز و سرعت پایین ربات).
- از اثرات زمین صرف‌نظر می‌شود.
- مرکز جرم و مبدأ محور مختصات بدنی، منطبق بر هم در نظر گرفته می‌شوند.
- تراست و نیروی پسای ایجادشده توسط ملخ‌ها نسبتی از مربع سرعت زاویه‌ای ملخ در نظر گرفته می‌شوند.

همین محقق برای مدلسازی کوادکوپتر ابتدا دستگاه مختصات مرجع را تعریف کرده و از نمایش زوایای اویلر جهت تعریف نگرش زاویه‌ای ربات استفاده کرده است. سپس بر اساس فرضیات فوق روابط سینماتیکی و دینامیکی کوادکوپتر را مورد تحلیل قرار داده و نیروها و گشتاورهای خارجی اعمال شده به کوادکوپتر را ارائه نمود. همچنین پس از در نظر گرفتن نیروی رانشی و هاب حاصل از چرخش ملخ‌ها، گشتاور دراگ حاصل از نیروی دراگ روی روتور و گشتاور Rolling مرتبط با سرعت جانبی کوادکوپتر را ارائه نمود. ایشان تاثیر ارتفاع ربات از سطح زمین بر نیروی بالا برنده، نیروهای اصطکاک و اثر Gyro را در مدلسازی مد نظر قرار داد. در نهایت معادلات حرکت مورد نیاز برای بیان دینامیک حرکت کوادکوپتر نظیر موقعیت، سرعت، زوایای اویلر و سرعت زاویه‌ای را بدست آورد.

با انجام مراحل اشاره شده رابطه سینماتیکی حرکت کوادکوپتر به دست می‌آید. اما در مورد اینکه سنسورها و عملگرها (موتورها) چه رفتاری دارند، بحث نشده است. همچنین رفتار دینامیکی محیط اطراف ربات مدلسازی نشده است. برای به دست آمدن نتیجه‌ای مطلوب در پایداری و مانورپذیری بیشتر ربات پرنده، لازم است موارد فوق به طور دقیق مد نظر قرار گیرد.

در ادامه، در مورد انواع روش‌ها و الگوریتم‌های کنترلی پیاده سازی شده برروی ربات کوادکوپتر بحث خواهد شد و با مقایسه بین انواع کنترلرهای موجود، بهترین آنها با استناد به مراجع ارائه خواهد شد.

۲-۳-۲ کنترل کوادکوپتر

از آنجا که اینگونه ربات‌ها زیرفعال[1] هستند و دارای رفتاری شدیداً غیر خطی می‌باشد، کنترل آن یک کار تجربی و به همراه آزمون و خطا می‌باشد[۱]. کنترل کننده‌های مختلفی برای این ربات در سالیان اخیر طراحی و مورد استفاده قرار گرفته است. در مرجع [۲۴] کنترلری بر مبنای یادگیری ماشین برای کوادکوپتر طراحی شده است.

۲-۳-۲-۱ کنترل کننده‌های خطی

در زمینه کنترل کننده‌های خطی، کنترل کننده تناسبی-انتگرالی-مشتقی[2] و رگولاتور درجه دو خطی[3] دوتا از پرکاربردترین کنترل کننده‌ها و از قدیمی‌ترین آنهاست. از این کنترلر در کنترل کوادکوپتر نیز استفاده شده است[۲۵].

در ادامه، سه نوع از کنترل کننده‌های خطی مورد بررسی قرار می‌گیرد.

- کنترل کننده PID که ضرایب تناسبی(K_p)، انتگرالی(K_i) و مشتقی(K_d) آن با استفاده از روش ITAE به دست آمده است.
- کنترل کننده LQR کلاسیک
- کنترل کننده PID که ضرایب آن با استفاده از حلقه LQR به دست می‌آید

[1] Underactuated robot

[2] PID

[3] LQR: Linear Quadratic Gaussian

در اینجا، کنترل کننده PID به دلیل تطبیق‌پذیری و اجرای آسان و در عین حال ارائه پاسخ ثابت، کنترل کننده LQR به دلیل مقاوم بودن نسبی و پاسخ سریع، و کنترل کننده PID که ضرایب آن با استفاده از LQR تنظیم شده است، مورد بحث و بررسی قرار می‌گیرد.[۲۵].

با مقایسه عملکرد سه کنترلر در جابجایی، مشخص است که کنترلر LQR-PID از باقی کنترلرها سریع‌تر عمل کرده است. اما در بخش کنترل جابجایی عمودی، کنترلر LQR-PID بسیار کندتر از کنترلر PID عمل کرده است[۲۵]. با اینحال از آنجا که در محیط واقعی با مجموعه‌ای از پارامترهای مزاحم و محیطی روبرو هستیم و اثرات شدید نویز و عدم قطعیت‌ها در اینجا در نظر قرار گرفته نشده است، نمی‌توان از روی نمودارهای فوق، تشخیص داد که کدام‌یک بهترین عملکرد را در عمل خواهد داشت. بنابراین روش LQR-PID که از نوع کنترلر مقاوم است، در شرایط واقعی بهتر از PID عمل خواهد کرد و گزینه مناسب‌تری برای کنترل کوادکوپتر خواهد بود[۲۵].

۲-۳-۲-۲ کنترل کننده‌های غیر خطی

در زمینه کنترل کننده‌های غیر خطی سیستم‌های دینامیکی، دو نوع از پرکاربردترین کنترل کننده‌ها مورد بحث قرار می‌گیرد که به شرح زیر هستند:

۱- کنترل پسگام

۲- کنترل کننده مد لغزشی

کنترل پسگام روشی مبتنی بر لیاپانوف است که برای طراحی کنترل سیستم‌های غیر خطی استفاده می‌شود[۲۶]. در ادامه سه روش کنترلی مرتبط با این کنترل کننده، با یکدیگر مقایسه می‌شوند.

- کنترل کننده PID با سیستم خطی سازی شده
- کنترل کننده پسگام
- کنترل کننده پسگام افزایشی[1] [۲۶]

[1] Incremental Backstepping method

با مقایسه عملکرد در شرایط اسمی یکسان و با فرض اینکه مدل کوادکوپتر به خوبی تعیین شده باشد، همه کنترل کننده‌ها به مانند یکدیگر عمل می‌کنند[۲۶]. دقت شود که در کنترل موقعیت ربات اعم از زوایای اویلر و موقعیت ۳ بعدی ربات، مقدار ناچیزی جهش اضافی دیده می‌شود. این جهش به خاطر این است که زمان پاسخگویی پایین‌تر از سیستم به دست آید. نتیجه این کار پاسخ با میرایی سریع و کارایی کلی بهتر می‌باشد[۲۶].

۲-۳-۲-۳ کنترل مد لغزشی[1]

از دیگر کنترل کننده‌های غیر خطی به کار برده شده برای کنترل کوادکوپتر، کنترل کننده مد لغزشی است [۲۷]. با مقایسه نتایج شبیه‌سازی عملکرد کوادکوپتر با استفاده از دو کنترل کننده پسگام و مد لغزشی در شرایط مشخص، بسیار نزدیک به یکدیگر است[۲۷].

۲-۴ تلفیق حسگرها[2]

در ادامه، در مورد تلفیق داده‌های حسگرها، دلیل انجام این کار و نحوه انجام آن به طور خلاصه بیان می‌شود.

۲-۴-۱ معرفی تلفیق حسگرها

حیوانات محیط زیست خود را با ارزیابی سیگنال‌های دریافتی از حسگرهای متعدد و چند وجهی می‌شناسند. یکپارچه‌سازی اطلاعات از منابع مختلف جهت شناخت قابل اعتماد یک فرضیه در طبیعت است. حتی در صورت عدم وجود حسگر، سیستم‌ها قادر به جبران اطلاعات با استفاده مجدد از اطلاعات به دست آمده از حسگرها با دامنه تداخل هستند. برای مثال، انسان با ترکیب سیگنال‌های

[1] Sliding Mod control
[2] Sensor fusion

حواس پنج‌گانه قادر به تهیه یک مدل پویا از جهان هستی برای شناخت محیط زیست است. بر اساس این اطلاعات است که افراد در تعامل با محیط زیست، تصمیم گیری کرده و عملکرد آینده خود را مشخص می‌کنند[۲۸].

این توانایی طبیعی ترکیب حس‌های مختلف در بسیاری از گونه‌های جانوری تکامل یافته، برای میلیون‌ها سال در حال استفاده می‌باشد. امروزه کاربرد مفاهیم ترکیب و تلفیق در بسیاری از حوزه‌های تخصصی شکل گرفته است.

۲-۴-۲ اصول تلفیق حسگرها

نوعی سردرگمی در تعریف سیستم‌های تلفیقی وجود دارد. واژه‌هایی نظیر "تلفیق حسگر"، "تلفیق داده"، "تلفیق اطلاعات"، "تجمیع چند حسگری" به طور گسترده‌ای در ادبیات تخصصی استفاده شده است. این واژه‌ها برای اشاره به انواع مختلفی از تکنیک‌ها، فن‌آوری‌ها، سیستم‌ها، و برنامه‌های کاربردی با استفاده از اطلاعات بدست آمده از چند منابع اطلاعاتی بکار می‌روند. دامنه کاربرد تلقیق از تلقیق برخط حسگرها برای ربات‌های سیار ناوبری تا تلقیق برون خط داده‌های اطلاعاتی استراتژیک یا انسانی بسیار وسیع می‌باشد[۲۹].

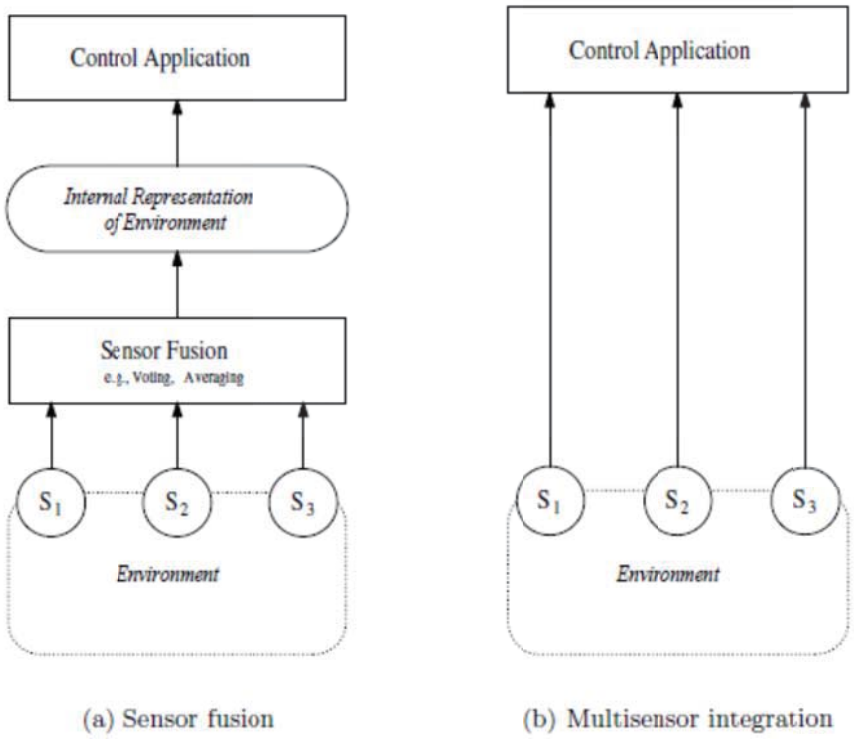
تلاش‌های متنوعی شده است تا تعریف و دسته‌بندی واضحی از واژه تلفیق و تکنیک‌های آن ارائه گردد[۲۹]. والد[1] پیشنهاد می‌کند که واژه "تلفیق داده" بعنوان واژه‌ای کلی برای تلفیق بکار رود. بهرحال، با اینکه فهم واژه تلفیق داده بسیار آسان است اما معنی دقیق آن برای افراد متفاوت فرق می‌کند. متاسفانه این واژه در سال‌های گذشته همیشه به یک معنی استفاده نگردیده است. در بعضی از مدل‌های تلفیق، واژه "تلفیق داده" به معنی تلفیق داده‌های خام استفاده می‌گردد[۳۰].

[1] Wald

کتاب‌هایی در زمینه تلفیق نظیر "تلفیق داده‌های چند حسگری"[۳۱] توسط والتز و لیناس[1] و "روش‌های ریاضی در تلفیق داده‌های چندحسگری"[۳۲] توسط هالز[2] واژه "تلفیق داده‌های چند حسگری" را پیشنهاد می‌دهند. تعریف این واژه اینگونه است: "تکنولوژی چگونگی تلفیق داده‌ها از حسگرهای مختلف بمنظور استنتاج در باره یک پدیده فیزیکی، یک فعالیت یا موقعیت"[۲۹]. برای پیشگیری از اشتباه در معنی تلفیق، دساراتی[3] تصمیم می‌گیرد که از واژه "تلفیق اطلاعات" بعنوان واژه کلی برای تلفیق تمامی انواع داده‌ها استفاده کند[۳۳]. بعنوان زیرمجموعه‌ای از تلفیق اطلاعات، تلفیق حسگرها عبارت است از ترکیب داده‌های حسگرهای مختلف به منظور بدست آوردن اطلاعاتی در مورد یک پدیده، و بهتر از حالتی که توسط هر حسگر بطور مستقل می‌توان بدست آورد.

واژه دیگری که عموماً استفاده می‌گردد واژه "تجمیع چند حسگر" است. تفاوت بین تلفیق حسگر و تجمیع چند حسگر در شکل (۲-۱۰) نشان داده شده است. دایره‌های S_1، S_2 و S_3 حسگرهای فیزیکی را نشان می‌دهد که یک رابطه برای پردازش محیط را ایجاد می‌کنند. شکل (۲-۱۰)(a) نشان می‌دهد که داده حسگر بوسیله بلوک تلفیق به متغیرهای پردازش محیط تبدیل شده است.

[1] Waltz and Llinas
[2] Hall's
[3] Dasarathy



شکل (۲-۱۰) دیاگرام تلفیق حسگر و مجموعه چندحسگری [۳۴]

این داده‌ها سپس توسط برنامه‌های کنترل مورد استفاده قرار می‌گیرد. برعکس، شکل (۲-۱۰) (b)معنی تجمیع چند حسگر را توضیح می‌دهد که داده‌های حسگرهای مختلف مستقیما توسط برنامه کنترل پردازش می‌گردد.

۲-۴-۳ انگیزه تلفیق حسگرها

سیستم‌هایی که تلفیق حسگرها را بکار می‌برند انتظار دارند تا بتوانند منافع بیشتری از سیستم‌های تک حسگری بدست بیاورند. یک حسگر فیزیکی معمولاً با مسائلی نظیر خرابی حسگر، محدودیت پوشش فضایی، محدودیت پوشش زمانی، دقت کم و عدم قطعیت مواجه است. بعنوان مثال، در نظر بگیرید که یک حسگر فاصله‌یاب در عقب یک خودرو نصب شده است تا به راننده به هنگام پارک

کردن کمک کند. این حسگر تنها می‌تواند اطلاعات در زمینه اشیای قرار گرفته در جلوی آن را بدهد اما در مورد اشیایی کناری اطلاعاتی نمی‌دهد. بنابر این پوشش فضا محدود شده است. [۳۴]

مزایایی نظیر قدرتمندی و قابلیت اطمینان، پوشش زمانی و فضایی، افزیش اعتماد، کاهش ابهامات و عدم قطعیت، مقاوم در برابر تداخل و وضوح بهبودیافته را می‌توان برای تلفیق داده‌های حسگرها برشمرد. علاوه بر این موارد، یکی دیگر از مزیت‌های تلفیق حسگرها کاهش پیچیدگی سیستم‌ها می‌باشد. [۳۴]

۲-۵ معرفی محیط نرم‌افزاری

در این قسمت، به چند نمونه از نرم‌افزارها و ابزارهایی که در طول سال‌های گذشته به منظور شبیه‌سازی عملکرد ربات و توسعه الگوریتم‌ها طراحی شده‌اند، اشاره می‌شود. به طور کلی، نرم افزارهای مورد نیاز در این پروژه، در زمینه‌های زیر به کار برده می‌شوند:

۱- استفاده از سیستم عامل ربات[1] جهت ماژولار کردن برنامه‌ها

۲- طراحی کنترل کننده و شبیه‌سازی عملکرد آن

۳- طراحی برنامه پردازش تصویر و شبیه‌سازی آن

در ادامه، نرم افزارهایی که در سه بخش فوق به کار برده می‌شوند، معرفی می‌گردند.

۲-۵-۱ استفاده از سیستم عامل ربات جهت ماژولار کردن برنامه‌ها

ROS یک پلتفرم برای توسعه کاربردی ربات است که ویژگی‌هایی همچون تبادل پیام، محاسبات پراکنده، استفاده مجدد از کد و غیره را دارد.

[1] Robot Operating System (ROS)

پروژه ROS در سال ۲۰۰۷ میلادی، به عنوان بخشی از پروژه ربات Stanford STAIR، تحت عنوان Switchyard آغاز به کار کرد و انجمن توسعه دهندگان ROS به سرعت توسعه پیدا کرد و امروزه در سراسر دنیا کاربران و توسعه دهندگان بسیاری از این پلتفرم استفاده می‌کنند. بسیاری از شرکت‌های بزرگ رباتیک، در حال حاضر ربات‌های خود را به توانایی استفاده از ROS مجهز کرده‌اند. این روند شامل ربات‌های صنعتی که از نرم افزارهای اختصاصی شرکت سازنده نیز استفاده می‌کردند، شده است.

۲-۵-۱-۱ دلیل ترجیح دادن ROS برای کارهای رباتیکی

تصور کنید که قصد ساخت ربات قابل حمل خودمختار را دارید. در ادامه، چند دلیل که مهندسان بر استفاده از این پلتفرم به جای استفاده از پلتفرم‌های دیگر نظیر Player، YARP، Orocos، MRPT و غیره دارند، ذکر می‌شود:

- قابلیت‌های باارزش: به همراه ROS قابلیت‌های با ارزش آماده به کار نظیر بسته[1]های SLAM و مکان‌یابی مونت کارلو تطبیقی[2] جهت پیاده‌سازی ناوبری خودمختار ربات قابل حمل، و بسته MoveIt برای برنامه‌ریزی حرکت محرک‌های ربات وجود دارند. این قابلیت‌ها، می‌توانند به طور مستقیم و بدون هیچ دردسری استفاده شوند. همه بسته‌های موجود در پلتفرم ROS در بهینه‌ترین شکل ممکن خود هستند و این امکان نیز برای کاربر وجود دارد که بر اساس نیازهای خود، تغییراتی را نیز در بسته‌ها ایجاد کنند.
- ابزارهای فراوان: ROS به همراه ابزارهای فراوانی جهت اشکال‌زدایی[3]، تجسم سه بعدی و انجام شبیه‌سازی‌ها ارائه شده است. ابزارهایی همانند rqt_gui، RViz و Gazebo به ترتیب، ابزارهای قدرتمند و متن باز در زمینه اشکال‌زدایی، تحسم سه‌بعدی و شبیه‌سازی هستند. چارچوب[4] نرم‌افزاری که شامل همه این موارد باشند، بسیار نادر می‌باشد.

[1] Package
[2] Adaptive Monte Carlo Localization (AMCL)
[3] Debugging
[4] Framework

- پشتیبانی از محرک‌ها و حسگرهای رده بالا: ROS به همراه درایورهای دستگاه و بسته‌های رابط حسگرها و محرک‌های متنوع مورد استفاده در رباتیک ارائه شده است. حسگرهای رده بالا شامل Velodyne-LIDAR، اسکنرهای لیزری، کینکت[1] و غیره می‌باشد. محرک‌ها شامل موتور سروو های Dynamixel و غیره می‌باشد. کاربران می‌توانند با استفاده از پلتفرم ROS، بدون هیچ دردسر و زحمتی از این دسته محرک‌ها و حسگرها را به کار ببرند.
- عملکرد بین پلتفرمی: قابلیت تبادل پیام موجود در ROS، امکان برقراری ارتباط با گره[2]های مختلف را در لحظه می‌دهد. این گره‌ها می‌توانند به زبان‌های مختلف برنامه‌نویسی همانند C++، C، Python و Java نوشته شوند. قابل ذکر است که از سال ۲۰۱۵ میلادی، نرم افزار محاسبات ماتریسی Matlab نیز به این مجموعه پیوسته است. اینگونه انعطاف‌پذیری در هیچ پلتفرم دیگری یافت نمی‌شود.
- ماژولار بودن[3]: یکی از مسایلی که ممکن است در هر پروژه رباتیکی مستقل بروز کند، این است که در مواقعی که کد برنامه اصلی با مشکل روبرو می‌شود، کل عملکرد ربات متوقف می‌شود. در این مورد نیز، ROS متفاوت عمل می‌کند. کاربر می‌تواند برای هر عملکرد ربات، به صورت کاملا مجزا برنامه نویسی کند و در این صورت، با بروز مشکل در یکی از برنامه‌ها، باقی برنامه ها به کار خود می‌توانند ادامه دهند. همچنین ROS این امکان را فراهم کرده است که در صورت فقدان سیگنال حسگر یا عملگر، عملیات خود را ادامه دهد.
- استفاده همزمان از منابع: استفاده همزمان بیش از دو برنامه از منابع سخت افزاری همیشه مشکلاتی را برای برنامه‌نویسان داشته است. فرض کنید، قصد پردازش تصاویر گرفته شده از دوربین برای آشکارسازی چهره و ردیابی دارید. می‌توان تمام برنامه را به صورت یکجا نوشت یا به صورت تک رشته‌ای این کار را انجام داد. اما در صورتی که قصد استفاده از بیش از چند

[1] Microsoft Xbox Kinect
[2] Node
[3] Modularity

رشته به طور همزمان را داشته باشیم، رفتار برنامه پیچیده شده و اشکال زدایی آن مشکل می‌شود. اما در ROS، می‌توان از طریق موضوعات[1] ROS به منابع سخت‌افزاری و نرم‌افزاری به طور همزمان دسترسی پیدا کرد. هر تعداد از گره‌های مرتبط با ROS، از این طریق می‌تواند به طور همزمان تصویر دوربین را دریافت کند و بر روی آن عملیاتی را انجام دهد. این کار باعث کاهش پیچیدگی در محاسبات و اشکال‌زدایی ساده کل سیستم می‌شود.

- داشتن جامعه فعال برنامه‌نویسان: وقتی ما قصد استفاده از کتابخانه[2] یا Framework را مخصوصاً از منابع متن باز داریم، یکی از فاکتورهای مهم در انتخاب آنها، پشتیبانی نرم‌افزاری و جامعه توسعه دهندگان فعال است. هیچ پشتوانه و دلیلی برای استمرار پشتیبانی از سوی ابزارهای متن باز وجود ندارد. در ROS جامعه پشتیبانی بسیار فعال است و قسمت پرسش و پاسخ فعال مخصوص کاربران عادی نیز وجود دارد.

علاوه بر نکات اشاره شده، دلایل مختلف دیگری نیز جهت ترجیح دادن ROS به سایر نرم افزارها و پلت‌فرم‌های دیگر وجود دارد که در اینجا به آنها اشاره نشده است. در این بخش، مزایای استفاده از ROS مورد بحث قرار گرفت. در ادامه، دلایلی که کاربران را از استفاده از ROS منصرف می‌کنند آورده می‌شود:

- یادگیری دشوار: یادگیری ROS از طریق صفحات wiki پیش‌فرض دشوار می‌باشد. بسیاری از کاربران، برای شروع یادگیری ROS، به کتاب مراجعه می‌کنند. با این وجود، کتاب‌هایی با تعداد صفحات ۵۰۰ صفحه نیز وجود دارد که تنها به مقدمات کاربردی استفاده از ROS اشاره کرده است. بنابراین، یادگیری کامل ROS، دشوار می‌باشد.
- دشواری در شروع شبیه‌سازی: شبیه‌ساز پیش‌فرض ROS نرم‌افزار Gazebo است. اگرچه این نرم‌افزار شبیه‌سازی را خوب انجام می‌دهد، ولی شروع به کار با آن کار ساده‌ای نمی‌باشد. این

[1] Topics
[2] library

شبیه‌ساز، در خود بخشی برای برنامه‌نویسی برای حرکت‌دادن ربات را ندارد. تنها راه برای شبیه‌سازی و حرکت‌دادن ربات در آن، استفاده از ROS است. با مقایسه بین Gazebo با دیگر نرم افزارهای شبیه سازی همانند Webots و V-REP، متوجه می‌شویم که این نرم افزارها قابلیت برنامه‌نویسی برای ربات را در محیط خودشان دارند و از ابزارهای رابط کاربری مناسبی برخوردارند. نرم افزار Webots، نرم افزار خصوصی است و نیاز به هزینه لایسنس دارد. اما کار با آن بسیار راحت‌تر از Gazebo می‌باشد. بنابراین سختی یادگیری استفاده از Gazebo و ROS یکی از دلایل عدم استفاده از آن در تحقیقات شده است.

- دشواری مدل‌سازی ربات: مدل‌سازی ربات در ROS با استفاده از URDF که به صورت xml ربات را توصیف می‌کند، انجام می‌شود. در V-REP ما قادر به طراحی سه بعدی ربات با استفاده از رابط کاربری و ابزارهای موجود در این نرم‌افزار هستیم. در ROS و Gazebo، ما مجبور به توصیف و طراحی ربات به صورت URDF هستیم. برای تبدیل مدل‌های طراحی شده در نرم‌افزار SolidWorks به فرم URDF، ابزاری طراحی شده است اما، اگر قصد استفاده از سایر نرم‌افزارهای دیگر را برای طراحی ربات داشته باشیم، هیچ ابزار و گزینه‌ای برای این کار وجود ندارد. یادگیری مدل‌سازی ربات در ROS، زمان بسیاری را می‌طلبد و طراحی ربات در قالب URDF نیز وقت بیشتری را در مقایسه با سایر نرم افزارها صرف می‌کند.

همانطور که اشاره شد، استفاده از ROS علاوه بر مزایای بسیاری که دارد، مشکلاتی را نیز به دنبال دارد که شروع به کار با آن را دشوار می‌کند. با این وجود، با توجه به امکانات موجود در آن نظیر ماژولار شدن برنامه‌ها، عملکرد بین پلت‌فرمی و استفاده همزمان از منابع، برای طراحی ربات پیشرفته پارامترهای بسیار جذاب و مهمی هستند.

در ادامه، چند اصطلاح مرتبط با ROS که در ادامه پروژه با آنها برخورد می‌شود، به اختصار توضیح داده می‌شود.

- گره‌ها در ROS: گره در ROS به پروسه‌هایی گفته می‌شود که عملیات محاسباتی و منطقی را

با استفاده از کتابخانه‌های سرویس گیرنده مانند roscpp و rospy انجام می‌دهد. گره‌ها می‌توانند از طریق Topic، خدمات[1] و پارامترها[2] با گره دیگر ارتباط برقرار کند.

هر ربات ممکن است چند گره در خود داشته باشد. به عنوان مثال، ممکن است رباتی یک گره برای پردازش تصاویر دوربین، یک گره برای مدیریت داده‌های دریافتی و ارسالی به ربات، یک گره برای انجام محاسبات ادومتری و گره‌های دیگر برای انجام سایر کارها داشته باشد.

استفاده از گره می‌تواند باعث مقاومت سیستم در مقابل خطا شود. در صورت بروز مشکل در گره‌ای، ربات به باقی عملیات که به طور سالم و صحیح درحال اجرا است، ادامه می‌دهد. همچنین استفاده از گره‌ها، باعث ماژولار شدن برنامه‌ها شده و در نتیجه باعث کاهش پیچیدگی برنامه‌نویسی و اشکال‌زدایی در آن می‌شود.

هر گره‌ای که در ROS تعریف می‌شود، باید دارای نام منحصر به فرد باشد. به عنوان مثال، گره ای به اسم"/camera_node" می‌تواند معرف یک توزیع کننده تصاویر دوربین در محیط ROS باشد.

- پیام در ROS: یکی از راه‌های برقراری ارتباط بین گره‌های تعریف شده، انتشار پیام به یک topic است. پیام‌ها داده‌هایی ساده، سازماندهی شده و در چارچوب قالب‌های از پیش تعیین شده هستند که اطلاعات از جمله اعداد، حروف و رشته‌ها را می‌تواند در خود داشته باشد.

  همانطور که گفته شد، گره‌ها از طریق serviceها نیز می‌توانند با یکدیگر تبادل اطلاعات کنند. Serviceها نیز نوعی پیام هستند که تعریف آنها در فایل‌های srv درج شده است.
- Topic در ROS: Topic یا موضوع در ROS حامل‌های پیام‌ها هستند. Topicها می‌توانند به صورت ناشناس پیام را انتشار و یا دریافت کنند. گره‌ها در ROS به این کاری ندارند که نام گره فرستنده یا گیرنده پیام چیست. آنها تنها به نام Topic که پیام‌ها را در خود گنجانده

[1] Services
[2] Parameters

است و تطابق نوع پیام در فرستنده و گیرنده توجه دارند.

ارتباط بین گره‌ها از طریق Topicها، ارتباط یک طرفه است. در صورتی که به ارتباط دوطرفه و دارای فیدبک نیاز باشد، جهت برقراری ارتباط بین گره‌ها از serviceها به جای پیام‌ها استفاده می‌شود.

گره‌ها در ROS در برقراری ارتباط از طریق topic، از روش انتقال داده TCP/IP به صورت پیشفرض استفاده می‌کند که به نام TCPROS نیز معروف است.

- Serviceها در ROS: در مواقعی که به ارتباطی از نوع بازگشتی و دوطرفه نیاز داشته باشد، باید از Serviceها در ROS استفاده کرد. از این نوع ارتباط بین گره‌ها، معمولا در سیستم پراکنده (توزیع شده)[1] استفاده می‌شود.

Serviceها به صورت جفت پیام تعریف می‌شوند. برای اطلاعات دریافتی و اطلاعات فیدبک، لازم است نوع داده در فایل srv تعیین شود. فایل‌های srv در پوشه srv داخل هر بسته نگه‌داری می‌شود.

در serviceهای ROS، یک گره به عنوان سرور عمل می‌کند که serviceهای کلاینت می‌توانند از سرور درخواست خدمات یا همان service کنند. در صورت به اتمام رسیدن انجام خدمات توسط سرور، نتیجه کار از طریق سرور به کلاینت ارسال می‌شود.

مطالب ارائه شده در خصوص زیر مجموعه‌های ROS هستند و وظایف خاص خود را برعهده دارند. در آخرین بخش از معرفی ROS، به بخش مرکزی آن که نقش مدیریت را برعهده دارد پرداخته می‌شود.

ROS Master ۲-۱-۵-۲

ROS Master همانند سرور DNS است. زمانی که گره‌ای در سیستم ROS شروع به کار می‌کند، این گره به دنبال ROS Master جستجو می‌کند و با یافتن آن، نام گره خود را در آن ثبت می‌کند.

[1] distributed system

بنابراین، ROS Master دارای همه جزئیات در خصوص همه گره‌های موجود در سیستم ROS می-باشد. هرگاه که جزئیاتی در گره‌ای تغییر می‌کند، طبق آخرین اطلاعات، اصلاحات در ROS Master اعمال می‌شود.

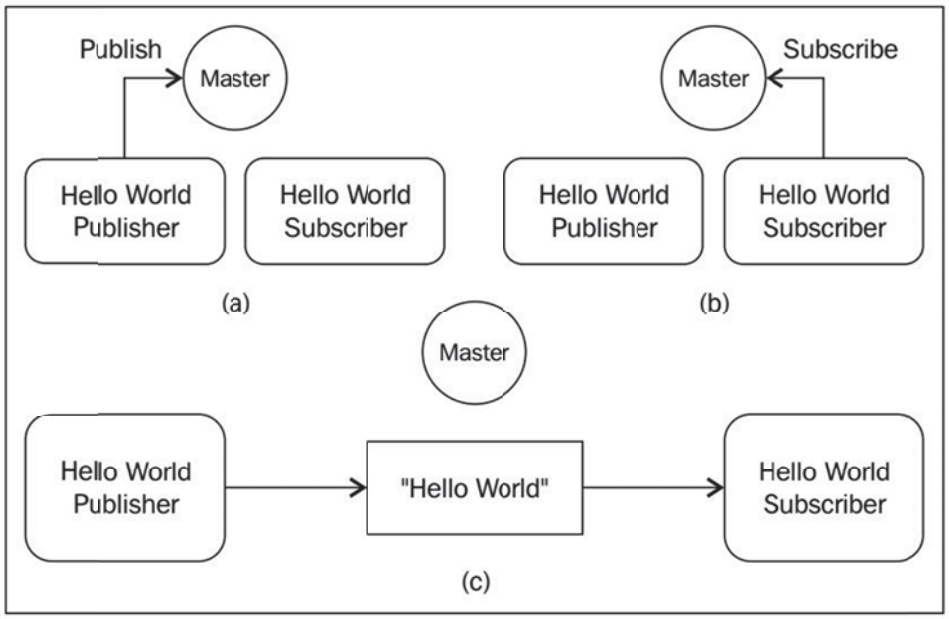
هرگاه که گره‌ای قصد ارسال پیام را داشته باشد، تمامی اطلاعات مرتبط با Topic نظیر نام و نوع اطلاعات ارسالی توسط گره به ROS Master تعریف می‌شود. و سپس ROS Master به دنبال گره‌ای می‌رود که درخواست دریافت اطلاعات ارسالی توسط گره فوق را دارند. در صورتی که تعداد گره‌های دریافت کننده بیش از یک عدد باشد، ROS Master، اطلاعات دریافتی از گره فرستنده را در بین گره‌های گیرنده به اشتراک می‌گذارد. پس از به دست آمدن جزئیات گره‌ها، ندها به واسطه پروتکل TCPROS که قبلاً نیز اشاره شد، به یکدیگر مرتبط می‌شوند. بعد از اتصال گره‌ها، ROS Master دیگر نقشی در کنترل گره‌ها ندارد. ما می‌توانیم گره فرستنده یا گره(های) گیرنده را غیر فعال کنیم که در این صورت، گره باقی مانده با کمک ROS Master، مجدداً دنبال گره فرستنده یا گیرنده غیر فعال شده می‌گردد. این عملکرد در مورد Serviceهای ROS نیز صادق است.

متغیر محیطی[1] "ROS_MASTER_URI" حاوی IP و درگاه ROS Master می‌باشد. با استفاده از این متغیر، گره‌ها می‌توانند ROS Master را شناسایی کنند. در صورتی که این متغیر غلط باشد، ارتباطی بین گره‌ها برقرار نخواهد شد. زمانی که از ROS تنها در یک سیستم استفاده می‌شود، می-توان از IP میزبان محلی[2] استفاده کرد. اما در شبکه‌های گسترده‌تر و توزیع شده، که محاسبات در رایانه دیگری درحال اجرا است، باید این متغیر با دقت تعریف شود تا گره‌ها قادر به پیدا کردن ROS Master باشند. قابل ذکر است که همیشه تنها به یک ROS Master نیاز است.

در انتهای معرفی پلتفرم ROS، در شکل (۲-۱۱)، نحوه برقراری ارتباط در یک سیستم بهره‌مند از ROS نمایش داده شده است. ROS Master با گره‌های فرستنده و گیرنده ارتباط برقرار می‌کند، گره

[1] Environment variable
[2] Localhost

فرستنده، Topic از نوع رشته با پیامی محتوی متن "Hello World" را ارسال می‌کند و گره گیرنده به این Topic اصطلاحاً گوش می‌دهد.



شکل (۲-۱۱) نحوه برقراری ارتباط در یک سیستم بهره‌مند از ROS

هنگامی که گره فرستنده، پیام فوق را در داخل قالب Topic خود ارسال می‌کند، ROS Master جزئیات Topic و گره را دریافت می‌کند و سپس به جستجوی گره‌ای می‌پردازد که منتظر همین Topic است. درصورتی که هیچ گره گیرنده‌ای همزمان با گره فرستنده فعال نباشد، گره‌ای به یکدیگر متصل نخواهد شد. زمانی که دو گره با Topic و جزئیات اطلاعات ارسالی و دریافتی یکسان، همزمان با هم فعال شوند، ROS Master این دو را به یکدیگر مرتبط ساخته و راه تبادل بین آنها را برقرار می‌کند.

۲-۵-۲ طراحی کنترل‌کننده و شبیه‌سازی عملکرد آن

جهت طراحی کنترل‌کننده و شبیه‌سازی آن، نرم افزارهای مختلفی وجود دارد. در خصوص انتخاب نرم افزار خاص و دلیل آن، در فصل سوم بحث خواهد شد.

۲-۵-۳ طراحی برنامه پردازش تصویر و شبیه‌سازی آن

برای طراحی برنامه‌های مرتبط با زمینه پردازش تصویر، دو راهکار به طور کلی وجود دارد که عبارتند از:

- استفاده از کتابخانه های متن باز مانند PCL، OPENCV و غیره و نوشتن کد در محیط‌هایی مانند C++ و C
- استفاده از توابع قدرتمند نرم افزار MATLAB در زمینه پردازش تصویر.

در خصوص انتخاب روش و دلایل آن در این زمینه، در فصل سوم بحث خواهد شد.

۲-۶ جمع‌بندی

در این فصل نمونه‌ای از جدیدترین پژوهش‌هایی که در حوزه کوادکوپتر انجام شده بودند، به اختصار بیان شدند. این حوزه به طور کلی به بخش‌های طراحی، مدلسازی دینامیکی و طراحی کنترل کننده تقسیم‌بندی می‌شوند.

در بخش بینایی ماشین و پردازش تصویر، ابتدا مهم‌ترین روش‌هایی که می‌توان با استفاده از آن اشیا مورد علاقه را در تصویر یک دوربین مشخص کرد و آن را رهگیری کرد، به طور اختصار بیان شد. سپس در مورد موقعیت‌یابی و نقشه برداری همزمان کوادکوپتر بحث گردید.

در بخش مدل‌سازی، با نوشتن معادلات دینامیکی کوادکوپتر، معادلات دقیق دینامیک کوادکوپتر به دست می‌آید که برای نهایی کردن آن به محاسبه دقیق اینرسی نیاز می‌باشد. از آنجا که این

مدل‌سازی بدون در نظر گرفتن اثر محیط واقعی انجام شده است، باید با اضافه نمودن عوامل اغتشاش و عدم قطعیت‌ها به مدل سیستم، عملکرد ربات در فضای واقعی به دست آید.

همچنین، کوادکوپتر مورد نظر در این پروژه، دارای یک بازوی گیرنده کابل هوایی می‌باشد که با باز و بسته شدن آن، ارتفاع مرکز جرم و اینرسی کوادکوپتر تغییر می‌کنند. بنابراین، این مورد نیز باید در مدل‌سازی نهایی مد نظر قرار گیرد.

در بخش کنترل، پنج نوع از کنترلرها با یک دیگر مقایسه شدند که کنترل کننده پسگام افزایشی در بین آنها از عملکرد مناسب‌تری برخوردار بود. با این حال هرکدام از کنترل کننده‌ها مزیت‌هایی نسبت به یکدیگر دارند و با داشتن نتایج کامپیوتری و شبیه‌سازی مهندسی آنها، بدون در نظر گرفتن نویز، نمی‌توان نظر قطعی در مورد کنترل کننده‌ها داد.

کنترل کننده‌های دیگری نیز همانند کنترل کننده مبتنی بر یادگیری ماشین وجود دارد که می‌توانند در کنترل کوادکوپتر مورد استفاده قرار بگیرد.

فصل سوم:

طراحی و شبیه‌سازی

۳-۱ مقدمه

در این فصل در مورد طراحی فیزیکی و الگوریتمی که در ربات پرنده جهت هدایت و کنترل عملکرد آن استفاده می‌شود، بحث می‌شود. ربات کوادکوپتر مورد نظر، باید بتواند موقعیت دقیق خود را نسبت به خطوط شبکه انتقال محاسبه کند و به گونه‌ای خود را کنترل کند که ضمن تامین پایداری و مقاومت در برابر عوامل محیطی، بر روی خط سوار شود و بر روی آن حرکت نماید.

برای مشخص کردن اهداف پایان نامه، در شکل (۳-۱) تصویری از دکل و هادی‌های شبکه انتقال نمایش داده شده است.

شکل (۳-۱) نمونه خط انتقال برق

در شکل (۳-۱) یکی از آرایش‌های شبکه انتقال نمایش داده شده است که در این آرایش، هادی‌ها به صورت دومداره هستند و از باندل ۲ تایی نیز استفاده شده است. در این نوع از شبکه‌ها، اگر رباتی بر روی یکی از خطوط برقدار سوار شود، با توجه به زاویه دید محدود دوربین‌ها، قادر به بازرسی اکثر

هادی‌ها نخواهد بود. بنابراین، بهترین گزینه برای نصب ربات بازرسی کننده از خطوط انتقال، هادی محافظ (گارد) است که در شکل (۳-۱) با دایره قرمز رنگ مشخص شده است. این هادی که جهت حفاظت خطوط از اصابت صاعقه به هادی‌های برقدار، در شبکه به کار برده می‌شود، در بالاترین نقطه از دکل نصب می‌شود تا تمامی خطوط برقدار را تحت حفاظت قرار دهد.

جهت سوار شدن کوادکوپتر بر روی این هادی، گیره نگهدارنده کوادکوپتر طراحی شده است که این گیره بر روی کوادکوپتر نصب می‌گردد. گیره نگهدارنده اشاره شده و کوادکوپتر در محیط solidworks طراحی می‌شود و در ادامه، پس از بحث در مورد الگوریتم پردازش تصویر مورد نیاز و پیاده سازی آن، الگوریتم کلی هدایت و کنترل ربات ارائه می‌شود.

۳-۲ طراحی گیره نگهدارنده کوادکوپتر

جهت طراحی گیره نگهدارنده، باید نکات زیر مد نظر قرار گیرد.

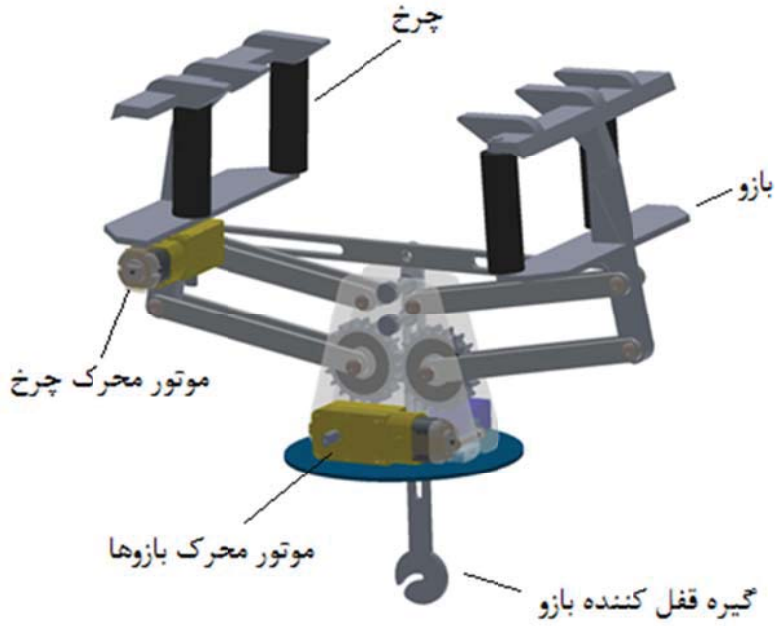
- از آنجا که این گیره بر روی کوادکوپتر نصب می‌شود، باید وزن آن به گونه‌ای باشد که مجموع وزن این گیره و سایر تجهیزات نصب شده بر روی کوادکوپتر، کمتر از حداکثر ظرفیت قابل حمل کوادکوپتر باشد.
- از آنجا که با افزایش ارتفاع نقطه مرکز جرم گیره، مرکز جرم کلی کوادکوپتر نیز مرتفع‌تر می‌شود، باید به گونه‌ای طراحی شود که مرکز جرم مجموعه کوادکوپتر در حالت بسته بودن گیره نگهدارنده، از سطح ملخ‌ها بالاتر نیاید.
- برای جلوگیری از پیچیده شدن الگوریتم کنترل کوادکوپتر، گیره نگهدارنده باید به گونه‌ای طراحی شود که نسبت به محورهای افقی x و y کاملا متقارن باشد. زیرا در غیر این صورت، در درایه‌های غیر قطری ماتریس اینرسی کوادکوپتر، اعداد غیر صفر به وجود می‌آیند که باعث افزایش میزان غیر خطی بودن کوادکوپتر می‌گردد و در نتیجه، کنترل کوادکوپتر دشوارتر

خواهد شد.

- با توجه به اینکه کوادکوپتر در حالت پرواز، هیچ تماس فیزیکی با مرجع دقیق و قابل اعتماد ندارد، موقعیت‌یابی آن با خطایی با مقدار نامشخص همراه است. بنابراین در طراحی گیره نگهدارنده، باید به گونه‌ای عمل کرد که این گیره، بتواند خطاهای انباشته شده در موقعیت‌یابی کوادکوپتر را جبران کند و برای گرفتن کابل، محیط بزرگی را -حین بسته شدن- دربر بگیرد.
- باید به گونه‌ای باشد که نیرو و گشتاور کافی را جهت نگهداری کوادکوپتر بر روی کابل فراهم کند.
- در مقابل وزش باد نسبتاً شدید مقاومت لازم را داشته باشد.
- به حداقل تعداد محرک نیاز داشته باشد.
- در حالت بسته بودن دسته گیره نگهدارنده، سطح موثر جانبی آن تا حد امکان کوچک باشد تا در وضعیت جوی بادی، کمترین تاثیر را بر روی عملکرد و پایداری کوادکوپتر داشته باشد.

با در نظر داشتن موارد فوق، گیره نگهدارنده طراحی می‌شود که حالات باز و بسته آن در شکل‌های (۳-۲) و (۳-۳) نمایش داده شده است. این گیره شامل قسمت‌های اصلی بدنه، چرخ‌ها، موتورهای محرک و بازوها می‌باشد.

مشخصات فیزیکی و هندسی این گیره نگهدارنده در دو حالت کاملاً باز و کاملاً بسته در جدول (۳-۱) نشان داده شده است.



شکل (۳-۲) گیره نگهدارنده طراحی شده جهت اتصال کوادکوپتر به هادی محافظ شبکه انتقال (در حالت باز) – نما از پشت

شکل (۳-۳) گیره نگهدارنده طراحی شده جهت اتصال کوادکوپتر به هادی محافظ شبکه انتقال (در حالت بسته) – نما از روبرو

همانطور که در جدول (۳-۱) مشخص است، به غیر از ابعاد هندسی گیره، ارتفاع مرکز جرم، ممان اینرسی حول محور x و ممان اینرسی حول محور z در بین دو حالت باز و بسته بودن بازوی گیرنده تفاوت چشم‌گیری دارد. از بین این تغییرات موجود، تغییرات ممان اینرسی حول محور x و ارتفاع مرکز جرم، در کنترل و پایداری کوادکوپتر تاثیرگذاری بیشتری را دارد. در بخش ۳-۵ با در نظر گرفتن تغییرات اشاره شده در کوادکوپتر، الگوریتم کنترل مناسب طراحی خواهد شد.

جدول (۳-۱) مشخصات فیزیکی گیره نگهدارنده در دوحالت کاملاً باز و کاملاً بسته

	حالت کاملاً بسته	حالت کاملاً باز	واحد
جرم	۵۳۳	۵۳۳	گرم
طول	۲۳۱.۷۶	۲۴۲.۶۵	میلی متر
عرض	۱۴۵.۴۵	۱۴۵.۴۵	میلی متر
ارتفاع	۲۴۸.۸۱	۱۶۸.۸۱	میلی متر
طول نقطه مرکز جرم	-۰.۴۵	-۴.۶۶	میلی متر
عرض نقطه مرکز جرم	۱.۱۶	-۳.۰۹	میلی متر
ارتفاع نقطه مرکز جرم	۱۳۸.۷۷	۸۶.۶۶	میلی متر
ممان اینرسی حول محور x	۰.۰۰۴۵۲۲۷۸۲	۰.۰۰۰۲۲۲۵۸	$Kg.m^2$
ممان اینرسی حول محور y	۰.۰۰۴۵۲۲۳۶۵	۰.۰۰۴۷۷۰۹۸۴	$Kg.m^2$
ممان اینرسی حول محور z	۰.۰۰۰۷۷۲۰۹۸	۰.۰۰۳۵۱۶۲۴۲	$Kg.m^2$

جنس قطعات به کاربرده شده در این گیره نگهدارنده اغلب از جنس آلیاژهای آلومینیوم است. تنها در قسمت‌هایی از آن مانند چرخ‌ها (فوم)، بلبرینگ‌ها (استیل) و قاب داخلی چرخ‌ها (استیل) از جنس‌های متناسب با آن قسمت استفاده شده است.

دلیل انتخاب جنس فوم برای چرخ‌ها به شرح زیر است:

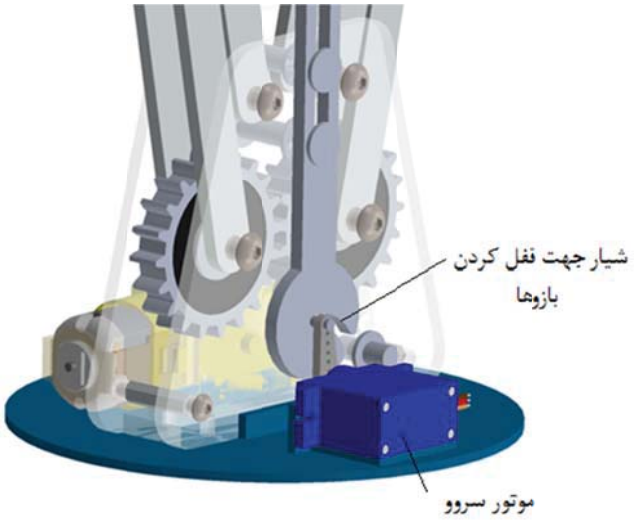
جهت گرفتن هادی محافظ شبکه انتقال که ممکن است اندازه‌های مختلفی را بسته به نوع دکل، آرایش هادی‌های برق‌دار خط و غیره داشته باشد، باید این بازو بتواند این بازه اختلاف اندازه قطر هادی‌ها را در نظر داشته و در مقابل آنها انعطاف نشان دهد. ازآنجا که نصب سیستم فنری برای ایجاد این انعطاف‌پذیری، باعث پیچیدگی سیستم و همچنین سنگین‌شدن آن می‌شود، بهترین و ساده‌ترین راه حل برای این کار استفاده از چرخ‌های نرم از جنس فوم تشخیص داده شده است.

این گیره نگهدارنده دارای ۲ موتور DC و یک موتور سروو[1] است که دارای جعبه دنده جهت افزایش گشتاور می‌باشد. به طور کلی، وظایف موتورهای اشاره شده به شرح زیر است.

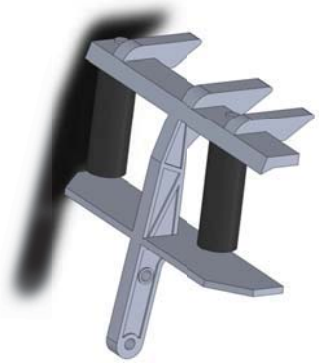
- موتور اول جهت باز و بسته کردن بازوها به کاربرده می‌شود و دارای نیرو و گشتاور بیشتری در مقایسه با سایر موتورها است.
- موتور دوم جهت راه‌اندازی یکی از چرخ‌های انتهای فوقانی گیره نگهدارنده برای حرکت کوادکوپتر روی کابل استفاده می‌شود.
- موتور سوم که از نوع سروو است، تنها برای حفظ حالت بسته بودن بازوهای گیره نگهدارنده به کار برده می‌شود. به این شکل که، بعد از بسته‌شدن کامل بازوهای گیره، روبروی این موتور، یک شیار منحنی شکل ظاهر می‌شود که با فعال شدن موتور و داخل شدن پین در داخل این شیار، بازوها در جای خود قفل می‌شوند. شکل (۳-۴) نمای نزدیک از این قسمت از گیره را نمایش می‌دهد.

دلیل اینکه این شیار به صورت قطاع دایره طراحی شد، این است که اولاً حرکت بازوی متصل به موتور سروو، حرکت دورانی است و ثانیاً با طراحی شیار به این شکل، از آنجا که راستای نیروی وارد بر پین متصل به موتور سروو، از محور این موتور می‌گذرد، گشتاوری بر محور این موتور وارد نشده و بنابراین حتی با قطع کردن کامل مدار موتور سروو، بازوها در جای خود خواهند ماند.

[1] Servo motor



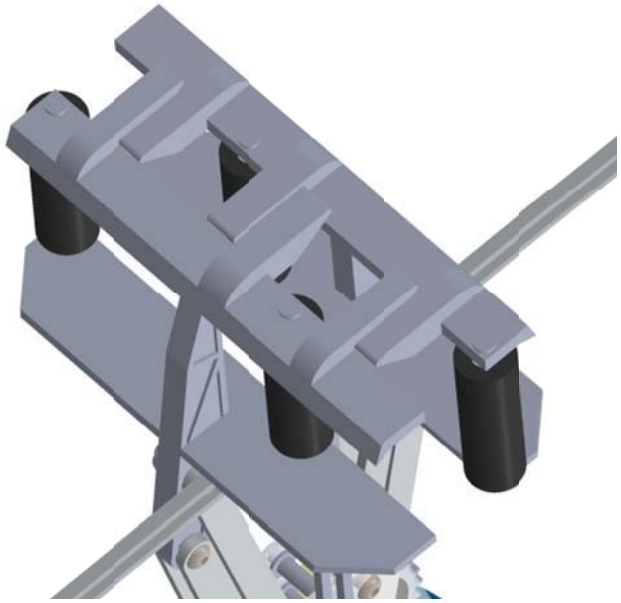
شکل (۳-۴) نمای نزدیک از موتور dc محرک بازوها و موتور سروو نگهدارنده بازوها



شکل (۳-۵) یکی از بازوهای طراحی شده برای گیره نگهدارنده

در شکل (۳-۵) بازوی به کار برده شده در این گیره، نمایش داده شده است. همانطور که در این شکل مشاهده می‌شود، هر بازو دارای دو چرخ است. دسته بازو به گونه‌ای طراحی شده است که نیروی وارد بر چرخ‌ها را تا حد امکان در داخل خود پخش می‌کند. از آنجا که بعد از بسته‌شدن گیره و شروع به

چرخش موتور محرک چرخ، ربات بر اثر نیروی وزن خود به سمت پایین رانده می‌شود، چنگه‌هایی در انتهای فوقانی بازو قرار داده شده‌اند تا از افتادن کوادکوپتر جلوگیری کند.



شکل (۳-۶) نمای بازوهای گیره نگهدارنده از بالا. (درحالت بسته)

در شکل (۳-۶) دو بازو را در حالت بسته بودن گیره نگهدارنده نمایش داده شده است. در این شکل، مشخص است که چرخ‌ها به گونه‌ای تعیین مکان شده‌اند که روبروی یکدیگر نباشند. این کار نیز جهت انعطاف‌پذیری بهتر گیره صورت گرفته است تا بتواند هادی‌هایی با قطرهای متفاوت را بین چرخ‌های خود جای دهند.

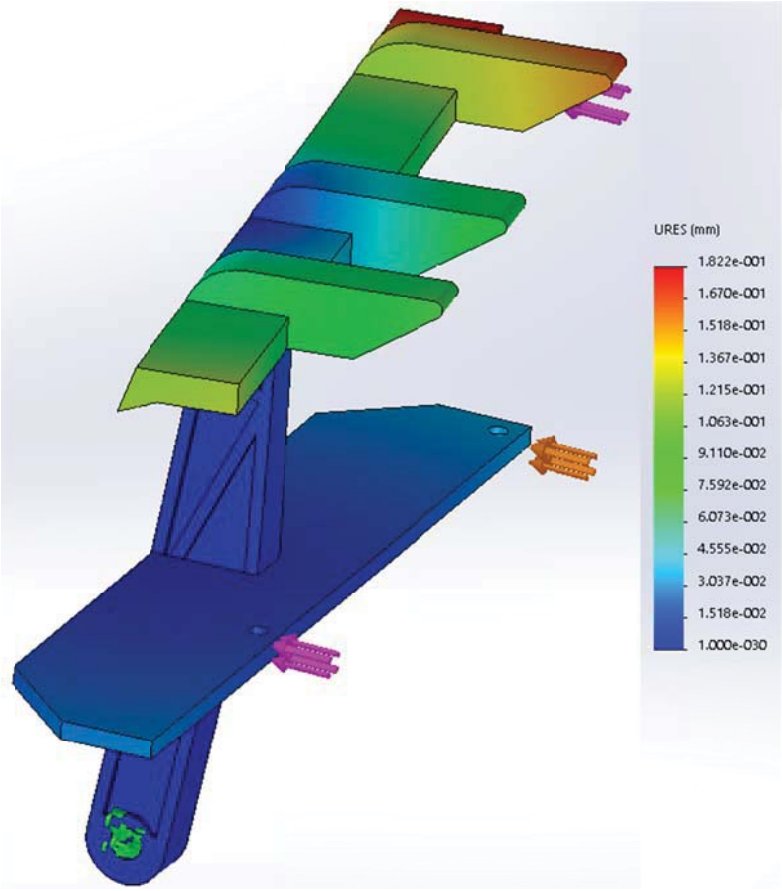
در ادامه، با استفاده از نرم افزار solidworks، گیره نگهدارنده طراحی شده را مورد بررسی استاتیکی قرار می‌دهیم.

برای اینکار، از نرم افزار solidworks استفاده می‌شود. جهت انجام این شبیه سازی، روند زیر طی می-شود.

- تعریف نقاط ثابت بدنه

- تعریف مفاصل و اتصالات موجود در قطعه شبیه سازی
- تعریف نیروی اعمالی و جهت آن
- مش بندی
- شبیه سازی قطعه

در شکل (۳-۷) نتیجه شبیه سازی استاتیکی بازوی گیره نگهدارنده، با اعمال نیروی ۱۰ نیوتنی نمایش داده شده است.



شکل (۳-۷) نتیجه شبیه سازی تحلیل استاتیکی بازوی گیره نگهدارنده

در شکل (۳-۷)، مشخص شده است حداکثر میزان انحراف بازو از حالت اولیه، ۰.۱۸۲۲ میلی متر است. از آنجا که نمی‌خواهیم هیچ انحرافی را در این بازو داشته باشیم، این مقدار به نظر مطلوب می-رسد.

بدین‌ترتیب، گیره نگهدارنده طراحی شد و آماده نصب بر روی کوادکوپتر گردید. در بخش بعد، به طراحی کوادکوپتر پرداخته می‌شود.

۳-۳ طراحی کوادکوپتر

در این بخش، به طراحی کوادرتور در محیط solidworks پرداخته می‌شود. تجهیزات مورد استفاده در کوادرتور این پروژه، به ترتیب زیر هستند:

- ۲ عدد دوربین رنگی (از شرکت Odroid) با دید به سمت بالا
- ۲ عدد دوربین رنگی (از شرکت Odroid) با دید به سمت پایین
- ۴ عدد واحد اندازه‌گیری حرکتی[1] که هرکدام از آن شامل حسگرهای شتاب‌سنج، ژیروسکوپ و قطب‌نما می‌باشد. خروجی این حسگرها با یکدیگر ترکیب می‌شوند که در مورد جزئیات آن در بخش بعد بحث خواهد شد.
- ۱ عدد حسگر GPS
- باطری ۲۲۰۰۰ میلی‌آمپر ساعت.
- ۴ عدد موتور براشلس[2] به همراه ملخ
- کنترل کننده مرکزی
- گیره نگهدارنده

[1] Inertial measurement unit (IMU)
[2] Brushless motor

قابل ذکر است که دوربین‌های رنگی مورد استفاده، با توجه به اینکه بزرگنمایی اپتیکال ندارند و صرفاً فقط حسگر دوربین را در خود دارد، از ابعاد و وزن بسیار پایینی برخوردارند. (طول و عرض ۳۰ میلی-متر. ارتفاع ۱۵ میلی‌متر. وزن بدون احتساب کابل ۵۰ گرم)

با در نظر گرفتن تمامی تجهیزات اشاره شده، کوادکوپتر باید به گونه‌ای طراحی شود که:

- جهت دستیابی به پایداری بهتر، ارتفاع مرکز جرم کوادکوپتر، باید از سطح ملخ‌ها پایین‌تر باشد.
- فریم و دسته‌های کوادکوپتر باید به گونه‌ای باشد که در مقابل وزش باد، کمترین مقاومت را داشته باشد. برای تحقق این امر، می‌توان قطعات به کار برده شده در کوادکوپتر را به صورت مشبک طراحی نمود.
- اندازه بازوها باید به گونه‌ای باشد که دوربین‌های استریو نصب شده جهت دید به سمت پایین، فاصله مناسبی را از یکدیگر داشته باشد. این کار از این جهت که لازم است برای محاسبه عمق تصویر استریو، باید فاصله مناسبی بین دوربین‌ها لحاظ شود، باید مد نظر قرار گیرد.

۳-۳-۱ انتخاب کنترل‌کننده مرکزی کوادکوپتر

جهت انتخاب کنترل‌کننده مرکزی کوادکوپتر، باید نکات زیر مد نظر قرار گیرد:

- این ربات بر روی خط در مدت طولانی قرار می‌گیرد و با حجم بالایی از اطلاعات سروکار دارد. بنابراین کنترل‌کننده مرکزی باید قادر به پاسخگویی به این حجم از اطلاعات باشد.
- این ربات در زمان رسیدن به زیر هادی محافظ، باید بتواند تصاویر گرفته شده توسط ۴ دوربین را به طور همزمان پردازش کند و موقعیت و سرعت خطی و زاویه‌ای لحظه‌ای به دست آمده را به بخش تلفیق حسگرها ارسال کند.
- از آنجا که در این پروژه قصد استفاده از پلتفرم ROS را داریم، پردازنده مرکزی باید از سیستم

عامل لینوکس پشتیبانی کند.

- قابلیت نصب تجهیزات مختلف از جمله حسگرها، محرک‌ها و تجهیزات ارتباطی نظیر wifi و شبکه رادیویی را به صورت ماژولار داشته باشد.
- وزن آن سبک باشد.
- ابعاد آن بیش از حد بزرگ نباشد.

با در نظر داشتن همه موارد فوق، می‌توان از ریز رایانه[1] شرکت Odroid به نام XU4 استفاده کرد.

در شکل (۳-۸)، مدل شبیه‌سازی شده این ریز رایانه در محیط solidworks نمایش داده شده است.

شکل (۳-۸) مدل شبیه‌سازی شده ریز رایانه Odroid XU4

این ریز رایانه از پردازنده شرکت سامسونگ با مدل Exynos5422 بهره می‌برد. دارای ۲ گیگابایت حافظه دسترسی تصادفی[2] از نوع DDR3 است. دارای ۶۴ گیگابایت حافظه داخلی از نوع eMMC5.0 به همراه درگاه نصب حافظه خارجی از نوع MicroSD است. دارای ۲ عدد درگاه usb3 به همراه ۱ عدد درگاه usb2 است و از ارتباط از نوع GPIO/IRQ/SPI/ADC و UART پشتیبانی می‌کند.

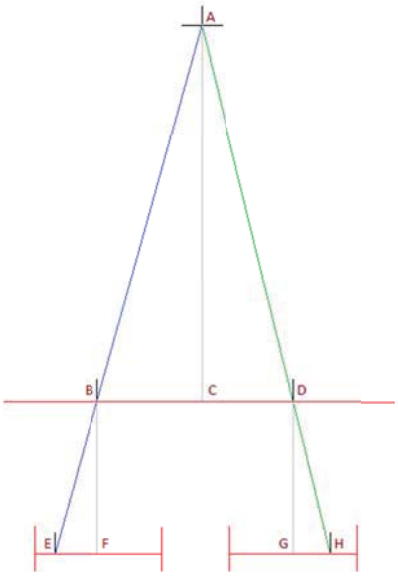
این ریز رایانه قابلیت نصب سیستم عامل لینوکس از جمله اوبونتو نسخه ۱۶.۰۴ را دارد و قابلیت نصب پلتفرم ROS را نیز به همراه خود دارد.

[1] Mini-PC
[2] Random-Access Memory (RAM)

۳-۳-۲ انتخاب دوربین و تعیین موقعیت نصب آن در کوادکوپتر

از دوربین‌های نصب شده در کوادکوپتر که ۲ عدد از آنها به صورت دید به سمت بالا و ۲ عدد دیگر آن به صورت دید به سمت زمین هستند، به ترتیب جهت شناسایی و راهنمایی کوادکوپتر در هنگام بسته شدن گیره نگهدارنده و انجام ادومتری تصویری استفاده می‌شود. همچنین در تعیین وضعیت جوی محیط، از ۱ دوربین فوقانی و ۲ دوربین تحتانی استفاده می‌شود.

با توجه به محدوده دید[1] هر دوربین که توسط فاصله کانونی لنز به کار برده شده در آن تعیین می-شود، لازم است فاصله دوربین‌ها با توجه به کاربردهای اشاره شده تعیین گردند. حال، به محاسبه فاصله کانونی و فاصله بین دو دوربین می‌پردازیم.



شکل (۳-۹) دیاگرام سیستم تصویر برداری استریو با فرض همسطح بودن سطوح تصویر دو دوربین

با توجه به شکل (۳-۹) که یک سیستم تصویر برداری استریو ساده را نمایش می‌دهد، رابطه زیر به دست می‌آید.

$$d = EF + GH$$

[1] Field of view

$$= \mathrm{BF}\left(\frac{EF}{BF} + \frac{GH}{BF}\right)$$
$$= \mathrm{BF}\left(\frac{EF}{BF} + \frac{GH}{DG}\right)$$
$$= \mathrm{BF}\left(\frac{BC + CD}{AC}\right)$$
$$= \mathrm{BF}\frac{BD}{AC}$$

(۱-۳)

دوربین به کار برده شده، دارای سنسوری به اندازه ۳۶۷۳.۶×۲۷۳۸.۴ میکرومتر می‌باشد. جهت داشتن بهترین دقت در فاصله ۲۰ متری و بالاتر از سطح زمین که کوادکوپتر در آنجا کار می‌کند، مقادیر زیر را در رابطه (۳-۱) قرار می‌دهیم:

$$AC=۲۰\,m$$

$$BF =۰.۰۵۵\ m$$

$$d=۲۷۳۸.۴\times\ ۰.۸\ \times۱۰^{-۶}m$$

از آنجا که دو تصویر گرفته شده توسط دوربین‌ها، باید دارای نواحی مشترک باشند تا امکان انجام محاسبات اندازه گیری عمق در تصویر وجود داشته باشد، ۲۰ درصد از تصاویر گرفته شده توسط دوربین‌ها در وضعیت ۲۰ متری از سطح زمین، برای اشتراک‌گذاری بین تصاویر لحاظ شده است. به همین منظور است که در محاسبات فوق الذکر، فاصله پیکسلی d را در عدد ۰.۸ ضرب نمودیم.

با قرار دادن مقادیر فوق در رابطه (۳-۱)، فاصله بین دو دوربین، ۰.۸ متر به دست می‌آید. بنابراین، دوربین‌هایی که به صورت دید به سمت پایین روی کوادکوپتر نصب می‌شوند، باید در فاصله یک متری از هم قرار گیرند. لازم به ذکر است، با توجه به مقدار فاصله کانونی استفاده شده، زاویه دید دوربین‌ها ۱.۴۳ درجه خواهد بود.

همچنین با قرار دادن مقادیر زیر در رابطه (۳-۱) می‌توان فاصله بین دو دوربین دید به سمت بالا را تعیین نمود.

$$AC=۰.۰۸m$$

$$BF = ۰.۰۰۲۶m$$

$$d=۲۷۳۸.۴\times ۰.۸ \times ۱۰^{-۶}m$$

از آنجا که برای گرفتن هادی محافظ شبکه انتقال از گیره نگهدارنده استفاده شده است و این گیره به نحوی طراحی شده که قابلیت گرفتن هادی‌های با فاصله عمودی بین ۹ سانتی متری و ۲۳ سانتی متری از سطح فوقانی کوادکوپتر را دارد، باید دوربین‌ها به گونه‌ای نصب شوند که بتوانند موقعیت هادی را در فاصله بین ۸ و ۵۰ سانتی متری از آن با دقت خوبی به دست آورد.

با قرار دادن مقادیر فوق در رابطه (۳-۱)، فاصله دو دوربین از یکدیگر ۸.۴۲ سانتی متر به دست می-آید.

۳-۳-۳ انتخاب باتری مناسب

از مهم‌ترین نقاط ضعف کوادکوپتر، مداومت پرواز کوتاه آن می‌باشد. در حالت پروازی با حداکثر توان، اینگونه ربات‌های پرنده نمی‌توانند تداوم پرواز طولانی داشته باشند. از آنجا که کوادکوپتر طراحی شده دارای ریز رایانه با مصرف ۲۰ واتی است، این مساله بیشتر خودش را نمایان می‌کند.

اما از آنجا که کوادکوپتر طراحی شده، همیشه به صورت خزشی بر روی هادی شبکه انتقال حرکت خواهد کرد و در این مدت، می‌تواند توقف کرده و با استفاده از صفحه خورشیدی، باتری خودش را شارژ نماید، این مشکل کم رنگ‌تر خواهد بود.

بنابراین، برخلاف سایر ربات‌های پرنده، مساله تعیین کننده ظرفیت باتری در اینجا، بزرگترین مصرف کننده آن یعنی ریزرایانه خواهد بود. البته حداکثر نرخ تخلیه باتری نیز جهت انجام پروازهای هرچند کوتاه مدت باید لحاظ گردد.

از آنجا که محاسبه این دو مقدار، ساده هستند و برای این منظور نرم‌افزارهایی نیز طراحی شده است، تنها به ذکر نتیجه محاسبات بسنده می‌کنیم. باتری مورد نیاز برای مداومت پرواز حدود ۱۲ دقیقه‌ای برای کوادکوپتر با وزن حدود ۳ کیلوگرم به صورت زیر است.

جدول (۳-۲) مشخصات باتری انتخاب شده

ظرفیت (mAh)	مقاومت داخلی (Ohm)	ولتاژ واحد (volt)	تعداد باطری (No.)	نرخ تخلیه مداوم (C)	نرخ تخلیه لحظه ای (C)
۲۲۰۰۰	۰.۰۰۰۵	۳.۷	۲ عدد به صورت سری	۶۵	۱۰۰

۳-۳-۴ طراحی کوادکوپتر

با توجه به مشخصات به دست آمده در بخش ۳-۳، کوادکوپتر مناسب جهت بازرسی از خطوط هوایی فشار قوی، به صورت شکل (۳-۱۰) طراحی شده است. مشخصات فیزیکی این کواد روتور در جدول (۳-۳) آمده است.

شکل (۳-۱۰) کوادکوپتر طراحی شده جهت بازرسی از خطوط فشار قوی

جدول (۳-۳) مشخصات ظاهری و فیزیکی کوادرکوپتر طراحی شده

مشخصه فیزیکی	عدد	واحد
طول	۱۲۰	cm
عرض	۱۲۰	cm
ارتفاع	۲۷.۰۶	cm
وزن	۱.۹ (بدون گیره نگهدارنده)	kg
ممان اینرسی حول محور X	۰.۰۱۹۶۲۵	Kg m^2
ممان اینرسی حول محور Y	۰.۰۱۷۲۷۳	Kg m^2
ممان اینرسی حول محور Z	۰.۰۳۴۷۶۴	Kg m^2
ارتفاع مرکز جرم	-۱۲	cm

در شکل (۳-۱۱) تصویر کوادکوپتر را در حالتی که گیره نگهدارنده بر روی آن نصب شده است، نمایش می‌دهد.

شکل (۳-۱۱) کوادکوپتر با گیره نگهدارنده نصب شده بر روی آن در دو حالت باز و بسته

قبل از پرداختن به بحث طراحی الگوریتم‌های بینایی ماشین و کنترل ربات، لازم به ذکر است که در این پروژه، فرض بر این است که تمامی اطلاعات محیطی گرفته شده توسط کوادکوپتر، اعم از تصاویر دوربین‌ها و خروجی حسگرها، به صورت زنده و با استفاده از هادی محافظ که دارای مغذی فیبر نوری است، به رایانه‌ای که در مکانی دیگر مستقر است ارسال می‌شود و پس از پردازش، نتیجه کار به صورت دستورات کنترلی به صورت زنده و خودکار، به کوادکوپتر ارسال می‌شود.

۳-۴ پردازش تصویر

در این بخش، به طور کلی در مورد دو مبحث زیر بحث می‌شود:

- تشخیص آب و هوا و وضعیت جوی
- تشخیص هادی محافظ خطوط انتقال نیرو

۳-۴-۱ تشخیص آب و هوا و وضعیت جوی

از آنجا که ربات‌های پرنده، در هوا معلق هستند و وضعیت جوی از جمله تابش آفتاب، ریزش باران، وزش باد و تغییرات شدت روشنایی محیط بر روی عملکرد آن‌ها بسیار تاثیرگذار هستند، لازم است جهت ایجاد اطمینان از عملکرد سالم آن، الگوریتمی طراحی گردد که کوادکوپتر را در شرایط جوی نامساعد (ابری و بارانی)، از ادامه بازرسی خط انتقال باز بدارد. در این قسمت، در مورد الگوریتم پردازش تصویر مورد استفاده جهت تشخیص وضعیت جوی محیط اطراف کوادکوپتر بحث می‌شود.
همانطور که در بخش ۲-۱-۴ نیز عنوان شد، تحقیقاتی در این زمینه صورت گرفته است که در یکی از بهترین آنها، با استفاده مشخصه‌های هوا شناختی از جمله سایه، شدت مه آلودگی، وجود یا عدم وجود بازتاب و کنتراست تصویر جهت بازرسی خطوط استفاده می‌شد. مشکل الگوریتم‌های طراحی شده را می‌توان در دقت نه چندان عالی آن‌ها در تشخیص وضعیت جوی دانست. به طوری که نمی‌توان انتظار دقتی بالاتر از ۸۰ الی ۸۵ درصد را از آنها داشت.
در این بخش، با توجه به فرضیات زیر، الگوریتمی جهت تشخیص وضعیت جوی محیط کوادکوپتر با استفاده از دوربین‌ها بیان می‌شود.

- این کار در زمانی از روز که روشنایی کافی در محیط وجود دارد انجام می‌شود.
- از تصاویر سه عدد از دوربین‌های نصب شده روی ربات برای این کار استفاده می‌شود. یکی از دوربین‌ها به سمت آسمان است و دو دوربین دیگر به سمت زمین تنظیم شده است (مراجعه

شود به بخش ۳-۳-۲). این کار جهت بهبود عملکرد و کاهش نرخ خطا انجام می‌شود.

با در نظر گرفتن فرضیات فوق، مساله تشخیص وضعیت جوی را به صورت زیر حل می‌کنیم.

با توجه به این نکته که از سه دوربین جهت انجام این کار استفاده می‌شود، باید الگوریتم نوشته شده، برای هریک از تشخیص‌های صورت گرفته روی سه تصویر، وزن‌گذاری داشته باشد. وزن پیش فرض تشخیص صورت گرفته توسط دوربین فوقانی کوادکوپتر، ۶۰، و وزن دوربین تحتانی آن ۴۰ می‌باشد. این وزن‌ها بر اثر پارامترهای زیر تغییر می‌کنند:

- تشخیص آسمان آبی در تصویر رو به آسمان، باعث افزایش وزن تشخیص وضعیت جوی صورت گرفته توسط این دوربین می‌شود. ضریب افزایش ناشی از این پارامتر، ۱۰ درصد است.
- درصورت وجود خورشید در تصویر گرفته شده توسط دوربین فوقانی، ارزش تشخیص وضعیت جوی توسط این دوربین به ۲۰ کاهش خواهد یافت.
- تعداد نقاط شاخص به دست آمده با استفاده از الگوریتم KAZE [۳۵] بر روی تصویر استریو از زمین، بیان کننده میزان جزئیات موجود بر روی سطح زمین است. تعداد جزئیات، بسته به اینکه سطح زمین مسطح باشد یا دارای پستی بلندی باشد و نقاط سایه و روشن داشته باشد، تغییراتی خواهد داشت. لازم به ذکر است، استخراج ویژگی KAZE بر روی تصویر عمق گرفته شده توسط دو دوربین تحتانی کوادکوپتر اجرا می‌شود. به ازاء هر ۵۰ نقطه شاخص پیدا شده در این تصویر، ۱ درصد وزن تشخیص صورت گرفته توسط دوربین دید به سمت پایین، افزایش پیدا می‌کند.

در ادامه، الگوریتم تشخیص وضعیت جوی با استفاده از دوربین فوقانی و تحتانی به صورت مختصر شرح داده می‌شود.

۳-۴-۱-۱ تشخیص وضعیت جوی با استفاده از دوربینهای تحتانی

برای هرکدام از تصاویر ورودی از دوربین‌های تحتانی، ویژگی‌های آب و هوایی آن را که متشکل از چهار دسته کلی زیر است، استخراج می‌کنیم:

$$[\alpha_{sk}; \alpha_{sh}; \alpha_{re}; \alpha_{co}; \alpha_{ha}]$$

که پارامترها در آن به ترتیب از سمت چپ، مبین سایه[۱]، بازتاب[۲]، کنتراست[۳] و تاری یا مه[۴] می‌باشد. این شاخص‌ها وضعیت و نشانه‌های یک وضعیت آب و هوا را شرح می‌دهد.

از آنجا که همه این نشانه‌ها لزوماً در همه تصاویر وجود ندارد، یک بردار وجود[۵] برای آن تعریف می‌کنیم.

$$[v_{sk}; v_{sh}; v_{re}; v_{ha}]^T$$

که هرکدام از آنها ممکن است صفر یا یک باشند که وجود و یا عدم وجود نشانه‌های فوق‌الذکر را در تصاویر مختلف شرح می‌دهد.

از آنجا که کنتراست در همه تصاویر وجود دارند، v_{co} که وجود یا عدم وجود کنتراست در تصویر را نشان می‌دهد، همیشه برابر با ۱ در نظر گرفته می‌شود و بنابراین در بردار وجود فوق از ذکر آن صرف نظر شده است.

تشخیص سایه

وجود سایه شدید می‌تواند یکی از نشانه‌های موجود در تصاویر گرفته شده در روز آفتابی باشد. برای محاسبه V_{sh} و α_{sh} ، به ابزار تشخیص سایه رجوع می‌کنیم.

[1] shadow
[2] reflection
[3] contrast
[4] haze
[5] existence vector

جهت تشخیص سایه، همانطور که در بخش ۳-۴-۱ گفته شد، از تصاویر استریو ثبت شده توسط دو دوربین تحتانی کوادکوپتر استفاده می‌شود. با استفاده از الگوریتم‌های تفکیک پیش زمینه و پس زمینه در تصاویر استریو که در مرجع [۳۶] آورده شده است، می‌توان بلندی‌های موجود در تصاویر را پیدا نمود و سپس با پردازش بر روی تصویر رنگی یکی از دوربین‌ها جهت یافتن لبه‌های ناشی از وجود سایه، و با در نظر داشتن جدول (۳-۴)، به وجود سایه در تصویر پی می‌بریم.

جدول (۳-۴) نحوه تشخیص سایه از راه تشخیص پیش زمینه و پس زمینه و یافتن نقاط لبه در تصویر RGB

		آیا تغییری در سطح خاکستری رنگ‌ها وجود دارد؟	
		بله	خیر
آیا تغییری در عمق داریم؟	بله	پیش زمینه	پس زمینه
	خیر	سایه	پس زمینه

پس از یافتن سایه ها، با استفاده از رابطه (۳-۲)، مقدار α_{sh} را تعیین می‌کنیم.

$$\begin{cases} \alpha_{sh} = \left|\log_{10}\left(10 \times \left(2 \times \frac{A_{sh}}{A_{im}}\right)\right)\right|; & A_{sh} > \frac{A_{im}}{10} \\ 0; & 0 < A_{sh} < \frac{A_{im}}{10} \end{cases} \qquad (۳-۲)$$

که در آن:

A_{sh} : مساحت نواحی سایه در تصویر

A_{im} : مساحت کل تصویر

این رابطه، مقدار α_{sh} را بین مقادیر صفر تا ۱۰ تنظیم می‌کند.

برای به دست آوردن باقی وزن‌ها (α ها)، مطابق مرجع [۱۴] عمل می‌شود.

۳-۴-۱-۲ تشخیص وضعیت جوی با استفاده از دوربین‌های فوقانی

همانطور که اشاره شد، از دوربین‌های فوقانی، علاوه بر مکان‌یابی کوادکوپتر در هنگام گرفتن هادی محافظ، جهت پیاده‌سازی الگوریتم تشخیص وضعیت جوی نیز استفاده می‌شود. در مرجع [۱۴] یکی از دلایل کاهش دقت تشخیص وضعیت جوی، عدم وجود آسمان در بعضی از تصاویر آموزشی و همچنین آزمون بوده است.

از آنجا که دکل‌های شبکه انتقال دارای حریم قانونی هستند و هیچ سازه‌ای نمی‌تواند در این حریم وجود داشته باشد، دوربین‌های فوقانی نصب شده بر روی کوادکوپتر، همیشه دید به سمت آسمان را دارد و می‌تواند در صورت عدم وجود خورشید در میدان دید دوربین‌ها، تصاویر خالص از آسمان را به دست آورد.

برای شبیه‌سازی این قسمت (تشخیص وجود یا عدم وجود خورشید در تصویر)، از کد از پیش نوشته استفاده شده است. نتیجه آن مطابق با وزن‌دهی‌های گفته شده در ابتدای بخش ۳-۴-۱، به صورت زیر خلاصه می‌شود.

۳-۴-۱-۳ نتیجه شبیه سازی تشخیص وضعیت جوی

نتیجه شبیه‌سازی تشخیص وضعیت جوی با استفاده از این الگوریتم در جدول (۳-۵) آورده شده است. کد این قسمت در محیط Matlab نوشته شده است. برای شبیه‌سازی این قسمت، از مجموعه تصاویر ۲۰۰۰ تایی از تصاویر آموزشی جهت آموزش دسته‌بند استفاده شده است. برای آزمایش این الگوریتم نیز، از ۵۰ عدد تصویر حالت ابری و ۵۰ عدد تصویر حالت آفتابی که در تصاویر آموزشی وجود نداشتند، استفاده شده است.

در جدول (۳-۵) نتیجه عددی این دسته‌بندی آورده شده است. با تقسیم مجموع تعداد تصاویر دسته بندی شده به صورت صحیح بر تعداد کل تصاویر آزمایشی، دقت ۹۱ درصدی این الگوریتم در دسته‌بندی تصاویر ورودی به دست می‌آید.

کد بخش پردازش تصویر جهت تشخیص وضعیت جوی، در پیوست ۱ آورده شده است.

جدول (۳-۵) نتیجه دسته بندی ۵۰ عدد تصویر ابری و ۵۰ عدد تصویر آفتابی

	دسته ابری	دسته آفتابی
۵۰ تصویر ابری	۴۸	۲
۵۰ تصویر آفتابی	۷	۴۳

۳-۴-۲ تشخیص هادی محافظ

برای شناسایی هادی محافظ خطوط انتقال شرایط زیر را داریم.

- هادی محافظ در بالاترین نقطه از دکل شبکه انتقال قرار دارد.
- در مسیر حرکت کوادکوپتر موانعی وجود دارد که از جمله آنها، می‌توان به گوی قرمز –که برای شناسایی خطوط انتقال توسط راننده های هواپیما و بالگردها استفاده می‌شود- و دکل‌ها اشاره کرد.

از آنجا که دوربین‌های فوقانی به صورت استریو درحال ثبت تصاویر از این هادی است، و بالاتر از این هادی، دیگر هیچ چیزی وجود ندارد، لذا با اعمال یک الگوریتم ساده تفکیک تصویر بر روی تصویر عمق، می‌توان موقعیت نسبی هادی را نسبت به کوادکوپتر تعیین نمود. پس از تشخیص هادی در تصویر عمق، با اجرای تبدیل هاف خط بر روی آن، میزان انحراف هادی از مرکز تصویر و زاویه چرخش آن نسبت به کوادکوپتر را به دست می‌آوریم.

بعد از به دست آمدن میزان انحراف ربات نسبت به هادی محافظ، این انحراف به کنترل کننده کوادکوپتر ارسال می‌گردد و نیروهای متناسب با آن، به موتورها اعمال می‌گردد تا این اختلاف از بین برود.

۳-۵ طراحی الگوریتم کنترل

در این بخش به طراحی کنترل کننده کوادکوپتر پرداخته می‌شود. برای این کار نیاز به طی نمودن چند مرحله وجود دارد که به ترتیب به آنها پرداخته خواهد شد.

۳-۵-۱ آماده سازی اطلاعات ورودی از محیط

برای کنترل و شبیه‌سازی نحوه عملکرد کوادکوپتر، به اطلاعات زیر نیاز خواهد بود.

- اطلاعات شتاب خطی و موقعیت و سرعت زاویه‌ای (به صورت زوایای اویلر) خوانده شده از روی IMU
- اطلاعات دریافتی از حسگر GPS
- اطلاعات موقعیتی دریافت شده از ادومتری تصویری

علت اینکه از ۱۲ پارامتر اشاره شده (۳ عدد برای موقعیت خطی، ۳ عدد برای موقعیت زاویه‌ای، ۳ عدد برای سرعت خطی و ۳ عدد برای سرعت زاویه‌ای) استفاده می‌شود، این است که در بخش طراحی کنترل (۳-۵-۲-۲)، جهت کنترل کوادکوپتر، از کنترل کننده‌های خطی استفاده می‌شود. اگر در این حالت، از رویت‌گر خطی نیز استفاده شود، پایداری کوادکوپتر در حالت دور از نقطه پایداری را نمی-توان تضمین نمود. زیرا رفتار رویت‌گر خطی با رفتار سیستم واقعی در این نقاط، تفاوت زیادی پیدا می‌کند

حال به مشکلات استخراج ۱۲ متغیر اشاره شده پرداخته می‌شود.

می‌دانیم که شتاب خطی به دست آمده از IMU با نویز همراه است. همچنین می‌دانیم، درصورتی که قصد به دست آوردن سرعت و حتی موقعیت از روی اطلاعات شتاب سنج نویزدار را داشته باشیم، با مشکل خطای انباشته مواجه خواهیم شد.

از طرفی، اگر از GPS برای موقعیت‌یابی کوادکوپتر استفاده نماییم، نمی‌توان انتظار خطای کمتر از چند متر را از موقعیت ربات داشت. زیرا GPS دارای خطایی در حدود ۲ الی ۵ متر است.

از طرفی دیگر، اگر تنها از اومتری تصویری که با استفاده از دوربین‌های تحتانی کوادکوپتر صورت می‌گیرد جهت موقعیت‌یابی ربات استفاده شود، با وزش باد و انحراف از حالت ایستا و تغییر در زوایای اویلر ربات، این الگوریتم، تصویر را گم کرده و با مشکل برخورد خواهد کرد.

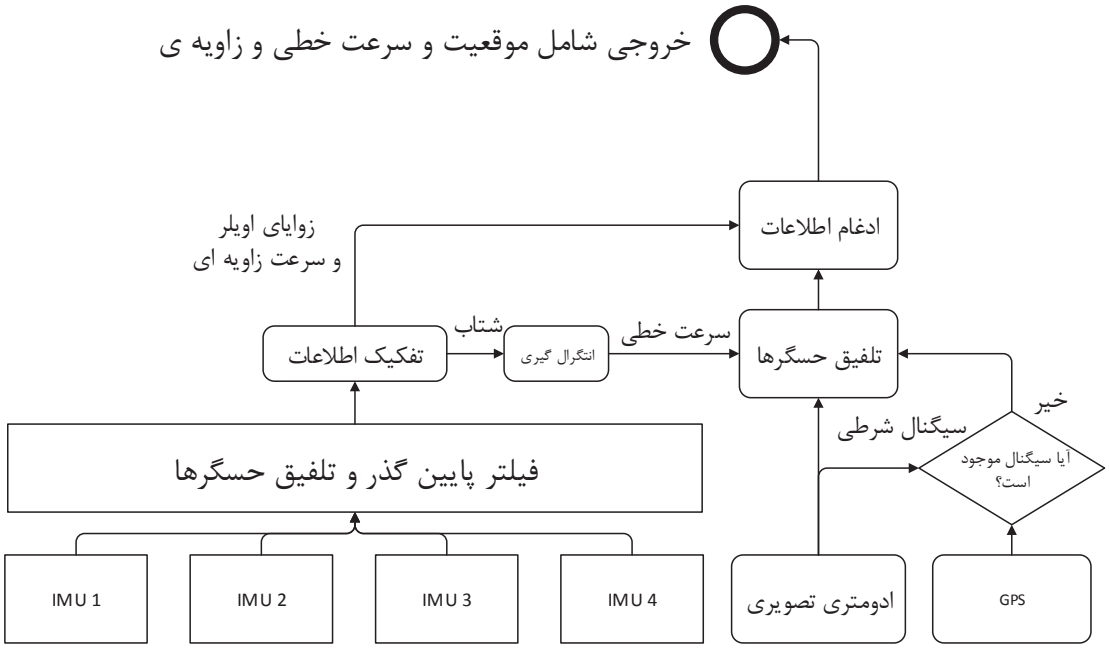
ارتفاع محدوده کار ربات از سطح زمین، حدود ۲۰ الی ۴۰ متر است و اگر در ارتفاعی مانند ۳۰ متر، یکی از زوایای رول[1] یا پیچ[2] انحراف ناگهانی به عنوان مثال ۱۵ درجه‌ای را بر اثر وزش باد پیدا کند، با محاسبات ساده مثلثاتی، متوجه جابجایی زیادی در تصویر این دو دوربین می‌شویم. اگر این جابجایی به صورت ناگهانی اتفاق افتد و نرخ پردازش تصویر جهت انجام ادومتری که حدود ۱۵ فریم در ثانیه است، نتواند با این سرعت جابجایی تصویر، خود را وفق دهد، این الگوریتم موقعیت خود را گم می‌کند.

بنابراین، هیچ یک از روش‌ها و حسگرهای گفته شده، به تنهایی قادر به موقعیت‌یابی دقیق کوادکوپتر نمی‌باشد. به همین دلیل از تلفیق حسگرها جهت ادغام اطلاعات موقعیتی به دست آمده استفاده می‌کنیم. برای این کار از فیلتر کالمن استفاده شده است.

از آنجا که الگوریتم استفاده شده در اینجا، مشابه روند فیلتر کالمن شکل (۲-۴) است، از شرح مجدد آن پرهیز می‌شود.

[1] Roll Angle
[2] Pitch Angle

در شکل (۳-۱۲) روند انجام تلفیق حسگرها نمایش داده شده است. باید توجه داشت که همه مراحل اشاره شده در این شکل، لازم است به صورت موازی و همزمان با یکدیگر انجام شود. زیرا در غیر این صورت، ممکن است اطلاعات لحظه صفری که از یکی از حسگرها دریافت می‌شود، با اطلاعات لحظه n ام از سایر حسگرها ادغام شود و نتیجه اشتباه در خروجی به دست آید.



شکل (۳-۱۲) روند آماده سازی اطلاعات ورودی از محیط جهت استفاده از آن در کنترل کوادکوپتر

برای انجام این کار از پلتفرم ROS استفاده می‌کنیم. همانطور که در بخش ۲-۴-۱ گفته شد، با ایجاد ندهای مختلف، و خواندن اطلاعات به اشتراک گذاشته شده در Topicها، تمامی پردازش‌هایی که لازم است به صورت موازی اجرا شوند، در گره‌های مختلف اجرا خواهند شد. تمامی عملیات فوق به زبان C++ نوشته شده است و در هنگام اجرای برنامه کنترل ربات، موقعیت و سرعت خطی و زاویه را از محیط می‌گیرد و بعد از پردازش، در گره‌ای به نام kalman_pose آنها را منتشر می‌کند تا الگوریتم

کنترلی که در متلب نوشته شده است، بتواند از آن استفاده کند. کد استفاده شده در این بخش، در پیوست ۲ آمده است.

در ادامه، با حرکت دادن کوادکوپتر در راستای عمودی و ثبت اطلاعات دریافتی از ندها، اثر استفاده از فیلتر کالمن را در افزایش دقت اندازه گیری سرعت و جابجایی خطی را نمایش می‌دهیم.

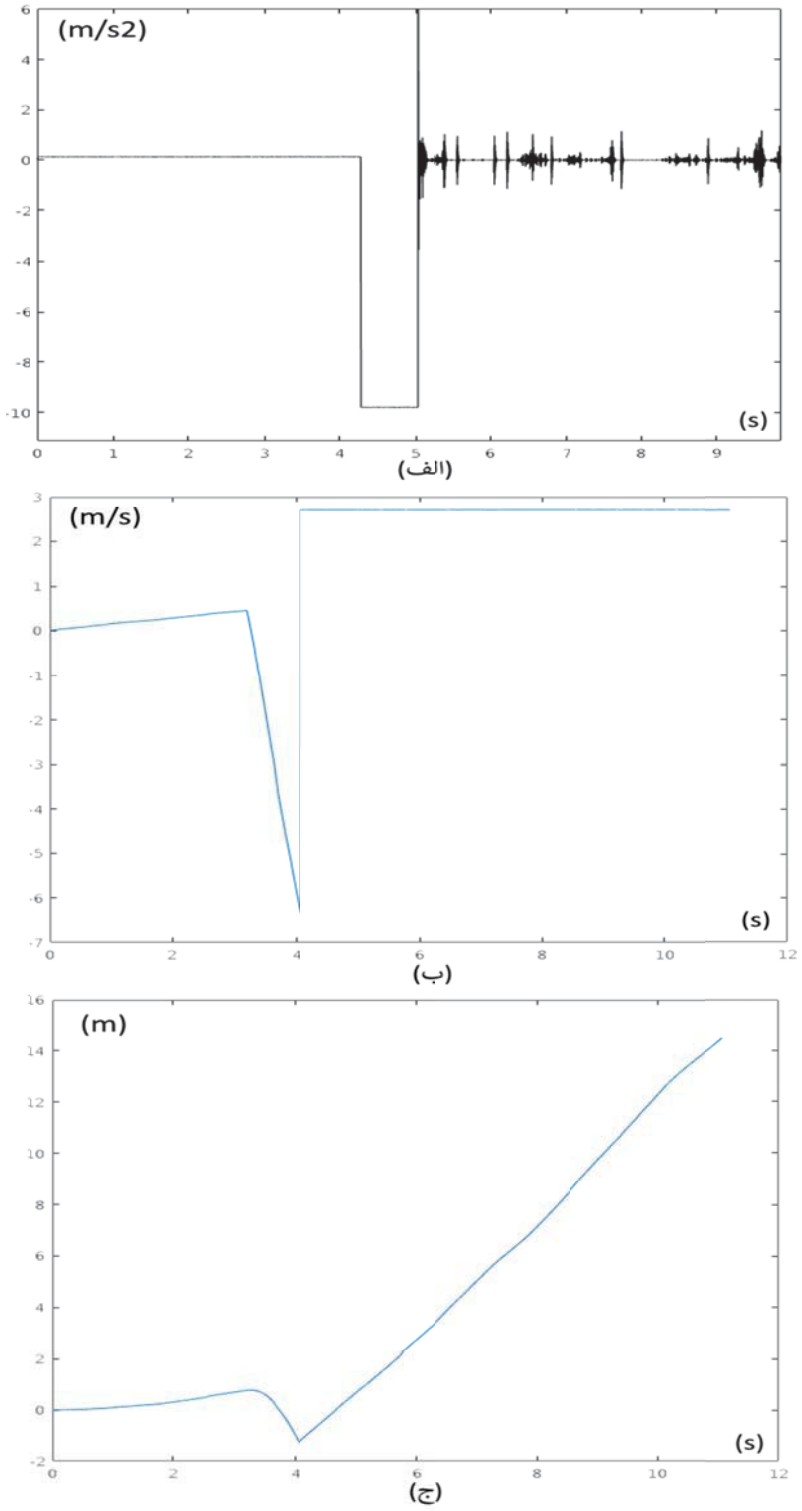
در حرکت عمودی فرضی، ابتدا کوادکوپتر با نیروی ثابت در جهت عمودی حرکت می‌کند و با رسیدن به نقطه اوج، ملخ‌ها را خاموش کرده و با شتاب جاذبه، به سمت زمین حرکت می‌کند و با زمین برخورد می‌کند.

در شکل (۳-۱۳) و در تصویر (الف) شتاب عمودی اندازه گیری شده توسط یکی از IMU ها نمایش داده شده است. شکل (۳-۱۳) (ب)، سرعت عمودی کوادکوپتر را نمایش می‌دهد که از انتگرال گیری از شکل (الف) به دست آمده است. شکل (۳-۱۳) (ج) نیز جابجایی کوادکوپتر در راستای عمودی را نمایش می‌دهد که با انتگرال گیری از شکل (ب) به دست آمده است.

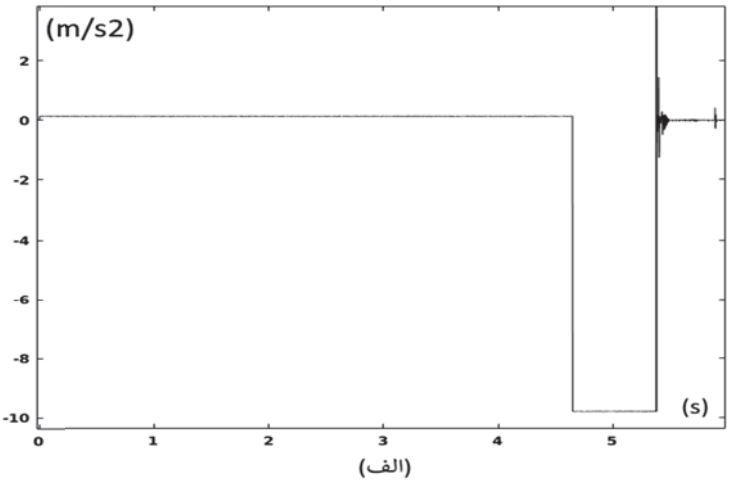
در شکل (۳-۱۳) (الف)، مشخص است که کوادکوپتر تا حدود ۴.۳ ثانیه با شتاب ثابت در حرکت است و در ادامه، با خاموش شدن موتورها شروع به سقوط آزاد می‌کند و در لحظه ۵ ثانیه بعد از شروع حرکت، با زمین برخورد می‌کند. با دقت در شکل (۳-۱۳) (ب)، مشخص می‌شود که بعد از برخورد با زمین و توقف حرکت، شتابسنج دچار خطا شده و سرعت را حدود ۲.۸ متر بر ثانیه نشان می‌دهد. این مقدار خطا در اندازه گیری سرعت، باعث به وجود آمدن خطای شدید در اندازه گیری جابجایی می-شود که در شکل (۳-۱۳) (ج) نیز قابل مشاهده است. در این شکل، بعد از برخورد کوادکوپتر با زمین، اطلاعات به دست آمده از شتابسنج، به خاطر خطای انباشته شده در روند انتگرال گیری، دارای خطای زیادی است و در نتیجه، این مقادیر غیر قابل استفاده هستند.

در شکل (۳-۱۴)، خروجی فیلتر کالمن که اطلاعات ۴ عدد IMU را با یکدیگر تلفیق کرده است، نمایش می‌دهد. با مقایسه شکل‌های (۳-۱۳) (ب) و (۳-۱۳) (ج) با به ترتیب (۳-۱۴) (ب) و (۳-۱۴) (ج)، مشخص است که مقدار خطا بعد از استفاده از تلفیق حسگرها، کاهش پیدا کرده است.

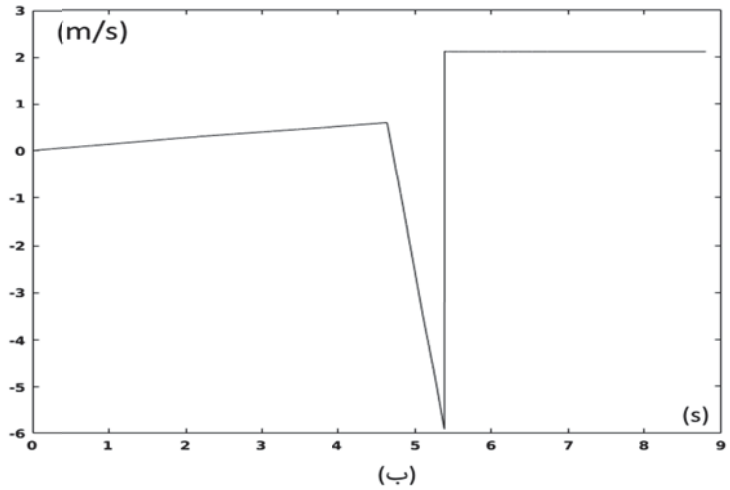
با این حال، همانطور که در شکل (۳-۱۴) (ج) نیز مشاهده می‌شود، هنوز خطای انباشته شده، در خروجی اندازه گیری جابجایی کوادکوپتر خطای بزرگی را ایجاد کرده است.



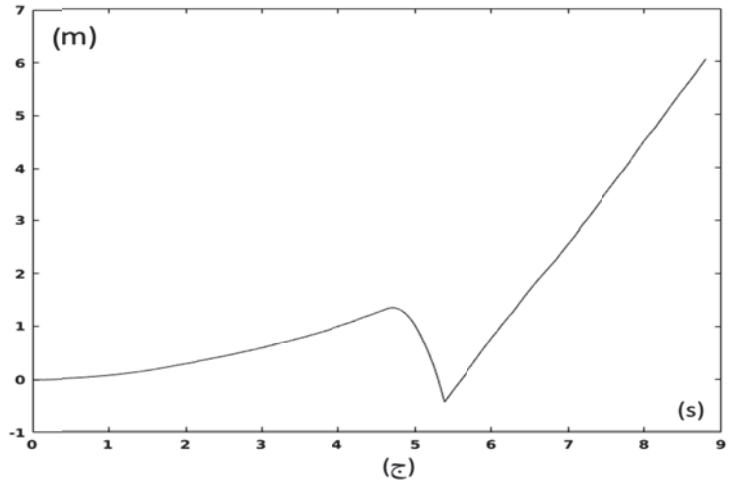
شکل (۳-۱۳): (الف) شتاب و (ب) سرعت و (ج) جابجایی عمودی اندازه گیری شده بدون استفاده از تلفیق حسگرها



(الف)



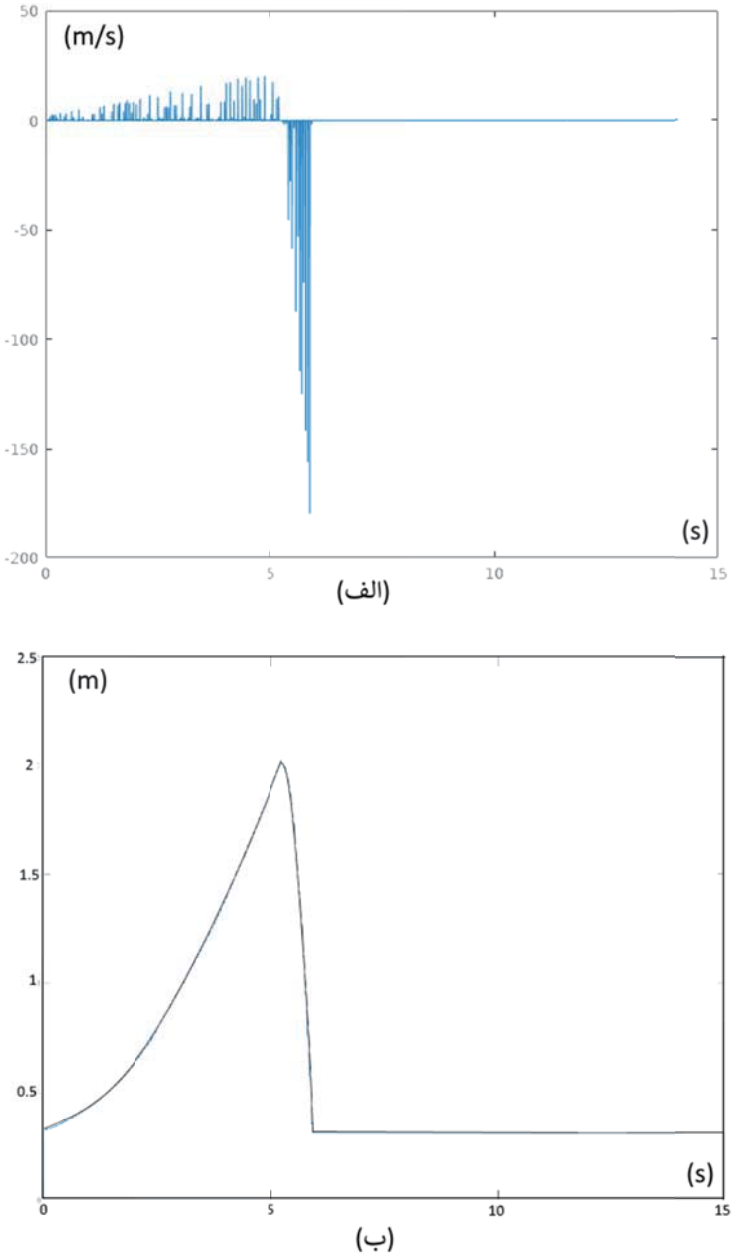
(ب)



(ج)

شکل (۳-۱۴): (الف) شتاب و (ب) سرعت و (ج) جابجایی عمودی اندازه گیری شده توسط ۴ عدد IMU بعد از تلفیق حسگرها

همانطور که در شکل (۳-۱۲) نمایش داده شده است، خروجی تلفیق ۴ عدد حسگر IMU، با مقادیر جابجایی و سرعت خطی به دست آمده از ادومتری تصویری یا GPS تلفیق می‌شود. علت این کار، از بین بردن خطای انباشته شتابسنج است. در شکل (۳-۱۵) نتیجه این عمل آورده شده است.



شکل (۳-۱۵): (الف) سرعت و (ب) جابجایی عمودی اندازه گیری شده در مرحله نهایی تلفیق حسگرها

با توجه به شکل (۳-۱۵) (ج)، خطای اندازه گیری به شکل مطلوبی از بین رفته است و برای فیدبک سیستم کنترلی، قابل استفاده می‌باشد.

با به دست آمدن مقادیر مورد نیاز جهت تکمیل فیدبک کنترل کننده کوادکوپتر، حال می‌توان به طراحی کنترل‌کننده کوادکوپتر پرداخت.

۳-۵-۲ طراحی کنترل کننده کوادکوپتر

در این بخش، ابتدا مدل غیر خطی کوادکوپتر به دست می‌آید و با در نظر داشتن فرضیاتی، کنترل کننده کوادکوپتر طراحی خواهد شد.

۳-۵-۲-۱ مدل سازی کوادکوپتر

قدم اول در طراحی کنترل‌کننده هر سیستمی، شناسایی آن سیستم و به دست آودن معادلات توصیف کننده رفتار آن سیستم می‌باشد. برای مدل‌سازی کوادکوپتر، ابتدا باید نگرش زاویه‌ای ربات تعریف شود.

سپس با نوشتن روابط سینماتیک و دینامیک آن، مدل‌سازی کوادکوپتر به اتمام می‌رسد.

قبل از شروع مدل‌سازی، فرضیات زیر را در نظر می‌گیریم.

- از آنجا که کوادکوپتر در ارتفاع ۲۰ الی ۴۰ متری از سطح زمین پرواز خواهد کرد، از اثر زمین در مدل‌سازی این ربات، صرف نظر می‌شود و سطح زمین هموار در نظر گرفته خواهد شد.
- با توجه به عدم نیاز سرعت خطی و زاویه‌ای بالا در پرواز کوادکوپتر، نیروی ایرودینامیکی وارد بر آن در شرایط عادی پروازی (بدون وزش باد) صفر در نظر گرفته می‌شود.
- ملخ‌ها و بدنه کوادکوپتر، با فرض حرکات کند آن، صلب در نظر گرفته می‌شود.
- از کهنه شدن و تغییر پارامترهای الکتریکی-مکانیکی موتورها صرف نظر می‌شود.

۳-۵-۲-۱-۱ تعیین نگرش

برای ساخت مدل کوادکوپتر دقیق، نگرش زاویه‌ای ربات باید تعریف شود. در اینجا از نمایش زوایای اویلر استفاده می‌شود. در این روش زاویه pitch محدود به بازه ۹۰- الی ۹۰ درجه است. این روش متداول‌ترین روش ارائه مدل سینماتیکی اجسام صلب در فضا است[۱]. همچنین زوایای اویلر دارای این مزیت است که به صورت بصری مشهود و واضح می‌باشد. با این وجود، واضح است که این روش در مقایسه با روش‌هایی همچون روش چرخش چهارگانه[1] -که المان اضافی را برای توصیف کامل استفاده می‌کند (و کمتر غیر خطی است)- بار محاسباتی سنگین‌تری دارد[۱].

حال دستگاه مختصات متعامد مرجع A و دستگاه نصب شده بر روی ربات B در نظر گرفته می‌شود. می‌توان با استفاده از تبدیلات دوران برای هرکدام از محورهای مختصات، دوران دستگاه B را نسبت به A تعیین کرد. هرکدام از این تبدیلات به صورت ماتریس دوران تعریف می‌شوند که در روابط آمده‌اند.

برای چرخش حول محور x به اندازه φ_1 ، ماتریس دوران حول آن به صورت زیر تعریف می‌شود:

$$\mathrm{R}_x(\varphi_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_1 & \sin\varphi_1 \\ 0 & -\sin\varphi_1 & \cos\varphi_1 \end{bmatrix} \qquad (۳-۳)$$

برای دوران حول محور y به اندازه φ_2، ماتریس دوران به صورت زیر خواهد بود:

$$\mathrm{R}_y(\varphi_2) = \begin{bmatrix} \cos\varphi_2 & 0 & -\sin\varphi_2 \\ 0 & 1 & 0 \\ \sin\varphi_2 & 0 & \cos\varphi_2 \end{bmatrix} \qquad (۴-۳)$$

و در نهایت برای دوران حول محور z ها به اندازه φ_3، ماتریس دوران به صورت زیر خواهد بود:

$$R_z(\varphi_3) = \begin{bmatrix} \cos\varphi_3 & \sin\varphi_3 & 0 \\ -\sin\varphi_3 & \cos\varphi_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (۵-۳)$$

دقت شود که این ماتریس‌ها متعامد هستن که به معنی برقرای رابطه زیر برای آنها است:

$$\mathrm{R}(\varphi)^{-1} = R(\varphi)^T \qquad (۶-۳)$$

[1] Quaternion of rotation

تبدیل کامل بین دستگاه‌های مختصات از ضرب سه ماتریس فوق به دست می‌آید که خود نیز متعامد خواهد بود. روش‌ها و ترتیب‌های مختلفی برای دوران وجود دارد که برای اشیاء پرنده (هواپیما و ربات)، یکی از انتخاب‌های اصلی، ترتیب ۱-۲-۳ برای دوران است[۱]. در این دوران، زوایا به صورت زیر بیان می‌شوند:

$$[\varphi_1 \quad \varphi_2 \quad \varphi_3] = [\varphi \quad \theta \quad \psi] \qquad (۷-۳)$$

که در آن، φ[1] زاویه چرخش حول محور x، θ[2] زاویه چرخش حول محور y و ψ[3] زاویه چرخش حول محور z می‌باشد.

ترتیب کار به صورت زیر می‌باشد:

- اول چرخش حول محور z به اندازه ψ
- دوم چرخش حول محور y به اندازه θ
- و سوم چرخش حول محور x به اندازه φ

بنابر تعریف ارائه شده، ماتریس تبدیل برای دوران دستگاه مختصات A به B به صورت زیر خلاصه می‌شود[۱]:

$$T^{NB} = R_z(\psi)R_y(\theta)R_x(\varphi) = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\varphi - \sin\psi\cos\varphi & \sin\psi\sin\theta\sin\varphi + \cos\psi\cos\varphi & \cos\theta\sin\varphi \\ \cos\psi\sin\theta\cos\varphi + \sin\psi\sin\varphi & \sin\psi\sin\theta\cos\varphi - \cos\psi\sin\varphi & \cos\theta\cos\varphi \end{bmatrix} \qquad (۸-۳)$$

از آنجا که ماتریس فوق متعامد می‌باشد، معکوس این ماتریس را که بیان کننده ماتریس تبدیل از دستگاه مختصات ربات به دستگاه مرجع است، به صورت زیر بیان نمود:

$$\mathrm{T}^{BN} = T^{NB^{-1}} = T^{NB^T} \qquad (۹-۳)$$

۳-۵-۲-۱-۲ روابط سینماتیک ربات

[1] Roll
[2] Pitch
[3] Yaw

روابط سینماتیکی، حرکت اجسام را بدون در نظر گرفتن نیروی عامل حرکت مورد تحلیل قرار می-دهد. مشتق زمانی زوایای اویلر که سرعت زاویه‌ای آنها را بیان می‌کند، در این بخش مورد استفاده قرار می‌گیرد. روند دوران سه‌تایی به صورت زیر اعمال می‌شود:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\varphi & \sin\varphi\cos\theta \\ 0 & -\sin\varphi & \cos\varphi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \qquad (۱۰-۳)$$

که صورت معکوس این عبارت به شکل زیر خواهد بود:

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi/\cos\theta & \cos\varphi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad (۱۱-۳)$$

۳-۵-۲-۱-۳ روابط دینامیک ربات

دینامیک حرکت کوادکوپتر را می‌توان با استفاده از قانون دوم نیوتن به دست آورد که خلاصه آن به صورت زیر بیان شده است:

$$\sum F = \frac{d}{dt}(mV) \qquad (۱۲-۳)$$

$$\sum M = I\alpha \qquad (۱۳-۳)$$

که در آن α شتاب زاویه‌ای و V سرعت خطی جسم صلب است. برای به دست آوردن روابط دقیق جهت مدلسازی کوادکوپتر، وزن و خصوصیات اینرسی ربات مورد نیاز است که در بخش ۳-۱ به دست آمدند.

نیروها و گشتاورهای خارجی

در مدلی که ارائه می‌شود، دو اثر ایرودینامیکی در نظر گرفته شده است. یکی از آنها ناشی از روتور و چرخش ملخ‌ها است و دیگری در ارتباط با اصطکاکی است که در تمامی بدنه کوادکوپتر در حالتی که سرعت آن غیر صفر باشد وجود دارد[۱].

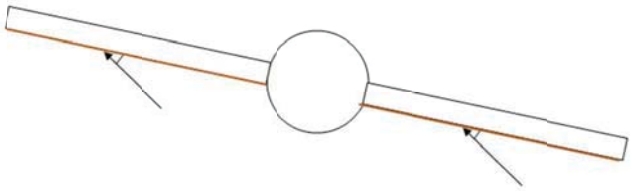
در ادامه، اطلاعات تکمیلی مورد نیاز جهت مدل سازی دقیق کوادکوپتر ارائه می‌شود.

۳-۵-۲-۱-۴ اثر دایهدرال[1]

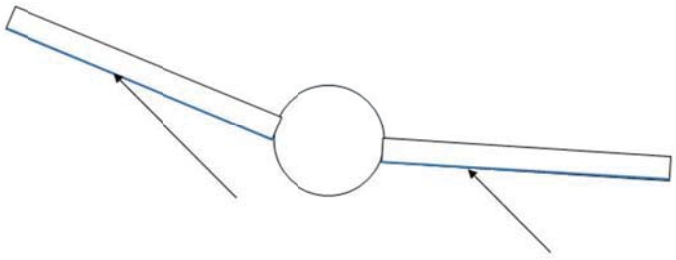
اگر روتورهای متصل به فریم[2]، نسبت به خط افق با زاویه ساخته شوند به این طرح دایهدرال می-گویند. در ادامه نشان داده خواهد شد که این پیکربندی تاثیرات خوبی بر روی پایداری خواهد داشت و در نتیجه کنترل شناوری آسان‌تر خواهد بود.

انحراف مثبت بال نسبت به خط افق را زاویه دایهدرال گویند. حرکت پرنده در زمانی که به آن ممان وارد می‌شود در دو حالت داشتن و نداشتن زاویه دایهدرال مقایسه شده است:

در شکل (۳-۱۶) جسم پرنده‌ای نشان داده شده است که زاویه دایهدرال ندارد و در حال چرخش است. اما در شکل (۳-۱۷) زاویه دایهدرال به جسم پرنده اضافه شده است. همانطور که مشاهده می-شود هر دوبال یک زاویه حمله را تجربه می‌کنند.



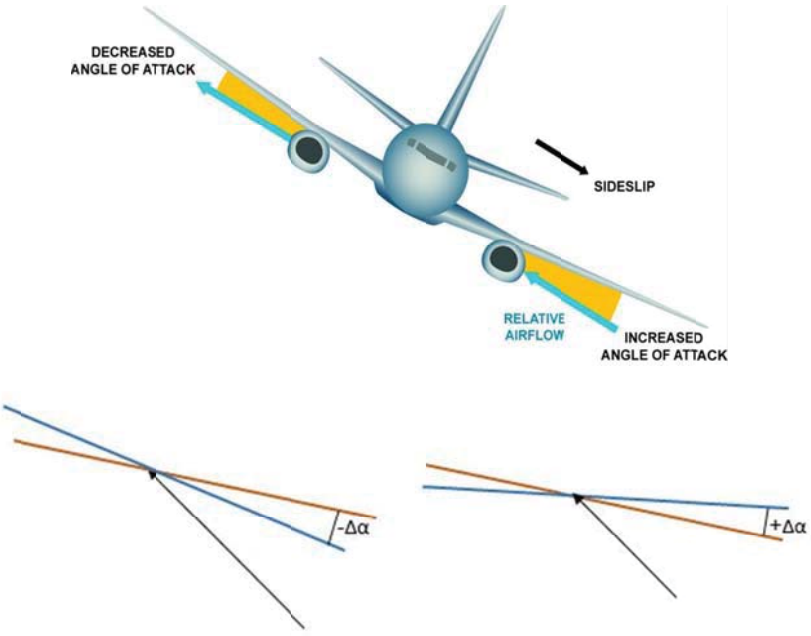
شکل (۳-۱۶) جسم پرنده بدون زاویه دایهدرال



شکل (۳-۱۷) جسم پرنده به‌همراه زاویه دایهدرال

[1] Dihedral
[2] Frame

زمانی که جسم پرنده دارای زاویه دایهدرال باشد ، بالی که در جهت وزش باد برخوردی قرار دارد، افزایش زاویه حمله به اندازه $\alpha\Delta$ و به تبع آن افزایش میزان لیفت به اندازه ΔL را تجربه می‌کند. در حالی‌که بال مقابل دقیقا خلاف این موضوع را تجربه می‌کند. یعنی زاویه حمله به اندازه $-\alpha\Delta$ تغییر می‌کند. این دو عملکرد متفاوت در دو بال جسم پرنده ،ممانی در جهت بازگشت هواپیما به وضعیت اولیه ایجاد می‌کند. از این موضوع به تاثیر دایهدرال یاد می‌شود. در شکل (۳-۱۸) اثر زاویه دایهدرال بر روی تغییرات زاویه حمله نشان داده شده است[۳۷].



شکل (۳-۱۸) اثر زاویه دایهدرال بر روی زاویه حمله

باید توجه داشت اثر دایهدرال اگرچه باعث پایداری بیشتر می‌شود ،اما باعث اتلاف انرژی و مصرف انرژی بیشتر می‌شود چراکه مقداری از نیروهای تراست تولیدی، در خلاف یکدیگر شکل می‌گیرند. حال با در نظر گرفتن این پدیده، مدل کوادکوپتر را تغییر می‌دهیم.

برای مدل‌سازی با درنظرگرفتن اثر دایهدرال باید نیرو و ممان‌های ناشی از روتورها را با توجه به زاویه دایهدرال تجزیه کرد.

۳-۵-۲-۱-۵ در نظر گرفتن مرکز جرم غیر همسطح با سطح ملخ‌ها

در حالت عادی، کوادکوپتر به گونه‌ای طراحی می‌شود که مرکز جرم مجموعه کوادکوپتر، بر روی سطح ملخ‌ها قرار داشته باشد. در غیر این صورت، با تغییر زوایای رول و یا پیچ، گشتاوری به کوادکوپتر اعمال می‌شود.

در ادامه با در نظر داشتن همه موارد بحث شده در بخش ۳-۵-۲، معادلات دینامیکی حرکت کوادکوپتر به دست می‌آید.

۳-۵-۲-۱-۶ به دست آوردن معادلات حرکت

در بخش‌های قبل، پارامترهایی که برای مدلسازی دینامیکی کوادکوپتر لازم بودند به دست آمدند. حال، روابطی که برای بیان دینامیک حرکت کوادکوپتر لازم هستند به صورت زیر تعریف می‌شوند.

$$\ddot{x} = \frac{u_1}{m}\cos(\alpha)\left(\cos(\varphi)\sin(\theta)\cos(\psi) + \sin(\varphi)\sin(\psi)\right) - \frac{u_2}{m}\cos(\psi)\sin(\varphi)\sin(\theta)\sin(\alpha) \times \left(1 - \frac{\sqrt{2}}{2}\right) + \frac{u3}{m}\cos(\psi)\cos(\theta)\sin(\alpha) \times \left(1 - \frac{\sqrt{2}}{2}\right) - k_1\frac{\dot{x}}{m}$$

$$\ddot{y} = \frac{u_1}{m}\cos(\alpha)\left(\cos(\varphi)\sin(\theta)\sin(\psi) - \sin(\varphi)\cos(\psi)\right) - \frac{u_2}{m}\sin(\alpha)\left(\cos(\theta)\cos(\psi) + \sin(\varphi)\sin(\theta)\sin(\psi)\right) \times \left(1 - \frac{\sqrt{2}}{2}\right) + \frac{u_3}{m}\cos(\theta)\sin(\psi)\sin(\alpha) \times \left(1 - \frac{\sqrt{2}}{2}\right) - k_2\frac{\dot{y}}{m}$$

$$\ddot{z} = \frac{u_1}{m}\left(\cos(\varphi)\cos(\theta)\cos(\alpha)\right) - \frac{u_2}{m}\cos(\theta)\sin(\varphi)\sin(\alpha) \times \left(1 - \frac{\sqrt{2}}{2}\right) - \frac{u_3}{m}\sin(\theta)\sin(\alpha) \times \left(1 - \frac{\sqrt{2}}{2}\right) - 9.81 - k_3\frac{\dot{z}}{m}$$

$$\ddot{\varphi} = \left(\frac{(I_{yy} - I_{zz})}{I_{xx}}\right)\dot{\theta}\dot{\psi} + \frac{u_2}{I_{xx}}\cos(\alpha)\, l - \frac{k_4}{I_{xx}}\dot{\varphi} + \frac{m \times 9.81}{I_{xx}}\mathrm{d} \times \sin(\varphi)$$
$$\ddot{\theta} = \left(\frac{(I_{zz} - I_{xx})}{I_{yy}}\right)\dot{\varphi}\dot{\psi} + \frac{u_3}{I_{yy}}\cos(\alpha)\, l - \frac{k_5}{I_{yy}}\dot{\theta} + \frac{m \times 9.81}{I_{yy}}\mathrm{d} \times \sin(\theta)$$
$$\ddot{\psi} = \left(\frac{(I_{xx} - I_{yy})}{I_{zz}}\right)\dot{\theta}\dot{\varphi} + \frac{u_4}{I_{zz}}\cos(\alpha) - \frac{k_6}{I_{yy}}\dot{\psi}$$

(۱۴-۳)

که در این رابطه با توجه به شکل (۲-۸) که نشان دهنده نیروها در ترکیب بعلاوه کواکوپترها می‌باشد، داریم:

$$\begin{aligned} k_1 &= k_2 \\ k_4 &= k_5 \\ u_1 &= F_1 + F_2 + F_3 + F_4 \\ u_2 &= F_4 - F_3 \\ u_3 &= F_1 - F_2 \\ u_4 &= F_1 + F_2 - F_3 - F_4 \end{aligned} \qquad (۱۵-۳)$$

در روابط (۳-۱۴) و (۳-۱۵)، F_i ها، نیروی ایجاد شده توسط ملخ i ام است. k_1 ، k_2 و k_3 به ترتیب ضرایب مقاومت ایرودینامیکی کوادکوپتر در حرکت در راستای محور x، y و z است. به همین ترتیب، k_4 ، k_5 و k_6 نیز، ضرایب مقاومت ایرودینامیکی کوادکوپتر در حرکت دورانی حول محور x، y و z است.

همچنین d فاصله عمودی بین مرکز جرم کوادکوپتر و سطح ملخ‌ها است و m وزن کل کوادکوپتر است که با جمع کردن وزن همه تجهیزات اشاره شده در بخش ۳-۳، ۳۱۰۰ گرم به دست می‌آید.

و در نهایت، α نشان دهنده زاویه دایهدرال می‌باشد.

بدین ترتیب، در این بخش مدل‌سازی کوادکوپتر به صورت خلاصه بیان گردید. در ادامه، با توجه به مدل به دست آمده و با در نظر داشتن عدم قطعیت‌های اشاره شده در بخش ۳-۲ که ارتفاع مرکز جرم و ممان اینرسی حول محور x را شامل می‌شود، کنترل کننده مناسب طراحی و شبیه سازی خواهد شد.

۳-۵-۲-۲ طراحی کنترل کننده

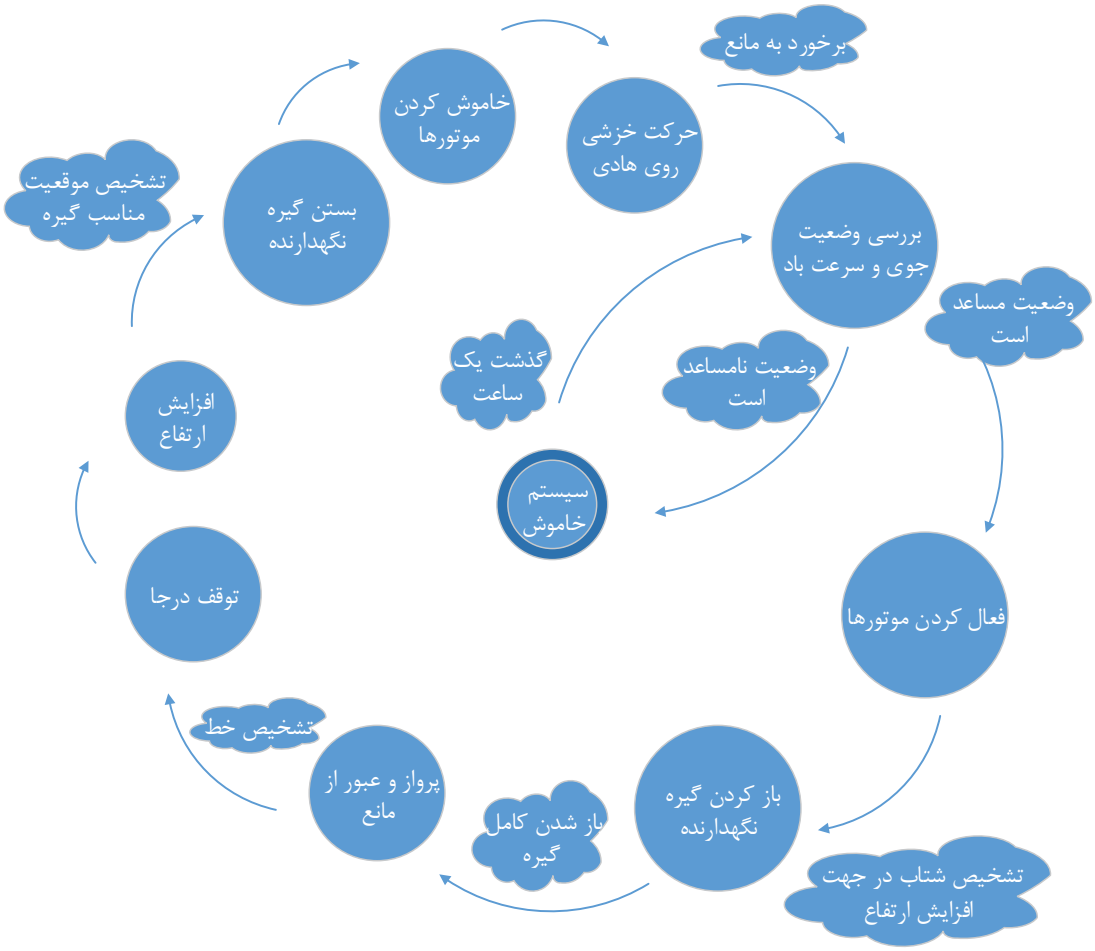
در بخش ۳-۵-۱، مدل غیر خطی کوادکوپتر به دست آمد. در ادامه با طی مراحلی، کنترل کننده خطی مناسب جهت ایجاد پایداری و کنترل زوایا و موقعیت‌های کوادکوپتر، طراحی خواهد شد.

از آنجا که کوادکوپتر چند نوع وظیفه را بر عهده دارد، کنترل کننده آن را به صورت کنترل کننده حالت محدود[1]، طراحی می‌کنیم. بنابراین ابتدا لازم است وظایف و حالت‌هایی که ممکن است برای کوادکوپتر به وجود آید، به صورت زیر شرح داده شود.

- کوادکوپتر در زمان بازرسی از خطوط انتقال، باید بر روی هادی محافظ شبکه حرکت کند.
- در حالت خزشی، دوربین مادون قرمز، از خطوط انتقال بازرسی کند. با یافتن عیبی در خطوط، تصاویر رنگی استریو، تصویر مادون قرمز و موقعیت فعلی ربات را که از GPS گرفته می‌شود، در حافظه خود ذخیره نماید.
- بتواند مانع موجود بر سر راه خود را به موقع تشخیص دهد.
- بعد از تشخیص مانع، ابتدا وضعیت جوی و سرعت باد را اندازه‌گیری کرده و سپس به صورت پروازی از هادی جدا شود.
- در حالت پروازی، دوربین‌های تحتانی، وظیفه ادومتری تصویری را بر عهده گیرد.
- با نزدیک شدن به هادی محافظ، دوربین‌های فوقانی، آن را تشخیص دهد.
- با نزدیک شدن به هادی محافظ، گیره نگهدانده بسته شود و بعد از آن ۴ موتور براشلس

[1] Finite State Controller

خاموش شوند.

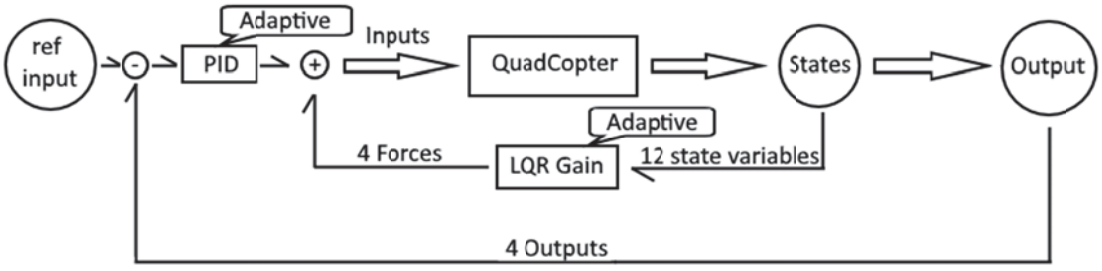


شکل (۳-۱۹) الگوریتم کنترل حالت محدود کوادکوپتر

با در نظر داشتن موارد فوق، می‌توان الگوریتم کنترل کننده حالت محدود را به صورت شکل (۳-۱۹) طراحی نمود.

حال برای ایجاد فرامین کنترلی که جهت اجرای الگوریتم شکل (۳-۱۹) مورد نیاز است، الگوریتم کنترل کوادکوپتر را طراحی و در محیط Matlab شبیه‌سازی می‌کنیم.

در شکل (۳-۲۰) دیاگرام ساده سازی شده الگوریتم کنترل مورد استفاده آورده شده است.



شکل (۳-۲۰) دیاگرام ساده سازی شده الگوریتم کنترل کوادکوپتر

همانطور که در شکل (۳-۲۰) مشاهده می‌شود، برای پیاده سازی کنترل کوادکوپتر، از دو حلقه کنترلی تو-در-تو استفاده می‌شود.

از آنجا که کوادکوپتر، یک سیستم ناپایدار است، ابتدا لازم است که با جایابی مناسب قطب‌ها، قطب-های سمت راست مجور موهومی را جایابی مجدد نمود و سیستم را پایدار کرد. برای این کار، در حلقه داخلی از الگوریتم رگولاتور درجه دو خطی[1] استفاده می‌شود. ورودی این فیدبک، ۱۲ عدد متغیر حالت سیستم است و خروجی آن، ۴ عدد نیرویی است که سعی می‌کند کوادکوپتر را در حالت پایدار حفظ کند.

سپس برای اینکه این سیستم پایدار سازی شده، بتواند نقطه هدف مشخصی را دنبال نماید، در حلقه بیرونی الگوریتم کنترل، از کنترل کننده PID استفاده می‌شود. ورودی این فیدبک، موقعیت‌های خطی و زاویه یاو کوادکوپتر است و خروجی آن نیز، ۴ عدد نیرویی است که سعی می‌کند کوادکوپتر را به نقطه هدف برساند.

قابل ذکر است، از آنجا که در صورت استفاده از رویتگر حالت در سیستم کنترلی (مانند فیلتر کالمن تعمیم یافته)، با فاصله گرفتن کوادرکوپتر از نقطه پایداری(زوایای رول و یا پیچ بیشتر از ۱۵ درجه)، رویتگرها دیگر قابلیت تخمین دقیق تمامی متغیرهای حالت را ندارند و امکان ناپایداری سیستم به وجود می‌آید، در این پایان نامه تصمیم گرفته شد تا با استفاده از حسگرها و با استفاده از قابلیت‌های

[1] Linear Quadratic Regulator (LQR)

نرم افزاری همچون ادومتری تصویری و تلفیق حسگرها، تمامی ۱۲ عدد متغیر حالت را به صورت مستقیم و بدون استفاده از رویتگر حالت، به دست آوریم. بدین ترتیب نواحی کار کوادکوپتر را از حالت محدود شده خارج می‌کنیم.

همانطور که در بخش ۳-۲ اشاره شد، بعضی از پارامترهای دینامیکی سیستم ربات، متناسب با باز و بسته شدن گیره نگهدارنده، تغییر پیدا می‌کند. بنابراین لازم است، ضرایب فیدبک به صورت زنده و متناسب با تغییرات به وجود آمده در پارامترهای دینامیکی کوادکوپتر تغییر پیدا کنند. این موضوع، لزوم طراحی کنترل کننده تطبیقی LQR+PID را نشان می‌دهد. این موضوع در شکل (۳-۲۰) نیز نمایش داده شده است. (کلمه Adaptive درج شده روی فیدبک‌ها)

برای اینکه به حالت تظبیقی دست پیدا کنیم، نیاز به ابزار زیر خواهیم داشت:

۱- تخمین‌گری که بتواند تغییرات به وجود آمده در پارامترهای دینامیکی سیستم را به صورت زنده محاسبه نماید.

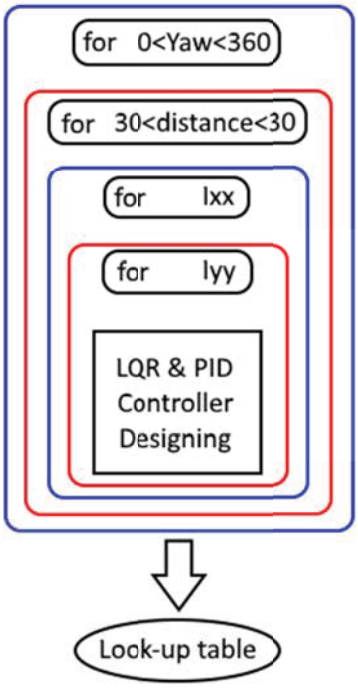
۲- با به دست آمدن پارامترها، نیاز به یک جدول جستجو[1] داریم تا با جستجو در آن، ضرایب فیدبک متناسب با پارامترهای دینامیکی تغییر یافته را، حین کنترل ربات به صورت زنده استفاده نماییم.

برای به دست آوردن این جدول، به صورت زیر عمل می‌کنیم.

مطابق شکل (۳-۲۱) ۴ حلقه تو-در-تو ایجاد می‌کنیم که در حلقه بیرونی، مقدار زاویه یاو، از صفر تا ۳۶۰ درجه با دقت نیم درجه تغییر داده می‌شود. در حلقه داخل این حلقه، مقدار فاصله عمودی مرکز جرم از سطح ملخ‌ها بین منفی ۳۰ سانتی‌متر و مثبت ۳۰ سانتی‌متر با دقت ۵ میلی متر تغییر می‌کند. در حلقه‌های داخل‌تر، مقدار ممان اینرسی x و y بین

[1] Look-up table

حداکثر و حداقل مقادیر ذکر شده در جدول (۳-۱) (بین ۰.۰۲ و .۰.۰۶ یا دقت ۰.۰۰۵) تغییر داده می‌شوند. در داخل‌ترین حلقه، معادلات غیر خطی کوادکوپتر، با توجه به پارامترهای تغییر یافته، باز تعریف می‌شود و بعد از خطی سازی و تعریف پارامترهای دیگر، ضرایب فیدبک LQR و PID در هر مرحله محاسبه می‌شوند.



شکل (۳-۲۱) نحوه محاسبه و به دست آوردن جدول جستجو

بنابر توضیحات داده شده، ما ضرایب فیدبک LQR و PID را در ۴ بعد بسط دادیم و در نهایت نیز جدول به دست آمده را ذخیره کردیم تا در شبیه سازی عملکرد کنترل کننده، مورد استفاده قرار گیرد.

حال به طراحی کنترل کننده می‌پردازیم. مراحل طراحی به صورت زیر است.

۱- خطی‌سازی معادلات غیر خطی

۲- تعیین ماتریس‌های مثبت معین و نیمه معین Q و R

۳- بررسی کنترل پذیری و رویت‌پذیری سیستم

۴- طراحی فیدبک حالت LQR

۵- طراحی سیستم حلقه بسته

۶- به دست آوردن توابع تبدیل بین ۴ ورودی و ۱۲ خروجی

۷- طراحی فیدبک PID

۸- طراحی تخمین‌گر حداقل مربعات بازگشتی[1] برای تخمین ارتفاع نقطه مرکز جرم و ممان اینرسی حول محور x

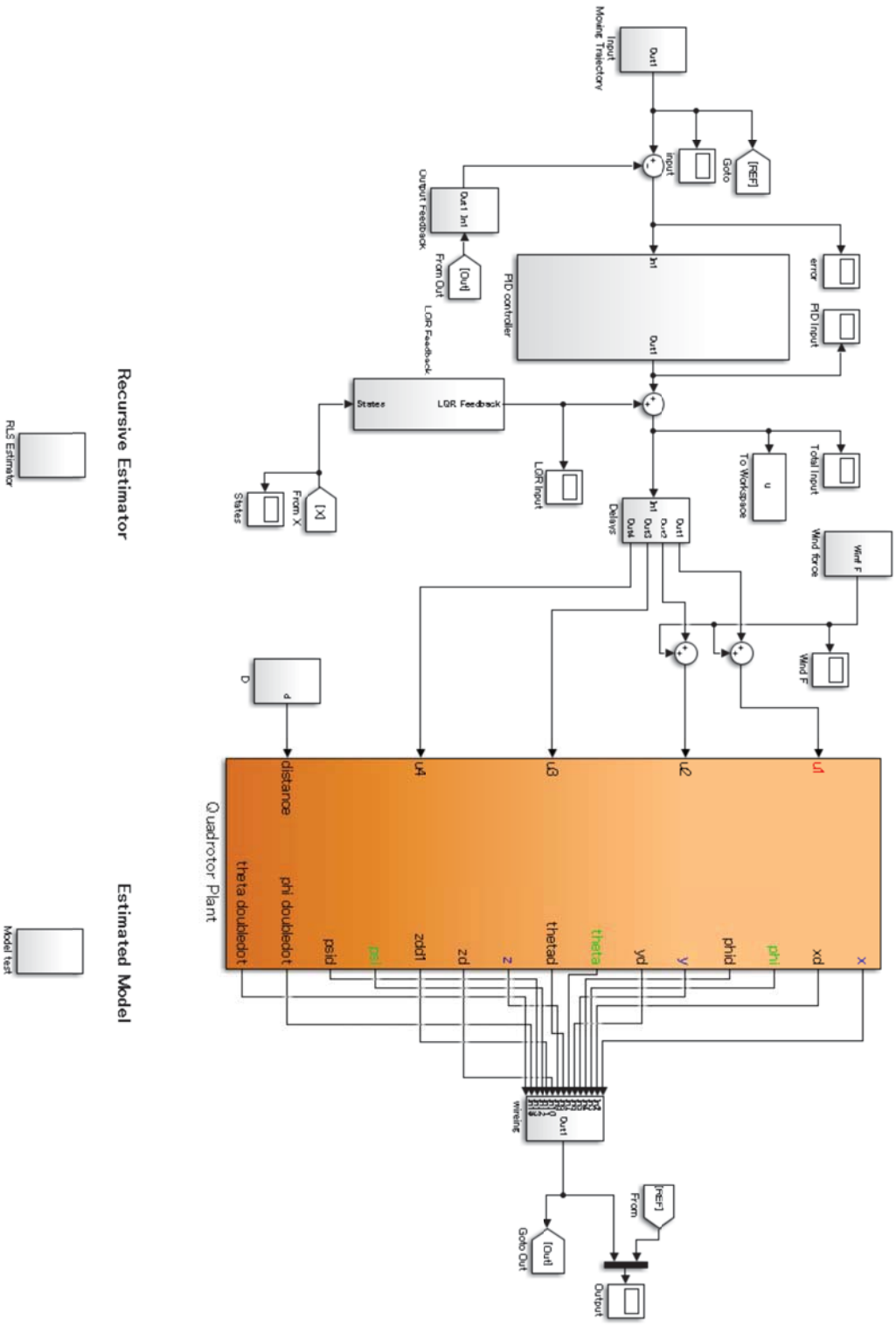
با طی مراحل فوق، کنترل کننده‌ای به صورت شکل (۳-۲۲) به وجود می‌آید. این کنترل کننده از نوع تطبیقی است و می‌تواند به صورت زنده، پارامترهای فیدبک خود را متناسب با تغییرات ارتفاع مرکز جرم کوادکوپتر و ممان اینرسی حول محور x تغییر دهد.

در شکل (۳-۲۳) دیاگرام داخلی بلوک توصیف کننده دینامیک کوادکوپتر، نمایش داده شده است.

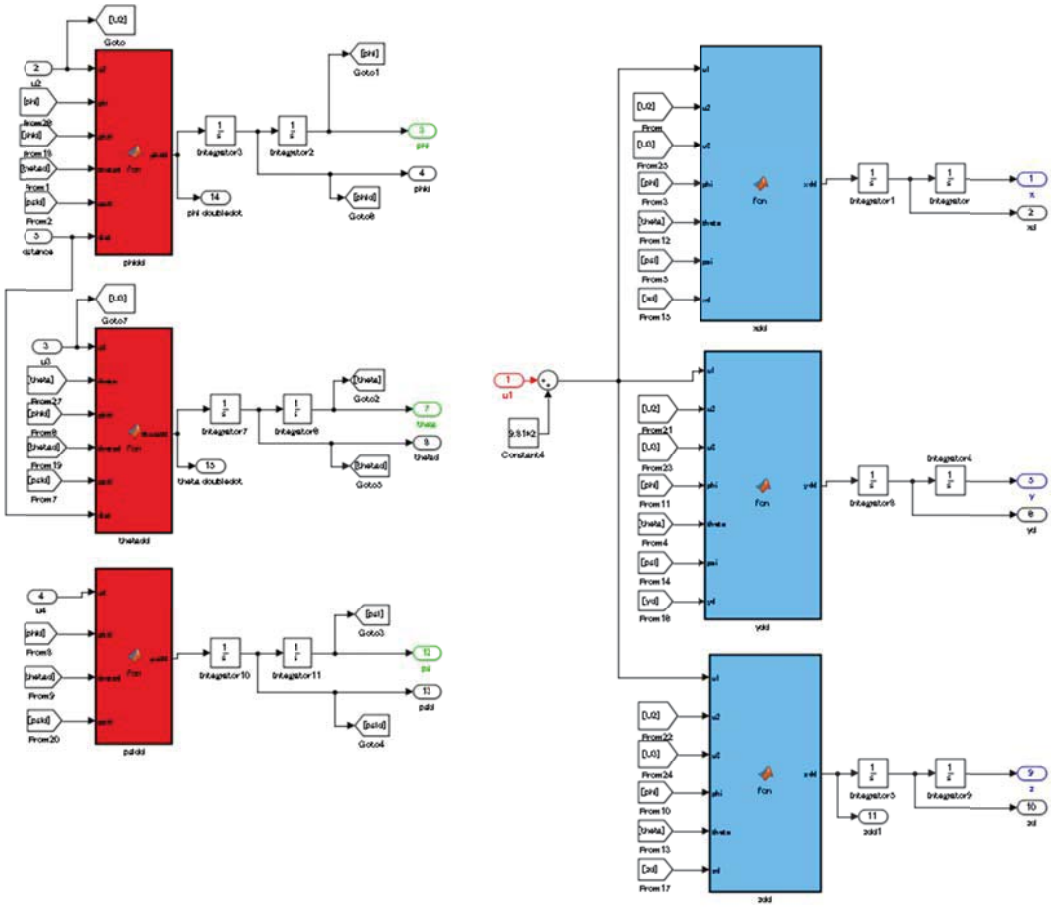
همانطور که در شکل (۳-۲۲) نمایش داده شده است، مقدار نیروی محاسبه شده از طریق فیدبک PID با فیدبک LQR باهم جمع می‌شوند و با عبور از فیلتر تاخیر زمانی و اعمال نویز محیطی (نیروی باد) به سیستم غیر خطی کوادکوپتر وارد می‌شود. با توجه به مطالب اشاره شده، کنترل کننده PID این سیستم، بر اساس سیستم پایدار شده توسط LQR طراحی و محاسبات آن صورت گرفته است، لذا کاملاً با یکدیگر سازگارند و جمع کردن این دو فیدبک مشکلی را متوجه سیستم نخواهد کرد. صحت این موضوع را در شکل‌های بعدی نمایش خواهیم داد.

در شکل (۳-۲۴) تخمین‌گر حداقل مربعات بازگشتی طراحی شده نمایش داده شده است.

[1] Recursive Least Square Estimator



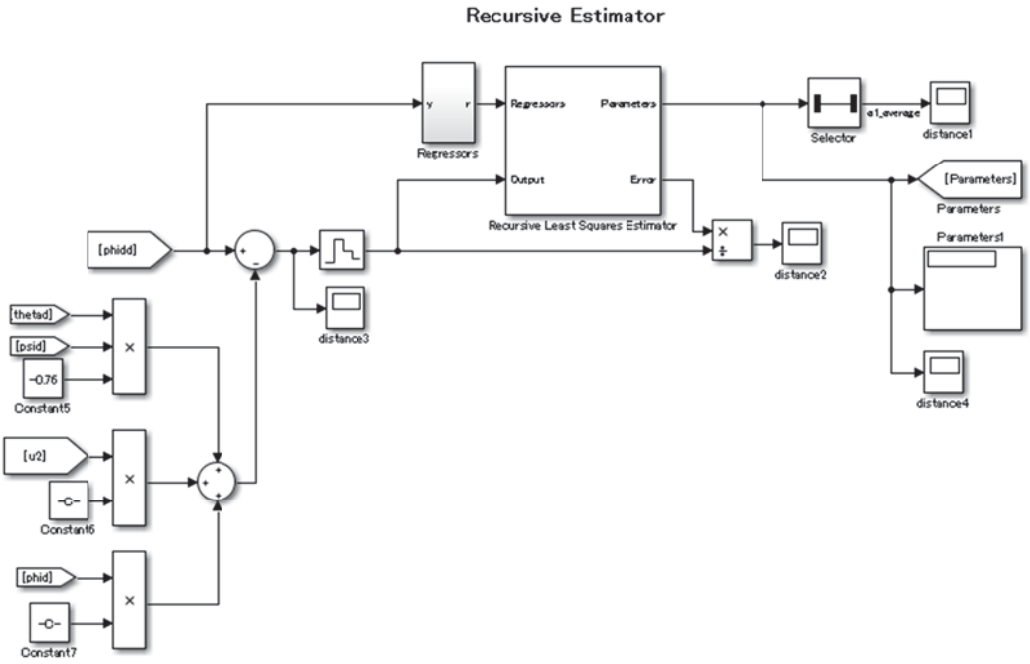
شکل (۳-۲۲) کنترل کننده تطبیقی طراحی شده



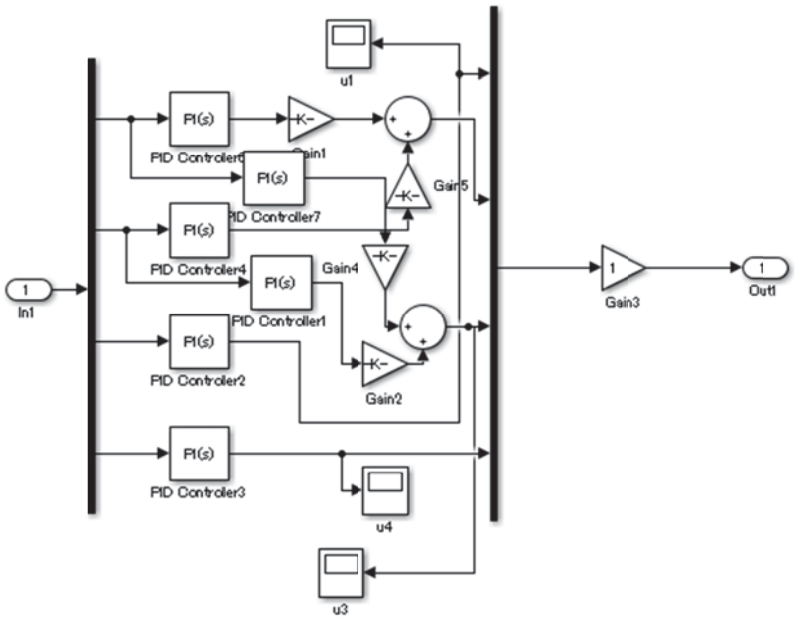
شکل (۳-۲۳) دیاگرام داخلی بلوک توصیف کننده دینامیک کوادکوپتر. ۶ معادله اصلی به صورت بلوکی طراحی شده است و مابقی به صورت چند بلوک انتگرال آورده شده است.

در شکل (۳-۲۵) دیاگرام داخلی بلوک کنترل کننده PID نمایش داده شده است.

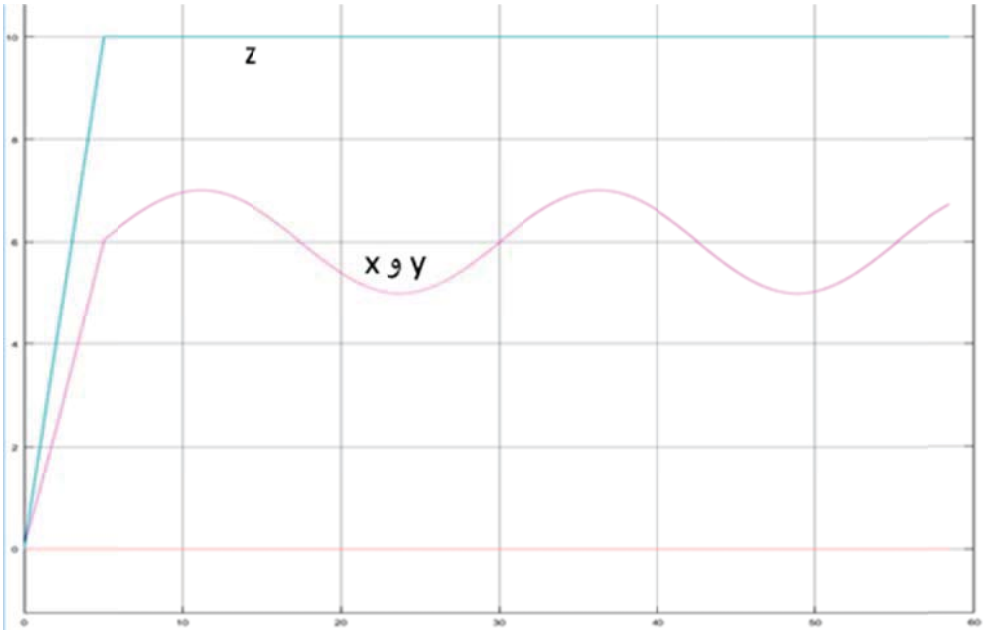
با اجرای کد، نتایج به صورت زیر به دست می‌آید. در شکل (۳-۲۶)، ورودی کنترل کننده که طول، عرض و ارتفاع مسیر مورد انتظار جهت عبور کوادکوپتر که شامل شیب و مسیر سینوسی است، نمایش داده شده است.



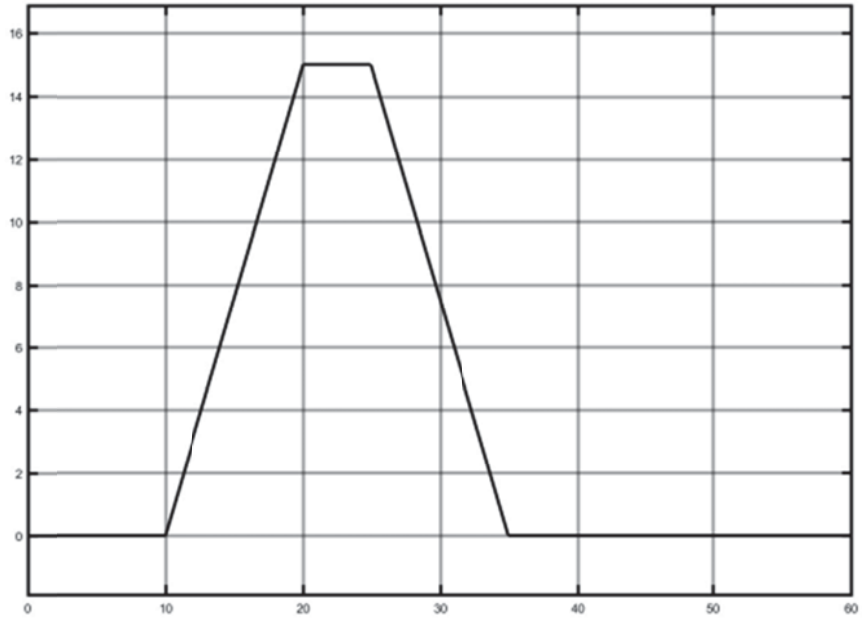
شکل (۳-۲۴) تخمین‌گر حداقل مربعات بازگشتی طراحی شده



شکل (۳-۲۵) دیاگرام داخلی بلوک کنترل کننده PID



شکل (۳-۲۶) ورودی اعمالی به سیستم



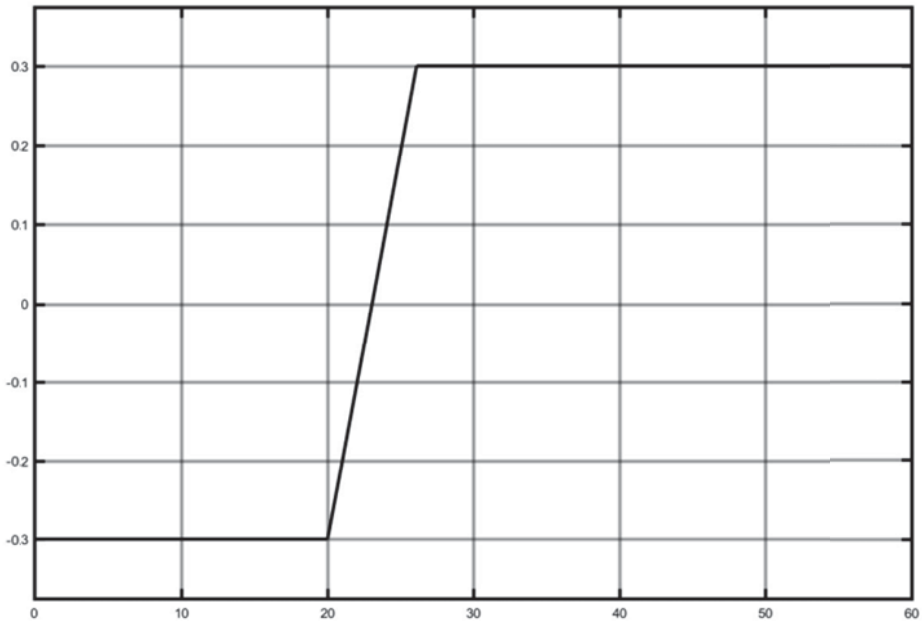
شکل (۳-۲۷) موج نیروی اغتشاش ناشی از باد

در شکل (۳-۲۷)، سیگنال نیروی اغتشاش که بر اثر وزش باد به کوادکوپتر اعمال می‌شود، نمایش داده شده است. نیرو یا گشتاور باد فرضی، ابتدا با شیب ثابت افزایش پیدا می‌کند و پس از رسیدن به

مقدار ۱۵ نیوتن، به مدت ۵ ثانیه ثابت می‌ماند و سپس با همان شیب کاهش پیدا کرده و با گذشته زمان ۱۰ ثانیه به صفر می‌رسد. این نیرو در جهت محور z، ۱۵ نیوتن مثبت را وارد کرده و حول محور x نیز گشتاور ۱۵ نیوتن متر را وارد می‌کند.

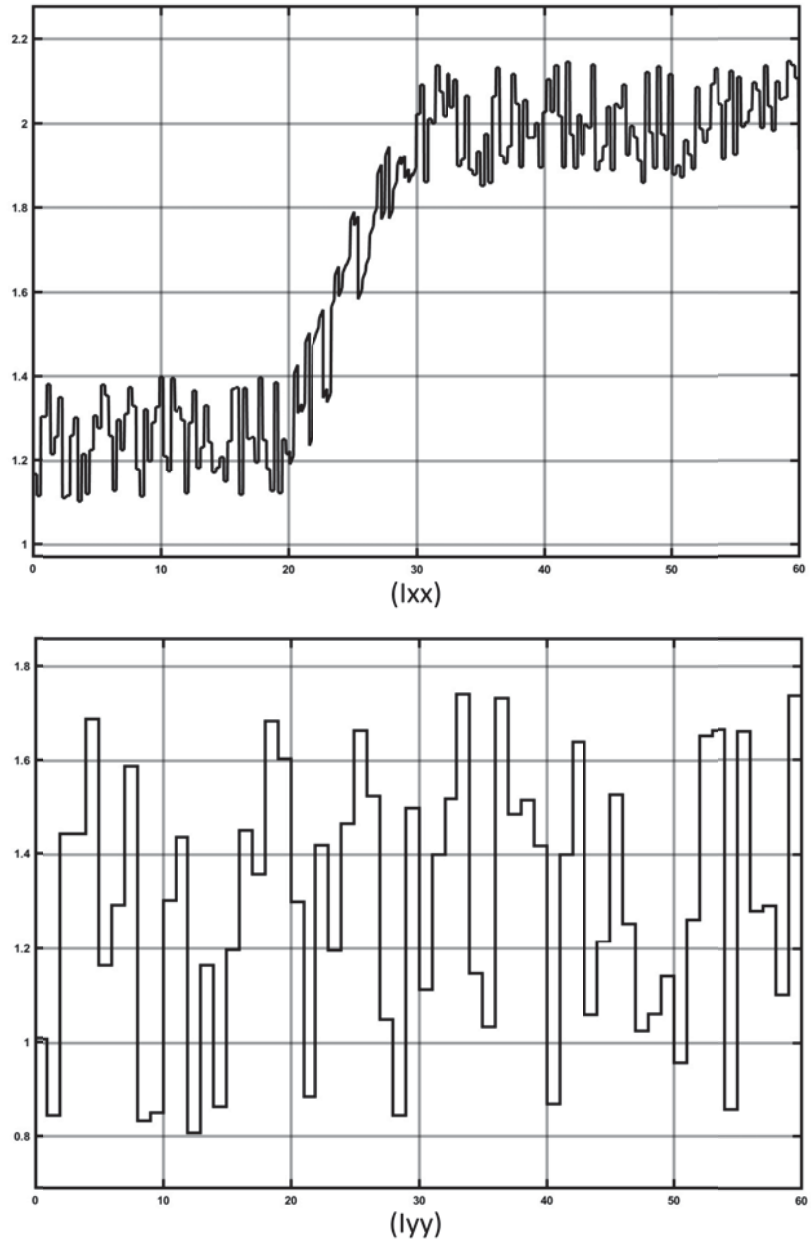
در شکل (۳-۲۸) نحوه تغییر ارتفاع مرکز جرم کوادکوپتر نمایش داده شده است. این تغییر، در زمان ۲۰ ثانیه شروع می‌شود که کوادکوپتر در آن زمان، به ثبات نسبی رسیده است (مطابق فرضیات مساله کنترل).

در شکل (۳-۲۹) نحوه تغییر ممان اینرسی x و y نمایش داده شده است. تغییرات ممان اینرسی x در زمان ۲۰ ثانیه شروع می‌شود و از تغییزات ممان اینرسی y که به صورت تصادفی بین ۰.۸ الی ۱.۷۵ با فرکانس ۱ هرتز در حال تغییر است، به عنوان عدم قطعیت در شبیه سازی استفاده شده است.



شکل(۳-۲۸) تغییر ارتفاع مرکز جرم کوادکوپتر در زمان ۲۰ ثانیه

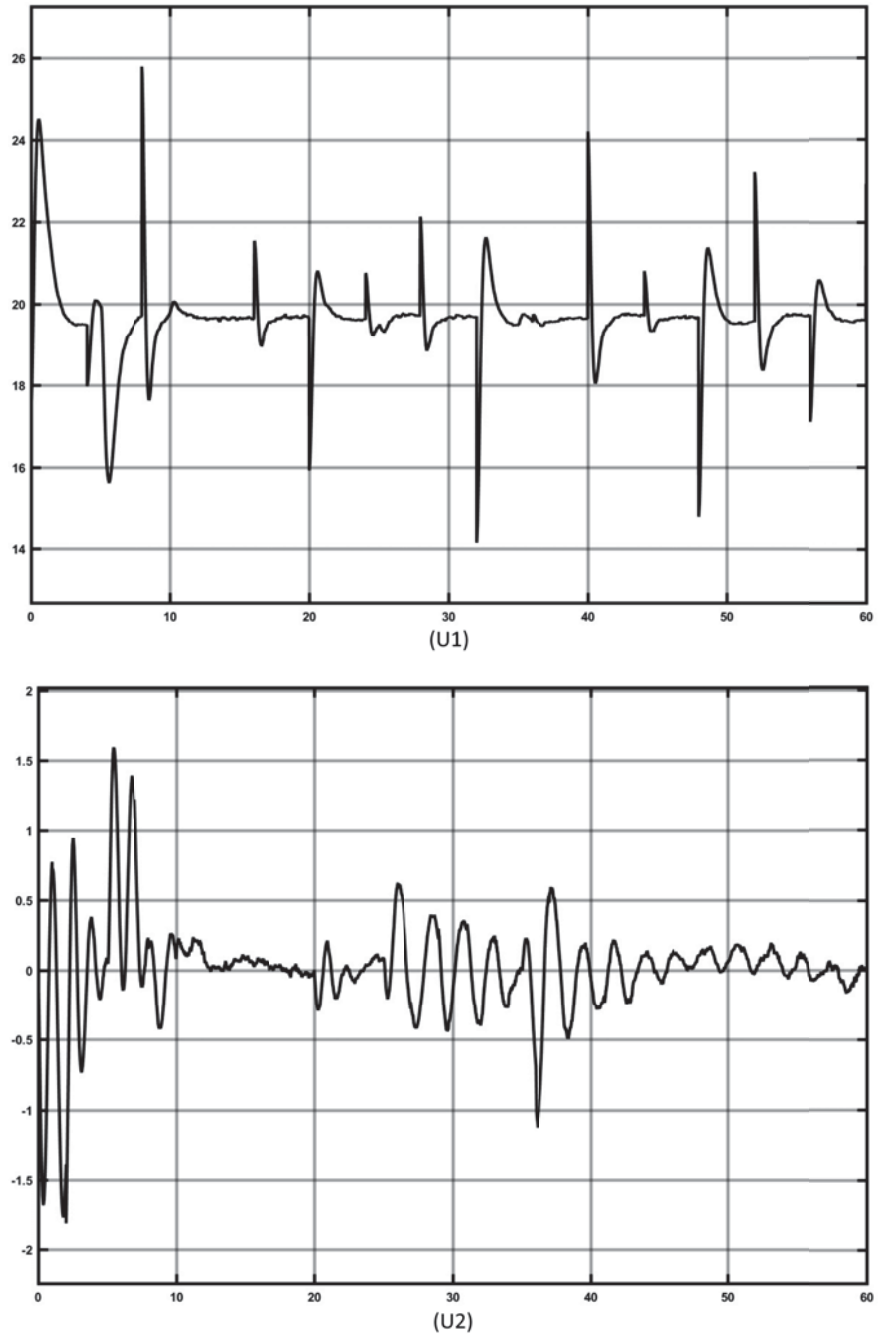
برای بررسی بهتر و واقع گرایانه‌تر عملکرد کوادروتور در شرایط واقعی، علاوه بر نیروی اغتشاش وارد شده مطابق شکل (۳-۲۷)، نیاز است که در پارامترهای فیزیکی (مانند ممان اینرسی، نیروی ایجاد شده توسط موتورها و ...) عدم قطعیتی را در نظر بگیریم.



شکل (۳-۲۹) تغییرات ممان اینرسی x و y

برای این منظور، در این شبیه سازی برای نیروی u_1 که مجموع جبری نیروهای ایجاد شده توسط موتورها در راستای عمود بر صفحه ملخ‌ها را نمایش می‌دهد، عدم قطعیت ۲۰± درصدی کاملاً تصادفی با فرکانس ۰.۲۵ الی ۵ هرتز اعمال می‌شود. همچنین برای نیروهای u_2، u_3 و u_4 نیز، عدم قطعیت ۲۰± درصدی کاملا تصادفی با فرکانس ۰.۲۵ الی ۱۰ هرتزی اعمال می‌شود. در شکل (۳-

۳۰)، نیروی u_1 و u_2 بعد از اعمال عدم قطعیت ۱ هرتزی تصادفی بر آن، برحسب نیوتن نمایش داده شده است. از آنجا که مشخصات عدم قطعیت برای u_3 و u_4 نیز مشابه است، از آوردن شکل مربوط به آن دو نیرو، صرف نظر می‌شود.



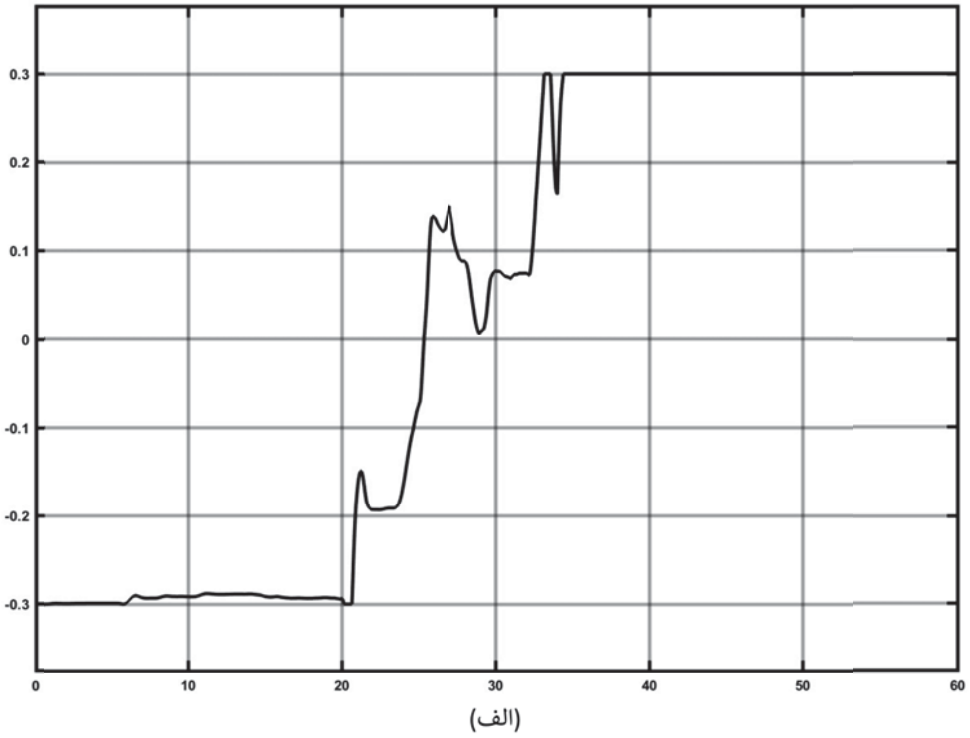
شکل (۳-۳۰) نیروهای اعمال شده به دینامیک غیر خطی کوادکوپتر بعد از اعمال عدم قطعیت

حال می‌خواهیم عملکرد تخمین‌گر بازگشتی حداقل مربعات را در ایجاد حالت تطبیقی به سیستم کنترل، مورد ارزیابی قرار دهیم. همانطور که در شکل های (۳-۲۸) و (۳-۲۹) نمایش داده شد، ارتفاع مرکز جرم و ممان اینرسی کوادکوپتر، با باز و بسته شدن گیره نگهدارنده نصب شده بر روی آن، دچار تغییرات می‌شود. برای اینکه سیستم کنترلی، بتواند ضرایب فیدبک را در حین این تغییرات اصلاح نماید و کوادکوپتر از حالت پایدار و کنترل پذیر خارج نشود، تخمین گر بازگشتی اشاره شده بایستی توانایی تخمین این تغییرات را به صورت زنده داشته باشد. ارتفاع مرکز جرم و ممان اینرسی تخمین زده شده توسط این تخمین‌گر، وارد جدول جستجو شده و ضرایب فیدبک متناسب را در سیستم کنترلی به کار می‌گیرد.

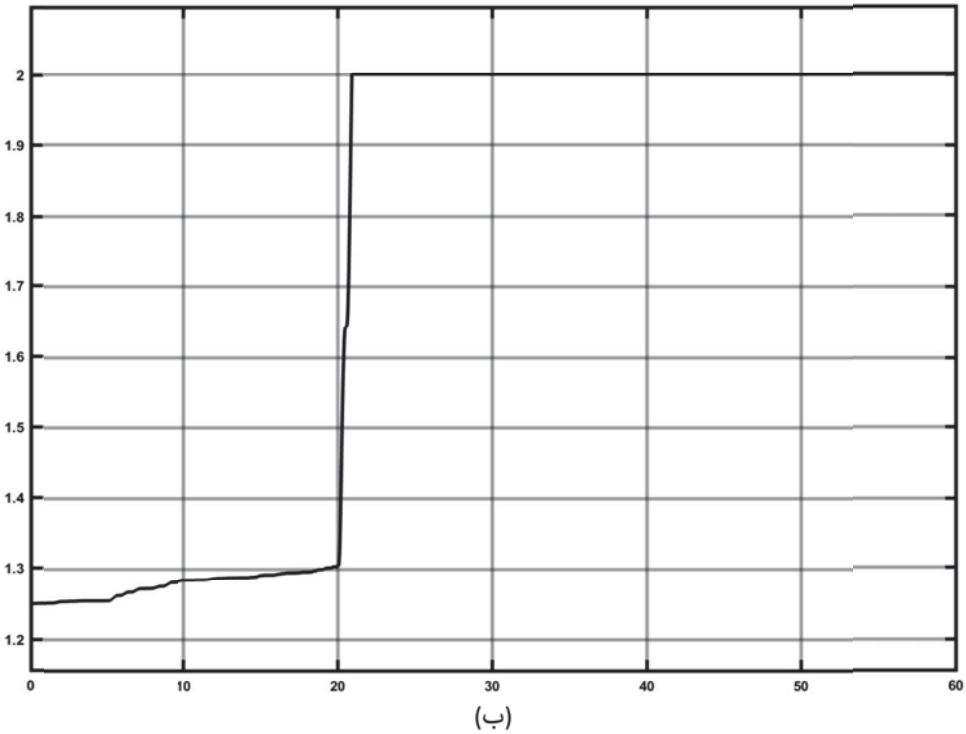
در شکل (۳-۳۱)، ارتفاع مرکز جرم و ممان اینرسی تخمین زده شده نمایش داده شده است.

با در نظر داشتن تمامی ورودی‌های فوق (اعم از ورودی مرجع، اغتشاش، عدم قطعیت و تغییرات پارامترها)، خروجی کوادکوپتر که موقعیت‌های خطی و زاویه‌ای را شامل می‌شود، در شکل (۳-۳۲) نمایش داده شده است. واحدهای اندازه گیری جابجایی خطی و زاویه ای در این شکل، به ترتیب متر و گرادیان است. در این شکل، کاملاً مشخص است که ربات توانسته سیگنال ورودی را با حداقل تاخیر دنبال کند و در حالت ماندگار نیز خطای رهگیری را به صفر برساند.

برای بررسی بهتر عملکرد کنترل کننده، نمودار خطای بین ورودی و خروجی سیستم کنترل در شکل (۳-۳۳) نمایش داده شده است.

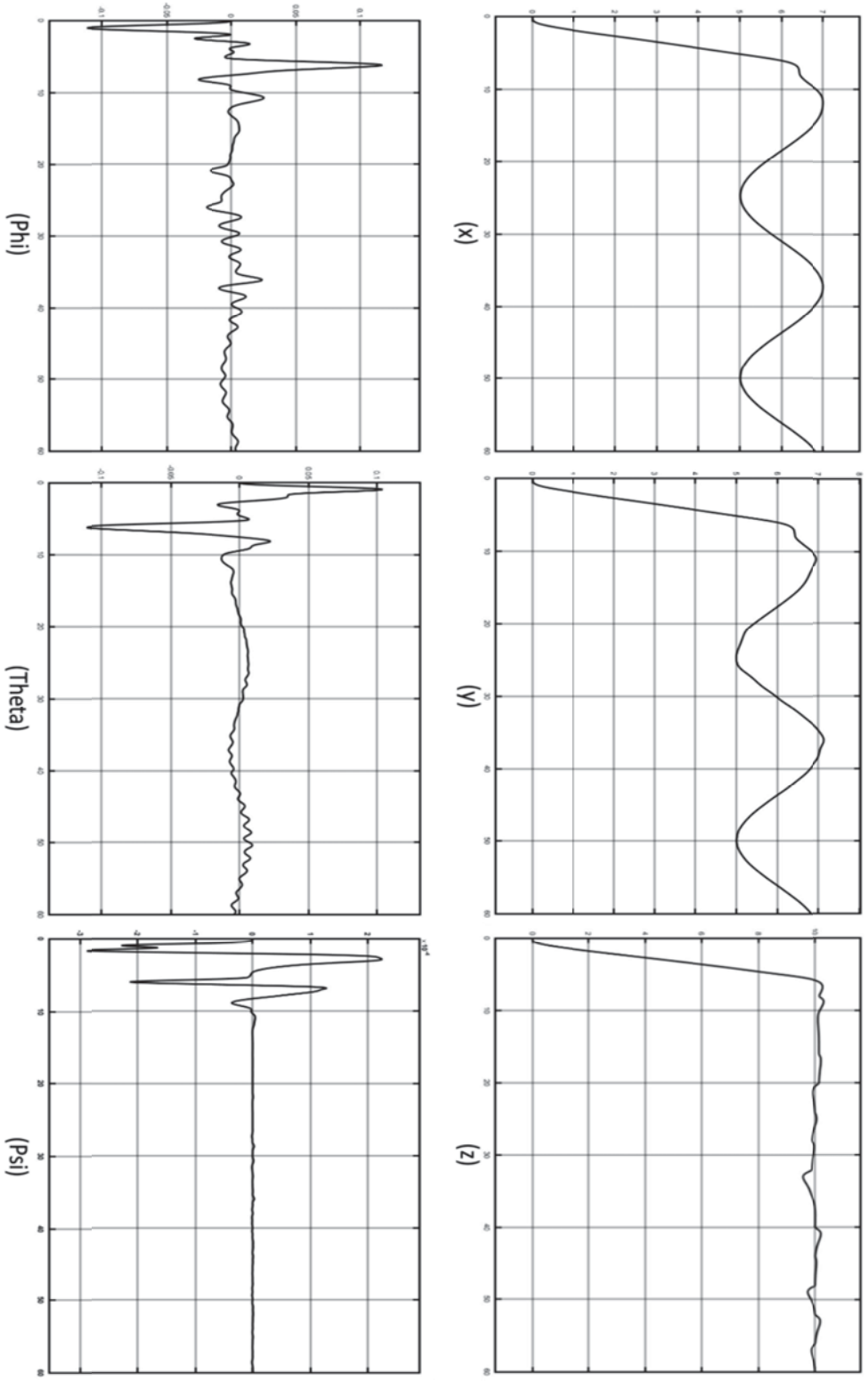


(الف)



(ب)

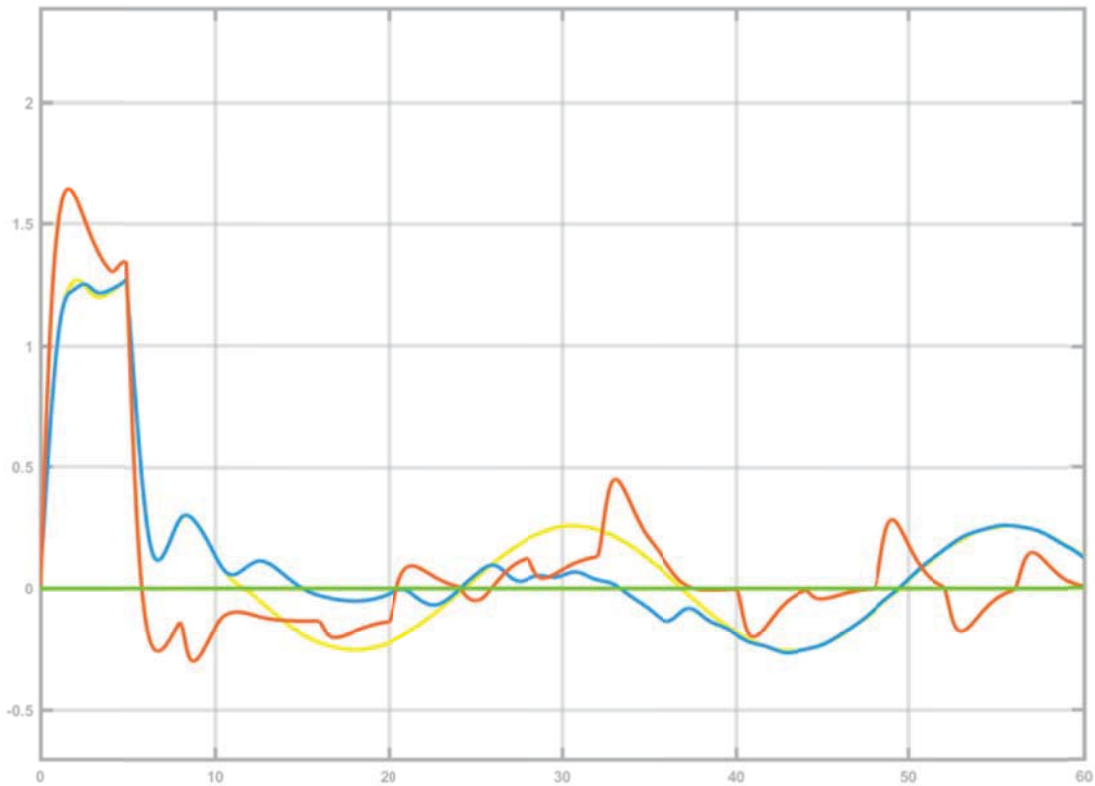
شکل (۳-۳۱): (الف) ارتفاع مرکز جرم تخمینی. (ب) ممان اینرسی x تخمینی


(Phi)
(x)
(Theta)
(y)
(Psi)
(z)

شکل (۳-۳۲) موقعیت‌های خطی و زاویه‌ای شبیه سازی شده کوادکوپتر

همانطور که در شکل (۳-۳۳) مشاهده می‌شود، خطای بین ورودی و خروجی کنترل کننده، در ابتدای حرکت زیاد است و به مرور زمان کاهش پیدا می‌کند و در نهایت به حداقل مقدار ممکن (صفر برای z و مقادیر سینوسی برای x و y) رسیده است. این شکل به خوبی نمایش می‌دهد که کنترل کننده طراحی شده برای کوادکوپتر مورد نظر، به خوبی تغییرات داخلی سیستم خود را شناسایی کند و مسیر داده شده را دنبال نماید.

با مشاهده همزمان شکل‌های (۳-۳۱) الی (۳-۳۳)، متوجه می‌شویم که علیرغم وجود اغتشاشات ناشی از باد و تغییرات پارامتری ناشی از جابجایی ارتفاع مرکز جرم، الگوریتم کنترل توانسته است ربات را در مسیر مطلوب خود هدایت کند و پایداری آن را حفظ نماید.



شکل (۳-۳۳) خطای بین ورودی و خروجی کنترل کننده

۳-۶ نتیجه گیری

در این فصل، مراحل مختلف طراحی فیزیکی کوادکوپتر بازرسی کننده از خطوط انتقال فشار قوی و الگوریتم‌های مورد استفاده در آن برای این کار مورد بحث قرار گرفت. ابتدا نیازها و وظایف این ربات به صورت کلی بیان گردید و سپس برای هریک از زیر بخش‌های آن همانند گیره نگهدارنده، کنترل کننده مرکزی و بدنه کوادکوپتر، ابتدا با تعریف تمامی وظایف و شرح ابزار آلات و شرایط مورد نیاز برای انجام آنها، طراحی مهندسی در محیط solidworks انجام شد.

از اطلاعات خواص فیزیکی و دینامیکی به دست آمده از محیط solidworks، برای بیان نیازهای کنترل کننده استفاده شد و کنترل کننده تطبیقی LQR+PID طراحی و شبیه سازی آن صورت گرفت. در نهایت نیز با مشاهده نتایج شبیه سازی انجام شده، از صحت عملکرد الگوریتم طراحی شده اطمینان حاصل گردید.

در بخش بینایی ماشین و پردازش تصویر، ابتدا برای توقف خودکار عملکرد کوادکوپتر در شرایط جوی نامساعد، الگوریتم تشخیص وضعیت جوی مورد بحث و طراحی قرار گرفت و با انجام آزمایشی در محیط Matlab، دقت ۹۱ درصدی در تشخیص وضعیت جوی حاصل گردید. این نتیجه با توجه به تحقیقات صورت گرفته تاکنون جز بهترین نتایج در این زمینه می‌باشد.

فصل چهارم:

جمع‌بندی و نتیجه‌گیری

۴-۱ جمع‌بندی

به منظور بازرسی و یا تعمیر خطوط انتقال برق از نیروی مجرب و متخصص در این زمینه استفاده می‌گردد. اما به خاطر شرایط خطرناک محیط کاری نظیر ولتاژ بالای خطوط فشار قوی و ارتفاع زیاد از سطح زمین، خطرات بسیار زیادی نیروی انسانی شاغل در این بخش از صنعت برق را تهدید می‌کند. امروزه بکارگیری ربات‌هایی که قابلیت استفاده در شرایط سخت محیطی را دارند در صنایع مختلف از جمله صنعت برق بسیار رایج گردیده است. در این پروژه سعی گردیده است تا استفاده از ربات‌های پرنده بعنوان راهکار پیشنهاد گردد. رباتی که در این پروژه برای بازرسی و نگهداری خطوط انتقال برق طراحی و کنترل آن ارائه گردیده است، ربات پرنده‌ای است که با سوار شدن بر خطوط فشار قوی توسط یک گیره نگهدارنده، عیوب عناصر خطوط انتقال را تصویربرداری کرده و تصاویر را توسط خطوط انتقال به مرکز کنترل خطوط ارسال می‌نماید. بعضی از ربات‌هایی که تا کنون مورد استفاده قرار می‌گیرند صرفاً پرنده و بعضی دیگر صرفاً بر روی خطوط سوار شده و با حرکت بر روی خط عیب‌یابی می‌کنند که هرکدام از این ربات‌ها شامل ایرادهایی هستند. ربات طراحی شده در این پروژه، ربات پرنده‌ای است که قابلیت پرواز و حرکت روی خطوط انتقال برق را همزمان دارا می‌باشد. بدین منظور گیره نگهدارنده طراحی گردیده که بر روی ربات پرنده عمودپرواز معمولی یا کوادکوپتر نصب شده و ربات توسط این گیره روی کابل محافظ خطوط انتقال (شیلد وایر) نگه داشته می‌شود. طراحی و کنترل این گیره و کوادکوپتر مناسب برای این منظور جهت بازرسی و نگهداری خطوط انتقال برق فشار قوی هدف اصلی این پروژه می‌باشد. بدین منظور حرکت ربات پرنده بر روی این کابل مورد بررسی، تجزیه و تحلیل و مدل‌سازی قرار گرفت. روش کار این ربات بدین ترتیب است که، این ربات ابتدا به صورت دستی راه اندازی شده و توسط ریموت کنترل تا نزدیکی خطوط انتقال هدایت می‌شود

و سپس به صورت کاملاً مختار، با استفاده از گیره نگهدارنده طراحی شده بر روی کابل محافظ خطوط انتقال سوار شده و شروع به حرکت می‌کند. ربات با حرکت روی خطوط انتقال شروع به عیب‌یابی کرده و اطلاعات ضبط شده را توسط خطوط انتقال برای مرکز کنترل خطوط انتقال ارسال می‌کند. ربات در حین حرکت روی خطوس انتقال، به محض رسیدن به موانع، از خط جدا شده و بعد از عبور از مانع مجدداً بر روی خط سوار شده و به مسیر خود ادامه می‌دهد. این حرکت تا رسیدن به برج‌های دارای مقره‌های کششی ادامه پیدا می‌کند. عمل بازرسی در این ربات‌ها با استفاده از دوربین مادون قرمز، دوربین ماورا بنفش و یا دوربین عادی صورت می‌گیرد.

طراحی ظاهر فیزیکی و الگوریتم کلی هدایت و کنترل این ربات بایستی به گونه‌ای باشد که بتواند موقعیت دقیق خود را نسبت به خطوط شبکه انتقال محاسبه کند و به گونه‌ای خود را کنترل کند که ضمن تامین پایداری و مقاومت در برابر عوامل محیطی، بر روی خط سوار شود و بر روی آن حرکت نماید.

۱- طراحی گیره نگهدارنده، براساس نکات اصلی زیر انجام گرفته است.

- حداقل وزن (کمتر از حداکثر ظرفیت قابل حمل کوادکوپتر)
- مرکز جرم مجموعه کوادکوپتر در حال بسته بودن گیره نگهدارنده پایین‌تر از سطح ملخ‌ها باشد.
- نسبت به محورهای افقی x و y کاملا متقارن باشد.
- گیره باید بتواند خطاهای انباشته شده در موقعیت‌یابی کوادکوپتر را جبران کند و برای گرفتن کابل، محیط بزرگی را دربر بگیرد.
- نیرو و گشتاور کافی را جهت نگهداری کوادکوپتر بر روی کابل فراهم کند.
- در مقابل وزش باد نسبتاً شدید مقاومت لازم را داشته باشد.
- به حداقل تعداد محرک نیاز داشته باشد.

- در حالت بسته بودن دسته گیره نگهدارنده، سطح موثر جانبی آن تا حد امکان کوچک باشد.

با توجه به اصول فوق، مشخصات فیزیکی و هندسی گیره نگهدارنده طراحی شده عبارت است از: جرم ۵۳۳ گرم، طول گیره باز و بسته بترتیب ۲۴۲.۶۵ و ۲۳۱.۷۶ میلی‌متر، عرض گیره باز و بسته ۱۴۵.۴۵ میلی‌متر، ارتفاع گیره باز و بسته بترتیب ۱۶۸.۸۱ و ۲۴۸.۸۱ میلی‌متر، طول مرکز جرم گیره باز و بسته بترتیب ۴.۶۶- و ۰.۴۵- میلی‌متر، عرض مرکز جرم گیره باز و بسته بترتیب ۳.۰۹- و ۱.۱۶ میلی‌متر، ارتفاع مرکز جرم گیره باز و بسته بترتیب ۸۶.۶۶ و ۱۳۸.۷۷ میلی‌متر می‌باشد. جنس قطعات این گیره نگهدارنده از جنس آلومینیوم بوده و از فوم و استیل نیز استفاه گردیده است. این گیره نگهدارنده دارای ۲ موتور DC و یک موتور سروو[1] و دو بازوی اصلی نگهدارنده می‌باشد.

۲- طراحی کوادکوپتر به گونه‌ای انجام شده است که:

- جهت دستیابی به پایداری بهتر، ارتفاع مرکز جرم کوادکوپتر باید از سطح ملخ‌ها پایین‌تر باشد..
- فریم و دسته‌های کوادکوپتر در مقابل وزش باد، کمترین مقاومت را داشته باشد.
- اندازه بازوها به گونه‌ای باشد که دوربین‌های استریو نصب شده جهت دید به سمت پایین، فاصله مناسبی را از یکدیگر داشته باشد.

در طراحی کوادکوپتر از تجهیزاتی نظیر ۴ عدد دوربین رنگی، ۱ عدد دوربین مادون قرمز، ۴ عدد واحد اندازه‌گیری حرکتی، ۱ عدد حسگر GPS، ۴ عدد موتور براشلس به همراه ملخ، باطری، کنترل کننده مرکزی و گیره نگهدارنده استفاده گردیده است.

[1] Servo motor

۳- انتخاب کنترل‌کننده مرکزی کوادکوپتر با توجه به نکات زیر انجام گرفته است.

- قادر به مدیریت حجم اطلاعات بسیار زیاد باشد.
- باید بتواند تصاویر گرفته شده توسط ۴ دوربین را به طور همزمان پردازش کند.
- از سیستم عامل لینوکس پشتیبانی کند.
- باید قابلیت نصب حسگرها، محرک‌ها و تجهیزات ارتباطی نظیر wifi و شبکه رادیویی را به صورت ماژولار داشته باشد.
- وزن آن سبک باشد.
- ابعاد آن بیش از حد بزرگ نباشد.

بعد از انتخاب پردازنده مرکزی، فاصله کانونی دوربین‌ها محاسبه گردید و سپس محل دقیق نصب آنها بر روی کوادکوپتر مشخص گردید. نتایج محاسبات نشان داد که فاصله بین دو دوربین در زیر کوادکوپتر ۱ متر و فاصله دوربین‌ها در بالای کوادکوپتر ۸.۴۲ سانتی‌متر مناسب است. کوادکوپتر طراحی شده دارای طول، عرض و ارتفاع بترتیب ۱۲۰، ۱۲۰ و ۲۷.۰۶ سانتی‌متر، ارتفاع مرکز جرم ۱۲- سانتی‌متر و وزن بدون گیره نگهدارنده ۱.۹ کیلوگرم می‌باشد.

۴- پردازش تصویر در کوادکوپتر شامل دو بخش اصلی تشخیص آب و هوا و وضعیت جوی و تشخیص هادی محافظ خطوط انتقال نیرو می‌باشد.

الگوریتم پردازش تصویر مورد استفاده جهت تشخیص وضعیت جوی محیط اطراف کوادکوپتر با استفاده از مشخصه‌های هوا شناختی از جمله سایه، شدت مه آلودگی، وجود یا عدم وجود بازتاب و کنتراست تصویر جهت بازرسی خطوط توسط دوربین‌های فوقانی و تحتانی طراحی گردیده است.استفاده می‌گردد. نتایج شبیه‌سازی تشخیص وضعیت جوی نشان داد که الگوریتم طراحی شده قادر به دسته‌بندی تصاویر ورودی با دقت ۹۱ درصد از یکدیگر می‌باشد.

شناسایی هادی محافظ خطوط انتقال با توجه به اینکه هادی محافظ در بالاترین نقطه از دکل شبکه انتقال قرار دارد و موانعی که بر سر راه کوادکوپتر در حرکت بر روی این هادی قرار دارد، علایم

اخطاری جهت جلوگیری از برخورد وسایل نقلیه با خطوط که به صورت شی کروی قرمز رنگ هستند، صورت می‌گیرد.

۵- طراحی الگوریتم کنترل کوادکوپتر با استفاده از اطلاعاتی نظیر شتاب خطی و موقعیت و سرعت زاویه‌ای،اطلاعات حسگر GPS و اطلاعات موقعیتی دریافت شده از ادومتری تصویری با استفاده از روش تلفیق حسگرها و فیلتر کالمن صورت می‌گیرد.

در مدل‌سازی کوادکوپتر ابتدا نگرش زاویه‌ای ربات (زوایای اویلر) تعریف گردید و سپس با نوشتن روابط سینماتیک و دینامیک آن، مدل‌سازی کوادکوپتر به اتمام می‌رسد.

کنترل کننده کوادکوپتر به صورت کنترل کننده حالت محدود[1]، طراحی گردید. ابتدا وظایف و حالت-هایی که ممکن است برای کوادکوپتر به وجود آید بیان گردید و سپس الگوریتم کنترل کننده حالت محدود طراحی گردید و در محیط Matlab شبیه‌سازی گردید. جهت جایگذاری قطب‌های ناپایدار سیستم و پایدار کردن آن از الگوریتم رگولاتور درجه دو خطی[2] استفاده گردید و جهت رهگیری نقطه هدف، از الگوریتم PID استفاده شد. در نهایت از روش طراحی کنترل کننده تطبیقی LQR+PID استفاده گردید که می‌تواند به صورت زنده، پارامترهای فیدبک خود را متناسب با تغییرات ارتفاع مرکز جرم کوادکوپتر تغییر دهد.

نتایج شبیه سازی بدست امده نشان می‌دهد که الگوریتم کنترل طراحی شده، در مقابل نیروی اغتشاش مقاوم بوده، در مقابل تغییرات پارامتر دینامیکی خود حالت تطبیقی دارد و ورودی داده شده را با تاخیر حدود ۱.۵ ثانیه دنبال می‌کند. همچنین در حالت ماندگار، خطای بین ورودی و خروجی را به سمت صفر همگرا کرده است.

[1] Finite State Controller

[2] Linear Quadratic Regulator (LQR)

۴-۲ پیشنهادات

با توجه به کارهای صورت گرفته در این پروژه، می‌توان پیشنهادات را به صورت زیر عنوان کرد:

۱- در این پروژه فرض بر این بوده است که برای شارژ باتری کوادکوکوپتر، از پنل‌های خورشیدی استفاده شده است. راه دیگری که برای این منظور می‌توان عنوان کرد، طراحی سیستم الکترومغناطیسی شارژ کننده کوادکوپتر است. از آنجا که این ربات در مجاورت خطوط انتقال، عملیات پروازی و بازرسی را انجام می‌دهد، همیشه تحت میدان‌های الکترومغناطیسی قرار دارد. بنابراین می‌توان سیستمی را طراحی کرد که همانند شارژرهای بی سیم تلفن‌های همراه، باتری این ربات را به شکل بیسیم شارژ نماید.

۲- در این پروژه از پلتفرم ROS استفاده شده است. همانطور که در فصل ۲ اشاره شد، این پلتفرم، اجازه اتصال دو رایانه از طریق شبکه را می‌دهد. از طرف دیگر، درحال حاضر برای هادی‌های محافظ شبکه انتقال، از کابلی به اسم OPGW استفاده می‌شود که دارای مغزی هایی از جنس فیبر نوری می‌باشد. بنابراین، در آینده می‌توان از دو خصوصیت اشاره شده استفاده کرد و تمامی اطلاعات از جمله تصاویر و خروجی حسگرها را از طریق این بستر، به صورت زنده به سامانه‌ای مستقر در دیپاچینگ یا وزارت نیرو ارسال نمود. از این طریق می‌توان حجم محاسباتی کوادکوپتر را تا حد زیادی کاهش داد.

۳- از آنجا که درحال‌حاضر عمل بازرسی از خطوط انتقال، موضوع از پیش حل شده تلقی می‌شوند و دوربین‌های مادون قرمز و ماورا بفنش مخصوصی نیز برای انجام همین بازرسی‌ها وجود دارند، می‌توان با نصب یکی از دوربین‌های اشاره شده، نسبت به بازرسی از این خطوط اقدام کرد. برای اجرایی نمودن این کار نیز تنها با ایجاد تغییری کوچک در ساختار کوادکوپتر، می‌توان دوربین‌ها و تجهیزات جدیدی به آن اضافه کرد.

۴- در این پایان نامه، به دلیل وظایفی که به کوادکوپتر تعریف شده بود، نیاز به کنترل کننده غیر

خطی وجود نداشت، اما با تعریف وظایفی دیگر و توسعه عملکرد ربات و در صورت نیاز، با توجه به استفاده از ریز رایانه در کوادکوپتر، می‌توان کنترل کننده‌های غیر خطی و پیچیده تری را نیز برای افزایش کارایی و سرعت عملکرد ربات استفاده کرد.

پیوست‌ها

۱- کد نوشته شده برای تشخیص وضعیت جوی

در این بخش، کد نوشته شده به زبان متلب برای تشخیص وضعیت جوی تصویری، آورده شده است.

```
function weather_classifier_test_majid_merati_93023105
% Load training and test data
load('C:\Users\majid\Desktop\majid arshad\binaii mahin safabakhsh\proje darsi\proje
vije\weather_release\image.mat','testImages');
% Update file name relative to matlabroot
dataSetDir     = fullfile('C:\Users\majid\Desktop\majid arshad\binaii mahin
safabakhsh\proje darsi\proje vije\weather_release');
testImages     = fullfile(dataSetDir, testImages);
%trainingLabels = load('C:\images.mat', 'trainingLabels');
%trainingLabels = trainingLabels.trainingLabels;
%double cloudy_true;
%double cloudy_false;
%double sunny_true;
%double sunny_false;
%% Test the Classifier
cloudy_false = 0;
sunny_true = 0;
cloudy_true = 0;
sunny_false = 0;
exTestImage  = imread(testImages{5,2});
figure;
imshow(exTestImage);
%for d = 1:2%numel(digits)
d=1;
   numImages    = size(testImages,1);

   for i = 1:numImages
      im = imread(testImages{i,d});
      im = im2double(im);
      [label, score] = weather_predict(im);
      if label == 1
         cloudy_true = cloudy_true + 1;
      else
         sunny_false = sunny_false + 1;
      end
   end
d=2;
   for i = 1:numImages
```

```
        im = imread(testImages{i,d});
        im = im2double(im);
        [label, score] = weather_predict(im);
        if label == 1
            cloudy_false = cloudy_false + 1;
        else
            sunny_true = sunny_true + 1;
        end
    end
tf_table={sunny_true, cloudy_false; sunny_false,cloudy_true};
%% Results
%  Tabulate the classification results for each SVM classifier.
digits = {char('cloudy'),char('sunny')};
displayTable(tf_table)
    function displayTable(labels)
        colHeadings = arrayfun(@(x)sprintf('svm(%d)',x),1:2,'UniformOutput',false);
        format = repmat('%-9s',1,11);
        header = sprintf(format,'digit  |',colHeadings{:});
        fprintf('\n%s\n%s\n',header,repmat('-',size(header)));
        for idx = 1:numel(digits)
            fprintf('%-9s', [digits{idx} ' |']);
            fprintf('%-9d%-9d', labels{idx,1},labels{idx,2});
            fprintf('\n')
        end
    end
end
```

۲- کد نوشته شده برای تلفیق حسگرها

در این بخش، به ترتیب کدهای نوشته شده به زبان C++ برای تلفیق ۴ عدد حسگر IMU و خروجی آن با GPS و ادومتری تصویری، آورده شده است.

۱-۱ تلفیق ۴ حسگر IMU

```
#include <ros/ros.h>
#include <sensor_msgs/Imu.h>
#include <geometry_msgs/Vector3.h>
#include <geometry_msgs/Wrench.h>
#include <math.h>

geometry_msgs::Vector3 linear_accel1;
geometry_msgs::Vector3 linear_accel2;
geometry_msgs::Vector3 linear_accel3;
geometry_msgs::Vector3 linear_accel4;

//geometry_msgs::Vector3 Quad_accel;
sensor_msgs::Imu kalman_accel;

//geometry_msgs::Wrench Wrench_applied;
float m = 1.39;//3;
float alpha = 0.05236; //PI/60 or 3degrees

float Q[3] = {0,0,0};
float R[3] = {0,0,0};
float Pk[3] = {0,0,0};
float S[3] = {0,0,0};
float Gain[3] = {0,0,0};
float xk[3] = {0,0,0};
float yk[3] = {0,0,0};

void accelCallback1(sensor_msgs::Imu Imu) {

   linear_accel1.x = Imu.linear_acceleration.x;
   linear_accel1.y = Imu.linear_acceleration.y;
   linear_accel1.z = Imu.linear_acceleration.z;

   /*double ysqr = Imu.orientation.y * Imu.orientation.y;
   double t0 = -2.0f * (ysqr + Imu.orientation.z * Imu.orientation.z) + 1.0f;
   double t1 = +2.0f * (Imu.orientation.x * Imu.orientation.y - Imu.orientation.w *
Imu.orientation.z);
   double t2 = -2.0f * (Imu.orientation.x * Imu.orientation.z + Imu.orientation.w *
Imu.orientation.y);
```

```
    double t3 = +2.0f * (Imu.orientation.y * Imu.orientation.z - Imu.orientation.w *
Imu.orientation.x);
    double t4 = -2.0f * (Imu.orientation.x * Imu.orientation.x + ysqr) + 1.0f;

    t2 = t2 > 1.0f ? 1.0f : t2;
    t2 = t2 < -1.0f ? -1.0f : t2;

    pitch = std::asin(t2);
    roll =  std::atan2(t3, t4);
    yaw  =  std::atan2(t1, t0);*/

    //set kalman_accel parameters
    kalman_accel.header = Imu.header;
    kalman_accel.linear_acceleration_covariance = Imu.linear_acceleration_covariance;
    kalman_accel.orientation = Imu.orientation;
    kalman_accel.orientation_covariance = Imu.orientation_covariance;
    kalman_accel.angular_velocity = Imu.angular_velocity;
    kalman_accel.angular_velocity_covariance = Imu.angular_velocity_covariance;

}

void accelCallback2(sensor_msgs::Imu Imu2) {

    linear_accel2.x = Imu2.linear_acceleration.x;
    linear_accel2.y = Imu2.linear_acceleration.y;
    linear_accel2.z = Imu2.linear_acceleration.z;

}
void accelCallback3(sensor_msgs::Imu Imu3) {

    linear_accel3.x = Imu3.linear_acceleration.x;
    linear_accel3.y = Imu3.linear_acceleration.y;
    linear_accel3.z = Imu3.linear_acceleration.z;

}
void accelCallback4(sensor_msgs::Imu Imu4) {

    linear_accel4.x = Imu4.linear_acceleration.x;
    linear_accel4.y = Imu4.linear_acceleration.y;
    linear_accel4.z = Imu4.linear_acceleration.z;

}

int main(int argc, char** argv){

  ros::init(argc, argv, "imu_accel_publisher");

  ros::NodeHandle nd;

  ros::Publisher accel_pub = nd.advertise<sensor_msgs::Imu>("imu_data", 1000);
```

```
  ros::Subscriber Imu1_sub = nd.subscribe("imu_raw1", 1000, accelCallback1);
  ros::Subscriber Imu2_sub = nd.subscribe("imu_raw2", 1000, accelCallback2);
  ros::Subscriber Imu3_sub = nd.subscribe("imu_raw3", 1000, accelCallback3);
  ros::Subscriber Imu4_sub = nd.subscribe("imu_raw4", 1000, accelCallback4);

  ros::Rate loop_rate(500);

  while (ros::ok()){
     ros::spinOnce();

     kalman_accel.linear_acceleration.x = (linear_accel1.x + linear_accel2.x + linear_accel3.x +
linear_accel4.x)/4;
     kalman_accel.linear_acceleration.y = (linear_accel1.y + linear_accel2.y + linear_accel3.y +
linear_accel4.y)/4;
     kalman_accel.linear_acceleration.z = (linear_accel1.z + linear_accel2.z + linear_accel3.z +
linear_accel4.z)/4;

     accel_pub.publish(kalman_accel);
     loop_rate.sleep();
  }
  return 0;
};
```

۲-۱ تلفیق حسگر IMU با GPS و خروجی ادومتری تصویری

```
#include <ros/ros.h>
#include <sensor_msgs/Imu.h>
#include <geometry_msgs/Vector3.h>
#include "geometry_msgs/Pose.h"
#include "gazebo_msgs/ModelState.h"
#include "nav_msgs/Odometry.h"

#include <iostream>
#include <ctime>

class Timer
{
public:
   Timer() { clock_gettime(CLOCK_REALTIME, &beg_); }

   double elapsed() {
      clock_gettime(CLOCK_REALTIME, &end_);
      return end_.tv_sec - beg_.tv_sec +
         (end_.tv_nsec - beg_.tv_nsec) / 1000000000.;
   }

   void reset() { clock_gettime(CLOCK_REALTIME, &beg_); }

private:
   timespec beg_, end_;
};
```

```
float deltaT = 0;
geometry_msgs::Pose Robot_Pose;
geometry_msgs::Vector3 V;
geometry_msgs::Pose Quad_Odo;


void velCallback(sensor_msgs::Imu Imu) {

   V.x += Imu.linear_acceleration.x * deltaT;
   V.y -= Imu.linear_acceleration.y * deltaT;
   V.z -= Imu.linear_acceleration.z * deltaT;

   //geometry_msgs::Pose Robot_Pose;
   Robot_Pose.position.x += V.x * deltaT;
   Robot_Pose.position.y += V.y * deltaT;
   Robot_Pose.position.z += V.z * deltaT;

   Robot_Pose.orientation.x = Imu.orientation.x;
   Robot_Pose.orientation.y = Imu.orientation.y;
   Robot_Pose.orientation.z = Imu.orientation.z;
   Robot_Pose.orientation.w = Imu.orientation.w;
}

void statecallback(nav_msgs::Odometry odo) {
   Quad_Odo.position = odo.pose.pose.position;
   Quad_Odo.orientation = odo.pose.pose.orientation;
}


//typedef message_filters::sync_policies::ApproximateTime<sensor_msgs::Imu,
rosgraph_msgs::Clock> SyncPolicy;

int main(int argc, char** argv){

  ros::init(argc, argv, "accel_publisher");

  ros::NodeHandle node;

  //ros::Publisher velocity_pub = node.advertise<geometry_msgs::Vector3>("accel_vel", 1000);

  ros::Publisher kalman_vel_pub =
    node.advertise< geometry_msgs::Vector3>("kalman_velocity", 1000);


  ros::Publisher kalman_pose_pub =
    node.advertise< geometry_msgs::Pose>("kalman_pose", 1000);

  ros::Subscriber Imu_sub = node.subscribe("imu_data", 1000, velCallback);
  ros::Subscriber pose_sub = node.subscribe("/Quad_odom", 1000, statecallback);
```

```
ros::Rate loop_rate(800);

geometry_msgs::Pose Quad_Odo_prev;
geometry_msgs::Vector3 Quad_Odo_Vel;

//geometry_msgs::Vector3 kalman_velocity;
//geometry_msgs::Pose kalman_Pose;


//define and initialize kalman variables
float Q[3] = {0,0,0};
float R[3] = {0,0,0};
float Pk[3] = {0,0,0};
float S[3] = {0,0,0};
float Gain[3] = {0,0,0};
float yk[3] = {0,0,0};

float Q2[3] = {0,0,0};
float R2[3] = {0,0,0};
float Pk2[3] = {0,0,0};
float S2[3] = {0,0,0};
float Gain2[3] = {0,0,0};
float yk2[3] = {0,0,0};


//initialize
Quad_Odo_prev.position.x = 0;
Quad_Odo_prev.position.y = 0;
Quad_Odo_prev.position.z = 0;
Quad_Odo_prev.orientation.x = 0;
Quad_Odo_prev.orientation.y = 0;
Quad_Odo_prev.orientation.z = 0;
Quad_Odo_prev.orientation.w = 0;

Timer tmr;
tmr.reset();
while (ros::ok()){
   deltaT = tmr.elapsed();
   tmr.reset();
   ros::spinOnce();

   Quad_Odo_Vel.x = (Quad_Odo.position.x - Quad_Odo_prev.position.x)/deltaT;
   Quad_Odo_Vel.y = (Quad_Odo.position.y - Quad_Odo_prev.position.y)/deltaT;
   Quad_Odo_Vel.z = (Quad_Odo.position.z - Quad_Odo_prev.position.z)/deltaT;
   Quad_Odo_prev = Quad_Odo;

   //velocity_pub.publish(V);

   Q[0] = 10; Q[1] = 10; Q[2] = 10; // Processing noise (accelerometer velocity)
   R[0] = 1; R[1] = 1; R[2] = 1;// Measuring noise (gazebo odometry)

   //state estimate
```

```
Pk[0] = Pk[0] + Q[0];
Pk[1] = Pk[1] + Q[1];
Pk[2] = Pk[2] + Q[2];

S[0] = Pk[0] + R[0];
S[1] = Pk[1] + R[1];
S[2] = Pk[2] + R[2];

//calculating inverse of Pk+R
S[0] = 1 / S[0];
S[1] = 1 / S[1];
S[2] = 1 / S[2];

Gain[0] = Pk[0] * S[0];
Gain[1] = Pk[1] * S[1];
Gain[2] = Pk[2] * S[2];

yk[0] = Quad_Odo_Vel.x - V.x;
yk[1] = Quad_Odo_Vel.y - V.y;
yk[2] = Quad_Odo_Vel.z - V.z;

V.x = V.x + (Gain[0] * yk[0]);
V.y = V.y + (Gain[1] * yk[1]);
V.z = V.z + (Gain[2] * yk[2]);

Pk[0] = (1 - Gain[0]) * Pk[0];
Pk[1] = (1 - Gain[1]) * Pk[1];
Pk[2] = (1 - Gain[2]) * Pk[2];

kalman_vel_pub.publish(V);

Q2[0] = 10; Q2[1] = 10; Q2[2] = 10; // Processing noise (accelerometer velocity)
R2[0] = 1; R2[1] = 1; R2[2] = 1;// Measuring noise (gazebo odometry)

//state estimate
Pk2[0] = Pk2[0] + Q2[0];
Pk2[1] = Pk2[1] + Q2[1];
Pk2[2] = Pk2[2] + Q2[2];

S2[0] = Pk2[0] + R2[0];
S2[1] = Pk2[1] + R2[1];
S2[2] = Pk2[2] + R2[2];

//calculating inverse of Pk+R
S2[0] = 1 / S2[0];
S2[1] = 1 / S2[1];
S2[2] = 1 / S2[2];

Gain2[0] = Pk2[0] * S2[0];
Gain2[1] = Pk2[1] * S2[1];
Gain2[2] = Pk2[2] * S2[2];
```

```
    yk2[0] = Quad_Odo.position.x - Robot_Pose.position.x;
    yk2[1] = Quad_Odo.position.y - Robot_Pose.position.y;
    yk2[2] = Quad_Odo.position.z - Robot_Pose.position.z;

    Robot_Pose.position.x = Robot_Pose.position.x + (Gain[0] * yk[0]);
    Robot_Pose.position.y = Robot_Pose.position.y + (Gain[1] * yk[1]);
    Robot_Pose.position.z = Robot_Pose.position.z + (Gain[2] * yk[2]);

    Pk[0] = (1 - Gain[0]) * Pk[0];
    Pk[1] = (1 - Gain[1]) * Pk[1];
    Pk[2] = (1 - Gain[2]) * Pk[2];

    //kalman_pose_pub.publish(Robot_Pose);
    kalman_pose_pub.publish(Quad_Odo);

    loop_rate.sleep();
  }

  return 0;
};
```

۳- کد نوشته شده برای محاسبه جدول جستجو

```
syms fi theta sai x y z xdot ydot zdot fidot thetadot saidot
solx soly solz solfi soltheta solsai m k1 k2 l jr Ix Iy Iz u1 u2
u3 u4;
s=tf('s');

alpha = pi/60;

data_num = 30;
for Yaw = 1:360
    for numer = 1:data_num
        dist = 0.6/data_num*numer-0.3;
        for counter = 1:data_num
            I_xx = 2/data_num*counter+0.45;
            solx = u1/m*cos(alpha)*(cos(fi)*sin(theta)*cos(sai)
+ sin(fi)*sin(sai)) -
u2/m*cos(sai)*sin(fi)*sin(theta)*sin(alpha)*(1-sqrt(2)/2) +
u3/m*cos(sai)*cos(theta)*sin(alpha)*(1-sqrt(2)/2) - k1*xdot/m;
            soly = u1/m*cos(alpha)*(cos(fi)*sin(theta)*sin(sai)
- sin(fi)*cos(sai)) -
u2/m*sin(alpha)*(cos(theta)*cos(sai)+sin(fi)*sin(sai)*sin(theta)
)*(1-sqrt(2)/2) + u3/m*cos(theta)*sin(sai)*sin(alpha)*(1-
sqrt(2)/2) - k1*ydot/m;
            solz = 1/m*(cos(fi)*cos(theta)*cos(alpha))*u1 -
u2/m*(1-sqrt(2)/2)*cos(theta)*sin(fi)*sin(alpha) - u3/m*(1-
sqrt(2)/2)*sin(theta)*sin(alpha) - 9.81 - k1*zdot/m;
```

```
            solfi = thetadot*saidot*(Iy-Iz)/Ix +
2*l*cos(alpha)/Ix*u2 - k2*l/Ix*fidot + dist/Ix*9.81*(m-
0.8)*sin(fi);
            soltheta = fidot*saidot*(Iz-Ix)/Iy +
2*l*cos(alpha)/Iy*u3 - k2*l/Iy*thetadot + dist/Iy*9.81*(m-
0.8)*sin(theta);
            solsai = fidot*thetadot*(Ix-Iy)/Iz +
1*cos(alpha)/Iz*u4 - k2/Iz*saidot;


jx=jacobian([solx,soly,solz,solfi,soltheta,solsai],[x y z xdot
ydot zdot fi theta sai fidot thetadot saidot]);

ju=jacobian([solx,soly,solz,solfi,soltheta,solsai],[u1 u2 u3
u4]);


            Ja = subs(jx,[m k1 k2 l jr Ix Iy Iz fi theta sai
fidot thetadot saidot u1 u2 u3],[3 0.01 0.012 0.5 1 I_xx 1.25
2.2 0 0 (Yaw-180)*pi/180 0 0 0 3*9.81 0 0]);
            Jb = subs(ju,[m k1 k2 l jr Ix Iy Iz fi theta sai
fidot thetadot saidot u1 u2 u3],[3 0.01 0.012 0.5 1 I_xx 1.25
2.2 0 0 (Yaw-180)*pi/180 0 0 0 3*9.81 0 0]);

            Ja=double(Ja);
            Jb=double(Jb);

            A(1,:) = [0 0 0 1 0 0 0 0 0 0 0 0];     A(2,:) = [0
0 0 0 1 0 0 0 0 0 0 0];
            A(3,:) = [0 0 0 0 0 1 0 0 0 0 0 0];     A(4,:) =
Ja(1,:);
            A(5,:) = Ja(2,:);                       A(6,:) =
Ja(3,:);
            A(7,:) = [0 0 0 0 0 0 0 0 0 1 0 0];     A(8,:) = [0
0 0 0 0 0 0 0 0 0 1 0];
            A(9,:) = [0 0 0 0 0 0 0 0 0 0 0 1];     A(10,:) =
Ja(4,:);
            A(11,:) = Ja(5,:);                      A(12,:) =
Ja(6,:);

            B(1,:) = [0 0 0 0];     B(2,:) = [0 0 0 0];
            B(3,:) = [0 0 0 0];     B(4,:) = Jb(1,:);
            B(5,:) = Jb(2,:);       B(6,:) = Jb(3,:);
            B(7,:) = [0 0 0 0];     B(8,:) = [0 0 0 0];
            B(9,:) = [0 0 0 0];     B(10,:) = Jb(4,:);
            B(11,:) = Jb(5,:);      B(12,:) = Jb(6,:);


            C = [1 0 0 0 0 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0 0 0 0
0; 0 0 1 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 1 0 0 0 0 0; 0 0 0 0 0 0
0 1 0 0 0 0; 0 0 0 0 0 0 0 0 1 0 0 0];
```

```
            D = [0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0
0];
            % C = [1 0 0 0 0 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0 0 0
0 0; 0 0 1 0 0 0 0 0 0 0 0 0; 0 0 0 0 0 0 0 0 1 0 0 0];
            % D = [0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];
            C2 = eye(12);
            D2 = [0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0
0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0;0 0 0 0];

            Q = [12 0 0 0 0 0 0 0 0 0 0 0;0 12 0 0 0 0 0 0 0 0 0
0;0 0 5 0 0 0 0 0 0 0 0 0;0 0 0 12 0 0 0 0 0 0 0 0;0 0 0 0 12 0
0 0 0 0 0 0;0 0 0 0 0 10 0 0 0 0 0 0;0 0 0 0 0 0 10 0 0 0 0 0;
                0 0 0 0 0 0 0 10 0 0 0 0;0 0 0 0 0 0 0 0 10 0 0
0;0 0 0 0 0 0 0 0 0 10 0 0; 0 0 0 0 0 0 0 0 0 0 10 0;0 0 0 0 0 0
0 0 0 0 0 10]; %10.*eye(12);
            R = [0.1 0 0 0; 0 0.2 0 0; 0 0 0.2 0; 0 0 0 0.1];

            %agar az simulink estefade mikonim(shabihsazie
dorost)
            %W =eye(12);%/1000
            %dar gheire in surat
            G = eye(12);
            H = zeros(6,12);

            W = eye(12);%/1000

            V=eye(6);%/10000

            almostzero = (abs(A)<1/1e6);
            A(almostzero) = 0;

            sys=ss(A,B,C,D);
            almostzero = (abs(sys.a)<1/1e6);
            sys.a(almostzero) = 0;

            almostzero = (abs(sys.b)<1/1e6);
            sys.b(almostzero) = 0;

            almostzero = (abs(sys.c)<1/1e6);
            sys.c(almostzero) = 0;

            almostzero = (abs(sys.d)<1/1e6);
            sys.d(almostzero) = 0;
            % [vlanda,landa]=eig(A);
            %% Pole place function (feedback design)
            [K(:,:,numer,Yaw,counter),~,~]=lqr(A,B,Q,R);

%             if numer>1
%                 rateK(:,:,numer-1,Yaw) = (K(:,:,numer,Yaw) -
K(:,:,numer - 1,Yaw))/(0.6/data_num);
%             end
        end
    end
% end
```

منابع و مراجع

[1] A.S. Mendes “Vision-based automatic landing of a quadrotor UAV on a floating platform”, Master of Science Thesis at Delft University of Technology, march 7, 2012

[2] Torkel Danielsson “Line inspection robot”, Master’s degree project of Uppsala University School of Engineering, April 2006

[3] Pedro B. Castellucci, Luiz C. Lucca, Marcelo, Sant’Anna, Gustavo Traballe, “Pole and crossarm identification in distribution power line images”, 2013 IEEE Latin American Robotics Symposium.

[4] Raul Mur-Artal*, J. M. M. Montiel, ´ *Member, IEEE,* and Juan D. Tardos, ´ *Member, IEEE.* “ORB-SLAM: a Versatile and Accurate Monocular SLAM System”, IEEE TRANSACTIONS ON ROBOTICS, 2015

[5] R. Chatila, J. P. Laumond, Position referencing and consistent world modeling for mobile robots, *Proceeding of IEEE International Conference on Robotics and Automation,* Silver Spring, USA: IEEE, pp. 138-145, 1985

[6] C. G. Harris, J. Pike, 3D positional integration from image sequences, *Image and Vision Computing,* Vol. 6, No. 2, pp. 87-90, 1988

[7] R. Smith, M. Self, P. Cheeseman, *Estimating Uncertain Spatial Relationships in Robotics*, pp. 167-193, New York: Springer, 1990

[8] A. Doucet, N. De Freitas, K. Murphy, S. Russell, Rao-Blackwellised particle filtering for dynamic Bayesian networks, *Proceeding of the Sixteenth Conference on Uncertainty in Artificial Intelligence,* San Francisco, USA: Morgan Kaufmann Publishers,pp. 176-183, 2000

[9] R. Martinez-Cantin, J. Castellanos, Unscented SLAM for largescale outdoor environments, *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems,* Edmonton, Canada: IEEE, pp. 3427-3432, 2005

[10] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, FastSLAM: A factored solution to the simultaneous localization and mapping problem, *Proceeding of the AAAI National Conference on Artificial Intelligence,* Edmonton, Canada: AAAI, pp. 593-598, 2002

[11] C. Kim, R. Sakthivel, W. K. Chung, Unscented FastSLAM: a robust and efficient

solution to the SLAM problem, *IEEE Transactions on Robotics,* Vol. 24, No. 4, pp. 808-820, 2008

[12] Z. Kurt-Yavuz, S. Yavuz, A comparison of EKF, UKF, FastSLAM2. 0, and UKF-based FastSLAM algorithms, *Proceeding of 16th International Conference on Intelligent Engineering Systems (INES),* Lisbon, Portugal: IEEE, pp. 37-43, 2012

[13] Erik Cuevas, Daniel Zaldivar, Raul Rojas, "Kalman filter for vision tracking", technical report, 10th August 2005

[14] Cewu Lu, Di Lin, Jiaya Jia, Chi-Keung Tang, "Two-Class Weather Classification", The Hong Kong University of Science and Technology, In *CVPR* 2014

[15] M. Roser and F. Moosmann. Classification of weather situations on single color images. In *IEEE Intelligent Vehicles Symposium*, 2008

[16] X. Yan, Y. Luo, and X. Zheng. Weather recognition based on images captured by vision system in vehicle. In *CVPR*, 2009

[17] C. Lu, J. Shi, and J. Jia. Scale adaptive dictionary learning. *IEEE Transactions on Image Processing (TIP)*, 2013

[18] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010

[19] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009

[۲۰] صافی، مصطفی.،"طراحی الگوریتم هدایت و کنترل کوادکوپتر,"، پایان‌نامه کارشناسی ارشد، دانشکده هوافضا، دانشگاه صنعتی امیرکبیر، ص ۳۲-۳۳، ۱۳۹۲

[۲۱] منافی، نیما.، " مدل‌سازی شش درجه آزادی کوادکوپتر با روتورهای دارای قاب کنترل شونده "، پایان‌نامه کارشناسی ارشد، دانشکده هوافضا، دانشگاه صنعتی امیرکبیر، ص ۱-۳۱، ۱۳۹۱

[22] Goldin, J.G., "Perching Using a Quadrotor with Onboard sensing", Master of science, In Electrical Engineering, UTAH State University, PP. 8, 2011

[23] Cetinsoy, E., Dikyar,S., Hancer, C., Oner, K.T., Sirimoglu, E., Unel, M., Aksit, M.F.," Design and construction of a novel quad tilt-wing UAV", Elsevier Journal, Vol. 22, No. 6, PP. 723-745, 2012

[24] Shayegan Omidshafiei, “Reinforcement Learning-based Quadcopter Control”, December 11, 2013

[25] Lucas M. Argentim, Willian C. Rezende, Paulo E. Santos, Renato A. Aguiar, “PID,LQR and LQR-PID on a Quadcopter Platform

[26] Raziyeh Babaei1, Amir Farhad Ehyaei “Robust Backstepping Control of a Quadrotor UAV using Extended Kalman Bucy Filter” ,IJMEC, PP. 2276-2291, Jul. 2015

[27] A. Swarup and Sudhir, “Comparison of Quadrotor Performance Using Backstepping and Sliding Mode Control”, International Conference on Circuits, Systems and Control, India, 2014

[28] P. Grossmann. Multisensor data fusion. The GEC journal of Technology, 15:27{37, 1998

[29] L.Wald. A european proposal for terms of reference in data fusion. International Archives of Photogrammetry and Remote Sensing, XXXII, Part 7:651{654, 1998

[30] B. V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. Proceedings of the IEEE, 85:24 1997

[31] E.Waltz and J. Llinas. Multisensor Data Fusion. Artech House, Norwood, Massachusetts, 1990

[32] D. L. Hall. Mathematical Techniques in Multi-Sensor Data Fusion. Artech House, Norwood, Massachusetts, 1992

[33] B. V. Dasarathy. Information fusion - what, where, why, when, and how? Information Fusion, 2(2):75{76, 2001

[34] Wilfried Elmenreich, An Introduction to Sensor Fusion, Research Report 47/2001, 2002

[35] Pablo Fernandez Alcantarilla, Adrian Bartoli, Andrew J. Davison, “KAZE Features”,ECCV 2012 Part VI LNCS 7577, pp. 214-227, 2012

[36] Claus B. Madsen, Thomas B. Moeslund, Amit Pal, and Shankkar Balasubramanian, “Shadow Detection in Dynamic Scenes Using Dense Stereo Information and an Outdoor Illumination Model”, Computer Vision and Media Technology Lab, Aalborg University, Aalborg, Denmark, 2010

[37] Roskam, J., “Airplane Flight Dynamics And Automaticic Flight Controls”, Vol. 4, The university of Kansas, PP. 195-197, 1979

Abstract

Today, the use of robots that have the ability to operate in the areas with harsh environmental conditions has attracted the attention of industrialists in various industries including electricity power industry. In this industry, for inspecting or repairing high-voltage transmission lines, the manpower is generally used. But due to dangerous conditions in high voltage lines such as harsh work environment and height from ground level, the electricity industry is faced manpower shortage in this sector. The aim of this project is to design and control a flying robot to mount on the high-voltage power lines and move on them for inspecting the damages and fractures of lines elements by image processing. In this project, the designed flying robot is assumed to be controlled by the operator from the ground, remotely. The flying robot flies from the ground and mounts on the guard cable (Shield wire) of transmission lines. Then, it moves on the line and inspects the damages and fractures of lines elements and report online to the operating system in the line control center on the ground. When the robot reaches an obstacle or the towers, it mounts off and passes the obstacle and remounts again on the line and continues tracking. The inspection on the robot uses infrared cameras, ultraviolet camera or conventional camera.

Solidworks environment was conducted to design the robot and its clipper according to the defined physical properties and dynamics of them. The adaptive LQR + PID alguritm was used to design the controller. The simulation results showed that the performance of the designed algorithm is perfectly reliable.

In machine vision and image processing, to automaticly stop the quadrocopters operation in adverse weather conditions, the weather detection algorithm is design and applied. The test results in Matlab showed 91% precision in detecting the weather condition. This is the best results according to research conducted in this field.

Keywords: flying robot, Quadrocopters, semi-autonomous, electricity power transmission lines, image processing

Shahrood University of Technology

Faculty of Mechanical and Mechatronics Engineering

M.Sc. Thesis in Mechatronics Engineering

Design and control of flying robot for inspecting high-voltage lines by image processing

By: Majid Merati

Supervisor(s):

Dr Alireza Ahmadifard

Dr Saeed Shiry Ghidary

Advisor

Dr Mahdi Bamdad

April 2017