Islamic Azad University
Khomeinishahr Branch
Faculty of Mechanical Engineering

"M.Sc." Thesis on
Mechatronics Engineering

Object Recognition by a Flying Robot Using Image Processing Algorithms

Thesis Advisor:

Dr. Ata Jahangir Moshayedi

Consulting Advisor:

Dr. Amin Kolahdooz

By:

Mohammad Zarei

Summer 2019

Abstract:

In the past few years, researchers and scientists have paid particular attention to unmanned aerial vehicles, especially multirotors. Various research groups around the world are working on these types of drones, which are used in various fields such as rescue operations, civil operations, surveying, and sampling of infected areas. In most applications, it is necessary to determine the position of a particular object using these robots. To do this, we extract features from the images sent by the robot and match them to the known image features to identify the object we are looking for. One application of this matching is to compare two or more images and stitch them together to extract two-dimensional or three-dimensional maps. In this study, image processing algorithms including SIFT, SURF, ORB, and BRISK were implemented on three types of hardware including Raspberry Pi3, Odroid C2, and Intel NUC, and they were evaluated in terms of speed and accuracy using the Python programming language and OpenCV machine vision library. Subsequently, the results of the study were analyzed, and appropriate algorithms were proposed in accordance with the applications of the flying robot. The ORB algorithm was found to be the fastest, while the SIFT and BRISK algorithms were the slowest. The Intel NUC hardware had the least mass-to-speed efficiency, and Odroid C2 averaged the best mass-to-speed efficiency.

On the other hand, the SIFT algorithm provided the most precision in finding image properties among the investigated algorithms, and the ORB algorithm provided the least precision in finding objects in the images.

Keywords: Multorotor, Image processing, Object detection, OpenCV, Python

دانشگاه آزاد اسلامی

واحد خمینی شهر

دانشکده مکانیک

پایان نامه برای دریافت درجه کارشناسی ارشد **M.Sc**

مهندسی مکاترونیک گرایش اتوماتیک و کنترل تولید

پیاده سازی تشخیص اشیا با استفاده از الگوریتم‌های پردازش تصویر در ربات پرنده

استاد راهنما:

دکتر آتا جهانگیر مشیدی

استاد مشاور:

دکتر امین کلاهدوز

نگارنده:

محمد زارعی

تابستان 98


بسم الله الرحمن الرحيم

دانشگاه آزاد اسلامی

واحد خمینی شهر

دانشکده مکانیک

پایان نامه برای دریافت درجه کارشناسی ارشد **M.Sc**

مهندسی مکاترونیک گرایش اتوماتیک و کنترل تولید

پیاده سازی تشخیص اشیا با استفاده از الگوریتم‌های پردازش تصویر در ربات پرنده

نگارنده:

محمد زارعی

در تاریخ ../../.... توسط کمیته تخصصی زیر مورد بررسی و تصویب نهائی قرار گرفت.

1- استاد راهنمای پایان نامه دکتر آتا جهانگیر مشیدی

2-استاد مشاور پایان نامه دکتر امین کلاهدوز

3-استاد داور دکترمهدی نصری

4- استاد داور دکتر احمد کشاورزی

5-مدیر تحصیلات تکمیلی گروه دکتر محمد هاشمیان

تابستان 98

معاونت پژوهش و فن آوری

به نام خدا

منشور اخلاق پژوهش

باباوری از خداوند سبحان و اعتقاد به این که عالم محضر خدا است و همواره ناظر بر اعمال انسان و به منظور پاس داشت مقام بلند دانش و پژوهش و نظر به اهمیت جایگاه دانشگاه در اعتلای فرهنگ و تمدن بشری، ما دانشجویان و اعضای هیات علمی واحدهای دانشگاه آزاد اسلامی متعهد می گردیم اصول زیر را در انجام فعالیت های پژوهشی مد نظر قرار داده و از آن تخطی نکنیم:

۱-۱ اصل حقیقت جویی: تلاش در راستای پی جویی حقیقت و وفاداری به آن و دوری از هرگونه پنهان سازی حقیقت.

۲-۱ اصل رعایت حقوق: التزام به رعایت کامل حقوق پژوهشگران و پژوهیدگان (انسان، حیوان و نبات) و سایر صاحبان حق.

۳-۱ اصل مالکیت مادی و معنوی: تعهد به رعایت کامل حقوق مادی و معنوی دانشگاه و کلیه همکاران پژوهش.

۴-۱ اصل منافع ملی: تعهد به رعایت مصالح ملی و در نظر داشتن پیشبرد و توسعه کشور در کلیه همکاران پژوهش.

۵-۱ اصل رعایت انصاف و امانت: تعهد به اجتناب از هرگونه جانب داری غیر علمی و محافظت از اموال، تجهیزات و منابع در اختیار.

۶-۱ اصل رازداری: تعهد به صیانت از اسرار و اطلاعات محرمانه افراد، سازمان ها و کشور و کلیه افراد و نهادهای مرتبط با تحقیق.

۷-۱ اصل احترام: تعهد به رعایت حریم ها و حرمت ها در انجام تحقیقات و رعایت جانب تقد و خودداری از هرگونه حرمت شکنی.

۸-۱ اصل ترویج: تعهد به رواج دانش و اشاعه نتایج آن به همکاران علمی و دانشجویان به غیر از مواردی که منع قانونی دارد.

۹-۱ اصل برائت: التزام به برائت جویی از هرگونه رفتار غیر حرفه ای و اعلام موضع نسبت به کسانی که حوزه علم و پژوهش را به شائبه های غیر علمی می آلایند.

اینجانب محمد زارعی دانش‌آموخته مقطع کارشناسی ارشد در رشته مکاترونیک که در تاریخ 1398/06/14 از پایان‌نامه خود تحت عنوان **پیاده سازی تشخیص اشیا با استفاده از الگوریتم‌های پردازش تصویر در ربات پرنده** با کسب نمره 18.5 و درجه ..... دفاع نموده‌ام بدینوسیله متعهد میشوم:

1- این پایان‌نامه / رساله حاصل تحقیق و پژوهش انجام شده توسط اینجانب بوده و در مواردی که از دستاوردهای علمی و پژوهشی دیگران (اعم از پایان‌نامه، کتاب، مقاله و...) استفاده نموده‌ام، مطابق ضوابط و رویه موجود، نام منبع مورد استفاده و سایر مشخصات آن را در فهرست مربوطه ذکر و درج کرده‌ام.

2- این پایان‌نامه قبلاً برای دریافت هیچ مدرک تحصیلی (هم‌سطح، پایین‌تر یا بالاتر) در سایر دانشگاه‌ها و مؤسسات آموزش عالی ارائه نشده است.

3- چنانچه بعد از فراغت از تحصیل، قصد استفاده و هرگونه بهره‌برداری اعم از چاپ کتاب، ثبت اختراع و... از این پایان‌نامه داشته باشم، از حوزه معاونت پژوهشی واحد مجوزهای مربوطه را اخذ نمایم.

4- چنانچه در هر مقطعی زمانی خلاف موارد فوق ثابت شود، عواقب ناشی از آن را می‌پذیرم و واحد دانشگاهی مجاز است با اینجانب مطابق ضوابط و مقررات رفتار نموده و در صورت ابطال مدرک تحصیلی‌ام هیچگونه ادعایی نخواهم داشت.

نام و نام خانوادگی

تاریخ و امضاء

تقدیم به

به پاس تعبیر عظیم و انسانی شان از کلمه ایثار و از خودگذشتگان به پاس عاطفه سرشار و گرمای امیدبخش وجودشان که در این سردترین روزگاران بهترین پشتیبان است به پاس قلب های بزرگشان که فریادرس است و سرگردانی و ترس در پناهشان به شجاعت می گراید و به پاس محبت های بی دریغشان که هرگز فروکش نمی کند **این مجموعه را به پدر و مادر عزیزم تقدیم می کنم.**

سپاسگزاری

در اینجا بر خود لازم می دانم که از زحمات بی دریغ، تلاش های بی وقفه و راهنمایی های حکیمانه استاد راهنمای محترم جناب آقای دکتر آناجهانگیر رشیدی کمال تشکر و قدردانی را داشته باشم.

همچنین از استاد گرامی جناب دکتر امین گلاهدوز که مشاوره این پژوهش را پذیرا شدند و در این مسیر از هیچ همکاری و کمکی به اینجانب دریغ نفرمودند؛ صمیمانه تشکر و قدردانی می نمایم.

در خاتمه از خانواده ام، دوستانم و تمام کسانی که در این مسیر کنارم بودند به غایت اظهار قدردانی میکنم.

فهرست

فهرست شکل‌ها

فهرست جداول

چکیده:

در چند سال گذشته محققین و دانشمندان توجه ویژه‌ای نسبت به پرنده‌های بدون سرنشین به خصوص پرنده‌های عمود پرواز داشته اند. گروه‌های تحقیقاتی مختلفی در دنیا در حال کار بر روی این نوع پرنده‌ها هستند که در زمینه‌های مختلفی از جمله امداد و نجات، عملیات‌های عمرانی، نقشـــه برداری، نمونه برداری از مناطق آلوده و ... فعالیت می کنند.تعیین موقعیت یک شـــی خاص با استفاده از این ربات‌ها در اکثر کاربری‌ها ضروری می‌باشد. برای انجام این کار، ویژگی‌های موجود در تصویر ارسالی از ربات با ویژگی‌های تصویری معلوم تطابق داده می‌شود تا جسم مورد نظر مشخص شود. یکی از کاربردهای این تطابق، مقایسه دو یا چند تصویر با یکدیگر و چسباندن تصاویر به یکدیگر و استخراج نقشه دو بعدی یا سه بعدی می‌باشد. در این پژوهش، الگوریتم‌های پردازش تصویر شامل SIFT، SURF، ORB و BRISK بر روی سه نوع سخت افزار شامل Raspberry Pi3، Odroid C2 و Intel NUC پیاده سازی شدند و از نظر سرعت و دقت با استفاده از زبان برنامه نویسی پایتون و کتابخانه بینایی ماشین OpenCV مورد بررسی قرار گرفتند. پس از آن، نتایج پژوهش مورد تحلیل قرار گرفت و متناسب با کاربردهای ربات پرنده، الگوریتم‌های مناسب پیشنهاد داده شد که الگوریتم ORB سریعترین الگوریتم و الگوریتم‌های SIFT و BRISK کندترین الگوریتم‌های مورد بررسی بودند و سخت افزار Intel NUC کمترین بازدهی جرمی را نسبت به سرعت و Odroid C2 به طور میانگین بهترین بازدهی جرمی را نسبت به سرعت ارائه دادند. از طرفی، الگوریتم SIFT بیشترین دقت را در یافتن ویژگی‌های تصاویر، در میان الگوریتم‌های مورد بررسی ارائه داد و الگوریتم ORB کمترین دقت را در یافتن ویژگی‌های تصاویر مورد بررسی ارائه داد.

واژه‌های کلیدی: مولتی روتور، پردازش تصویر، تشخیص اشیا، OpenCV، Python

1- فصل یکم: کلیات پژوهش

1–1 مقدمه

استفاده از ربات های پرنده در سال‌های اخیر دارای رشد زیادی بوده است. این ربات‌ها در ابتدا توسط نیروهای نظامی استفاده می‌شدند و امروزه در کاربردهای غیرنظامی جای خود را پیدا کردند. دامنه کاربردی این ربات‌ها شامل کشاورزی[۱]، نقشه برداری سه بعدی [۲]، عملیات امداد و نجات [۳] و حتی در صنعت سینما[۴]، می‌باشد.

این نوع ربات‌ها قابلیت کنترل توسط خلبان یا پرواز خودکار توسط GPS در مکانهای باز را دارند و جهت انجام مانور های هوایی پیچیده، پرواز خودکار، شناسایی مصدومان، بازرسی [۵] و نقشه برداری در مکانهای بسته، به کار می‌روند در بعضی مأموریت‌ها، نیاز به یافتن یک جسم مشخص در یک محیط نا ایمن برای انسانها و یا یافتن یک مصدوم در عملیات نجات تنها به کمک اپراتور ممکن نیست، این مسئله باعث شد که با استفاده از پردازش تصویر بر روی پرنده، امکان یافتن جسم فراهم شود. خطای انسانی کمتر، سرعت عمل بیشتر، عملیات خودکار، امکان استفاده از چندین ربات همزمان بدون نیاز به نیروی انسانی بیشتر و... بر اهمیت خودکار سازی عملکرد این نوع ربات ها می‌افزاید. در پرنده‌های بدون سرنشین، نیرویی که باعث پرواز می‌شود، تعیین کننده نوع پرنده است. پرنده‌های بدون سرنشین به چند دسته تقسیم می‌شوند[۶]: سیستم‌های بال ثابت[۱]، مولتی روتورها[۲] و سیستم های نامتداول که در ادامه به بررسی آن‌ها پرداخته می‌شود.

1–1–1 سیستم‌های بال ثابت:

اصطلاح بال ثابت بیانگر این است که وسیله پرنده از بالِ ثابت یا استاتیک استفاده می‌کند. بال پرنده با کمک نیروی جلوبرندگی، نیروی بالابرندگی ایجاد می‌کند و قدرت پرواز را به وسیله پرنده می‌دهد. به عنوان نمونه می‌توان به هواپیماها، کایت‌ها و گلایدرها اشاره نمود که نمونه‌ی بدون سرنشین این پرنده‌ها، پرنده‌ی PARROT DISCO است[۷] که در شکل 1–1 نشان داده شده است.

شکل 1–1 پرنده بدون سرنشین Parrot Disco [۷]

[۱]- Fixed-Wing Systems

[۲]- Multirotor Systems

2–1–1 مولتی روتورها:

چرخ‌بال[۱] به پرنده ای گفته می‌شود که از نیروی چرخش بال‌ها[۲] برای بلند شدن استفاده می‌کند. هلیکوپترها نمونه‌ای از چرخ بال‌ها هستند که از یک محور چرخان یا روتور استفاده می‌کنند. چرخ‌بال‌ها می‌توانند یک یا چند روتور داشته باشند. پرنده‌های بدون سرنشینی[۳] که از سیستم‌های گردان[۴] استفاده می‌کنند در بیشتر مواقع بیش از یک محور چرخان دارند. مولتی روتورها زیر مجموعه‌ای از چرخ‌بال‌ها هستندکه دارای چندین محورچرخان می‌باشند و معمول‌ترین آن‌ها کوادکوپترها هستند که دارای 4 موتور می‌باشند[۶].

این نوع خاص از پرنده‌های بدون سرنشین بر خلاف هلیکوپترهای معمول، از اختلاف دور ملخ برای کنترل نیروی بالابرندگی و گشتاور، استفاده می‌کنند[۸] و در آنها سیستم های پیچیده مکانیکی مشاهده نمی‌شود. نیروی بالا برندگی هر موتور در این پرنده، نقش اساسی در قدرت مانورپذیری این پرنده ایفا می‌کند. سایز کوچک و سرعت مولتی‌روتورها به آنها کمک می‌کند مانورهای پیچیده هوایی را انجام دهند[۹]. پرنده‌های مدل Phantom، نمونه بسیار محبوب از این نوع ربات‌ها را که ساخت شرکت چینی DJI دارای چهار موتور (کوادروتور یا کوادکوپتر) می‌باشد، که در شکل 1–2 نشان داده شده است. این شرکت با داشتن 72٪ از سهم بازار پهپادها، برجسته‌ترین سازنده پهپاد در بازار است[۱۰] و مدل Phantom نزدیک به 36٪ از فروش این شرکت را شامل می‌شود[۱۱]. این مدل پهپاد دارای دوربین، گیمبال، رادیو کنترل و ارسال تصویر می‌باشد و زمان پرواز آن 28 دقیقه تخمین زده می‌شود.

شکل 1–2 ربات تجاری فانتوم ساخت شرکت DJI [۱۲]

3–1–1 سیستم‌های نامتعارف:

دسته سوم متعلق به بعضی دیگر از انواع پرنده‌ها هستند که نمی‌توان آنها را بال ثابت یا مولتی روتور نامید. به این دلیل که نه دارای بال ثابتی هستند و نه دارای چندین روتور و یا دارای ویژگی هر دو گونه پرنده هستند[۶]. مولتی روتورهای هیبریدی در این

۱- Rotorcrafts

۲- Rotary Wings

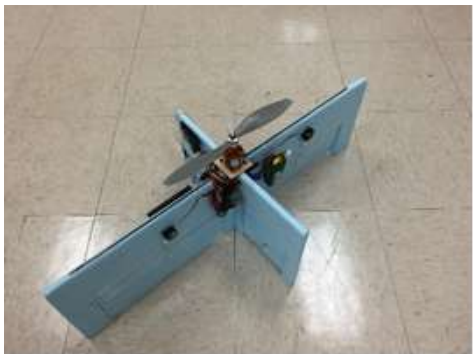
۳- Drones

۴- Rotary Systems

دسته قرار می‌گیرند. این نوع پرنده‌ها در حالی برای پرواز از بال ثابت بهره می‌برند که از چندین موتور برای فراز و فرود استفاده می‌کنند، لذا قادر به طی نمودن مسافت‌های طولانی‌تری هستند. در شکل 1–3 نمونه ساخته شده توسط شرکت Latitude نشان داده شده است[۱].این پرنده برای اوج‌گیری از چهارموتور الکتریکی استفاده می‌کند و برای حرکت رو به جلو، از یک موتور بنزینی استفاده می‌کند و این امکان را به پرنده می‌دهد که بدون نیاز به باند پرواز و لانچر[۲] نیاز ماموریت خود را انجام دهد. در اصلاح به این نوع پرنده‌ها، VTOL[۳] گفته می‌شود[۱۳].

شکل 1–3 پرنده HQ-40 ساخت شرکت Latitude [۱۴]

به عنوان نمونه می‌توان از پرنده‌های غیرمتداولی نیز مانند پرنده‌های Singlecopter و پرنده بالدار Delfly Explorer در این دسته نام برد که در شکل 1–4 و شکل 1–5 قابل مشاهده هستند.



شکل 1–4 ربات Single Copter [۱۵]

شکل 1–5 ربات Delfty Explorer ساخت دانشگاه دلفت هلند

۱- Latitudeengineering.com/products/hq

۲- Launcher

۳- Vertical Take Off and Landing

جدول 1–1 مقایسه انواع پرنده‌های بدون سرنشین

نوع پرنده	ویژگی‌ها	معایب	محدوده قیمت
پرنده‌های بال ثابت	- زمان پرواز طولانی - پوشش بالای مساحت - سرعت پرواز بالا	- برای پرواز و فرود نیازمند باند می‌باشد - کنترل مشکل پرنده و نیاز به آموزش زیاد - گران قیمت	25-120 هزار دلار
مولتی روتورها	- در دسترس بودن - راحتی استفاده - پرواز و فرود عمودی - کنترل مناسب دوربین - منطقه محدود پروازی	- کوتاه بودن زمان پرواز - ظرفیت کم حمل بار	5-65 هزار دلار
SINGLE-ROTOR	- پرواز و فرود عمودی - زمان پرواز طولانی(موتور بنزینی) - ظرفیت بالای حمل بار	- خطرناک - کنترل مشکل پرنده و نیاز به آموزش زیاد - گران قیمت	25-300 هزار دلار
VTOL	- پرواز و فرود عمودی - زمان طولانی پرواز - پوشش بالای مساحت	- نه در پرواز عمودی و نه در پرواز رو به جلو کامل نیست - هنوز در حال توسعه می‌باشد	در حال توسعه

2–1 بیان مسئله تحقیق:

1–2–1 تشخیص و تعقیب اشیاء[1]:

پایش ویدئویی[2] از موضوعات تحقیقاتی بسیار فعال در زمینه پردازش تصویر است که تلاش می‌کند اشیاء را شناسایی، تشخیص و تعقیب کند که از تعدادی عکس متوالی برای این کار استفاده می‌کند. در همین حال رفتار اشیاء را درک می‌کند و تلاش می‌کند جایگزین روش قدیمی مانیتورینگ توسط نیروی انسانی شود[۱۶]. تشخیص اشیاء مسئله بسیار مهم و چالش برانگیزی در فعالیت‌های امروزی مانند نظارت، ناوبری وسایل نقلیه و ناوبری ربات‌های هوشمند است. تمام روش‌های تعقیب اشیاء، نیازمند الگوریتم‌های تشخیص اشیا می‌باشند، که یا در تمام عکس‌های متوالی اجرا می‌شوند یا زمانی که شیء تشخیص داده شد، الگوریتم تعقیب کننده شروع به کار می‌کند. تعقیب شیء عبارت است از مکان‌یابی یک یا چند شیء در طول زمان که با استفاده از دوربین صورت بگیرد. رایانه‌های قدرتمند و همچنین در دسترس بودن دوربین‌های با کیفیت و ارزان و نیاز روز افزون به پردازش تصویر، محققین بسیاری را نسبت به الگوریتم‌های تشخیص اشیاء ترغیب نموده است[۱۶].

برای طراحی سیستم بینایی ماشین درک "ویژگی" مهم است. سیستم بینایی ماشین ویژگی‌ها را از داده‌های ورودی استخراج کرده و اطلاعات مفیدی از آن‌ها می‌سازد.

2–2–1 معرفی سیستم بینایی ماشین:

برای یک سیستم بینایی ماشین می‌توان سه مرحله تعریف کرد:

مرحله جمع آوری داده‌ها، که از یک سری سنسورها یا رابطه‌های خارجی برای به دست آوردن تصاویر استفاده می‌کند. در این مرحله می‌توان از تکنیک‌های پردازش تصویر مانند الگوریتم‌های حذف نویز برای تنظیم ورودی‌ها استفاده کرد[۱۶].

مرحله ارائه اطلاعات که مهمترین قسمت یک سیستم بینایی ماشین می‌باشد. داده‌های جمع‌آوری شده در این مرحله به ساختارهایی تبدیل می‌شوند که برای سیستم مورد نظر موثرتر باشند. در این مرحله می‌توان اطلاعاتی که برای سیستم بی اهمیت هستند را حذف و روی اطلاعاتی که برای سیستم مهم هستند، تاکید کرد. اگر ارائه اطلاعات در این مرحله به خوبی انجام شود، در مرحله بعد کار بسیار ساده‌تر می‌شود و در نتیجه زمان زیادی از طراحی سیستم بینایی ماشین صرف تعریف درست از ارائه مناسب می‌شود.

مرحله تصمیم‌گیری، در این مرحله می‌باید با توجه به روش‌های موجود، جهت خروجی سیستم تصمیم‌گیری کرد. برای کارهای پیچیده‌ای مانند شناسایی اشیا یا شناسایی چهره این مراحل احتیاج به استفاده از الگوریتم‌های یادگیری ماشین و شناسایی الگو پیدا می‌کند.

---

[1]- Object Detection and Tracking

[2]- Video surveillance

یک آنالیز ویدئویی شامل سه مرحله می‌باشد:

۱- تشخیص سوژه های متحرک[۱]

۲- تشخیص اشیاء از هر فریم تا فریم بعدی[۲]

۳- استفاده از حرکت شیء (برای تشخیص رفتار شیء)[۳]

در تشخیص اشیاء، مراحل شکل ۱–۶ طی می‌شود[۱۶]:

شکل ۱–۶ مراحل تشخیص هدف

در یک سیستم نظارتی، دوربین‌های ویدئویی در مکان‌های ثابت قرار گرفته‌اند که تصاویر خود را به نقطه خاصی ارسال می‌کنند در حالی که در ربات‌های پرنده، دوربین متحرک است، هیچ جای ثابتی ندارد و در شش درجه آزادی قابلیت حرکت دارد این امر، باعث مشکل شدن تشخیص، مکان‌یابی و تعقیب اشیا می‌شود. که ممکن است توسط یکی از عوامل زیر بوجود آمده باشد[۱۶]:

- ✓ از دست رفتن اطلاعات به خاطر تبدیل تصاویر دنیای ۳بعدی به ۲بعدی
- ✓ وجود نویز در تصویر
- ✓ وجود اجسام غیر صلب در تصویر
- ✓ وجود موانع جزئی یا کلی برای دیده شدن جسم
- ✓ اشیاء با شکل‌های پیچیده
- ✓ تغییر نور تصویر
- ✓ نیاز به پردازش همزمان (Real-Time processing)

---

[۱]- Detection of interesting moving objects

[۲]- Tracking of such objects from frame to frame

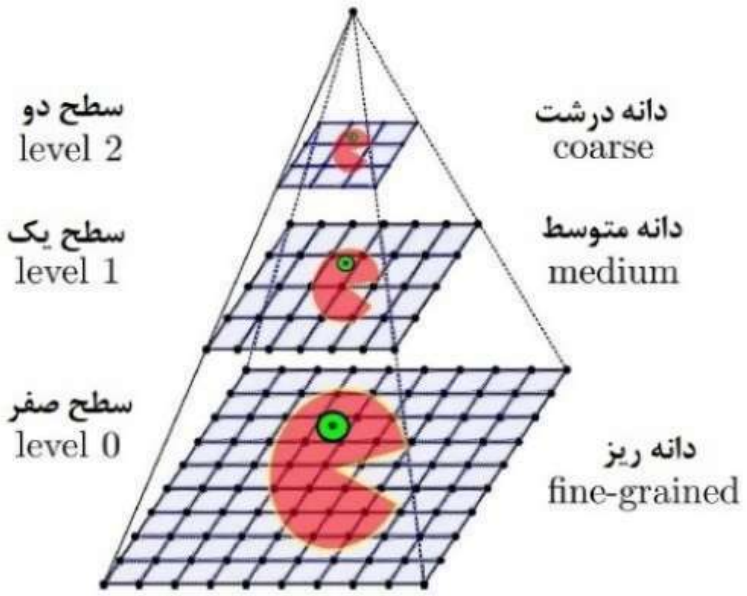
[۳]- Analysis of object tracks to recognize their behavior

3–2–1 روش‌های پرکاربرد تشخیص اشیا:

از انواع این روش‌ها می‌توان به انطباق الگو، تشخیص رنگ و تشخیص و مطابقت ویژگی نام برد که در ادامه به بررسی هر یک پرداخته می‌شود.

- **انطباق الگو[1]:**

الگوریتم انطباق الگو، الگوریتم نسبتا ساده‌ای است. برای انطباق تکه‌ای از تصویر، با تصویری که از قبل به ما داده شده است به این صورت عمل می‌شود که با حرکت دادن الگو، بر روی تصویر دریافتی و محاسبه خطاها از اختلافِ پیکسل‌های دو تصویر، مکانی که این خطاها کمترین مقدار را دارند بدست آورده می‌شود. این امکان وجود دارد که بتوان این روش را ارتقا بخشید و با اسکن کردن تصویر با الگوی تغییر مقیاس داده شده، بتوان یک شی را با وجود تغییر در سایز آن، پیدا کرد. امکان دیگر، انتخاب مکان احتمالی شی و جستجوی اطراف آن مکان، برای یافتن شی است. با استفاده از این روش و تکرار این عملیات، می‌توان به هرم تصویری دست پیدا کرد که در شکل ۱–۷ نمایش داده شده است[۱۷].



شکل 1–7 هرم تصویری [۱۷]

همانطور که شکل ۱–۷ نشان می‌دهد، این روش تصویر نمونه را به صورت بالا به پایین از پایین‌ترین رزولوشن تا بالاترین رزولوشن با تصویر داده شده مطابقت می‌دهد که هر سطح بیانگر یک مقیاس می‌باشد.

[1]- Template Matching

- **تشخیص رنگ:**

یکی از روش‌هایی که برای تشخیص اشیاء استفاده می‌شود، تشخیص رنگ است. به این صورت که تمام پیکسل‌های تصویر، بررسی می‌شوند تا زمانی که رنگ مورد نظر یافت شود. خروجی الگوریتم که فیلتر شده تصویر اولیه می‌باشد، تصویری باینری[۱] است که رنگ‌های مورد نظر ما به صورت سفید رنگ و رنگ‌های دیگر به صورت مشکی نشان داده می‌شوند.

از معایب روش مذکور می توان به حساسیت نسبت به تغییر نور اشاره کرد بدین نحو با تغییر نور محیط، رنگ تشخیص داده شده تغییر می کند و لذا در عمل، نیازمند در نظر گرفتن آستانه‌ای جهت تشخیص رنگ می‌باشد به علاوه در صورتی که در محیط، رنگ دیگری همرنگ با رنگ جسم وجود داشته باشد، این باعث خروجی اشتباه از طرف ربات می‌شود و بنابراین مکان‌یابی را با مشکل مواجه می‌سازد. در شکل 1–8 ورودی و خروجی الگوریتم تشخیص رنگ نمایش داده شده است.

ب- تصویر خروجی الگوریتم الف- تصویر اصلی دریافت شده از دوربین

شکل 1–8 خروجی الگوریتم تشخیص رنگ که به صورت صحیح رنگ‌های نارنجی را مشخص کرده است [۱۷]

- **تشخیص و مطابقت ویژگی[۲]:**

در پردازش تصویر، ویژگی[۳] مشخص کننده چیزی است که قابل توجه باشد، چیزی که بتوان بواسطه آن اطلاعاتی را از تصویر دریافت کرد. برای مثال نقاط کلیدی(گوشه‌ها) که توسط نحوه قرار گیری نقاط همسایگی تعریف می‌شود از این دست ویژگی ها است. مثال های دیگر می‌توان به لبه ها و خطوط اشاره کرد. شکل 1–9، ویژگی های تشخیص داده شده یک تصویر را نشان می‌دهد که در آغاز، تصویری که شامل سوژه مورد نظر می‌باشد در نظر گرفته می‌شود و الگوریتم تشخیص ویژگی بر روی آن پیاده

---

۱ Binary Image

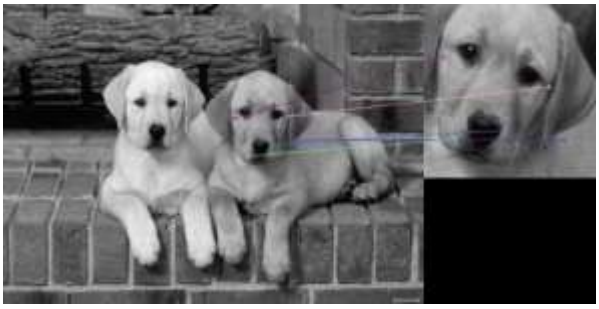
۲ Feature Detection and Matching

۳ Feature

سازی می‌شود سپس الگوریتم تلاش می‌کند ویژگی‌های مد نظر را پیدا کند و آن‌ها را منطبق کند. به این مرحله، شرح ویژگی یا توصیف ویژگی گفته می‌شود. در این مرحله الگوریتم سعی می‌کند ویژگی‌هایی که یافت شده اند را به صورت برداری[۱] و فشرده ذخیره کند. بعد از آن، فرآیند مشابهی بر روی تصویر دوم نیز پیاده‌سازی می‌شود تا ویژگی‌های تصویر دوم نیز بدست آید. در آخر الگوریتمی برای تطبیق ویژگی‌های دو تصویر به کار گرفته می‌شود.

الف

ب



پ

شکل 1–9 الف) نمایش ویژگی‌های یافت شده تصویر با دایره‌های رنگی ب)ویژگی‌های یافت شده در تصویر نمونه پ) تطابق صحیح ویژگی‌های تصاویر الف و ب [۱۷]

روش‌های مختلفی برای تشخیص ویژگی وجود دارد که الگوریتم‌های زیر، کاربردی‌ترین آنها هستند[۱۷]:

1- تبدیل ویژگی مستقل از مقیاس (SIFT)[۲]

2- ویژگی‌های مقاوم تسریع شده (SURF)[۳]

3- نقاط باینری مقیاس‌پذیر مقاوم (BRISK)[۴]

۱- Vector

۲- Scale-invariant Feature Transform

۳- Speeded-up Robust Features

۴- Binary Robust Invariant Scalable Keypoints

4- ویژگی‌های باینری بنیادی مستقل مقاوم (BRIEF)[۱]

5- ویژگی‌های تسریع شده از آزمایش قطعه‌ای (FAST)[۲]

6- FAST جهت‌دار و BREIF چرخیده (ORB)[۳]

این الگوریتم‌ها به خوبی در کتابخانه بینایی ماشین OpenCV پیاده سازی شده اند و نتایج بهتری نسبت به تطابق الگو و تشخیص رنگ ارائه می‌دهند[۱۸].

در انتخاب یک الگوریتم برای استخراج ویژگی به چند چالش اصلی باید توجه کرد که می‌توان به مواردی همچون: مقاوم در برابر تغییر مقیاس، مقاوم در برابر چرخش، مقاوم در برابر نویز، مقاوم در برابر کشیدگی[۴]، مقاوم در برابر تغییرات روشنایی اشاره کرد.

3–1 اهمیت موضوع تحقیق و ضرورت انجام آن:

از آنجایی که کنترل پهپاد در نقاط دوردست و به خصوص در زمان بروز حوادث غیرمترقبه با محدودیت‌هایی مواجه خواهد بود و کنترل پهپاد از فاصله دور امری مشکل و در مواردی غیرقابل انجام می باشد، ضرورت ایجاد سیستم کنترلی خودمختار برای انجام عملیات های امداد و نجات و تحویل بسته‌های امدادی به وضوح غیر قابل انکار می باشد. الگوریتم‌های پیشنهادی به جز اینکه در تشخیص اشیا کاربران را کمک می‌کنند، برای تهیه نقشه‌های بزرگ هوایی نیز اصلی‌ترین ابزار می‌باشند. چسباندن عکس‌های بزرگ به صورت دستی برای کاربر بسیار زمان‌بر و خسته‌کننده می‌باشد به همین دلیل خودکاری سازی چسباندن تصاویر دریافتی بسیار کاربردی و مهم می‌باشد. تعدادی از سازمان‌های منتفع از این پژوهش در زیر آورده شده اند:

1- امکان استفاده در حوادث غیر مترقبه توسط هلال احمر جهت شناسایی مصدومان و حمل بسته های امدادی

2- انجام عملیات بازرسی در صنایع مختلف و بازرسی جاده ها توسط نیروی انتظامی

3- نقشه برداری هوایی در مزارع، معادن، ساختمان سازی، لوله کشی و ... در کاربردهای وزارت جهاد کشاورزی

---

[۱]- Binary Robust Independent Elementary Features

[۲]- Features from Accelerated Segment Test

[۳]- Oriented FAST and Rotated BRIEF

[۴] Shear invariant

1–4 جنبه جدید بودن و نوآوری در پژوهش:

استفاده از الگوریتم‌های پردازش تصویر بر روی پهپادها و ترکیب الگوریتم‌های تطبیق الگو به منظور شناسایی بهتر، امکان شناسایی اهداف و اجسام را میسر می‌سازد که این امر باعث دستیابی بهتر به نقاط دوردست خواهد شد که امکان کنترل برای اپراتور فراهم نیست. با توجه به اینکه GPS های غیر نظامی در تعیین موقعیت همواره دارای خطا می‌باشند(تا یک متر) مخصوصا در مواقعی که مولتی روتور ساکن است، خطای GPS افزایش بیشتری پیدا می‌کند. در این شرایط، بینایی ماشین و کنترل ربات بر اساس بینایی ماشین می‌تواند نقش به سزایی در انجام ماموریت‌های پهپادها ایفا کند. در این پژوهش سعی شده تعدادی از الگوریتم‌های پرکاربرد در بینایی ماشین بر روی ربات‌های پرنده آزمایش و نتایج آن با یکدیگر مقایسه شود.

1–5 اهداف پژوهش:

1–5–1 هدف اصلی:

پیاده سازی عملی الکوریتم‌های پردازش تصویر و مقایسه سرعت و دقت الگوریتم‌ها با یکدیگر اصلی‌ترین هدف این پژوهش می‌باشد.

1–5–2 فرضیات پژوهش:

با توجه به اینکه ویدئو، دنباله‌ای از تصویر است، به همین دلیل، در عوضِ تحلیل ویدئویی، تصاویر تحلیل خواهند شد. تصاویر برای هر الگوریتم ثابت خواهند بود تا مقایسه انجام شده، ورودی‌های یکسانی داشته باشند.

1–5–3 محدودیت و پیش‌فرض‌های پژوهش:

- **پیش فرض‌های مسئله:**
  - ✓ پهپاد یک جسم صلب است.
  - ✓ دوربین به صورت صلب به پهپاد متصل شده و جابجایی مستقل ندارد.
  - ✓ سرعت حرکت پهپاد حداکثر تا 5 متربرثانیه در نظر گرفته خواهد شد.
  - ✓ عوامل فیزیکی مانند سرعت ربات و نور در نظر گرفته نخواهند شد.
  - ✓ تصاویر مورد تحلیل برای تمام الگوریتم‌ها ثابت خواهد بود.

- **محدودیت‌ها:**
  - ✓ در دسترس نبودن کارت کپچر HDMI برای استفاده از سیستم ارسال تصویر دیجیتال
  - ✓ نویز پذیر بودن تصاویر دریافتی از ارسال تصویر آنالوگ
  - ✓ کاهش کیفیت و سرعت تصاویر با ارسال تصاویر تحت شبکه وای-فای

2- فصل دوم: مروری بر ادبیات و پیشینه تحقیق

2–1 مقدمه:

رباتیک در سال‌های اخیر با سرمایه‌گذاری مناسب پیشرفت‌های بسیاری زیادی داشته است. در این بین ربات‌های پرنده نیز مورد استفاده بسیاری قرار گرفتند. یکی از بزرگترین مشکلات پیش رو برای این ربات‌ها، خودکار کردن آنها و کاهش عملکرد مستقیم اوپراتور بر روی ربات‌هاست.روش‌های مختلفی برای تشخیص اشیا معرفی شده‌اند برای بررسی بهتر، مفاهیمی در زمینه پهپادها، بینایی ماشین و روش‌های تشخیص اشیا و پژوهش‌های پیشین، ارائه خواهد گردید. هدف از این کار نقد و بررسی روش‌های انجام آزمایش به منظور فراهم سازی مقدمات آزمایش می‌باشد.

2–2 ادبیات پژوهش:

در این بخش مفاهیم پایه تحقیق شامل انواع پهپادها، روش‌های تشخیص ویژگی در تصاویر و روش‌های تطابق ویژگی در تصاویر به تفصیل بیان خواهند شد. برای این منظور در ابتدا، دلایل استفاده از پهپادها مورد بررسی قرار می‌گیرد و سپس کاربردهای تشخیص اشیا شرح داده خواهد شد.

پهپاد هایی که از راه دور کنترل می‌شوند به تعداد زیاد در انواع ماموریت‌ها مورد استفاده قرار می‌گیرند. از جمله این ماموریت‌ها می توان به امداد و نجات[۱۹]، جستجو و اکتشاف[۲۰]، کشاورزی[۲۱]، نظارت بر چاه‌های نفت[۲۲]، آزمایشات علمی[۲۳] و نقشه برداری محیطی[۲۴] اشاره کرد. لزوم ایجاد ارتباط با پهپاد، تنوع سنسورها و روش‌های کنترل منجر به تولید پهپادها در اشکال و چینش و خصوصیات مختلف شده است. مرسوم‌ترین نوع پهپادها نوع بال ثابت می‌باشد. از مزایای پهپادهای بال ثابت می‌توان به ساختار ساده‌تر، سرعت زیاد پرواز و مداومت پروازی بیشتر نسبت به مولتی روتورها، اشاره کرد. پهپادهای بال ثابت برای برخواستن از زمین و فرود نیازمند فضای زیادی هستند که از مزایای مولتی روتورها می توان به عدم نیاز به فضای زیاد برای برخواستن و فرود آمدن، قدرت مانور بسیار بالا و قدرت بیشتر در تصویربرداری هوایی اشاره کرد. در اکثر سیستم‌های خودکار پرواز برخواستن از زمین بصورت خودکار انجام می‌پذیرد. انجام این عملیات ریسک چندانی را به همراه ندارد. همانطور که زمین جسمی سلب در برابر پهپاد به شمار می‌رود، در عملیات فرود بعلت برخورد به زمین ریسک بسیار بالاست در مواردی که بسته امدادی باید در مقصد تحویل شود نیز این خطر وجود دارد از این رو، دقت در انجام عملیات فرود و تشخیص محل فرود یا رها کردن بسته باید بسیار بالا باشد. عمل تشخیص باید در زمان کم و فضای کم انجام پذیرد. به همین منظور استفاده از روش‌های تشخیصی و استفاده از حسگرهای قوی و سریع و همچنین کنترل دقیق در این عملیات به وضوح مشاهده می‌شود. عوامل بسیاری در این فرآیند نقش دارند مانند: نوع محیط(محیط بسته یا محیط باز)، نوع پوشش محیط، عوامل جوی مانند باد و باران و ...

استفاده از دوربین بر روی پهپادها یکی از تکنیک‌های مرسوم و پرکاربرد به منظور درک بهتر محیط، مسیریابی و همچنین تخمین موقعیت و جهت پهپاد، می‌باشد. تعدادی از کاربردهای تشخیص اشیا را می‌توان به صورت زیر نام برد:

- ✓ تشخیص اشیا خاص در تصویر

- ✓ نظارت بصری[1]
- ✓ تشخیص عابر پیاده[2]
- ✓ تشخیص ناهنجاری[3]
- ✓ شمارش اشیا
- ✓ تشخیص چهره
- ✓ خودروهای خودران

در این پژوهش ، تمرکز بر استفاده از الگوریتم‌های تشخیص و استخراج ویژگی، به منظور تشخیص شی معلوم در تصویر دریافتی از ربات انجام خواهد شد. از راه حل‌های مورد استفاده برای این منظور، می‌توان به الگوریتم‌های SURF، SIFT، ORB، FAST و ... اشاره کرد. این الگوریتم ها در فصل 3 به تفصیل توضیح داده خواهند شد.

2–3 سوابق تحقیق:

در این بخش سعی بر آن است تا پژوهش‌های انجام شده، و پیشینه و سوابق پژوهش مورد مطالعه قرار گرفته، و تحت عنوان سوابق تحقیق ارائه گردند.

2–3–1 بررسی سوابق پژوهش:

طرح‌های اولیه این نوع پرنده‌ها در سال 1920 توسط اومیشن[4] فرانسوی و همکارانش طراحی و ساخته شد. این طراحی‌ها هیچگاه توجه عموم را جلب نکرد حتی در صنایع نظامی. با وجود تولید انبوه نشدن طراحی های اومیشن، هیچگاه مزایای مولتی روتورها انکار نشد.

در دانشگاه صنعتی پراگ در جمهوری چک یک تیم با سرپرستی کراژنیک[5] پلتفورم یک ربات به نام Ar.Drone را برای تحقیقات و آموزش رباتیک توسعه داد [۲۵] در این پژوهش روش‌هایی برای پایداری موقعیت توسط بینایی ربات معرفی شده است. همچنین با استفاده از الگوی منحصر به فرد رنگی بر روی ربات زمینی موقعیت پهپاد پایدار شده است. در پژوهش انجام شده این

---

[1]- Video Surveillance

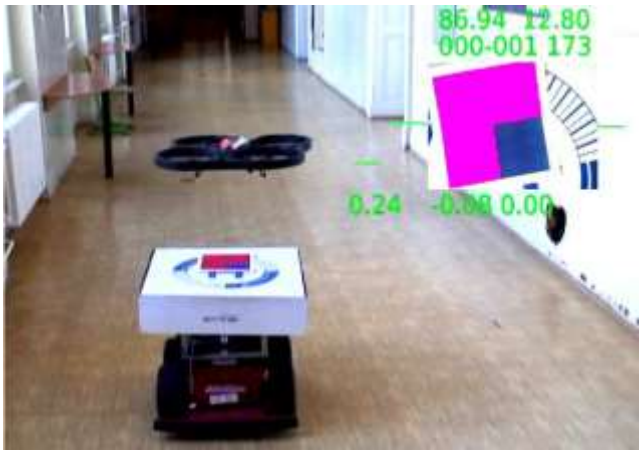
[2]- Pedestrian Detection

[3]- Anomaly detection

[4]- Étienne Oehmichen

[5]- Krajník

نتیجه حاصل شده است که برای تعقیب جسم ارتفاع پروازی بالاتر از 1/5 متر مناسب می باشد. همچنین در این تحقیق از الگوریتم تطبیق الگو SURF به منظور موقعیت یابی و مسیریابی از طریق دوربین جلو در پهپاد استفاده شده است. در این روش به کمک اپراتور مسیر حرکت از قبل آموزش داده می شود. سپس در مراحل بعدی پهپاد با استفاده از الگوریتم تطبیق الگو SURF مسیر را تشخیص داده و در مسیر خود حرکت می کند.

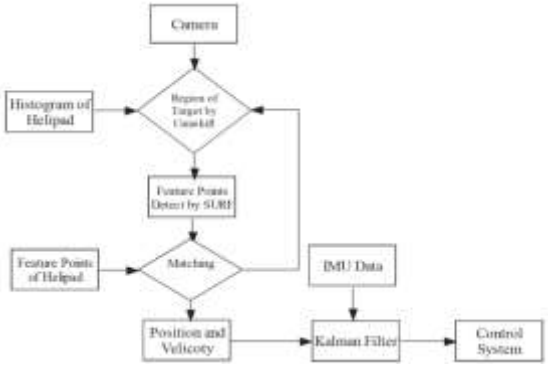


شکل 2–1 تعیقب محل فرود متحرک رنگی [۲۶]

ژائو[۱] و همکارش (2013) در پژوهشی از الگوریتم بهینه‌شده SURF استفاده کردند[۲۷]. این روش از سه مرحله تشکیل شده است. در مرحله اول با استفاده از الگوریتم Camshift محل هدف بر روی تصویر شناسایی می‌شود و سپس در مرحله دوم با استفاده از الگوریتم SURF نقاط ویژگی استخراج شده و در مرحله سوم با الگوی هدف تطبیق داده می‌شود. الگوریتم CamShift یک الگوریتم تکرار شونده مبتنی بر رنگ می‌باشد که با استفاده از اطلاعات رنگ شیئ مورد نظر در تصویر، به منظور دنبال کردن شی استفاده می‌شود. این الگوریتم به صورت مرتب اجرا می‌شود تا به درستی به تشخیص حرکت بپردازد. در این تحقیق بدلیل اینکه Camshift به اطلاعات رنگ حساس می باشد، ابتدا تصویر از فضای رنگ RGB به HSV تغییر می کند. در این پژوهش از رنگ سبز برای H استفاده شده است. الگوریتم Camshift از Mean-Shift استفاده می کند که در ابتدا مرکز هدف را تشخیص می‌دهد و سپس مقیاس و دوران هدف را محاسبه می کند. در شکل 2–2 و شکل 2–3 روش پیشنهادی و خروجی پژوهش ژائو آمده است.

[۱]- Zhao

شکل 2–3 رد یابی محل فرود [۲۷]



شکل 2–2 فلوچارت عملیات فرود خودکار [۲۷]

در سال 2014، انگل[۱] و تیم تحقیقاتیش در دانشگاه مونیخ بر روی پردازش تصویر و سیستم ناوبری بر اساس تصویر و شیوه مکان‌یابی و نقشه برداری همزمان(SLAM)[۲] مطالعاتی انجام دادند[۲۸]. رافائلو دآندره[۳] با استفاده از سیستمهای ویدئو کپچر مادون قرمز مانورهای بسیار مشکلی را در رباتهای پرنده در سالن ورزشی به نمایش گذاشت[۲۹] مانورهایی که توسط انسان غیر قابل اجرا هستند. این سخت افزارها بسیار هزینه بر هستند و باید در نقاط خاصی ثابت شوند تا عملکرد مناسب خود را داشته باشند که پیاده سازی این سیستم در فضای باز، علاوه بر پر هزینه بودن زمان‌بر نیز می‌باشد.

هریک[۴] و همکاران در سال 2015 در تحقیقی با استفاده از تشخیص رنگ و الگو، نشانگر نصب شده بر روی ربات زمینی را شناسایی کردند. در این تحقیق دوربینی زیر ربات پرنده‌ای نصب شد و با استفاده از الگوریتم تخمین موقعیت، توانست موقعیت هدف را بدست بیاورد[۳۰]. در این پژوهش از کتابخانه OpenCV برای تشخیص نشانگر استفاده شد.

یانگ[۵] و همکاران (2014) در تحقیقی از الگوریتم تطبیق الگو ORB برای تشخیص هدف استفاده کردند [۳۱]. در این پژوهش که با استفاده از PTAM[۶] موقعیت نسبی پهپاد تخمین زده می‌شود نقاط کلیدی برای تشخیص و تشکیل نقشه به دست می‌آید.

مرتز[۷] و همکارانش در یک تحقیق، از یک بالگرد به همراه یک برد کنترل‌کننده برخط و همچنین یک دوربین استفاده نموده‌اند. در این روش بر خلاف پژوهش‌های دیگر از سنسورهای خارجی مانند GPS استفاده نشده است. دوربین استفاده شده در

۱- Engel

۲- Simultaneous Localization and Mapping

۳- Raffaello D’Andrea

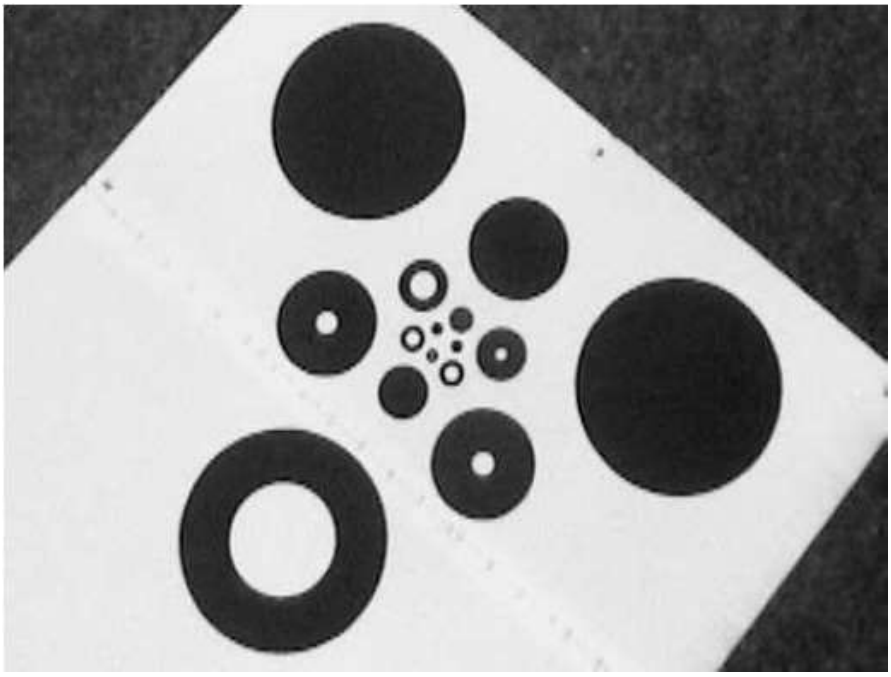
۴- Harik

۵- Yang

۶- Parallel Tracking and Mapping

۷- Merz

این سیستم از دو درجه آزادی[1] بهرمند می باشد و با استفاده از جابجایی آن هدف شناسایی خواهد شد. در این پژوهش از یک الگوی خاص برای فرود استفاده شده است که این الگوی ارائه شده برای تشخیص یک هدف ساده بوده ولی شناسایی آن سریع بوده است. استفاده از دوربین با دو درجه آزادی این امکان را ارائه می دهد که تغییرات رفتاری بالگرد در تصویر تاثیری نداشته باشد و همواره تصویر در زاویه مشخص شده نسبت به محل فرود ثابت بماند. موقعیت بالگرد و ارتفاع بالگرد با استفاده از تصویر و نشانگر محل فرود تخمین زده شده است. به منظور نمایش کارایی سیستم بینایی و کیفیت فرود، نتایج تست این روش بصورت واقعی و شبیه‌سازی ارائه شده است. در این روش از طراحی خاصی برای نشانگر محل فرود به منظور تشخیص سریع هدف، تخمین موقیت دقیق در فواصل طولانی و نزدیک، مستقل از مقیاس و کمینه کردن عدم تقارن محل فرود، استفاده شده است. بهمین منظور نشانگر از دایره‌های سیاه بر روی پس زمینه سفید با چینش خاص استفاده شده است. همانطور که در شکل 2–4 مشاهده می کنید این دوایر بصورت یک مثلث در کنار هم چیده شده‌اند و با بدست آوردن مراکز این دوایر موقعیت و زاویه پهپاد نسبت به نشانگر محاسبه می‌شود. در این روش از الگوریتم‌های محدوده یابی و تشخیص دایره و بیضی استفاده شده است و پس از یافتن مختصات مرکز آنها مختصات نسبی کل نسبت به پهپاد محاسبه می‌شود.



شکل 2–4 نشانگر محل فرود با استفاده از ترکیب دایره های مشکی و سفید [۳۲]

[1]- Pan-Tilting Camera

جدول زیر مقایسه مزایا و معایب روش‌های به کار گرفته شده را نشان می‌دهد:

جدول 2–1 – جمع بندی پژوهش های بررسی شده

ردیف	نویسنده	عملکرد	مزایا/معایب
۱	Polvara و همکاران [۳۳](۲۰۱۸)	با استفاده از گذاشتن و نشانگر ArUco بر روی یک قایق روی آب و تشخیص آن و همچنین محاسبه موقعیت نسبی پهپاد با استفاده از EKF	استفاده از EKF و ثبات در شرایط ناپایدار جوی محدود به نشانگر
۲	Guevarra و همکاران [۳۴](۲۰۱۸)	اهداف اصلی تشخیص هدف با استفاده از پردازش تصویر و تشخیص رنگ و تخمین ارتفاع پهپاد می باشد.	عدم استفاده از GPS حساس بودن رنگ به تغییرات روشنایی
۳	Hoang و همکاران [۳۵](۲۰۱۷)	استفاده از الگوریتم یادگیری نظارت شده برای تشخیص هدف	
۴	Araar و همکاران [۳۶](۲۰۱۶)	استفاده از نشانگرهای ArUco	سرعت بالای تشخیص و تخمین زاویه و موقعیت – محدودیت نشانگر
۵	Yang و همکاران [۳۷](۲۰۱۵)	استفاده از ترکیب IMU و SRUKF و همچنین استفاده از نشانگر ArUco	استفاده از SRUKF حساس بودن به نویز
۶	Yang و همکاران (۲۰۱۴) [۳۱].	در این پژوهش که با استفاده از PTAM موقعیت نسبی پهپاد تخمین زده می شود نقاط کلیدی برای تشخیص و تشکیل نقشه به دست می آید.	استفاده از الگوریتم تطبیق الگو ORB
۷	Barták و همکاران [۳۸](۲۰۱۴)	استفاده از دو دایره رنگی و تشخیص الگو و رنگ آن ها	سهولت پیاده سازی و تشخیص حساس بودن رنگ به تغییرات روشنایی
۸	Engel و تیم تحقیقاتیش [۳۹](۲۰۱۴)	استفاده از الگوریتم تطبیق الگو FAST و استفاده از EKF	عدم استفاده از GPS و نشانگر اضافه – استفاده از KF
۹	Jae Keun Lee و همکاران (۲۰۱۴)[۴۰]	استفاده از تشخیص نشانگر ArUco و تشخیص رنگ	دقت بالا تشخیص موقعیت حساس بودن رنگ به تغییرات روشنایی
۱۰	Tiago Gomes Carreira (۲۰۱۳)[۴۱]	استفاده از نشانگرهای زیاد ArUco	دقت بالا تشخیص موقعیت استفاده از نشانگر های زیاد
۱۱	Krajnık و همکاران (۲۰۱۲) [۲۶]	توسعه پلتفورم یک ربات برای تحقیقات و آموزش رباتیک و استفاده از روش تشخیص رنگ و SURF برای تشخیص هدف	استفاده الگوریتم تطبیق الگو SURF به منظور موقعیت یابی و مسیریابی
۱۲	Wenzel و همکاران [۴۲](۲۰۱۰)	استفاده از فرستنده و گیرنده امواج مادون قرمز	هزینه بر و خطاپذیر در صورت ایجاد نویز
۱۳	Saripalli و همکارش [۴۳](۲۰۰۹)	الگوریتم تطبیق الگو برای پیدا کردن هدف و ترکیب آن با الگوریتم مسیریابی، استفاده از از یک کنترل کننده خطی مبنی بر مدل سینماتیک بالگرد برای مسیریابی و فرود	دقت بالا تعیین مسیر از پیش تعریف شده
۱۴	Merz و همکاران (۲۰۰۶) [۳۲]	تشخیص رنگ و الگوی تصویر	تشخیص سریع الگو الگوی ساده

3- فصل سوم: معرفی و ارائه روش‌های پیشنهادی

3–1 مقدمه:

با مطالعات و بررسی‌های انجام گرفته بر روی ویژگی و خصوصیات روش های پیشین و در آنچه تاکنون ارائه گردید؛ مشخص شد در این دسته از پژوهش ها به صورت کامل از الگوریتم های تشخیص ویژگی استفاده نشده است. دستیابی به اطلاعات تصویر با استفاده از دوربین‌ها برای ربات‌ها، امری حیاتی می‌باشد. برای بدست آورد اطلاعاتی که بتوان بوسیله آن‌ها اطلاعات دنیای واقعی را دریافت کرد با استفاده از الگوریتم‌های بینایی ماشین میسر می‌شود[۴۴]. در شکل 3–2 روند تشخیص شی توسط رایانه نمایش داده شده است. تطبیق ویژگی های تصویر یکی از مهم‌ترین مباحث در زمینه ثبت تصویر[۱] و موزائیک[۲] است. با توجه به نیاز پردازش سریع و آنی بر روی پهپاد، انتخاب مناسب‌ترین الگوریتم استخراج ویژگی حائز اهمیت می‌باشد ازین رو باید از روش‌های سریع‌تر و کارامدتر استفاده کرد.

3–2 ویژگی[۳]:

تعریف کلی یا دقیقی از ویژگی وجود ندارد و تعریف دقیق ویژگی معمولا نسبت به کاربری، متفاوت است. با این وجود ویژگی را می‌توان از نظر کاربری، نقطه‌ای جالب[۴] در تصویر عنوان کرد. ویژگی‌ها نقطه شروع در اکثر الگوریتم‌های پردازش تصویر می‌باشند. به دلیل اینکه الگوریتم‌های پردازش تصویر از ویژگی‌های استخراج شده بهره می‌برند به همین دلیل، دقت الگوریتم‌ها به شناساگرهای ویژگی[۵] وابسته اند. پارامتر تعیین کننده در شناساگرهای ویژگی، قابلیت تکرارپذیری[۶] آنها هستند که آیا در دو یا چند تصویر جدا از یک شی، ویژگی‌های مشابهی شناسایی می‌شود یا خیر[۴۵].

تشخیص ویژگی(شناسایی)، عملگری سطح پایین در پردازش تصویر است و در مراحل اولیه پردازشِ یک تصویر از آن استفاده می‌شود و تک تک پیکسل‌های موجود در تصویر را بررسی می‌کند. پیشنیاز الگوریتم‌های شناساگر ویژگی معمولا فیلتر گاوسی است که تصویر را صاف[۷] می‌کند که در بخش‌های آینده به آن پرداخته خواهد شد. در صورتی که الگوریتم شناساگر، از نظر پردازشی سنگین باشد یا زمان پردازش برای ما اهمیت داشته باشد، الگوریتمی با سطح بالاتر به کار گرفته می‌شود تا به شناساگر کمک کند تا فقط نقاطی خاص از تصویر را جستجو کند. بسیاری از الگوریتم‌های پردازش تصویر از تشخیص ویژگی به عنوان مرحله آغازین استفاده می‌کنند. انواع ویژگی را می‌توان به صورت زیر دسته بندی کرد:

---

[۱] Image Registration

[۲] Mosaic

[۳] Feature

[۴] Interesting Point

[۵] Feature Detector

[۶] Repeatability

[۷] Smooth

3–2–1 لبه ها[1]:

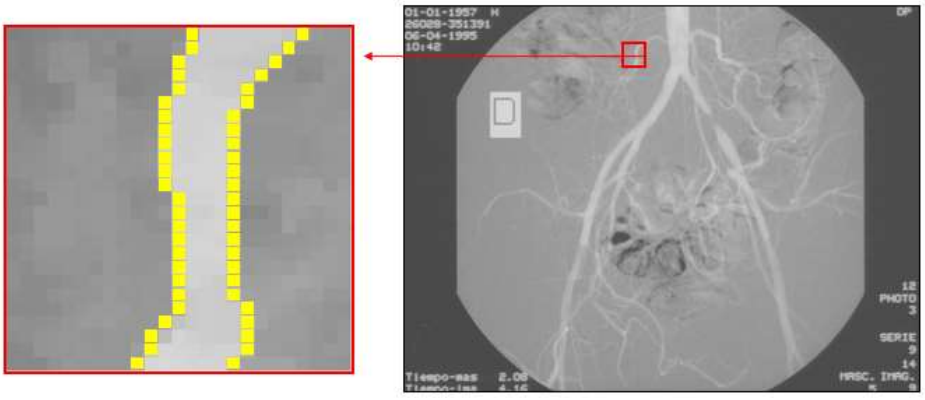
لبه‌ها می‌توانند به هر شکلی باشند. لبه‌ها در واقعیت به عنوان دسته‌ای از نقاط شناخته می‌شوند که تغییرات بالایی از نظر شدت نور داشته باشند. دلیل استفاده از این لبه‌های تیز اینست که فرض می‌شود تصویر ناپیوستگی پیدا کرده است و به دلایل زیر این ناپیوستگی بوجود آمده است:

- ✓ نا پیوستگی در عمق
- ✓ ناپیوستگی در سطح
- ✓ تغییرات در جنس اشیا

لبه‌هایی که در تصاویر به صورت طبیعی پیدا می‌شوند به صورت ایده آل دارای لبه پله‌ای نیستند و معمولا تحت تاثیر عوامل زیر هستند[۴۶]:

- ✓ تارشدگی تصویر به دلیل تمرکز[2] نامناسب دوربین
- ✓ تارشدگی بر اثر سایه مربوط به منبع نوری گسترده
- ✓ شی با سطح صاف یا ملایم

در سمت چپ تصویر پزشکی نشان داده شده در شکل 3–1یک لبه به نمایش درآمده است.



شکل 3–1 نمایش لبه در یک تصویر پزشکی[۴۷]

[1] Edges

[2]- Focus

3–2–2 گوشه ها[1]:

گوشه‌ها را می‌توان محل تقاطع دو لبه توصیف کرد[۴۸]. این گونه نقاط در اثر ناپیوستگی هندسی اشیا بوجود می‌آیند[۴۹]. الگوریتم-های گوشه یاب در ابتدا از لبه استفاده می‌کردند بعد برای یافتن تغییرات سریع در مسیر لبه تلاش می‌کردند. بعدها این الگوریتم‌ها توسعه پیدا کردند و عنصر لبه یابی برای یافتن گوشه حذف شد.

مشکل این ویژگی این است که ممکن است نقاطی را به عنوان گوشه شناسایی کند که در حقیقت گوشه نیست برای مثال یافتن نقطه‌ای سفید در زمینه مشکی.

3–2–3 جباب ها یا لکه ها[2]:

حباب‌ها یا لکه‌ها بر خلاف گوشه‌ها که نقطه‌ای عمل می‌کردند، به صورت منطقه‌ای خصوصیات یا ویژگی‌ها را شناسایی می‌کنند. با این وجود الگوریتم‌های توصیفگر حباب ممکن است یک نقطه را به عنوان حباب یا لکه بر اساس مرکز منطقه مورد نظر محاسبه کنند. الگوریتم‌های شناساگر حباب با هدف تشخیص مناطقی از تصویر که در خصوصیات مانند شدت نور یا رنگ متفاوت هستند، بوجود آمده‌اند. در حقیقت حباب‌ها مناطقی از یک تصویر هستند که خصوصیات ثابت یا نزدیک به یکدیگر دارند. در جدول 3–1 معروف‌ترین شناساگرهای ویژگی آورده شده‌اند[۵۰].

جدول 3–1 شناساگرهای ویژگی متداول و دسته بندی‌های آنها

شناساگر	لبه	گوشه	حباب
Canny	✓		
Sobel	✓		
Harris & Stephens / Plessey / Shi–Tomasi	✓	✓	
FAST		✓	✓
Laplacian of Gaussian		✓	✓
Difference of Gaussians		✓	✓
Determinant of Hessian		✓	✓

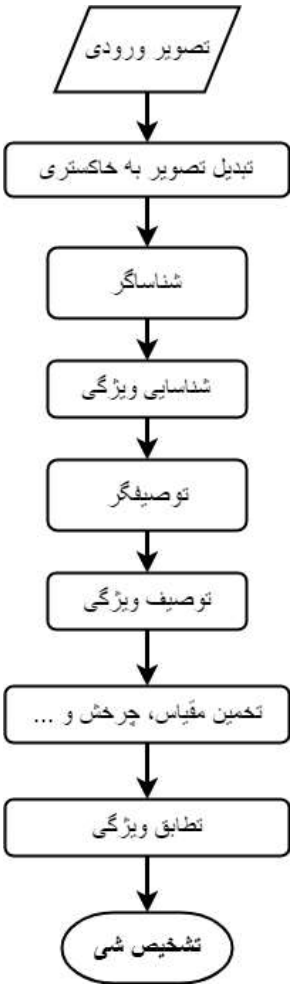
تصاویر بعد از اینکه توسط الگوریتم‌های شناساگر، شناسایی شدند باید برای انجام پردازش‌های بعدی آماده شوند در این مرحله الگوریتم‌های توصیفگر این ویژگی‌ها را آماده پردازش می‌کنند. این مرحله وظیفه دارد ویژگی‌های محلی را از نظر مقیاس، چرخش تخمین بزند[۵۱].

---

[1]- Corners

[2]- Blobs or Region of interest points

در شکل 3–2 فلوچارت عملیات تشخیص شی را می‌توان مشاهده کرد.



شکل 3–2 مراحل تشخیص شی

در جدول 3–2 نیز فهرست شناساگرها و توصیفگرهای پرکاربرد آورده شده است:

جدول 3–2 شناساگرها و توصیفگرهای متداول

الگوریتم	شناساگر	توصیفگر
SIFT	✓	✓
SURF	✓	✓
ORB	✓	✓
BRISK	✓	✓
FAST	✓	
KAZE	✓	✓
FREAK		✓
BRIEF		✓

3–3 معرفی روش‌های استخراج ویژگی و تطبیق الگو:

تطبیق تصویر یکی از فناوری‌های کلیدی در زمینه پردازش تصویر می‌باشد. محققان در این زمینه تحقیقات زیادی انجام داده اند. تمرکز اصلی در این فناوری بر روی دقت تطبیق و زمان تطبیق می‌باشد. تطبیق تصویر عموماً به دو قسمت تطبیق بر اساس همبستگی خاکستری[۱] و ویژگی تصویر[۲]،تقسیم می شود[۵۲]. تطبیق بر اساس همبستگی خاکستری روشی است که برای جستجو شی در تصویر استفاده می شود. هزینه محاسباتی این روش بسیار بالاست و همچنین نسبت به مقیاس‌پذیری و چرخش تصویر بسیار حساس است. برای مثال، در الگوریتم Harris به منظور تطبیق ویژگی‌های تصویر، اطلاعات نقاط گوشه‌ها استخراج می‌شود. سپس، با استفاده از یک ضریب همبستگی، منطبق‌ترین مختصات در تصویر را به منظور تطبیق بدست می‌آورد. ولی این روش نیز در مقابل مقیاس پذیری و چرخش تصویر حساس می باشد. در سال 2004 الگوریتم SIFT توسط لو[۳] معرفی شد[۵۳] و با استفاده از لاپلاس گوسی[۴] تا حدودی مشکل مقاومت در مقابل مقیاس و چرخش حل شد ولی بارِ محاسباتی سنگینی به همراه داشت و در کاربرد های پردازش آنی[۵] نمی‌توانست مورد استفاده قرار گیرد. در سال 2006، بی[۶] الگوریتم SURF را بر مبنی SIFT ارائه داد[۵۴] که هدف اصلی آن محاسبه توسط نقاط ویژگی موجک هار[۷] بود. استفاده از موجک هار به خوبی سرعت تطبیق تصاویر را افزایش می‌دهد. موجک هار ترتیبی از توابع مربعی است که در سال 1909 توسط آلفرد هار مطرح شد که ساده‌ترین نوع موجک می‌باشد. الگوریتم SURF نیز برای کاربرد های بلادرنگ کارایی چندانی از خود نشان نداده است در حالی که برای جنبه های دیگر مانند استخراج ویژگی و ساختار توصیف کننده[۸] نیز بهبودی حاصل نشده است.

برای تصاویری که بافت زیادی دارند روستن[۹] الگوریتم FAST را ارائه داد[۵۵] . با توجه به اینکه پردازش بلادرنگ در تطبیق تصاویر از اهمیت بالایی برخوردار می‌باشد، نقاط ویژگی با استفاده از الگوریتم تشخیص نقاط ویژگی در الگوریتم FAST استخراج خواهند شد و این نقاط با استفاده از عملگر لاپلاسی وزن داده شده[۱۰] بهینه‌سازی خواهد شد[۵۶]. همچنین با استفاده از توصیف کننده

۱- Gray correlation

۲- Image feature

۳- Lowe

۴- Laplace of Gaussian

۵- Real-Time

۶- Bay

۷- Haar wavelet

۸- Descriptor

۹- Rosten

۱۰- Weighted Laplace operator

SURF قابلیت مقاومت در برابر چرخش و مقیاس را حفظ کرده و به نتیجه دلخواه در تطبیق تصاویر بصورت بلادرنگ دست یافته می‌شود.

در اینجا به تعریف دو مفهوم چهارچوب فضا-مقیاس و هرم تصویری[۱] پرداخته خواهد شد.

3-3-1 تئوری فضا-مقیاس[۲]:

چهارچوبی برای نمایش چند مقیاسه سیگنال ها است که یک تئوری معمول برای بررسی داده‌های تصاویر در مقیاس‌های متفاوت می‌باشد به این صورت که تصویر را به صورت گروهی از تصاویر تک پارامتری صاف[۳]شده نمایش می‌دهد. اصلی‌ترین نوع تئوری فضا-مقیاس، فضا-مقیاس خطی یا گاوسی[۴] است چهار چوب فضا-مقیاس مربوطه، شامل تئوری عملگرهای مشتق‌گیر گاوسی می‌شود که مبنای عملگرهای مختلف بینایی ماشین می‌باشد. این چهارچوب به عملگرهای بینایی ماشین امکان این را می‌دهد که بتوانند مستقل از مقیاس عمل کنند. این کاربرد برای اشیاء در دنیای واقعی بسیار ضروری است چرا که ممکن است مقیاس اشیا متفاوت بوده و همچنین ممکن است فاصله بین شی و دوربین، نامشخص یا متغییر می‌باشد. مفهوم فضا-مقیاس می‌تواند بر روی سیگنال‌هایی با تعداد متغییرهای گوناگون اعمال شود ولی معمول‌ترین حالت استفاده از این چهارچوب، استفاده از آن برای تصاویر دو بعدی است. رابطه 3-1 بیان می‌کند که نمایش فضا-مقیاس خطی (گاوسی) تصویر $f(x,y)$ از کانولوشن تصویر f و گاوسین g که در رابطه 3-2 نمایش داده شده، بدست می‌آید[۵۷].

عملیات صاف کردن گاوسی[۵] وظیفه دارد لبه‌های تیز و ناپیوستگی‌ها را در یک تصویر کِدِرتر[۶] کند. این عملیات برای عملگرهایی که به نویز حساس هستند بسیار کمک‌کننده است. مقدار این کِدِر شدگی را پارامتر σ تعیین می‌کند. با افزایش مقدار σ میزان فیلتر شدگی نویزهای تصویر افزایش می‌یابد ولی از طرف دیگر ممکن است لبه‌های مهم در تصویر، حذف شوند که در عملکرد الگوریتم‌های لبه یاب اختلال ایجاد کند. از طرف دیگر مقدار پایین σ ممکن از باعث نویزی شدن بیش از حد تصویر شود[۵۸].

$$L(x,y;t) = g(x,y;t) * f(x,y) \qquad 3-1$$

$$g(x,y;t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/2t} \qquad 3-2$$

در رابطه 3-1 و در تابع L، کانولوشن فقط بر روی متغیرهای (x,y) اعمال می‌شود در حالی که پارامتر مقیاس t که بعد از ";" آمده بیان می‌کند عملیات در کدام مقیاس معین شده است. پارمتر مقیاس $t = \sigma^2$ مقدار اختلاف فیلترهای گاوسی است و محدوده

۱- Image Pyramid

۲- Scale-Space

۳- Smoothed

۴- Gaussian

۵- Gaussian smoothing operation

۶- blurring

آن $t = 0$ است که فیلتر g را بی اثر می کند و مقدار آن برابر $L(x, y; 0) = f(x, y)$ می‌شود به این معنا که با مقیاس $t = 0$ نمایش فضا-مقیاس آن برابر تصویر اصلی می‌شود که در شکل 3–3 قابل مشاهده است.

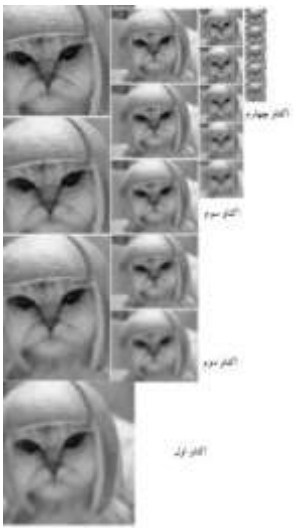
الف-نمایش فضا-مقیاس $L(x, y; t)$ تصویر f در مقیاس $t = 1$

ب-نمایش فضا-مقیاس $L(x, y; t)$ تصویر f در مقیاس $t = 0$

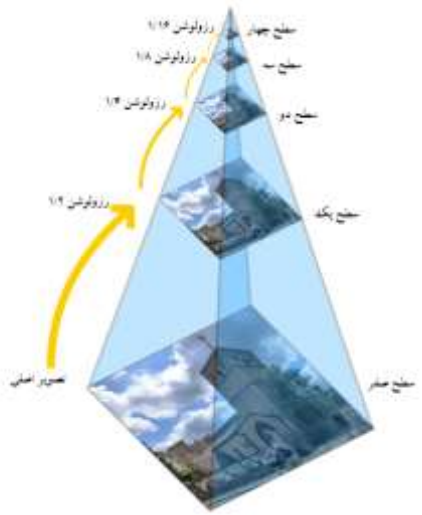
پ-نمایش فضا-مقیاس $L(x, y; t)$ تصویر f در مقیاس $t = 64$

ت-نمایش فضا-مقیاس $L(x, y; t)$ تصویر f در مقیاس $t = 8$

شکل 3–3 نمایش فضا مقیاس تصویر f در مقیاس‌های متفاوت[۵۹]



شکل 3–4 چهار مرحله کانولوشن گاوسی[۶۰]



شکل 3–5 نمایش هرم تصویری[۵۹]

3–3–2 الگوریتم SIFT[1]:

الگوریتم SIFT توسط Lowe در سال 2004 مطرح شد[۵۳] که مشکل چرخش تصاویر، تبدیل همگر[2]، شدت نور[3] و تغییرات نقطه دید[4] را حل کرد. الگوریتم SIFT از چهار مرحله تشکیل شده است. این الگوریتم هم یک شناساگر و هم یک توصیفگر است که در مرحله اول بیشینه فضا–مقیاس با استفاده از اختلاف گاوسین[5] (DOG) تخمین زده می‌شود، دوم نقاط کلیدی مکان‌یابی می‌شوند به این صورت که نقاط مورد نظر با حذف نقاطی با کنتراست پایین بدست می‌آیند، سوم، تعیین جهت‌گیری نقاط کلیدی با استفاده از گرادیان تصویر محلی می‌باشد و در نهایت، برای هر نقطه کلیدی، یک توصیف‌گر بر اساس گرادیان و جهت‌گیری تصاویر،در تصاویر محلی محاسبه می‌کند. الگوریتم SIFT، الگوریتم قدرتمندی است که نسبت به چرخش تصاویر، مقیاس شیء مورد نظر و تبدیل همگر مستقل عمل می‌کند با این وجود مشکل اصلی این الگوریتم این است که توان پردازشی بالایی را طلب می‌کند. در الگوریتم SIFT، تئوری فضا–مقیاس به این گونه عمل می‌کند که بعد از دریافت تصویر اصلی، در چند مرحله کدر می‌شوند پس از آن مقیاس تصویر نصف می‌شود و باز هم در چند مرحله کدر می‌شود و به همین صورت ادامه پیدا می‌کند. در شکل 3–4 تصویر اصلی، در چهار اکتاو بررسی شده و در هر اکتاو نیز چهار مرتبه کدر شده است. در شکل 3–5 هرم تصویری در چهار مرحله کوچک سازیِ مقیاس مشاهده می‌شود.

رابطه 3–3 اختلاف کانولوشن دو گاوسین در مقیاس‌های متفاوت را از تصویر $I(x,y)$ بدست می‌آورد.

$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) \qquad 3–3$$

در رابطه 3–3، D تصویر اختلاف گاوسین DOG را نشان می‌دهد. G بیانگر عملگر گاوسین می‌باشد و x و y مختصات مکانی هستند σ فاکتور مقیاس می‌باشد که مقدار صافی را تنظیم می‌کند. شکل 3–6 شماتیکی از فرآیند عملکرد SIFT نشان می‌دهد. در مرحله اول، از تصویر ورودی، هرم گاوسی تشکیل می‌شود بعد از آن اختلاف مراحل هرم به صورت دو به دو محاسبه می‌شود و پس از آن، با بدست آوردن اکسترمم‌های هرم اختلاف گاوسی، نقاط دارای ویژگی محاسبه می‌شوند. Lowe در مقاله خود تعداد اکتاوهای کدر شدگی را 5 سطح و تعداد سطوح مقیاس‌ها را 4 سطح در نظر گرفته است و $\sigma = 1.6$ قرار داده است.

---

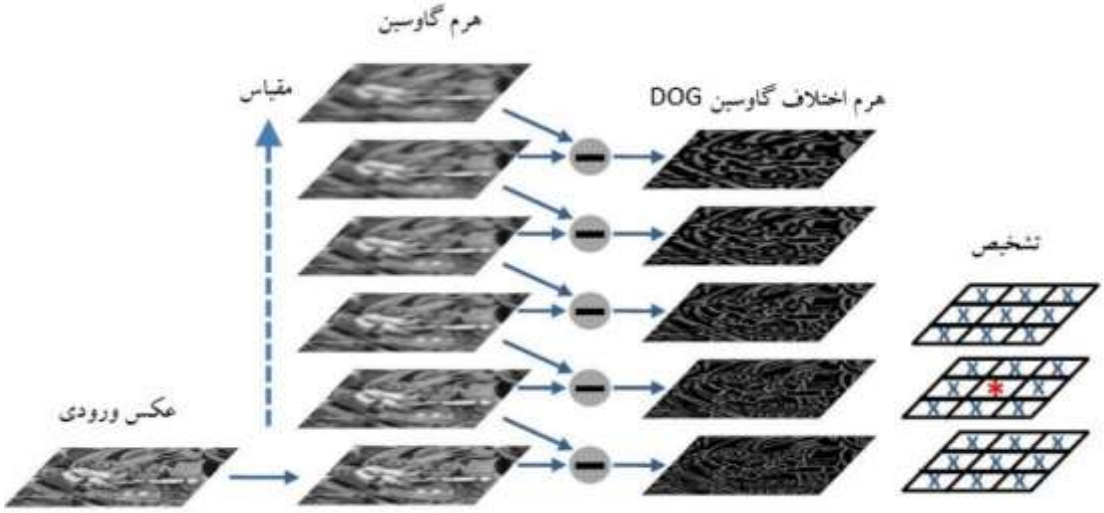
[1]- Scale Invariant Feature Transform

[2]- Affine

[3]- Intensity

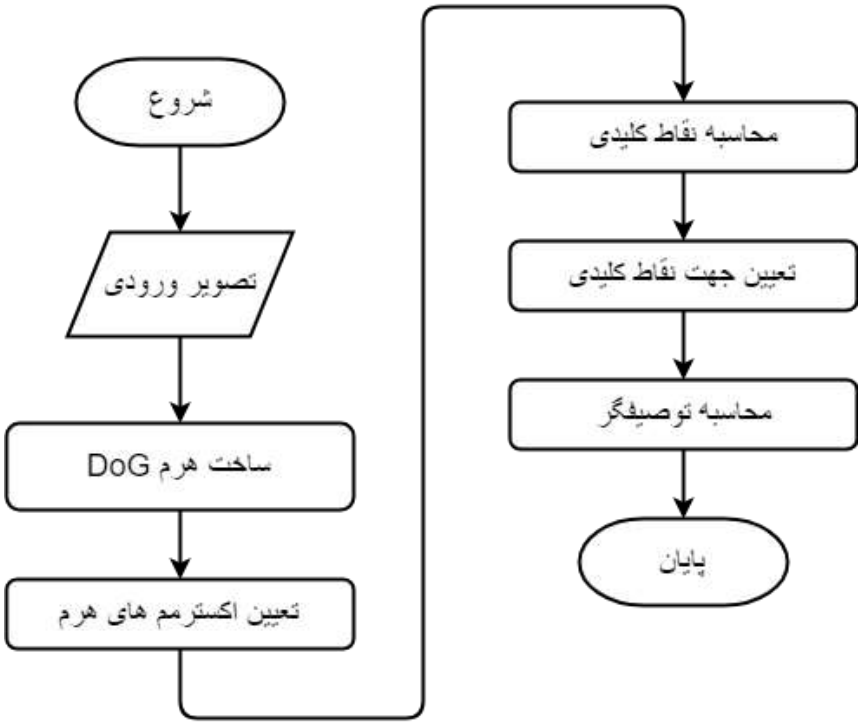
[4]- Viewpoint

[5]- Difference of Gaussian



شکل 3–6 نمایش دیاگرام الگوریتم SIFT با استفاده از هرم 6 لایه[۶۱]

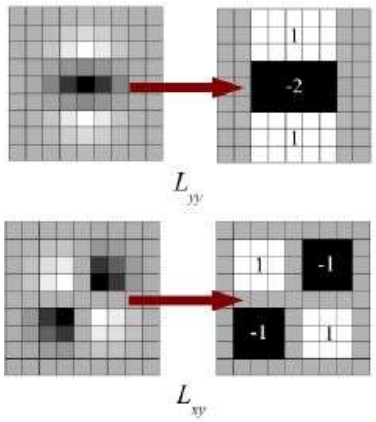
در شکل 3–7 فلوچارت الگوریتم SIFT نمایش داده شده است.



شکل 3–7 فلوچارت الگوریتم SIFT

3–3–3 الگوریتم SURF[1]:

این الگوریتم در سال 2006 توسط هربرت بِی[2] پیشنهاد داده شد[۱۶]. این الگوریتم نیز مانند SIFT، به آنالیز گاوسین در چهارچوب فضا–مقیاس تکیه دارد و هم یک شناساگر[3] و یک توصیفگر[4] نیز هست. الگوریتم SURF، به جای تخمین گاوسینِ کل عکس از الگوریتم تشخیص حباب[5] بر اساس ماتریس Hessian استفاده می‌کند که تصاویر درونی را استخراج می‌کند و سرعت تشخیص ویژگی‌های تصویر را افزایش می‌دهد[۶۲]. این روش، هم در کاهش سرعت پردازش تاثیر زیادی دارد هم در دقت عملیات. دترمینان ماتریس Hessian برای اندازه گیری تغییرات محلی حول نقطه‌ای که دترمینان بیشینه است استفاده می‌شود. دترمینان ماتریس Hessian، برای انتخاب مقیاس تصویر نیز مورد استفاده قرار می‌گیرد. ماتریس Hessian با فرض نقطه $P = (x, y)$ در تصویر I با مقیاس σ دررابطه 3–4 نشان داده شده است که $L_{ij}(p, \sigma)$ برابر است با کانولوشن مشتق مرتبه دوم گاوسین تصویر $I(x, y)$. درالگوریتم SURF، محققان یک گام فراتر رفته و لاپلاس گاوسین را با استفاده از فیلتر جعبه ای تخمین میزند شکل ۳–۸ تخمین فیلتر جعبه ای 9 در 9 را نشان می‌دهد. مزیت بزرگ این تخمین این است که کانولوشن یک فیلتر جعبه ای می‌تواند به راحتی توسط انتگرال تصاویر بدست بیاید و به طور موازی با اندازه‌های مختلف تصاویر نیز محاسبه شود. نقاط مورد نظر ممکن است در تصاویر با مقیاسهای مختلف پیدا شوند. در الگوریتم‌های شناساگر دیگر، چهار چوب فضا–مقیاس معمولاً به عنوان هرم تصویری شناخته می‌شود. تصاویر به صورت مداوم توسط فیلتر گاوسین، فیلتر می‌شوند بعد از آن دوباره از آنها نمونه گیری می‌شود تا به مرحله بعدی هرم تصویری برسد.



شکل 3–8 تخمین لاپلاس گاوسی با فیلتر جعبه ای

---

[1]- Scale Invariant Feature Transform

[2]- Herbert Bay

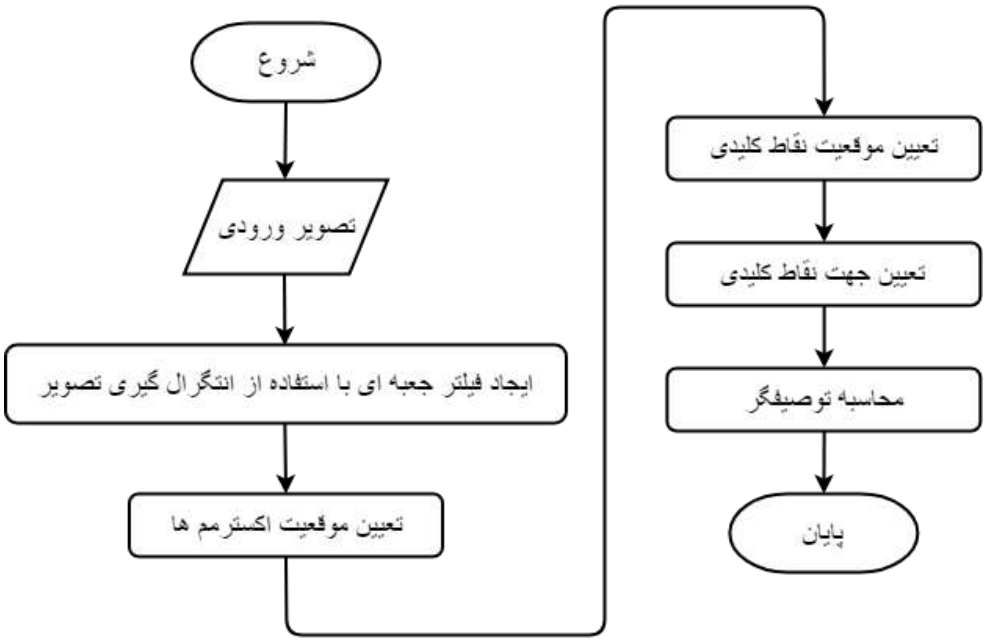
[3]- Detector

[4]- Descriptor

[5]- Blob detector

$$H(p,\sigma) = \begin{pmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{yx}(p,\sigma) & L_{yy}(p,\sigma) \end{pmatrix} \qquad 4\text{–}3$$

در شکل 3–9 فلوچارت الگوریتم SURF نمایش داده شده است.



شکل 3–9 فلوچارت الگوریتم SURF

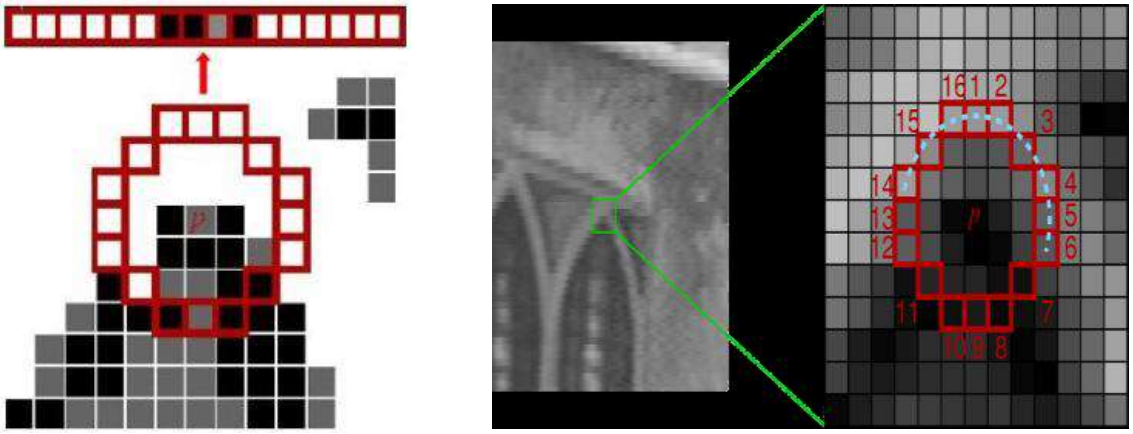
4–3–3 الگوریتم FAST:

این الگوریتم برای تشخیص گوشه‌ها در تصاویر مورد استفاده قرار می‌گیرد. این الگوریتم از یک دایره 16 پیکسلی استفاده می‌کند برای دسته‌بندی اینکه آیا نقطه مورد نظر گوشه می‌باشد یا خیر؟ همانطور که شکل 3–10-الف نشان می‌دهد فرض می‌شود پیکسل P با شدت نور I_p [1] معین شده که هر پیکسل از 1 تا 16 به صورت ساعت گرد شماره گذاری می‌شوند اول از همه برای افزایش سرعت پردازش، شدت نور پیکسل‌های 1، 5، 9 و 13 را با I_p مقایسه می‌کند و در صورتی که 3 عدد از این نقاط آستانه مورد نظر را داشته باشند، نقطه P به عنوان کاندید انتخاب می‌شود. در مقابل، در صورتی که این سه نقطه آستانه مورد نظر را نداشته باشند، نقطه P گوشه مناسبی نخواهد بود و در صورتی که 3 عدد ازاین نقاط، مقدار آستانه را داشتند تمام 12 پیکسل همجوار را نیز مورد بررسی قرار می‌دهد و به همین صورت این فرآیند برای تمام پیکسل‌های درون تصویر انجام می‌شود. این روش محدودیت‌های خاصی دارد برای مثال انتخاب پیکسل‌ها کارآمد نیست و ممکن است چند ویژگی [2] در فاصله نزدیکی از یکدیگر یافت شوند به همین دلیل رویکرد

[1] Intensity

[2] Feature

یادگیری ماشین[۱] به کار گرفته شد تا چنین مشکلاتی رفع شود. به این منظور مجموعه یادگیری[۲] ساخته شد که از نقطه P یک بردار با مقدار 16 پیکسل اطراف آن ذخیره می‌کند. که رابطه هر پیکسل با پیکسل کناری آن می‌تواند سه حالت تیره‌تر[۳]، مشابه[۴] و روشن‌تر[۵] باشد که با توجه به دستوراتی که به آن داده می‌شود، بردار ویژگی V به سه زیر مجموعه P_s، P_d و P_b تقسیم می‌شود. سپس یک الگوریتم طبقه‌بندی کننده[۶] اجرا می‌شود تا در نهایت نقطه‌ای را با توجه به آستانه آنتروپی[۷] به عنوان گوشه شناسایی کند[۶۳].



الف- نقطه مورد نظر پردازش شده و 16 پیکسل همجوار

ب-نمایش 16 پیکسل همجوار به صورت برداری

شکل 3–10 عملکرد الگوریتم FAST برای تشخیص گوشه‌ها

3–3–5 الگوریتم BRISK[۸]:

با وجود اینکه ویژگی‌های محلی که با استفاده از توصیفگرهای بر پایه بردار مانند SIFT و SURF و روش‌های مشابه به خوبی عمل می‌کنند در حالی که نسبت به چرخش، مقیاس، جابه‌جایی و تغییرات نقطه دید مستقل عمل می‌کنند، استفاده از آنها ممکن است کاملاً به صرفه نباشد مخصوصاً زمانی که با دستگاههایی کار می‌شود که قدرت پردازشی اندکی دارند و سیار هستند که هم از نظر پهنای باند و هم توان باطری محدودیت دارند. برای مواجه شدن با این چالش، انواعی از توصیفگرهای باینری معرفی شدند که BRISK نیز یکی از آنهاست. BRISK بر اساس FAST طراحی شده و به صورت کلی از سه بخش تشکیل شده است:

---

[۱] Machine Learning

[۲] Training Set

[۳] Darker

[۴] Similar

[۵] Brighter

[۶] Classifier

[۷] Entropy Criteria

[۸] Binary Robust Invariant Scalable Keypoints

الف- الگوی نمونه‌برداری[۱]

ب- جبران جهت‌گیری[۲]

پ- جفت‌های نمونه‌گیری[۳]

الگوی نمونه‌برداری در BRISK کاملاً مشابه نقطه مرکزی و دایره‌ای از پیکسل‌های اطراف آن در الگوریتم FAST است که معین می‌کند نقطه مورد نظر گوشه است یا خیر. بعد از نمونه‌گیری، به دو زیر مجموعه شامل جفت‌های دور و جفت‌های نزدیک تقسیم‌بندی می‌شوند. برای اینکه الگوریتم مستقل از چرخش شود، جهت هر نقطه کلیدی نیز با استفاده از محاسبه گرادیان محلی[۴] بین جفت‌های دور و جفت‌های نزدیک انجام می‌شود. در آخر برای همه جفت نقطه‌ها، شدت نور نقطه اول و نقطه دوم مقایسه می‌شوند به این معنا که در صورتی که مقدار نقطه اول بزرگتر از نقطه دوم بود خروجی برابر "1" خواهد شد در غیر این صورت خروجی برابر "0" خواهد شد کاملاً شبیه الگوریتم BREIF. برای تطابق از الگوریتم Hamming استفاده می‌شود و فقط جمع XOR بین دو توصیفگر باینری برای تطابق دو تصویر کافی می‌باشد.

3–3–6 الگوریتم ORB[۵]:

ORB ترکیبی از FAST و BRIEF می‌باشد[۶۴]. برای استخراج نقاط کلیدی از الگوریتم FAST استفاده شده تا با ساخت یک هرم تصویری از تصویر اصلی، الگوریتم مورد نظر، مستقل از مقیاس شود. در هر مقیاس از تصویر، الگوریتم FAST اجرا می‌شود و زمانی که نقاط کلیدی تشخیص داده شد، لبه‌های بدست آمده توسط الگوریتم Harris رتبه‌بندی شده تا بر اساس مقدار آستانه تعریف شده، رتبه‌های اول تا n لبه‌ها جمع‌آوری شوند. به این دلیل که الگوریتم BRIEF در مقابل چرخش تصویر نتایج قابل اطمینانی از خود نشان نمی‌دهد، برای اینکه الگوریتم مستقل از چرخش نیز شود، تغییراتی در الگوریتم BRIEF انجام شده است.

برای محاسبه جهت یک گوشه، از پارامتری به نام مرکز شدت نور استفاده می‌شود که اولین بار روزین[۶] آن را ارائه کرد[۶۵]. روزین 3–5 و 3–6 را برای محاسبه جهت گوشه بیان کرد.

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \qquad 5–3$$

۱- Sampling Pattern

۲- Orientation Compensation

۳- Sampling Pairs

۴- Local Gradient

۵- Oriented FAST and Rotated BRIEF

۶- Rosin

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \quad 6–3$$

که m بیانگر مومنتوم تصویر می‌باشد که رابطه 3–6 بوسیله آن‌ها مرکز شدت نور را بدست می‌آورد. جهت گوشه مورد نظر نیز توسط رابطه 3–7 بدست می‌آید.

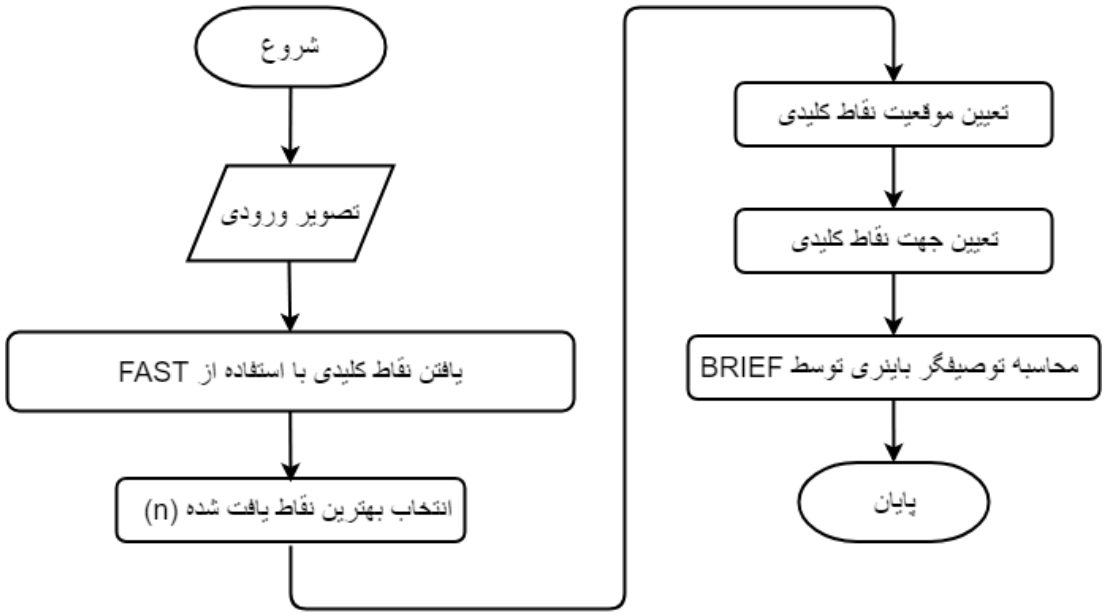
$$\theta = \arctan(m_{01}, m_{10}) \quad 7–3$$

ORB با توجه به جهت‌گیری نقاط کلیدی، ماتریس دوران تصویر را بدست می‌آورد[۶۶] و برای هر دسته ویژگی مورد آزمایش در نقطه (X_i, Y_i)، ماتریس 2*n مانند رابطه ساخته می‌شود. که آن را با استفاده از رابطه 3–9 روی نقاط اعمال می‌کند. ماتریس موقعیت ویژگی‌های استخراج شده در رابطه 3–8 نشان داده شده است:

$$S = \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{pmatrix} \quad 8–3$$

$$S_\theta = R_\theta * S \quad 9–3$$

که در رابطه 3–9، S_θ دوران یافته ماتریس S و R_θ ماتریس دوران می‌باشند. در شکل 3–11 فلوچارت الگوریتم ORB نمایش داده شده است.



شکل 3–11 فلوچارت الگوریتم ORB

4- فصل چهارم: ساخت و پیاده سازی

4–1 مقدمه:

با توسعه کاربردهای پهپادها، سیستم‌های بی‌سیم پهپادها نیز قابلیت اطمینان و سرعت بیشتری پیدا کردند و رقابت برای بهبود و تولید این سیستم‌ها افزایش یافت. مسئله مهم در این سیستم‌ها صحت ارسال و یکپارچگی ارسال اطلاعات است. در این فصل ساختمان اصلی ربات‌های پرنده بررسی می‌شود و به روش‌های ارتباط پرنده با رایانه پرداخته خواهد شد. سخت افزار و نرم‌افزارهای استفاده شده در این پژوهش، شرح داده خواهند شد. برای انجام آزمایش با توجه به روش‌های ارتباطی معرفی شده و با توجه به اینکه ارسال تصویر به صورت دیجیتال، نیازمند تجهیزات گران‌قیمت می‌باشد امکان انجام آزمایش با استفاده از این روش وجود نداشت و ارسال تصویر به صورت آنالوگ با وجود داشتن تاخیر پایین در ارسال تصویر، شدیداً نویز پذیر بوده و نمی‌توان دقت الگوریتم‌ها را به خوبی نمایش داد، روش پردازش On-Board در این پژوهش مورد بررسی قرار گرفت و به دلیل اینکه که زمینه ای برای پیاده سازی کنترل On-Board نیز برای پرنده ایجاد می‌کند، پردازش On-Board روش اصلی آزمایش انتخاب شد. به این صورت که در ابتدا هر الگوریتم بر روی رایانه شخصی آزمایش قرار گرفت و بعد از اطمینان از صحت کارکرد هر الگوریتم، کدهای مربوطه بر روی هر ربات به اجرا در آمد. آزمایش‌های صورت گرفته بر اساس تصاویر از پیش تعیین شده‌ای است که چاپ شده و در محلی قرار گرفته بود.

4–2 سخت افزار:

هر ربات پرنده معمولا شامل بدنه، موتور، سیستم کنترل‌گر و باطری می‌باشد. قطعات اصلی یک ربات در جدول 4–2 آمده است که متناسب با کاربری ربات، این قطعات نیز تغییر خواهند یافت.

جدول 4–1 لیست قطعات اصلی یک ربات پرنده معمولی

قطعه	تعداد
موتور	4
ESC	4
بدنه	1
فرستنده رادیویی(رادیو کنترل)	1
گیرنده رادیو کنترل	1
مدار کنترل پرواز	1
باطری	1

در این پژوهش از چهار ربات مختلف استفاده شده است که تصویر هر ربات را می‌توان در شکل 4–1 مشاهده کرد. هر ربات دارای بدنه مخصوص به خود و سیستم‌های پردازشی و کنترلی خاص خود می‌باشد که در ادامه به هر یک از آنها خواهیم پرداخت.

الف- ربات 1

ب- ربات 2

ت- ربات 4

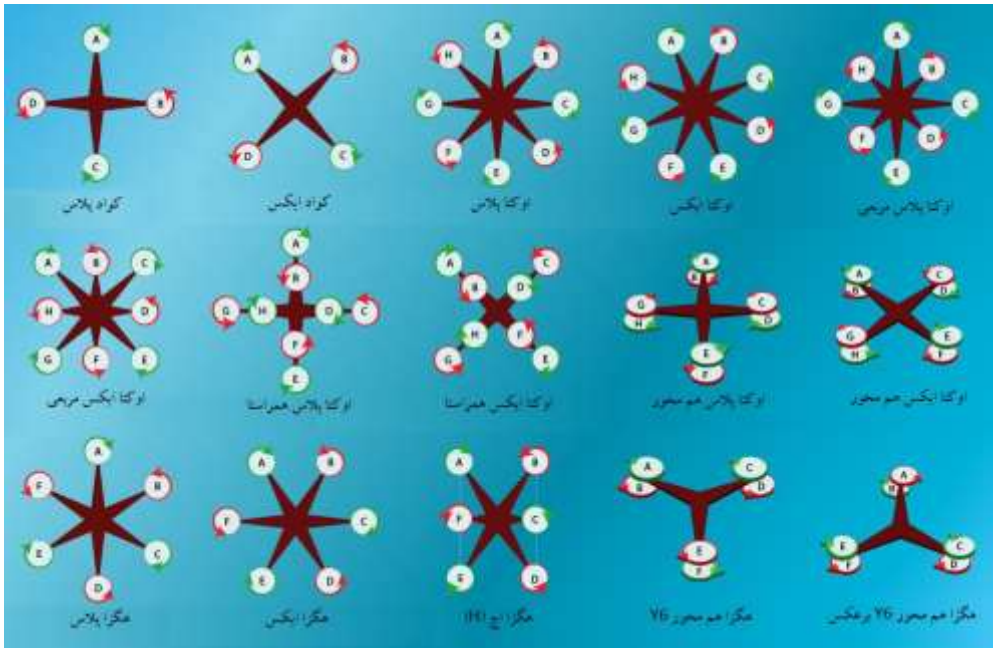
پ- ربات 3

شکل 4–1 ربات‌های استفاده شده در این پژوهش

4–2–1 بدنه:

بدنه ساختار اصلی یک ربات را تشکیل می‌دهد که تجهیزات مختلف بر روی آن نصب می‌شود. جنس بدنه ربات باید از مواد سبک و مقاوم تشکیل شود. با توجه به نحوه جایگذاری موتورها در این ربات‌ها، بدنه این ربات‌ها نامگذاری می‌شود که در شکل 4–2 مدل‌های متفاوت این بدنه‌ها نشان داده شده است.

طراحی این ربات‌ها در نرم افزار کتیا V5R20 انجام شده است و برای ساخت نیز از ماشین‌های کنترل عددی کامپیوتری یا CNC استفاده شده است.



شکل 4–2 انواع بدنه مولتی روتورها[۶۷]

ربات 1 اندازه بزرگ و دارای 8 موتور پشت به پشت به صورت هگزا هم محور ایکس می‌باشد تا با حفظ اندازه، همزمان دوبرابر قدرت بالابردنگی ایجاد کند و توانایی حمل رایانه نصب شده بر روی آن و حسگرهای جانبی را داشته باشد. طراحی بدنه ربات توسط نرم افزارهای طراحی به کمک کامپیوتر[۱] انجام شده است و جنس آن فیبر کربن با ضخامت 2 میلیمتر می‌باشد که توسط دستگاه فرز CNC سه محور برش داده شده است. در شکل 4–3 نمایی از طراحی این ربات در نرم افزار کتیا V5R20 و تصویر مونتاژ شده این ربات آمده است و همچنین مشخصات دقیق‌تر این ربات در جدول 4–6 آورده شده است.

الف- رندر نرم افزار کتیا ب- طرح ساخته شده ربات

شکل 4–3 طراحی بدنه ربات 1 در نرم افزار کتیا

[۱]- Computer Aided Design

ربات 2 بسیار سبکتر و کوچکتر از ربات 1 است و دارای مداومت پروازی بالاتری می‌باشد بدنه این ربات نیز فیبر کربن 2 میلیمتری است که همانند ربات 1 در نرم افزار کتیا طراحی شده است و توسط فرز CNC سه محور برش داده شده است. مشخصات این ربات را می‌توان در جدول 4–7 مشاهده نمود. در شکل 4–4 الف ربات طراحی شده در نرم افزار کتیا و در شکل 4–4 ب ربات مونتاژ شده نهایی را می‌توان مشاهده کرد.

رایانه نصب شده بر روی این ربات کوچکتر، سبک‌تر و با مصرف باطری کمتر است ولی نسبت به Intel NUC توان پردازشی بسیار پایین‌تری دارد. مشخصات کامل Odroid C2 در جدول 4–8 آمده است.

الف- رندر نرم افزار کتیا ب- طرح ساخته شده ربات

شکل 4–4 طراحی بدنه ربات 2 در نرم افزار کتیا

ربات 3 نیز از بدنه فیبر کربن ساخته شده و از نظر سخت افزاری بسیار مشابه ربات 2 می‌باشد با این تفاوت که موتورهای قوی‌تر و ملخ‌های بزرگتر این ربات را مناسب برای ماموریت‌های خارج از ساختمان[1] می‌کند و رایانه استفاده شده بر روی این ربات رایانه معروف Raspberry Pi3 می‌باشد که همراه با دوربین مخصوص خود وظیفه پردازش تصاویر را به عهده دارد. در شکل 4–5 الف ربات طراحی شده در نرم افزار کتیا و در شکل 4–5 ربات مونتاژ شده نهایی را می‌توان مشاهده نمود. در جدول 4–9 نیز، مشخصات دقیق ربات و در جدول 4–10 مشخصات رایانه Raspberry Pi3 ارائه شده است.

[1] Outdoor Missions

الف- رندر نرم افزار کتیا ب- طرح ساخته شده ربات

شکل 4–5 طراحی بدنه ربات 2 در نرم افزار کتیا

4–2–2 موتور:

در این نوع ربات‌ها نیروی بالابردنگی، توسط موتورهای بدون جاروبک یا Brushless Motors تامین می‌شود که ویژگی سرعت بالا وتلفات پایین برتری این موتورها نسبت به انواع دیگر آنهاست. اشکال اساسی این نوع موتورها نیز قیمت بالای آنهاست به دلیل اینکه برای راه اندازی به یک مدار کنترل کننده سرعت پیچیده نیاز دارند دوم اینکه کاربردهای این موتورهای در موارد تجاری فراهم نیامده است.

4–2–3 ملخ:

ملخ قطعه ای مهم در ربات‌های پرنده می‌باشد که با دوران موتور باعث ایجاد نیروی بالابرندگی می‌شود. در ربات‌های پرنده دو نوع ملخ ساعتگرد و پاد ساعتگرد وجود دارد که متناسب با جهت چرخش موتورها بر روی ربات بسته می‌شود. با توجه به اینکه موتورهای ربات به صورت دو به دو ساعتگرد و پاد ساعتگرد حرکت می‌کنند. ملخ ساعتگرد بر روی موتور ساعتگرد و ملخ پاد ساعتگرد بر روی موتور پاد ساعتگرد نصب می‌شود.

ملخ‌ها معمولا استاندارد هستند و دارای پارامترهای گام، طول و تعداد پره و جهت پره می‌باشند. نمونه ملخ استفاده شده بر روی ربات3 را در شکل 4–6 می‌توان دید.

شکل 4–6 ملخ های بسته AirGear شرکت T-Motor با طول 7 اینچ و گام 4.5

در جدول 4–2 مشخصات ملخ‌های استفاده شده نشان داده شده است.

جدول 4–2 مشخصات ملخ ربات‌ها

	طول ملخ(اینچ)	تعداد پره	گام ملخ
ربات1	6	2	4.5
ربات2	4	4	4
ربات3	7	2	4.5
ربات4	9	2	نامشخص

4–2–4 مدار کنترل پرواز:

پردازش اصلی ربات بر روی مدار کنترل پرواز یا فلایت کنترلر می‌باشد که وظیفه حفظ تعادل، هدایت ربات پرنده و عمل به دستورات کاربر را به عهده دارد. مدار کنترل پرواز هسته مرکزی سیستم الکترونیکی ربات است که تمام حسگرها و عملگرهای ربات به آن متصل می‌شوند. در چند سال گذشته سیستم‌های کنترل پرواز بسیاری طراحی و ساخته شده‌اند که هر کدام نسبت به کاربردهای مورد نیاز طراحی شده اند و تعدادی از آن‌ها در جدول 4–3 آمده است.

جدول 4–3 کنترلرهای معروف و نوع میکروکنترلرهای آنها

نام مدار کنترل پرواز	مدل میکروکنترلر
ArduPilot (APM) 2.6	ATmega2560
Pixhawk	32bit ARM Cortex M4 Processor
MultiWii 328P	ATmega328
CC3D Atom	ARM STM32

متداول‌ترین این سیستم ها، کنترل‌های بر پایه Arduino هستند که Ardupilot Mega و PixHawk معروف‌ترین و پر استفاده‌ترین سیستم‌های کنترل پرواز هستند که در این پژوهش از فلایت کنترلر PixHawk استفاده شدکه تصویر آن در شکل 4–7 نمایش داده شده و مشخصات این کنترلر در جدول 4–4 آمده است.

جدول 4–4 امکانات و مشخصات کنترلر پرواز PixHawk

امکانات	توضیحات
پردازنده	32 بیت با فرکانس 168 مگاهرتز
	حافظه 256 کیلوبایت
	2 مگابایت حافظه فلش
سنسورها	ژایروسکوپ L3GD20
	شتابسنج و قطب نما 14 بیتی LSM303D
	بارو متر MS5611
	شتابسنج و ژایروسکوپ MPU6000
رابط ها	5 رابط UART
	2 رابط CAN
	I2C
	SPI
	USB داخلی و خارجی

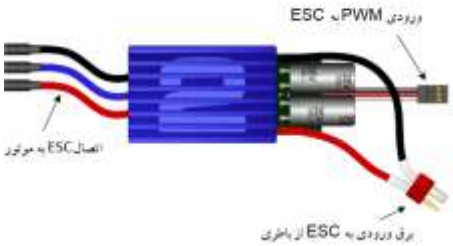
فلایت کنترلر PixHawk امکان نصب سنسورهای جانبی مانند GPS، تله متری[1]، فاصله سنج Ultrasonic، فاصله سنج لیزری و ... را نیز دارد.

شکل 4–7 کنترلر مولتی روتور PixHawk

[1] Telemetry

4–2–5 مدار کنترل سرعت ESC[1]:

مدار کنترل سرعت یا اسپید کنترلر، مداری است که وظیفه تنظیم سرعت موتورهای براشلس را بر عهده دارد. ربات نیاز دارد که دستورات خروجی از میکروکنترلر را به بدون تاخیر به موتورها ارسال کند تا تعادل ربات دچار اختلال نشود. دیاگرام الکترونیکی ESC در شکل 4–8 آورده شده است. در ... لیست ESC ربات‌ها آورده شده است.



شکل 4–8 نحوه اتصال مدار کنترل پرواز، باطری و موتور به ESC

جدول 4–5 مشخصات ESC ربات‌ها

	شدت جریان (آمپر)
ربات1	15
ربات2	35
ربات3	30
ربات4	نامشخص

4–2–6 فرستنده و گیرنده رادیویی:

ارسال دستورات بی سیم به مدار کنترل پرواز، نیازمند نظارت و کنترل عامل انسانی است. در این فرآیند بدون نویز بودن و سرعت پارامترهای بسیار مهمی هستند. شرکت‌های سازنده با رمز گذاری بر روی داده‌های انتقالی از افتادن نویز بر روی داده‌ها جلوگیری کرده اند. فرکانس کاری این رادیو کنترل‌ها معمولا 2.4 گیگاهرتز هستند. رادیو کنترل استفاده شده در این پژوهش ساخت شرکت JR می‌باشد و 8 کانال ارسال داده را پشتیبانی می‌کند. تصویر فرستنده و گیرنده استفاده شده در شکل 4–9 آمده است.

[1]- Electronic Speed Controller

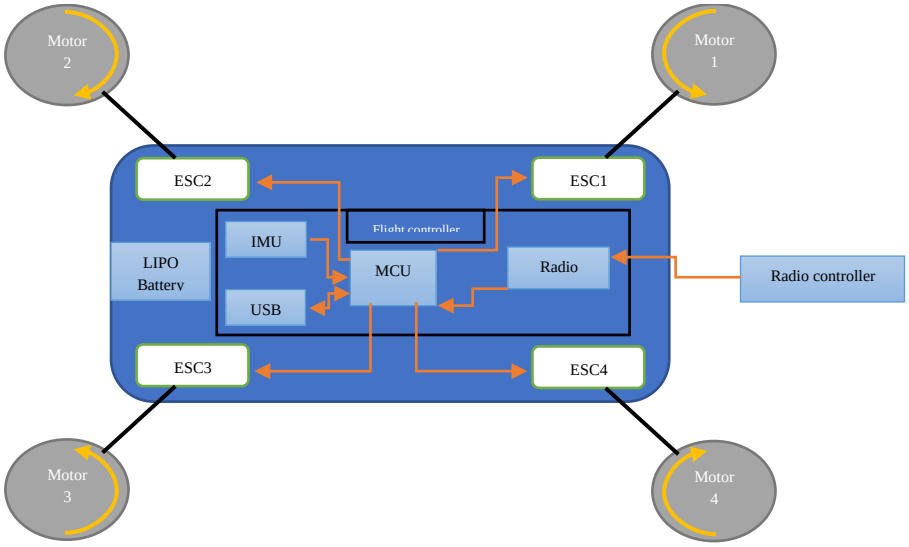
شکل 4–9 رادیو کنترل JR XG8

4–2–7 باطری لیتیوم پلیمر:

باطری منبع انرژی الکتریکی ربات می‌باشد. این باطری ها از نوع لیتیوم پلیمر با قابلیت تخلیه بالا هستند باطری تعیین کننده زمان پرواز ربات است. با افزایش ظرفیت باطری، زمان پرواز نیز افزایش می‌یابد ولی باعث افزایش وزن ربات و مصرف باطری بیشتر ربات می‌شود پس متناسب بودن ظرفیت باطری و وزن ربات نسبت قابل تاملی می‌باشد پس باطری مناسب با ظرفیت بالا و وزن پایین می‌تواند تعیین کننده باشد. تصویر باطری‌های مورد استفاده در ربات‌ها در شکل 4–10 و شکل 4–11 آمده است.

شکل 4–10 باطری لیتیوم پلیمر 2250میلی آمپر ساعت شکل 4–11 باطری لیتیوم پلیمر2300 میلی آمپر ساعت

در شکل 4–12 دیاگرام الکترونیکی یک ربات پرنده نمایش داده شده است. رادیو کنترل ربات با فرکانس 2.4 گیگاهرتز رادیویی با دریافت کننده خود(Receiver) ارتباط برقرار می‌کند و دستورات خلبان را ارسال می‌کند.واحد اندازه گیری اینرسی(IMU) مسئول اندازه گیری زوایای ربات می‌باشد که با استفاده از پروتکل SPI داده‌های زوایای ربات را به میکروکنترلر ربات ارسال می‌کند. بعد از پردازش داده‌ها ورودی از رادیو کنترل و واحد اندازه‌گیری اینرسی، میکروکنترلر دستورات مربوطه را به ESC ها با استفاده از PWM ارسال می‌کند و ESC دستورات PWM را به سیگنال‌های مناسب موتورها(3Phase-AC) تبدیل و ارسال می‌کند.



شکل 4–12 معماری الکترونیکی یک کوادکوپتر

4–2–8 جدول سخت افزار ربات‌ها:

در این بخش جداول قطعات و سخت افزارهای ربات‌های ساخته شده برای مقایسه بهتر آورده شده است.

جدول 4–6 مشخصات سخت افزاری ربات 1

قطعه	تعداد	مشخصات
موتور	8	براشلس Air-Gear
ESC	8	Air-Gear
بدنه	1	فیبر کربن 2 میلیمتر
فرستنده رادیویی(رادیو کنترل)	1	JR XG8
گیرنده رادیو کنترل	1	JR XG8
مدار کنترل پرواز	1	PixHawk
باطری	2	Tattu 2300mAh
رایانه	1	NUC Intel Corei7
دوربین	1	وبکم logitech C922

جدول 4–7 مشخصات سخت افزاری ربات 2

قطعه	تعداد	مشخصات
موتور	4	براشلس Air-Gear
ESC	4	Air-Gear
بدنه	1	فیبر کربن 2 میلیمتر
فرستنده رادیویی(رادیو کنترل)	1	JR XG8
گیرنده رادیو کنترل	1	JR XG8

مدار کنترل پرواز	1	PixHawk
باطری	1	Tattu 2300mAh
رایانه	1	Odroid C2
دوربین	1	وبکم logitech C922

جدول 4–8 مشخصات رایانه Odroid C2 ربات 2

امکانات	تعداد
پردازنده	64-bit quad-core @ 1.5 GHz Cortex-A53
پردازنده گرفیکی	Pixel +2 Vertex Shader 3 Triple Core
RAM	2 GB DDR3 @ 912 MHz
USB	4xUSB 2.0 Host
Power	5V 2 A DC input
سیستم عامل	Linux-Android

جدول 4–9 مشخصات سخت افزاری ربات 3

قطعه	تعداد	مشخصات
موتور	4	براشلس Air-Gear
ESC	4	Air-Gear
بدنه	1	فیبر کربن 3 میلیمتر
فرستنده رادیویی(رادیو کنترل)	1	JR XG8
گیرنده رادیو کنترل	1	JR XG8
مدار کنترل پرواز	1	PixHawk
باطری	1	Tattu 2300mAh
رایانه	1	Raspberry Pi 3
دوربین	1	وبکم logitech C922

جدول 4–10 مشخصات رایانه رسپبری پای ربات 3

امکانات	تعداد
پردازنده	64-bit @ 1.4 GHz Cortex-A53 64-bit
پردازنده گرفیکی	VideoCore IV
RAM	1GB DDR2
USB	4xUSB 2.0 Host
Power	5V 1.13A 6.7watt
سیستم عامل	Linux-Android-Raspbian

جدول 4–11 مشخصات رایانه ربات 4

امکانات	تعداد
پردازنده	ARM Cortex A8 1 GHz 32-bit processor
پردازنده گرفیکی	DSP video 800 MHz TMS320DMC64x
RAM	DDR2 1 GB at 200 MHz
USB	High-speed USB 2.0 for extensions
سیستم عامل	Linux 2.6.32
دوربین	720p 30fps HD camera
ارتباط	Wifi

4–3 نرم افزار:

بخش نرم افزاری این نوع ربات‌ها شمال چند زیر مجموعه می‌باشدکه شامل بخش‌های زیر است:

- ✓ کنترل‌گر یا مدار کنترل پرواز
- ✓ رابط کاربری گرافیکی
- ✓ کنترلر سطح بالا
- ✓ پردازش تصویر

تمرکز این پژوهش بر روی الگوریتم‌های پردازش تصویر می‌باشد ولی در ادامه توضیحات کلی در مورد 3 بخش دیگر ارائه خواهد شد.

4–3–1 کنترل‌گر یا مدار کنترل پرواز:

بخش نرم افزاری کنترل‌گر کاملا با سخت افزار انتخابی مرتبط است به این صورت که هر کنترل‌گر نسبت به میکروپردازنده این که بر روی آن نصب شده برنامه نویسی می‌شود. در این پژوهش با توجه به متن باز بودن و دسترس بودن (PixHawk)، این کنترلر برای آزمایش مورد استفاده قرار گرفته است لازم به ذکر است زبانی که توسعه دهندگان برای برنامه نویسی این کنترلر به کار گرفتند، زبان C می‌باشد.

4–3–2 رابط کاربری گرافیکی:

برای ارتباط راحت‌تر، تنظیمات ربات و مانیتورینگ وضعیت ربات نرم افزار گرافیکی ایجاد شدند که به نرم افزارهای ایستگاه زمینی یا GSC معروف هستند. این نرم افزارها داده‌های دریافتی از ربات را که با استفاده از سیستم‌های بی سیم دریافت می‌کنند به صورت بلادرنگ نمایش می‌دهند. از این نرم افزارها می‌توان برای کنترل ربات نیز استفاده نمود به همین دلیل به آن‌ها اتاقک مجازی[1] نیز گفته می‌شود. نرم افزارهای پر استفاده برای کار با ربات‌های پرنده در جدول 4–12 آورده شده‌اند:

- ✓ Mission Planner
- ✓ Apm Planner 2.0
- ✓ Qground Control
- ✓ UgCS - Universal Ground Control Station

جدول 4–12 نرم افزارهای رابط گرافیکی تنظیم کنترلر

نرم افزار	پلتفورم	Licence
Mission Planner	ویندوز، MAC	Open source (GPLv3)
Apm Planner 2.0	ویندوز، MAC، لینوکس	Open source (GPLv3)
Qground Control	ویندوز، MAC، لینوکس، اندروید، iOS	Open source (GPLv3)
UgCS - Universal Ground Control Station	ویندوز، MAC، لینوکس	خصوصی

[1] Virtual Cockpit

نمایی از نرم افزارهای Mission Planner، Apm Planner، Qground Control و در به ترتیب در شکل 4–13، شکل 4–14، شکل 4–15 آمده است. ساختار کلی این نرم‌افزارها بسیار به یکدیگر نزدیک هستند پروتکل ارتباطی این نرم افزارها با کنترلرها، در اکثر مواقع بر اساس کتابخانه MAVLink می‌باشد.

در این پژوهش به دلیل امکانات بالای نرم افزارهای متن باز جدول 4–12 و امکان استفاده از آن بر روی پلتفرم‌های مورد نظر ما و موجود بودن آموزش‌های کافی برای استفاده از این نرم افزارها، از آنها به عنوان نرم افزار اصلی تنظیم و راه اندازی کنترلر ربات‌های مورد آزمایش، استفاده شد. در محیط ویندوز از نرم افزار Mission Planner استفاده شد و به این دلیل که نرم ازار Mission Planner با زبان #C نوشته شده است و با سیستم عامل‌های لینوکس مطابقت ندارد در محیط لینوکس از نرم افزارهای APM Planner و Qground Controll استفاده گردید.

شکل 4–13 تصویری از محیط کاربری Mission Planner

شکل 4–14 تصویری از محیط کاربری Apm Planner2

شکل 4–15 تصویری از محیط کاربری QGround Control

4–3–3 کنترلر سطح بالا:

همانطور که قبل‌تر نیز گفته شد، کنترلر، مسئولیت کنترل تعادل ربات را به عهده دارد و مسیریابی به عهده اپراتور با خلبان می‌باشد. محققین برای خودکار سازی مسیریابی این ربات‌ها، کنترلرهایی ایجاد کردند که با استفاده از پردازش تصویر، سنسورهای بیکن[۱] و ... مسیریابی ربات را تسهیل می‌بخشند. کنترلر ربات، وظیفه حفظ تعادل ربات و پایداری ربات را به عهده دارد اما کنترلرهای سطح بالا با ارسال دستور به کنترلر ربات، درجه بالاتری از هوشمندی را برای ربات ایجاد می‌کنند.

4–3–4 پردازش تصویر:

همانگونه که پیش‌تر نیز گفته شد، در صورتی که بخواهیم از پردازش تصویر برای کنترل رفتار ربات استفاده کنیم، خروجی‌های پردازش تصویر به کنترلر سطح بالا، برای تصمیم‌گیری ارسال می‌شوند. در پژوهش تمرکز اصلی بر این بخش از سیستم می‌باشد که روند کار در ادامه شرح داده خواهد شد.

در ابتدا تصاویر از پیش تعیین شده‌ای تهیه و چاپ شد. تصاویر بعد از اینکه توسط ربات گرفته شد با تصاویر اصلی مورد مقایسه قرار گرفت و زمان پردازش هر الگوریتم، تعداد نقاط یافت شده و تعداد نقاط تطابق داده شده مورد تحلیل قرار گرفت.

در این پژوهش از کتابخانه متن باز OpenCV برای به کارگیری الگوریتم‌های بینایی استفاده شد. این کتابخانه مجموعه‌ای از توابع را شامل می‌شود که اساساً برای بینایی ماشین بلادرنگ[۲] طراحی شده است. شرکت اینتل توسعه دهنده این کتابخانه بود تا اینکه Willow Garage و پس از آن Itseez توسعه آن را بر عهده گرفتند و در نهایت مجدداً Intel آن را بدست گرفت. این کتابخانه بیش از 2500 الگوریتم بهینه شده بینایی ماشین را در خود جای داده است و با استفاده از زبان برنامه نویسی C++ نوشته شده است و تحت سیستم عامل‌های ویندوز، لینوکس، مکینتاش، اندروید و Ios قابل استفاده می‌باشد. زبان‌های برنامه نویسی که توسط این کتابخانه پشتیبانی می‌شوند عبارت اند از:

- ✓ Python
- ✓ Java
- ✓ Matlab/Octave

تعدادی از کاربردهای کتابخانه OpenCV در زیر آمده است:

- ✓ تخمین حرکت
- ✓ تشخیص چهره
- ✓ تشخیص ژست[۳]

---

[۱] Beacon

[۲] real-time

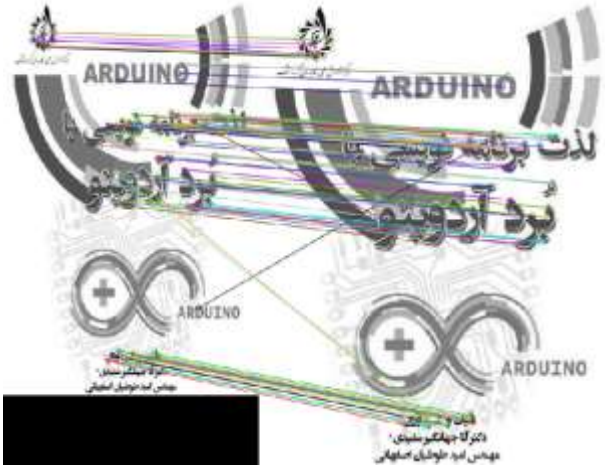
[۳] Gesture Recognition

- ✓ تعامل انسان و کامپیوتر[1]
- ✓ ربات‌های متحرک
- ✓ تشخیص اشیا
- ✓ بینایی استریو[2]
- ✓ واقعیت افزوده[3]

در این پژوهش از زبان برنامه نویسی Python تحت سیستم عامل Ubuntu بهره گرفته شد و کتابخانه OpenCV، Numpy، Time و Matplotlib برای استفاده بهینه از کتابخانه OpenCV استفاده شد. نتایج آزمایش‌های صورت گرفته و مقایسه نتایج، در ادامه آمده است.

در این پژوهش برای بررسی الگوریتم‌ها، تصاویر تحت شرایط چرخش، مقیاس و تغییرات زاویه دید در شرایط ایده آل مورد آزمایش قرار گرفتند بعد از آن ربات Ar.Drone 2.0 برای تصویر برداری به کار گرفته شد تا تصاویر به بهترین نحو مورد آنالیز قرار بگیرند. طبق شکل 4–20 در هر کدام از شرایط، 7 تصویر جمع‌آوری شد تحت بررسی قرار گرفت و میانگین پارامترهای تعیین کننده به عنوان خروجی در نظر گرفته شد.

تغییر مقیاس در 7 سایز مختلف از 100٪ تا 40٪ سایز واقعی در نظر گرفته شد و پارامترهای پژوهش جمع آوری شدند. ویژگی‌های یافت شده در سایز 60٪ را در شکل 4–16 می‌توان دید.



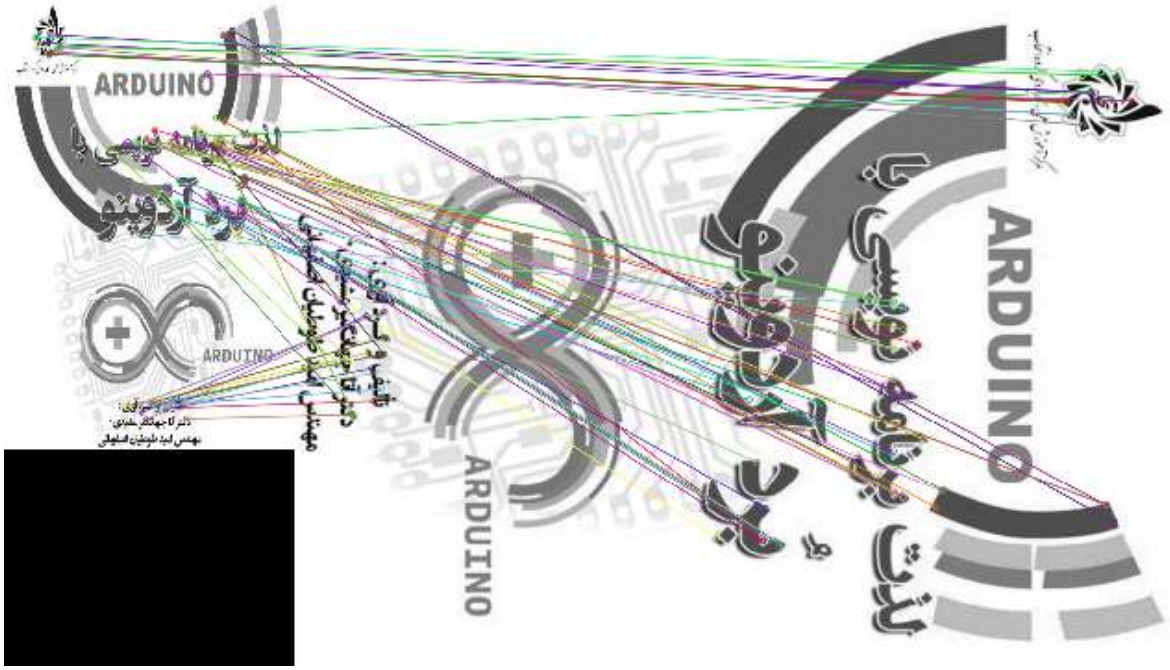
شکل 4–16 نمایش ویژگی‌های تطابق یافته تصویر اصلی(چپ) در مقابل مقیاس 60 درصد(راست)

[1] Human-Computer Interaction

[2] Stereo Vision

[3] Augmented Reality

برای آزمایش الگوریتم‌ها در زوایای مختلف، تصاویر با گام 30 درجه از 0 تا 180 درجه چرخش داده شدند و ویژگی‌های آنها و پارامترهای آن‌ها استخراج شدند. که ویژگی های تطابق یافته تصویر با زاویه 90 درجه در شکل 4–17 نمایش داده شده اند.



شکل 4–17 تطابق ویژگی‌های استخراج شده تصویر اصلی و تصویر چرخش داده شده

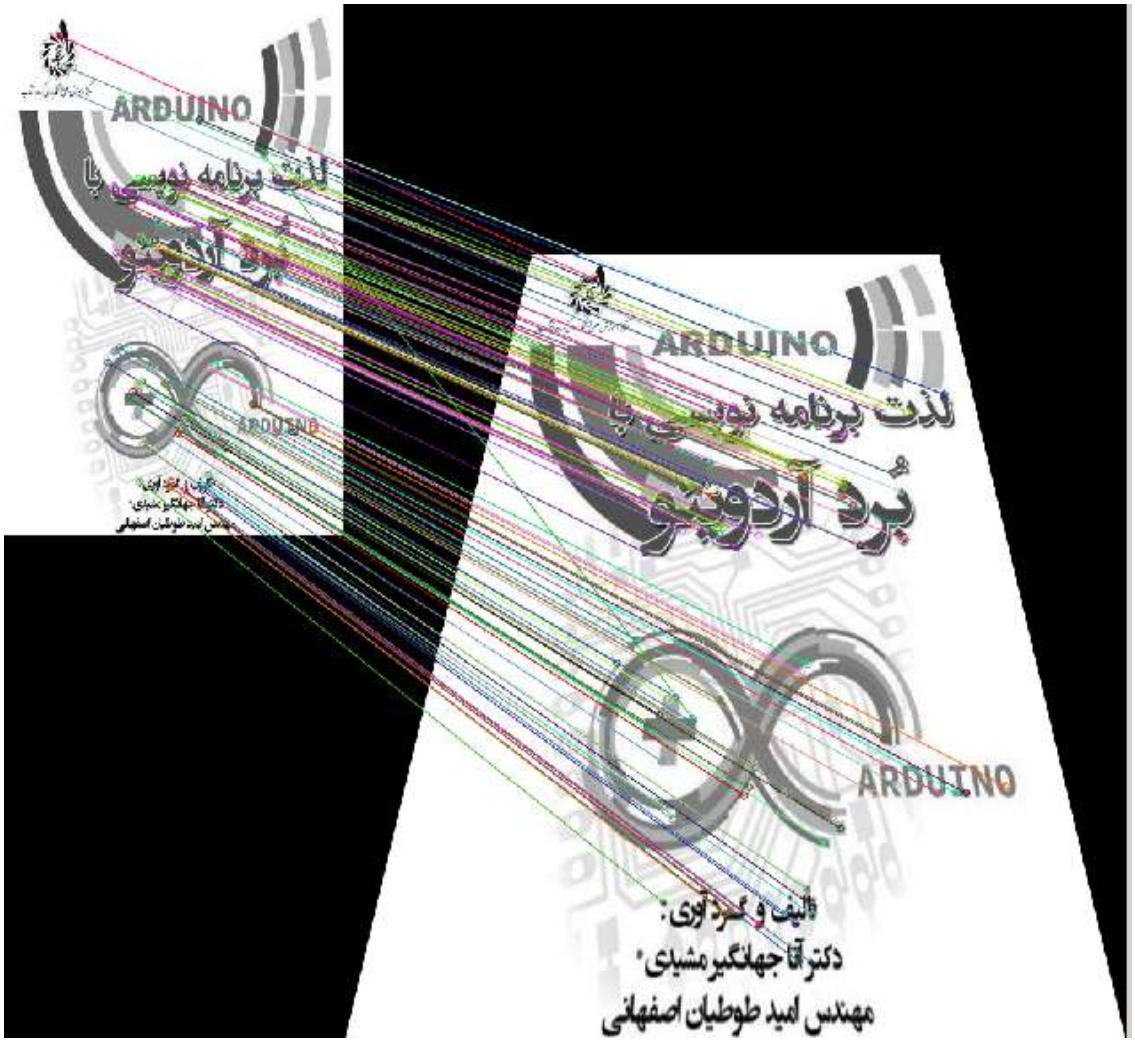
آزمایش تغییرات زاویه دید به این صورت انجام گرفت که ارتفاع H و عرض X هر بار 0.6 درصد کاهش پیدا کردند به صورتی که در شکل 4–19 ویژگی‌های تطابق یافته را می‌توان مشاهده نمود. 4–1 رابطه محاسبه اندازه تصویر جدید را نشان می‌دهد.

$$H_n = ratio * n * H \qquad 1–4$$
$$W_n = ratio * n * W$$

که در 4–1، W و H، ارتفاع و عرض تصاویر اصلی و Ratio درصد تغییرات ارتفاع و عرض را نشان می‌دهد که در اینجا 0.6 درصد می‌باشد و H_n و W_n، ارتفاع و عرض تصویر n ام می‌باشد. نمایی از آن در شکل 4–18 مشاهده می‌شود.

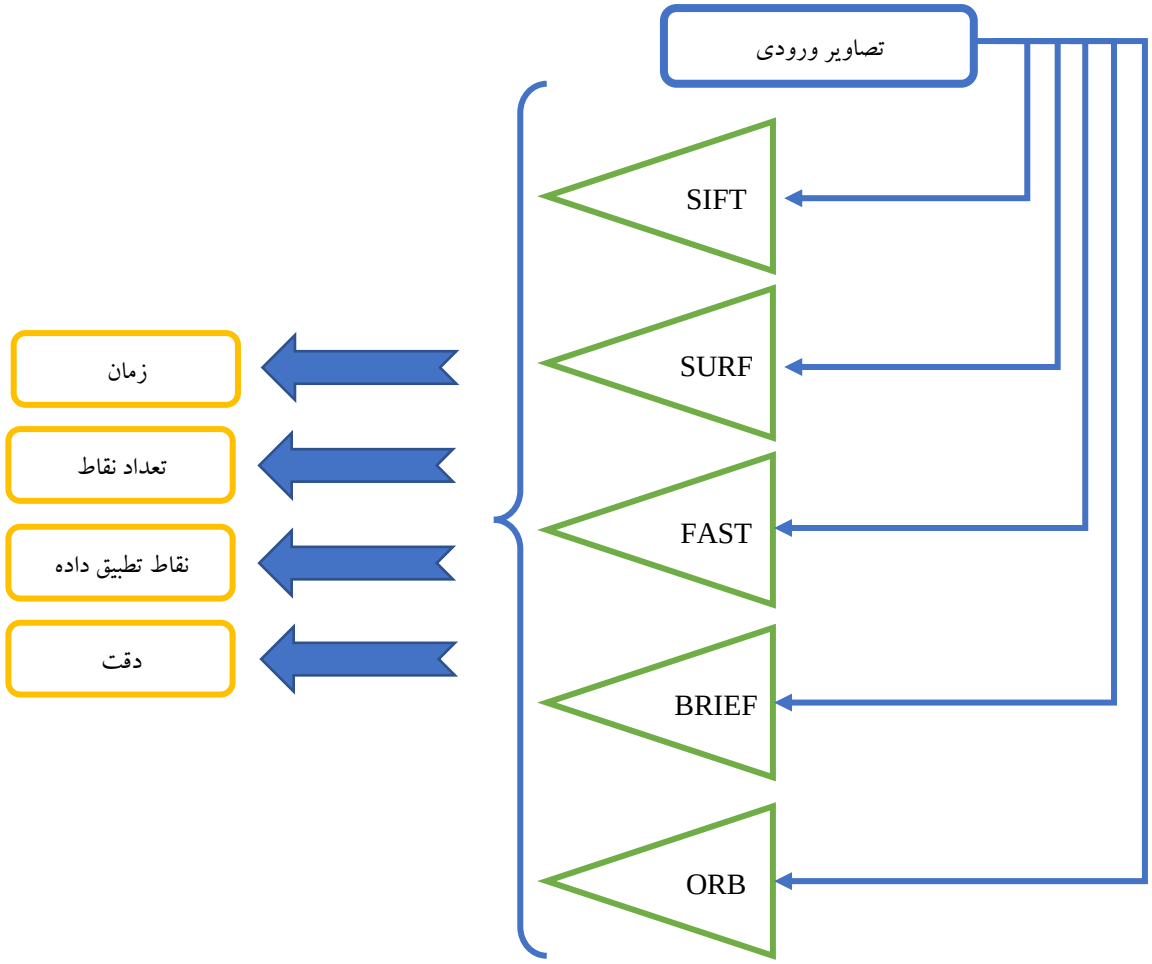
شکل 4–18 تغییرات زاویه دید با استفاده از تغییر ارتفاع و عرض تصویر

در شکل 4–19 ویژگی‌های تطابق یافته در شرایط تغییرات زاویه دید نشان داده شده است.



شکل 4–19 ویژگی‌های تطابق یافته در شرایط تغییرات زاویه دید

همانگونه که در شکل 4–20 نشان داده شده است تصاویر ورودی توسط الگوریتم‌های SIFT، SURF، FAST، ORB و BRISK آنالیز می‌شوند و بعد از آن پارامترهای زمان، تعداد ویژگی‌های یافت شده، ویژگی‌های مطابقت داده شده و دقت هر الگوریتم با توجه به ویژگی‌های یافت شده و تطبیق داده شده بر روی رایانه هر ربات محاسبه می‌شوند. در این مرحله از آنالیز الگوریتم‌ها، هر 5 الگوریتم، 19 تصویر در شرایط متفاوت چرخش مقیاس و زاویه دید را مورد بررسی قرار دادند و میانگین پارامترهای مورد نظر محاسبه گردید.



شکل 4–20 مراحل ارزیابی الگوریتم‌ها

برای محاسبه دقت الگوریتم مورد نظر، بعد از استخراج ویژگی های تصویر اصلی و تصویر دریافتی از دوربین ربات، تصاویر مقایسه می‌شوند و موارد تطبیق داده شده از موارد غیر قابل قبول جدا می‌شوند با این کار درصد دقت هر الگوریتم برای هر عکس طبق رابطه 4–2 محاسبه می‌شود.

$$Accuracy\% = \frac{Correct\ Matches}{Total\ Match\ Found} * 100 \qquad 2–4$$

به دلیل اینکه تعداد عکس‌های نمونه گیری متعدد است رابطه 4–3 برای سنجش دقت هر الگوریتم مورد استفاده قرار گرفت.

$$Accuracy\% = \frac{1}{n}\sum_{i=1}^{n}\frac{I_m}{I_{ex}} * 100 \qquad 3–4$$

که در این رابطه I_m برابر با تعداد ویژگی‌ها تطابق داده شده می‌باشد و I_{ex} تعداد کل ویژگی‌های یافت شده در تصویر می‌باشد و n تعداد عکس‌های نمونه گیری شده می‌باشد[۶۸]. جدول نتایج ربات‌های اول تا سوم به ترتیب در جدول 4–13 ،جدول 4–14 و جدول 4–15 آمده است.

جدول 4–13 میانگین پارامترهای محاسبه شده هر الگوریتم در ربات 1

الگوریتم	نوع تصویر	زمان اجرا(ثانیه)	ویژگی‌های استخراج شده		ویژگی‌های تطابق یافته	دقت(درصد)
			تصویر اصلی	تصویر ربات		
ORB	مقیاس	۰.۰۳۶	۵۰۰	۵۰۰	۹۱	۱۸.۳۱
	چرخش	۰.۰۴۷	۵۰۰	۵۰۰	۵۲	۱۰.۴۲
	زاویه دید	۰.۰۴۱	۵۰۰	۵۰۰	۴۳	۸.۶
SURF	مقیاس	۰.۲۴۲	۱۲۷۷	۲۲۴۴	۵۸۵	۴۵.۸۲
	چرخش	۰.۴۲۵	۱۲۷۷	۴۴۰۶	۵۱۵	۳۱.۸۹
	زاویه دید	۰.۳۵	۱۲۷۷	۳۲۶۳	۴۷۵	۳۷.۸۷
SIFT	مقیاس	۰.۱۶۸	۸۵۵	۱۵۳۲	۵۲۸	۶۱.۷۵
	چرخش	۰.۲۸۶	۸۵۵	۲۲۴۱	۵۱۵	۶۰.۳
	زاویه دید	۰.۲۱۸	۸۵۵	۱۸۰۴	۴۷۵	۵۵.۶۳
FAST	مقیاس	۰.۰۳۵۰۹۳	۸۵۰	۱۰۰۱	۱۲۷	۱۵.۰۲
	چرخش	۰.۰۴	۸۵۰	۱۰۶۹	۴۷	۵.۵
	زاویه دید	۰.۰۳۶	۸۵۰	۱۱۲۰	۸۴	۱۵.۰۲
BRISK	مقیاس	۰.۳۷۷	۱۱۶۱	۱۶۴۱	۱۹۱	۱۶.۴۵
	چرخش	۰.۴۳۵	۱۱۶۱	۲۲۸۵	۱۲۹	۱۳.۹۹
	زاویه دید	۰.۳۹۱	۱۱۶۱	۲۰۲۵	۱۶۲	۱۱.۱۱

جدول 4–14 میانگین پارامترهای محاسبه شده هر الگوریتم در ربات 2

الگوریتم	نوع تصویر	زمان اجرا(ثانیه)	ویژگی‌های استخراج شده		ویژگی‌های تطابق یافته	دقت(درصد)
			تصویر اصلی	تصویر ربات		
ORB	مقیاس	۰.۱۷۸۱	۵۰۰	۵۰۰	۹۲	۱۸.۴۲
	چرخش	۰.۲۷۴۴	۵۰۰	۵۰۰	۴۹	۹.۹۴
	زاویه دید	۰.۲۲۵۵	۵۰۰	۵۰۰	۴۱	۸.۲۲
SURF	مقیاس	۱.۱۰۲۲	۱۲۷۵	۲۲۴۴	۵۸۹	۴۶.۲۵
	چرخش	۲.۰۵۴۷	۱۲۷۵	۴۴۰۶	۴۰۷	۳۱.۹۵
	زاویه دید	۱.۶۱۱	۱۲۷۵	۳۲۶۳	۴۹۳	۳۸.۶۸
SIFT	مقیاس	۱.۲۲۰۶	۸۵۱	۱۵۳۲	۵۲۶	۶۱.۸
	چرخش	۲.۶۰۰۶	۸۵۱	۲۲۴۰	۵۰۸	۵۹.۷
	زاویه دید	۱.۸۰۱۲	۸۵۱	۱۸۰۵	۴۷۴	۵۵.۷۴
FAST	مقیاس	۰.۱۷۶۳	۸۵۵	۱۰۰۱	۱۲۹	۱۵.۱۸
	چرخش	۰.۲۱۰۵	۸۵۵	۱۰۶۹	۴۵	۵.۲
	زاویه دید	۰.۲۰۰۳	۸۵۵	۱۱۲۰	۸۵	۱۰.۰۲
BRISK	مقیاس	۲.۱۰۰۵	۱۱۶۲	۱۶۴۱	۱۸۸	۱۶.۲۲
	چرخش	۲.۴۰۷	۱۱۶۲	۲۲۸۵	۱۵۸	۱۳.۶۴
	زاویه دید	۲.۲۹۰۹	۱۱۶۲	۲۰۲۵	۱۲۶	۱۰.۸۸

جدول 4–15 میانگین پارامترهای محاسبه شده هر الگوریتم در ربات 3

الگوریتم	نوع تصویر	زمان اجرا(ثانیه)	ویژگی‌های استخراج شده		ویژگی‌های تطابق یافته	دقت(درصد)
			تصویر اصلی	تصویر ربات		
ORB	مقیاس	۰.۳۵۲۹	۵۰۰	۵۰۰	۹۱	۱۸.۲۸
	چرخش	۰.۵۷۴۹	۵۰۰	۵۰۰	۴۸	۹.۷۱
	زاویه دید	۰.۴۳۶۴	۵۰۰	۵۰۰	۴۲	۸.۴۵
SURF	مقیاس	۲.۴۱۴۹	۱۲۷۷	۲۲۴۴	۵۸۵	۴۵.۸
	چرخش	۴.۴۰۴۹	۱۲۷۷	۴۴۰۶	۴۰۷	۳۱.۸۹
	زاویه دید	۳.۵۴۲۸	۱۲۷۷	۳۲۶۳	۴۸۳	۳۷.۸۷
SIFT	مقیاس	۴.۲۸۸۶	۸۵۵	۱۵۳۲	۵۲۷	۶۱.۷۳
	چرخش	۷.۹۹۳۵	۸۵۵	۲۲۴۰	۵۱۵	۶۰.۲۸
	زاویه دید	۶.۱۵۵۳	۸۵۵	۱۸۰۵	۴۷۵	۵۵.۶۳
FAST	مقیاس	۰.۳۹۱۶	۸۵۰	۱۰۰۱	۱۲۷	۱۵.۰۲
	چرخش	۰.۴۵۹۷	۸۵۰	۱۰۶۹	۴۷	۵.۵۲
	زاویه دید	۰.۴۴۴۴	۸۵۰	۱۱۲۰	۸۴	۹.۹
BRISK	مقیاس	۲.۹۹۶۶	۱۱۶۱	۱۶۴۲	۱۸۳	۱۵.۷۹
	چرخش	۳.۴۱۳۱	۱۱۶۱	۲۲۸۵	۱۵۵	۱۳.۳۸
	زاویه دید	۳.۲۲۷۳	۱۱۶۱	۲۰۲۶	۱۲۱	۱۰.۴۸

در جدول 4–16، پارامترهای جداول بالا به نحو مطلوبتری برای مقایسه ربات‌ها به نمایش گذاشته شده است.

جدول 4–16 مقایسه پارامترهای محاسبه شده هر سه ربات

الگوریتم	نوع تصویر	پارامتر	ربات اول	ربات دوم	ربات سوم
ORB	مقیاس	زمان اجرا(ثانیه)	۰.۰۳۶۱	۰.۱۷۸۱	۰.۳۵۲۹
		دقت(درصد)	۱۸.۳۱	۱۸.۴۳	۱۸.۲۸۶
	چرخش	زمان اجرا(ثانیه)	۰.۰۴۷۳	۰.۲۷۴۴	۰.۵۷۴۹
		دقت(درصد)	۱۰.۴۳	۹.۹۴۳	۹.۷۱۴۳
	زاویه دید	زمان اجرا(ثانیه)	۰.۰۴۱	۰.۲۲۵۵	۰.۴۳۶۴
		دقت(درصد)	۸.۶۲۹	۸.۲۲۹	۸.۴۵۷۱
SURF	مقیاس	زمان اجرا(ثانیه)	۰.۲۴۲	۱.۱۰۲۲	۲.۴۱۴۹
		دقت(درصد)	۴۵.۸۲	۴۶.۲۵۲	۴۵.۸۲۲
	چرخش	زمان اجرا(ثانیه)	۰.۴۲۵۱	۲.۰۵۴۷	۴.۴۰۴۹
		دقت(درصد)	۳۱.۸۹	۳۱.۹۶	۳۱.۸۹۴
	زاویه دید	زمان اجرا(ثانیه)	۰.۳۵۰۳	۱.۶۱۱	۳.۵۴۲۸
		دقت(درصد)	۳۷.۸۸	۳۸.۶۹	۳۷.۸۷۹
SIFT	مقیاس	زمان اجرا(ثانیه)	۰.۱۶۷۶	۱.۲۲۰۶	۴.۲۸۸۶
		دقت(درصد)	۶۱.۷۵	۶۱.۸۱	۶۱.۷۳۸
	چرخش	زمان اجرا(ثانیه)	۰.۲۸۵۷	۲.۶۰۰۶	۷.۹۹۳۵
		دقت(درصد)	۶۰.۳	۵۹.۷۴	۶۰.۲۸۴
	زاویه دید	زمان اجرا(ثانیه)	۰.۲۱۷۸	۱.۸۰۱۲	۶.۱۵۵۳
		دقت(درصد)	۵۵.۶۴	۵۵.۷۵	۵۵.۶۳۹
FAST	مقیاس	زمان اجرا(ثانیه)	۰.۰۳۵۱	۰.۱۷۶۳	۰.۳۹۱۶
		دقت(درصد)	۱۵.۰۳	۱۵.۱۹	۱۵.۰۲۵
	چرخش	زمان اجرا(ثانیه)	۰.۰۳۹۶	۰.۲۱۰۵	۰.۴۵۹۷
		دقت(درصد)	۵.۵۲۹	۵.۲۶۳	۵.۵۲۹۴
	زاویه دید	زمان اجرا(ثانیه)	۰.۰۳۶۳	۰.۲۰۰۳	۰.۴۴۴۴
		دقت(درصد)	۹.۹۳۳	۱۰.۰۳	۹.۹۳۲۸
BRISK	مقیاس	زمان اجرا(ثانیه)	۰.۳۷۷	۲.۱۰۰۵	۲.۹۹۶۶
		دقت(درصد)	۱۶.۴۵	۱۶.۲۳	۱۵.۷۹۹
	چرخش	زمان اجرا(ثانیه)	۰.۴۳۴۸	۲.۴۰۷	۳.۴۱۳۱
		دقت(درصد)	۱۳.۹۹	۱۳.۶۵	۱۳.۳۸۷
	زاویه دید	زمان اجرا(ثانیه)	۰.۳۹۱۳	۲.۲۹۰۹	۳.۲۲۷۳
		دقت(درصد)	۱۱.۱۱	۱۰.۸۸	۱۰.۴۸۴

در جدول 4–17 زمان پردازش هر الگوریتم در شرایط مختلف مورد مقایسه قرار گرفته است. که الگوریتم SIFT بر روی رایانه Raspberry Pi 3 بیشترین زمان اجرا را در شرایط چرخش نشان داده است که با توجه به افزایش ماتریس تصویر یا همان اندازه تصویر هنگام چرخش، کاملاً قابل پیش‌بینی بود واین نکته را در الکوریتم‌های دیگر نیز می‌توان مشاهده کرد که سرعت پایین تمامی الگوریتم‌های مورد آزمایش به دلیل افزایش اندازه ماتریس تصویر چرخش داده شده است.

الگوریتم FAST نیز به دلیل اینکه فقط یک استخراج کننده ویژگی هست، سریعترین الگوریتم بین الگوریتم‌های مورد آزمایش بود که این نیز قابل پیش‌بینی بود. باید توجه داشت الگوریتم ORB بر روی هر سه ربات عملکرد سریعی از خود نشان داده است با وجود اینکه اختلاف سخت افزارهای پردازشی ربات اول و سوم بسیار زیاد می‌باشد نتایج کاملا قابل قبولی را از الگوریتم ORB شاهد بودیم این توجیه مناسبی برای استفاده از این الگوریتم در ربات‌های پایه متحرک می‌باشد که امکان استفاده از سخت افزارهای سنگین و قدرتمند وجود ندارد.

جدول 4–17 مقایسه زمان اجرا در الگوریتم‌های مختلف بر حسب ثانیه

الگوریتم		ربات اول	ربات دوم	ربات سوم
ORB	مقیاس	۰.۰۳۶۱	۰.۱۷۸۱	۰.۳۵۲۹
	چرخش	۰.۰۴۷۳	۰.۲۷۴۴	۰.۵۷۴۹
	زاویه دید	۰.۰۴۱	۰.۲۲۵۵	۰.۴۳۶۴
SURF	مقیاس	۰.۲۴۲	۱.۱۰۲۲	۲.۴۱۴۹
	چرخش	۰.۴۲۵۱	۲.۰۵۴۷	۴.۴۰۴۹
	زاویه دید	۰.۳۵۰۳	۱.۶۱۱	۳.۵۴۲۸
SIFT	مقیاس	۰.۱۶۷۶	۱.۲۲۰۶	۴.۲۸۸۶
	چرخش	۰.۲۸۵۷	۲.۶۰۰۶	۷.۹۹۳۵
	زاویه دید	۰.۲۱۷۸	۱.۸۰۱۲	۶.۱۵۵۳
FAST	مقیاس	۰.۰۳۵۱	۰.۱۷۶۳	۰.۳۹۱۶
	چرخش	۰.۰۳۹۶	۰.۲۱۰۵	۰.۴۵۹۷
	زاویه دید	۰.۰۳۶۳	۰.۲۰۰۳	۰.۴۴۴۴
BRISK	مقیاس	۰.۳۷۷	۲.۱۰۰۵	۲.۹۹۶۶
	چرخش	۰.۴۳۴۸	۲.۴۰۷	۳.۴۱۳۱
	زاویه دید	۰.۳۹۱۳	۲.۲۹۰۹	۳.۲۲۷۳

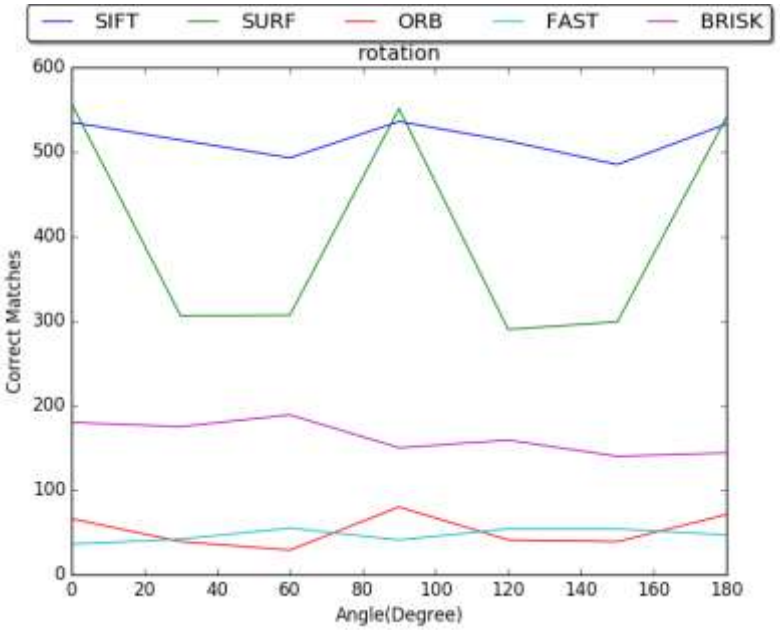
با توجه به رابطه 4–2 به وضوح می‌توان دید که محاسبه دقت بین رایانه‌های مورد نظر به قدرت پردازشی بستگی ندارد و صرفاً ساختار هر الگوریتم تعیین کننده دقت می‌باشد. برای مثال دقت ربات اول و ربات سوم با وجود اختلاف بسیار زیاد قدرت پردازشی رایانه‌های آنها، در الگوریتم SIFT و شرایط چرخش کاملاً یکسان است. در الگوریتم‌های ORB، SIFT و BRISK بیشترین و کمترین دقت، به ترتیب در شرایط مقیاس، چرخش و زاویه دید می‌باشد. در حالی که الگوریتم‌های SURF و FAST در شرایط چرخش تصویر ضعیف‌تر عمل کرده‌اند.

جدول 4–18 مقایسه دقت در الگوریتم‌های مختلف بر حسب درصد

الگوریتم		ربات اول	ربات دوم	ربات سوم
ORB	مقیاس	۱۸.۳۱	۱۸.۴۳	۱۸.۲۸۶
	چرخش	۱۰.۴۳	۹.۹۴۳	۹.۷۱۴۳
	زاویه دید	۸.۶۲۹	۸.۲۲۹	۸.۴۵۷۱
SURF	مقیاس	۴۵.۸۲	۴۶.۲۵۲	۴۵.۸۲۲
	چرخش	۳۱.۸۹	۳۱.۹۶	۳۱.۸۹۴
	زاویه دید	۳۷.۸۸	۳۸.۶۹	۳۷.۸۷۹
SIFT	مقیاس	۶۱.۷۵	۶۱.۸۱	۶۱.۷۳۸
	چرخش	۶۰.۳	۵۹.۷۴	۶۰.۲۸۴
	زاویه دید	۵۵.۶۴	۵۵.۷۵	۵۵.۶۳۹
FAST	مقیاس	۱۵.۰۳	۱۵.۱۹	۱۵.۰۲۵
	چرخش	۵.۵۲۹	۵.۲۶۳	۵.۵۲۹۴
	زاویه دید	۹.۹۳۳	۱۰.۰۳	۹.۹۳۲۸
BRISK	مقیاس	۱۶.۴۵	۱۶.۲۳	۱۵.۷۹۹
	چرخش	۱۳.۹۹	۱۳.۶۵	۱۳.۳۸۷
	زاویه دید	۱۱.۱۱	۱۰.۸۸	۱۰.۴۸۴

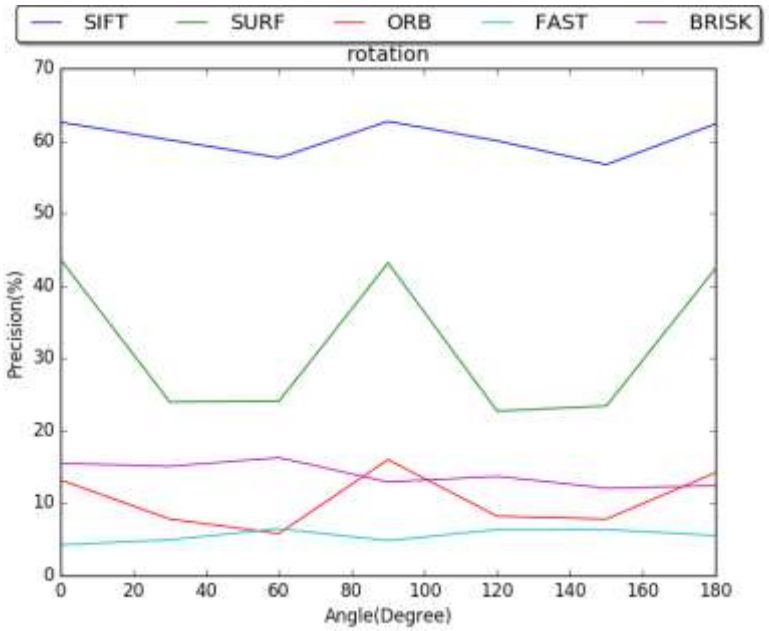
تعداد نقاط صحیح که در صورت کسر 4–2 آورده شده است از تصاویر چرخش یافته استخراج شده و در شکل 4–21 نمایش داده شده است. در شکل 4–21 رابطه بین زاویه چرخش و تعداد نقاط صحیح یافت شده توسط الگوریتم نمایش داده شده است.

در شکل 4–21 می‌توان مشاهده نمود که قوی‌ترین عملکرد در یافتن تعداد نقاط صحیح را الگوریتم SURF داشته است و الگوریتم FAST کمترین نقاط صحیح را در تصاویر یافته است و الگوریتم SURF بیشتر تغییرات را در زوایای غیر از 0، 90 و 180 را نیز داشته است.

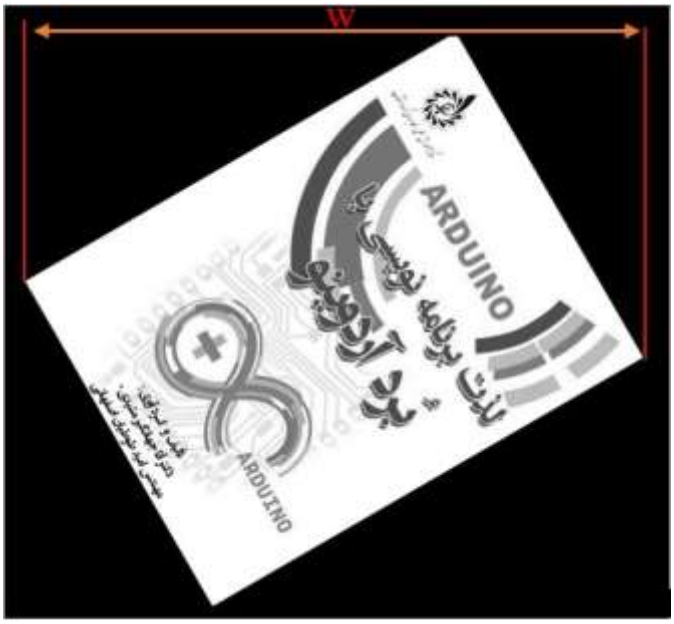


شکل 4–21 نسبت نقاط صحیح هر الگوریتم به زاویه چرخش تصویر

در شکل 4–22 دقت الگوریتم‌های مختلف در زوایای مختلف مورد بررسی قرار گرفت که دقت الگوریتم BRISK کمترین و SIFT بیشتر دقت را از خود نشان دادند. الگوریتم SURF نیز همانطور که مشاهده می‌شود در زوایای غیر از 0، 90 و 180 درجه، افت دقت شدیدی را دارد. این افت دقت در تمامی الگوریتم‌ها به چشم می‌خورد ولی شدت افت دقت در الگوریتم SURF بسیار زیاد است. دلیل این امر افزایش سایز ماتریس تصویر هنگام چرخش است همانطور که در شکل 4–23 نشان داده شده است زمانی تصویر در زاویه‌هایی غیر از زوایای 0، 90، 180 یا 270 چرخش پیدا می‌کند اندازه تصویر افزایش پیدا می‌کند با این وجود می‌توان از شکل 4–23 دریافت حساسیت کدام الگوریتم نسبت به سایز تصویر ورودی بیشتر و کدامیک کمتر است.

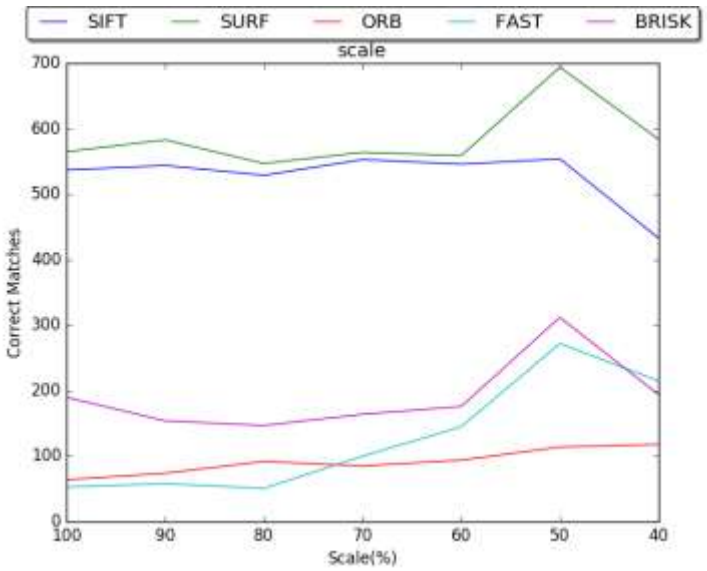


شکل 4–22 نمایش دقت الگوریتم‌ها در حالت چرخش عکس بر حسب زاویه



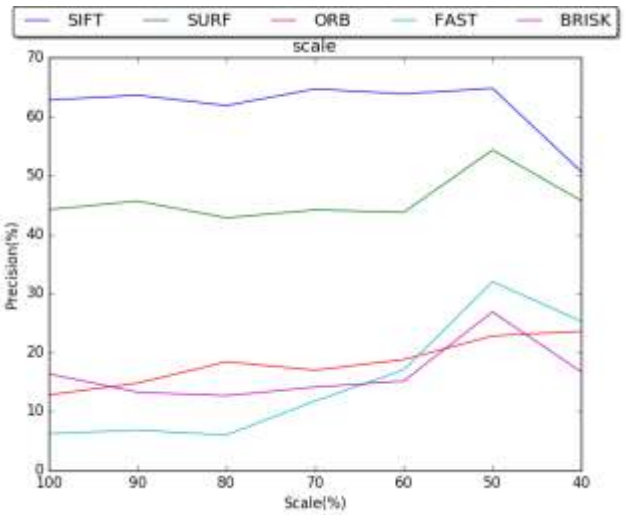
شکل 4–23 افزایش سایز تصویر بعد از چرخش

در شکل 4–24 نسبت تغییرات مقیاس به نقاط صحیح یافت شده نشان داده شده است. با وجود اینکه انتظار می‌رفت با کاهش مقیاس تصاویر، تعداد نقاط صحیح یافت شده نیز کاهش پیدا کند، ولی کاهش چندانی صورت نگرفته است. الگوریتم‌های SIFT و SURF در این شرایط نیز عملکرد بهتری را نسبت به دیگر الگوریتم‌ها داشتند و الگوریتم ORB و FAST در بین این الگوریتم‌ها کمترین تعداد نقاط صحیح را یافته اند و با کاهش مقیاس، کاهش شدیدی در تعداد نقاط صحیح یافت شده در این الگوریتم‌ها دیده نشده است.



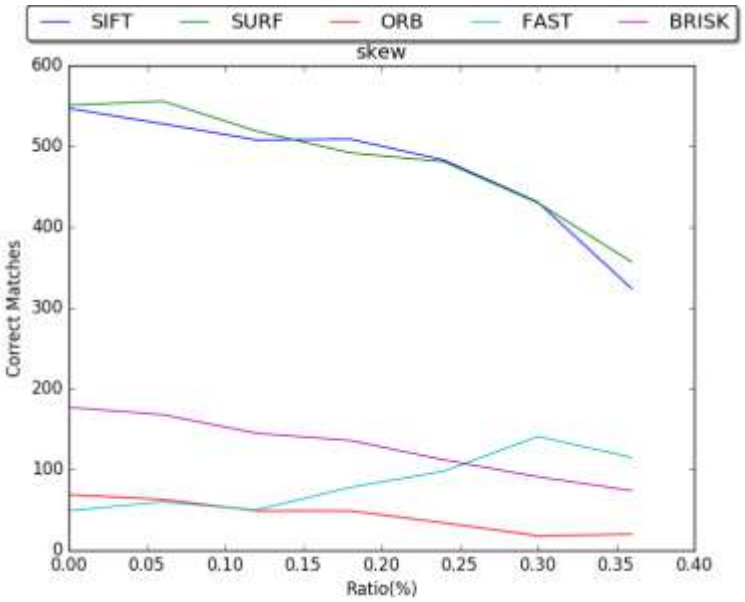
شکل 4–24 نمودار تعداد نقاط صحیح نسبت به مقیاس تصاویر در الگوریتم‌های مختلف

در شکل 4–25 می‌توان نسبت دقت هر الگوریتم به مقیاس تصاویر را مشاهده کرد و الگوریتم SIFT دقت بهتری را در تمامی مقیاس‌ها از خود نشان داده است. الگوریتم FAST همانگونه که پیش بینی می‌شد عملکرد نه چندان مطلوبی را در تغییرات مقیاس نسبت به دیگر الگوریتم‌ها داشته است.



شکل 4–25 نسبت دقت الگوریتم‌ها به مقیاس تصاویر

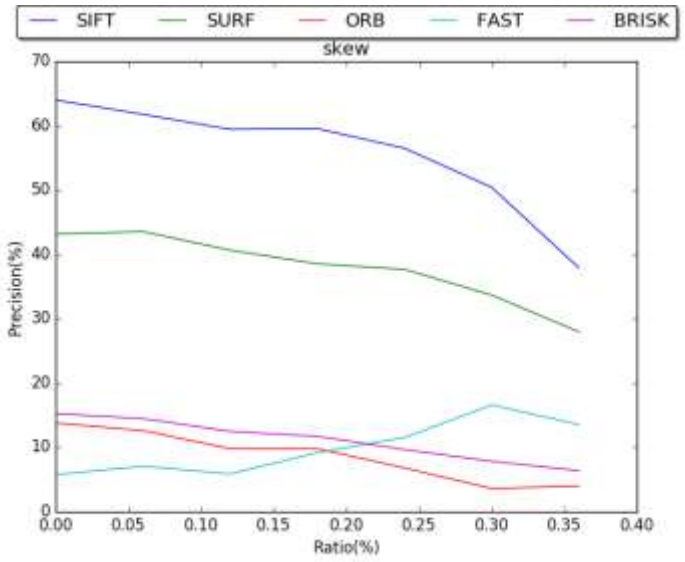
تغییرات زاویه دید در الگوریتم‌های SIFT و SURF بسیار مشابه و کاملا قابل قبول بوده است و همانطور که در شکل 4–26 می‌توان مشاهده کرد الگوریتم ORB و FAST کمترین تعداد نقطه صحیح را در بین الگوریتم‌های مورد بررسی در شرایط تغییرات زاویه دید داشته است.



شکل 4–26 نسبت تغییر زاویه دید و تعداد نقاط صحیح

دقت الگوریتم SIFT در تغییرات زاویه دید بیشتر از تمامی الگوریتم‌های مورد آزمایش بوده و دقت الگوریتم SURF در شرایط تغییر زاویه دید، مطلوب بوده است. اما با وجود اینکه تعداد نقاط صحیح یافت شده در الگوریتم ORB بسیار پایین بوده اما

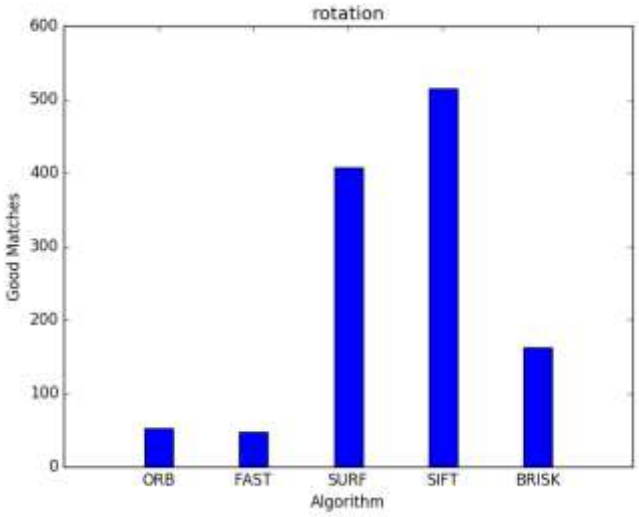
همانطور که در شکل 4–27 مشاهده می‌شود، دقت این نقاط یافت شده از الگوریتم FAST بیشتر بوده و بسیار نزدیک به BRISK بوده است.



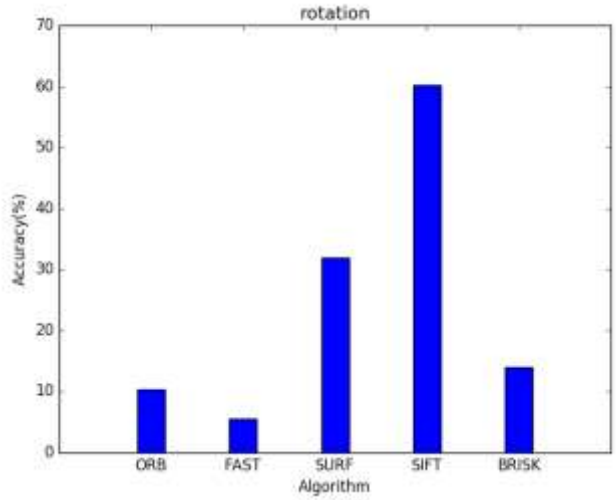
شکل 4–27 نسبت تغییرات زاویه دید و دقت هر الگوریتم

در ادامه میانگین پارامترهای مورد نظر در 7 تصویر را مورد بررسی قرار خواهیم داد. در شکل 4–28 میانگین تعداد نقاط تطابق یافته صحیح مقایسه شده اند که الگوریتم SIFT با 515 نقطه، بیشترین تعداد نقطه صحیح و FAST با 47 نقطه، کمترین تعداد نقاط صحیح را دارا بوده اند.

در شکل 4–29 میانگین دقت الگوریتم‌ها را تحت شرایط چرخش تصاویر، می‌توان ملاحظه نمود. الگوریتم SIFT، بیشترین دقت را میان الگوریتم‌های انتخابی در حالت چرخش تصاویر داشته است(60.3٪) و الگوریتم FAST، کمترین میزان دقت را در حالت چرخش تصاویر داشته است(5.52٪).

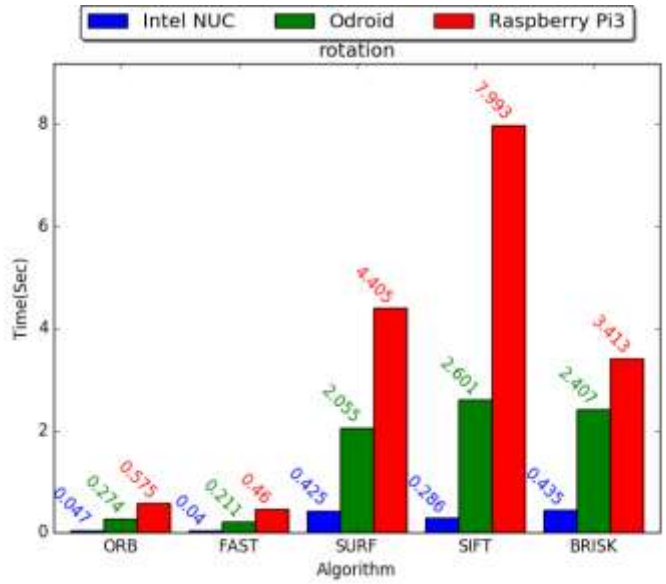


شکل 4–28 میانگین تعداد نقاط صحیح هر الگوریتم در شرایط چرخش تصاویر

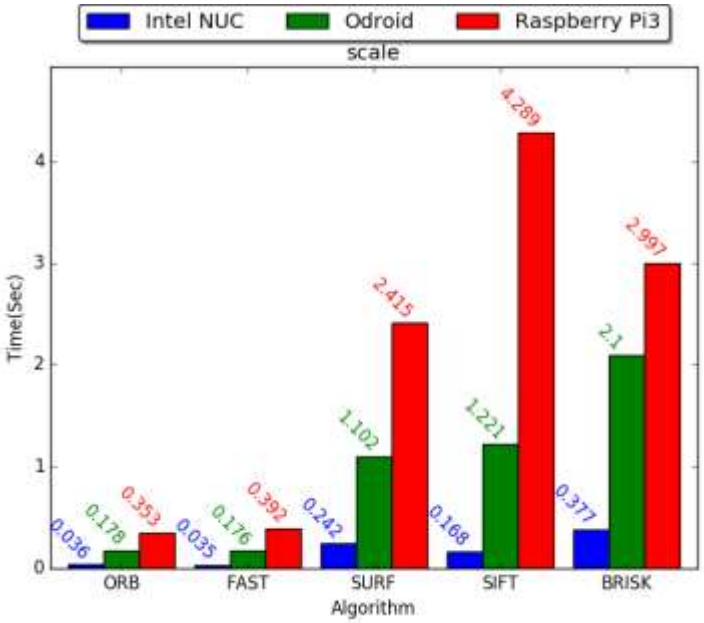


شکل 4–29 میانگین دقت الگوریتم‌های تحت آزمایش با شرایط چرخش تصاویر

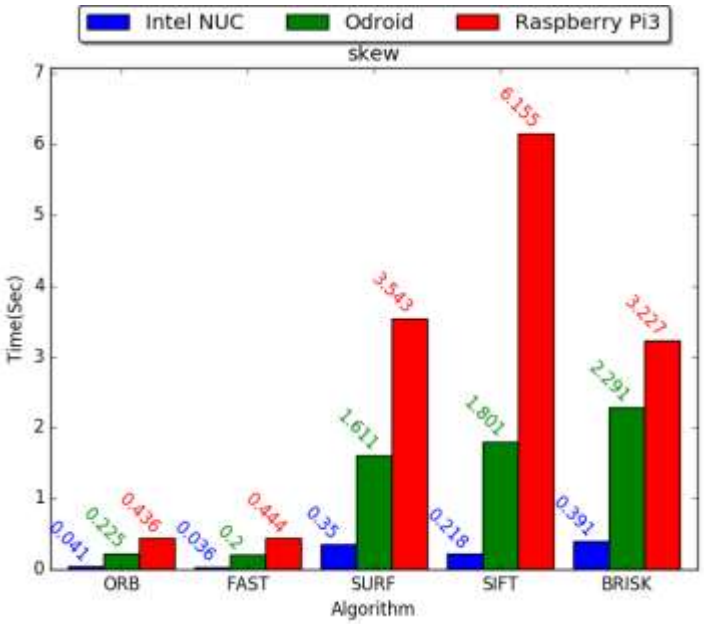
با مقایسه زمان محاسبه هر الگوریتم با یکدیگر می‌توان مشاهده کرد که الگوریتم ORB با زمان میانگین 0.047 ثانیه سریعترین الگوریتم در سخت افزار Intel NUC در حالت چرخش و الگوریتم BRISK با 0.43 ثانیه کندترین الگوریتم مورد بررسی بر روی Intel NUC بوده اند که در شکل 4–30 نمودار میله‌ای میانگین زمان هر الگوریتم بر روی هر سه سخت افزار در حالت چرخش تصاویر قابل مشاهده است و نمودار زمان پردازش این الگوریتم‌ها در شکل 4–31 و شکل 4–32 در شرایط تغییر مقیاس و تغییر زاویه دید نیز آورده شده است.



شکل 4–30 مقایسه میانگین زمان پردازش هر الگوریتم بر روی سخت افزارهای مختلف در شرایط چرخش تصویر

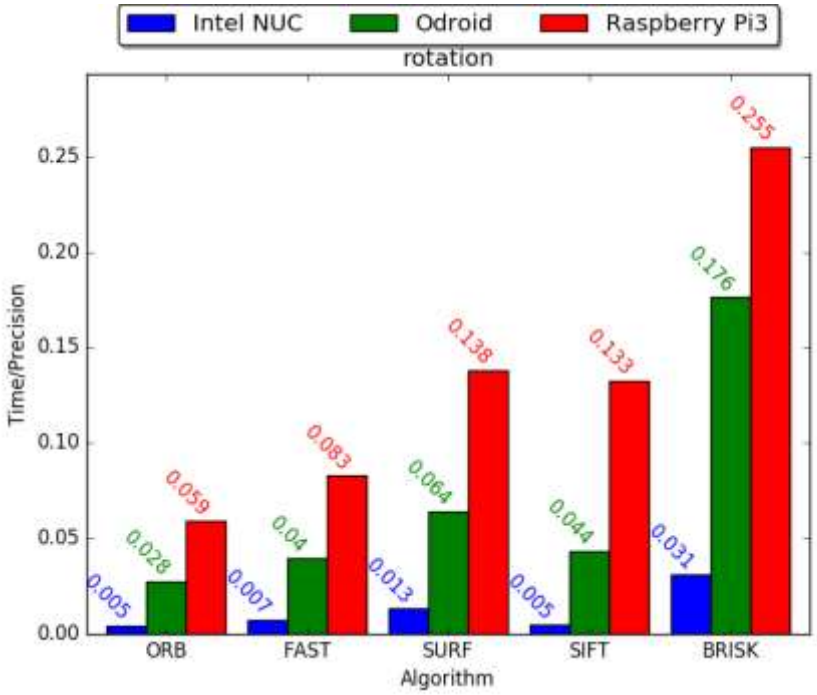


شکل 4–31 مقایسه میانگین زمان پردازش هر الگوریتم بر روی سخت افزارهای مختلف در مقیاس‌های مختلف

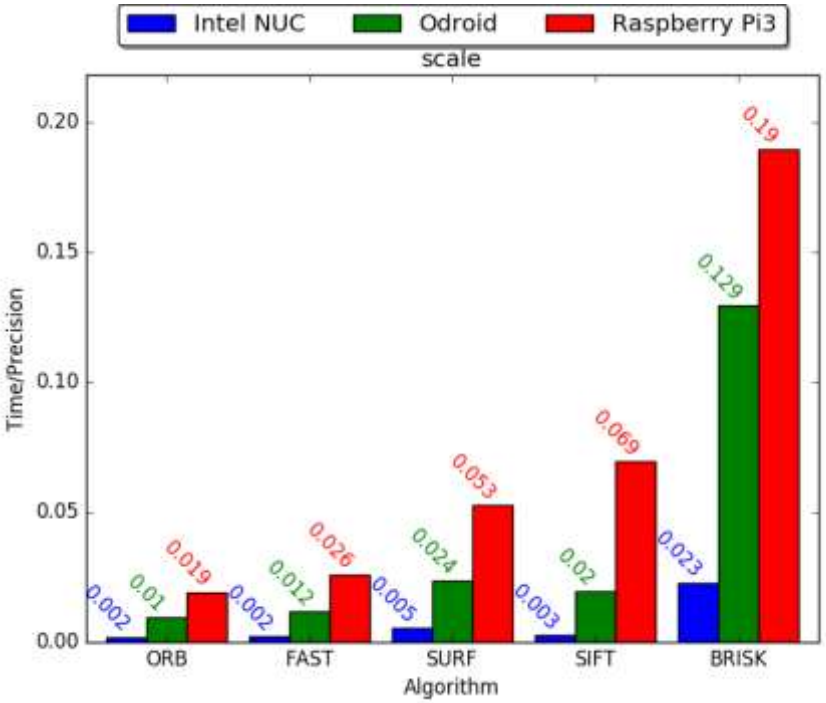


شکل 4–32 مقایسه میانگین زمان پردازش هر الگوریتم بر روی سخت افزارهای مختلف در حالت تغییر زاویه دید

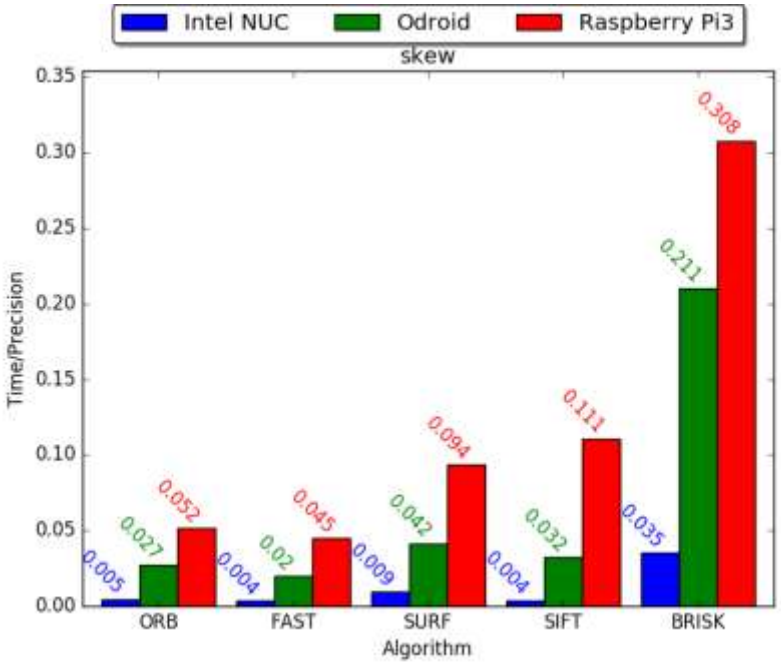
برای بدست آوردن پربازده‌ترین الگوریتم از نظر سرعت و دقت، تلاش شد میانگین زمان بدست آمده توسط هر الگوریتم بر دقت هر الگوریتم تقسیم شود و مقدار بدست آمده با یکدیگر مقایسه شود واضح است مقادیر نزدیک به صفر، بیانگر بازدهی بالاتر الگوریتم می‌باشد و همانطور که در شکل 4–33، شکل 4–34 و شکل 4–35 دیده می‌شود، الگوریتم ORB بیشترین بازدهی و الگوریتم BRISK کمترین بازدهی را از نظر دقت و زمان در بین الگوریتم‌های مورد بررسی دارا بوده‌اند.



شکل 4–33 مقایسه بازده هر الگوریتم بر روی سخت افزارهای مختلف در حالت چرخش تصاویر



شکل 4–34 مقایسه بازده هر الگوریتم بر روی سخت افزارهای مختلف در حالت تغییر مقیاس



شکل 4–35 مقایسه بازده هر الگوریتم بر روی سخت افزارهای مختلف در حالت تغییر زاویه دید

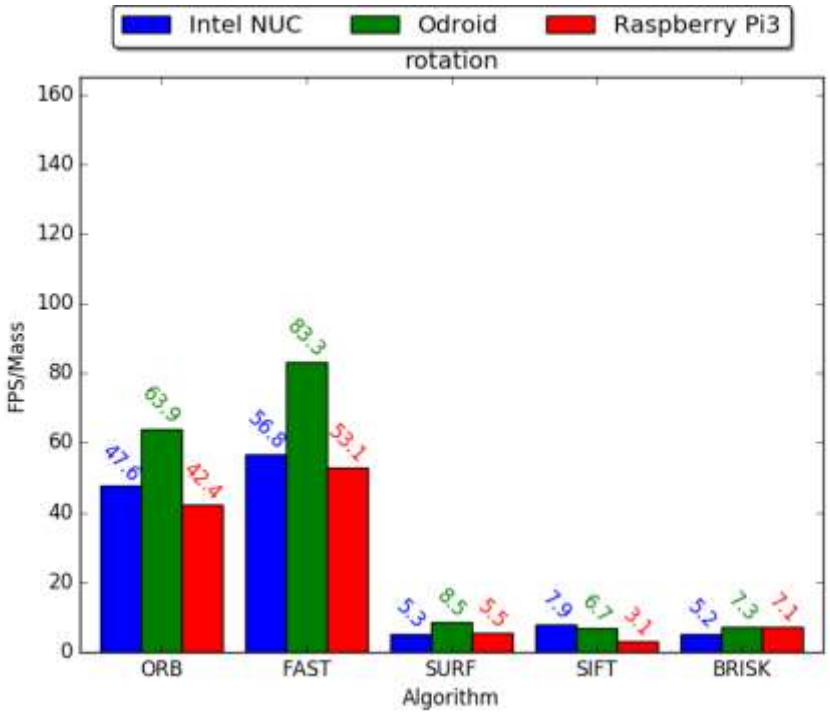
در ربات‌های پرنده همانطور که پیشتر نیز اشاره شد، ملاحظات مختلفی از جمله جرم تجهیزات، مصرف باطری تجهیزات پر اهمیت می‌باشند. که با درک بهتری از الگوریتم‌هایی که کاربران در حال توسعه آن‌ها هستند باید بهینه‌ترین الگوریتم و سخت افزار انتخاب شود برای این منظور fps خروجی هر الگوریتم با توجه به زمان پردازش آن محاسبه گردید و با اندازه‌گیری جرم هر ربات که در جدول 4–19 آمده است، نسبت fps هر الگوریتم به جرم هر ربات محاسبه گردید که در شکل 4–36 و شکل 4–37 و شکل 4–38 در شرایط چرخش، تغییرات مقیاس و تغییرات زاویه دید نمایش داده شده است.

جدول 4–19 جرم رایانه‌های مورد آزمایش

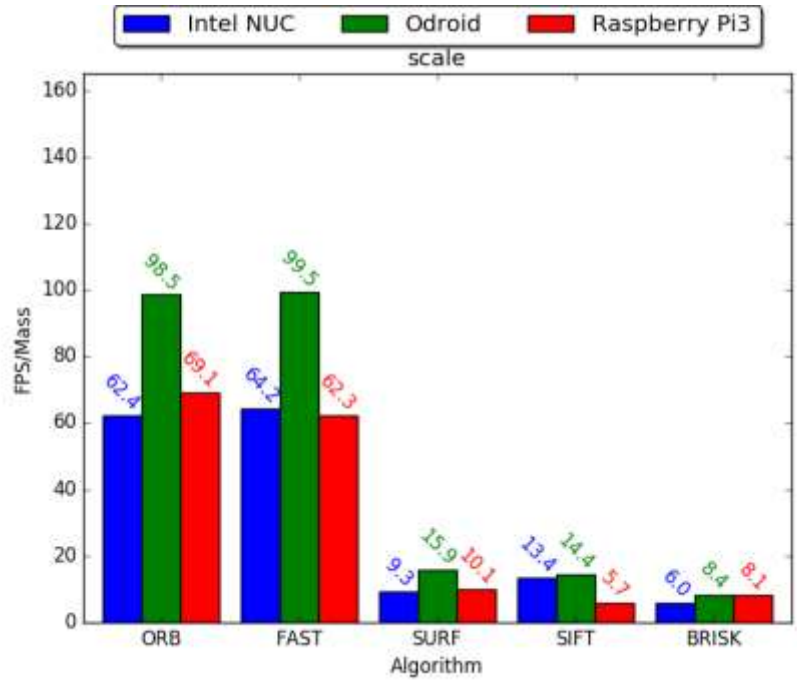
رایانه	جرم
Intel NUC	444 گرم
Odroid C2	57 گرم
Raspberrypi 3	41 گرم

الگوریتم‌های FAST و ORB نسبت به جرم رایانه‌های مورد آزمایش بیشترین fps را در اختیار گذاشتند و باید توجه داشت که الگوریتم FAST نسبت به چرخش اشیاء مستقل نیست و ORB این مشکل را ندارد. همانطور که شکل 4–36 و شکل 4–37

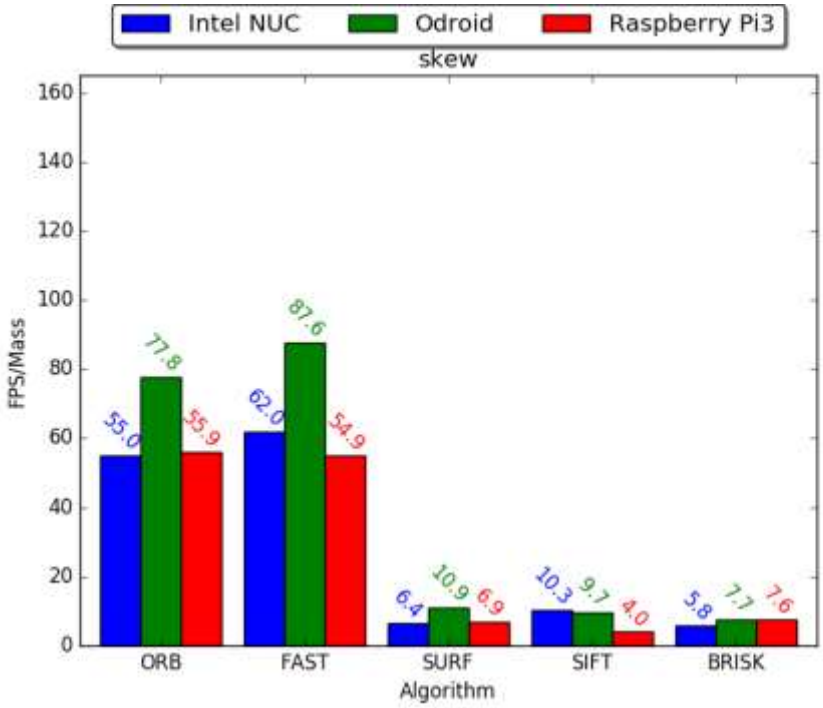
وشکل 4–38 نشان می‌دهند سخت افزار Odroid C2 نسبت به Intel NUC و Raspberry Pi 3 بازدهی بالاتری را از نظر نسبت fps به جرم از خود نشان داده است.



شکل 4–36 نسبت fps به جرم در شرایط چرخش تصاویر



شکل 4–37 نسب fps به جرم در شرایط تغییرات مقیاس



شکل 4–38 نسبت fps به جرم در شرایط تغییرات زاویه دید

4–4 تشخیص اشیاء بر اساس تحلیل ویدئویی تصاویر:

در این بخش تصاویر ویدئویی گرفته شده توسط ربات Ar.Drone 2.0 مورد بررسی قرار می‌گیرد این ربات به دلیل دارا بودن پایداری بسیار مناسب در پرواز، تصاویر قابل قبولی را برای تحلیل ویدئویی در اختیار می‌گذارد. پایداری این ربات امکان اندازه گیری فاصله ربات تا اشیاء مورد نظر را برای تحلیل بهتر نیز فراهم می‌کند.

4–4–1 پیش پردازش(کالیبراسیون):

در پردازش تصویر گاهی لازم است بر تصاویر خام دریافتی تغییراتی اعمال شود تا بتوان پارامترهای عددی مورد نظر را بدست آورد. صحت و اطمینان از اندازه‌گیری با استفاده از تصاویر می‌تواند تحت تاثیر فاکتورهای زیاد قرار گیرد. یکی از این فاکتورها، اعوجاج در تصاویر است. بنابراین کالیبره کردن تصاویر برای تکرارپذیری اندازه‌گیری و بهبود صحت اندازه‌گیری ضروری می‌باشد. در صورتی که اندازه‌گیری‌ها اصلاح نشوند مقایسه داده‌ها سخت می‌شود و تحلیل کمّی آن‌ها را با مشکل مواجه می‌سازد[۶۹].

قبل از اینکه تصاویر وارد الگوریتم‌ها شوند برای حذف اعوجاج از تصاویر باید مورد پیش‌پردازش و کالیبراسیون قرار گیرند. در شکل 4–39 می‌توان اعوجاج موجود در تصویر خام ربات را که خطوط صاف به صورت منحنی به نمایش در آمده است مشاهده کرد.

شکل 4–39 مقایسه تصویر خام ربات که درب را به صورت منحنی نمایش داده است با خط قرمز صاف

کاربرد دیگر عملیات کالیبراسیون این است که امکان استخراج داده های فاصله و اندازه اشیاء را بر حسب متر فراهم می کند. که در ادامه به آن خواهیم پرداخت.

برای کالیبره کردن تصاویر ربات، از یک صفحه شطرنجی 7*8 با اندازه مشخص استفاده شد. به منظور کالیبره کردن تصاویر در کتابخانه OpenCV، پیش از هر کاری باید تصاویری از صفحه شطرنجی توسط ربات گرفته می‌شد و پس از آن تصاویر مورد پردازش قرار میگرفت. در شکل 4–40 می‌توان صفحه شطرنجی یافت شده را مشاهده نمود.

شکل 4–40 شناسایی صفحه شطرنجی و موقعیت گوشه‌های آن

تعداد 50 عدد عکس در شرایط مختلف از صفحه شطرنجی گرفته شد تا بتوان دوربین ربات را با دقت کالیبره نمود تعدادی از این تصاویر در شکل 4–41 قابل مشاهده است.برای شناسایی صفحه شطرنجی در هر عکس از دستور `findChessboardCorners()` استفاده شد.

شکل 4–41 فرآیند کالیبراسیون دوربین ربات

برای بدست آوردن فاصله صفحه شطرنجی تا دوربین از رابطه 4–4 استفاده شد.

$$Distance\ to\ Object\ (mm) = \frac{f(mm) * Real\ Height(mm) * image\ height\ (pixels)}{Object\ Height\ (pixels) * Sensor\ Height\ (mm)} \qquad 4–4$$

در رابطه 4–4، f بیانگر فاصله کانونی، Real Height بیانگر ارتفاع واقعی جسم بر حسب میلی‌متر، Image Height بیانگر ارتفاع تصویر بر حسب پیکسل، Object Height بیانگر ارتفاع جسم در تصویر بر حسب پیکسل و Sensor Height نیز ارتفاع سنسور دوربین بر حسب میلی‌متر می‌باشد. باید توجه داشت اکثر دوربین‌های موجود در بازار معمولا اطلاعتی مانند ارتفاع سنسور و فاصله کانونی نا مشخص است در این پژوهش با علم به ثابت بودن فاصله کانونی دوربین ارتفاع سنسور دوربین، این دو پارامتر ثابت در نظر گرفته شدند.

برای محاسبه نسبت فاصله کانونی و ارتفاع سنسور از صفحه شطرنجی استفاده شد به این صورت که با قرار دادن صفحه شطرنجی در فاصله ای معین و بدست آوردن تمامی متغیرهای رابطه 4–4، نسبت فاصله کانونی به ارتفاع سنسور محاسبه شد. در شکل 4–42 فرآیند محاسبه نسبت فاصله کانونی به ارتفاع سنسور را می‌توان دید.

شکل 4–42 محاسبه پارامترهای دوربین ربات با استفاده از صفحه شطرنجی

پس از محاسبه پارامترهای دوربین ربات شرایط برای کالیبره کردن دوربین ربات فراهم می‌شود و خروجی برنامه کالیبراسیون تصاویر مناسبی را برای تعیین فاصله و اندازه اجسام فراهم می‌کند. در شکل 4–43 تصویر خام دوربین ربات و تصویر کالیبره شده ربات نشان داده شده است.

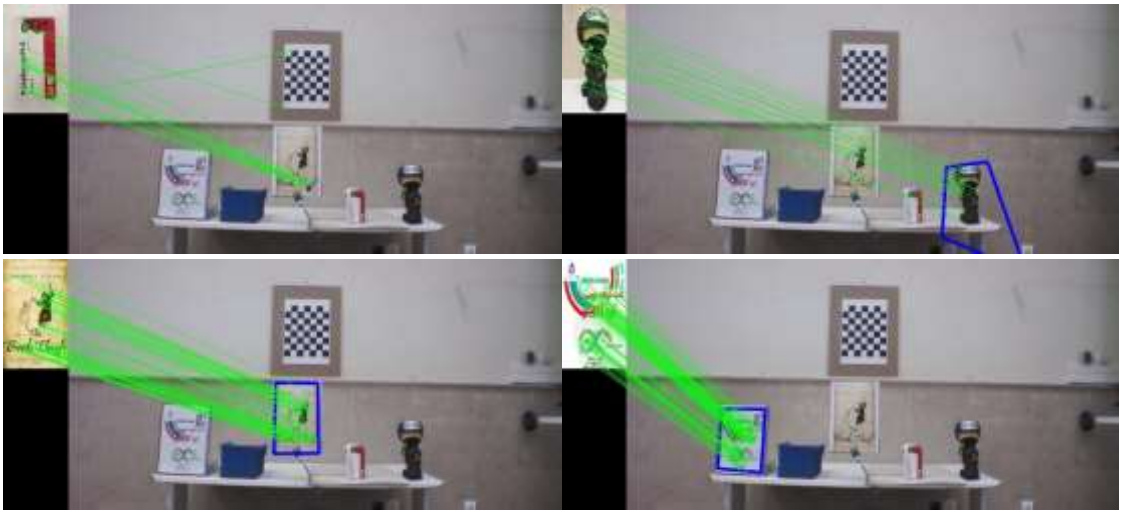
شکل 4–43 مقایسه تصویر کالیبره شده(راست) و تصویر خام(چپ)

بعد از کالیبره کردن دوربین ربات، این تصاویر وارد الگوریتم‌های تشخیص شی شدند و در فواصل مختلف مورد بررسی قرار گرفتند. برای این کار تصاویری در فواصل مختلف از میز اشیا گرفته شد و به الگوریتم‌ها داده شد تا بررسی شودکه اشیا تا چه فاصله‌ای قابل تشخیص هستند. برای این آزمایش، چهار شی در نظر گرفته شده که مجسمه، کتاب دزد، کتاب آدوینو و جعبه رسپبری پای را شامل شد که در شکل 4–44 نشان داده شده اند.

جعبه رسپبری‌پای　　مجسمه　　کتاب دزد　　کتاب آردوینو

شکل 4–44 اشیا تحت آزمایش فاصله

قابل ذکر است درصد موفقیت الگوریتم در تشخیص اشیا کاملا به نور محیط، درخشندگی جسم اندازه جسم وابسته است. در این آزمایش، از فاصله 150 سانتی متری و هر 50 سانتی متر تا فاصله 400 سانتی متری تصویر گرفته شد و با توجه به اینکه الگوریتم FAST یک شناساگر است و نسبت به چرخش اشیا مستقل نیست از 4 الگوریتم دیگر شامل SIFT، SURF، ORB و BRISK برای آزمایش استفاده شد. در شکل 4–45 خروجی الگوریتم SIFT در فاصله 150 سانتی متری اشیا نشان داده شده است.



شکل 4–45 خروجی الگوریتم SIFT در فاصله 150 سانتی متری اشیا

همانطور که مشاهده می‌شود الگوریتم SIFT در یافتن جعبه Raspberry pi دچار اشتباه شده است ولی در تشخیص اشیا دیگر بسیار عالی عمل کرده که این موضوع می‌تواند به دلیل اندازه کوچک جعبه رسپبری باشد. خروجی الگوریتم‌ها در تشخیص اشیا در جدول 4–20 نمایش داده شده است.

جدول 4–20 نتایج تشخیص اشیا در فواصل 150 تا 400 سانتی‌متری

	BRISK				ORB				SURF				SIFT			
فاصله (cm)	مجسمه	جعبه	آردوینو	دزد	مجسمه	جعبه	آردوینو	دزد	مجسمه	جعبه	آردوینو	دزد	مجسمه	جعبه	آردوینو	دزد
150	✗	✗	✗	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓
200	✗	✗	✗	✓	✓	✗	✓	✗	✓	✗	✓	✓	✗	✓	✓	✓
250	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✓
300	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓
350	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗
400	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✓

4-4-2 پردازش ویدئویی:

برای آزمایش دقت الگوریتم‌های تشخیص اشیا هر فریم[1] از فیلم می‌بایست مورد پردازش قرار گیرد. به منظور ارزیابی و نتیجه‌گیری مناسب‌تر از عملکرد الگوریتم‌ها فیلم یک دقیقه‌ای تهیه شد و توسط الگوریتم‌های SIFT، SURF، ORB و BRISK پردازش شد. خروجی پردازش هر الگوریتم، به صورت عکس ذخیره شد به این صورت که با ضبط ویدئوی 1 دقیقه‌ای با سرعت 30 فریم بر ثانیه(خروجی دوربین ربات) در کل 1832 عکس از ویدئوی تصویر برداری شده قابل استخراج است. با احتساب این 1832 عکس برای هر الگوریتم، در کل 7328 عکس به عنوان مجموعه داده ذخیره گردید. پس از ذخیره خروجی الگوریتم‌ها، یک ناظر تمامی تصاویر را از نظر صحت مورد بررسی قرار خواهد داد سپس با استفاده از معیارهای ارزیابی صحت و بازخوانی و امتیاز F1 دقت هر الگوریتم بررسی شد.

- **معیارهای ارزیابی صحت و بازخوانی و امتیاز F1:**

به منظور ارزیابی عملکرد الگوریتم‌ها، خروجی هر الگوریتم بر روی فایل جداگانه‌ای ذخیره خواهد شد.

نتایج ارزیابی هر الگوریتم را می‌توان به صورت زیر دسته بندی کرد:

- ✓ درست مثبت[2] – TP
- ✓ درست منفی[3] – TN
- ✓ نادرست مثبت[4] –FP (خطای نوع یک)
- ✓ نادرست منفی[5] – FN (خطای نوع دو)

این معیارهای ارزیابی را در این پژوهش می‌توان به صورت زیر تعریف کرد:

- ✓ TP: هدف در تصویر باشد تشخیص داده شده باشد
- ✓ TN: هدف در تصویر نباشد و تشخیص داده نشده باشد
- ✓ FP: هدف در تصویر نباشد و تشخیص داده شده باشد

---

[1]- Frame

[2]- True Positive

[3]- True Negative

[4]- False Positive

[5]- False Negative

✓ FN: هدف در تصویر باشد و تشخیص داده نشده باشد

این نوع دسته بندی از معتبرترین دسته‌بندی‌ها در علم آمار و همچنین کلان داده‌ها[1] می‌باشد. در جدول 4–21 ماتریس درهم ریختگی و مدل دسته بندی نشان داده شده است. ماتریس درهم ریختگی یا ماتریس اختشاش در حقیقت، توزیع دسته‌ها را از لحاظ درستی با نادرستی نمایش می‌دهد. جدول 4–21 یک ماتریس دو در دو را برای یک مسئله دسته بندی دوگانیِ وجود داشتن یا نداشتن نشان می‌دهد. در حالت کلی برای یک مسئله دسته بندی N حالته لازم است یک ماتریس N*N رسم شود. در واقع بهترین الگوریتم الگوریتمی است که مقادیر FN و TN آن به صفر نزدیک‌تر است یعنی تشخیص‌های غلط کمتری را انجام داده است.

جدول 4–21 ماتریس درهم ریختگی

		نتایج واقعی (تشخیص جسم در تصویر)	
		مثبت	منفی
نتایج مورد انتظار (وجود شی در تصویر)	مثبت	TP	FN
	منفی	FP	TN

- **معیار صحت:**

این معیار از نسبت تعداد تشخیصهای درست مثبت به کل تشخیص های مثبت محاسبه می‌شود. هرچه این سنجش به صفر نزدیکتر باشد، الگوریتم ضعیف تر عمل کرده است و همانگونه که رابطه 4–5 نشان می‌دهد هر چه مقدار دقت به 1 نزدیک‌تر باشد، یعنی نرخ نادرست‌های مثبت کمتر می‌باشد.

$$Precision = \frac{TP}{TP + FP} \qquad 5–4$$

[1]- Big Data

- **بازخوانی:**

این معیار از تعداد تشخیصهای درست مثبت به کل تشخیصهای مثبت واقعی محاسبه می‌شود. در دسته بندیهای دوگانه از این پارامتر به نام حساسیت نام برده می شود. هرچه این پارامتر به صفر نزدیک‌تر باشد الگوریتم ضعیف‌تر عمل کرده است. همانگونه که از 4–6 مشخص است هرچه مقدار بازخوانی به 1 نزدیکتر باشد یعنی نرخ نادرستهای منفی کمتر می‌باشد.

$$Recall = \frac{TP}{TP + FN} \qquad 6–4$$

- **معیار امتیاز F :**

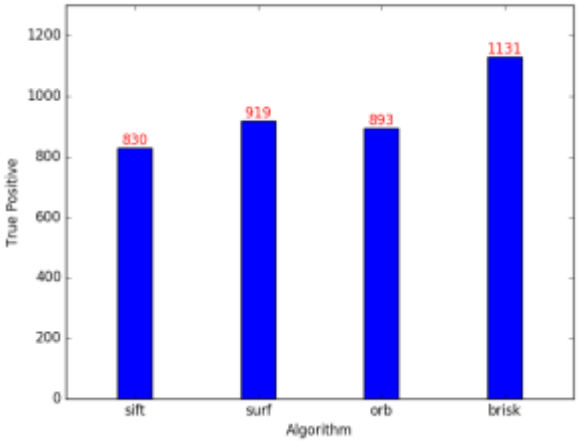
این معیار، میانگین هارمونی برای دو مقدار بازخوانی و صحت می باشد. همانطور که در 4–7 مشاهده می کنید اگر یکی از دو مقدار عددی کوچک یا حتی صفر باشد، نتیجه نهایی عددی کوچک و یا صفر خواهد بود. چون دو معیار بازخوانی و صحت اعدادی بین صفر تا یک هستند و در صورت کسر در هم دیگر ضرب شده اند بنابراین نتیجه نهایی به سمت عدد کوچکتر، متمایل خواهد بود و اگر هر دو با هم، عددی بزرگ (نزدیک 1) باشند، نتیجه نهایی به سمت یک حرکت خواهد کرد.

$$F = 2 * \frac{Precision * Recall}{Precision + Recall} \qquad 7–4$$

4–4–3 نتایج پردازش ویدئویی:

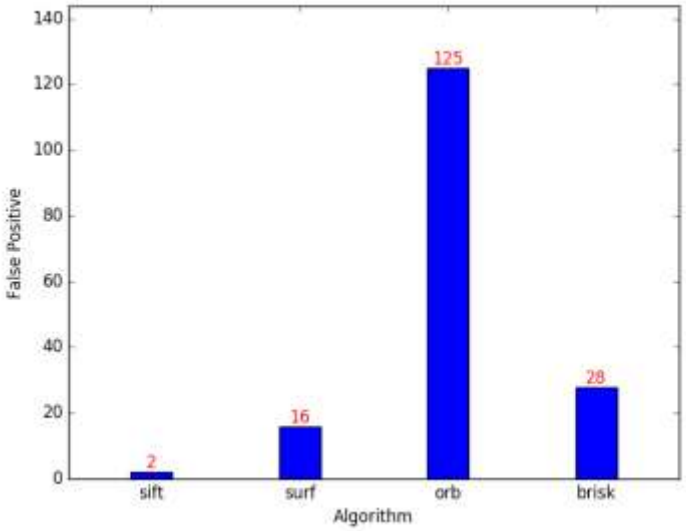
در این بخش خروجی الگوریتم‌ها که شامل 7328 عکس بود به صورت نظارت بر طبقه بندی[1] انجام گرفت.

در شکل 4–46، شکل 4–47، شکل 4–48 و شکل 4–49 به ترتیب نتایج TP، FP، TN و FN آورده شده است.

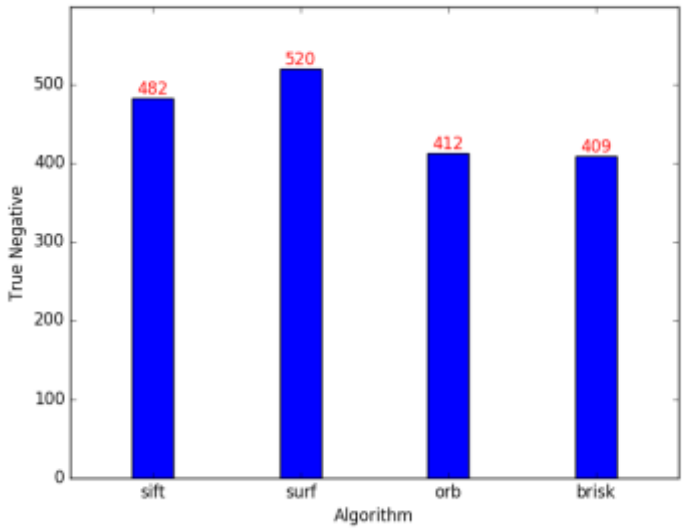


شکل 4–46 نتایج درست مثبت در پردازش ویدئویی

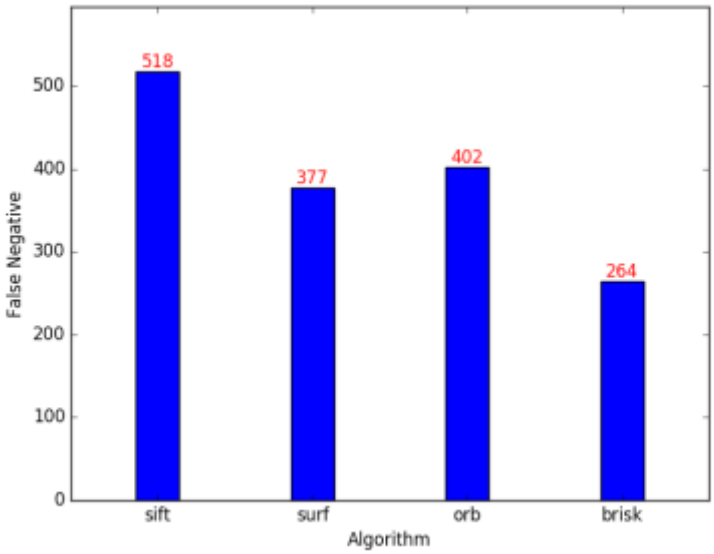
[1]- Supervised Calssification



شکل 4–47 نتایج نادرست مثبت در پردازش ویدئویی

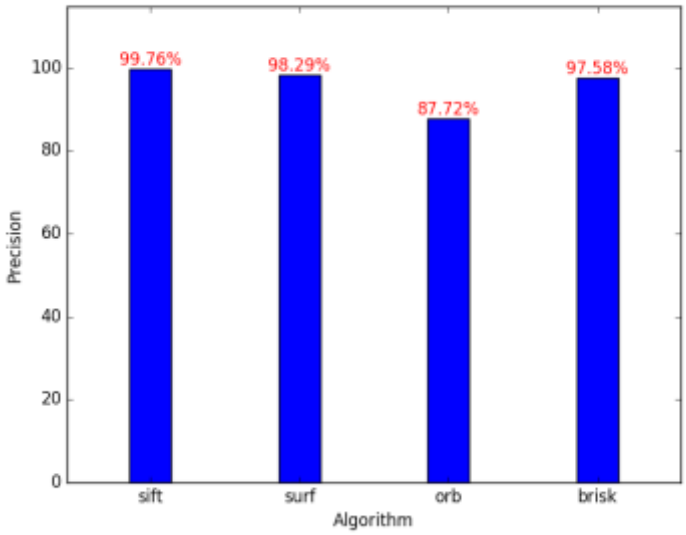


شکل 4–48 نتایج درست منفی در پردازش ویدئویی



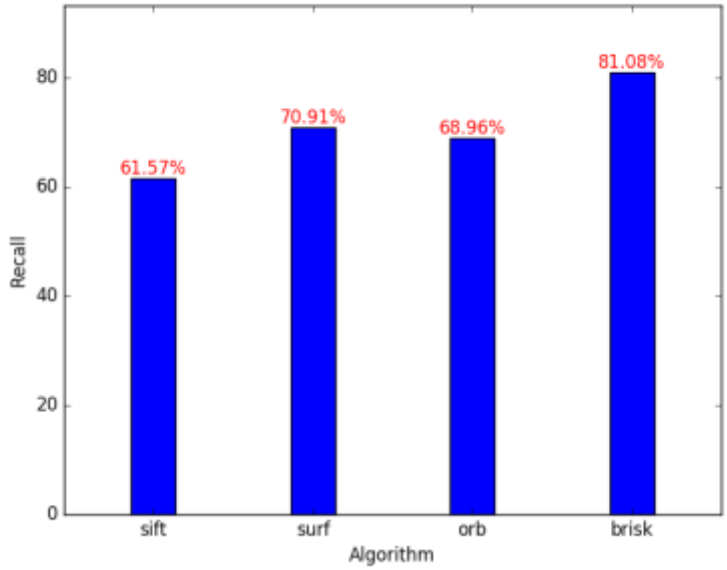
شکل 4–49 نتایج نادرست منفی در پردازش ویدئویی

نتایج دقت الگوریتم‌های در شکل 4–50 نمایش داده شده است می‌توان دید با وجود اینکه تعداد نتایج نادرست منفی در الگوریتم SIFT زیاد به نظر می‌رسد ولی بالاترین دقت را به خود اختصاص داده است. الگوریتم ORB نیز کمترین دقت را بین الگوریتم‌های مورد بررسی دارا بود.



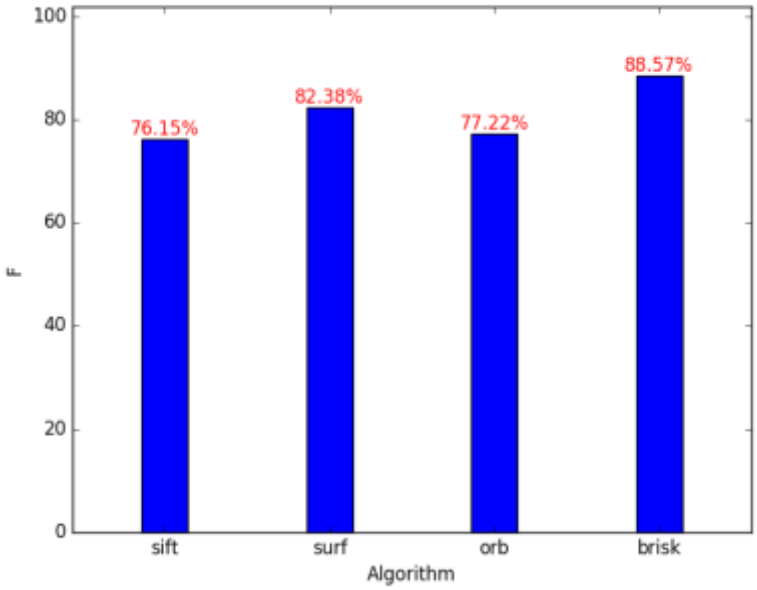
شکل 4–50 نتایج دقت الگوریتم‌ها در پردازش ویدئویی

در شکل 4–51 نیز نمودار بازخوانی نمایش داده شده است.الگوریتم SIFT به دلیل اینکه تعداد نادرست‌های منفی بیشتری دارد در این ارزیابی چندان عملکرد خوبی را نمایش نداده است.



شکل 4–51 نتایج بازخوانی الگوریتم‌ها در پردازش ویدئویی

در نهایت معیار ارزیابی F بررسی شد که نتایج آن در شکل 4–52 نمایش داده شده است.



شکل 4–52 نتایج معیار F الگوریتم‌ها در پردازش ویدئویی

5- فصل پنجم: نتیجه گیری و پیشنهادات

5–2 نتیجه گیری:

در این پژوهش تلاش شد با پیاده سازی الگوریتم‌های تشخیص اشیا بر روی سخت افزارهای مختلف عملکرد سخت افزارها در مقابل الگوریتم‌های مختلف مورد بررسی قرار بگیرد تا در نهایت بتوان با داشتن اطلاعات مناسب در مورد عملکرد هر الگوریتم بر روی این سخت افزارها، پژوهشگران بتوانند متناسب با کاربری خود اقدام به ساخت و تجهیز ربات پرنده خود کنند. نتیجه‌ای که از این بحث حاصل گردید آن است که الگوریتم‌های تشخیص و استخراج ویژگی به دلیل محدودیت‌های منابع محاسباتی به شدت در ارتباط با حوزه تشخیص ویژگی‌های پیچیده و تکرارشونده، چالش‌پذیر و پر مسئله هستند.

بهره‌گیری از مرحله پیش پردازش در تشخیص اشیا، در راستای کم کردن تعداد مقایسات و تطبیق‌های انجام شده، بر مبنای تخمین رنگ، شکل و اندازه شی با داشتن فاصله ربات تا اشیا، منجر به بهبود سرعت و دقت تشخیص خواهد شد.

با توجه به داده‌های خروجی در این پژوهش، الگوریتم SIFT بدون توجه به زمان پردازش، قابل اطمینان‌ترین الگوریتم مورد آزمایش و ORB بیشترین بازده را نسبت به دیگر الگوریتم‌ها داشتند. در صورتی که قدرت پردازشی مناسبی در اختیار نیست، الگوریتم SIFT در زمینه پردازش‌های ویدئویی بلادرنگ انتخاب مناسبی نخواهد.

در این حالت ORB و BRISK هم قابلیت اطمینان مناسب و هم سرعت پردازش مناسبی را از خود نشان داده‌اند که این الگوریتم‌ها را الگوریتم مناسبی برای کاربری ربات‌های پایه متحرک مانند ربات‌های پرنده می‌کند و همانطور که شکل 4–38 نشان می‌دهد الگوریتم ORB بازدهی جرمی(نسبت FPS خروجی به جرم سخت افزارهای مورد آزمایش) مناسبی را از خود نشان داده است. البته نباید از یاد برد که الگوریتم FAST به تنهایی قابلیت تشخیص اشیا را ندارد و باید در کنار یک توصیفگر مورد استفاده قرار گیرد. علاوه بر این Odroid از نظر بازدهی جرمی به مراتب بازدهی جرمی بالاتری نسبت به Intel NUC از خود نشان داده است و نکته جالب اینکه حتی بازدهی جرمی Raspberry pi دو برابر بازدهی جرمی Intel NUC در الگوریتم ORB بوده است . جرم رایانه‌های مورد آزمایش در جدول 4–19 آمده است و لازم به ذکر است Intel NUC، بدون جعبه فلزی بر روی ربات 1 نصب شده است و جرم این ربات بدون جعبه و با احتساب رگلاتور ولتاژ در این جدول آورده شده است.

در مورد سخت افزار نیز با توجه به اینکه Intel NUC قدرت پردازشی بسیار عالی از خود نشان داد با توجه به بازدهی وزنی این رایانه، Intel NUC وزن مناسبی برای ربات‌های پرنده ندارد و کاربری ربات‌های زمینی یا هواپیماهای بال ثابت برای این سخت افزار بسیار مناسب‌تر پیشنهاد می‌شود جدای از وزن این رایانه، توان مصرفی این سخت افزار نسبت به سخت افزارهای جدید ساخته شده شرکت Intel مانند Up Board با توان 15 وات، بسیار بالاتر بوده و سخت افزار Intel NUC برای تشخیص اشیا در ربات‌های پرنده پیشنهاد نمی‌شود. قابل ذکر است که با توجه به اینکه ربات‌های پرنده زیر مجموعه از از ربات‌های پایه متحرک محسوب می‌شوند، ملاحظات وزنی در انتخاب سخت افزار پردازشی ربات‌های پرنده را می‌توان به ربات‌های پایه متحرک دیگر مانند ربات‌های ROV تعمیم داد.

5–3 پیشنهادات:

- افزایش دقت الگوریتم‌ها با استفاده از محدودیت های هندسی یا الگوریتم‌های تشخیص هندسه[1]
- استفاده از سخت افزارهای جدیدتر و سبک‌تر مانند Intel Up Board
- استفاده از گیمبال[2] به عنوان لرزشگیر و متعادل سازی دوربین ربات
- در صورتی که از پردازش تصویر برای فرود دقیق ربات استفاده می‌شود، اکیدا توصیه می‌شود تصاویر انتخابی از ARuco Code ها باشند در این صورت با توجه به آزمایش‌های انجام شده، سرعت پردازش تصاویر شدیدا افزایش خواهند یافت به علاوه تشخیص محل و جهت فرود با خطای کمتری نیز مواجه خواهد شد.

5–4 پژوهش‌های آینده:

- نقشه برداری دوبعدی و سه بعدی به صورت بلادرنگ
- تعیین موقیت ربات نسبت به مکان معین با استفاده از دوربین نصب شده بر روی ربات
- پیاده سازی کنترلP ، PID و شبکه عصبی برای فرود خودکار ربات
- ناوبری ربات بدون استفاده از GPS و با استفاده از پردازش تصویر یا تعقیب اشیا
- پیاده سازی سیستم‌های اجتناب از برخورد با استفاده از پردازش تصویر

[1]- Shape Detection

[2]- Camera Gimbal

6- فهرست منابع

[1] L. Kuhnert, “object localization on agricultural areas using an autonomous team of cooperation ground and air robots,” 2012.

[2] N. Michael, “Collaborative mapping of an earthquakedamaged building via ground and aerial robots,” 2012.

[3] C. Luo, “Multi-robot search and rescue team,” Pages 296–, 2011.

[4] “NEW YORK CITY DRONE FILM FESTIVAL,” *NEW YORK CITY DRONE FILM FESTIVAL*. 2015.

[5] D. Mellinger, “Control of quadrotors for robust perching and landing.”

[6] B. C. B Vergouw, H Nagel, G Bondt, “Drone Technology: Types, Payloads, Applications, Frequency Spectrum Issues and Future Developments,” *The Future of Drone Use*, http://www.springer.com/978-94-6265-131-9, 2016.

[7] Parrot, “PRESENTATION OF THE PARROT DISCO,” *Catalogue*. Available at: https://parrotcontact.parrot.com/website/user-guides/download-user-guides.php?pdf=disco-fpv/Disco-fpv_User-guide_UK.pdf.

[8] G. M. Hoffman, “Quadrotor helicopter trajectory tracking control,” 2008.

[9] D. Mellinger, “Trajectory generation and control for preciseaggressive maneuvers with quadrotors.”

[10] C. Snow, “Quick Start Guide to Construction.”

[11] “2018 DRONE MARKET SECTOR REPORT,” 2019.

[12] “phantom.” Available at: https://www.dji.com/phantom-4.

[13] A. Intwala, “A Review on Vertical Take Off and Landing (VTOL) Vehicles,”

[14] “latitude.” Available at: https://www.latitudeengineering.com/2014/07/hq-40-test-flights.

[15] OwlCity, “single copter.” Available at: https://www.rcgroups.com.

[16] R. K. Rout R. Kumar, “ A Survey on Object Detection and Tracking Algorithms,”

[17] Adam Vyskovsky, “Object tracking by a flying drone,” Department of Theoretical Computer Science and Mathematical Logic, 2014.

[18] B. Thesis, “Adam Vyˇ skovsk´ y Object tracking by a flying drone,” 2014.

[19] L. Geng, Y. F. Zhang, J. J. Wang, J. Y. H. Fuh, و S. H. Teo, “Mission planning of autonomous UAVs for urban surveillance with evolutionary algorithms,” *IEEE International Conference on Control and Automation, ICCA*, 2013.

[20] S. R. Herwitz “Imaging from an unmanned aerial vehicle: Agricultural surveillance and decision support,” *Comput. Electron. Agric.*, 2004.

[21] G. Grenzdörffer, A. Engel, B. Teichert, “The photogrammetric potential of low-cost UAVs in forestry and agriculture,” *International Archives of Photogrammetry Remote*

Sensing and Spatial Information Sciences Vol. XXXVII. Part B1, 2008.

[22] A. R. Girard, A. S. Howell, J. K. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles," 43*rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601),* 2004.

[23] V. Ramanathan "Warming trends in Asia amplified by brown cloud solar absorption," *Nature,* 2007.

[24] M. Nagai, T. Chen, R. Shibasaki, H. Kumagai, A. Ahmed, "UAV-borne 3-D mapping system by multisensor integration," *IEEE Trans. Geosci. Remote Sens.*

[25] K. T. Tomáš et al, "AR-Drone as a Platform for Robotic Research and Education.," *Int. Conf. Res. Educ. Robot.,* 2011.

[26] T. Krajník, V. Vonásek, D. Fišer, J. Faigl, "AR-drone as a platform for robotic research and education," *Communications in Computer and Information Science,* 2011.

[27] Y. J. Zhao H. L. Pei, "Improved Vision-Based Algorithm for Unmanned Aerial Vehicles Autonomous Landing," *Appl. Mech. Mater*.

[28] Engel, "Autonomous Camera-Based Navigation of a QuadCopter," 2014.

[29] R. D'Andrea, "Flying machine arena."

[30] E. H. C. Harik, F. Guerin, F. Guinand, J.-F. Brethe, H. Pelvillain, A. Zentout, "Vision based target tracking using an unmanned aerial vehicle," *2015 IEEE Int. Work. Adv. Robot. its Soc. Impacts*.

[31] S. Yang, S. A. Scherer, K. Schauwecker, A. Zell, "Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision," *J. Intell. Robot. Syst. Theory Appl.,* 2014.

[32] T. Merz, S. Duranti, G. Conte, "Autonomous landing of an unmanned helicopter based on vision and inertial sensing," *Springer Tracts Adv. Robot.,* 2006.

[33] R. Polvara, S. Sharma, J. Wan, A. Manning, R. Sutton, "Vision-Based Autonomous Landing of a Quadrotor on the Perturbed Deck of an Unmanned Surface Vehicle," *Drones*.

[34] G. E. C. Guevarra, A. R. T. Koizumi, J. N. B. Moreno, J. C. B. Reccion, C. M. O. Sy, و J. R. B. Del Rosario, "Development of a quadrotor with vision-based target detection for autonomous landing," *J. Telecommun. Electron. Comput. Eng*.

[35] T. Hoang, E. Bayasgalan, Z. Wang, G. Tsechpenakis, D. Panagou, "Vision-based target tracking and autonomous landing of a quadrotor on a ground vehicle," *Proceedings of the American Control Conference,* 2017.

[36] O. Araar, N. Aouf, I. Vitanov, "Vision Based Autonomous Landing of Multirotor UAV on Moving Platform," *J. Intell. Robot. Syst. Theory Appl.,* 2017.

[37] S. Yang, J. Ying, Y. Lu, Z. Li, "Precise quadrotor autonomous landing with SRUKF vision perception," *Proceedings - IEEE International Conference on Robotics and Automation,* 2015.

[38] R. Bartak, A. Hrasko, D. Obdrzalek, "A controller for autonomous landing of AR.Drone," *26th Chinese Control Decis. Conf. CCDC 2014*.

[39] J. Engel, J. Sturm, D. Cremers, “Scale-aware navigation of a low-cost quadrocopter with a monocular camera,” *Robotics and Autonomous Systems*, 2014.

[40] A. Engineering, “Collaborative Control of UAV / UGV,” در *The 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2014)*, 2014.

[41] T. G. Carreira, “Quadcopter Automatic Landing on a Docking Station,” 2013.

[42] K. E. Wenzel, A. Masselli, A. Zell, “A Quadrocopter Hovering Above a Person Wearing a Modified Cap,” *Proc. Int. Micro Air Veh. Conf. Flight Compet.*, 2010.

[43] S. Saripalli, “Vision-based Autonomous Landing of an Helicopter on a Moving Target,” *AIAA Guid. Navig. Control Conf. Exhib.*, 2009.

[44] E. Bayraktar P. Boyraz, “Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics,” *Turkish J. Electr. Eng. Comput. Sci.*, 2017.

[45] G. Caridakis, R. S. Choras, T. Arif, “Automatic Sign Language Recognition: vision basedfeature extraction and probabilistic recognition schemefrom multiple cues,”121-2018

[46] T. Lindeberg, “Edge Detection and Ridge Detection with Automatic Scale Selection,” 1998.

[47] Machanguillo1, “Edge detection,” *wikipedia*, 2018. Available at: https://en.wikipedia.org/wiki/File:Subpixel_edge_detection.png.

[48] T. P. Patel S. R. Panchal, “Corner Detection Techniques : An Introductory Survey,” *IJEDR*, 2014.

[49] E. Rosten, R. Porter, T. Drummond, “Faster and better: A machine learning approach to corner detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

[50] M. Hassaballah, A. A. Abdelmgeid, H. A. Alshazly, *Image Features Detection , Description and Matching*, 2016.

[51] R. Szeliski, “Computer Vision: Algorithms & Applications,” *Computer (Long. Beach. Calif).*, 2010.

[52] J. Le Moigne ,“First evaluation of automatic image registration methods,” *IGARSS ’98. Sens. Manag. Environ. 1998 IEEE Int. Geosci. Remote Sensing. Symp. Proceedings. (Cat. No.98CH36174)*, 1998.

[53] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, 2004.

[54] H. Bay, T. Tuytelaars, L. Van Gool, “SURF: Speeded up robust features,” در *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.

[55] E. Rosten T. Drummond, “Machine learning for high-speed corner detection,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.

[56] S. D. Babacan, R. Molina, A. K. Katsaggelos, “Bayesian compressive sensing using laplace priors,” *IEEE Trans. Image Process.*, 2010.

[57] T. Lindeberg, “Scale-Space,” *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

[58] R. S. Jain, Ramesh; Kasturi B. G., “Machine Vision: Chapter 5 - Edge Detection,” 1998

[59] “File:Image pyramid.svg - Wikimedia Commons.”. Available at: https://commons.wikimedia.org/wiki/File:Image_pyramid.svg. 2019.

[60] Utkarsh Sinha, “SIFT: Theory and Practice: The scale space - AI Shack.”. Available at: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-scale-space. 2019.

[61] G. Wang, B. Rister, J. R. Cavallaro, “Workload analysis and efficient OpenCL-based implementation of SIFT algorithm on a smartphone,” 2013 *IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, 2013.

[62] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, “Speeded-Up Robust Features (SURF),” *Comput. Vis. Image Underst.*, 2008.

[63] Ş. Işık, “A Comparative Evaluation of Well-known Feature Detectors and Descriptors,” *Int. J. Appl. Math. Electron. Comput.*, 2014.

[64] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[65] P. L. Rosin, “Measuring Corner Properties,” *Comput. Vis. Image Underst.*, 1999.

[66] E. Karami, S. Prasad, M. Shehata, “Image Matching Using SIFT , SURF , BRIEF and ORB : Performance Comparison for Distorted Images,” *Newfoundl. Electr. Comput. Eng. Conf. IEEE, Newfoundl. Labrador Sect.*, 2015.

[67] “AQ-All-Frames farsi.”. Available at: http://autoquad.org/wiki.

[68] P. Ghosh, A. Pandey, U. C. Pati, “Comparison of Different Feature Detection Techniques for Image Mosaicing,” *Accent. Trans. Image Process. Comput. Vis. ISSN*, 2015.

[69] P. Francus, “3 . IMAGE CALIBRATION , FILTERING , AND PROCESSING ALEXANDRA J . NEDERBRAGT (a.nederbragt@ucl.ac.uk) UK Climate System Research Center Currently at Department of Earth Sciences MICHAEL J . SOREGHAN (msoreg@ou.edu),” *Image Process.*,2004.

7- پیوست

1–7 کدهای کالیبراسیون تصاویر ربات:

```
import numpy as np
import cv2
import glob


font = cv2.FONT_HERSHEY_SIMPLEX

aruco_dict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
parameters =  cv2.aruco.DetectorParameters_create()



global _x1
global _y1q
global _width
global _height
_x1=0
_y1=0
_width=0
_height=0

ED_pixel=469    #pixel
ED_cm=23        #cm
pixel2cm_ratio=ED_pixel/ED_cm

'''''
cap = cv2.VideoCapture(0)
resx=1280
resy=720
cap.set(3,resx)
cap.set(4,resy)
'''
#cap = cv2.VideoCapture('/home/atron/Desktop/ardrone_camera_calibration/vid.mp4')




cv2.namedWindow('calib',cv2.WINDOW_NORMAL)
cv2.resizeWindow('calib', 600,320)
#cv2.namedWindow('dis',cv2.WINDOW_NORMAL)
#cv2.resizeWindow('dis', 600,320)

##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv Camera calibration vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv##


print('\nwait for calibration...\n')

# termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((6*7,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)

# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.

images = glob.glob('ardrone_camera_calibration/*.jpg')
#cv2.namedWindow('calib',cv2.WINDOW_NORMAL)
```

```python


for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, (7,6),None)

    # If found, add object points, image points (after refining them)
    if ret == True:
        objpoints.append(objp)

        corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
        imgpoints.append(corners2)


        # Draw and display the corners
        img = cv2.drawChessboardCorners(img, (7,6), corners2,ret)
        cv2.imshow('calib',img)
        cv2.resizeWindow('calib', 600,320)
        cv2.waitKey(2)


#cv2.destroyAllWindows()
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)
print('Calibration Done\n')
cv2.destroyWindow('calib')
print(mtx)
##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Camera Calibration ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##


cv2.namedWindow('img',cv2.WINDOW_NORMAL)
cv2.resizeWindow('img', 600,320)
cv2.namedWindow('undis(left) vs dis(right)',cv2.WINDOW_NORMAL)
cv2.resizeWindow('undis(left) vs dis(right)', 640,180)

##vvvvvvvvvvvvvvvvvvvvvvvvvvv Undistortion vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv##

def undistort(h,w,img):

    newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),1,(w,h))
    # undistort
    dst = cv2.undistort(img, mtx, dist, None, newcameramtx)

    # crop the image
    x,y,w,h = roi
    #dst = dst[y:y+h, x:x+w]
    img=dst[y:y+h, x:x+w]
    img=cv2.resize(img,(w,h))
    return(img)


##^^^^^^^^^^^^^^^^^^^^^^^^^^^ Undistortion ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##


img=cv2.imread('/home/atron/Desktop/ardrone_camera_calibration/315cm.jpg')
img=cv2.imread('/home/atron/Desktop/A5-80cm-33.jpg')
img=cv2.imread('/home/atron/Desktop/A5-395cm.jpg')
img=cv2.imread('/home/atron/Desktop/ardrone_camera_calibration/A5/250.jpg')

h,  w = img.shape[:2]
img_copy=img
img=undistort(h,w,img)

##vvvvvvvvvvvvvvvvvvvvvvvvvvv 3D cube vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv##
```

```
127.
128.
129. def draw(img, corners, imgpts):
130.     imgpts = np.int32(imgpts).reshape(-1,2)
131.     #print(imgpts[:4][0])       #array of the fist 4 points on chessboard
132.     # draw ground floor in green
133.
134.
135.     #img = cv2.drawContours(img, [imgpts[:4]],-1,(0,255,0),-3)   # Draw green rectangle on chessboard
136.
137.
138.     _width=imgpts[:4][1][0]-imgpts[:4][2][0]
139.     _height=imgpts[:4][1][1]-imgpts[:4][2][1]
140.
141.
142.     ############################ Distance calulation ############################
143.     chessBoard='A5'
144.     if chessBoard=='A4':
145.         chessboard_h_mm=230         #230mm
146.         chessboard_h_pixels=331         #331    #pixel in relation with distance to camera
147.         chessboard_Distance_to_camera=500   #500mm
148.         F_divide_by_sensor_height=chessboard_Distance_to_camera/chessboard_h_mm
149.         #print(F_divide_by_sensor_height)
150.
151.         error=(-2.161*_height)+373              #ba azmun o khata moadele error bedast umad
152.         dist=(F_divide_by_sensor_height*chessboard_h_mm*chessboard_h_pixels)/_height
153.         print('dist without error:',dist)
154.         dist=dist+(error)
155.         print('_width=',_width,'\t_height=',_height,'\tdist=',dist)
156.
157.
158.     elif chessBoard=='A5':
159.         chessboard_h_mm=330         #230mm
160.         chessboard_h_pixels=331         #331    #pixel in relation with distance to camera
161.         chessboard_Distance_to_camera=800   #500mm
162.         F_divide_by_sensor_height=chessboard_Distance_to_camera/chessboard_h_mm
163.         #print(F_divide_by_sensor_height)
164.
165.         error=(-2.161*_height)+373              #ba azmun o khata moadele error bedast umad
166.
167.         dist=(F_divide_by_sensor_height*chessboard_h_mm*chessboard_h_pixels)/_height
168.         print('dist==',dist)
169.         error2=(0.2027*dist)-192.3
170.         dist2=dist-(error2)
171.         dist3=dist-(0.002353732*dist)
172.         #dist=(dist*1.4)+dist
173.         print('_width=',_width,'\t_height=',_height,'\tdist=',dist,'\tdist2=',dist2,'\tdist3=',dist3)
174.         dist_text=str(str(int(dist/10))+'cm')
175.         cv2.putText(img,dist_text,(50,70), font, 1.5,(0,255,255),3,cv2.LINE_AA)
176.     ############################ Distance calulation ############################
177.
178.
179.     #print points position
180.     for i in range(0,4):
181.
182.         _x1=imgpts[:4][i][0]
183.         _y1=imgpts[:4][i][1]
184.         #pos = "X= %d , Y= %d" % (_x1, _y1)
185.         #cv2.putText(img,pos,(int(_x1)+20,int(_y1)+20), font, 0.4,(200,250,250),2,cv2.LINE_AA) # put text
      on 4 corners
186.     '''''
187.     # draw pillars in blue color
188.     for i,j in zip(range(4),range(4,8)):
189.         img = cv2.line(img, tuple(imgpts[i]), tuple(imgpts[j]),(255),3)
190.
191.    # draw top layer in red color
```

```
        img = cv2.drawContours(img, [imgpts[4:]],-1,(0,0,255),3)
    '''
    return img


cube_n=6
axis = np.float32([[-1,0,0], [-1,cube_n,0], [cube_n+1,cube_n,0], [cube_n+1,0,0],
                   [0,0,-cube_n],[0,cube_n,-cube_n],[cube_n,cube_n,-cube_n],[cube_n,0,-cube_n] ])

##^^^^^^^^^^^^^^^^^^^^^^^^^^^^ 3D cube ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##



criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
objp = np.zeros((6*7,3), np.float32)                    #np.zeros((6*7,3)
objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)              #np.mgrid[0:7,0:6]


#for fname in glob.glob('camera_calibration/2018-07-31-075110_1.jpg'):


''''
while(True):
    _, img = cap.read()
    frame=img
    img_copy=img
    h,  w = img.shape[:2]
    img=undistort(h,w,img)
    #print('h,w',h,w)
    #print('shape',img.shape[:2])
    #img = cv2.imread(fname)

    new_size=0.2
    _h=int(h*new_size)
    _w=int(w*new_size)
    _img=cv2.resize(img,(_w,_h))
    _img_copy=cv2.resize(img_copy,(_w,_h))
    vertical_concat = np.concatenate((_img,_img_copy), axis=1)
    cv2.imshow('dis vs undis',vertical_concat)

    #cv2.imshow('dis',img0)

'''


gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, corners = cv2.findChessboardCorners(gray, (7,6),None)      #findChessboardCorners(gray, (7,6),None)


    #corners, ids, rejectedImgPoints = cv2.aruco.detectMarkers(gray, aruco_dict, parameters=parameters)
if ret == True:
    corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
    # Find the rotation and translation vectors.
    _,rvecs, tvecs, inliers = cv2.solvePnPRansac(objp, corners2, mtx, dist)
    # project 3D points to image plane
    imgpts, jac = cv2.projectPoints(axis, rvecs, tvecs, mtx, dist)
    img = draw(img,corners2,imgpts)



    new_size=0.8
    _h=int(h*new_size)
```

```
    _w=int(w*new_size)
    _img=cv2.resize(img,(_w,_h))
    _img_copy=cv2.resize(img_copy,(_w,_h))
    vertical_concat = np.concatenate((_img,_img_copy), axis=1)
    cv2.imshow('undis(left) vs dis(right)',vertical_concat)



    #cv2.namedWindow('img',cv2.WINDOW_NORMAL)
    #cv2.resizeWindow('img', 600,320)
    cv2.imshow('img',img)
    cv2.imwrite('calib-img.jpg',vertical_concat)

cv2.imshow('img',img)
if cv2.waitKey(0) & 0xFF == ord('q'):
    cv2.destroyAllWindows()

```

2–7 کدهای کالیبراسیون ویدئویی:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import time
import os
import glob
MIN_MATCH_COUNT = 33
Distance=75


img0 = cv2.imread('the_book_thief.jpg',0) # trainImage
img0=cv2.resize(img0,(319,499))


cv2.namedWindow('trainImage',cv2.WINDOW_NORMAL)
cv2.resizeWindow('trainImage', 240,450)
# Initiate SIFT detector
#sift = cv2.SIFT()
sift = cv2.xfeatures2d.SIFT_create()
# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(img0,None)

img1 = cv2.drawKeypoints(img0, kp1, None)
cv2.imshow("trainImage", img1)









def get_vid(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'Ar_Drone'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','input =',video_file)
    return(video_file)


def set_output_vid_address(video_name):
    video_name=str(video_name)
```

```
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'images'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','saving to >>',video_file)
    return(video_file)


def img_address(img_name):
    video_name=str(img_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'images'))
    pathX=os.path.abspath(os.path.join(pathX,'sift'))
    img_file=pathX+"/"+img_name+'.png'
    print('********>>>> ','saving images to >>',img_file)
    return(img_file)


vid='2.mp4'
video_name=get_vid(vid)
cap = cv2.VideoCapture(video_name)
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
print('w:','h:',frame_width,frame_height)
output_name=set_output_vid_address('calibrated'+'.mp4')

fourcc = cv2.VideoWriter_fourcc(*'mp4v')
#out=cv2.VideoWriter(output_name,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (frame_width,frame_height))

out=cv2.VideoWriter(output_name,fourcc, 30, (frame_width,frame_height))
if (cap.isOpened()== False):
    print("Error opening video stream or file")


print('w:','h:',frame_width*2,frame_height)
output_name2=set_output_vid_address('calibrated side by side'+'.mp4')
#out=cv2.VideoWriter(output_name,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (frame_width,frame_height))

out2=cv2.VideoWriter(output_name2,fourcc, 30, (frame_width*2,frame_height))

'''''
resx=480
resy=360
cap.set(3,resx)
cap.set(4,resy)
'''

#text_file= open("sift_video_precision_complex.txt","w+")


# BFMatcher with default params
'''''
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)
'''
cv2.namedWindow('calibrated',cv2.WINDOW_NORMAL)
cv2.resizeWindow('calibrated', 600,480)
cv2.namedWindow('undis(left) vs dis(right)',cv2.WINDOW_NORMAL)
cv2.resizeWindow('undis(left) vs dis(right)', 640,180)
```

```
382.
383. def nothing(x):
384.     pass
385.
386. #cv2.createTrackbar('Min match count','SIFT Parametric',MIN_MATCH_COUNT,50,nothing)
387. #cv2.createTrackbar('Distance','SIFT Parametric',Distance,100,nothing)
388.
389. fps=0
390. counter_1=0
391. counter_2=0
392. start_time=time.time()      # start time of the loop
393.
394.
395.
396.
397.
398. ##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv Camera calibration vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv##
399. cv2.namedWindow('calib',cv2.WINDOW_NORMAL)
400. cv2.resizeWindow('calib', 600,320)
401. print('\nwait for calibration...\n')
402.
403. # termination criteria
404. criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
405.
406. # prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
407. objp = np.zeros((6*7,3), np.float32)
408. objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)
409.
410. # Arrays to store object points and image points from all the images.
411. objpoints = [] # 3d point in real world space
412. imgpoints = [] # 2d points in image plane.
413.
414. images = glob.glob('/home/atron/Desktop/ardrone_camera_calibration/*.jpg')
415. #cv2.namedWindow('calib',cv2.WINDOW_NORMAL)
416.
417.
418. for fname in images:
419.     img = cv2.imread(fname)
420.     gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
421.
422.     # Find the chess board corners
423.     ret, corners = cv2.findChessboardCorners(gray, (7,6),None)
424.
425.     # If found, add object points, image points (after refining them)
426.     if ret == True:
427.         objpoints.append(objp)
428.
429.         corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
430.         imgpoints.append(corners2)
431.
432.
433.         # Draw and display the corners
434.         img = cv2.drawChessboardCorners(img, (7,6), corners2,ret)
435.         cv2.imshow('calib',img)
436.         cv2.resizeWindow('calib', 600,320)
437.         cv2.waitKey(2)
438.
439.
440. #cv2.destroyAllWindows()
441. ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)
442. print('Calibration Done\n')
443. cv2.destroyWindow('calib')
444. print(mtx)
445. ##^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Camera Calibration ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
446.
447.
```

```
448.
449.
450. ##vvvvvvvvvvvvvvvvvvvvvvvvvvvv Undistortion vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv##
451.
452. def undistort(h,w,img):
453.
454.     newcameramtx, roi=cv2.getOptimalNewCameraMatrix(mtx,dist,(w,h),0,(w,h))
455.     # undistort
456.     dst = cv2.undistort(img, mtx, dist, None, newcameramtx)
457.
458.     # crop the image
459.     x,y,w,h = roi
460.     #dst = dst[y:y+h, x:x+w]
461.     img=dst[y:y+h, x:x+w]
462.     img=cv2.resize(img,(w,h))
463.     return(img)
464.
465.
466. ##^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Undistortion ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^##
467.
468.
469.
470.
471.
472. while(cap.isOpened()):
473.
474.     #MIN_MATCH_COUNT = cv2.getTrackbarPos('Min match count','SIFT Parametric')
475.     ret, frame0 = cap.read()
476.     if (ret==False):
477.         break
478.     counter_1+=1
479.
480.     h,  w = frame0.shape[:2]
481.     #print('h2','w2',h,w)
482.     frame1=frame0
483.     frame1=undistort(frame_height,frame_width,frame1)
484.     frame1=cv2.resize(frame1,(frame_width,frame_height))
485.
486.     out.write(frame1)
487.
488.     cv2.imshow("calibrated",frame1)
489.     new_size=1
490.     _h=int(h*new_size)
491.     _w=int(w*new_size)
492.     frame0=cv2.resize(frame0,(_w,_h))
493.     frame1=cv2.resize(frame1,(_w,_h))
494.     vertical_concat = np.concatenate((frame0,frame1), axis=1)
495.     cv2.imshow('undis(left) vs dis(right)',vertical_concat)
496.     out2.write(vertical_concat)
497.
498.     if cv2.waitKey(1) & 0xFF == ord('q'):
499.         #text_file.close()
500.         break
501.
502. cap.release()
503. out.release()
504. out2.release()
505. print('\n\nDone\n')
506. cv2.destroyAllWindows()
```

3–7 کدهای پردازش ویدئویی و استخراج تصاویر خروجی SIFT :

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import time
import os
MIN_MATCH_COUNT = 33
Distance=75


img0 = cv2.imread('the_book_thief.jpg',0) # trainImage
img0=cv2.resize(img0,(319,499))


cv2.namedWindow('trainImage',cv2.WINDOW_NORMAL)
cv2.resizeWindow('trainImage', 240,450)
# Initiate SIFT detector
#sift = cv2.SIFT()
sift = cv2.xfeatures2d.SIFT_create()
# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(img0,None)

img1 = cv2.drawKeypoints(img0, kp1, None)
cv2.imshow("trainImage", img1)









def get_vid(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'Ar_Drone'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','input =',video_file)
    return(video_file)


def set_output_vid_address(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'video_output'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','saving to >>',video_file)
    return(video_file)


vid='2.mp4'
video_name=get_vid(vid)
cap = cv2.VideoCapture(video_name)
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
output_name=set_output_vid_address('SIFT_'+'complex.mp4')
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
```

```
#out=cv2.VideoWriter(output_name,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (frame_width,frame_height))

out=cv2.VideoWriter(output_name,fourcc, 30, (frame_width,frame_height))
if (cap.isOpened()== False):
    print("Error opening video stream or file")


'''''
resx=480
resy=360
cap.set(3,resx)
cap.set(4,resy)
'''
text_file= open("sift_video_precision_complex.txt","w+")


# BFMatcher with default params
'''''
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)
'''
cv2.namedWindow('SIFT Parametric',cv2.WINDOW_NORMAL)
cv2.resizeWindow('SIFT Parametric', 600,480)

def nothing(x):
    pass

cv2.createTrackbar('Min match count','SIFT Parametric',MIN_MATCH_COUNT,50,nothing)
cv2.createTrackbar('Distance','SIFT Parametric',Distance,100,nothing)

fps=0
counter_1=0
counter_2=0
start_time=time.time()      # start time of the loop

while(cap.isOpened()):

    MIN_MATCH_COUNT = cv2.getTrackbarPos('Min match count','SIFT Parametric')
    ret, frame0 = cap.read()
    if (ret==False):
        break
    counter_1+=1

    #frame=cv2.resize(frame,(480,320))
    frame = cv2.cvtColor(frame0, cv2.COLOR_BGR2GRAY)
    kp2, des2 = sift.detectAndCompute(frame,None)
    frame = cv2.drawKeypoints(frame, kp2, None)




    #bf = cv2.BFMatcher()
    #matches = bf.knnMatch(des1,des2, k=2)

    '''''
    #bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    #matches = bf.match(des1,des2)
    #matches = sorted(matches, key = lambda x:x.distance)


    # Initialize lists
    # Apply ratio test

    good = []
    for m,n in matches:
```

```
        if m.distance < 0.75:
            good.append([m])
    '''
    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
    search_params = dict(checks = 50)

    flann = cv2.FlannBasedMatcher(index_params, search_params)
    if not (np.any(des2==None)):
        matches = flann.knnMatch(des1,des2,k=2)
    else:
        print("Cannot match")



    good_distance=(cv2.getTrackbarPos('Distance','SIFT Parametric')/100)

    good = []
    for m,n in matches:
        if m.distance < good_distance*n.distance:
            good.append(m)




    if len(good)>MIN_MATCH_COUNT:
        src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
        dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)

        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
        matchesMask = mask.ravel().tolist()
        if not (np.any(M==None)):
            h,w,_ = img1.shape
            pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
            dst = cv2.perspectiveTransform(pts,M)
            frame0 = cv2.polylines(frame0,[np.int32(dst)],True,255,5, cv2.LINE_AA)
            counter_2+=1
        else:
            print("not enough 'M'")

    else:
        print ("Not enough matches are found - %d/%d" % (len(good),MIN_MATCH_COUNT))
        matchesMask = None

    draw_params = dict(matchColor = (0,255,0), # draw matches in green color
        singlePointColor = None,
        matchesMask = matchesMask, # draw only inliers
        flags = 2)
    if not (np.any(des2==None)):
        img3 = cv2.drawMatches(img1,kp1,frame,kp2,good,None,**draw_params)
    else:
        print("Cannot match")

    # cv2.drawMatchesKnn expects list of lists as matches.
    #img4 = cv2.drawMatchesKnn(img1,kp1,frame,kp2,good,None,flags=2)




    #cv2.imshow("frame",frame)


    diff = time.time() - start_time
    if (diff >= 1):
        start_time = time.time()
```

```
        cv2.setWindowTitle("SIFT Parametric","SIFT   "+str(int(fps))+"  fps   "+str(len(good)))          # taghire title window be meghdare fps
        #text_file.write("FPS= %f\r\n" % (fps))
        fps = 0
    out.write(frame0)
    cv2.imshow("SIFT Parametric",frame0)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        #text_file.close()
        break


video_precision=(counter_2/counter_1)*100
print('counter_2/counter_1=video_precision=',video_precision)
print('counter 1=',counter_1)
print('counter 2=',counter_2)
text_file.write("video_precision= %f\t" % (video_precision)+'\n')
text_file.write("counter_2= %f\t" % (counter_2)+'\n')
text_file.write("total= %f\t" % (counter_1)+'\n')
text_file.close()
cap.release()
out.release()
cv2.destroyAllWindows()
```

4–7 کدهای پردازش ویدئویی و استخراج تصاویر خروجی SURF :

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import time
MIN_MATCH_COUNT = 33
Distance=75
import os



img0 = cv2.imread('the_book_thief.jpg',0) # trainImage
img0=cv2.resize(img0,(319,499))


cv2.namedWindow('trainImage',cv2.WINDOW_NORMAL)
cv2.resizeWindow('trainImage', 240,450)
# Initiate surf detector
#surf = cv2.surf()
surf = cv2.xfeatures2d.SURF_create()
# find the keypoints and descriptors with SURF
kp1, des1 = surf.detectAndCompute(img0,None)

img1 = cv2.drawKeypoints(img0, kp1, None)
cv2.imshow("trainImage", img1)



def get_vid(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
```

```
    pathX=os.path.abspath(os.path.join(parentDir,'Ar_Drone'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','input =',video_file)
    return(video_file)


def set_output_vid_address(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'video_output'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','saving to >>',video_file)
    return(video_file)


vid='2.mp4'
video_name=get_vid(vid)
cap = cv2.VideoCapture(video_name)
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
output_name=set_output_vid_address('SURF_'+'complex.mp4')
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
#out=cv2.VideoWriter(output_name,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (frame_width,frame_height))

out=cv2.VideoWriter(output_name,fourcc, 30, (frame_width,frame_height))
if (cap.isOpened()== False):
  print("Error opening video stream or file")



text_file= open("surf_video_precision_complex.txt","w+")


# BFMatcher with default params
'''''
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)
'''
cv2.namedWindow('SURF Parametric',cv2.WINDOW_NORMAL)
cv2.resizeWindow('SURF Parametric', 600,480)

def nothing(x):
    pass

cv2.createTrackbar('Min match count','SURF Parametric',MIN_MATCH_COUNT,150,nothing)
cv2.createTrackbar('Distance','SURF Parametric',Distance,100,nothing)
fps=0
counter_1=0
counter_2=0

start_time = time.time()                    # start time of the loop

while(1):

    MIN_MATCH_COUNT = cv2.getTrackbarPos('Min match count','SURF Parametric')
    ret, frame0 = cap.read()
    if (ret==False):
        break
    counter_1+=1
    start_time2 = time.time()                   # start time of the loop
    frame = cv2.cvtColor(frame0, cv2.COLOR_BGR2GRAY)
    kp2, des2 = surf.detectAndCompute(frame,None)
    frame = cv2.drawKeypoints(frame, kp2, None)
```

```
    #bf = cv2.BFMatcher()
    #matches = bf.knnMatch(des1,des2, k=2)

    '''''
    #bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    #matches = bf.match(des1,des2)
    #matches = sorted(matches, key = lambda x:x.distance)


    # Initialize lists
    # Apply ratio test

    good = []
    for m,n in matches:
        if m.distance < 0.75:
            good.append([m])
    '''
    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
    search_params = dict(checks = 50)

    flann = cv2.FlannBasedMatcher(index_params, search_params)
    if not (np.any(des2==None)):
        matches = flann.knnMatch(des1,des2,k=2)
    else:
        #matches=[]
        print("Cannot match")



    good_distance=(cv2.getTrackbarPos('Distance','SURF Parametric')/100)

    good = []
    for m,n in matches:
        if m.distance < good_distance*n.distance:
            good.append(m)




    if len(good)>MIN_MATCH_COUNT:
        src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
        dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)

        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
        matchesMask = mask.ravel().tolist()
        if not (np.any(M==None)):
            h,w,_ = img1.shape
            pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
            dst = cv2.perspectiveTransform(pts,M)
            frame0 = cv2.polylines(frame0,[np.int32(dst)],True,255,5, cv2.LINE_AA)
            counter_2+=1
        else:
            print("not enough 'M'")

    else:
        print ("Not enough matches are found - %d/%d" % (len(good),MIN_MATCH_COUNT))
        matchesMask = None

    draw_params = dict(matchColor = (0,255,0), # draw matches in green color
        singlePointColor = None,
        matchesMask = matchesMask, # draw only inliers
        flags = 2)
    if not (np.any(des2==None)):
        img3 = cv2.drawMatches(img1,kp1,frame,kp2,good,None,**draw_params)
```

```
    else:
        #matches=[]
        #img3=img3 = cv2.drawMatches(img1,kp1,frame,kp2,matches,None,**draw_params)
        print("Cannot match")

    # cv2.drawMatchesKnn expects list of lists as matches.
    #img4 = cv2.drawMatchesKnn(img1,kp1,frame,kp2,good,None,flags=2)




    #cv2.imshow("frame",frame)



    end_time=time.time()-start_time2
    print('end_time:',(time.time()-start_time))
    if ((time.time()-start_time)>1):
        fps=(1.0 / (end_time))
        #text_file.write("FPS= %f\r\n" % (fps))
        cv2.setWindowTitle("SURF Parametric","SURF    "+" fps="+str(int(fps))+' good='+str(len(good)))
      #taghire title window be meghdare fps
        start_time=time.time()
    '''''
    diff = time.time() - start_time
    if (diff >= 1):
        start_time = time.time()
        cv2.setWindowTitle("SURF Parametric","SURF    "+" fps="+str(int(fps))+' good='+str(len(good)))
      #taghire title window be meghdare fps
        text_file.write("FPS= %f\r\n" % (fps))
        fps = 0
    '''

    out.write(frame0)
    cv2.imshow("SURF Parametric",frame0)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        #text_file.close()
        break



video_precision=(counter_2/counter_1)*100
print('counter_2/counter_1=video_precision=',video_precision)
print('counter 1=',counter_1)
print('counter 2=',counter_2)
text_file.write("video_precision= %f\t" % (video_precision)+'\n')
text_file.write("counter_2= %f\t" % (counter_2)+'\n')
text_file.write("total= %f\t" % (counter_1)+'\n')
text_file.close()
cap.release()
out.release()
cv2.destroyAllWindows()
```

5–7 کدهای پردازش ویدئویی و استخراج تصاویر خروجی ORB :

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import time
import os
```

```
MIN_MATCH_COUNT = 10
MIN_MATCH=50

FLANN_INDEX_LSH=6

trainImg = cv2.imread("the_book_thief.jpg", cv2.IMREAD_GRAYSCALE)
#trainImg=cv2.resize(trainImg,(360,720))
orb = cv2.ORB_create(nfeatures=1500)

kp1, des1 = orb.detectAndCompute(trainImg, None)
trainImg = cv2.drawKeypoints(trainImg, kp1, None)
#cv2.namedWindow('Image',cv2.WINDOW_NORMAL)
#cv2.resizeWindow('Image', 360,720)
#cv2.imshow("Image", trainImg)




def get_vid(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'Ar_Drone'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','input =',video_file)
    return(video_file)


def set_output_vid_address(video_name):
    video_name=str(video_name)
    Dir=os.path.dirname(os.path.abspath(__file__))
    #go to parent directory
    parentDir=os.path.abspath(os.path.join(Dir,".."))
    #go to sift folder or surf or ....
    pathX=os.path.abspath(os.path.join(parentDir,'video_output'))
    video_file=pathX+"/"+video_name
    print('********>>>> ','saving to >>',video_file)
    return(video_file)


vid='2.mp4'
video_name=get_vid(vid)
cap = cv2.VideoCapture(video_name)
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
output_name=set_output_vid_address('ORB_'+'complex.mp4')
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
#out=cv2.VideoWriter(output_name,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (frame_width,frame_height))

out=cv2.VideoWriter(output_name,fourcc, 30, (frame_width,frame_height))
if (cap.isOpened()== False):
  print("Error opening video stream or file")


def nothing(x):
    pass




        cv2.namedWindow('ORB',cv2.WINDOW_NORMAL)
        cv2.resizeWindow('ORB', 480,360)
```

```
        #cv2.createTrackbar('Min match count','ORB',20,1400,nothing)
        cv2.createTrackbar('Min match','ORB',1000,1400,nothing) #350
        text_file= open("ORB2_Video_precision_complex.txt","w+")
        fps=0
        counter_1=0
        counter_2=0
        start_time=time.time()
        while(1):
            ret, frame0 = cap.read()
            if (ret==False):
                break
            counter_1+=1
            #if (counter_1>1280):
            #   break
            print('cnt1=',counter_1)

            frame = cv2.cvtColor(frame0, cv2.COLOR_BGR2GRAY)
            kp2, des2 = orb.detectAndCompute(frame,None)
            #MIN_MATCH_COUNT = cv2.getTrackbarPos('Min match count','ORB')
            MIN_MATCH = cv2.getTrackbarPos('Min match','ORB')

            bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
            matches = bf.match(des1,des2)

            dmatches = sorted(matches, key = lambda x:x.distance)
            dmatches=dmatches[:MIN_MATCH_COUNT]
            img3=frame0

            src_pts  = np.float32([kp1[m.queryIdx].pt for m in dmatches]).reshape(-1,1,2)
            dst_pts  = np.float32([kp2[m.trainIdx].pt for m in dmatches]).reshape(-1,1,2)
            M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
            h,w = trainImg.shape[:2]
            pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)









            if (len(matches)>MIN_MATCH):
                dst = cv2.perspectiveTransform(pts,M)
                img3 = cv2.polylines(frame0, [np.int32(dst)], True, (255,0,0), 3, cv2.LINE_AA)  #draw bl
    ue rectangle around object
                counter_2+=1
            #img3 = cv2.drawMatches(trainImg,kp1,frame0,kp2,dmatches,None, flags=2)         #show matche
    s by lines









            #top_left=min(top_left)

            #print((top_left), top_left[0],top_left[1])
            #print((bottom_right), bottom_right[0],bottom_right[1])
            #cv2.circle(img3,(round(top_left[0]),round(top_left[1])), 5, (0,0,255), -1)
            #cv2.circle(img3,(round(bottom_right[0]),round(bottom_right[1])), 5, (0,0,255), -1)
            #cv2.rectangle(img3,top_left, bottom_right,(0,0,255), 2)




            #img3 = cv2.drawMatchesKnn(trainImg,kp1,frame,kp2,matches,None,cv2.DRAW_MATCHES_FLAGS_DRAW_R
    ICH_KEYPOINTS)
            #plt.imshow(img3,),plt.show()

```

```
1066.
1067.            diff = time.time() - start_time
1068.            if (diff >= 1):
1069.                start_time = time.time()
1070.                cv2.setWindowTitle("ORB","ORB    "+" fps="+str(int(fps))+' good='+str(len(matches)))
         #taghire title window be meghdare fps
1071.                #text_file.write("FPS= %f\r\n" % (fps))
1072.                fps = 0
1073.
1074.
1075.
1076.
1077.
1078.
1079.
1080.            out.write(img3)
1081.            cv2.imshow("ORB",img3)
1082.            #cv2.waitKey(1)
1083.            if cv2.waitKey(1) & 0xFF == ord('q'):
1084.                #text_file.close()
1085.                break
1086.        #cv2.waitKey(0)
1087.        #cv2.destroyAllWindows()
1088.
1089.
1090.        video_precision=(counter_2/counter_1)*100
1091.        print('counter_2/counter_1=video_precision=',video_precision)
1092.        print('counter 1=',counter_1)
1093.        print('counter 2=',counter_2)
1094.        text_file.write("video_precision= %f\t" % (video_precision)+'\n')
1095.        text_file.write("counter_2= %f\t" % (counter_2)+'\n')
1096.        text_file.write("total= %f\t" % (counter_1)+'\n')
1097.        text_file.close()
1098.        cap.release()
1099.        out.release()
1100.        cv2.destroyAllWindows()
1101.
```

6–7 کدهای پردازش ویدئویی و استخراج تصاویر خروجی BRISK :

```
1102.        import cv2
1103.        import numpy as np
1104.        from matplotlib import pyplot as plt
1105.        import time
1106.        import os
1107.
1108.
1109.        MIN_MATCH_COUNT = 10
1110.        MIN_MATCH=50
1111.
1112.        FLANN_INDEX_LSH=6
1113.
1114.        trainImg = cv2.imread("the_book_thief.jpg", cv2.IMREAD_GRAYSCALE)
1115.        #trainImg=cv2.resize(trainImg,(360,720))
1116.        #orb = cv2.ORB_create(nfeatures=1500)
1117.        brisk = cv2.BRISK_create()
1118.        kp1, des1 = brisk.detectAndCompute(trainImg, None)
1119.        trainImg = cv2.drawKeypoints(trainImg, kp1, None)
1120.        #cv2.namedWindow('Image',cv2.WINDOW_NORMAL)
```

```
        #cv2.resizeWindow('Image', 360,720)
        #cv2.imshow("Image", trainImg)


        def get_vid(video_name):
            video_name=str(video_name)
            Dir=os.path.dirname(os.path.abspath(__file__))
            #go to parent directory
            parentDir=os.path.abspath(os.path.join(Dir,".."))
            #go to sift folder or surf or ....
            pathX=os.path.abspath(os.path.join(parentDir,'Ar_Drone'))
            video_file=pathX+"/"+video_name
            print('********>>>> ','input =',video_file)
            return(video_file)


        def set_output_vid_address(video_name):
            video_name=str(video_name)
            Dir=os.path.dirname(os.path.abspath(__file__))
            #go to parent directory
            parentDir=os.path.abspath(os.path.join(Dir,".."))
            #go to sift folder or surf or ....
            pathX=os.path.abspath(os.path.join(parentDir,'video_output'))
            video_file=pathX+"/"+video_name
            print('********>>>> ','saving to >>',video_file)
            return(video_file)


        vid='2.mp4'
        video_name=get_vid(vid)
        cap = cv2.VideoCapture(video_name)
        frame_width = int(cap.get(3))
        frame_height = int(cap.get(4))
        output_name=set_output_vid_address('BRISK_'+'complex.mp4')
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        #out=cv2.VideoWriter(output_name,cv2.VideoWriter_fourcc('M','J','P','G'), 30, (frame_width,frame
    _height))
        out=cv2.VideoWriter(output_name,fourcc, 30, (frame_width,frame_height))
        if (cap.isOpened()== False):
          print("Error opening video stream or file")


        def nothing(x):
            pass




        cv2.namedWindow('BRISK',cv2.WINDOW_NORMAL)
        cv2.resizeWindow('BRISK', 480,360)
        #cv2.createTrackbar('Min match count','BRISK',120,1400,nothing)
        cv2.createTrackbar('Min match','BRISK',340,1400,nothing)
        text_file= open("BRISK_video_precision_complex.txt","w+")
        fps=0
        counter_1=0
        counter_2=0
        start_time=time.time()
        while(cap.isOpened()):
            ret, frame0 = cap.read()
            fps=fps+1
            if (ret==False):
                break
            counter_1+=1
            frame = cv2.cvtColor(frame0, cv2.COLOR_BGR2GRAY)
```

```
            kp2, des2 = brisk.detectAndCompute(frame,None)
            #MIN_MATCH_COUNT = cv2.getTrackbarPos('Min match count','BRISK')
            MIN_MATCH = cv2.getTrackbarPos('Min match','BRISK')

            bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
            matches = bf.match(des1,des2)

            dmatches = sorted(matches, key = lambda x:x.distance)
            dmatches=dmatches[:MIN_MATCH_COUNT]
            img3=frame0

            src_pts  = np.float32([kp1[m.queryIdx].pt for m in dmatches]).reshape(-1,1,2)
            dst_pts  = np.float32([kp2[m.trainIdx].pt for m in dmatches]).reshape(-1,1,2)
            M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0)
            h,w = trainImg.shape[:2]
            pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)

            if (len(matches)>MIN_MATCH):
                dst = cv2.perspectiveTransform(pts,M)
                img3 = cv2.polylines(frame0, [np.int32(dst)], True, (255,0,0), 3, cv2.LINE_AA)  #draw bl
    ue rectangle around object
                counter_2+=1
            #img3 = cv2.drawMatches(trainImg,kp1,frame0,kp2,dmatches,None, flags=2)         #show matche
    s by lines



            #top_left=min(top_left)

            #print((top_left), top_left[0],top_left[1])
            #print((bottom_right), bottom_right[0],bottom_right[1])
            #cv2.circle(img3,(round(top_left[0]),round(top_left[1])), 5, (0,0,255), -1)
            #cv2.circle(img3,(round(bottom_right[0]),round(bottom_right[1])), 5, (0,0,255), -1)
            #cv2.rectangle(img3,top_left, bottom_right,(0,0,255), 2)




            #img3 = cv2.drawMatchesKnn(trainImg,kp1,frame,kp2,matches,None,cv2.DRAW_MATCHES_FLAGS_DRAW_R
    ICH_KEYPOINTS)
            #plt.imshow(img3,),plt.show()

            diff = time.time() - start_time
            if (diff >= 1):
                start_time = time.time()
                cv2.setWindowTitle("BRISK","BRISK    "+" fps="+str(int(fps))+' good='+str(len(matches)))
        #taghire title window be meghdare fps
                #text_file.write("FPS= %f\r\n" % (fps))
                fps = 0


            out.write(img3)
            cv2.imshow("BRISK",img3)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        video_precision=(counter_2/counter_1)*100
        print('counter_2/counter_1=video_precision=',video_precision)
        print('counter 1=',counter_1)
        print('counter 2=',counter_2)
        text_file.write("video_precision= %f\t" % (video_precision)+'\n')
        text_file.write("counter_2= %f\t" % (counter_2)+'\n')
        text_file.write("total= %f\t" % (counter_1)+'\n')
        text_file.close()
        cap.release()
        out.release()
```

```
1248.         cv2.destroyAllWindows()
1249.
```

7–7 کدهای آنالیز عکس و ذخیره داده‌ها در فایل اکسل SIFT :

```
1250.         import numpy as np
1251.         import cv2
1252.         from matplotlib import pyplot as plt
1253.         import glob
1254.         import time
1255.
1256.         import openpyxl
1257.         from openpyxl import load_workbook
1258.         import os
1259.
1260.
1261.         path="skew/"
1262.         txt_name="sift_"+path.rstrip("/")+".txt"
1263.         txt_name_single="sift_single_"+path.rstrip("/")+".txt"
1264.
1265.         i=0
1266.         sum_time=0
1267.         sum_kp1=0
1268.         sum_kp2=0
1269.         sum_good=0
1270.         sum_rejected_points=0
1271.
1272.         win_size_w=320
1273.         win_size_h=500
1274.
1275.
1276.
1277.         number_of_images=len(glob.glob1(path,"*.jpg"))
1278.         name_of_images=glob.glob1(path,"*.jpg")
1279.
1280.
1281.         print("********************\n\n\n")
1282.
1283.         if (number_of_images==0):
1284.             print("********************\n")
1285.             print("No image present in *%s*"%path,">>> Abort\n")
1286.             print("********************\n\n\n")
1287.             quit()
1288.         else:
1289.             print("number of images=\t",number_of_images)
1290.             print("names=\t",name_of_images)
1291.
1292.         #img2=cv2.imread(name_of_images[1])
1293.         #cv2.namedWindow('imggg',cv2.WINDOW_NORMAL)
1294.         #cv2.resizeWindow('img', 600,320)
1295.         #cv2.imshow("imggg",img2)
1296.         #cv2.imshow("img0",cv2.imread(name_of_images[0]))
1297.
1298.         img1 = cv2.imread('arduino.jpg',0)              # queryImage #second image
1299.         img1=cv2.resize(img1,(320,499))
1300.         #cv2.namedWindow('img1',cv2.WINDOW_NORMAL)
1301.         #cv2.resizeWindow('img1', win_size_w,win_size_h)
1302.         #cv2.imshow("img1",img1)
1303.
1304.         cv2.namedWindow('img2',cv2.WINDOW_NORMAL)
1305.         cv2.resizeWindow('img2', win_size_w,win_size_h)
```

```
1306.         cv2.namedWindow('img3',cv2.WINDOW_NORMAL)
1307.         cv2.resizeWindow('img3', win_size_w*2,win_size_h)
1308.         cv2.namedWindow('kp1-kp2',cv2.WINDOW_NORMAL)
1309.         cv2.resizeWindow('kp1-kp2', win_size_w,win_size_h)
1310.
1311.         text_file= open(path+txt_name,"w+")
1312.         text_file_single=open(path+txt_name_single,"w+")
1313.
1314.         ###################################################   SIFT   ##################################
    ##################
1315.         print("Press 'n' to go to the next image, 'q' to exit...\n\n")
1316.         while(i<number_of_images):
1317.             start_time=time.time()
1318.             image_name=path+name_of_images[i]
1319.
1320.             print(image_name)
1321.             img2=cv2.imread(image_name,0)
1322.             #img2=cv2.resize(img2,(320,499))
1323.
1324.
1325.             cv2.imshow("img2",img2)
1326.             cv2.waitKey(10)
1327.
1328.
1329.             # Initiate SIFT detector
1330.             #sift = cv2.SIFT()
1331.             sift = cv2.xfeatures2d.SIFT_create()
1332.             # find the keypoints and descriptors with SIFT
1333.             kp1, des1 = sift.detectAndCompute(img1,None)
1334.             kp2, des2 = sift.detectAndCompute(img2,None)
1335.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)
1336.             cv2.imshow("img2_kp",img2_kp)
1337.
1338.
1339.
1340.
1341.             # BFMatcher with default params
1342.             bf = cv2.BFMatcher()
1343.             matches = bf.knnMatch(des1,des2, k=2)
1344.
1345.
1346.
1347.
1348.
1349.             # Apply ratio test
1350.             good = []
1351.             rejected_points=[]
1352.             for m,n in matches:
1353.                 if m.distance < 0.8*n.distance:
1354.                     good.append([m])
1355.                 else:
1356.                     rejected_points.append([m])
1357.
1358.
1359.
1360.
1361.             end_time=time.time()-start_time
1362.
1363.             # cv2.drawMatchesKnn expects list of lists as matches.
1364.             img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
1365.
1366.             cv2.imshow("img3",img3)
1367.
1368.             # side-by-side images
1369.             img1_kp=cv2.drawKeypoints(img1,kp1,None,(255,0,0),0)    #0 to 4 to draw circles
1370.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)    #0 to 4 to draw circles
```

```
            #frameup = np.concatenate((img1_kp, img2_kp), axis=1)   #up
            #frames = np.concatenate((frameup, img3), axis=0)   #up+down
            #cv2.imshow("kp1-kp2",frames)


            sum_time=(sum_time+end_time)
            sum_kp1=(sum_kp1+len(kp1))
            sum_kp2=(sum_kp2+len(kp2))
            sum_good=(sum_good+len(good))
            sum_rejected_points=(sum_rejected_points+len(rejected_points))


            single_accuracy=(len(good)/(len(matches)))*100


            print("sift: %s:\tt(sec):%f\tkp1:%d\tkp2:%d\tG_matches:%d\trjct_pnts:%d\tacc:%f %%\n"%
            (name_of_images[i],
            end_time,
            len(kp1),
            len(kp2),
            len(good),
            len(rejected_points),
            single_accuracy))

            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SIFT_time:\t%.4f "%end_time+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SIFT_kp1:\t%.4f "%len(kp1)+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SIFT_kp2:\t%.4f "%len(kp2)+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SIFT_good:\t%.4f "%len(good)+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SIFT_accuracy:\t%.4f "%single_accuracy+"\n")


            #cv2.waitKey(1000)


            i=i+1



            if cv2.waitKey(0) & 0xFF == ord('n'):
                continue
            if cv2.waitKey(0) & 0xFF == ord('b'):
                i=i-2
                continue
            if cv2.waitKey(0) & 0xFF == ord('q'):
                break

        mean_time=sum_time/number_of_images
        mean_kp1=sum_kp1/number_of_images
        mean_kp2=sum_kp2/number_of_images
        mean_good=sum_good/number_of_images
        mean_rejected_points=sum_rejected_points/number_of_images
        accuracy=(mean_good/(len(matches)))*100



        ######################################################## saving to exel ############################
    #######
        #path to here
        Dir=os.path.dirname(os.path.abspath(__file__))
        #go to parent directory
        parentDir=os.path.abspath(os.path.join(Dir,".."))
```

```
        #go to sift folder or surf or ....
        pathX=os.path.abspath(os.path.join(parentDir,"exel"))

        print(Dir)
        print(pathX)
        print(glob.glob1(pathX,"*"))


        exel_file=pathX+'/'+'resault.xlsx'
        wb = load_workbook(exel_file)           #path to load exel file that was genetated befor by generator.py
        ws=wb['Sheet1']                     #take worksheet from workbook

        for k in range (1,40):                  #search for scale or rotation or skew in rows and accuracy in columns
            if (ws.cell(row=k, column=1).value)==path.rstrip("/") and(ws.cell(row=k, column=2).value)=='SIFT':
                for j in range(1,10):
                    if (ws.cell(row=1, column=j).value)=='time':
                        ws.cell(row=k, column=j).value=mean_time
                    if (ws.cell(row=1, column=j).value)=='kp1':
                        ws.cell(row=k, column=j).value=mean_kp1
                    if (ws.cell(row=1, column=j).value)=='kp2':
                        ws.cell(row=k, column=j).value=mean_kp2
                    if (ws.cell(row=1, column=j).value)=='good matches':
                        ws.cell(row=k, column=j).value=mean_good
                    if (ws.cell(row=1, column=j).value)=='accuracy':
                        ws.cell(row=k, column=j).value=accuracy
        wb.save(exel_file)

        print("\nsaving to file", end=" > > > > ")
        for i in range (1,30):
            print("> ", end="",flush=True)
            cv2.waitKey(20)
        print("\n")

        ###################################################### saving to exel #################################


        ##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv saving to text vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
        print("**********************************\n")

        print("SIFT_mean time:\t\t%.4f "%mean_time," sec")
        text_file.write("SIFT_mean_time:\t%.4f "%mean_time)

        print("SIFT_mean_kp1:\t\t%d"%mean_kp1)
        text_file.write("\nSIFT_mean_kp1:\t%d"%mean_kp1)

        print("SIFT_mean_kp2:\t\t%d"%mean_kp2)
        text_file.write("\nSIFT_mean_kp2:\t%d"%mean_kp2)

        print("SIFT_mean_good:\t\t%d"%mean_good)
        text_file.write("\nSIFT_mean_good:\t%d"%mean_good)

        print("SIFT_mean_rjctd_pnts:\t%d"%mean_rejected_points)
        text_file.write("\nSIFT_mean_rejected_points:\t%d"%mean_rejected_points)

        print("SIFT_accuracy:\t\t%.4f "%accuracy," %")
        text_file.write("\nSIFT_accuracy:\t%.4f "%accuracy)

        #^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^saving to text ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

        print("\n")
```

```
1491.         print("*********************************\n")
1492.         text=("")
1493.
1494.         ################################
1495.
```

8–7 کدهای آنالیز عکس و ذخیره داده‌ها در فایل اکسل SURF :

```
1496.         import numpy as np
1497.         import cv2
1498.         from matplotlib import pyplot as plt
1499.         import glob
1500.         import time
1501.
1502.         import openpyxl
1503.         from openpyxl import load_workbook
1504.         import os
1505.
1506.
1507.         path="skew/"
1508.         txt_name="surf_"+path.rstrip("/")+".txt"
1509.         txt_name_single="surf_single_"+path.rstrip("/")+".txt"
1510.
1511.         i=0
1512.         sum_time=0
1513.         sum_kp1=0
1514.         sum_kp2=0
1515.         sum_good=0
1516.         sum_rejected_points=0
1517.
1518.         win_size_w=320
1519.         win_size_h=500
1520.
1521.
1522.
1523.         number_of_images=len(glob.glob1(path,"*.jpg"))
1524.         name_of_images=glob.glob1(path,"*.jpg")
1525.
1526.
1527.         print("********************\n\n\n")
1528.
1529.         if (number_of_images==0):
1530.             print("********************\n")
1531.             print("No image present in *%s*"%path,">>> Abort\n")
1532.             print("********************\n\n\n")
1533.             quit()
1534.         else:
1535.             print("number of images=\t",number_of_images)
1536.             print("names=\t",name_of_images)
1537.
1538.         #img2=cv2.imread(name_of_images[1])
1539.         #cv2.namedWindow('imggg',cv2.WINDOW_NORMAL)
1540.         #cv2.resizeWindow('img', 600,320)
1541.         #cv2.imshow("imggg",img2)
1542.         #cv2.imshow("img0",cv2.imread(name_of_images[0]))
1543.
1544.         img1 = cv2.imread('arduino.jpg',0)  # queryImage
1545.         img1=cv2.resize(img1,(320,499))
1546.         #cv2.namedWindow('img1',cv2.WINDOW_NORMAL)
1547.         #cv2.resizeWindow('img1', win_size_w,win_size_h)
1548.         #cv2.imshow("img1",img1)
```

```
1549.
1550.         cv2.namedWindow('img2',cv2.WINDOW_NORMAL)
1551.         cv2.resizeWindow('img2', win_size_w,win_size_h)
1552.         cv2.namedWindow('img3',cv2.WINDOW_NORMAL)
1553.         cv2.resizeWindow('img3', win_size_w*2,win_size_h)
1554.         cv2.namedWindow('kp1-kp2',cv2.WINDOW_NORMAL)
1555.         cv2.resizeWindow('kp1-kp2', win_size_w,win_size_h)
1556.
1557.         text_file= open(path+txt_name,"w+")
1558.         text_file_single=open(path+txt_name_single,"w+")
1559.
1560.         ##################################################   surf   ##################################
    ##################
1561.         print("Press 'n' to go to the next image, 'q' to exit...\n\n")
1562.         while(i<number_of_images):
1563.             start_time=time.time()
1564.             image_name=path+name_of_images[i]
1565.
1566.             print(image_name)
1567.             img2=cv2.imread(image_name,0)
1568.             #img2=cv2.resize(img2,(320,499))
1569.
1570.
1571.             cv2.imshow("img2",img2)
1572.             cv2.waitKey(10)
1573.
1574.
1575.             # Initiate surf detector
1576.             #surf = cv2.SURF()
1577.             surf = cv2.xfeatures2d.SURF_create()
1578.             # find the keypoints and descriptors with SURF
1579.             kp1, des1 = surf.detectAndCompute(img1,None)
1580.             kp2, des2 = surf.detectAndCompute(img2,None)
1581.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)
1582.             cv2.imshow("img2_kp",img2_kp)
1583.
1584.
1585.
1586.
1587.             # BFMatcher with default params
1588.             bf = cv2.BFMatcher()
1589.             matches = bf.knnMatch(des1,des2, k=2)
1590.
1591.
1592.
1593.
1594.
1595.             # Apply ratio test
1596.             good = []
1597.             rejected_points=[]
1598.             for m,n in matches:
1599.                 if m.distance < 0.8*n.distance:
1600.                     good.append([m])
1601.                 else:
1602.                     rejected_points.append([m])
1603.
1604.
1605.
1606.
1607.             end_time=time.time()-start_time
1608.
1609.             # cv2.drawMatchesKnn expects list of lists as matches.
1610.             img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
1611.
1612.             cv2.imshow("img3",img3)
1613.
```

```
            # side-by-side images
            img1_kp=cv2.drawKeypoints(img1,kp1,None,(255,0,0),0)    #0 to 4 to draw circles
            img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)    #0 to 4 to draw circles
            #frameup = np.concatenate((img1_kp, img2_kp), axis=1)   #up
            #frames = np.concatenate((frameup, img3), axis=0)   #up+down
            #cv2.imshow("kp1-kp2",frames)


            sum_time=(sum_time+end_time)
            sum_kp1=(sum_kp1+len(kp1))
            sum_kp2=(sum_kp2+len(kp2))
            sum_good=(sum_good+len(good))
            sum_rejected_points=(sum_rejected_points+len(rejected_points))


            single_accuracy=(len(good)/(len(matches)))*100


            print("surf: %s:\tt(sec):%f\tkp1:%d\tkp2:%d\tG_matches:%d\trjct_pnts:%d\tacc:%f %%\n"%
            (name_of_images[i],
            end_time,
            len(kp1),
            len(kp2),
            len(good),
            len(rejected_points),
            single_accuracy))

            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SURF_time:\t%.4f "%end_time+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SURF_kp1:\t%.4f "%len(kp1)+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SURF_kp2:\t%.4f "%len(kp2)+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SURF_good:\t%.4f "%len(good)+"\n")
            text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"SURF_accuracy:\t%.4f "%single_accuracy+"\n")


            #cv2.waitKey(1000)


            i=i+1



            if cv2.waitKey(0) & 0xFF == ord('n'):
                continue
            if cv2.waitKey(0) & 0xFF == ord('b'):
                i=i-2
                continue
            if cv2.waitKey(0) & 0xFF == ord('q'):
                break

        mean_time=sum_time/number_of_images
        mean_kp1=sum_kp1/number_of_images
        mean_kp2=sum_kp2/number_of_images
        mean_good=sum_good/number_of_images
        mean_rejected_points=sum_rejected_points/number_of_images
        accuracy=(mean_good/(len(matches)))*100



        ######################################################## saving to exel ############################
        #######
        #path to here
```

```
1674.         Dir=os.path.dirname(os.path.abspath(__file__))
1675.         #go to parent directory
1676.         parentDir=os.path.abspath(os.path.join(Dir,".."))
1677.         #go to surf folder or surf or ....
1678.         pathX=os.path.abspath(os.path.join(parentDir,"exel"))
1679.
1680.         print(Dir)
1681.         print(pathX)
1682.         print(glob.glob1(pathX,"*"))
1683.
1684.
1685.         exel_file=pathX+'/'+'resault.xlsx'
1686.         wb = load_workbook(exel_file)           #path to load exel file that was genetated befor by gene
      rator.py
1687.         ws=wb['Sheet1']                    #take worksheet from workbook
1688.
1689.         for k in range (1,40):                  #search for scale or rotation or skew in rows and accura
      cy in columns
1690.             if (ws.cell(row=k, column=1).value)==path.rstrip("/") and(ws.cell(row=k, column=2).value)=='
      SURF':
1691.                 for j in range(1,10):
1692.                     if (ws.cell(row=1, column=j).value)=='time':
1693.                         ws.cell(row=k, column=j).value=mean_time
1694.                     if (ws.cell(row=1, column=j).value)=='kp1':
1695.                         ws.cell(row=k, column=j).value=mean_kp1
1696.                     if (ws.cell(row=1, column=j).value)=='kp2':
1697.                         ws.cell(row=k, column=j).value=mean_kp2
1698.                     if (ws.cell(row=1, column=j).value)=='good matches':
1699.                         ws.cell(row=k, column=j).value=mean_good
1700.                     if (ws.cell(row=1, column=j).value)=='accuracy':
1701.                         ws.cell(row=k, column=j).value=accuracy
1702.         wb.save(exel_file)
1703.
1704.         print("\nsaving to file", end=" > > > > ")
1705.         for i in range (1,30):
1706.             print("> ", end="",flush=True)
1707.             cv2.waitKey(20)
1708.         print("\n")
1709.
1710.         ########################################################## saving to exel ##########################
      #######
1711.
1712.
1713.         ##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv saving to text vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
      vvvvv
1714.         print("*********************************\n")
1715.
1716.         print("SURF_mean time:\t\t%.4f "%mean_time," sec")
1717.         text_file.write("SURF_mean_time:\t%.4f "%mean_time)
1718.
1719.         print("SURF_mean_kp1:\t\t%d"%mean_kp1)
1720.         text_file.write("\nSURF_mean_kp1:\t%d"%mean_kp1)
1721.
1722.         print("SURF_mean_kp2:\t\t%d"%mean_kp2)
1723.         text_file.write("\nSURF_mean_kp2:\t%d"%mean_kp2)
1724.
1725.         print("SURF_mean_good:\t\t%d"%mean_good)
1726.         text_file.write("\nSURF_mean_good:\t%d"%mean_good)
1727.
1728.         print("SURF_mean_rjctd_pnts:\t%d"%mean_rejected_points)
1729.         text_file.write("\nSURF_mean_rejected_points:\t%d"%mean_rejected_points)
1730.
1731.         print("SURF_accuracy:\t\t%.4f "%accuracy," %")
1732.         text_file.write("\nSURF_accuracy:\t%.4f "%accuracy)
1733.
1734.         #^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^saving to text ^^^^^^^^^^^^^^^^^^^^^^^^
```

```
1735.
1736.         print("\n")
1737.         print("*********************************\n")
1738.         text=("")
1739.
1740.         ###################################################   surf   ########################
```

9–7 کدهای آنالیز عکس و ذخیره داده‌ها در فایل اکسل ORB :

```
1741.         import numpy as np
1742.         import cv2
1743.         from matplotlib import pyplot as plt
1744.         import glob
1745.         import time
1746.
1747.         import openpyxl
1748.         from openpyxl import load_workbook
1749.         import os
1750.
1751.
1752.         path="skew/"
1753.         txt_name="orb_"+path.rstrip("/")+".txt"
1754.         txt_name_single="orb_single_"+path.rstrip("/")+".txt"
1755.
1756.         i=0
1757.         sum_time=0
1758.         sum_kp1=0
1759.         sum_kp2=0
1760.         sum_good=0
1761.         sum_rejected_points=0
1762.
1763.         win_size_w=320
1764.         win_size_h=500
1765.
1766.
1767.
1768.         number_of_images=len(glob.glob1(path,"*.jpg"))
1769.         name_of_images=glob.glob1(path,"*.jpg")
1770.
1771.
1772.         print("********************\n\n\n")
1773.
1774.         if (number_of_images==0):
1775.             print("********************\n")
1776.             print("No image present in *%s*"%path,">>> Abort\n")
1777.             print("********************\n\n\n")
1778.             quit()
1779.         else:
1780.             print("number of images=\t",number_of_images)
1781.             print("names=\t",name_of_images)
1782.
1783.         #img2=cv2.imread(name_of_images[1])
1784.         #cv2.namedWindow('imggg',cv2.WINDOW_NORMAL)
1785.         #cv2.resizeWindow('img', 600,320)
1786.         #cv2.imshow("imggg",img2)
1787.         #cv2.imshow("img0",cv2.imread(name_of_images[0]))
1788.
1789.         img1 = cv2.imread('arduino.jpg',0)  # queryImage
1790.         img1=cv2.resize(img1,(320,499))
1791.         #cv2.namedWindow('img1',cv2.WINDOW_NORMAL)
1792.         #cv2.resizeWindow('img1', win_size_w,win_size_h)
```

```
1793.         #cv2.imshow("img1",img1)
1794.
1795.         cv2.namedWindow('img2',cv2.WINDOW_NORMAL)
1796.         cv2.resizeWindow('img2', win_size_w,win_size_h)
1797.         cv2.namedWindow('img3',cv2.WINDOW_NORMAL)
1798.         cv2.resizeWindow('img3', win_size_w*2,win_size_h)
1799.         cv2.namedWindow('kp1-kp2',cv2.WINDOW_NORMAL)
1800.         cv2.resizeWindow('kp1-kp2', win_size_w,win_size_h)
1801.
1802.         text_file= open(path+txt_name,"w+")
1803.         text_file_single=open(path+txt_name_single,"w+")
1804.
1805.         ##################################################   orb   ##################################
     ##################
1806.         print("Press 'n' to go to the next image, 'q' to exit...\n\n")
1807.         while(i<number_of_images):
1808.             start_time=time.time()
1809.             image_name=path+name_of_images[i]
1810.
1811.             print(image_name)
1812.             img2=cv2.imread(image_name,0)
1813.             #img2=cv2.resize(img2,(320,499))
1814.
1815.
1816.             cv2.imshow("img2",img2)
1817.             cv2.waitKey(10)
1818.
1819.
1820.             # Initiate orb detector
1821.             #orb = cv2.ORB()
1822.             orb = cv2.ORB_create()
1823.             # find the keypoints and descriptors with ORB
1824.             kp1, des1 = orb.detectAndCompute(img1,None)
1825.             kp2, des2 = orb.detectAndCompute(img2,None)
1826.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)
1827.             cv2.imshow("img2_kp",img2_kp)
1828.
1829.
1830.
1831.
1832.             # BFMatcher with default params
1833.             bf = cv2.BFMatcher()
1834.             matches = bf.knnMatch(des1,des2, k=2)
1835.
1836.
1837.
1838.
1839.
1840.             # Apply ratio test
1841.             good = []
1842.             rejected_points=[]
1843.             for m,n in matches:
1844.                 if m.distance < 0.8*n.distance:
1845.                     good.append([m])
1846.                 else:
1847.                     rejected_points.append([m])
1848.
1849.
1850.
1851.
1852.             end_time=time.time()-start_time
1853.
1854.             # cv2.drawMatchesKnn expects list of lists as matches.
1855.             img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
1856.
1857.             cv2.imshow("img3",img3)
```

```
1858.
1859.             # side-by-side images
1860.             img1_kp=cv2.drawKeypoints(img1,kp1,None,(255,0,0),0)    #0 to 4 to draw circles
1861.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)    #0 to 4 to draw circles
1862.             #frameup = np.concatenate((img1_kp, img2_kp), axis=1)   #up
1863.             #frames = np.concatenate((frameup, img3), axis=0)   #up+down
1864.             #cv2.imshow("kp1-kp2",frames)
1865.
1866.
1867.             sum_time=(sum_time+end_time)
1868.             sum_kp1=(sum_kp1+len(kp1))
1869.             sum_kp2=(sum_kp2+len(kp2))
1870.             sum_good=(sum_good+len(good))
1871.             sum_rejected_points=(sum_rejected_points+len(rejected_points))
1872.
1873.
1874.             single_accuracy=(len(good)/(len(matches)))*100
1875.
1876.
1877.             print("orb: %s:\tt(sec):%f\tkp1:%d\tkp2:%d\tG_matches:%d\trjct_pnts:%d\tacc:%f %%\n"%
1878.             (name_of_images[i],
1879.             end_time,
1880.             len(kp1),
1881.             len(kp2),
1882.             len(good),
1883.             len(rejected_points),
1884.             single_accuracy))
1885.
1886.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"ORB_time:\
    t%.4f "%end_time+"\n")
1887.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"ORB_kp1:\t
    %.4f "%len(kp1)+"\n")
1888.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"ORB_kp2:\t
    %.4f "%len(kp2)+"\n")
1889.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"ORB_good:\
    t%.4f "%len(good)+"\n")
1890.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"ORB_accura
    cy:\t%.4f "%single_accuracy+"\n")
1891.
1892.
1893.             #cv2.waitKey(1000)
1894.
1895.
1896.             i=i+1
1897.
1898.
1899.
1900.             if cv2.waitKey(0) & 0xFF == ord('n'):
1901.                 continue
1902.             if cv2.waitKey(0) & 0xFF == ord('b'):
1903.                 i=i-2
1904.                 continue
1905.             if cv2.waitKey(0) & 0xFF == ord('q'):
1906.                 break
1907.
1908.         mean_time=sum_time/number_of_images
1909.         mean_kp1=sum_kp1/number_of_images
1910.         mean_kp2=sum_kp2/number_of_images
1911.         mean_good=sum_good/number_of_images
1912.         mean_rejected_points=sum_rejected_points/number_of_images
1913.         accuracy=(mean_good/(len(matches)))*100
1914.
1915.
1916.
1917.         ######################################################## saving to exel ############################
    #######
```

ت‌ت

```
        #path to here
        Dir=os.path.dirname(os.path.abspath(__file__))
        #go to parent directory
        parentDir=os.path.abspath(os.path.join(Dir,".."))
        #go to orb folder or orb or ....
        pathX=os.path.abspath(os.path.join(parentDir,"exel"))

        print(Dir)
        print(pathX)
        print(glob.glob1(pathX,"*"))


        exel_file=pathX+'/'+'resault.xlsx'
        wb = load_workbook(exel_file)           #path to load exel file that was genetated befor by generator.py
        ws=wb['Sheet1']                    #take worksheet from workbook

        for k in range (1,40):                  #search for scale or rotation or skew in rows and accuracy in columns
            if (ws.cell(row=k, column=1).value)==path.rstrip("/") and(ws.cell(row=k, column=2).value)=='ORB':
                for j in range(1,10):
                    if (ws.cell(row=1, column=j).value)=='time':
                        ws.cell(row=k, column=j).value=mean_time
                    if (ws.cell(row=1, column=j).value)=='kp1':
                        ws.cell(row=k, column=j).value=mean_kp1
                    if (ws.cell(row=1, column=j).value)=='kp2':
                        ws.cell(row=k, column=j).value=mean_kp2
                    if (ws.cell(row=1, column=j).value)=='good matches':
                        ws.cell(row=k, column=j).value=mean_good
                    if (ws.cell(row=1, column=j).value)=='accuracy':
                        ws.cell(row=k, column=j).value=accuracy
        wb.save(exel_file)

        print("\nsaving to file", end=" > > > > ")
        for i in range (1,30):
            print("> ", end="",flush=True)
            cv2.waitKey(20)
        print("\n")

        ######################################################## saving to exel ############################
        #######


        ##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv saving to text vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
        print("*********************************\n")

        print("ORB_mean time:\t\t%.4f "%mean_time," sec")
        text_file.write("ORB_mean_time:\t%.4f "%mean_time)

        print("ORB_mean_kp1:\t\t%d"%mean_kp1)
        text_file.write("\nORB_mean_kp1:\t%d"%mean_kp1)

        print("ORB_mean_kp2:\t\t%d"%mean_kp2)
        text_file.write("\nORB_mean_kp2:\t%d"%mean_kp2)

        print("ORB_mean_good:\t\t%d"%mean_good)
        text_file.write("\nORB_mean_good:\t%d"%mean_good)

        print("ORB_mean_rjctd_pnts:\t%d"%mean_rejected_points)
        text_file.write("\nORB_mean_rejected_points:\t%d"%mean_rejected_points)

        print("ORB_accuracy:\t\t%.4f "%accuracy," %")
        text_file.write("\nORB_accuracy:\t%.4f "%accuracy)

```

```
#^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^saving to text ^^^^^^^^^^^^^^^^^^^^^^

print("\n")
print("**********************************\n")
text=("")

####################################################   orb   #########################

```

7–10کدهای آنالیز عکس و ذخیره داده‌ها در فایل اکسل BRISK :

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import glob
import time

import openpyxl
from openpyxl import load_workbook
import os


path="scale/"
txt_name="brisk_"+path.rstrip("/")+".txt"
txt_name_single="brisk_single_"+path.rstrip("/")+".txt"

i=0
sum_time=0
sum_kp1=0
sum_kp2=0
sum_good=0
sum_rejected_points=0

win_size_w=320
win_size_h=500



number_of_images=len(glob.glob1(path,"*.jpg"))
name_of_images=glob.glob1(path,"*.jpg")


print("********************\n\n\n")

if (number_of_images==0):
    print("********************\n")
    print("No image present in *%s*"%path,">>> Abort\n")
    print("********************\n\n\n")
    quit()
else:
    print("number of images=\t",number_of_images)
    print("names=\t",name_of_images)

#img2=cv2.imread(name_of_images[1])
#cv2.namedWindow('imggg',cv2.WINDOW_NORMAL)
#cv2.resizeWindow('img', 600,320)
#cv2.imshow("imggg",img2)
#cv2.imshow("img0",cv2.imread(name_of_images[0]))

img1 = cv2.imread('arduino.jpg',0)  # queryImage
img1=cv2.resize(img1,(320,499))
```

```
2037.         #cv2.namedWindow('img1',cv2.WINDOW_NORMAL)
2038.         #cv2.resizeWindow('img1', win_size_w,win_size_h)
2039.         #cv2.imshow("img1",img1)
2040.
2041.         cv2.namedWindow('img2',cv2.WINDOW_NORMAL)
2042.         cv2.resizeWindow('img2', win_size_w,win_size_h)
2043.         cv2.namedWindow('img3',cv2.WINDOW_NORMAL)
2044.         cv2.resizeWindow('img3', win_size_w*2,win_size_h)
2045.         cv2.namedWindow('kp1-kp2',cv2.WINDOW_NORMAL)
2046.         cv2.resizeWindow('kp1-kp2', win_size_w,win_size_h)
2047.
2048.         text_file= open(path+txt_name,"w+")
2049.         text_file_single=open(path+txt_name_single,"w+")
2050.
2051.         ##################################################   brisk   ##################################
    ###################
2052.         print("Press 'n' to go to the next image, 'q' to exit...\n\n")
2053.         while(i<number_of_images):
2054.             start_time=time.time()
2055.             image_name=path+name_of_images[i]
2056.
2057.             print(image_name)
2058.             img2=cv2.imread(image_name,0)
2059.             #img2=cv2.resize(img2,(320,499))
2060.
2061.
2062.             cv2.imshow("img2",img2)
2063.             cv2.waitKey(10)
2064.
2065.
2066.             # Initiate brisk detector
2067.             #sift = cv2.xfeatures2d.SIFT_create()
2068.             brisk = cv2.BRISK_create(60)
2069.             # find the keypoints and descriptors with brisk
2070.             kp1, des1 = brisk.detectAndCompute(img1,None)
2071.             kp2, des2 = brisk.detectAndCompute(img2,None)
2072.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)
2073.             cv2.imshow("img2_kp",img2_kp)
2074.
2075.
2076.
2077.
2078.             # BFMatcher with default params
2079.             bf = cv2.BFMatcher()
2080.             matches = bf.knnMatch(des1,des2, k=2)
2081.
2082.
2083.
2084.
2085.
2086.             # Apply ratio test
2087.             good = []
2088.             rejected_points=[]
2089.             for m,n in matches:
2090.                 if m.distance < 0.7*n.distance:
2091.                     good.append([m])
2092.                 else:
2093.                     rejected_points.append([m])
2094.
2095.
2096.
2097.
2098.             end_time=time.time()-start_time
2099.
2100.             # cv2.drawMatchesKnn expects list of lists as matches.
2101.             img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
```

```
2102.
2103.             cv2.imshow("img3",img3)
2104.
2105.             # side-by-side images
2106.             img1_kp=cv2.drawKeypoints(img1,kp1,None,(255,0,0),0)    #0 to 4 to draw circles
2107.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)    #0 to 4 to draw circles
2108.             #frameup = np.concatenate((img1_kp, img2_kp), axis=1)   #up
2109.             #frames = np.concatenate((frameup, img3), axis=0)   #up+down
2110.             #cv2.imshow("kp1-kp2",frames)
2111.
2112.
2113.             sum_time=(sum_time+end_time)
2114.             sum_kp1=(sum_kp1+len(kp1))
2115.             sum_kp2=(sum_kp2+len(kp2))
2116.             sum_good=(sum_good+len(good))
2117.             sum_rejected_points=(sum_rejected_points+len(rejected_points))
2118.
2119.
2120.             single_accuracy=(len(good)/(len(matches)))*100
2121.
2122.
2123.             print("brisk: %s:\tt(sec):%f\tkp1:%d\tkp2:%d\tG_matches:%d\trjct_pnts:%d\tacc:%f %%\n"%
2124.             (name_of_images[i],
2125.             end_time,
2126.             len(kp1),
2127.             len(kp2),
2128.             len(good),
2129.             len(rejected_points),
2130.             single_accuracy))
2131.
2132.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"BRISK_time
    :\t%.4f "%end_time+"\n")
2133.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"BRISK_kp1:
    \t%.4f "%len(kp1)+"\n")
2134.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"BRISK_kp2:
    \t%.4f "%len(kp2)+"\n")
2135.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"BRISK_good
    :\t%.4f "%len(good)+"\n")
2136.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"BRISK_accu
    racy:\t%.4f "%single_accuracy+"\n")
2137.
2138.
2139.             #cv2.waitKey(1000)
2140.
2141.
2142.             i=i+1
2143.
2144.
2145.
2146.             if cv2.waitKey(0) & 0xFF == ord('n'):
2147.                 continue
2148.             if cv2.waitKey(0) & 0xFF == ord('b'):
2149.                 i=i-2
2150.                 continue
2151.             if cv2.waitKey(0) & 0xFF == ord('q'):
2152.                 break
2153.
2154.         mean_time=sum_time/number_of_images
2155.         mean_kp1=sum_kp1/number_of_images
2156.         mean_kp2=sum_kp2/number_of_images
2157.         mean_good=sum_good/number_of_images
2158.         mean_rejected_points=sum_rejected_points/number_of_images
2159.         accuracy=(mean_good/(len(matches)))*100
2160.
2161.
2162.
```

```
        ######################################################## saving to exel ##########################
    #######
        #path to here
        Dir=os.path.dirname(os.path.abspath(__file__))
        #go to parent directory
        parentDir=os.path.abspath(os.path.join(Dir,".."))
        #go to brisk folder or surf or ....
        pathX=os.path.abspath(os.path.join(parentDir,"exel"))

        print(Dir)
        print(pathX)
        print(glob.glob1(pathX,"*"))


        exel_file=pathX+'/'+'resault.xlsx'
        wb = load_workbook(exel_file)           #path to load exel file that was genetated befor by gene
    rator.py
        ws=wb['Sheet1']                    #take worksheet from workbook

        for k in range (1,40):                  #search for scale or rotation or skew in rows and accura
    cy in columns
            if (ws.cell(row=k, column=1).value)==path.rstrip("/") and(ws.cell(row=k, column=2).value)=='
    BRISK':
                for j in range(1,10):
                    if (ws.cell(row=1, column=j).value)=='time':
                        ws.cell(row=k, column=j).value=mean_time
                    if (ws.cell(row=1, column=j).value)=='kp1':
                        ws.cell(row=k, column=j).value=mean_kp1
                    if (ws.cell(row=1, column=j).value)=='kp2':
                        ws.cell(row=k, column=j).value=mean_kp2
                    if (ws.cell(row=1, column=j).value)=='good matches':
                        ws.cell(row=k, column=j).value=mean_good
                    if (ws.cell(row=1, column=j).value)=='accuracy':
                        ws.cell(row=k, column=j).value=accuracy
        wb.save(exel_file)

        print("\nsaving to file", end=" > > > > ")
        for i in range (1,30):
            print("> ", end="",flush=True)
            cv2.waitKey(20)
        print("\n")

        ######################################################## saving to exel ##########################
    #######


        ##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv saving to text vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
    vvvvv
        print("*********************************\n")

        print("BRISK_mean time:\t\t%.4f "%mean_time," sec")
        text_file.write("BRISK_mean_time:\t%.4f "%mean_time)

        print("BRISK_mean_kp1:\t\t%d"%mean_kp1)
        text_file.write("\nBRISK_mean_kp1:\t%d"%mean_kp1)

        print("BRISK_mean_kp2:\t\t%d"%mean_kp2)
        text_file.write("\nBRISK_mean_kp2:\t%d"%mean_kp2)

        print("BRISK_mean_good:\t\t%d"%mean_good)
        text_file.write("\nBRISK_mean_good:\t%d"%mean_good)

        print("BRISK_mean_rjctd_pnts:\t%d"%mean_rejected_points)
        text_file.write("\nBRISK_mean_rejected_points:\t%d"%mean_rejected_points)

        print("BRISK_accuracy:\t\t%.4f "%accuracy," %")
```

```
        text_file.write("\nBRISK_accuracy:\t%.4f "%accuracy)

        #^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^saving to text ^^^^^^^^^^^^^^^^^^^^^^^^^^

        print("\n")
        print("**********************************\n")
        text=("")

        ##################################################   brisk   ##############################
```

7–11کدهای آنالیز عکس و ذخیره داده‌ها در فایل اکسل FAST :

```
        import numpy as np
        import cv2
        from matplotlib import pyplot as plt
        import glob
        import time

        import openpyxl
        from openpyxl import load_workbook
        import os


        path="rotation/"
        txt_name="fast_"+path.rstrip("/")+".txt"
        txt_name_single="fast_single_"+path.rstrip("/")+".txt"

        i=0
        sum_time=0
        sum_kp1=0
        sum_kp2=0
        sum_good=0
        sum_rejected_points=0

        win_size_w=320
        win_size_h=500



        number_of_images=len(glob.glob1(path,"*.jpg"))
        name_of_images=glob.glob1(path,"*.jpg")


        print("********************\n\n\n")

        if (number_of_images==0):
            print("********************\n")
            print("No image present in *%s*"%path,">>> Abort\n")
            print("********************\n\n\n")
            quit()
        else:
            print("number of images=\t",number_of_images)
            print("names=\t",name_of_images)

        #img2=cv2.imread(name_of_images[1])
        #cv2.namedWindow('imggg',cv2.WINDOW_NORMAL)
        #cv2.resizeWindow('img', 600,320)
        #cv2.imshow("imggg",img2)
        #cv2.imshow("img0",cv2.imread(name_of_images[0]))
```

```
        img1 = cv2.imread('arduino.jpg',0)  # queryImage
        img1=cv2.resize(img1,(320,499))
        #cv2.namedWindow('img1',cv2.WINDOW_NORMAL)
        #cv2.resizeWindow('img1', win_size_w,win_size_h)
        #cv2.imshow("img1",img1)

        cv2.namedWindow('img2',cv2.WINDOW_NORMAL)
        cv2.resizeWindow('img2', win_size_w,win_size_h)
        cv2.namedWindow('img3',cv2.WINDOW_NORMAL)
        cv2.resizeWindow('img3', win_size_w*2,win_size_h)
        cv2.namedWindow('kp1-kp2',cv2.WINDOW_NORMAL)
        cv2.resizeWindow('kp1-kp2', win_size_w,win_size_h)

        text_file= open(path+txt_name,"w+")
        text_file_single=open(path+txt_name_single,"w+")

        ###################################################   fast   ##################################
    ##################
        print("Press 'n' to go to the next image, 'q' to exit...\n\n")
        while(i<number_of_images):
            start_time=time.time()
            image_name=path+name_of_images[i]

            print(image_name)
            img2=cv2.imread(image_name,0)
            #img2=cv2.resize(img2,(320,499))


            cv2.imshow("img2",img2)


            ###########################################################################
            # Initiate fast detector >>>>> orb descriptor
            #sift = cv2.xfeatures2d.SIFT_create()
            fast = cv2.FastFeatureDetector_create(50)              #detect by fast
            surf = cv2.xfeatures2d.SURF_create()               #descriptor orb
            kp1 = fast.detect(img1 ,None)
            kp2 = fast.detect(img2 ,None)
            # find the keypoints and descriptors with orb
            kp1, des1 = surf.compute(img1,  kp1)
            kp2, des2 = surf.compute(img2,  kp2)
            ###############################################################################
            '''''
            # Initiate fast detector >>>>> brief descriptor
            fast = cv2.FastFeatureDetector_create()             #detect by fast
            brief= cv2.xfeatures2d.BriefDescriptorExtractor_create()    #brief descriptor
            kp1 = fast.detect(img1 ,None)
            kp2 = fast.detect(img2 ,None)
            # find the keypoints and descriptors with orb
            kp1, des1 = brief.compute(img1,  kp1)
            kp2, des2 = brief.compute(img2,  kp2)
            '''


            img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)
            cv2.imshow("img2_kp",img2_kp)




            # BFMatcher with default params
            bf = cv2.BFMatcher()
            matches = bf.knnMatch(des1,des2, k=2)


```

```
2346.
2347.
2348.             # Apply ratio test
2349.             good = []
2350.             rejected_points=[]
2351.             for m,n in matches:
2352.                 if m.distance < 0.77*n.distance:
2353.                     good.append([m])
2354.                 else:
2355.                     rejected_points.append([m])
2356.
2357.
2358.
2359.
2360.             end_time=time.time()-start_time
2361.
2362.             # cv2.drawMatchesKnn expects list of lists as matches.
2363.             img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,good,None,flags=2)
2364.
2365.             cv2.imshow("img3",img3)
2366.
2367.             # side-by-side images
2368.             img1_kp=cv2.drawKeypoints(img1,kp1,None,(255,0,0),0)    #0 to 4 to draw circles
2369.             img2_kp=cv2.drawKeypoints(img2,kp2,None,(255,0,0),0)    #0 to 4 to draw circles
2370.             #frameup = np.concatenate((img1_kp, img2_kp), axis=1)   #up
2371.             #frames = np.concatenate((frameup, img3), axis=0)   #up+down
2372.             #cv2.imshow("kp1-kp2",frames)
2373.
2374.
2375.             sum_time=(sum_time+end_time)
2376.             sum_kp1=(sum_kp1+len(kp1))
2377.             sum_kp2=(sum_kp2+len(kp2))
2378.             sum_good=(sum_good+len(good))
2379.             sum_rejected_points=(sum_rejected_points+len(rejected_points))
2380.
2381.
2382.             single_accuracy=(len(good)/(len(matches)))*100
2383.
2384.
2385.             print("fast: %s:\tt(sec):%f\tkp1:%d\tkp2:%d\tG_matches:%d\trjct_pnts:%d\tacc:%f %%\n"%
2386.             (name_of_images[i],
2387.             end_time,
2388.             len(kp1),
2389.             len(kp2),
2390.             len(good),
2391.             len(rejected_points),
2392.             single_accuracy))
2393.
2394.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"FAST_time:
    \t%.4f "%end_time+"\n")
2395.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"FAST_kp1:\
    t%.4f "%len(kp1)+"\n")
2396.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"FAST_kp2:\
    t%.4f "%len(kp2)+"\n")
2397.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"FAST_good:
    \t%.4f "%len(good)+"\n")
2398.             text_file_single.write(path.rstrip("/")+":"+name_of_images[i].rstrip(".jpg")+":"+"FAST_accur
    acy:\t%.4f "%single_accuracy+"\n")
2399.
2400.
2401.             #cv2.waitKey(1000)
2402.
2403.
2404.             i=i+1
2405.
2406.
```

```
2407.
2408.             if cv2.waitKey(0) & 0xFF == ord('n'):
2409.                 continue
2410.             if cv2.waitKey(0) & 0xFF == ord('b'):
2411.                 i=i-2
2412.                 continue
2413.             if cv2.waitKey(0) & 0xFF == ord('q'):
2414.                 break
2415.
2416.         mean_time=sum_time/number_of_images
2417.         mean_kp1=sum_kp1/number_of_images
2418.         mean_kp2=sum_kp2/number_of_images
2419.         mean_good=sum_good/number_of_images
2420.         mean_rejected_points=sum_rejected_points/number_of_images
2421.         accuracy=(mean_good/(len(matches)))*100
2422.
2423.
2424.
2425.         ###################################################### saving to exel ########################
     #######
2426.         #path to here
2427.         Dir=os.path.dirname(os.path.abspath(__file__))
2428.         #go to parent directory
2429.         parentDir=os.path.abspath(os.path.join(Dir,".."))
2430.         #go to fast folder or fast or ....
2431.         pathX=os.path.abspath(os.path.join(parentDir,"exel"))
2432.
2433.         print(Dir)
2434.         print(pathX)
2435.         print(glob.glob1(pathX,"*"))
2436.
2437.
2438.         exel_file=pathX+'/'+'resault.xlsx'
2439.         wb = load_workbook(exel_file)           #path to load exel file that was genetated befor by gene
     rator.py
2440.         ws=wb['Sheet1']                     #take worksheet from workbook
2441.
2442.         for k in range (1,40):                  #search for scale or rotation or skew in rows and accura
     cy in columns
2443.             if (ws.cell(row=k, column=1).value)==path.rstrip("/") and(ws.cell(row=k, column=2).value)=='
     FAST':
2444.                 for j in range(1,10):
2445.                     if (ws.cell(row=1, column=j).value)=='time':
2446.                         ws.cell(row=k, column=j).value=mean_time
2447.                     if (ws.cell(row=1, column=j).value)=='kp1':
2448.                         ws.cell(row=k, column=j).value=mean_kp1
2449.                     if (ws.cell(row=1, column=j).value)=='kp2':
2450.                         ws.cell(row=k, column=j).value=mean_kp2
2451.                     if (ws.cell(row=1, column=j).value)=='good matches':
2452.                         ws.cell(row=k, column=j).value=mean_good
2453.                     if (ws.cell(row=1, column=j).value)=='accuracy':
2454.                         ws.cell(row=k, column=j).value=accuracy
2455.         wb.save(exel_file)
2456.
2457.         print("\nsaving to file", end=" > > > > ")
2458.         for i in range (1,30):
2459.             print("> ", end="",flush=True)
2460.             cv2.waitKey(20)
2461.         print("\n")
2462.
2463.         ###################################################### saving to exel ########################
     #######
2464.
2465.
2466.         ##vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv saving to text vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
     vvvvv
```

```
2467.         print("*********************************\n")
2468.
2469.         print("FAST_mean time:\t\t%.4f "%mean_time," sec")
2470.         text_file.write("FAST_mean_time:\t%.4f "%mean_time)
2471.
2472.         print("FAST_mean_kp1:\t\t%d"%mean_kp1)
2473.         text_file.write("\nFAST_mean_kp1:\t%d"%mean_kp1)
2474.
2475.         print("FAST_mean_kp2:\t\t%d"%mean_kp2)
2476.         text_file.write("\nFAST_mean_kp2:\t%d"%mean_kp2)
2477.
2478.         print("FAST_mean_good:\t\t%d"%mean_good)
2479.         text_file.write("\nFAST_mean_good:\t%d"%mean_good)
2480.
2481.         print("FAST_mean_rjctd_pnts:\t%d"%mean_rejected_points)
2482.         text_file.write("\nFAST_mean_rejected_points:\t%d"%mean_rejected_points)
2483.
2484.         print("FAST_accuracy:\t\t%.4f "%accuracy," %")
2485.         text_file.write("\nFAST_accuracy:\t%.4f "%accuracy)
2486.
2487.         #^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^saving to text ^^^^^^^^^^^^^^^^^^^^^^^^^^
2488.
2489.         print("\n")
2490.         print("*********************************\n")
2491.         text=("")
2492.
2493.         ##################################################   fast   ##########################
```

7–12کدهای ذخیره داده‌های تصاویر از فایل متنی:

```
2494.         import xlsxwriter
2495.         import openpyxl
2496.         from openpyxl import load_workbook
2497.         import os
2498.
2499.
2500.         wb = load_workbook('single_resault.xlsx')
2501.         ws=wb['Sheet1']
2502.
2503.
2504.         #algorithm='surf'
2505.         #algorithm_C='SURF'
2506.         condition=['rotation','scale','skew']
2507.
2508.
2509.
2510.
2511.
2512.
2513.         time=0
2514.         kp1=0
2515.         kp2=0
2516.         accuracy=0
2517.         good=0
2518.
2519.         rotation_file = ''
2520.         scale_file = ''
2521.         skew_file =''
2522.
2523.
```

```
2524.         def read_path(algorithm_C):
2525.
2526.                   global rotation_file
2527.                   global scale_file
2528.                   global skew_file
2529.
2530.                   algorithm=algorithm_C.lower()
2531.                   Dir=os.path.dirname(os.path.abspath(__file__))
2532.                   #go to parent directory
2533.                   parentDir=os.path.abspath(os.path.join(Dir,".."))
2534.                   #go to sift folder or surf or ....
2535.                   pathX=os.path.abspath(os.path.join(parentDir,algorithm))
2536.
2537.                   path_rotation=os.path.abspath(os.path.join(pathX,"rotation"))
2538.                   path_scale=os.path.abspath(os.path.join(pathX,"scale"))
2539.                   path_skew=os.path.abspath(os.path.join(pathX,"skew"))
2540.
2541.                   txt_name_rotation=algorithm+"_"+"single"+"_"+"rotation"+".txt"
2542.                   txt_name_scale=algorithm+"_"+"single"+"_"+"scale"+".txt"
2543.                   txt_name_skew=algorithm+"_"+"single"+"_"+"skew"+".txt"
2544.
2545.                   rotation_file=path_rotation+"/"+txt_name_rotation
2546.                   scale_file=path_scale+"/"+txt_name_scale
2547.                   skew_file=path_skew+"/"+txt_name_skew
2548.
2549.                   rotation_file = open(rotation_file, "r")
2550.                   scale_file = open(scale_file, "r")
2551.                   skew_file = open(skew_file, "r")
2552.                   return(rotation_file,scale_file,skew_file)
2553.
2554.
2555.
2556.         time_array=[]
2557.         kp1_array=[]
2558.         kp2_array=[]
2559.         good_array=[]
2560.         accuracy_array=[]
2561.         cond=[]  #condition
2562.         numb=[]  #meghdare condition
2563.
2564.
2565.
2566.         def get_data(algorithm_C,_file):
2567.
2568.
2569.                   split_lines=_file.read().splitlines()
2570.
2571.                   #print(split_lines)
2572.                   for i in split_lines:
2573.                            data=i.split(':')
2574.                            #print(data[0],data[1],data[2],data[3],"\n")
2575.
2576.                            cond.append(data[0])
2577.                            numb.append(data[1])
2578.
2579.                            if (data[2]==algorithm_C+"_"+'time'):
2580.                                     time=data[3].lstrip("\t")
2581.                                     #print(time)
2582.                                     time_array.append(time)
2583.                            elif (data[2]==algorithm_C+"_"+'kp1'):
2584.                                     kp1=data[3].lstrip("\t")
2585.                                     kp1_array.append(kp1)
2586.                            elif (data[2]==algorithm_C+"_"+'kp2'):
2587.                                     kp2=data[3].lstrip("\t")
2588.                                     kp2_array.append(kp2)
2589.                            elif (data[2]==algorithm_C+"_"+'good'):
```

```
                                        good=data[3].lstrip("\t")
                                        good_array.append(good)
                                elif (data[2]==algorithm_C+"_"+'accuracy'):
                                        accuracy=data[3].lstrip("\t")
                                        accuracy_array.append(accuracy)
                                #img_data=[data[0],data[1],data[2],data[3]]

                        return(time_array,
                        kp1_array,
                        kp2_array,
                        good_array,
                        accuracy_array,
                        cond,
                        numb)



                def write_data(algorithm_C,condition,_file):
                        i=0
                        cnt=0

                        for i in range (1,200):                                         #write mean_time to
    exel
                                if (ws.cell(row=i, column=1).value)==algorithm_C and (ws.cell(row=i,
    column=2).value)==condition:


                                        ws.cell(row=i, column=3).value=float(numb[cnt*5])
                                        ws.cell(row=i, column=4).value=float(time_array[cnt])
                                        ws.cell(row=i, column=5).value=float(kp1_array[cnt])
                                        ws.cell(row=i, column=6).value=float(kp2_array[cnt])
                                        ws.cell(row=i, column=7).value=float(good_array[cnt])
                                        ws.cell(row=i, column=8).value=float(accuracy_array[cnt])

                                        cnt=cnt+1
                        #print("numb=",numb)
                        time_array.clear()
                        kp1_array.clear()
                        kp2_array.clear()
                        good_array.clear()
                        accuracy_array.clear()
                        cond.clear()       #condition
                        numb.clear()       #meghdare condition


                def update(algorithm_C):
                        for i in condition:
                                '''
                                time_array=[]
                                kp1_array=[]
                                kp2_array=[]
                                good_array=[]
                                accuracy_array=[]
                                cond=[]  #condition
                                numb=[]  #meghdare condition
                                '''
                                rotation_file = read_path(algorithm_C)[0]
                                scale_file = read_path(algorithm_C)[1]
                                skew_file = read_path(algorithm_C)[2]

                                _file=eval(i+"_file")

                                time_array=get_data(algorithm_C,_file)[0]
                                kp1_array=get_data(algorithm_C,_file)[1]
                                kp2_array=get_data(algorithm_C,_file)[2]
```

```
                            good_array=get_data(algorithm_C,_file)[3]
                            accuracy_array=get_data(algorithm_C,_file)[4]
                            cond=get_data(algorithm_C,_file)[5]  #condition
                            numb=get_data(algorithm_C,_file)[6]  #meghdare condition


                            write_data(algorithm_C,i,_file)









        ################################################################################

        update('SURF')
        update('SIFT')
        update('ORB')
        update('FAST')
        update('BRISK')




        wb.save('single_resault.xlsx')
        print("\n>>> Done \n")

        ################################################################################









        '''
        #VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV Rotation
    VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
        time_array=[]
        kp1_array=[]
        kp2_array=[]
        good_array=[]
        accuracy_array=[]
        cond=[]  #condition
        numb=[]  #meghdare condition

        time_array=get_data(rotation_file)[0]
        kp1_array=get_data(rotation_file)[1]
        kp2_array=get_data(rotation_file)[2]
        good_array=get_data(rotation_file)[3]
        accuracy_array=get_data(rotation_file)[4]
        cond=get_data(rotation_file)[5]      #condition
        numb=get_data(rotation_file)[6]      #meghdare condition

        write_data(algorithm_C,'rotation',rotation_file)


```

```
2719.
2720.         #VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV SCALE
    VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV
2721.         time_array=[]
2722.         kp1_array=[]
2723.         kp2_array=[]
2724.         good_array=[]
2725.         accuracy_array=[]
2726.         cond=[]  #condition
2727.         numb=[]  #meghdare condition
2728.
2729.         time_array=get_data(scale_file)[0]
2730.         kp1_array=get_data(scale_file)[1]
2731.         kp2_array=get_data(scale_file)[2]
2732.         good_array=get_data(scale_file)[3]
2733.         accuracy_array=get_data(scale_file)[4]
2734.         cond=get_data(scale_file)[5]#condition
2735.         numb=get_data(scale_file)[6]#meghdare condition
2736.
2737.         write_data(algorithm_C,'scale',scale_file)
2738.
2739.
2740.         ########################################################## SKEW
    ########################################
2741.
2742.
2743.         time_array=[]
2744.         kp1_array=[]
2745.         kp2_array=[]
2746.         good_array=[]
2747.         accuracy_array=[]
2748.         cond=[]  #condition
2749.         numb=[]  #meghdare condition
2750.
2751.
2752.
2753.         time_array=get_data(skew_file)[0]
2754.         kp1_array=get_data(skew_file)[1]
2755.         kp2_array=get_data(skew_file)[2]
2756.         good_array=get_data(skew_file)[3]
2757.         accuracy_array=get_data(skew_file)[4]
2758.         cond=get_data(skew_file)[5] #condition
2759.         numb=get_data(skew_file)[6] #meghdare condition
2760.         print(numb)
2761.         write_data(algorithm_C,'skew',skew_file)
2762.
2763.
2764.         ###############################################################################
```

7–13کدهای ذخیره نمودارها از داده‌های آنالیز شده:

```
2765.         import matplotlib.pyplot as plt
2766.         import openpyxl
2767.         from openpyxl import load_workbook
2768.         import os
2769.         import time
2770.         import numpy as np
2771.
2772.
2773.         wb = load_workbook('single_resault.xlsx')
2774.         ws=wb['Sheet1']
2775.
2776.         wb_resault = load_workbook('resault.xlsx')
2777.         ws_resault=wb_resault['Sheet1']
2778.
2779.
2780.         number_of_images=7
2781.         x_array=[]
2782.         y_array=[]
2783.         z_array=[]
2784.
2785.
2786.         def plot(xdata,ydata,x_title,y_title):
2787.             plt.plot(xdata,ydata)
2788.             plt.xlabel(x_title)
2789.             plt.ylabel(y_title)
2790.             #save_plot(plt,'2')
2791.             fig = plt
2792.             return(fig)
2793.
2794.         def bar(xdata,ydata,x_title,y_title):
2795.             ind = np.arange(5)
2796.             width=0.3
2797.             plt.bar(ind,ydata, align='center',width=width,color='b')
2798.             plt.xticks(range(len(xdata)),xdata)
2799.             plt.xlabel(x_title)
2800.             plt.ylabel(y_title)
2801.             return(fig)
2802.
2803.
2804.         def save_plot(fig,name):
2805.             Dir=os.path.dirname(os.path.abspath(__file__))
2806.             #go to parent directory
2807.             parentDir=os.path.abspath(os.path.join(Dir,".."))
2808.             #go to plots
2809.             pathX=os.path.abspath(os.path.join(parentDir,"plots"))
2810.             print(pathX)
2811.             plot_name=pathX+"/"+name
2812.             fig.savefig(plot_name)
2813.
2814.
2815.         '''''
2816.         def read_x(column_name,x_range_top,x_range_bottom):
2817.             x_array.clear()
2818.             for i in range(x_range_top,x_range_bottom+1):
2819.                 column=column_name+str(i)
2820.                 x=ws[column].value
2821.                 x_array.append(x)
2822.                 #print(column)
2823.             x_title=ws[(column_name+str(1))].value
2824.             print("x axis=",x_title)
2825.             return(x_array,x_title)
2826.
2827.         def read_y(column_name,y_range_top,y_range_bottom):
```

```
            y_array.clear()
            for i in range(y_range_top,y_range_bottom+1):
                column=column_name+str(i)
                y=ws[column].value
                y_array.append(y)
                #print(y_array)
            y_title=ws[(column_name+str(1))].value
            print("y axis=",y_title)
            return(y_array,y_title)
        '''


        ##########################################################################################################
        def xAxis(algorithm_C,condition,x_axis):
            x_array.clear()
            for i in range (1,200):                 #write mean_time to exel
                    if (ws.cell(row=i, column=1).value)==algorithm_C and (ws.cell(row=i, column=2).value
    )==condition:
                        for j in range(1,20):
                            if (ws.cell(row=1, column=j).value)==x_axis:
                                x_array.append(ws.cell(row=i, column=j).value)
                                #print(x_array)
            return(x_array)


        def yAxis(algorithm_C,condition,y_axis):
            y_array.clear()
            for i in range (1,200):                 #write mean_time to exel
                    if (ws.cell(row=i, column=1).value)==algorithm_C and (ws.cell(row=i, column=2).value
    )==condition:
                        for j in range(1,20):
                            if (ws.cell(row=1, column=j).value)==y_axis:
                                y_array.append(ws.cell(row=i, column=j).value)
                                #print(y_array)
            return(y_array)


        def xAxis_resault(algorithm_C,condition,x_axis):
            x_array.clear()

            for i in range (1,200):                 #write mean_time to exel
                    if (ws_resault.cell(row=i, column=1).value)==condition:# and (ws_resault.cell(row=i,
     column=2).value)==algorithm_C:
                        for j in range(1,20):
                            if (ws_resault.cell(row=1, column=j).value)==x_axis:
                                x_array.append(ws_resault.cell(row=i, column=j).value)
                                #print(x_array)

            return(x_array)

        def yAxis_resault(algorithm_C,condition,y_axis):
            y_array.clear()

            for i in range (1,200):                 #write mean_time to exel
                    if (ws_resault.cell(row=i, column=1).value)==condition:# and (ws_resault.cell(row=i,
     column=2).value)==algorithm_C:
                        for j in range(1,20):
                            if (ws_resault.cell(row=1, column=j).value)==y_axis:
                                y_array.append(ws_resault.cell(row=i, column=j).value)
                                print(y_array)

            return(y_array)

        def zAxis_resault(algorithm_C,condition,z_axis):
            z_array.clear()
```

```
2889.
2890.             for i in range (1,200):                #write mean_time to exel
2891.                     if (ws_resault.cell(row=i, column=1).value)==condition:# and (ws_resault.cell(row=i,
      column=2).value)==algorithm_C:
2892.                         for j in range(1,20):
2893.                             if (ws_resault.cell(row=1, column=j).value)==z_axis:
2894.                                 z_array.append(ws_resault.cell(row=i, column=j).value)
2895.                                 print(z_array)
2896.
2897.             return(z_array)
2898.         ##########################################################################################################
      ####
2899.         algorithm_C=['SIFT','SURF','ORB','FAST','BRISK']
2900.         condition='rotation'                    #change every time (rotation,scale,skew)
2901.         title_for_condition=['rotation','scale','skew']
2902.
2903.
2904.
2905.
2906.         _titlex=''
2907.         if condition=='rotation':
2908.             _titlex='Angle(Degree)'
2909.         elif condition=='scale':
2910.             _titlex='Scale(%)'
2911.         elif condition=='skew':
2912.             _titlex='Ratio(%)'
2913.
2914.         '''''
2915.         ### SIFT rotation time accuracy
2916.         x,x_title=read_x('C',i,i+number_of_images-1)
2917.         y,y_title=read_y('H',i,i+number_of_images-1)
2918.         print("x=",x)
2919.         print("y=",y)
2920.         fig=plot(x,y,'Angle(Degree)','Accuracy%')
2921.         fig.pause(2)
2922.         #fig.show()
2923.         img_name=algorithm_C+"_"+condition+"_"+x_title+"_"+y_title
2924.         save_plot(fig,img_name)
2925.         '''
2926.
2927.
2928.
2929.         #>>>>>>>> 'accuracy' & 'value'
2930.         for al in algorithm_C:
2931.             to_compare_x='value'
2932.             to_compare_y='accuracy'
2933.             x=xAxis(al,condition,to_compare_x)
2934.             y=yAxis(al,condition,to_compare_y)
2935.
2936.             style = dict(size=10, color='gray')
2937.             fig=plot(x,y,_titlex,'Acuuracy(%)')
2938.             fig.title(condition)
2939.             fig.legend(algorithm_C,ncol=5,bbox_to_anchor=[0.5, 1.145],loc='upper center', fancybox=True,
      shadow=True)
2940.             #fig.annotate('local max', xy=(0, 70), xytext=(0, 70))
2941.             '''''
2942.             print(x)
2943.             print(y)
2944.             print(al)
2945.             '''
2946.         if condition=='scale':
2947.             ax = fig.gca()
2948.             ax.invert_xaxis()
2949.
2950.
2951.         img_name=str(condition+"_"+to_compare_x+"_"+to_compare_y)
```

```
        save_plot(fig,img_name)
        fig.pause(1)
        
        
        
        
        #>>>>>>>> 'good matches' & 'value'
        fig.close()
        for al in algorithm_C:
        
            to_compare_x='value'
            to_compare_y='good matches'
            x=xAxis(al,condition,to_compare_x)
            y=yAxis(al,condition,to_compare_y)
        
            style = dict(size=10, color='gray')
            fig=plot(x,y,_titlex,'Good Matches')
            fig.title(condition)
            fig.legend(algorithm_C,ncol=5,bbox_to_anchor=[0.5, 1.145],loc='upper center', fancybox=True, shadow=True)
            #fig.annotate('local max', xy=(0, 70), xytext=(0, 70))
        
            #print(x)
            #print(y)
            #print(al)
        
        if condition=='scale':
            ax = fig.gca()
            ax.invert_xaxis()
        
        
        img_name=str(condition+"_"+to_compare_x+"_"+to_compare_y)
        save_plot(fig,img_name)
        fig.pause(1)
        
        
        
        
        
        
        ###################################################### get plot from resault.xlsx
        
        condition_array=['rotation','scale','skew']
        
        #>>>>>>>>>>>>>>>>>>>>>>>>>> name= scale_algorithm to good matches.png
        
        for i in condition_array:
            fig.close()
            to_compare_x='algorithm'    #algorithm column
            to_compare_y='good matches' #good matches column
            x=xAxis_resault(algorithm_C[0],i,to_compare_x)
            y=yAxis_resault(algorithm_C[0],i,to_compare_y)
        
        
            fig2=bar(x,y,'Algorithm','Good Matches')
            fig2=plt
            fig2.title(i)
            #fig2.pause(1)
            img_name=str(i+"_"+to_compare_x+" to "+to_compare_y)
            save_plot(fig2,img_name)
            #fig.pause(0.1)
            #print(x)
            #print(y)
        
        #>>>>>>>>>>>>>>>>>>>>>>>>>> name= scale_algorithm to time.png
        for i in condition_array:
            fig.close()
```

```
3017.             to_compare_x='algorithm'    #algorithm column
3018.             to_compare_y='time' #good matches column
3019.             x=xAxis_resault(algorithm_C[0],i,to_compare_x)
3020.             y=yAxis_resault(algorithm_C[0],i,to_compare_y)
3021.             fig2=bar(x,y,'Algorithm','Time(Sec)')
3022.             fig2=plt
3023.             fig2.title(i)
3024.             #fig2.pause(1)
3025.             img_name=str(i+"_"+to_compare_x+" to "+to_compare_y)
3026.             save_plot(fig2,img_name)
3027.             #fig.pause(0.1)
3028.             #print(x)
3029.             #print(y)
3030.
3031.         #>>>>>>>>>>>>>>>>>>>>>>>>>> name= scale_algorithm to accuracy.png
3032.         for i in condition_array:
3033.             fig.close()
3034.             to_compare_x='algorithm'    #algorithm column
3035.             to_compare_y='accuracy' #good matches column
3036.             x=xAxis_resault(algorithm_C[0],i,to_compare_x)
3037.             y=yAxis_resault(algorithm_C[0],i,to_compare_y)
3038.
3039.             fig2=bar(x,y,'Algorithm','Accuracy(%)')
3040.             fig2=plt
3041.             fig2.title(i)
3042.             #fig2.pause(1)
3043.             img_name=str(i+"_"+to_compare_x+" to "+to_compare_y)
3044.             save_plot(fig2,img_name)
3045.
3046.             #print(x)
3047.             #print(y)
3048.
3049.
3050.
3051.         #>>>>>>>>>>>>>>>>>>>>>>>>>> Algorithm to (time/accurancy) >>>>>>>>name= scale_algorithm to time
     on accuracy.png
3052.         for i in condition_array:
3053.
3054.             fig.close()
3055.             to_compare_x='algorithm'    #algorithm column
3056.             to_compare_y='accuracy' #good matches column
3057.             to_compare_z='time'
3058.             x=xAxis_resault(algorithm_C[0],i,to_compare_x)
3059.             y=yAxis_resault(algorithm_C[0],i,to_compare_y)
3060.             z=zAxis_resault(algorithm_C[0],i,to_compare_z)
3061.             result = []
3062.
3063.             # get last index for the lists for iteration
3064.
3065.             for k in range(len(z)):             #taghsime tamame araye'haye z bar y
3066.                 result.append(z[k]/y[k])
3067.             print('r=',result)
3068.
3069.             fig2=bar(x,result,'Algorithm','Time/Accuracy')
3070.             fig2=plt
3071.             fig2.title(i)
3072.             #fig2.pause(1)
3073.             img_name=str(i+"_"+to_compare_x+" to "+to_compare_z+" on "+to_compare_y)
3074.             save_plot(fig2,img_name)
3075.             result.clear()
3076.             #print('x=',x)
3077.             #print('y=',y)
3078.             #print('z=',z)
3079.
3080.
3081.
```

```
3082.
3083.
3084.
3085.
3086.         #fig2.legend(algorithm_C,ncol=5,bbox_to_anchor=[0.5, 1.145],loc='upper center', fancybox=True, s
     hadow=True)
3087.
3088.         #fig2.show()
3089.         #fig2.pause(10)
```