به نام خدا

دانشگاه صنعتی امیرکبیر

دانشکده مهندسی کامپیوتر

اصول علم ربات

تمرین سری چهارم بخش تئوری

نام و نام خانوادگی	مهدی رحمانی
شماره دانشجویی	۹۷۳۱۷۰۱
تاریخ ارسال گزارش	۱۴۰۲/۰۴/۰۳

فهرست گزارش سوالات

سوال ۱ – الگوریتم‌های Bug

الگوریتم های Bug1 و Bug2 را توضیح دهید و بگویید هرکدام برای چه موقعیت‌هایی مناسب‌تر هستند.

این الگوریتم‌ها در حقیقت برای یافتن مسیری از نقطه مبدا به مقصد هستند و از حشرات الهام گرفته شده اند. این دو الگوریتم نیازی به Map ندارند و با Local decision و داشتن دانش محلی و هدف نهایی پیش میروند چون از محیط اطلاعات چندان و جامعی ندارند و این برخلاف بسیاری از الگوریتم‌های planning هست که از دانش global استفاده میکنند. هردو الگوریتم‌های Completeای هستند و چنانچه مسیر مناسبی وجود داشته باشد، آن را پیدا میکنند و در غیر این صورت failure برمیگردانند. همچنین لزوما بهینه نیستند و کوتاه ترین مسیر را برنمیگردانند. چنانچه محیط ما شلوغ باشد و بسته به شکل موانع و تصمیم گیری چپ یا راست رفتن میتواند اوضاع پیدا کردن مسیر سخت باشد.

رفتار کلی این الگوریتم‌ها ساده است،

۱) دنبال کننده دیوار هستند (حال میتوانند دنبال کننده راست یا چپ باشند)
۲) در یک خط مستقیم به سمت هدف حرکت میکنند.

این الگوریتم‌ها به صورت Tactile sensing اجرا میشوند همچنین مفروضات زیر را داریم:

جهت‌گیری به سمت هدف مشخص می‌باشد.

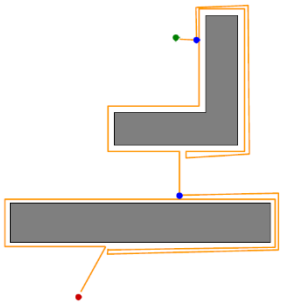
ربات میتواند، فاصله‌اش بین نقاط x و y را اندازه بگیرد د رغیر این صورت از local sensing استفاده میکند.

جهان و محیطی که ربات در آن هست هم باید منطقی باشد یعنی:

۱- تعداد موانع در محیط finite و محدود باشد (که complete باشند).
۲- یک خط باید مانع را به تعداد محدودی بار قطع کند.
۳- همچنین workspace باید محدود باشد.

حال در ادامه به هریک از الگوریتم‌ها به صورت جداگانه میپردازیم:

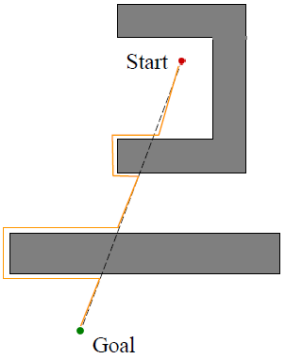
- **Bug1** :
  - ✓ هدینگ ربات به سمت هدف قرار میگیرد و در آن راستا حرکت میکند.
  - ✓ اگر به یک مانع رسیدیم دور آن مانع میچرخیم و از یک حافظه استفاده میکنیم و اینکه در حین این گردش به دور مانع، چه مقدار به هدف نزدیک شدیم را نگه میداریم.
  - ✓ حال باتوجه به مقادیر ثبت شده در حافظه، به کمک دنبال کردن دیوار دوباره به نزدیک ترین نقطه به هدف برمیگردیم.



همچنین اگر بخواهیم آن را از نظر مسافتی که ربات طی میکند، آنالیز کنیم :

$$\begin{cases} D = \text{فاصله روی خط مستقیم از نقطه استارت تا هدف} \\ P_i = \text{محیط مانع ام} i \end{cases} \rightarrow \begin{cases} Lower\ bound:\ D \\ Upper\ bound: D + 1.5 \sum_i P_i \end{cases}$$

- **Bug2**:
  - ✓ ربات میچرخد تا هدینگ ربات روبروی هدف روی m-line قرار گیرد.
  - ✓ اگر در مسیر مانعی بود، دور آن میچرخید تا دوباره به m-line برسید و به شرطی خط m-line را دنبال میکنید که در این حالت به این خط رسیدید نسبت به قبل به هدف نزدیکتر شده باشید.
  - ✓ وقتی با شرایط گفته شده به m-line رسیدید، روی خط m-line به سمت هدف حرکت میکنید.



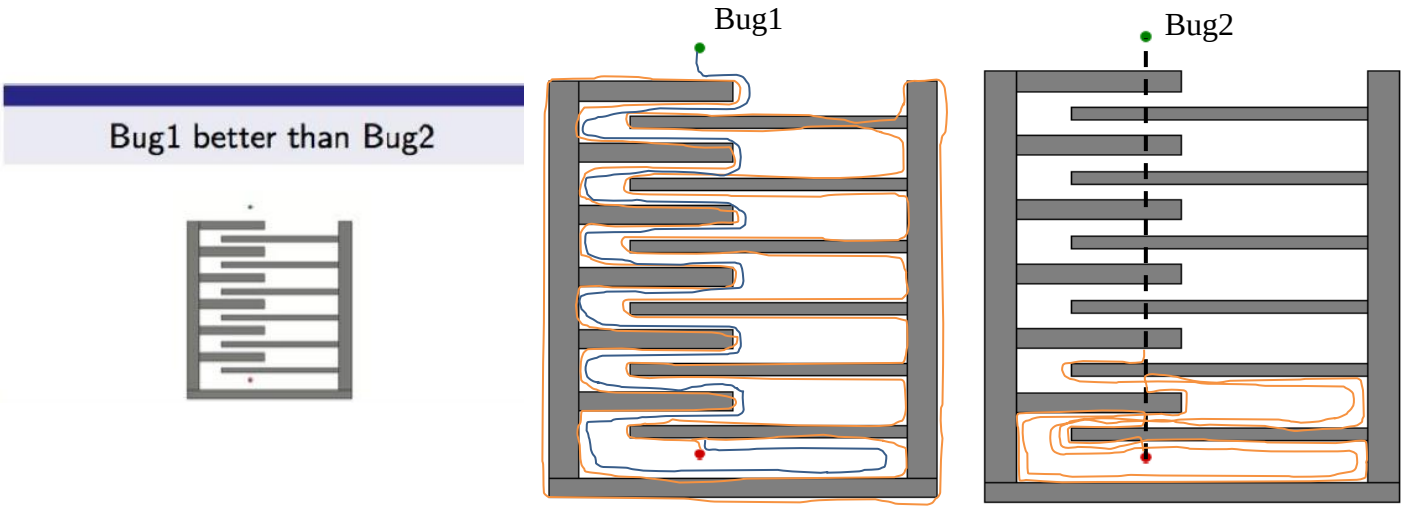
حال اگر بخواهیم Bug2 را از نظر مسافتی که ربات طی میکند، آنالیز کنیم :

$$\begin{cases} D = \text{روی خط مستقیم از نقطه استارت تا هدف} \\ P_i = \text{محیط مانع } i\text{ام} \\ n_i = \ number\ of\ m-line\ intersections\ of\ \ ith\ obstacle \end{cases} \rightarrow \begin{cases} Lower\ bound:\ D \\ Upper\ bound: D + \sum_i \frac{n_i}{2} P_i \end{cases}$$

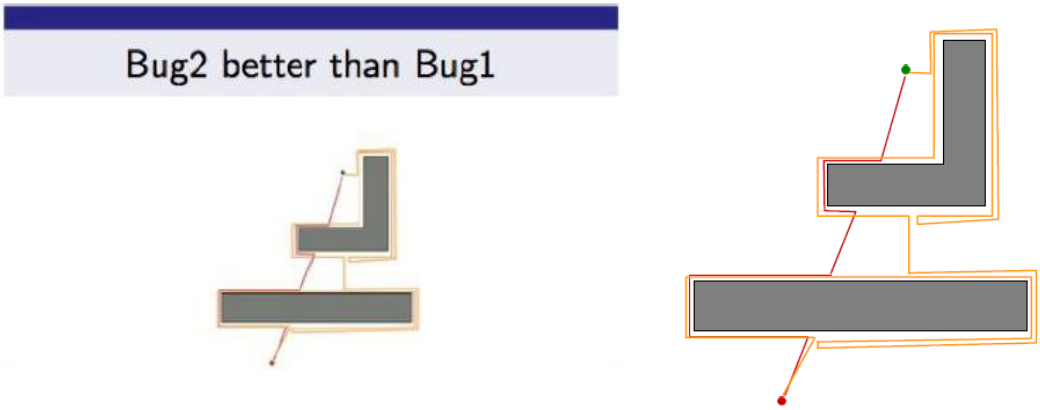
موقعیت مناسب برای به کارگیری:

نمیتوان گفت که Bug2 لزوما از Bug1 بهتر است یا خیر و بستگی به کاربرد دارد. همانطور که در اسلایدهای درس هم اشاره شده است Bug1 یک الگوریتم سرچ کامل است و همه حالت‌ها را بررسی میکند. ولی الگوریتم Bug2 یک الگوریتم حریصانه است که اولین چیزی که به نظر مطلوب بیاید را انتخاب میکند. در خیلی موارد Bug2 بهتر از Bug1 عمل میکند ولی به صورت کلی Bug1 کارایی قابل پیشبینی تری دارد.

- در مواردی که مانع ما پیچیده است Bug1 بهتر است مثل شکل زیر. اگر دقت شود در شکل راست که با باگ۲ رفتیم هر یک دندانه‌ای که جلو میرویم دوباره برمیگردیم و کل مسیر را جاروب میکنیم ولی در شکل سمت چپ آن با باگ۱ رفتیم و کل شکل را یک دور طی کرده و سپس نصف آن را برمیگردد که از همان نزدیکترین نقطه به هدف که پیدا کرده بود خارج شود که در مجموع مسیر کمتری میشود.



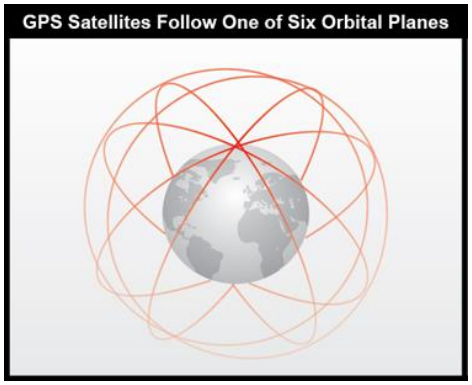
در موانعی که مانع ما ساده‌تر است بهتر است از همان Bug2 استفاده کرد و به صورت حریصانه میتوان نتیجه مطلوب تری گرفت. در شکل سمت راست به طور واضحتری هردو الگوریتم نشان داده شده که خط قرمز برای Bug2 و نارنجی برای Bug1 میباشد. همانطور که دیده میشود با الگوریتم Bug2 مسیر بهینه‌تری رفته ایم.



سوال ۲ – نحوه مکان یابی با GPS

نحوه مکان یابی یک جسم با استفاده از سنسورGPS را به صورت دقیق شرح دهید.

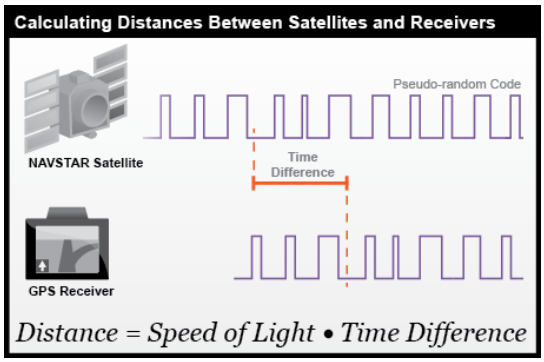
میدانیم که ۲۴ تا ماهواره در ارتافع ۲۰۱۹۰ کیلومتری در طول هر ۱۲ ساعت یک دور به دور زمین میچرخند. در هر ۶ تا orbit به تعداد ۴ تا ماهواره هستند که با زاویه ۶۰ درجه نسبت به هم قرار دارند. برای تعیین مکان یک جسم ما نیاز داریم که حداقل ۴ تا ماهواره در دید گیرنده GPS قرار گیرد. پس درواقع این صفحات اوربیتالی به نحوی قرار گرفتند که در هر زمانی در هر نقطه‌ای از زمین حداقل ۴ ماهواره در دید باشد. مکان هر گیرنده GPS به کمک اندازه گیری Time Of Flight مشخص میشود به این صورت ماهواره‌ها مکان اوربیتال خود (orbital location) را به همراه تایم (زمانی که این ارسال انجام شده) میفرستند و گیرنده هم به کمک trilateration و time correction مکان خود را میابد. برای تعیین مکان در روی سطح زمین (یعنی مکان دو بعدی) فقط ۳ ماهواره نیاز است (محل برخورد ۳ کره) و ماهواره چهارم برای اصلاح ارور زمانی موجود در گیرنده GPS است. اما اگر بخواهیم مکان یک جسم در هوا تعیین کنیم و یعنی ارتفاع هم داشته باشد ناگزیریم برای پیدا کردن مکان از ۴ ماهواره استفاده کنیم که درواقع ماهواره چهارم در پیدا کردن ارتفاع و برطرف کردن ارور زمانی ذکر شده کمک میکند.



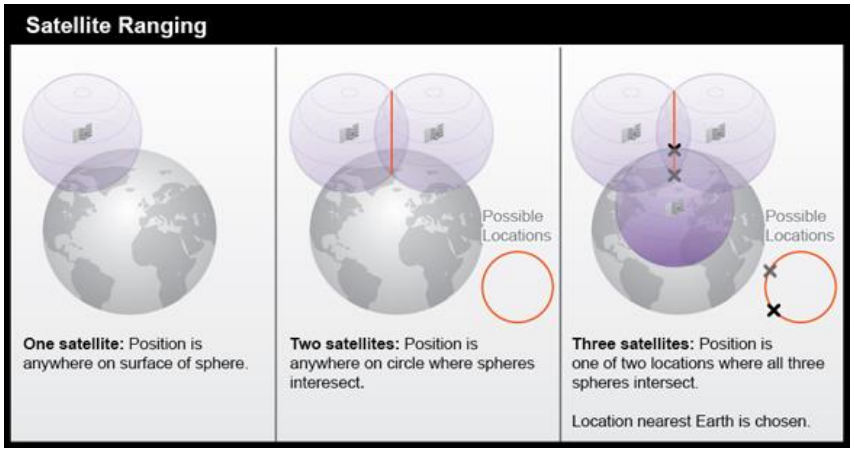
حال در ادامه به صورت دقیقتری توضیحات داده شده است:

هر ماهواره GPS مجهز به یک ساعت اتمی است که زمان را با دقت بسیار بالایی نگه می دارد. به طور مشابه، هر گیرنده GPS یک ساعت نیز دارد (که طبیعتا ساعت اتمی نیست). زمان نگه‌داشته‌شده توسط این ساعت‌ها برای تعیین مدت زمان لازم برای رسیدن سیگنال ماهواره به گیرنده استفاده می‌شود. به طور دقیق‌تر، ماهواره‌های GPS «کدهای شبه تصادفی» را پخش می‌کنند که حاوی اطلاعات زمان و مسیر مداری ماهواره است. سپس گیرنده این کد را تفسیر می کند تا بتواند تفاوت بین ساعت خود و زمان ارسال سیگنال را محاسبه کند. وقتی در سرعت سیگنال (که با سرعت نور حرکت می کند) ضرب شود، می توان

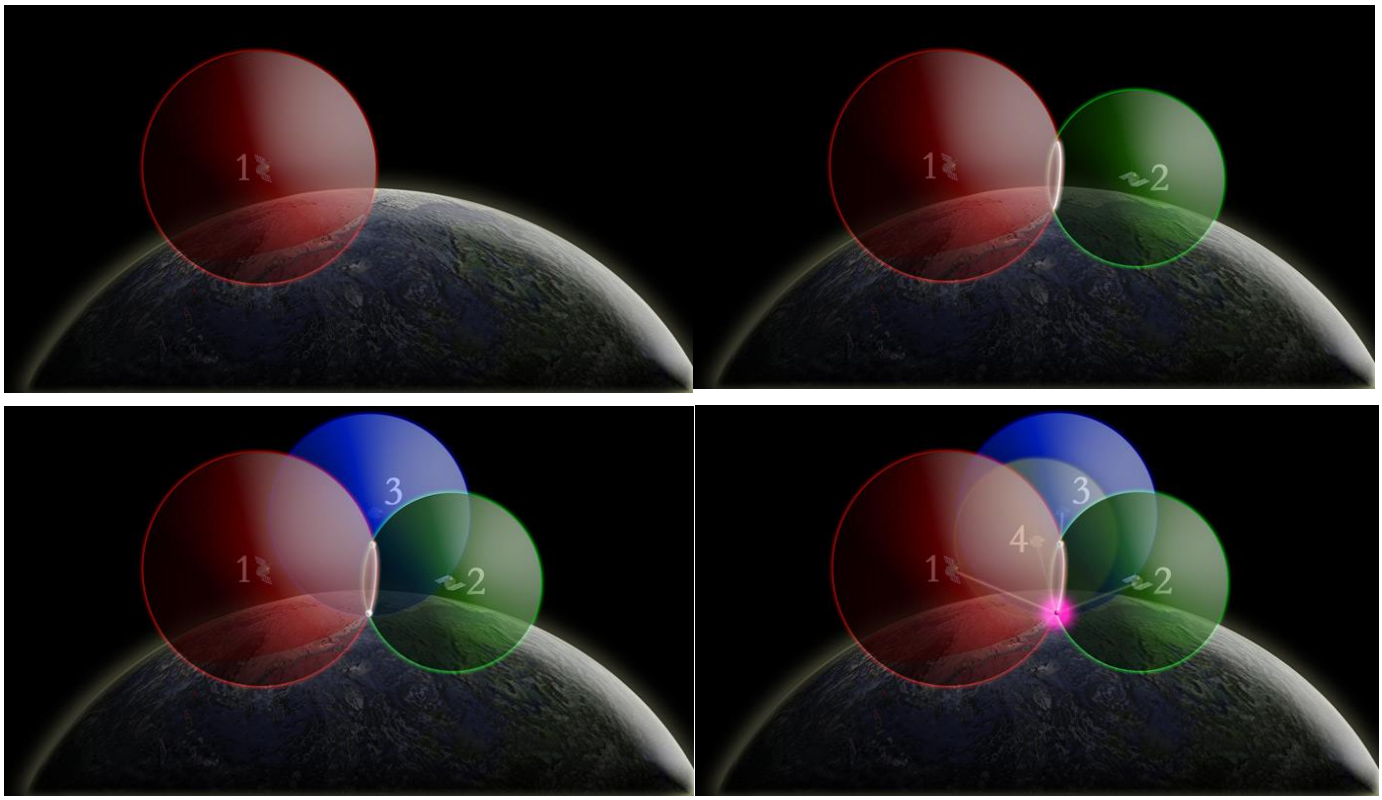
از اختلاف زمان ها برای تعیین فاصله بین ماهواره و گیرنده استفاده کرد که در شکل زیر نشان داده شده است.



همانطور که در بالا توضیح داده شد، مدارهای اوربیتالی GPS به گونه ای پیکربندی شده‌اند که حداقل چهار ماهواره همیشه در همه جای زمین در معرض دید قرار دارند. اگر فقط یک سیگنال ماهواره در دسترس یک گیرنده بود، بهترین کاری که یک گیرنده می توانست انجام دهد استفاده از زمان سیگنال برای تعیین فاصله آن از آن ماهواره است، اما موقعیت گیرنده می تواند، هر یک از تعداد نامحدود نقاطی روی یک کره خیالی با شعاعی برابر با فاصله حساب شده به مرکز آن ماهواره باشد. گر دو ماهواره در دسترس باشد، یک گیرنده می تواند بگوید که موقعیت آن در جایی در در روی دایره‌ای است که از تقاطع دو کره تشکیل شده است. هنگامی که فاصله از سه ماهواره مشخص است، فقط دو نقطه از آن دایره‌ی قبلی معتبر است و موقعیت گیرنده باید یکی از دو نقطه در تقاطع سه کره باشد. گیرنده های GPS معمولا به اندازه کافی هوشمند هستند تا نزدیک ترین مکان به سطح زمین را انتخاب کنند. حداقل، سه ماهواره برای مکانیابی دو بعدی مورد نیاز است. دقت شود که یک ساعت اتمی هماهنگ سنکرون شده با ماهواره‌ی GPS برای محاسبه برد از این سه سیگنالِ رسیده، مورد نیاز است. اگر بخواهیم در هریک از گیرنده‌ها ساعت اتمی بگذاریم که اصلا مقرون به صرفه نیست پس با اندازه گیری از ماهواره چهارم، گیرنده دیگر نیازی به ساعت اتمی ندارد. پس در کل ۴ ماهواره برای تعیین موقعیت و اصلاح زمان لازم است. فرآیند به دست آوردن مکان دو بعدی به کمک ۳ ماهواره در شکل زیر نشان داده شده است.



اما اگر بخواهیم ارتفاع را هم تعیین کنیم یعنی در کل (longitude و latitude و altitude) به صورت دقیقی مشخص کنیم حتما به چهار ماهواره لاز است. در این حالت در حقیقت ماهواره چهارم هم برای تعیین ارتفاع و هم اصلاح ارور زمانی در گیرنده به کار میرود. در شکل زیر میتوانید تعیین مکان ۳ بعدی به کمک ۴ ماهواره را مشاهده کنید.



برای این مطالب از اسلایدهای درس و لینک۱ و لینک۲ و لینک۳ کمک گرفته شده است.

سوال ۳ – سنسورهای Local و Global

دو نمونه از سنسورهای Local و دو نمونه از سنسورهای Global را نام ببرید.

مثال‌های سنسورهای Local:

۱- سنسورهای دمایی: این سنسورها دمای یک مکان یا جسم خاص را اندازه گیری می کنند. آنها معمولاً در ترموستات ها، فرآیندهای صنعتی، ایستگاه های هواشناسی و دستگاه های الکترونیکی مختلف استفاده می شوند.

۲- دوربین: دوربین نیز به عنوان یک سنسور لوکال شناخته میشود و تصاویر را نسبت image frame ثبت میکند و اگر بخواهیم از آن برای فاصله سنجی استفاده کنیم فاصله را نسبت به فریم لوکال خود به ما میدهد و یک فاصله نسبی است و در مقیاسی کوچک (به نسبت GPS)

۳- IMU : این سنسور که از شتاب سنج و ژایروسکوپ تشکیل شده است اندازه گیری‌هایش به صورت لوکال میباشد و در ترکیب با سیستمی که معادلات navigation را پیاده کند و INS را بدهد، اگر برای مکان یابی هم استفاده شود به صورت لوکال و نسبی به ما مکان را میدهد.

مثال‌های سنسورهای Global:

۱- GPS (Global Positioning System) : گیرنده‌های GPS عمدتاً برای ناوبری استفاده می‌شوند، و می‌توانند به عنوان حسگر Global عمل کنند. در حقیقت از نوع Satellite radio beacons هستند. آنها اطلاعات موقعیت یابی و زمان بندی دقیق را با دریافت سیگنال از چندین ماهواره در فضا ارائه می دهند. (توضیحات کامل در سوال قبلی داده شد) در حقیقت میتوان اینگونه در نظر گرفت که مکان absolute یک جسم را در اتمسفر زمین به ما میدهد و به عنوان یک مرجع در نظر گرفت در صورتی که مثلا به کمک IMU و روش Odometry میتوان مکان را به صورت نسبی محاسبه کرد.

۲- رادارهای هوایی: این حسگرها از امواج رادیویی برای تشخیص بارش و اندازه گیری شدت بارش، حرکت ابرها و نوع آن‌ها استفاده می کنند. آنها برای نظارت بر شرایط آب و هوایی در یک منطقه وسیع و ارائه اطلاعات ضروری برای پیش بینی آب و هوا استفاده می شوند.

۳- Ground Radio beacons : مانند UWB یا WiFi anchors . همچنین مثل RFID markers که این موارد در اسلاید درس آمدند.

توضیح بیشتر درباره سنسورهای Global:

حسگرهای Global که به عنوان سنسورهای از راه دور(remote sensors) نیز شناخته می شوند، برای جمع آوری داده ها یا اطلاعات در یک منطقه وسیع یا از راه دور طراحی شده اند. آنها معمولاً برای نظارت بر پدیده های در مقیاس بزرگ یا به دست آوردن یک نمای کلی از یک منطقه یا سیستم خاص استفاده می شوند. حسگرهای Global بسته به کاربرد می توانند ماهواره ای یا زمینی باشند. در حقیقت رفرنس فریمی که در آن اندازه گیری را انجام میدهند یک رفرنس Global است و مقادیر را به صورت absolute اندازه میگیرند و نه به صورت نسبی و نسبت به فریم لوکال خود سنسور.

توضیح بیشتر درباره سنسورهای Local:

حسگرهای local، همانطور که از نام آن پیداست، برای جمع آوری داده ها از یک نقطه یا مکان خاص در یک سیستم یا محیط طراحی شده اند. آنها معمولاً برای نظارت بر شرایط، پارامترها یا رویدادها در یک منطقه Local یا درون یک شی خاص استفاده می شوند.

سوال ۴– ناوبری کور

در ناوبری کور (Dead reckoning) چه سنسوری و چگونه مورد استفاده قرار میگیرد.

در ناوبری کور عموما از سنسورهای Local و Onboard استفاده می‌شود. به این کاری که در غیاب زیرساخت‌های خارجی و سنسورهای Global مثل GPS موقعیت را حساب کنیم و به کمک مثلا سرعت چرخ در طول زمان و محاسبه هدینگ با ژایرو، بتوانیم Pos ربات را دنبال کنیم بهش Dead Reckoning گویند. در این روش تعیین Pos با این سنسورها موجب می‌شود انباشت خطا داشته باشیم و لذا به کمک سنسورهایی مانند GPS این خطا را مرتب اصلاح میکنند. در حقیقت GPS فرکانس ۱ هرتز دارد که اگر برای مثال خودرو با سرعت ۱۰۰ کیلومتر بر ساعت برود یعنی توی ۱ ثانیه ۳۰ متر میرود و خب اگر صرفا بخواهیم از GPS با این فرکانس کم استفاده کنیم در طول این ۳۰ متر مکان خودرو را نمیدانیم و در خیلی از کاربردها ممکن است برای ما دردسر ساز شود. پس به کمک سنسورهای نام برده که غالبا فرکانس بیشتری دارند مثلا به کمک یک IMU با فرکانس ۵۰۰ هرتز و یک ژایرو با فرکانس ۱۰۰ هرتز این مکان یابی را دائما انجام میدهند و هر ۱ ثانیه ۱ بار خطای آن را اصلاح میکنند. برای این کار و درواقع sensor fusion هم روش‌های متعددی وجود دارد و مثلا میتوان از Kalman Filter استفاده کرد.

طبق اسلایدها سنسورهای مورد استفاده در ناوبری کور:

- **سنسورهای Odometry** : مانند Motor Encoderها که برای اندازه گیری چرخش چرخ‌ها، روتورها، helice(مارپیچ) ها و ... استفاده میشوند.
- **سنسورهای Inertial**: که در حقیقت نیروها و اثرات non-inertial را اندازه میگیرند مانند ژایروسکوپ و شتاب سنج
- **سنسورهای Heading و Orientation** : مانند قطب نما یا شیب سنج(Inclinometer) که به کمک آها میتوان فهمید ربات چه چرخش و Orientationای دارد.

در ادامه توضیحاتی درباره این سنسورها آمده است:

IMU : چندین سنسور اینرسی اغلب با هم جمع می شوند تا یک واحد اندازه گیری اینرسی را تشکیل دهند. معمولا از ۳ تا شتاب‌سنج عمود برهم و ۳ تا ژایروسکوپ عمود برهم تشکیل شده است و همچنین قطب نما هم میتواند داشته باشد. در کابردهای Odometry به شدت استفاده میشود.

شتاب سنج هم به صورت خیلی ساده یک سیستم جرم و فنر و دمپر میباشد و به کمک قانون نیوتن و نوشتن معادله نیرو و اینکه در نقاطه تعادل $\ddot{x} = 0$ است میتوان گفت $F = \frac{kx}{m}$ و نیرو را حساب کرد. برای

محاسبه نیرو در ۳ جهت باید ۳ تا شتاب سنج عمود برهم بگذاریم. و معمولا فرکانس نهایتا ۵۰۰ هرتز در نوع مکانیکی و حتی بالاتر از 100kHz در نوع پیزو الکتریک خود دارند.

ژایروسکوپ هم به کمک اثر ژیروسکوپی میتوان سرعت زاویه‌ای را حول spin خود محاسبه کند. در حقیقت ژیروسکوپ ها حسگرهایی هستند که جهت گیری را در رابطه با یک reference frame ثابت حفظ می‌کنند و امکان اندازه گیری سرعت زاویه ای ω را نسبت به فضای اینرسی فراهم می کنند. نوع Optical هم دارد که باتوجه به سرعتی که نور دارد و میتواند یک مسیر را در یک تایمی طی کند اختلاف آن با حالتی که جسم بچرخد و این زمان تغییر کند، سرعت زاویه‌ای تعیین میشود.

قطب نما در حقیقت به صورت مستقیم میدان مغناطیسی را اندازه گیری و دنبال میکند و چون در زمین میدان مغناطیسی داریم میتوان به کمک آن جهت گیری را مشخص کرد. قطب نما چون به میدان های مغناطیسی اطراف حساسیت دارد و در زمین هم میدان مغناطیسی نسبتا ضعیف میباشد، در صورت وجود یک میدان مغناطیسی قوی میتوانند مختل شوند. همچنین قطب نما فرکانس و سرعت کمی دارد و دواقع فرکانس 0.5 HZ دارد که در سرعت‌های بالا نامناسب میشود و کلا هم برای کاربدهای indoor برای تعیین جهت گیری به صورت absolute مناسب نیست ولی تعیین جهت گیری به صورت locally را میتوان هندل کرد.

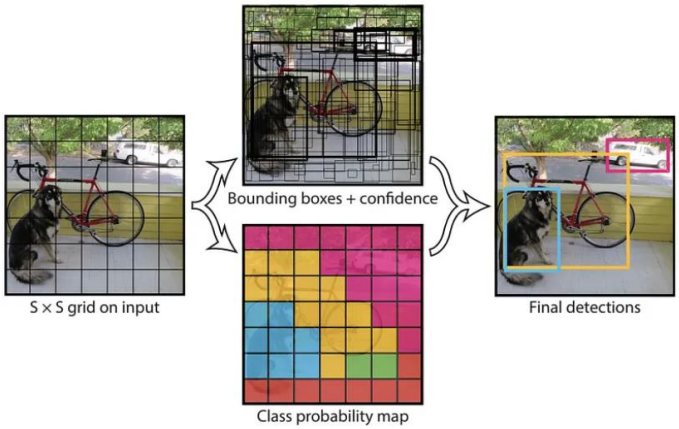
در تشخیص هدینگ، شیب سنج نیز به کار گرفته شده و شیب سنج این قابلیت را دارد که در هر دو جهت یعنی بالا و پایین، شیب ها را اندازه گیری کند. در این دستگاه یک صفحه ی توخالی قرار دارد که نیمی از آن با مایعی سنگین و در یک حباب پر شده است. صفحه ی شیشه ای برای این که زاویه ی سطح مایع را نسبت به صفحه ی ثابت نشان دهد، مدرج شده است در واقع با اندازه گیری محل قرار گیری مایع در حالت های مختلف این شیب سنج درجه بندی می شود، جهت مایع هم نمایانگر جهت شیب است. مایع درون حباب هدینگ را نشان می دهد، با توجه با این که چگونه و به کدام سمت است

سوال ۵– مدل YOLO (امتیازی)

در مورد مدل YOLO تحقیق کنید و خلاصه از نحوه کارکرد آن بنویسید. سپس ورژن های مختلف آن را از لحاظ سرعت و دقت عملکرد با هم مقایسه کنید.

در درس با الگوریتم‌های تشخیص Object مثل R-CNN یا Fast R-CNN یا Faster R-CNN آشنا شدیم. این الگوریتم‌ها با اینکه یکی پس از دیگری بهبود پیدا کردند ولی برای کاربردهای بلادرنگ و سیستم‌هایی که از نظر سخت افزاری خیلی قوی نیستند مناسب نمیباشند و همگی از یک شبکه عصبی یا روش مجزا برای تشخیص Region Proposalها و یا به از روش‌های selective search برای یافتن region proposalها استفاده میکنند و درنهایت هم ممکن است به یک لایه fully connected داده شوند. به هرصورت پردازش‌های سنگینی دارند.

YOLO یا You Only Look Once بر اساس اصل "شما فقط یک بار به یک تصویر نگاه می کنید" عمل می کند؛ به این معنی که یک تصویر را در عبور از یک شبکه عصبی عمیق پردازش می کند. این برخلاف سیستم های سنتی تشخیص اشیا است که از دو شبکه مجزا برای تشخیص و طبقه بندی اشیا استفاده می‌کنند. این رویکرد نوآورانه به YOLO اجازه می دهد تا اشیا را در زمان واقعی با زمان پردازش تنها چند میلی ثانیه در هر فریم شناسایی کند. درواقع به کمک یک شبکه عصبی CNN میتواند هم Bounding boxها و احتمالات هریک را پیش بینی کند.



یولو برای اولین بار در سال ۲۰۱۵ توسط جوزف ردمون معرفی شد و از آن زمان به یکی از پرکاربردترین سیستم های تشخیص اشیا در دانشگاه و صنعت تبدیل شده است که با میانگین دقت متوسط (mAP) بیش از ۷۰٪ در مجموعه داده های معیار استاندارد، از دقت بالایی برخوردار است.

نحوه کارکرد آن در ادامه آمده است:

درواقع ما به عنوان ورودی یک عکس به آن میدهیم و آن را به یک گرید S*S تقسیم میکند و درون هر یک از گریدها ما m تا bounding box در نظر میگیریم. برای هریک از boundingboxها، شبکه عصبی مورد نظر به ما یک احتمال کلاس (اینکه با چه احتمالی عضو کدام کلاس است) و همچنین تعدادی Offset values برای bounding box میدهد. اون boundingboxهایی که class probability بالاتر از یک مقدار threshold را داشته باشند، انتخاب میشوند و برای مشخص کردن Object در تصویر استفاده میشوند.

مراحل به صورت زیر است:

- پیش پردازش: اندازه تصویر ورودی به اندازه ثابت تغییر می کند و نرمال می شود.
- لایه های کانولوشن: تصویر از یک سری لایه های کانولوشن عبور داده می شود تا ویژگی هایی از تصویر استخراج شود.
- لایه های تشخیص: نقشه ویژگی تولید شده توسط لایه های کانولوشن از لایه های تشخیص عبور داده می شود تا حضور اشیا در تصویر را پیش بینی کند.
- حداکثر عدم انتشار (Non-Maximal Suppression): برای از بین بردن همپوشانی جعبه‌های مرزی و کاهش تشخیص‌های مثبت کاذب، YOLO سرکوب غیر حداکثری را برای مجموعه جعبه‌های مرزی تولید شده توسط لایه‌های تشخیص اعمال می‌کند.
- خروجی: خروجی نهایی YOLO مجموعه ای از جعبه های محدود کننده با احتمالات کلاس برای هر جعبه است.

YOLO نسبت به سایر الگوریتم‌های تشخیص اشیا، سریع‌تر است (۴۵ فریم در ثانیه). در کنار تمامی ویژگی های مثبت و قابلیت های بالا YOLO میتوان به محدودیت اندازه ورودی تصاویر اشاره کرد. (که البته میتوان با ابزار های دیگری، این مشکل را برطرف نمود) همچنین در مواردی که اشیا کوچک هستند، امکان ارزیابی اشتباه وجود دارد و مثلا موقعی که دسته‌ای از پرنده ها که در تصویر هریک کوچک هستند را بخواهد مشخص کند به مشکلاتی میخورد. توان محاسباتی بالای موردنیاز یولو، مانع از استفاده از آن برروی گوشی های هوشمند یا دستگاه های تعبیه شده ارزان قیمت میشود.

مقایسه ورژن‌های مختلف:

برای YOLO ورژن‌های متعددی آمده است که در هریک سعی شده است ایرادات موجود در قبلی‌ها را بپوشاند. اکنون ما YOLO ورژن ۸ را هم داریم. در ادامه به صورت مختصر به مقایسه آن‌ها پرداختیم.

- YOLOv1:

YOLOv1 نسخه اصلی YOLO بود که در سال ۲۰۱۵ معرفی شد. با تقسیم تصویر ورودی به یک شبکه و پیش‌بینی جعبه‌های محدود و احتمالات کلاس برای هر سلول شبکه، به تشخیص شی بلادرنگ دست یافت.

YOLOv1 نسبت به نسخه‌های بعدی mAP (میانگین دقت متوسط) نسبتاً پایینی داشت.

سرعت: سریع (۴۵ فریم در ثانیه در پردازنده گرافیکی Titan X).

دقت: متوسط

- YOLOv2 (YOLO9000):

YOLOv2 که با نام YOLO9000 نیز شناخته می شود، در سال ۲۰۱۶ منتشر شد و برخی از محدودیت های YOLOv1 را برطرف کرد. anchor boxها را برای کنترل بهتر اجسام با اندازه ها و نسبت های مختلف معرفی کرد. YOLOv2 از معماری "Darknet-19" با ۱۹ لایه کانولوشن استفاده کرد.YOLO9000 همچنین تشخیص تعداد زیادی از دسته بندی اشیاء (حدود ۹۰۰۰) را با آموزش بر روی هر دو مجموعه داده COCO و ImageNet گنجانده است.

سرعت: سریعتر از YOLOv1 (تقریباً ۶۷ فریم در ثانیه در پردازنده گرافیکی Titan X).

دقت: نسبت به YOLOv1 بهبود یافته و به mAP بهتری دست می یابد.

- YOLOv3:

YOLOv3 که در سال ۲۰۱۸ معرفی شد، عملکرد و دقت الگوریتم YOLO را بیشتر بهبود بخشید.

این معماری از معماری "Darknet-53" با ۵۳ لایه کانولوشن استفاده می کند که شبکه ای عمیق تر و قدرتمندتر را ارائه می دهد. YOLOv3 تشخیص چند مقیاسی را پیاده سازی کرد و امکان تشخیص اشیاء در اندازه ها و مقیاس های مختلف را فراهم کرد. YOLOv3 همچنین استفاده از تکنیکی به نام «شبکه هرمی ویژگی (FPN)» را برای استخراج ویژگی‌ها در وضوح‌های مختلف و بهبود دقت گنجانده است.

سرعت: کمی کمتر از YOLOv2 (حدود ۴۵ فریم در ثانیه در پردازنده گرافیکی Titan X).

دقت: نسبت به YOLOv2 بهبود یافته و به mAP بالاتری دست می یابد.

- YOLOv4:

YOLOv4 که در سال ۲۰۲۰ منتشر شد، با هدف جلو بردن مرزهای عملکرد تشخیص اشیا انجام شد.

چندین پیشرفت معماری از جمله ستون فقرات CSPDarknet53 برای عملکرد بهتر و PANet (شبکه تجمع مسیر) برای ترکیب دقیق‌تر ویژگی‌ها معرفی شد. YOLOv4 همچنین تکنیک‌هایی مانند فعال‌سازی Mish، از دست دادن CIOU و الگوریتم EfficientNMS را برای بهبود دقت پیاده‌سازی کرد.

سرعت: مشابه YOLOv3، اگرچه نسخه های بهینه شده ای وجود دارند که ادعا می کنند سرعت بالاتری دارند.

دقت: به طور کلی نسبت به YOLOv3 بهبود یافته و به mAP بالاتری دست می یابد.

- YOLOv5:

YOLOv5 که در سال ۲۰۲۰ منتشر شد، به عنوان یک پیاده سازی مستقل توسط یک گروه تحقیقاتی متفاوت توسعه داده شد. تمرکز آن بر بهبود سرعت و سهولت استفاده در عین حفظ دقت رقابتی بود.

YOLOv5 از یک معماری شبکه متفاوت بر اساس مدل EfficientDet استفاده کرد و استفاده از تشخیص شی بدون لنگر را معرفی کرد.

سرعت: سریعتر از YOLOv4، دستیابی به فریم های بالا در ثانیه در GPU های مدرن.

دقت: قابل رقابت با نسخه های قبلی، دستیابی به mAP قابل مقایسه.

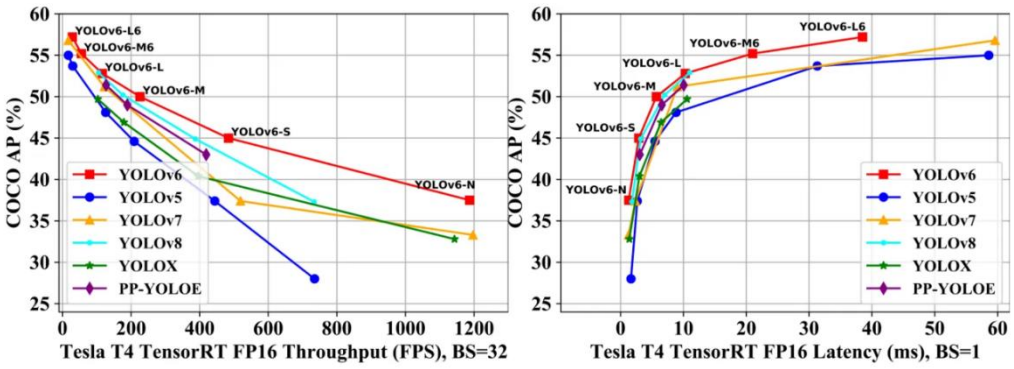
در نهایت نسخه‌های YOLO6 و YOLO7 و YOLO8 نیز آمدند. تیم Meituan عملکرد سرعت همه مدل‌های رسمی را با دقت FP16 روی همان GPU Tesla T4 با TensorRT آزمایش کرد و YOLOv6 ارتقا یافته را با YOLOv5، YOLOX، PPYOLOE، YOLOv7 و YOLOv8 مقایسه کرد.

YOLOv6 v3.0 در مقایسه با YOLOv8 دقت کمی کمتری دارد، اما با سرعت پردازش سریعتر و تأخیر کمتر آن را جبران می کند. این باعث می شود آن را برای برنامه های بلادرنگ که سرعت در آن ها ضروری است مناسب تر باشد. از سوی دیگر، YOLOv8 دقت بالاتری را ارائه می دهد و آن را به انتخاب بهتری برای برنامه هایی تبدیل می کند که دقت در آنها بسیار مهم است.

همچنین نتایج مقایسه نشان می دهد که YOLOv6-N به طور قابل توجهی ۹.۵٪ در مقایسه با YOLOv5-N پیشرفت کرده است، همچنین بهترین عملکرد سرعت را از نظر توان عملیاتی و تأخیر دارد. YOLOv6-S می تواند AP را ۳.۵٪ در مقایسه با YOLOX-S و ۰.۹٪ در مقایسه با PPYOLOE-S با سرعت بالاتر بهبود بخشد. YOLOv6-M با سرعت مشابه ۴.۶٪ از AP بالاتر از YOLOv5-M بهتر عمل می کند و با سرعت بالاتر به AP 3.1٪ و ۱.۰٪ بالاتر از YOLOX-M و PPYOLOE-M دست می یابد.

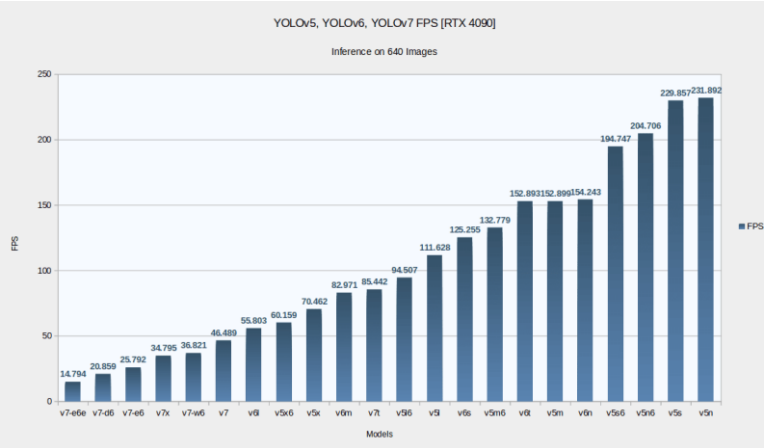
از طرفی YOLOv8 توسط Ultralytics توسعه یافته و جدیدترین و پیشرفته ترین نسخه مدل های YOLO به حساب می آید. YOLOv8 به دلیل سرعت پردازش سریع، دقت بالا و توانایی تشخیص چندین شی در یک تصویر شناخته شده است. همچنین دارای یک شبکه backboneجدید، است که دارای یک loss function جدید و anchore-free detection است که مقایسه عملکرد مدل با مدل های قدیمی تر در خانواده YOLO را آسان می کند.

میتوانید مقایسه آن ها را در ادامه در نمودار ببینید:

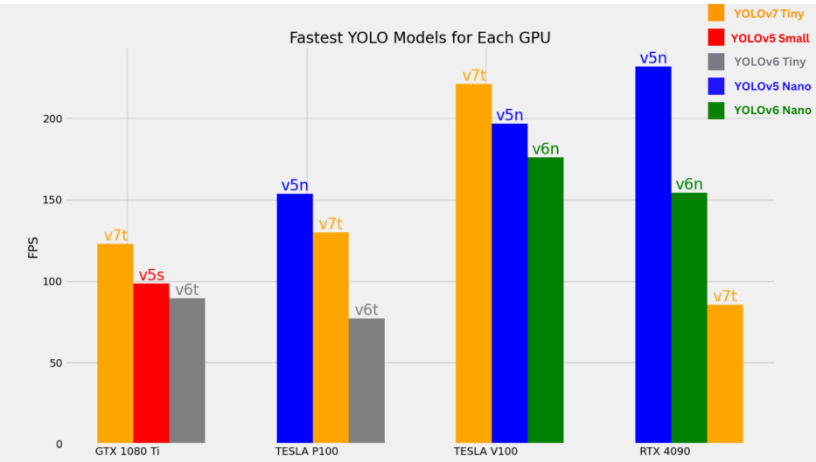


YOLO Comparisons

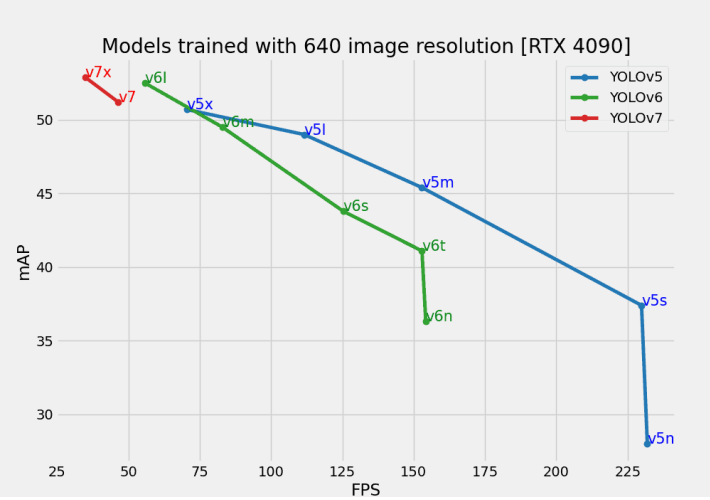
همچنین در سایتی FPS مربوط به YOLOهای مختلف که بیانگر سرعت آن‌هاست را مقایسه کرده بود. این سرعت‌ها برای تست بر روی NVIDIA RTX 4090 GPU میباشند:



و همچنین مقایسه سرعت آن‌ها روی GPUهای مختلف در زیر آمده است:



و همچنین مقایسه mean Average Precision یا همان mAP که بیانگر دقت است را در زیر میبینید:



برای جواب دادن به این بخش از لینک‌های لینک۱ و لینک۲ و لینک۳ و لینک۴ کمک گرفته شد.