کنترل موقعیت مبتنی بر بینایی کوادکوپتر AR.Drone 2.0 شناور با استفاده از منطق فازی

سهند عیوضی عدلی[۱]، دانشجوی کارشناسی ارشد؛ مریم شعاران[۲]، استادیار؛ سیدمحمدرضاسیدنورانی[۳]، استادیار

۱- دانشکده مهندسی فناوری‌های نوین - دانشگاه تبریز - تبریز - ایران – sahand.eyvazi94@ms.tabrizu.ac.ir
۲- دانشکده مهندسی فناوری‌های نوین - دانشگاه تبریز - تبریز - ایران - mshoaran@tabrizu.ac.ir
۳- دانشکده مهندس فناوری‌های نوین - دانشگاه تبریز - تبریز - ایران - smrs.noorani@tabrizu.ac.ir

چکیده: خودکارسازی شناوری کوادکوپتر AR.Drone 2.0 که موضوعی مهم و پیش‌نیاز سایر خودکارسازی‌ها است هدف این مقاله می‌باشد. در این مقاله الگوریتم جدیدی به نام GSPnP برای تخمین موقعیت ربات پرنده با استفاده از تک دوربین پیشنهاد می‌گردد. همچنین یک کنترلر فازی بهینه موسوم به TGM برای پایدارسازی شناوری کوادکوپتر طراحی و پارامترهای بهینه کنترلر فازی تعیین می‌شوند. موقعیت فعلی کوادکوپتر نسبت به مارکر کتابخانه ArUco با پردازش تصاویر دوربین زیرین ربات توسط الگوریتم پیشنهادی GSPnP محاسبه و به کنترلر ارسال می‌شود. خروجی کنترلر بر اساس درایور ربات متعلق به سیستم عامل رباتیک (ROS) محاسبه شده و به ربات شبیه‌سازی شده در محیط شبیه‌سازی Gazebo ارسال می‌شود. نتایج به‌دست‌آمده نشان‌دهنده عملکرد دقیق‌تر و مطلوب‌تر روش GSPnP و روش کنترل فازی TGM با خطای کمتر از ۳۰، ۴۰ و ۲۰ میلی‌متر در کنترل طول، عرض و ارتفاع نسبت به سایر روش‌ها، در کنترل موقعیت ربات است.

واژه‌های کلیدی: کوادکوپتر AR.Drone 2.0، شناوری خودکار، تخمین موقعیت PnP، کنترلر فازی، شبیه‌ساز Gazebo، سیستم عامل رباتیک

Position Based Visual Hovering Control of the AR.Drone 2.0 Quadcopter Using Fuzzy Logic

S. E. Adli[1], M.S. Student; M. Shoaran[2], Assistant Professor; S. M. S. Noorani[3], Assistant Professor

1- Faculty of Engineering-Emerging Technologies, University of Tabriz, Tabriz, Iran, Email: sahand.eyvazi94@ms.tabrizu.ac.ir
2- Faculty of Engineering-Emerging Technologies, University of Tabriz, Tabriz, Iran, Email: mshoaran@tabrizu.ac.ir
3- Faculty of Engineering-Emerging Technologies, University of Tabriz, Tabriz, Iran, Email: smrs.noorani@tabrizu.ac.ir

Abstract: Autonomous hovering of AR.Drone 2.0 quadcopter, which is an important subject and prerequisite for other autonomous UAV applications, is the goal of this paper. We propose a new method, called GSPnP, for pose estimation using only the bottom camera of the robot. Moreover, an optimal fuzzy controller, called TGM, is designed in order to stabilize the quadcopter hovering. Then, the optimal parameter values for the controller are obtained.The current position of the robot, relative to the ArUco library marker, is computed using our proposed GSPnP algorithm and the images of the bottom camera. The current position is sent to the controller and the output is computed based on the ROS AR.Drone 2.0 driver and is sent to the robot simulated in the Gazebo world. The results indicate a more accurate and desirable performance of GSPnP method and TGM fuzzy controller in controlling the robot position compared with other methods with an error of less than 30, 40, and 20 millimeters in *x*, *y*, and *z* directions, respectively.

Keywords: Quadcopter AR.Drone 2.0, Autonomous hovering, PnP pose estimation, Fuzzy controller, Gazebo simulator, ROS

تاریخ ارسال مقاله: ۱۳۹۷/۰۶/۱۷
تاریخ اصلاح مقاله:
تاریخ پذیرش مقاله: ۱۳۹۷/۰۷/۲۶
نام نویسنده مسئول: مریم شعاران
نشانی نویسنده مسئول: ایران - تبریز - بلوار ۲۹ بهمن - دانشگاه تبریز - دانشکده مهندسی فناوری‌های نوین

۱ -مقدمه

کوادکوپتر[۱] یا کوادروتور[۲] یک بالگرد چند ملخه عمودپرواز است. کوادکوپترها امروزه در بسیاری از زمینه‌ها از جمله نظامی، تحقیقاتی، کنترل و نظارت بر ترافیک، تصویربرداری، امداد و نجات و حمل و نقل به مناطق مرتفع و صعب العبور کاربرد دارند. ازاین‌رو شرکت‌هایی همچون DJI [۱] و Parrot [۲] شروع به ساخت کوادکوپترها به‌صورت تجاری کرده‌اند. در سال ۲۰۱۲ شرکت فرانسوی Parrot کوادکوپتر AR.Drone 2.0 [۳] را معرفی کرد. این ربات به دلیل هزینه نسبتاً پایین و وجود درایور برای سیستم عامل رباتیک[۳] و نرم‌افزار LabVIEW[۴] [۴] یک ربات مناسب برای انجام کارهای تحقیقاتی است. AR.Drone 2.0 دارای دو دوربین در قسمت جلو و زیر با مشخصات جدول ۱ است.

جدول ۱: مشخصات دوربین‌های کوادکوپتر AR.Drone 2.0

دوربین جلویی	720p (1280×720), 30FPS, 92°wide-angle diagonal lens
دوربین زیرین	QVGA (320×240), 60FPS, 64° field of view

خودکارسازی شناوری کوادکوپتر از اهمیت بالایی برخوردار است. زیرا که بدون تضمین شناور ماندن ربات انجام کارهای دیگر نظیر گریز از موانع، دنبال کردن اشیا، شناوری و مکان‌یابی و نقشه‌برداری هم‌زمان[۵] ناممکن است. به همین دلیل در این مقاله هدف ما کنترل شناوری خودکار کوادکوپتر AR.Drone 2.0 نسبت به مارکر آماده کتابخانه ArUco [۵] تنها با استفاده از دوربین زیرین ربات است. از نوآوری‌های انجام‌گرفته در این مقاله می‌توان به الف) استفاده از فقط تک دوربین زیرین کوادکوپتر به‌عنوان سنسور تخمین موقعیت، ب) ارائه الگوریتم جدید پیشنهادی GSPnP برای تخمین موقعیت ربات و ج) طراحی یک کنترلر فازی موسوم به TGM برای پایدارسازی شناوری کوادکوپتر و تعیین پارامترهای بهینه کنترلر فازی اشاره نمود.

در این مقاله یک کنترلر خارجی بر اساس منطق فازی جهت کنترل موقعیت و شناوری ربات طراحی شده است. ورودی کنترلر شامل موقعیت مطلوب و فعلی ربات است. موقعیت فعلی کوادکوپتر نسبت به مارکر کتابخانه ArUco توسط پردازش تصاویر دوربین زیرین ربات با استفاده از کتابخانه OpenCV و توسط الگوریتم پیشنهادی جدید GSPnP محاسبه می‌شود. خروجی کنترلر براساس درایور ربات متعلق به سیستم عامل رباتیک، محاسبه شده و به ربات شبیه‌سازی‌شده در محیط شبیه‌سازی Gazebo ارسال می‌شود. جهت دستیابی به بهترین روش کنترلی فازی برای کنترل موقعیت ربات، انواع توابع عضویت شامل گاوسی و مثلثی به دو شکل متفاوت به همراه سه روش مختلف غیرفازی سازی ۱-مرکز مجموع ۲-میانگین وزن‌دار ۳-بزرگ‌ترین و کوچک‌ترین بیشینه اعمال شده‌اند. نتایج حاکی از عملکرد دقیق‌تر و مطلوب‌تر روش GSPnP و روش کنترل فازی TGM با خطای کمتر از ۳۰، ۴۰ و ۲۰ میلی‌متر در کنترل طول، عرض و ارتفاع نسبت به سایر روش‌ها، در کنترل موقعیت ربات است.

شایان ذکر است دلیل استفاده از کنترلر فازی توانایی آن در مقابله با سیستم‌های غیرخطی، مقاومت ذاتی آن در مقابل نویز و اغتشاشات و سادگی نسبی طراحی آن در مقایسه با کنترلرهای کلاسیک [۲۵] است. به‌عنوان مثال تعیین دقیق ضرایب یک کنترلر PID بسیار مهم است و باید سعی زیادی در بهینه‌سازی ضرایب کنترلر PID نمود.

در ادامه و در بخش ۲ کارهای تحقیقاتی مرتبط در این زمینه مورد نقد و بررسی قرار گرفته است. در بخش ۳ الگوریتم پیشنهادی شرح داده شده است. نتایج در بخش ۴ شرح داده شده است و جمع‌بندی مقاله در بخش ۵ آمده است.

۲ -مروری بر تحقیقات پیشین

تحقیقات متعددی در زمینه کنترل کوادروتور AR.Drone 2.0 انجام‌گرفته است که ما در اینجا تنها به تعدادی معدود اشاره خواهیم کرد. Micheal [۶] با معرفی یک درایور مبتنی بر LabVIEW با نام AR.Drone Labview Toolkit توانست با استفاده از یک ایستگاه زمینی با کوادکوپتر AR.Drone ارتباط برقرار کرده و آن را کنترل کند. با معرفی این درایور زمینه برای افزودن کاربردهای متفاوت مانند کنترل موقعیت، شناوری بر روی یک شی و جهت‌یابی بر پایه بینایی فراهم شد [۷]. SunYue [۸] برای کنترل موقعیت کوادکوپتر AR.Drone 2.0 دو کنترلر کلی و محلی را با استفاده از روش کنترلی PID طراحی کرد. ضرایب کنترلر PID با استفاده از تجربه به‌دست‌آمده بودند. کنترلر محلی با استفاده از کنترلر PID وظیفه کنترل کردن پارامترهای کنترلی مربوط به ارتفاع، زاویه انحراف، زاویه اوج‌گیری و زاویه پیچش را بر عهده داشت. درحالی‌که کنترلر کلی، موقعیت ربات را به‌وسیله کنترل پارامترهای x, y آن بر عهده داشت.

Prayitno، Indrawati و همکاران [۹-۱۱] برای دنبال کردن مسیر، یک کنترلر براساس منطق فازی طراحی کردند که فاصله و زاویه ربات نسبت به مسیر را به‌عنوان ورودی دریافت می‌کرد و دو پارامتر کنترلی زاویه انحراف و اوج‌گیری را به‌عنوان خروجی به ربات ارسال می‌کرد. در این حالت اگر پارامتر زاویه انحراف به‌درستی کنترل نشود، خطای موجود در این کنترل بر روی کنترل پارامتر زاویه اوج‌گیری تأثیرگذار بوده و ربات به مسیر اشتباه هدایت خواهد شد. در ضمن پارامتر زاویه پیچش و ارتفاع در طول پرواز ربات به‌هیچ‌وجه کنترل نمی‌شوند. یک سال بعد وی با ارتقاء کنترلر طراحی شده برای دنبال کردن مسیر توانست پارامتر ارتفاع را کنترل کند. در همان سال وی از یک روش دیگر برای کنترل کوادکوپتر استفاده کرد و این بار سه پارامتر کنترلی زاویه پیچش، زاویه اوج‌گیری و ارتفاع ربات را به‌وسیله کنترل فازی با تابع ورودی به شکل مثلث و غیرفازی سازی باینری کنترل کرد. نقطه‌ضعف غیرفازی سازی باینری نادیده گرفتن اعداد بین دو عدد متوالی باینری بود.

Tang [۱۲] از سه کنترل فازی برای هدایت ربات به یک هدف دارای طول و عرض مشخص استفاده کرد. این سه کنترلر به ترتیب دارای ورودی‌های $(x), (y), (z)$ (بیانگر موقعیت فعلی ربات) بودند و خروجی سه کنترلر شامل پارامترهای کنترلی مربوط به زاویه اوج‌گیری، زاویه پیچش و ارتفاع بود. پارامتر کنترلی زاویه انحراف در این کار کنترل نمی‌شود.

Boudjit [۱۳] از کتابخانه fuzzylite [۱۴] برای طراحی کنترلر منطق فازی با تابع عضویت گاوسی برای ورودی و خروجی استفاده کرد. تصاویر دوربین ربات به کتابخانه ar-track-alvar برای تشخیص مارکرهای موجود در محیط و پیدا کردن فاصله ربات از مارکر فرستاده می‌شوند. کوادکوپتر AR.Drone 2.0 با استفاده از بسته tum_ardrone متعلق به سیستم عامل رباتیک شبیه‌سازی‌شده است.

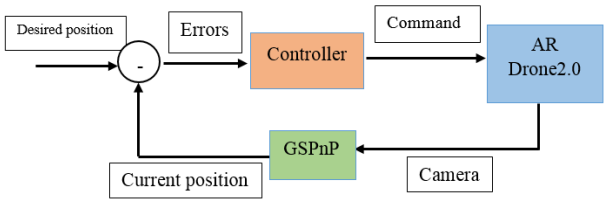
Tao [۱۵] از کنترلر PID برای کنترل کوادکوپتر AR.Drone 2.0 استفاده نمود. و از یک کنترلر فازی با توابع عضویت مثلثی برای ورودی و خروجی جهت بهبود ضرایب PID استفاده کرد. به اینصورت که در هر بار اجرا شدن حلقه کنترلی مقادیر $\Delta k_i, \Delta k_p, \Delta k_d$ توسط کنترلر فازی محاسبه شده و به کنترلر اصلی PID فرستاده می‌شوند تا ضرایب آن با این مقادیر جمع شده و تغییر یابند.

Chen-huan Pi [۱۶] یک کنترلر فازی را بر اساس فیدبکی که از پردازش تصاویر دوربین ربات توسط کتابخانه OpenCV [۱۷] دریافت می‌کرد، بر روی کوادکوپتر AR.Drone 2.0 با استفاده از ایستگاه زمینی و کتابخانه fuzzylite پیاده‌سازی کرد. ورودی و خروجی کنترلر طراحی شده به‌صورت تابع گاوسی بود. آن‌ها به این نتیجه رسیدند که کنترل ربات با کنترلر فازی از طریق ایستگاه خارجی عملکرد مناسبی نسبت به کنترل آن تنها از طریق کنترلر داخلی موجود در ربات ارائه می‌کند. ولی آن‌ها انواع مختلف توابع (مثلثی، مثلثی بدون قله و ...) را برای ورودی و خروجی در نظر نگرفتند. همچنین اشاره‌ای به روش غیرفازی سازی نشده است و به احتمال زیاد از روش پیش‌فرض کتابخانه fuzzylite استفاده شده است. کنترلر طراحی شده سه پارامتر طول، عرض و زاویه انحراف ربات را کنترل می‌کند درحالی‌که پارامتر ارتفاع بدون کنترل رها می‌شود. Atulya Shivam Shree و همکارانش [۲۱] برای تخمین هم‌زمان موقعیت و تشکیل نقشه محیط توسط یک کوادروتور از بینایی ماشین استفاده نمودند. برای این منظور آن‌ها اطلاعات حاصل از دوربین را با سنسورهای اینرسی توسط فیلتر کالمن ترکیب نمودند. M.Bergamasco و همکارش [۲۲] به شناسایی دینامیک یک کوادروتور پرداختند. B.T.M. Leong و همکارانش [۲۳] برای کنترل موقعیت یک کوادکوپتر شناور از یک کنترلر ساده PID استفاده نمودند. Mohd Ariffanan Mohd Basri در سال ۲۰۱۸ [۲۴] با ترکیب کنترلرهای backstepping و فازی برای دنبال کردن مسیر و پایدارسازی یک کوادکوپتر پرداخت. شایان ذکر است که کار ایشان فقط به شبیه‌سازی محدود می‌باشد و فرض بر این است که موقعیت فعلی کوادکوپتر مشخص می‌باشد.

۳ -الگوریتم پیشنهادی

برای شبیه‌سازی الگوریتم پیشنهادی در این مقاله از سیستم عامل رباتیک (ROS) و شبیه‌ساز ربات Gazebo استفاده شده است. سیستم عامل رباتیک یک میان‌افزار رباتیک است که توسط مجموعه‌ای از بسته‌ها، کتابخانه‌ها و ابزارها پیچیدگی‌های موجود بر سر راه کنترل و طراحی انواع ربات‌ها را مدیریت می‌کند. کوادکوپتر AR.Drone 2.0 به‌وسیله بسته tum_simulator [۱۸] متعلق به سیستم عامل رباتیک در محیط شبیه‌سازی Gazebo شبیه‌سازی‌شده است. جهت کنترل AR.Drone 2.0 ما از درایور آن در سیستم عامل رباتیک به نام ardrone-autonomy [۱۹] استفاده می‌کنیم. مارکرهای استفاده شده در اینجا مربوط به کتابخانه Aruco است. این کتابخانه مارکرهای مختلفی را در اندازه‌ها و شکل‌های متفاوت معرفی می‌کند. برای پردازش تصاویر نیز از کتابخانه OpenCV استفاده شده است.

برای ایجاد شناوری خودکار کوادکوپتر AR.Drone 2.0 یک حلقه کنترلی بسته با استفاده از سیستم عامل رباتیک (ROS) ایجاد می‌کنیم (شکل ۱).



شکل ۱: کنترلر حلقه بسته برای شناوری خودکار ربات در محیط ROS

در شکل ۱ ابتدا تصاویر دوربین زیرین کوادکوپتر AR.Drone 2.0 شبیه‌سازی‌شده در محیط Gazebo به کتابخانه‌های OpenCV و ArUco فرستاده می‌شوند. تصاویر فرستاده شده پردازش شده و نقاط تشکیل‌دهنده مارکر استخراج می‌شوند. نقاط مارکر جهت محاسبه موقعیت فعلی ربات نسبت به هدف (مرکز دستگاه مختصات سراسری) به الگوریتم پیشنهادی GSPnP به‌عنوان ورودی داده می‌شوند. الگوریتم پیشنهادی GSPnP از جمله الگوریتم‌های تخمین موقعیت دوربین (معروف به مسئله پرسپکتیو n نقطه) است. پس از محاسبه موقعیت فعلی، اختلاف موقعیت فعلی از مطلوب (خطا) به بلوک کنترلر طراحی شده فرستاده می‌شود و بلوک کنترل براساس خطای ورودی دستورات هدایتی را جهت هدایت و انجام شناوری خودکار نسبت به هدف تعیین‌شده، محاسبه کرده و به ربات شبیه‌سازی‌شده ارسال می‌کند.

۳ ـ۱ الگوریتم پیشنهادی GSPnP

تخمین ۶ پارامتر مجهول انتقال و دوران دوربین کالیبره شده (پارامترهای داخلی دوربین معلوم) نسبت به چارچوب سراسری با فرض مشخص بودن n تا نقطه سه‌بعدی معلوم در چارچوب سراسری و نقاط تصویر شده دوبعدی آن‌ها در صفحه تصویر [۲۶] به مسئله پرسپکتیو n نقطه (PnP) مشهور است. الگوریتم پیشنهادی GSPnP یک روش

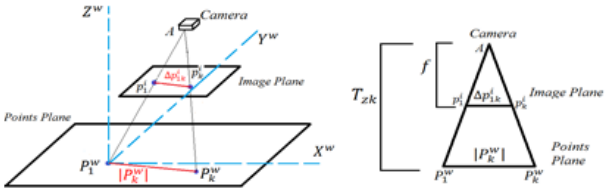
غیرتکرارشونده بسیار سریع و برپایه هندسه افکنش است. حال فرض کنیم n تا نقطه سه‌بعدی بر روی مارکر با مختصات زیر موجود هستند.

$$P_k^w = \left(X_k^w, Y_k^w, 0\right), k = 1, \ldots, n \quad (۱)$$

در رابطه ۱، k شمارنده تعداد نقاط، w نشان دهنده دستگاه مختصات سراسری و P نشان دهنده نقطه است. تصاویر نقاط ۳ بعدی در صفحه تصویر دارای مختصاتی به‌صورت رابطه ۲ هستند که در آن i نشان دهنده دستگاه مختصات پیکسلی (دستگاه مختصاتی با مرکزیت گوشه صفحه تصویر، واقع در صفحه تصویر) است.

$$p_k^i = (u_k^i, v_k^i), k = 1, \ldots, n \quad (۲)$$

در شکل ۲ شمای کلی نقاط ۳ بعدی در دستگاه مختصات سراسری و تصاویر ۲ بعدی آن‌ها در دستگاه مختصات تصویر به همراه دوربین نشان داده شده است. در این شکل فرض بر این است که صفحه تصویر و صفحه نقاط باهم موازی هستند.



شکل ۲: موقعیت صفحه نقاط (مارکر) و تصویر نسبت به دوربین (تصویر سمت چپ)، مثلث‌های به وجود آمده از نحوه تصویر شدن نقاط توسط دوربین به صفحه تصویر (سمت راست)

مرکز دستگاه مختصات سراسری را در هر نقطه‌ای از دستگاه مختصات سراسری می‌توان تعریف کرد. در شکل ۲ (چپ) مرکز دستگاه مختصات سراسری به‌دلخواه در نقطه P_1^w مشخص شده است. A محل دوربین، $|P_k^w|$ (رابطه۳) فاصله نقطه kام از مرکز دستگاه مختصات سراسری و Δp_{1k}^i (رابطه۴) فاصله پیکسلی تصویر نقطه kام از تصویر مرکز دستگاه مختصات سراسری است و p_1^i تصویر نقطه P_1^w در صفحه تصویر است. در شکل ۲ (راست)، f فاصله کانونی و T_{zk} ارتفاع دوربین در دستگاه مختصات سراسری است. حال رابطه تالس را برای مثلث‌های شکل ۲ برای محاسبه ارتفاع دوربین در دستگاه مختصات سراسری به‌مانند رابطه ۵ می‌نویسیم.

$$\left|P_k^w\right| = \sqrt{X_k^{w\,2} + Y_k^{w\,2}}, k = 2, \ldots, n \quad (۳)$$

$$\Delta p_{1k}^i = p_1^i - p_k^i = \sqrt{\left(u_1^i - u_k^i\right)^2 + \left(v_1^i - v_k^i\right)^2}, k = 2, \ldots, n \quad (۴)$$

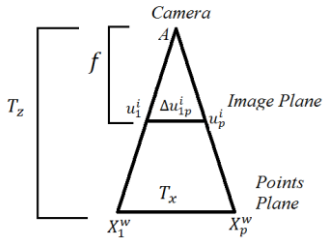
$$T_{zk} = f \times \frac{\left|P_k^w\right|}{\Delta p_{1k}^i}, k = 2, \ldots, n \quad (۵)$$

در روابط ۳ الی ۵ شمارنده k از عدد ۲ شروع شده است زیرا که عدد ۱ برای مرکز دستگاه مختصات رزرو شده است. حال مقدار T_z را با میانگین‌گیری T_{zk} ها محاسبه می‌کنیم.

جهت محاسبه طول و عرض دوربین در دستگاه مختصات سراسری لازم است فاصله تصویر مرکز دستگاه مختصات سراسری (p_1^i) را از تصویر مرکز دوربین $(p_p^i = (u_p^i, v_p^i))$ در دو راستای طول و عرض برحسب پیکسل در دستگاه مختصات پیکسلی محاسبه کنیم. سپس مثلث شکل ۲ را به دو راستای طول و عرض تصویر می‌کنیم تا دو عدد مثلث ایجاد شود. با نوشتن رابطه تالس برای مثلث‌های ایجاد شده می‌توان طول و عرض دوربین را در دستگاه مختصات سراسری محاسبه کرد. در شکل ۳ مثلث شکل ۲ در راستای طول تصویر شده است.

نقطه مجازی سه‌بعدی متناظر نقطه p_p^i (مرکز دوربین در صفحه تصویر) در دستگاه مختصات سراسری به‌صورت $P_p^w = (X_p^w, Y_p^w, 0)$ تعریف می‌شود. طول دوربین در دستگاه مختصات سراسری (T_x) از طریق روابط ۶ و ۷ محاسبه می‌شود.

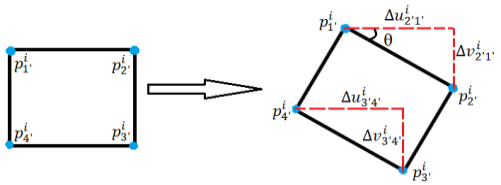
$$\Delta u_{1p}^i = \left(u_1^i - u_p^i\right) \quad (۶)$$



شکل ۳: مثلث‌ها برای محاسبه طول دوربین در دستگاه سراسری (T_{xk})

$$T_X = \frac{T_z \times \Delta u_{1p}^i}{f}, k = 2, \ldots, n \quad (۷)$$

مشابه محاسبه طول دوربین در دستگاه مختصات سراسری (T_x) همین روند را برای محاسبه عرض دوربین (T_y) در دستگاه یاد شده انجام خواهیم داد. برای محاسبه زاویه انحراف دوربین (چرخش حول محور ارتفاع) از این واقعیت استفاده می‌کنیم که نقاط موجود برروی مارکر (صفحه نقاط) تشکیل یک مربع می‌دهند و با چرخش دوربین حول محور ارتفاع، مربع حاصل از تصویر نقاط واقع برروی مارکر دچار چرخش می‌شود (شکل ۴).



شکل ۴: مربع تشکیل شده در صفحه تصویر قبل و بعد از دوران به‌اندازه زاویه θ حول محور ارتفاع

در شکل ۴، چهار عدد نقطه $p_{4'}^{i}, p_{3'}^{i}, p_{2'}^{i}, p_{1'}^{i}$ واقع در صفحه تصویر تشکیل یک مربع می‌دهند. همان‌طورکه قابل مشاهده است با دوران دوربین حول محور ارتفاع، مربع دچار دوران می‌شود. حال با استفاده از روابط ۸ الی ۱۳ می‌توان زاویه انحراف θ را محاسبه کرد. در روابط ۹ و ۱۱ پارامتر ns تعداد مربع‌های موجود در صفحه نقاط، h و j شمارنده تعداد مربع‌ها است.

$$\Delta u_{2'1'}^{i} = u_{2'}^{i} - u_{1'}^{i}, \Delta v_{2'1'}^{i} = v_{2'}^{i} - v_{1'}^{i} \quad (۸)$$

$$\theta_h = \tan^{-1} \frac{\Delta v_{2'1'}^{i}}{\Delta u_{2'1'}^{i}}, h = 1, ...ns \quad (۹)$$

$$\Delta u_{3'4'}^{i} = u_{3'}^{i} - u_{4'}^{i}, \Delta v_{3'4'}^{i} = v_{3'}^{i} - v_{4'}^{i} \quad (۱۰)$$

$$\theta_j = \tan^{-1} \frac{\Delta v_{3'4'}^{i}}{\Delta u_{3'4'}^{i}}, j = 1, ..., ns \quad (۱۱)$$

با محاسبه میانگین θ_h و θ_j مقدار θ مشخص می‌شود. حال مقادیر طول، عرض، ارتفاع و زاویه انحراف دوربین (T_z, T_y, T_x, θ) در دستگاه مختصات سراسری محاسبه شده‌اند. این مقادیر موقعیت فعلی کوادکوپتر در دستگاه مختصات سراسری را بیان می‌کنند.

۳ ۴ نحوه کنترل AR.Drone 2.0

درایور ardrone-autonomy سیستم عامل رباتیک کوادکوپتر AR.Drone 2.0 توسط ۶ تا پارامتر کنترل می‌شود. این ۶ پارامتر (Linear.x, Linear.y, Linear.z, Angular.x, Angular.y, Angular.z) می‌توانند مقادیری بین ۱/۰- و ۱/۰ را داشته باشند. پارامترهای شروع شونده با نام Linear حرکت خطی ربات در سه جهت اصلی طول، عرض و ارتفاع را کنترل می‌کنند و پارامترهای شروع شونده با نام Angular چرخش ربات حول محورهای اصلی را ممکن می‌سازند. ما در این مقاله تنها چهار عدد از این پارامترها را کنترل می‌کنیم. به این دلیل که دو پارامتر Angular.x و Angular.y برای خارج کردن کوادکوپتر از وضعیت شناوری خودکار (با دادن مقادیر غیر صفر) وقتی که پارامترهای کنترلی جدول (۲) مقادیر صفر را دارند، بکار می‌روند.

جدول ۲: پارامترهای کنترلی و نحوه عملکردشان

پارامتر کنترلی	علامت پارامتر	جهت حرکت کوادکوپتر
Linear.x	-	Move backward
Linear.x	+	Move forward
Linear.y	-	Move right
Linear.y	+	Move left
Linear.z	-	Move down
Linear.z	+	Move up
Angular.z	-	Turn right
Angular.z	+	Turn left

کوادکوپتر AR.Drone 2.0 خود یک کنترلر داخلی دارد که ما در اینجا آن را به‌صورت یک جعبه سیاه در نظر می‌گیریم که فقط اطلاعات مربوط به چهار پارامتر کنترلی یاد شده در جدول (۲) را دریافت می‌کند و نسبت به مقادیر داده شده حرکت می‌کند. درنتیجه منظور ما از طراحی کنترلر همان کنترلر خارجی است که دستورات را از طریق درایور آن به کنترلر داخلی کوادکوپتر ارسال می‌کند در جدول (۲) چهار پارامتر کنترلی و نحوه عملکردشان آورده شده است.

پیش‌تر در مورد موقعیت فعلی (خروجی روش GSPnP) و موقعیت مطلوب بحث کردیم. حال بردارهای مربوط به این دو موقعیت را در رابطه ۱۲ بیان می‌کنیم.

$$DesiredPosition \begin{vmatrix} x = 0 \\ y = 0 \\ z = Z_d \\ \theta = 0^\circ \end{vmatrix}, CurrentPosition \begin{vmatrix} x = T_x \\ y = T_y \\ z = T_z \\ \theta = \theta_c^\circ \end{vmatrix} \quad (۱۲)$$

در رابطه ۱۲، x, y, z مکان کوادکوپتر درمختصات سراسری و θ زاویه انحراف ربات (زاویه دوران حول محور z) در مختصات سراسری است. مرکز مختصات سراسری در نقطه‌ای دلخواه بر روی مارکر قرار دارد. مقادیر $T_x, T_y, T_z, \theta_c^\circ$ خروجی پردازش تصاویر دوربین زیرین ربات توسط الگوریتم پیشنهادی GSPnP هستند و موقعیت فعلی ربات را مشخص می‌کنند. جهت جلوگیری از اشتباه شدن زاویه انحراف فعلی (به‌دست‌آمده از الگوریتم GSPnP) و زاویه انحراف دوربین (θ) از این پس زاویه انحراف فعلی ربات را θ_c° می‌نامیم. موقعیت مطلوب را در هر نقطه‌ای از مختصات سراسری می‌توان تعریف کرد. ما در اینجا نقطه‌ای با مختصات $x = 0, y = 0, z = Z_d$ را به‌عنوان نقطه مطلوب، و برای زاویه انحراف ربات مقدار 0° را در نظرگرفته‌ایم. حال با استفاده از رابطه‌های ۱۳ و ۱۴ خطای موجود در مکان ربات را محاسبه می‌کنیم. این خطاها ورودی کنترلر خارجی هستند.

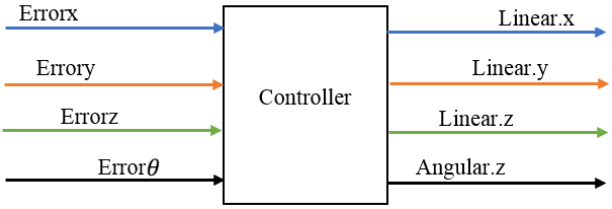
$$Error = CurrentPosition - DesiredPosition \quad (۱۳)$$

$$Error \begin{vmatrix} errorx = T_x - 0 \\ errory = T_y - 0 \\ errorz = T_z - Z_d \\ error\,\theta = \theta_c^\circ - 0^\circ \end{vmatrix} \quad (۱۴)$$

پس از محاسبه خطاها بایستی کنترلر خارجی ربات طراحی شود. ورودی این کنترلر خطاهای محاسبه شده در رابطه ۱۴ و خروجی آن اعدادی بین ۱/۰- و ۱/۰ خواهد بود که به چهار پارامتر کنترلی نام برده شده در جدول (۲) در قالب یک بردار با نام command فرستاده خواهند شد. در رابطه ۱۵ اجزای بردار command، و در شکل ۵ ورودی‌ها و خروجی‌های کنترلر خارجی نشان داده شده است.

$$Command = \begin{vmatrix} Linear.x \\ Linear.y \\ Linear.z \\ Angular.z \end{vmatrix} \quad (۱۵)$$

هدف ما کنترل شناوری خودکار ربات به‌وسیله روش‌های مختلف کنترل فازی است. لذا ابتدا یک کنترلر PID را براساس تجربه‌ای که از کنترل دستی ربات در فضای شبیه‌سازی داشتیم (ضرایب کنترلر PID به‌صورت تجربی به‌دست‌آمده است)، پیاده‌سازی می‌کنیم.



شکل ۵: ورودی‌ها و خروجی‌های کنترلر خارجی

پس از پیاده‌سازی کنترلر PID، کنترلری بر مبنای منطق فازی با استفاده از روش استنباط فازی Mamdani [۲۰] پیاده‌سازی گردید. برای این کار لازم است بر مبنای اطلاعاتی که از عملکرد کنترلر PID به‌دست‌آمده، قوانین فازی را طراحی کنیم. در حالت کلی قوانین به این صورت هستند که اگر خطا در هر جهتی بزرگ، متوسط و کوچک باشد حرکت ربات به ترتیب بزرگ، متوسط و کوچک در خلاف جهت خواهد بود. سپس توابع عضویت را برای ورودی و خروجی کنترلر استخراج کنیم. انواع مختلف توابع عضویت وجود دارد که ما در اینجا از دو نوع مثلثی و گاوسی به دو صورت مختلف استفاده خواهیم کرد. ما ورودی و خروجی (خطا و حرکت) کنترلر فازی را به هفت مجموعه (۱- بزرگ و منفی ۲- بزرگ و مثبت ۳-متوسط و منفی ۴- متوسط و مثبت ۵- کوچک و منفی ۶- کوچک و مثبت ۷- صفر و نزدیک صفر) تقسیم می‌کنیم. برخلاف بیشتر کارهای انجام شده [۹-۱۱] و [۱۳] که از پنج مجموعه استفاده شده است. با بالا رفتن تعداد مجموعه‌ها دقت عملکرد کنترلر فازی بالا می‌رود، اما تعداد بالای مجموعه‌ها می‌تواند باعث عملکرد کند کنترلر شود. رنج مجموعه‌ها و توابع عضویت ورودی براساس خطاهای به وجود آمده از اعمال کنترلر PID تنظیم شده‌اند. برای توابع عضویت خروجی، رنج خروجی براساس مقادیر محاسبه شده توسط کنترلر PID به ازای بیشترین خطای به وجود آمده در هر مجموعه ورودی محاسبه می‌شوند. به‌این‌ترتیب که حد بالا و پایین هر مجموعه ورودی را در کنترلر PID وارد کرده و خروجی را محاسبه می‌کنیم، خروجی متناظر حد بالا به‌عنوان حد بالای مجموعه خروجی و خروجی متناظر حد پایین به‌عنوان حد پایین مجموعه خروجی خواهد بود. پس از این محاسبات کنترلر فازی به وجود آمده را اعمال می‌کنیم و براساس عملکرد آن حدود مجموعه‌های ورودی و خروجی را تغییر می‌دهیم.

در ادامه ابتدا قوانین فازی را بیان خواهیم کرد (جدول ۳) و سپس نمودارهای توابع عضویت را بر مبنای اطلاعات عملکرد کنترلر PID رسم می‌کنیم.

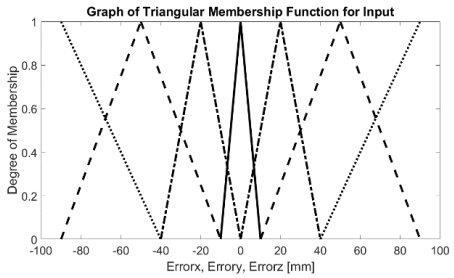
جدول ۳: قوانین فازی کنترل ربات در جهت طول، عرض و ارتفاع

قوانین	اگر خطا در جهت (طول، عرض و ارتفاع)	آنگاه حرکت ربات در جهت (طول، عرض و ارتفاع)
۱	بزرگ و مثبت	بزرگ و منفی
۲	متوسط و مثبت	متوسط و منفی
۳	کوچک و مثبت	کوچک و منفی
۴	نزدیک صفر	نزدیک صفر
۵	کوچک و منفی	کوچک و مثبت
۶	متوسط و منفی	متوسط و مثبت
۷	بزرگ و منفی	بزرگ و مثبت

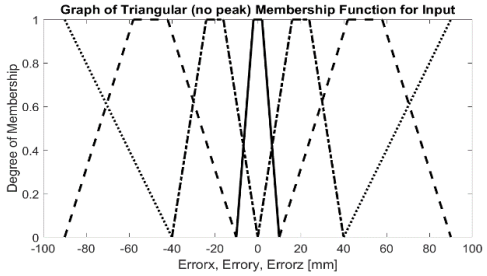
حال قوانین ذکر شده را به‌صورت ریاضی بازنویسی می‌کنیم. برای این کار مجموعه‌های، بزرگ و مثبت را با LP[۶]، متوسط و مثبت AP[۷]، کوچک و مثبت SP[۸]، نزدیک صفر NZ[۹]، بزرگ و منفی LN[۱۰]، متوسط و منفی AN[۱۱] و کوچک و منفی SN[۱۲] نشان می‌دهیم.

$$\left\{ if Error \begin{vmatrix} x \\ y \\ z \end{vmatrix} \in \begin{Vmatrix} LP \\ AP \\ SP \\ NZ \\ LN \\ AN \\ SN \end{Vmatrix} \rightarrow Linear. \begin{vmatrix} x \\ y \\ z \end{vmatrix} \in \begin{Vmatrix} LN \\ AN \\ SN \\ NZ \\ LP \\ AP \\ SP \end{Vmatrix} \right\} \quad (۱۶)$$

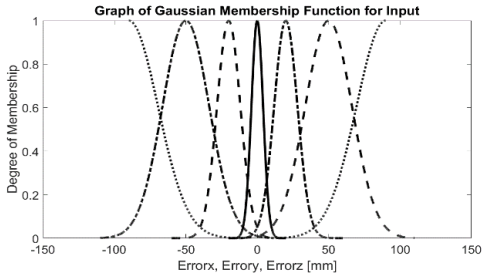
در معادله ۱۶ قوانین فازی کنترل ربات در جهت طول، عرض و ارتفاع به‌صورت ریاضی بیان شده‌اند. نوع تابع استفاده شده برای توابع عضویت ورودی و خروجی شامل مثلثی، مثلثی بدون قله، گاوسی و گاوسی بدون قله است. در شکل‌های ۶ الی ۱۱ توابع عضویت نشان داده شده‌اند.



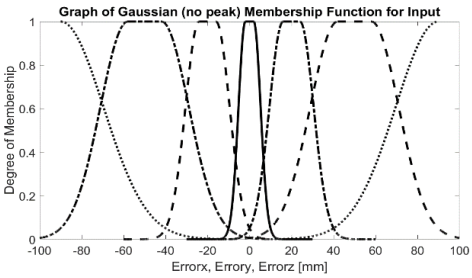
شکل ۶: تابع عضویت مثلثی برای ورودی‌ها، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (-.)
NZ (-) SP راست (-.) AP راست (--) LP راست (..)



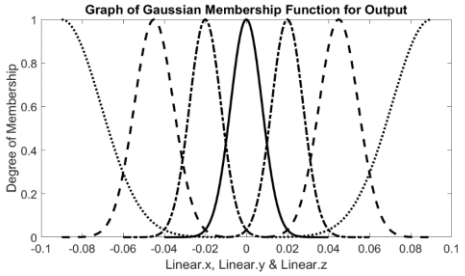
شکل ۷: تابع عضویت مثلثی بدون قله برای ورودی‌ها، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (-.) NZ (-) SP راست (.-) AP راست (--) LP راست (..)



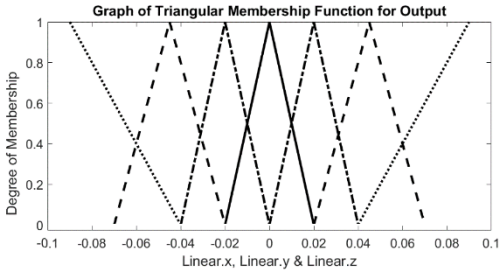
شکل ۸: تابع عضویت گاوسی برای ورودی‌ها، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (-.) NZ (-) SP راست (.-) AP راست (--) LP راست (..)



شکل ۹: تابع عضویت گاوسی بدون قله برای ورودی‌ها، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (-.) NZ (-) SP راست (.-) AP راست (--) LP راست (..)



شکل ۱۰: تابع عضویت گاوسی برای خروجی‌ها، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (-.) NZ (-) SP راست (.-) AP راست (--) LP راست (..)



شکل ۱۱: تابع عضویت مثلثی برای خروجی‌ها، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (-.) NZ (-) SP راست (.-) AP راست (--) LP راست (..)

۳ ۳ -روش‌های غیرفازی سازی

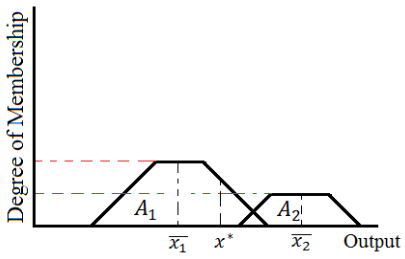
روش‌های مختلفی برای غیرفازی سازی وجود دارد که ما در اینجا از سه روش مرکز مجموع[۱۳]، میانگین وزن‌دار[۱۴] و دو روش بزرگ‌ترین و کوچک‌ترین بیشینه[۱۵] (به‌طور هم‌زمان) استفاده خواهیم کرد.

۳ ۳ ۱ -روش مرکز مجموع

در این روش توابع عضویت خروجی را به‌وسیله درصد عضویت به‌دست‌آمده از توابع عضویت ورودی قطع می‌دهیم و سپس مساحت زیرنمودار را برای هر تابع عضویت قطع داده شده (شکل۱۲) محاسبه می‌کنیم.

$$x^* = \frac{\sum_{n=1}^{p} \bar{x}_n \times A_n}{\sum_{n=1}^{p} A_n} \qquad (۱۷)$$

در رابطه ۱۷ A_n مساحت زیرنمودار تابع عضویت مجموعه nام است. p تعداد مجموعه‌هایی با درصد عضویت غیر صفر به ازای ورودی کنترلر است. $\overline{x_n}$ طول مرکز سطح نمودار تابع عضویت مجموعه nام و x^* خروجی کنترلر است.



شکل ۱۲: نحوه قطع توابع عضویت خروجی در روش مرکز مجموع

۳ ۳ ۲ -روش میانگین وزن‌دار

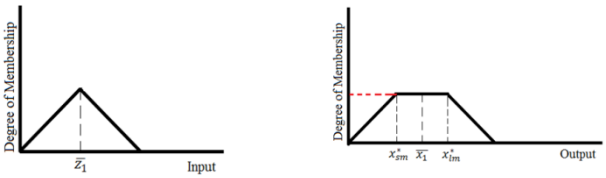
در این روش پس از محاسبه درصد عضویت ورودی کنترلر نسبت به هر یک از مجموعه‌ها، با استفاده از رابطه ۱۸ خروجی مورد نظر محاسبه می‌شود.

$$x^* = \frac{\sum_{n=1}^{p} \mu\left(\bar{x}_n\right) \times \bar{x}_n}{\sum_{n=1}^{p} \mu\left(\bar{x}_n\right)} \qquad (۱۸)$$

در رابطه ۱۸ $\overline{x_n}$ طول مرکز سطح نمودار تابع عضویت nام و $\mu(\overline{x_n})$ درصد عضویت مربوط به مجموعه‌ای با طول مرکز سطح نمودار $\overline{x_n}$ است.

۳ ۳ ۳ -روش بزرگ‌ترین و کوچک‌ترین بیشینه

این روش از دو روش مجزای بزرگ‌ترین بیشینه و کوچک‌ترین بیشینه تشکیل یافته است که ما در اینجا این دو روش را هم‌زمان و در کنار هم بکار می‌بریم. درصد عضویت محاسبه شده برای ورودی، تابع عضویت خروجی را در دو نقطه قطع می‌کند. حال اگر ورودی بزرگ‌تر از طول مرکز تابع عضویت ورودی باشد، خروجی متناظر نقطه قطع دوم به‌عنوان خروجی نهایی معرفی می‌شود. و اگر ورودی کوچک‌تر از طول مرکز تابع عضویت ورودی باشد، خروجی متناظر نقطه قطع اول را به‌عنوان خروجی نهایی معرفی می‌کنیم (شکل۱۳).



شکل ۱۳: تابع عضویت خروجی قطع شده با درصد عضویت و طول مرکز سطح نمودار $\overline{x_1}$ و خروجی نهایی x^*_{lm} برای روش بزرگ‌ترین بیشینه و خروجی نهایی x^*_{sm} برای کوچک‌ترین بیشینه (راست). تابع عضویت ورودی با طول مرکز مساحت زیرنمودار $\overline{z_1}$ (چپ)

در شکل ۱۳ (چپ) تابع عضویت ورودی با طول مرکز سطح نمودار، $\overline{z_1}$ نمایش داده شده است. اگر ورودی کنترلر از عدد $\overline{z_1}$ بزرگ‌تر باشد، از روش بزرگ‌ترین بیشینه، و اگر ورودی کنترلر از عدد $\overline{z_1}$ کوچک‌تر باشد، از روش کوچک‌ترین بیشینه استفاده خواهیم کرد. در شکل ۱۳ (راست) تابع عضویت خروجی با طول مرکز سطح نمودار، $\overline{x_1}$ نمایش داده شده است. این تابع با درصد عضویت قطع داده شده است. حال اگر از روش بزرگ‌ترین بیشینه استفاده کنیم از خروجی x^*_{lm}، و اگر از روش کوچک‌ترین بیشینه استفاده کنیم از خروجی x^*_{sm} به‌عنوان خروجی کنترلر استفاده خواهیم کرد.

۴ -نتایج

روش‌های مختلف کنترل فازی در اینجا بکار برده شدند تا بهترین روش برای ایجاد شناوری خودکار کوادکوپتر AR.Drone 2.0 شبیه‌سازی‌شده در محیط Gazebo معرفی شود. خروجی این کنترلرها شامل پارامترهای Linear.x, Linear.y, Linear.z هستند، که تنها به‌وسیله این سه پارامتر، می‌توان شناوری ربات را کنترل کرد. پارامتر کنترلی Angular.z در حالت کلی بر روی شناوری بی‌تأثیر است اما دقت شناوری را تحت تأثیر قرار می‌دهد. لذا ما ابتدا روش‌های کنترلی مختلف را بدون کنترل پارامتر Angular.z پیاده‌سازی و مقایسه کرده‌ایم تا بهترین روش کنترلی فازی مشخص شود. سپس جهت افزایش دقت شناوری ربات پارامتر Angular.z را به‌وسیله بهترین روش کنترل فازی به‌دست‌آمده از حالت اول (بدون کنترل پارامتر Angular.z) کنترل خواهیم کرد.

همان‌طورکه اشاره شد، برای کنترل فازی می‌توان به صورت‌های متفاوتی نوع توابع عضویت اعم از مثلثی، مثلثی بدون قله، گاوسی و گاوسی بدون قله را تعریف کرد. همچنین روش‌های متفاوتی اعم از مرکز مجموع، میانگین وزن‌دار و بزرگ‌ترین و کوچک‌ترین بیشینه، برای غیر فازی سازی وجود دارند. نتیجه اینکه همه روش های ذکر شده بایستی مورد ارزیابی قرار گیرند. در جدول ۴ اختصار این روش‌ها آورده شده است. هر روش تقریباً به مدت ۹۰ ثانیه در شبیه ساز Gazebo بر روی کوادکوپتر AR.Drone 2.0 به اجرا درآمده و داده‌های مربوط به موقعیت لحظه‌ای ربات (خروجی پردازش تصاویر دوربین زیرین ربات) جمع آوری شده‌اند.

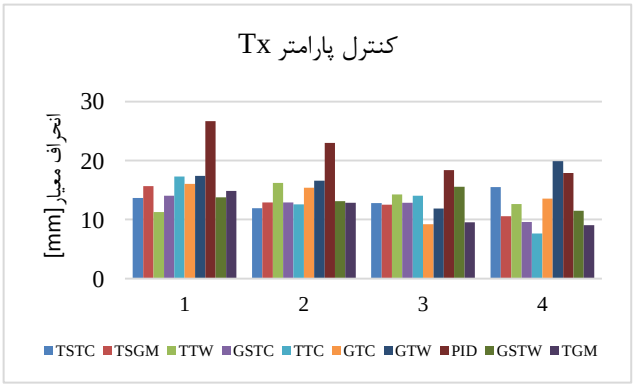
جدول ۴: اختصار روش‌های فازی بکار رفته

اختصار	روش غیر فازی سازی	نوع تابع خروجی	نوع تابع ورودی
TTW	میانگین وزن‌دار	مثلثی	مثلثی
TTM	بزرگ‌ترین کوچک‌ترین بیشینه	مثلثی	مثلثی
TTC	مرکز مجموع	مثلثی	مثلثی
TSTW	میانگین وزن‌دار	مثلثی	مثلثی بدون قله
TSTM	بزرگ‌ترین کوچک‌ترین بیشینه	مثلثی	مثلثی بدون قله
TSTC	مرکز مجموع	مثلثی	مثلثی بدون قله
GTW	میانگین وزن‌دار	مثلثی	گاوسی
GTM	بزرگ‌ترین کوچک‌ترین بیشینه	مثلثی	گاوسی
GTC	مرکز مجموع	مثلثی	گاوسی
GSTW	میانگین وزن‌دار	مثلثی	گاوسی بدون قله
GSTM	بزرگ‌ترین کوچک‌ترین بیشینه	مثلثی	گاوسی بدون قله
GSTC	مرکز مجموع	مثلثی	گاوسی بدون قله
GGM	بزرگ‌ترین کوچک‌ترین بیشینه	گاوسی	گاوسی
TGM	بزرگ‌ترین کوچک‌ترین بیشینه	گاوسی	مثلثی
TSGM	بزرگ‌ترین کوچک‌ترین بیشینه	گاوسی	مثلثی بدون قله
GSGM	بزرگ‌ترین کوچک‌ترین بیشینه	گاوسی	گاوسی بدون قله
GGC	مرکز مجموع	گاوسی	گاوسی
GSGC	مرکز مجموع	گاوسی	گاوسی بدون قله
TGC	مرکز مجموع	گاوسی	مثلثی
TSGC	مرکز مجموع	گاوسی	مثلثی بدون قله

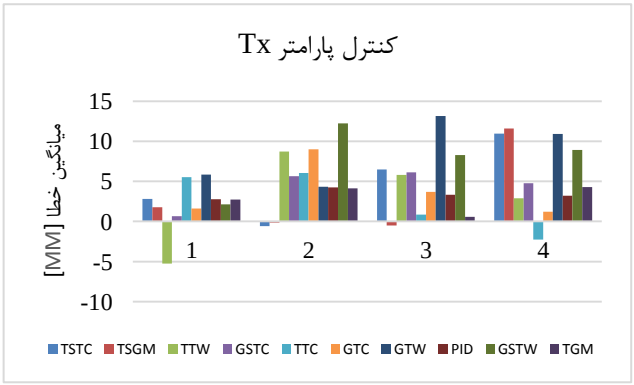
در شکل‌های ۱۴ الی ۱۹ نتایج انحراف معیار و میانگین خطا برای ۱۰ روش انتخاب شده از بین روش‌های موجود در جدول ۴ آورده شده است. در این شکل‌ها هر روش با یک رنگ خاص مشخص شده است و

در هر شکل ۴ گروه متفاوت با اعداد ۱ تا ۴ آورده شده است. عدد ۱ برای $T_z = 700mm$ و اعداد ۲ تا ۴ به ترتیب برای $T_z =$ $600, 500, 400\ mm$ هستند.

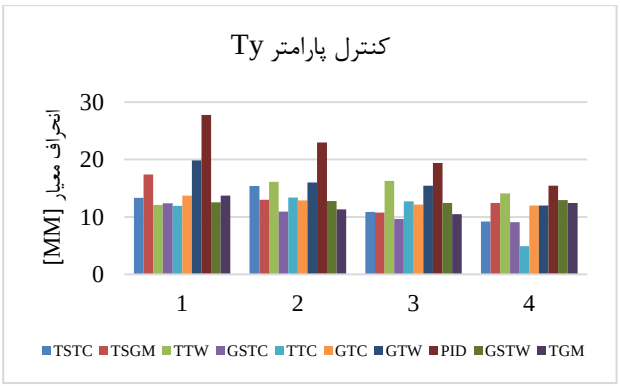
با توجه به شکل‌های ۱۴ تا ۱۹ بهترین روش برای کنترل پارامترهای T_z, T_y, T_x به ترتیب روش‌های TTC, TTC, TGM هستند و در حالت کلی روش کنترلی TGM بهترین روش کنترلی برای کنترل همه پارامترها است. حال پارامتر Angular.z را توسط بهترین روش به‌دست‌آمده از حالت قبل (روش TGM) کنترل می‌کنیم. برای این کار لازم است توابع عضویت ورودی و خروجی را برای پارامتر زاویه انحراف تعریف کنیم. به‌مانند مجموعه‌های تعریف شده برای پارامترهای طول، عرض و ارتفاع برای پارامتر زاویه انحراف از هفت مجموعه استفاده شده است که قوانین و توابع عضویت آن در جدول ۵ آورده شده است. لازم به ذکر است که روند محاسبه محدوده توابع مانند دیگر پارامترها بوده و دارای اصول ثابتی هستند.



شکل ۱۴: نتایج انحراف معیار خطا مربوط به کنترل پارامتر Tx

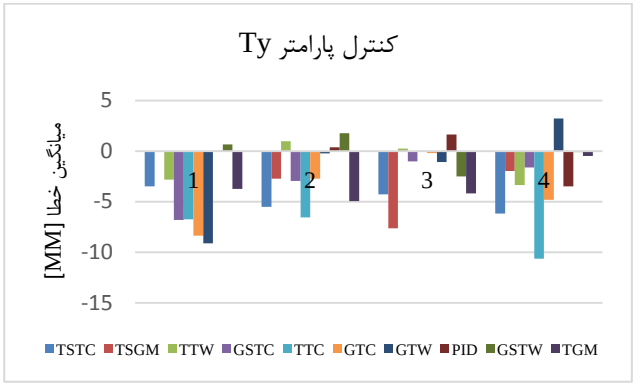


شکل ۱۵: نتایج میانگین خطا مربوط به کنترل پارامتر Tx

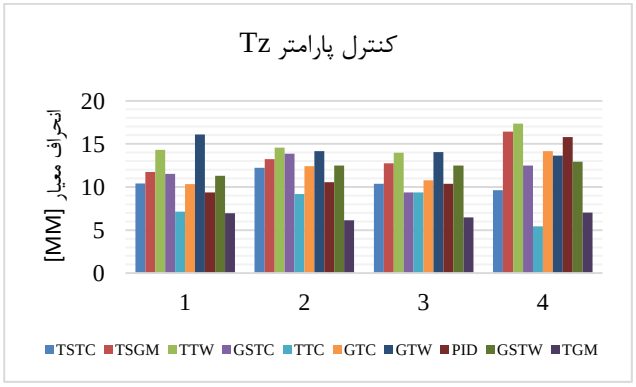


شکل ۱۶: نتایج انحراف معیار خطا مربوط به کنترل پارامتر Ty

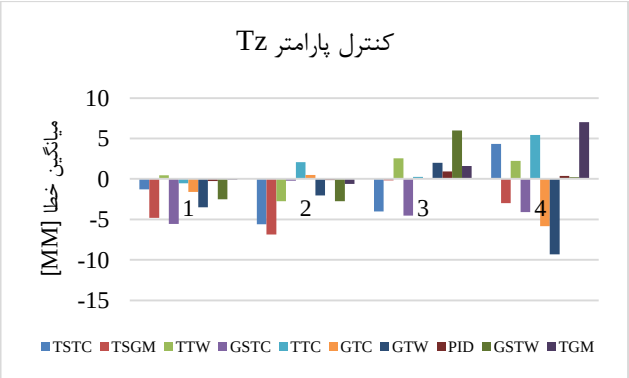
حال قوانین اشاره شده در بالا را به‌صورت ریاضی بیان می‌کنیم. به این صورت که مجموعه‌های بزرگ و پادساعت‌گرد را با LCC[۱۶]، متوسط و پادساعت‌گرد ACC[۱۷]، کوچک و پادساعت‌گرد SCC[۱۸]، بزرگ و ساعت‌گرد LC[۱۹]، متوسط و ساعت‌گرد AC[۲۰] و کوچک و ساعت‌گرد SC[۲۱] نشان می‌دهیم.



شکل ۱۷: نتایج میانگین خطا مربوط به کنترل پارامتر Ty



شکل ۱۸: نتایج انحراف معیار خطا مربوط به کنترل پارامتر Tz



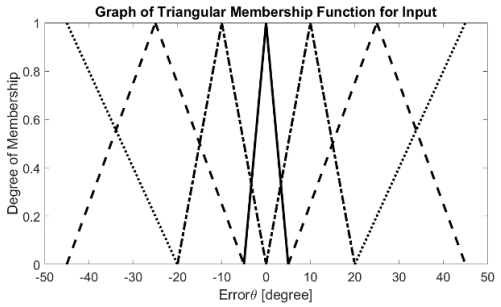
شکل ۱۹: نتایج میانگین خطا مربوط به کنترل پارامتر Tz

جدول ۵: قوانین فازی کنترل زاویه انحراف ربات

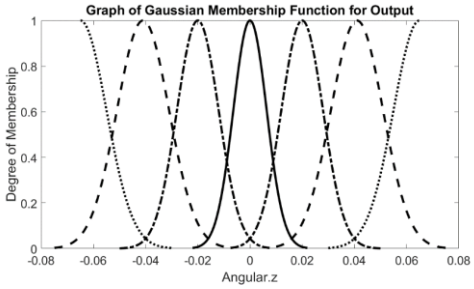
قوانین	اگر خطا در جهت	آنگاه چرخش در جهت
۱	پادساعت‌گرد بزرگ	ساعت‌گرد بزرگ
۲	پادساعت‌گرد متوسط	ساعت‌گرد متوسط
۳	پادساعت‌گرد کوچک	ساعت‌گرد کوچک
۴	نزدیک صفر	نزدیک صفر
۵	ساعت‌گرد بزرگ	پادساعت‌گرد بزرگ
۶	ساعت‌گرد متوسط	پادساعت‌گرد متوسط
۷	ساعت‌گرد کوچک	پادساعت‌گرد کوچک

$$\left\{ if Error\ \theta \in \begin{vmatrix} LCC \\ ACC \\ SCC \\ NZ \\ LC \\ AC \\ SC \end{vmatrix} \rightarrow Angular\ z \in \begin{vmatrix} LN \\ AN \\ SN \\ NZ \\ LP \\ AP \\ SP \end{vmatrix} \right\} \quad (۱۹)$$

در شکل‌های ۲۰ و ۲۱ تابع عضویت ورودی به شکل مثلثی و خروجی به شکل گاوسی برای پارامتر زاویه انحراف نشان داده شده است.



شکل ۲۰: تابع عضویت مثلثی برای ورودی، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LC چپ (..) AC چپ (--) SC چپ (.-)
NZ (-) SCC راست (.-) ACC راست (--) LCC راست (..)



شکل ۲۱: تابع عضویت گاوسی برای خروجی، مجموعه‌ها توسط شکل خطوط تفکیک شده‌اند. LN چپ (..) AN چپ (--) SN چپ (.-)
NZ (-) SP راست (.-) AP راست (--) LP راست (..)

در جدول‌های ۷ تا ۹ نتایج روش‌های آورده شده در جدول ۶ مورد مقایسه قرارگرفته‌اند. هدف از این مقایسه یافتن بهترین روش کنترلی با در نظر گرفتن کنترل پارامتر Angular.z است.

جدول ۶: معرفی روش‌های کنترلی I، II، III و IV

پارامترهای کنترلی / روش‌های کنترلی	Linear.x	Linear.y	Linear.z	Angular.z
I	TGM	TGM	TGM	-
II	TGM	TGM	TGM	TGM
III	TTC	TTC	TTC	-
IV	TTC	TTC	TGM	TGM

جدول ۷: خطای حاصل از روش‌های منتخب برای کنترل T_x

پارامتر کنترل شده	ارتفاع	روش	میانگین	انحراف معیار	روش	میانگین	انحراف معیار
	۷۰۰	I	۲/۷۳	۱۴/۸۳	II	۳/۸۰	۱۱/۹۷
	۶۰۰		۴/۱۱	۱۲/۸۶		۰/۰۶	۱۰/۹۲
	۵۰۰		۰/۵۹	۹/۵۲		۰/۷۹	۸/۸۵
	۴۰۰		۴/۳۰	۹/۰۶		۱/۶۱	۸/۲۵
	۷۰۰	III	۵/۶۲	۱۴/۷۹	IV	-۰/۹۷	۱۲/۴۶
	۶۰۰		۴/۵۴	۱۴/۰۴		-۷/۶۰	۱۲/۷۹
	۵۰۰		-۳/۲۹	۱۲/۱۷		۳/۰۴	۱۳/۴۶
	۴۰۰		۸/۷۸	۱۴/۰۴		۷/۹۹	۱۴/۱۳

جدول ۸: خطای حاصل از روش‌های منتخب برای کنترل T_y

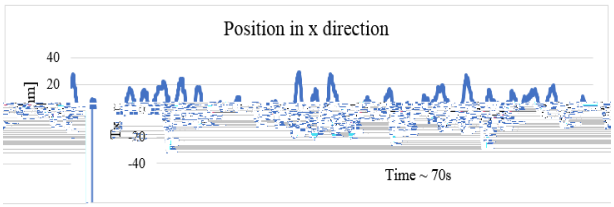
پارامتر کنترل شده	ارتفاع	روش	میانگین	انحراف معیار	روش	میانگین	انحراف معیار
	۷۰۰	I	-۳/۷۴	۱۳/۶۹	II	-۱/۱۵	۱۳/۸۸
	۶۰۰		-۴/۹۶	۱۱/۲۹		۰/۷۱	۱۲/۸۶
	۵۰۰		-۴/۱۹	۱۰/۴۶		-۰/۰۵	۹/۹۹
	۴۰۰		-۰/۴۹	۹/۰۸		۱/۶۱	۸/۲۵
	۷۰۰	III	۱/۸۹	۱۴/۴۳	IV	-۴/۱۴	۱۲/۵۰
	۶۰۰		-۳/۵۶	۱۳/۵۲		۴/۸۳	۱۴/۱۸
	۵۰۰		۱/۷۸	۱۲/۷۱		۱/۰۳	۸/۳۶
	۴۰۰		۵/۷۰	۸/۴۴		-۵/۰۲	۱۱/۲۷

جدول ۹: خطای حاصل از روش‌های منتخب برای کنترل T_z

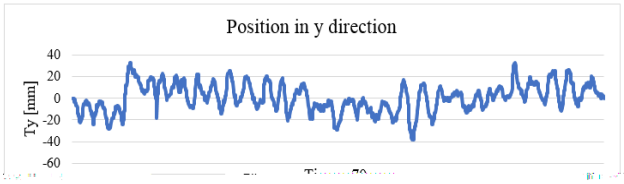
پارامتر کنترل شده	ارتفاع	روش	میانگین	انحراف معیار	روش	میانگین	انحراف معیار
	۷۰۰	I	-۰/۰۹	۶/۹۴	II	-۰/۹۷	۷/۵۷
	۶۰۰		-۰/۶۲	۶/۱۳		-۰/۱۱	۶/۸۲
	۵۰۰		۱/۶۱	۶/۴۶		-۱/۱۵	۶/۲۵
	۴۰۰		۳/۱۳	۷/۰۱		-۰/۹۳	۵/۹۸
	۷۰۰	III	-۳/۱۷	۷/۴۵	IV	-۴/۲۲	۶/۷۲
	۶۰۰		-۱/۹۱	۵/۹۰		-۳/۰۷	۶/۰۵
	۵۰۰		-۰/۳۹	۴/۹۵		۱/۲۵	۶/۷۲
	۴۰۰		-۰/۲۰	۷/۷۰		-۱/۵۱	۶/۶۳

بهترین نتیجه در هر ارتفاع با یک خط مشخص شده است. همان‌طورکه پیداست روش II در هفت مورد بهترین نتیجه را داشته است. به همین دلیل روش مذکور را می‌توان به‌عنوان مطلوب‌ترین روش معرفی کرد.

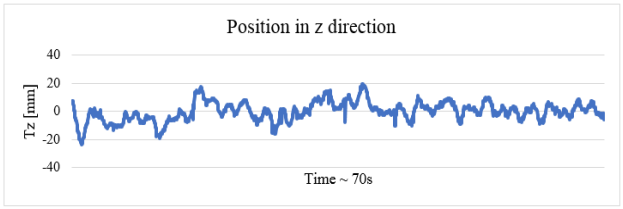
حال بهترین روش کنترلی (روش II) را جهت شناوری خودکار کوادکوپتر در شبیه‌ساز Gazebo بکار می‌گیریم. نتایج مربوط به مکان لحظه‌ای ربات برحسب میلی‌متر و زاویه انحراف آن برحسب درجه در دستگاه مختصات سراسری و در طی مدت زمان تقریبی ۱۲۰ ثانیه در شکل‌های ۲۲ الی ۲۴ نمایش داده شده است. در شکل‌های مذکور کنترلر سعی دارد ربات را به مکان مطلوب $(x = 0, y = 0, z = 600\ mm, \theta = 0^\circ)$ هدایت کند. همان‌طورکه قابل مشاهده است در شکل ۲۲ بیشترین انحراف ربات از حالت مطلوب $(x = 0)$ کمتر از ۳۰ میلی‌متر، و در شکل ۲۳ انحراف از حالت مطلوب $(y = 0)$ کمتر از ۴۰ میلی‌متر است. همچنین ارتفاع مطلوب ربات در ۶۰۰ میلی‌متر تنظیم شده است که تنها کمتر از ۲۰ میلی‌متر انحراف حول این ارتفاع مشاهده می‌شود (شکل ۲۴).



شکل ۲۲: خطای مکان ربات در امتداد محور طول‌ها برحسب میلی‌متر



شکل ۲۳: خطای مکان ربات در امتداد محور عرض‌ها برحسب میلی‌متر



شکل ۲۴: خطای مکان ربات در امتداد محور ارتفاع بر حسب میلی‌متر

۵ -نتیجه‌گیری

در این مقاله هدف ما شناوری خودکار کوادکوپتر AR.Drone 2.0 بود. برای شبیه‌سازی ربات از شبیه‌ساز Gazebo استفاده کرده‌ایم. یک الگوریتم جدید موسوم به GSPnP برای تخمین موقعیت ربات پیشنهاد گردید. الگوریتم پیشنهادی که با استفاده از اطلاعات هندسی سیستم به تخمین موقعیت می‌پردازد نیازی به محاسبات هندسی پیچیده ندارد و بنابراین پیاده‌سازی آن ساده است و به دلیل ماهیت غیرتکراری آن از سرعت بالایی برخوردار است. همچنین نتایج شبیه‌سازی در بخش ۴ نیز بیانگر دقت قابل‌قبول الگوریتم در کاربردهای شناوری ربات است. جهت رسیدن به مطلوب‌ترین حالت شناوری عملکردهای مختلف کنترلر فازی با یکدیگر مقایسه شده‌اند. برای مقایسه عملکرد کنترلرها ما انحراف معیار و میانگین خطا را موردتوجه قرار داده‌ایم. چرا که هدف ما شناوری خودکار ربات حول نقطه هدف تعریف شده با کمترین انحراف است. ما مکان لحظه‌ای ربات را در طی مدت زمان معین برای کنترلرهای مختلف ضبط کردیم و بعد از استخراج میانگین و انحراف معیار خطا برای پارامترهای طول، عرض و ارتفاع توانستیم بهترین کنترلر (TGM) را معرفی کنیم.

در کنترلر TGM توابع عضویت ورودی از نوع مثلثی و خروجی از نوع گاوسی انتخاب شده است. همچنین برای غیرفازی سازی روش بزرگ‌ترین و کوچک‌ترین بیشینه بکار برده شده است. لازم به ذکر است که سرعت عملکرد کنترلر طراحی شده بسیار حائز اهمیت است. به‌طوری‌که چنانچه سرعت عملکرد کنترلر پایین باشد ولی دقت محاسبات بالا، به نتیجه مطلوب نخواهیم رسید. برعکس این موضوع (سرعت عملکرد بالای کنترلر و دقت پایین محاسبات) نیز صادق است. به همین جهت باید مصالحه‌ای بین سرعت عملکرد و دقت محاسبات انجام شود. کنترلر TGM با استناد بر نتایج به‌دست‌آمده به بهترین مصالحه بین سرعت عملکرد و دقت محاسبات می‌رسد.

کنترل زاویه انحراف ربات می‌تواند بر روی دقت شناوری آن تأثیرگذار باشد. لذا ما با طراحی کنترلر فازی بر مبنای کنترلر TGM برای کنترل زاویه انحراف ربات این موضوع را بررسی کردیم و با مقایسه میانگین و زاویه انحراف نتایج به‌دست‌آمده از مکان لحظه‌ای ربات به این نتیجه رسیدیم که کنترل زاویه انحراف ربات توسط کنترلر TGM دقت شناوری را بالا می‌برد. در خاتمه با بکار بردن کنترلر TGM

در کنترل پارامترهای طول، عرض، ارتفاع و زاویه انحراف مکان لحظه‌ای ربات را طی ۱۲۰ ثانیه ضبط کردیم. نتایج نشان دهنده خطای (انحراف از مقدار مطلوب) کم‌تر از ۳۰، ۴۰ و ۲۰ میلی‌متر به ترتیب در جهت‌های طول، عرض و ارتفاع است. درحالی‌که نتایج مرجع [۱۶] خطای کم‌تر از ۲۵۰ و ۱۵۰ میلی‌متر را در کنترل طول و عرض ربات را نشان می‌دهد.

مراجع

[1] https://www.dji.com/
[2] https://www.parrot.com/us/
[3] Parrot AR.Drone 2.0. http://ardrone2.parrot.com/
[4] https://www.ni.com/en-us/shop/labview.html
[5] ArUco: a minimal library for Agumented Reality applications based on OpenCV. https://www.uco.es/investiga/grupos/ava/node/26
[6] M. A. Mogensen, *The AR Drone LabVIEW Toolkit: A Software Framework for the Control of Low Cost Quadrotor Aerial Robots*, M.Sc. Thesis, TUFTS University, Medford, MA, 2012.
[7] T. Krajnik, V. Vonasek, D. Fiser and J. Faigel, "AR-Drone as a Platform for Robotics," Int. Conf. on Research and Education in Robotics-EUROBOT, pp. 172-186, 2011.
[8] S. Yue, *Modeling, Identification and Control of a Quadrotor Drone Using Low-Resolution Sensing*, M.Sc. Thesis, University of Illinois, Champaign, IL, 2012.
[9] A. Prayitno, V. Indrawati and G. Utomo, "Trajectory Tracking of AR.Drone Quadrotor Using Fuzzy Logic Controller," TELKOMNIKA, vol. 12, no. 4, pp. 819-828, 2014.
[10] V. Indrawati, A. Prayitno and G. Utomo, "Comparison of Two Fuzzy Logic Controller Schemes for Position Control of AR.Drone," IEEE Int. Conf. on Information Technology and Electrical Engineering (ICITEE), pp. 360-363, 2015.
[11] V. Indrawati, A. Prayitno and T. A. Kusuma, "Waypoint Navigation of AR.Drone Quadrotor Using Fuzzy Logic Controller," TELKOMNIKA, vol. 13, no. 3, pp. 930-939, 2013.
[12] Sarah Y. Tang, *Vision-Based Control for Autonomous Quadrotor*, Undergraduated Senior Thesis, Princeton University, NJ, 2013.
[13] K. Boudjit and C. Larbes, "Detection and Target Tracking With a Quadrotor using Fuzzy Logic," Int. Conf. on Modeling, Identification and Control (ICMIC), pp. 127-132, 2016.
[14] The FuzzyLite Libraries for Fuzzy Logic Control. http://www.fuzzylite.com/
[15] Y. Tao, G. Xie, Y. Chen, H. Xiong, H. Liu, J. Zheng and J. Gao, "A PID and Fuzzy Logic Based Method for Quadrotor Aircraft Control Motion," Journal of Intelligent and Fuzzy Systems, vol. 31, no. 6, pp. 2975-2983, 2016.
[16] C. H. Pi, V. B. Sheng and S. Cheng, "A Dual-loop Approach with Visual Servoing Fuzzy Control for Marker Navigation Quadcopter," Int. Conf. on Intelligent Systems and Image Processing, pp. 384-390, 2016.
[17] OpenCV: Open Source Computer Vision Library. https://opencv.org/
[18] https://github.com/tum-vision/tum_simulator/
[19] Ardrone autonomy ros stack. https://github.com/AutonomyLab/ardrone_autonomy/
[20] E. H. Mamdani, "Application of Fuzzy Logic to Approximate Reasoning using Linguistic Synthesis," IEEE Tran. On Computers, vol. 100, no. 12, pp. 1182-1191, 1977.
[21] Atulya Shivam Shree, Radhe Shyam Sharma,Laxmidhar Behera and K.S. Venkatesh, "Position Based Visual Control of the Hovering Quadcopter," International Conference on Intelligent Human Computer Interaction, IHCI 2016, pp. 15-26.
[22] M.Bergamasco and M.Lovera. "Identification of Linear Models for the Dynamics of a Hovering Quadrotor ," IEEE Transactions on Control Systems Technology, vol. 22, no. 5, pp. 1696–1707, Sept 2014.
[23] B.T.M. Leong, S.M. Low, and M.P.L. Ooi, "Low-cost microcontroller-based hover control design of a quadcopter," *Procedia Engineering*, vol. *41*, pp. 458-464. 2012.
[24] Mohd Ariffanan Mohd Basri, "Robust Backstepping Controller Design with a Fuzzy Compensator for Autonomous Hovering Quadrotor UAV," Iranian Journal of Science and Technology, Transactions of Electrical Engineering, Vol. 42, Issue 3, pp. 379–391, September 2018.
[۲۵] علیرضا مدیرروستا و مهدی خدابنده، "طراحی یک روش کنترل مد لغزشی انتگرالی تطبیقی برای پایدارسازی زمان محدود و مقاوم پرنده چهارملخه"، مجله مهندسی برق دانشگاه تبریز، دوره ۴۶، شماره ۱، صفحه ۳۲۱-۳۳۲ ، بهار ۱۳۹۵.
[۲۶] الناز قنبری، مهرام محبوب خواه و قادر کریمیان، "تعیین موقعیت عملگر نهایی یک ربات موازی چهار درجه آزادی با استفاده از روش بینایی ماشین "، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۴، ۱۳۹۵.

زیرنویس‌ها

[1] Quad Copter
[2] Quad-Rotor
[3] Robotic Operating System (ROS)
[4] Laboratory Virtual Instrument Engineering Workbench
[5] Simultaneous Localization and Mapping
[6] Large Positive
[7] Average Positive
[8] Small Positive
[9] Near Zero
[10] Large Negative
[11] Average Negative
[12] Small Negative
[13] Center of Sums (COS)
[14] Weighted Average (WA)
[15] Smallest and Largest of Max
[16] Large Counter Clockwise
[17] Average Counter Clockwise
[18] Small Counter Clockwise
[19] Large Clockwise
[20] Average Clockwise
[21] Small Clockwise