
بسم الله الرحمن الرحیم


**پژوهشکده مخابرات و الکترونیک نصر**

# طراحی و پیاده سازی الگوریتم هوشمند خوشه‌بندی در جهت یاب راداری

**نگارش**

سیاوش خجسته

**استاد راهنما:**

**مهندس حسنی**

**استاد مشاور:**

**مهندس ساریجلو**

**پاییز ۱۳۹۱**

# چکیده

نمایش جهت و مکان تقریبی رادارهای یافت شده توسط دستگاه جهت‌یاب راداری بر روی نقشه‌های تولید شده توسط تکنولوژی GIS در کنار لایه‌های مختلف اطلاعاتی به شکل چشمگیری موجب بالا رفتن قابلیت استفاده از این نوع سامانه‌ها می‌شود. در این پروژه نحوه طراحی و پیاده‌سازی الگوریتم خوشه بندی در نرم‌افزار NRDF که نرم‌افزاری برای کار با دستگاه جهت‌یاب راداری می‌باشد شرح داده می‌شود. به علت آنکه میزان اطلاعات دریافتی از جهت‌یاب راداری بسیار زیاد می باشد، می بایست یک الگوریتم کارا و بسیار سریع طراحی و پیاده سازی شود تا با دریافت اطلاعات با سرعت بسیار زیاد بتواند اجسام را تشخیص دهد. برای این منظور الگوریتم هوشمند و بسیار سریع خوشه بندی طراحی شد تا با استفاده از آن بتوان با سرعت بسیار بالا و با استفاده از اطلاعات ورودی، اجسام را تشخیص داد.

**کلید واژه:** خوشه بندی، داده کاوی، GIS، رادار، جهت‌یاب راداری

# فهرست مطالب

**عنوان** **صفحه**

# فصل ۱– مقدمه

## ۱–۱– پیشگفتار

تشخیص وسیله‌های نقلیه جنگی و غیر جنگی با استفاده از علائم راداری ارسال شده از آنها یکی از مهمترین روش‌ها برای نظارت، دیده‌بانی و جلوگیری از پیامدهای غیر مترقبه و از نظر امنیتی مهم است. برای انجام این کار از ابزاری به نام جهت‌یاب راداری استفاده می‌شود. همانطور که دقت سخت‌افزار مورد استفاده به عنوان جهت‌یاب راداری از اهمیت بالایی برخوردار است، طریقه پیاده سازی الگوریتم و اجرای آن و توانایی ذخیره و پردازش اطلاعات به دست آمده، در آینده نیز از اهمیت بالایی برخوردار است و چه بسا بدون وجود چنین ابزارهایی عملاً دقت بالای سخت‌افزار مورد استفاده واقع نمی‌شود.

در پروژه حاضر، نرم‌افزاری برای تعامل و استفاده از جهت‌یاب راداری مورد استفاده پژوهشکده نصر تهیه شده است؛ در تولید این نرم‌افزار سعی شده است که علاوه بر امکانات تخصصی که توسط نرم‌افزار ارائه می‌شود، رابط کاربری تا حد امکان ساده و با استفاده از فن‌آوری‌های روز باشد. یکی از مهمترین قابلیت‌هایی که در این نرم‌افزار قرار داده شده است الگوریتم بسیار سریع خوشه بندی می‌باشد. این الگوریتم به این منظور در این نرم افزار ایجاد شده است تا بتوان با استفاده از آن، اجسام را تشخیص داد. به علت آنکه جهت‌یاب راداری اطلاعات بسیاری را ارسال می کند، می‌بایست الگوریتمی پیاده سازی می شد تا با کمترین زمان بعد از دریافت اطلاعات از جهت‌یاب، بتوان تشخیص دهد جسمی یافت شده است یا خیر. به عبارت دیگر این الگوریتم وظیفه تشخیص اجسام را با سرعت بسیار زیاد دارد.

به علت آنکه این الگوریتم وظیفه تشخیص اجسام را دارد می بایست به صورت هوشمند و با استفاده از تعاریف داده کاوی پیاده سازی شود، علاوه بر آن، برای آنکه اجرای الگوریتم با سرعت بالایی باشد می بایست الگوریتم بسیار بهینه پیاده سازی شده باشد، علاوه بر پیاده سازی بهینه الگوریتم کامپایلر برنامه نیز در سرعت اجرای برنامه تاثیر گذار می باشد. برای پیاده سازی کردن این الگوریتم از کامپایلر netframework ٤.٠. و زبان C# استفاده شده است.

ساختار این مستند به این صورت شکل گرفته است که ابتدا اصول رادار ها و دستگاه های جهت‌یاب راداری معرفی می شوند. سپس GIS به شکل کلی شرح داده می شود. در ادامه نرم افزار

NRDF و نحوه کار و فرم‌های این نرم افزار توضیح داده می شود. در فصل بعد، در مورد تکنولوژی netframework ٤.٠ صحبت شده است و در ادامه تعریفی از داده کاوی و الگوریتم های خوشه بندی ارائه شده است و در انتها به شرح الگوریتم خوشه بندی در جهت یاب راداری پرداخته شده است.

# فصل ۲– خلاصه در مورد رادار بخش های گزارش و ترتیب آنها

## ۲–۱– مقدمه

معمول‌ترین سنسور فعال[۱] که عمل تصویربرداری را انجام می‌دهد رادار می‌باشد. رادار مخفف عبارتRadio Detection and Ranging و به معنای آشکارسازی به کمک امواج مایکرویو است. به طور کلی می‌توان عملکرد رادار را در چگونگی عملکرد سنسورهای آن خلاصه کرد. سنسورها سیگنال‌های مایکرویو را به سمت اهدف مورد نظر ارسال کرده وسپس سیگنال‌های باز تابیده شده از سطوح مختلف را شناسایی می‌کند. قدرت (میزان انرژی) سیگنال‌های پراکنده شده جهت تفکیک اهداف، مورد استفاده قرار می‌گیرد. با اندازه‌گیری فاصه زمانی بین ارسال و دریافت سیگنال‌ها می‌توان فاصله اهداف مشخص کرد. از مزایای ویژه رادار می‌توان به عملکرد رادار در شب یا روز و همچنین قابلیت تصویربرداری درشرایط آب و هوایی مختلف اشاره کرد. امواج مایکرویو قادر به نفوذ در ابر، مه, گرد و غبار و باران می‌باشند. از آنجایی که عملکرد رادار با طرز کار سنسورهایی که با طیف‌های مرئی و مادون قرمز[۲] کار می‌کنند متفاوت است لذا می‌توان با تلفیق اطلاعات بدست آمده از این روش‌ها تصاویر دقیقی را بدست آورد.

## ۲–۲– تاریخچه

نخستین بار در سال ۱۹۰۱ « هوگو ژرنسبارک » که او را «ژول ورن» آمریکایی می‌نامند، در یک داستان علمی _ تخیلی ، آن را طرح ریزی کرد. در سال ۱۹۰٦ ، یک دانشجوی ۲۳ ساله آلمانی ، به نام « هولفس یر » دستگاهی ساخت که با اصول رادارهای امروزی می‌توانست امواجی را بسوی موانع بفرستد و بازتاب آنها را دریافت دارد. آزمایش اساسی ارسال امواج الکترومغناطیسی بسوی هواپیماهای در حال پرواز ، بوسیله یک دانشمند فرانسوی به نام « پیر داوید » انجام یافت. در آغاز جنگ دوم جهانی بود که تکنسینهای انگلیسی موفق شدند، نخستین مدلهای راداری امروزی را بسازند. اما کار آنها یک مشکل اساسی داشت. امواج تا نقطه‌ای که می‌خواستند نمی‌رسید و تنها تا پنج هزار متر برد داشت. به همین دلیل یک فرانسوی دیگر به نام "موریس پونت" در سال ۱۹۳۰

---

[۱] Active Sensor

[۲] Infra Red

موفق به اختراع دستگاهی جالب به نام "مانیترون" شد که امواج بسیار کوتاه رادیویی را بوجود می‌آورد و به همین دلیل رادارهایی که به کمک این وسیله تکمیل شدند توانستند تا دهها کیلومتر بیش از رادار قبلی امواج را ارسال کنند. دستگاه اختراعی پونت در سال ۱۹۳۵ ابتدا در کشتی معروفی به نام نرماندی نصب شد و توانست آن را از خطر برخورد با کوههای عظیم یخی شناور در اقیانوس محافظت کند و بدین ترتیب رادار علاوه بر استفاده وسیع در هوا ، سطح دریاها را هم به تسخیر خود در آورد.

## ۲-۳- چگونگی عملکرد رادارها

امواج رادار چیزی است که در تمام اطراف ما وجود دارد، اگر چه دیده نمی‌شود. مرکز کنترل ترافیک فرودگاهها برای ردیابی هواپیماها چه آنها که بر روی باند فرودگاه قرار دارند و چه آنها که در حال پرواز هستند و هدایت آنها از رادار استفاده می‌کنند. در برخی از کشورها پلیس از رادار برای شناسایی خودروهای با سرعت غیر مجاز استفاده می‌کند. ناسا از رادار برای شناسایی موقعیت کرۀ زمین و دیگر سیارات استفاده می‌کند، همین طور برای دنبال کردن مسیر ماهواره‌ها و فضاپیماها و برای کمک به کشتی‌ها در دریا و مانورهای رزمی از آن استفاده می‌شود. مراکز نظامی نیز برای شناسایی دشمن و یا هدایت جنگ‌افزارهایشان از آن استفاده می‌کنند چیزی است که در تمام اطراف ما وجوددارد، اگر چه دیده نمی‌شود. مرکز کنترل ترافیک فرودگاهها برای ردیابی هواپیماها چهآنها که بر روی باند فرودگاه قرار دارند و چه آنها که در حال پرواز هستند و هدایتآنها از رادار استفاده می‌کنند. در برخی از کشورها پلیس از رادار برای شناساییخودروهای با سرعت غیر مجاز استفاده می‌کند. ناسا از رادار برای شناسایی موقعیت کرۀزمین و دیگر سیارات استفاده می‌کند، همین طور برای دنبال کردن مسیر ماهواره‌ها وفضاپیماها و برای کمک به کشتی‌ها در دریا و مانورهای رزمی از آن استفاده می‌شود. مراکز نظامی نیز برای شناسایی دشمن و یا هدایت جنگ‌افزارهایشان از آن استفادهمی‌کنند.

هواشناسان برای شناسایی طوفانها،تندبادهای دریایی و گردبادها از آن استفاده می‌برند. شما حتی نوعی خاص از رادار رادر مدخل ورودی فروشگاهها می‌بینید که در هنگام قرار گرفتن اشخاص در مقابلشان، دربرا باز می‌کنند. بطور واضح می‌بینید که رادار وسیله‌ای بسیار کاربردی می‌باشد. دراین بخش از مقالات ما به اسرار رادار می‌پردازیم. استفاده از رادار عموماً در راستای زیر می‌باشد:

شناسایی حضور یاعدم حضور یک جسم در فاصله‌ای مشخص – عمدتاً آنچه که شناسایی می‌شود متحرکاست و مانند هواپیما، اما رادار قادر به شناسایی حضور اجسام که مثلاً در زیرزمینیز مدفون شده‌اند، می‌باشد. در بعضی از موارد حتی رادار می‌تواند ماهیت آنچه را کهمی‌یابد مشخص کند، مثلاً نوع هواپیمایی که شناسایی می‌کند. شناسایی سرعت آنجسم- دقیقاً همان هدفی که پلیس از آن در بزرگراه‌ها برای کنترل سرعت خودروهااز آن استفاده می‌کند. جابه‌جاییاجسام – شاتل‌های فضایی و ماهواره‌های دوار بر دور کره زمین از چیزی بهعنوان رادار حفره‌های مجازی برای تهیه نقشه جزئیات، نقشه‌های عوارض جغرافیایی سطحماه و دیگر سیارات استفاده می‌کنند. تمام این سه عملیات می‌تواند با دوپدیده‌ای که شما در زندگی روزمره با آن آشنائید پیاده شود: «پژواک» و «پدیده داپلر» این دو پدیده به سادگی قابل فهم می‌باشند، چرا که هر روزه شما با آنها در حوزهشنوایی خویش برخوردارید. رادار از این دو پدیده در حوزهٔ امواج رادیویی استفادهمی‌برد. بگذارید ابتدا با این پدیده در حوزهشنیداری یا صوتی خویش بیشتر آشنا شویم. پژواک و پدیده داپلر پژواک پدیده‌ای است که شما هر روزه با آن برخورد دارید، اگرشما به داخل یک چاه و یا در یک دره فریاد بزنید، پژواک صدای شما چند لحظه بعد بهگوشتان می‌رسد. در واقع شما صدایتان را باز خواهید شنید. پژواک بدین جهت رخ می‌دهدکه بعضی از امواج صدای شما (به این دلیل واژه بعضی را آوردیم که صدای برخی ازحیوانات مانند اردک در فرکانس خاص امواج صدای این حیوان هیچگاه پژواکی ندارد) پس ازبرخورد به یک سطح (که این سطح می‌تواند سطح آب، انتهای چاه یا دیوارهٔ کوه موجود درانتهای دره باشد) به سمت شما باز می‌گردد و گوش شما دوباره آنرا می‌شنود. فاصلهزمانی‌ای که بین فریاد شما تا شنیدن پژواک آن طول می‌کشد با فاصله مکانی بین شما وآن سطح بازگردانندهٔ پژواک ارتباط دارد. هنگامی که شما به داخل یک چاه فریاد می‌کشید، صدای شما از دهانهٔ چاه به سمت انتهای چاه رفته و پس از برخورد با سطح آب انتهای چاه منعکس می‌شود. در این حالت اگر شما سرعت صدا را به طور دقیق بدانید، با اندازه‌گیری زمان رفت‌وبرگشت صدا می‌توانید عمق چاه را حساب کنید پدیدهٔ داپلر نیز بسیار معمول است. شماهر روز (بدون اینکه حتی از آن درکی داشته باشید) آن را تجربه می‌کنید. این پدیدهزمانی رخ می‌دهد که یک مولد امواج صوتی و یا منعکس کننده امواج صوتی دارای حرکتباشد. مثلاً یک خودرو که در حال بوق زدن است. حالت تشدید شدهٔ پدیدهٔ داپلر در شکستن «دیوار صوتی» رخ می‌دهد. در این جا به درک این پدیده می‌پردازیم (ممکن است شما برایاینکه بهتر این پدیده را درک کنید کنار یک اتوبان آن را

تجربه کنید) فرض کنید کهخودرویی با سرعت ۱۰۰ کیلومتر بر ساعت در حال بوق زدن به سمت شما در حرکت باشد. تازمانی‌که خودرو در حال نزدیک شدن به شماست فقط یک نت صوتی را می‌شنوید (در واقع یکفرکانس ثابت، در شماره گذشته راجع به فرکانس صحبت کردیم)، اما هنگامی که خودرو بهکنار شما می‌رسد صدای بوق ناگهان تغییر کرده و به عبارتی «بم» تر می‌شود و بعد ازلحظه‌ای که از شما عبور کرد (و اگر همچنان راننده در حال بوق زدن بود) ناگهان صدابم‌تر نیز می‌شود، در صورتی که شما می‌دانید که صدای بوق همیشه ثابت است، کما اینکهراننده داخل خودرو در تمام مدت بوق زدن فقط نت واقعی بوق را می‌شنود. این تغییراتصوت شنیده شده توسط شما بوسیلهٔ پدیدهٔ داپلر قابل توضیح است. اما آنچه که رخ می‌دهد: «سرعت صوت» مقداری ثابت است، برای ساده‌تر شدن محاسباتمان سرعت صورت را ۱۰۰۰کیلومتردر ساعت در نظر بگیرید. (سرعت واقعی صوت وابسته به دما، فشار هوا و رطوبت هواست.) فرض کنید که خودرویی در فاصله یک کیلومتری شما قرار دارد (بصورت غیر متحرک). رانندهداخل خودرو به مدت یک دقیقه شستی بوق را فشرده تا صدا به گوش ما برسد، این صدا باسرعتی برابر با ۱۰۰۰کیلومتر بر ساعت به سمت شما حرکت می‌کند، بعد از ٦ ثانیه ازفشرده شدن شستی بوق توسط راننده، شما چه صدایی را خواهید شنید؟ (این ٦ ثانیه درواقع مدت زمانی است که طول می‌کشد صدا به شما برسد) و به مدت یک دقیقه پس از آن چهمی‌شنوید؟ مسلماً صدای بوق را بدون هیچ تغییری. پدیدهداپلر: شخص پشت سر خودرویی را با بسامدی (فرکانس) پایین‌تر و بم‌تر از آنچهکه راننده داخل خودرو و در حال حرکت می‌شنود. راننده از شخصی که خودرو به سمت آن درحال حرکت است صدا را با نت پایین‌تر می‌شنود. حال فرض کنید خودرو از فاصله‌ای دور باسرعتی معادل ۱۰۰ کیلومتر بر ساعت به سمت شما حرکت کند، همان راننده با همان خودرو وبا همان صدای بوق و به مدت همان یک دقیقه شستی بوق را فشارمی‌دهد می‌شود. جالب است! شما صدای بوق را فقط به مدت ٥٤ ثانیه خواهید شنید آن هم به خاطر حرکت خودرو رخ دادهاست. در واقع تعداد اعوجاجهای موج صوتی ثابتبوده ولی در زمان کوتاه‌تری به سمت شما آمده و از آنجائی‌که تعریف فرکانس تعدادنوسانات موج در واحد زمان است لذا اگر قبلاً این نوسانات را ۱ بر ٦۰ ثانیه تقسیم‌کردیم و فرکانس f۱ بدست می‌آمد، حال باید این تعداد نوسانات را بر ٥٤ تقسیم کنیمکه مطمئناً عددی بزرگتر خواهد شد. این عدد بزرگتر یا فرکانس بالاتر یعنی صدای «زیر»تر. همین توجیه نیز برای خودرویی که از شما وجود دارد، در این حالت شما ٦٤ثانیه صدای بوق را می‌شنوید که فرکانس حاصله در این حالت کمتر (یا صدای بم‌تر) خواهد بود. شکستن دیوارصوتی اینک که ما در حال بحث بر روی

رابط صداو سرعت هستیم می‌توانیم در مورد شکستن دیوار صوتی هم صحبت کنیم. فرض کنید آنخودرویی که صحبتش بود با سرعتی معادل ۱۰۰ کیلومتر در ساعت به سوی شما، آن هم درحال بوق زدن، حرکت کند، امواج صوتی چون سرعتی معادل همان سرعت خودرو را دارند، لذانه از آن جلو زده و نه عقب می‌مانند، لذا در کل مدت حرکت خودرو شما صدایی رانخواهید شنید. اما در لحظه‌ای که خودرو به شما می‌رسد، تمام امواج صوتی جمع شده ویکجا شما آنها را می‌شنوید. صدای بسیار بلند و با فرکانس بسیار بالا. این صدا توسط هواپیمایی که قادرند باسرعتی معادل با سرعت صوت حرکت کنند می‌تواند موجبات وحشت بسیاری از افرادی که درزیر مسیر این هواپیما قرار دارند بوجود آورده قدرت این صدا به قدری است که می‌تواندشیشه‌ها را بشکند. چنین اتفاقی برای قایقها نیز رخ می‌دهد. منتهی در این میان تجمع امواج آب که سرعتی در حدود سرعت این قایقها دارند. این موجمتمرکز بصورت v شکل از جلو قایق به طرفین حرکت می‌کند که زاویه این موج توسط سرعتقایق کنترل می‌شود. در واقع تجمع امواجی که قایق در هر لحظه تولید می‌کند و هر لحظهبر آن می‌افزاید نیز توسط پدیده داپلر قابل توضیح است. شما می‌توانید با استفاده از ترکیبی از پژواک و پدیده داپلر بصورتی که در زیر می‌آید استفاده کنید در محلی که ایستاده‌ایدبه سمت خودرویی که در حال حرکت (به سمت شما یا در خلاف جهت) اصواتی را بفرستید. بعضی از این اصوات پس از برخورد با خودرو به سمت شما باز می‌گردند. (پژواک) ازآنجایی که خودرو در حال حرکت است لذا اصوات منعکس شده یا به هم فشرده می‌شوند (درحالی که خودرو به سمت شما می‌آید) و یا از هم باز می‌شوند. در حالت حرکت مخالف درهر دو صورت شما می‌توانید با مقایسه موج فرستاده شده و بازگشته سرعت خودرو را بدستآورید.مفهوم رادار: دیدیم که می‌توان با استفاده از مفهومپژواک به فاصله اجسام دور پی برد و همین طور با استفاده از تغییر پدیده داپلر بهسرعت این جسم پی ببریم. با توجه به این مفاهیم می‌توان فهمید که رادار صوتی چیست؟این گونه رادار در زیردریایی‌ها و کشتی‌ها کاربرد دارد و همیشه در حال کار است. می‌توان از رادار صوتی در محیط آزاد نیز استفاده کرد، اما بخاطر چند اشکال ریز اینگونه رادار در هوا استفاده نمی‌شود.

– صدا در هوا مسافت زیادی را نمی‌تواندبپیماید.... شاید در حدود ۱/۵ کیلومتر و یا کمی بیشتر

– هرکسی می‌تواند صدا را بشنود لذااستفاده از صدا در محیط آزاد موجب آزار دیگران می‌شود که البته می‌توان با بالابردن فرکانس صدای مورد استفاده و استفاده از امواج «فراصوت» این مشکل را حل کرد.

– صدای منعکس شده حاصل از پدیده پژواکبسیار ضعیف می‌باشد به طوری که دریافت آن بسیار سخت است.

سمت چپ: آنتن های مجموعه مخابراتی فضاییگلدستون (بخشی از شبکه ارتباطی فضایی ناسا) که به ارتباطات مخابراتی رادیوییفضاپیماهای میان سیاره‌ای ناسا کمک می‌کند.

سمت راست: رادار جست وجوی سطح و هوا کهبر روی نوک دکل یک موشک هدایت شونده قرار گرفته است.

حال بیایید در مورد یک نمونه واقعیراداری که برای شناسایی هواپیماهای در حال پرواز بکار می‌رود صحبت کنیم. سیستمرادار در ابتدا با روشن کردن فرستنده قوی‌اش یک دسته موج رادیویی متراکم در آسمان ودر جهات مختلف پخش می‌کند. این ارسال برای چند میکروثانیه صورت می‌پذیرد، حالفرستنده خاموش شده و گیرنده سیستم رادار مترصد دریافت پژواک امواج که به همراهاطلاعات حاصل از پدیده داپلر نیز هستند می‌ماند.

امواج رادیویی با سرعتی معادل سرعت نورحرکت می‌کنند، تقریباً در هر میکروثانیه ۳۰۰ متر را در فضا طی می‌کنند؛ حال اگرسیستم رادار مذکور دارای یک ساعت بسیار دقیق و قوی باشد، می‌تواند با دقت بسیاربالایی موقعیت هواپیما را مشخص کند، با استفاده از روشهای خاص پردازش سیگنال برایتحلیل پدیده داپلر بر روی موجهای برگشتی می‌توان به دقت سرعت هواپیما را مشخص کرد.

آنتن رادار یک دسته کوچک اما قدرتمندپالس امواج رادیویی از یک فرکانس مشخص را در فضا می‌فرستند. هنگامی که امواج به یکجسم برخورد می‌کنند منعکس شده و در اثر پدیده داپلر فشرده‌تر یا گسسته‌تر می‌شوند. همان آنتن وظیفه دریافت امواج منعکس شده را که البته بسیار کمتر از امواج ارسالیهستند بر عهده دارد.

در رادارهای زمینی قضیه خیلی پیچیده‌تراز رادارهای هوایی است، هنگامی که یک رادار پلیس به ارسال پالس موج رادیوییمی‌پردازد بخاطر وجود اجسام بسیار در سر راهش مانند نرده‌ها، پلها، تپه‌ها وساختمانها پژواکهای بسیاری را دریافت می‌دارد، اما از آنجایی که تمام این اجسامثابت هستند به جزء خودروها مورد نظر، لذا سیستم رادار خودروهای پلیس از میان امواجمنعکس شده، فقط آنهایی را انتخاب می‌کند که در آنها پدیده داپلر قابل شناسایی است،آن هم به اندازه‌ای که جسم متحرک اضافه سرعت داشته باشد، در ضمن آنتن این رادارهابسیار دهانه تنگی دارند، چرا که فقط بر روی یک خودرو تنظیم می‌شوند.

البته امروزه پلیسها در برخی کشورها از جمله کشور خودمان از تکنولوژی لیزر برای تعیین سرعت خودروها در بزرگراهها استفاده می‌کنند. تکنولوژی به نام «لیدار» شناخته می‌شود. در این مدل بجای امواج رادیویی از اشعه نوری متمرکز (یا همان لیزر) استفاده می‌شود.

## ۲-۴- کاربردهای غیرنظامی

به برخی از کاربردهای غیر نظامی رادار که در جهت صلح و آرامش و راحتی زندگی انسان استفاده می‌شود بطور اختصار اشاره می‌کنیم :

۱-کنترل ترافیک هوایی: کنترل ترافیک و اعلام وضعیت هوایی در اطراف فرودگاه‌ها و در برخی از هواپیماهای پیشرفته در یاری رساندن به خلبان هنگام فرود و در وضعیت بد آب و هوایی که دید کافی وجود ندارد

۲-ناوبری هوایی و دریایی: جهت نشان دادن موقعیت، سرعت، مسافت طی شده و مسیر یابی در هر لحظه

۳-جلوگیری از تصادف کشتی‌ها: استفاده از یک رادار کوچک با برد محدود در جلوی کشتی جهت شناسایی موانع مقابل کشتی

٤-فضایی: سنجش از دور[۳] و شناسایی اجرام و کرات آسمانی

٥-کنترل سرعت: کنترل سرعت خودروها در بزرگراه‌ها توسط پلیس

٦-کنترل خط تولید: کنترل خط تولید و سرعت بهره برداری از خطوط

۷-هواشناسی: پیش بینی وضعیت آب و هوای مناطق مختلف با استفاده از جهت وزش باد و سایر عوامل موثر

۸-زمین شناسی: بررسی و شناسایی وضعیت اقیانوس‌ها، دریاها، منابع زیر زمینی، معادن و آتشفشان‌ها

۹-کشاورزی: محاسبه میزان اراضی زیر کشت و برآورد محصولات مختلف کشاورزی با استفاده از واقعیت که محصولات مختلف کشاورزی دارای خواص الکترومغناطیسی (انعکاس امواج) متفاوتی هستند

[۳] Remote Sensing

### ۲–۵– کاربردهای نظامی

کاربردهای نظامی رادار دارای طیف و انواع گسترده‌ای است که در این جا به چهار نمونه از این کاربردها اشاره می‌شود :

۱–دیده‌بانی، مراقبت و تعیین مشخصات هدف[٤]: با توجه به نوع کاربرد، باند فرکانسی این رادارها و مشخصات آنها متفاوت است و برد آنها تا حدود ٤۰۰ کیلومتر قابل افزایش است.

۲–ناوبری نظامی: هدایت هواپیما در حین پرواز و هنگام فرود و صعود و تعیین ارتفاع و سرعت هواپیمای نظامی

۳–کنترل و هدایت آتش[٥]: کنترل و هدایت آتش که بنا به چگونگی بهره برداری (هوا به هوا، زمین به هوا، زمین به دریا و هوا به زیر دریا) متفاوت می‌باشد (در این مورد از رادارهای تک پالسی استفاده می‌شود)

٤–ردیابی[٦]: مشخص کردن مسیر و مقصد اهداف متحرک مانند هواپیما یا موشک‌های بالستیک (برد رادارهای فوق بسیار بیشتر از رادارهای کنترل و هدایت آتش است ولی از نظر اصول کار شبیه یکدیگرند)

## ۲–۶– انواع رادار

رادارها با توجه به فرکانس کار، محیط عمل، قدرت فرستنده، حساسیت گیرنده، نوع آنتن و چندین عامل دیگر دسته‌بندی و هر یک در موارد خاصی به کارگیری می‌شوند و معمولا هر دسته نوع خاصی از فرستنده و سیستم پردازش سیگنال را مورد استفاده قرار می‌دهند. در ادامه انواع رادار معرفی می‌شوند:

### ۲–۶–۱– رادارهای پالسی[۷]

در این رادارها موج ارسالی به صورت یک پالس با فرکانس مشخص می‌باشد که به فرکانس تکرار پالس[۸] معروف است. نسبت دوره تناوب به عرض پالس را نسبت زمان کار می‌گویند.

---

[٤] Early Warning

[٥] Fire Control System

[٦] Tracking

[۷] Pulse Transmission

[۸] Pulse Repeat Frequency (PRF)

رادارهای پالسی با توجه به دوره تناوب و نسبت زمان کار دارای تنوع بوده که به مواردی از آنها اشاره می‌شود:

۱-رادارهای پالسی معمولی: در این رادارها معمولا عرض پالس در حدود چند میکروثانیه است و نسبت زمان کار بین حدود ۰.۰۱ تا ۰.۰۰۱ تغییر می‌کند. از این رادارها جهت هواشناسی و دیده‌بانی و مراقبت هوایی استفاده می‌شود.

۲-رادارهای پالسی با قدرت تفکیک بالا[۹]: در این رادارها عرض پالس بسیار کوچک انتخاب می‌شود و چون میزان دقت در تشخیص فاصله توسط عرض پالس مشخص می‌گردد دارای دقت بالایی در تشخیص فاصله هدف می‌باشد. (هر قدر عرض پالس کوچکتر باشد محاسبه فاصله دقیقتر است.) این رادارها برای آشکار سازی اهداف ساکن در حضور کلاتر[۱۰] (سیگنالهای برگشتی ناخواسته به صفحه رادار) و نیز تشخیص یک هدف در میان چند هدف نزدیک به هم قابل استفاده می‌باشد.

۳-رادار پالس فشرده[۱۱]: این رادار از پالس‌های با عرض زیاد استفاده می‌نماید و برای افزایش دقت از مدولاسیون فاز یا فرکانس در هر پالس استفاده می‌کند. در نتیجه ضمن افزایش پهنای باند، تشخیص دقیق فاصله اهداف نیز حاصل می‌شود و نسبت به رادار نوع قبلی دارای این مزیت است که توان پیک (حداکثر توان) فرستنده را در حد معتدلی نگاه می‌دارد.

### ۲-۶-۲- رادار موج پیوسته[۱۲]

این رادار دارای نسبت زمان کار واحد می‌باشد، یعنی موج ارسالی به صورت پیوسته است. این نوع رادار نیز دارای انواع مختلف به ترتیب زیر است :

۱-رادار موج پیوسته معمولی: در این نوع رادار می‌توان سرعت و جهت حرکت هدف را در راستای خط واصل رادار تشخیص داد و امکان تشخیص فاصله به دلیل عدم استفاده از هرگونه مدولاسیون وجود ندارد و معمولا در ناوبری هوایی کاربرد دارند .

---

[۹] High Resolution

[۱۰] Clutter

[۱۱] Pulse Compression

[۱۲] Continuous Wave Radar

۲-رادار موج پیوسته مدوله شده فرکانس۱۳: در این رادار از مدولاسیون۱٤ فرکانس برای افزایش پهنای باند و ایجاد امکان تشخیص فاصله استفاده می‌شود. از مهم ترین کاربردهای این نوع رادار ارتفاع سنج های هواپیما می‌باشد.

۳-رادار موج پیوسته چند فرکانسه۱۵: در این نوع رادار با توجه به اختلاف فاز موج دریافتی از یک هدف در فرکانس‌های مختلف می‌توان فاصله هدف را تشخیص داد.

### ۲-۶-۳- رادار روزنه مصنوعی۱۶

در این رادار معمولا پرتوی[۱۷] آنتن در جهت عمود بر مسیر حرکت تنظیم می‌شود و دارای دقت بالایی در زاویه است . عملکرد رادارها مانند یک آنتن ساکن با تعداد زیادی آرایه می‌باشد. خروجی این نوع رادار یک تصویر با دقت بالا از صحنه مورد نظر می‌باشد. رادارهای فوق به دلیل ایجاد تصاویر دقیق کاربردهای فراوانی در علم زمین شناسی و جغرافی و همچنین در امور نظامی پیدا کردند. این رادار حتی قادر به ایجاد تصاویر سه بعدی از اشیا و اهداف می‌باشد.

### ۲-۶-۴- رادار SLAR۱۸

اساس کار این رادار مانند رادار روزنه مصنوعی است اما جهت دید آنها از کنار می‌باشد یعنی به کمک این رادار می‌توان بدون نزدیک شدن به هدف از فواصل بسیار دور از آن تصویر برداری کرد. البته دقت تصاویر ایجاد شده کمتر از رادار روزنه مصنوعی است.

### ۲-۶-۵- رادار TWS۱۹

این رادار دو نوع دارد: نوع اول دارای آنتن گردان است و ردیابی هدف از طریق مقایسه انعکاسات مختلف از هدف در طی دو چرخش متوالی آنتن حاصل می‌شود. از این نوع رادار جهت دیده بانی و مراقبت هوابرد استفاده می‌شود. نوع دوم راداری است که آنتن آن سریعا زاویه کوچکی

---

[۱۳] Frequency Modulated

[۱٤] Modulations

[۱۵] Multi-Frequency

[۱٦] Synthetic Aperture Radar (SAR)

[۱۷] Beam

[۱۸] Side Looking Airborne Radar

[۱۹] Track While Scan

را جاروب می‌کند تا موقعیت زاویه‌ای هدف را بیابد. از این رادار در فرود هواپیما و کنترل آتش استفاده می‌گردد.

### ۲-۶-۶- رادار HF-OTH[۲۰]

این رادار در باند فرکانس بالا[۲۱] عمل می‌کند و برد آن به دلیل انتشار در لایه اتمسفر بسیار زیاد است و در بعضی مواقع به ٤۰۰۰ کیلومتر نیز می‌رسد. این رادار قادر به آشکارسازی اهداف هوایی بزرگ مانند هواپیماها و موشک‌های بالستیک و همچنین اهداف دریایی می‌باشد و به دلیل برد زیاد توان ارسالی بالایی حدود چند صد کیلو وات نیز دارد.

### ۲-۶-۷- رادار پالس دوپلر[۲۲]

این رادار به صورت پالسی عمل می‌کند و می‌تواند اهداف متحرک را در حضور کلاترهای قوی آشکارسازی نماید. در این رادار ابهام در تشخیص سرعت از بین می‌رود و سرعت کور وجود ندارد .

### ۲-۶-۸- رادار دریایی

علی‌رغم اینکه کار گیرندگی و فرستندگی رادار دریایی و غیر دریایی یکسان است اما با توجه به وجود کلاترهای خاص دریایی باید ملاحظات ویژه ای را در مورد استفاده از رادار در دریا لحاظ نمود. از مهمترین عوامل کلاترها می‌توان وجود طوفان، باد و رطوبت را برشمرد. سیگنال‌های کلاتر دریایی عامل محدود کننده عمده در شناسایی اهداف نزدیک آب و یا روی آب می‌باشد.

### ۲-۶-۹- رادار میلیمتری

فرکانس کار این رادار ۳ الی ۳۰۰ گیگاهرتز است و مزیت عمده آن این است که برای ابعاد آنتن یکسان پهنای پرتوی آنتن کوچکتر و در نتیجه بهره آن بیشتر می‌گردد و حتی برای افزایش بهره می‌توان آنتن را نیز کوچکتر کرد. این قابلیت در مورد رادارهای ردیاب داخل موشک‌ها برای پیدا کردن مسیر صحیح بسیار لازم و حیاتی است. از دیگر مزیت‌های این سیستم‌ها، سبک و قابل حمل بودن آنهاست. اما مشکل عمده آنها اثرات تضعیف‌کنندگی شدید جو اتمسفر در انتشار امواج در این

---

[۲۰] High Frequency Over The Horizon

[۲۱] High Frequency Band (HF Band)

[۲۲] Pulse Doppler Radar

محدوده فرکانس می‌باشد، لذا برد این رادارها بسیار کمتر از سایر رادارهاست. در ضمن عواملی مثل باران و گرد و غبار و رطوبت و برف نیز باعث ایجاد تلفات اضافی در سیگنال در باند میلیمتری می‌شوند. این نوع رادارها عموما در امور نقشه برداری و هدایت و کنترل آتش نزدیک کاربرد دارند.

# فصل ۳– معرفی جهت‌یاب راداری

## ۳–۱– جهت یاب راداری چیست

جهت‌یاب راداری وسیله‌ای برای یافتن جهت رادارهای موجود در یک محدوده می‌باشد. این وسیله باید قادر به ارائه جهت کلیه رادارهای نصب شده بر روی هواپیماها، کشتی‌ها و یا رادارهای استقراری باشد. به منظور انجام این کار دسته‌بندی بسیار دقیقی بر روی پالس‌های دریافتی صورت می‌گیرد و سامانه باید بتواند به طور همزمان چندین هدف را جهت‌یابی نماید. ارائه داده‌های مربوط به این پالس‌های تفکیک شده در قالب یک محیط گرافیکی و گویا و توان تجزیه و تحلیل Offline قابلیت‌هایی است که برای نرم‌افزار متصل به جهت‌یاب راداری ضروری است.

جهت‌یاب راداری[۲۳] مورد استفاده نرم افزار NRDF از یک روش مقایسه دامنه برای استخراج جهت رادار بهره می‌برد. در این سامانه دو آنتن وجود دارد که یکی آنتن جهت‌دار[۲۴] و تمام‌جهت[۲۵] است. همچنین یک کارت پردازش‌گر نیز در کنار این دو آنتن قرار دارد که وظیفه تبدیل سیگنال‌های دریافت شده توسط آنتن‌ها و تبدیل آنها به داده‌های دودویی[۲۶] برای استفاده توسط رایانه را بر عهده دارد.

نحوه کار سامانه جهت‌یاب راداری استفاده شده در این پروژه به این صورت است که با شروع جستجو، آنتن جهت‌دار که بر روی یک موتور چرخشی قرار گرفته است شروع به چرخش کرده و سیگنال‌های موجود را در ۳۶۰ درجه اطراف خود جمع‌آوری می‌کند. از طرف دیگر آنتن بدون جهت کلیه سیگنال‌های موجود در اطراف دستگاه جهت‌یاب راداری را جمع‌آوری می‌کند. سیگنال‌های دریافت شده توسط دو آنتن به همراه جهت دریافت سیگنال جهت‌دار به کارت پردازش‌گر ارسال می‌گردد. کارت پردازش‌گر با پردازش سیگنال‌های دریافت شده پالس‌های مربوط به رادارها را از بین این سیگنال‌ها استخراج کرده و به صورت که PDW[۲۷] به نرم‌افزار ارسال می‌کند. نرم‌افزار با پردازش این اطلاعات و مقایسه قدرت سیگنال‌های دریافتی از آنتن‌های جهت دار و بدون جهت

---

[۲۳] Radar Direction Finder

[۲۴] Directional Antenna

[۲۵] Omni Antenna

[۲۶] Digital

[۲۷] Pulse Descriptive Word

رادارهای موجود را تشخیص داده و نتیجه را برای کاربر نمایش می‌دهد. در این عملیات در صورتی که قدرت سیگنال آنتن جهت‌دار بیشتر از آنتن بدون‌جهت باشد، سیگنال دریافت شده از آنتن جهت‌دار به عنوان یک رادار بالقوه در نظر گرفته می‌شود.

# فصل ۴– سیستم های مکان‌یاب بلادرنگ

## ۴–۱– مقدمه

تکنولوژی‌های بسیاری برای ساخت سیستم‌های مکان‌یابی بلادرنگ مورد استفاده قرار می‌گیرند. برخی ایز این سیستم‌ها از برچسب‌های فرکانس رادیویی و برخی از شبکه‌های بی‌سیم موجود استفاده می‌کنند و قابلیت‌های مکان‌یابی را به این شبکه‌ها اضافه می‌کنند. با این حال، اکثر این سیستم‌ها متشکل از دو بخش اساسی هستند. یکی مجموعه‌ای از سنسورهای بی‌سیم که برای محاسبات مقادیر پایه‌ای (که در ادامه شرح می‌دهیم) به کار می‌روند و یک موتور مکان‌یابی که با استفاده از این داده‌های خام مکان را محاسبه می‌کند. در این قسمت ما روش‌های متداولی که اغلب سیستم‌های مکان‌یابی بلادرنگ دوبعدی و سه‌بعدی از آن‌ها استفاده می‌کنند را از نظر می‌گذرانیم.

## ۴–۲– متدولوژی‌های مکان‌یابی بلادرنگ

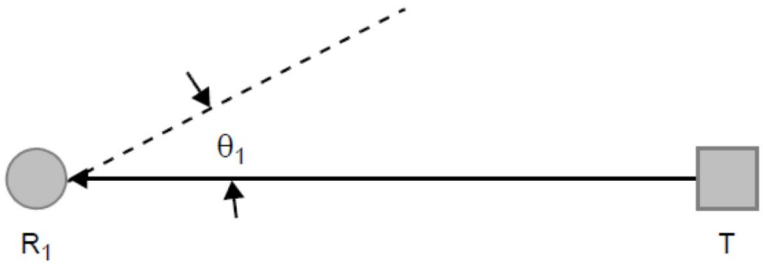
روش‌های بسیاری برای محاسبه‌ی حوزه (range) قابل استفاده است که بر اساس ویژگی‌های متعددی چون نوع و کیفیت سیگنال، گستردگی استفاده، امنیت در برابر تخریب و ... انتخاب می‌شوند. تعدادی از عمومی‌ترین روش‌های مورد استفاده در این حوزه به شرح زیرند:

- زاویه‌ی دریافت (Angle of Arrival - AoA)
- زمان دریافت (Time of Arrival - ToA)
- اختلاف زمان‌های دریافت (Time Difference of Arrival - TDoA)
- قدرت سیگنال دریافتی (Received Signal Strength - RSS)

در عمل از ترکیب برخی از این روش‌ها نیز استفاده می‌شود.

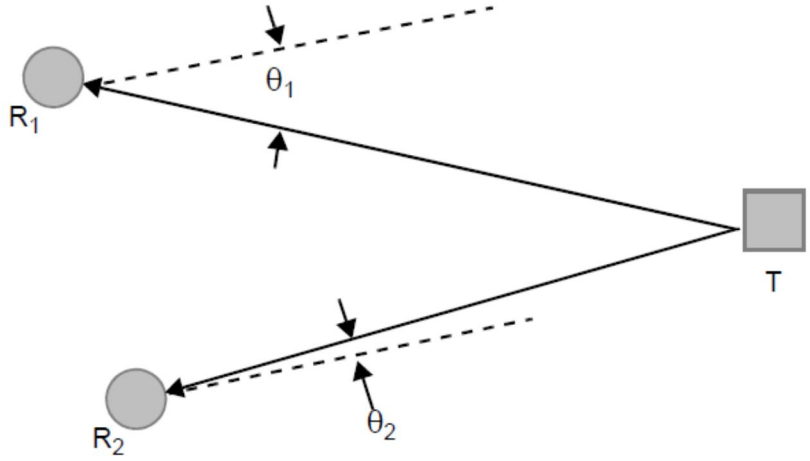
### ۴–۲–۱– زاویه‌ی دریافت (AoA)

زاویه‌ی دریافت، روشی برای تشخیص جهت انتشار یک سیگنال رادیویی است. با استفاده از آنتن‌ها حساس به جهت، جهت نسبی فرستنده قابل تشخیص است. زاویه‌ی دریافت از روی زاویه‌ی خطی که فرستنده را به گیرنده وصل می‌کند، با یک خط فرضی (برای مثال شمال) به دست می‌آید. این روش را می‌توان در شکل زیر نشان داد که در آن T هدف و R گیرنده هستند.




**تصویر ۱ – روش زاویه‌ی دریافت**

با استفاده از مکان دو گیرنده که در نقاط مشخصی قرار می‌گیرند، با استفاده از محاسبات بسیار ساده‌ی مثلث‌سازی[۲۸]، مکان فرستنده به راحتی قابل تشخیص است. زاویه‌ی هدف با هریک از گیرنده‌ها، در آن‌ها حساب می‌شود و موتور محاسبات با دریافت این داده‌ها، و انجام محاسبات، مکان هدف را مشخص می‌کند. شکل ط این روش را نشان می‌دهد که در آن دو گیرنده‌ی R۱ و R۲ برای شناسایی مکان هدف T به کار می‌روند.




**تصویر ۲ – تشخیص مکان هدف با استفاده از AoA**

اندازه‌گیری‌ها در این روش معمولاً نیاز به مجموعه‌ی پیچیده‌ای از ٤ تا ۱۲ آرایه از آنتن‌ها دارد. دقت این روش با افزایش تعداد آنتن‌های استفاده شده، افزایش می‌یابد که هزینه‌ی زیادی را در پی دارد. استفاده از این روش در نواحی شهری با تعداد زیاد ساختمان‌ها به دلیل پدیده‌ی انتشار چند مسیره با خطای بسیار زیادی همراه است، چرا که سیگنال دریافت شده در آنتن‌ها الزاماً به صورت مستقیم از فرستنده نرسیده است، بنابراین استفاده از این روش عموماً محدود به مکان‌هایی است که خط دید[۲۹] مستقیم نسبت به هدف وجود دارد. به دلیل همین مشکل در این نوع مکان‌یابی، امکان

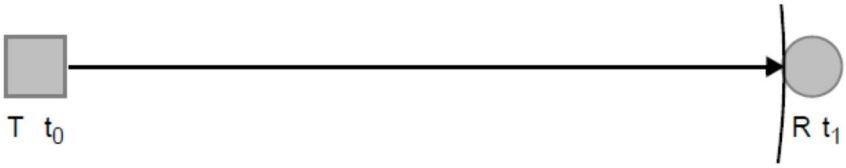
[۲۸] Triangulation معادل کلمه‌ی

[۲۹] Line of Sight معادل عبارت

تهدادهای امنیتی بر علیه این‌گونه از مکان‌یابی نیز با ارسال سیگنال از نقاط دیگر یا ارسال بازتابی آن وجود دارد.

### ۴-۲-۲- زمان دریافت (ToA)

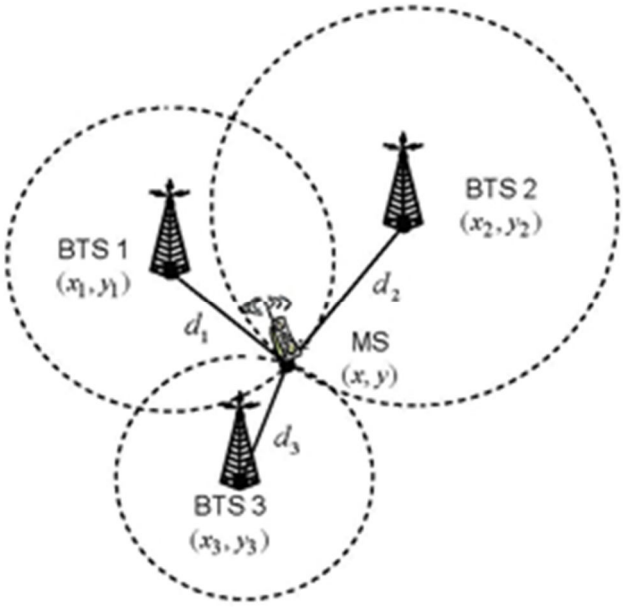
روش زمان دریافت براساس ایده‌ی تاخیر انتشار سیگنال‌های رادیویی مابین فرستنده و گیرنده کار می‌کند. تاخیر انتشار، تفاوت زمانی ارسال سیگنال در فرستنده تا دریافت آن در گیرنده است. به عبارت دیگر این زمان، زمان لازم برای انتقال سیگنال از فرستنده به گیرنده است.




**تصویر ۳ – روش زمان دریافت**

با ضرب تاخیر انتشار در سرعت انتشار سیگنال، می‌توان از روی میزان تاخیر، میزان فاصله‌ی گیرنده از هدف را مشخص کرد. برای مشخص کردن مکان هدف در صفحه‌ی دو بعدی، حداقل نیاز به سه گیرنده برای استفاده از روش زمان دریافت داریم. برای مکان‌یابی در فضای سه بعدی ٤ گیرنده مورد نیاز است که مکان هدف به ترتیب نقاط تقاطع دایره‌ها و کره‌ها است.

شکل زیر استفاده از روش زمان دریافت را در فضای دو بعدی نشان می‌دهد که در آن T هدف و $R_1$، $R_2$ و $R_3$ گیرنده‌ها هستند. سیگنال در زمان $t_0$ از هدف ارسال می‌شود و در زمان‌های $t_1$، $t_2$ و $t_3$ توسط گیرنده‌های مربوطه دریافت می‌شود.




**تصویر ٤ – مشخص کردن مکان هدف در ToA**

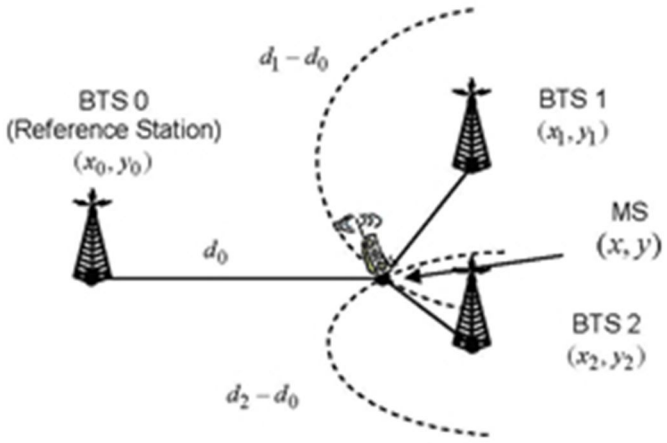
برای اندازه‌گیری تاخیر انتشاری بایستی ساعت فرستنده و گیرنده هم‌زمان باشد که این به نوبه‌ی خود یعنی هم‌زمانی تمام اجزای سیستم. هرچه میزان هم‌زمانی بیشتر باشد، دقت مکان‌یابی نیز بیشتر خواهد بود و خطای مکان‌یابی ارتباط مستقیم با میزان هم‌زمانی دارد. مکانیزم‌های هم‌زمانی معمولاً روش‌های بسیار پرهزینه‌ای هستند و استفاده از روش‌های کم‌هزینه به ندرت دقت هم‌زمانی در حدود چند صد میکروثانیه می‌دهد. این درحالی است که اختلاف زمانی مورد قبول در این روش (و روش‌های مشابه، نظیر TDoA) تا حداکثر چند ده نانوثانیه است. به علاوه استفاده از حداقل ۳ گیرنده در فضای دو بعدی و ٤ گیرنده در فضای سه بعدی هزینه‌های اجرایی را نیز بالاتر می‌برد. دقت کنید که این مقادیر کمینه‌ی تعداد مورد نیاز هستند و دقت مناسب عملاً با تعداد بیشتری گیرنده فراهم می‌شود.

### ۴–۲–۳– اختلاف زمان‌های دریافت (TDoA)

حال اگر به جای زمان‌های دریافت به اختلاف زمان‌های دریافت توجه کنیم، پارامتر زمان ارسال از معادلات حذف می‌شود و نیاز به هم‌زمانی هدف با گیرنده‌ها از بین می‌رود. در حالی که در روش زمان دریافت مکان هندسی هدف از دید یک گیرنده، یک دایره در حالت دو بعدی و یک کره در حالت سه بعدی است، در روش اختلاف زمان‌های دریافت، مکان هندسی هدف از دید هر گیرنده یک هذلولی (یک پوسته‌ی هذلولی در حالت سه بعدی) است. سیستم‌هایی که از اختلاف زمان

دریافت استفاده می‌کنند، تفاوت زمان‌های انتشار را از هدف به تک تک گیرنده‌ها در نظر می‌گیرند، به همین دلیل به این روش *مکان‌یابی هذلولی سه بعدی* نیز می‌گویند.

با استفاده از اختلاف زمان دریافت، نیاز به ۳ یا ٤ (مانند حالت قبل) گیرنده با موقعیت مشخص است. هریک از گیرنده‌ها به صورت سنکرون سیگنال را از گیرنده گرفته و لحظه‌ی دریافت سیگنال را (عموماً بر اساس پیک سیگنال) ثبت می‌کنند. این اطلاعات به موتور مکان‌یابی منتقل می‌شود و در آن‌جا تفاضل زمانی‌های دریافت محاسبه می‌شود. این اطلاعات با استفاده از الگوریتمی برای محاسبه‌ی مکان هدف مورد استفاده قرار می‌گیرد. به لحاظ ریاضی، مکان هدف نقطه‌ی تلاقی هذلولی‌ها در صفحه‌ی دو بعدی است (به طور مشابه نقطه تلاقی رویه‌های هذلولی در فضای سه بعدی). شکل ط شیوه‌ی عملکرد این روش را نشان می‌دهد.



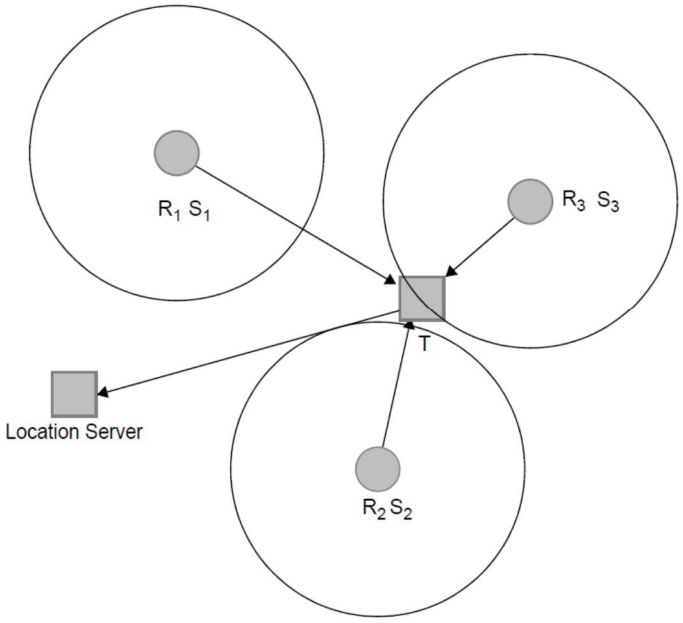

**تصویر ٥ – تشخیص مکان هدف با TDoA**

مشکلات این روش مشابه روش زمان‌های ریافت است با این تفاوت که در روش اختلاف زمان‌های دریافت نیازی به هم‌زمانی هدف با گیرنده‌ها نیست. دقت مکان‌یابی با دقت ساعت‌های گیرنده‌ها هم‌بسته است، به این ترتیب ساعت‌های دقیق‌تر منجر به مکان‌یابی دقیق‌تر می‌شوند و از طرفی هزینه‌های ساخت سیستم را بالا می‌برند. در کاربردها غیر نظامی معمولاً از الگوریتم‌ها هم‌زمانی مبتنی بر شبکه استفاده می‌کنند که دقت‌هایی در حدود چند صد میکروثانیه ارائه می‌کند. در این حالت دقت مکان‌یابی چند صد متر خواهد بود. در سیستم‌های نظامی معمولاً از ساعت GPS با دقت چند ده نانو استفاده می‌شود در این حالت دقت مکان‌یابی حدود چند متر خواهد بود. به علاوه، انتشار از چند مسیر، نویز و تداخل نیز بر این روش تاثیرگذار است و ممکن است منجر به اشتراک نادقیق هذلولی‌ها شود. قاعدتاً ترجیح بر خط دید مستقیم و فضای باز است.

### ۴-۲-۴- نشانه‌ی قدرت سیگنال دریافتی (RSSI)

روش قدرت سیگنال دریافتی از تعدادی اکسس پوینت بی‌سیم IEEE ۸۰۲.۱۱ برای مکان‌یابی استفاده می‌کند. قدرت سیگنال دریافتی از سه اکسس پوینت برای کشف مکان شیء هدف استفاده می‌شود. برای افزایش دقت، در کاربرهای عملی از یک نقشه‌ی همپوشانی سیگنال‌ها با نام *اثر انگشت رادیویی*[۳۰] در منطقه‌ی از پیش تعیین شده، استفاده می‌شود.

در سیستم‌هایی که بر این اساس کار می‌کنند، فاصله‌ی بین هدف و اکسس پوینت با تبدیل مقدار قدرت سیگنال در هدف به معیار فاصله‌ی مبتنی بر قدرت سیگنال مشخص و نقشه‌ی پوشش و مدل مستخرج از آن حساب می‌شود. روش اندازه‌گیری اندازه در این مدل بر اساس قدرت سیگنال است، با این حال روش محاسبات کاملاً مشابه روش زمان دریافت انجام می‌شود. شکل ط نمونه‌ای از این مدل را نشان می‌دهد.




**تصویر ٦ – تشخیص مکان هدف با استفاده از RSSI**

روش *سرویس‌های مبتنی بر مکان* در گوشی‌های تلفن همراه که به عنوان جایگزینی برای GPS در نقاطی که دسترسی به آن مقدور نیست، بر پایه‌ی روش قدرت سیگنال دریافتی از ایستگاه‌های گیرنده/فرستنده است. گوشی‌های بلک‌بری سال‌ها است که این سرویس را به رایگان بر روی گوشی‌ها ارائه می‌دهند. دقت این سیستم‌ها بسته به نوع کاربرد و مدل استخراج شده متفاوت است. هر چه موانع بیشتر باشد، مدل (اثرانگشت) پوشش پیچیده‌تر می‌شود. در سرویس‌ها رایگان مبتنی بر

[۳۰] معادل عبارت Radio Fingerprint

مکان این دقت بسته به تعداد آنتن‌های BTS از چند متر تا چند صد متر است. این روش برای ردیابی تلفن‌های همراه نیز مورد استفاده است.

این روش برای این‌که مفید باشد، نیاز به تعداد زیادی اکسس پوین دارد که هزینه را به شدت بالا می‌برد. با این حال، مسئله‌ی اصلی در رابطه با این سیستم‌ها مدل مناسب *فقدان در مسیر* برای حالت‌هایی که خط دید مستقیم وجود ندارد و هم‌چنین شرایط مختلف (مثلاً جوی) است. نتیجتاً، در عمل فاصله‌های تخمین زده شده بسیار بی‌دقت هستند. عیب دیگر این سیستم‌ها این است که در برابر تهدیدهای امنیتی بسیار ضعیف هستند و یک مهاجم به راحتی می‌تواند با تقویت یا تضعیف سیگنال‌ها مکان‌یابی را به شدت دچار خطا کند.

### ۴-۲-۵- سایر روش‌ها

در کنار این متدولوژی‌ها، روش‌های دیگری مانند زمان پرواز[۳۱]، زمان رفت و برگشت[۳۲]، فاصله‌یابی متقارن دو طرفه‌ی رفت و برگشتی[۳۳] نیز مورد استفاده قرار می‌گیرد که نامرتبط با کاری است که در این پروژه انجام شده است و به همین دلیل به ارجاع آن ها بسنده می‌کنیم. پروژه‌ی انجام شده از روش TDoA و اصل چندجانبگی[۳۴] یا مکان‌یابی هذلولی[۳۵] استفاده می‌کند.

## ۴-۳- مسئله‌ی دوگان: موقعیت‌یابی گیرنده

یک گیرنده از تکنیک‌هایی مشابه روش‌های فوق می‌تواند برای موقعیت‌یابی خودش استفاده کند. مثلاً با اندازه‌گیری TDoAسیگنال‌های انتشار یافته از سه یا بیشتر فرستنده‌ی هم‌زمان‌شده در مکان‌های معلوم و به کار گیری منطق دوگان مطرح‌شده موقعیت خودش را به دست آورد. سیستم ناوبری DECCA که در جنگ جهانی دوم توسط بریتانیا استفاده می‌شد، با محاسبه‌ی اختلاف فاز فرستنده‌ها مکان‌یابی را انجام می‌داد.

---

[۳۱] معادل عبارت Time of Flight (ToF)

[۳۲] معادل عبارت Round Trip Time (RTT)

[۳۳] معادل عبارت Symmetric Double Sided Two Way Ranging

[۳۴] معادل عبارت Principle of Multilateration

[۳۵] معادل عبارت Hyperboloid Positioning

## ۴-۴- معرفی چندجانبگی

تکنیک‌های چندجانبگی برای مدت تقریباً طولانی است که برای نظارت در فرودگاه‌ها استفاده می‌شود. امروزه، این تکنیک در طول پرواز و هنگام فرود نیز استفاده می‌شود. به این سیستم‌ها، *سیستم چندجانبگی در فضای گسترده*[۳۶] گفته می‌شود.

چندجانبگی نوعی از *نظارت مستقل مشارکتی*[۳۷] است که از سیگنال‌هایی که توسط یک پرنده ارسال می‌شود، برای محاسبه‌ی مکان پرنده استفاده می‌شود. از آن‌جایی که سیستم‌های چندجانبگی می‌توانند از سیگنال‌هایی که پیشتر به دلایل دیگر از هواپیماها ارسال می‌شوند، استفاده کنند، سیستم‌های WAM را می‌توان بدون هیچ تغییری در زیرساخت هوابرد به کار گرفت.

**برای پردازش سیگنال‌ها، ایستگاه‌های دریافت‌کننده‌ی مناسب و یک ایستگاه پردازش مرکزی مورد نیاز است. لازم به ذکر است که بر اساس سیگنال‌هایی که مورد استفاده قرار می‌گیرند، استفاده از تجهیزات خاصی ضروری خواهد بود.**

## ۴-۵- معماری سیستم‌های WAM

سیستم‌های WAM را می‌توان با استفاده از دو معیار جدا دسته‌بندی کرد. اول، می‌توان روش مورد استفاده برای محاسبه‌ی تفاوت زمان‌های دریافت (TDoA) سیگنال‌ها و دوم روش هم‌زمان‌کردن گیرنده‌ها را معیاری برای رده‌بندی قرار داد. در ادامه این روش‌ها را از نظر می‌گذرانیم.

### ۴-۵-۱- روش‌های محاسبه‌ی TDoA

دو روش برای محاسبه‌ی اختلاف زمان دریافت‌ها وجود دارد. یا سیگنال‌های دریافتی با استفاده از همبستگی متقابل[۳۸] تحلیل می‌شوند و TDoA آن‌ها حساب می‌شود، یا زمان رسیدن (ToA) اندازه‌گیری می‌شود و با استفاده از آن، اختلاف‌ها محاسبه می‌شود.

روش ToA معمولاً با شکل موج‌هایی استفاده می‌شود که محاسبه‌ی لبه‌ی پالش مشخصی آسان است، مثلاً سیگنال‌های رادار پایش ثانویه (SSR) هواپیماها. روش همبستگی متقابل را می‌توان با هر سیگنالی استفاده کرد، اما مناسب بودن در این حالت متناسب است با خصوصیات همبستگی متقابل

---

[۳۶] معادل عبارت Wide Area Multilateration (WAM)

[۳۷] معادل عبارت Cooperative Independent Surveillance

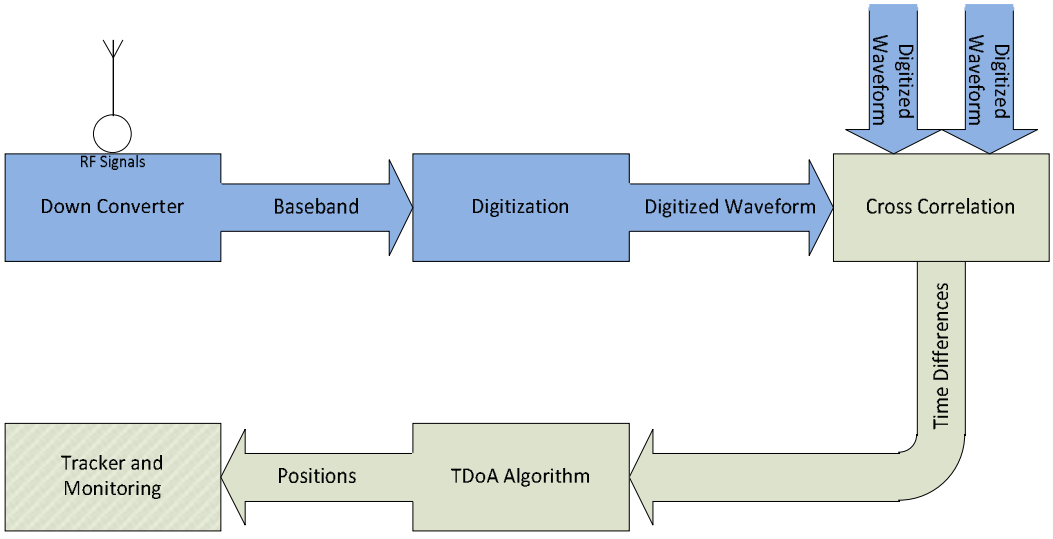
[۳۸] معادل عبارت Cross Correlation. که معیاری از شباهت دو سیگنال به عنوان تابعی از تاخیر زمانی اعمال شده به یکی از آن‌ها است

سیگنال با خودش[۳۹]. روش ToA روش متداولی در چندجانبگی برای سیگنال‌های SSR است. در ادامه دو روش را شرح می‌دهیم.

### ٤-٥-١-١- سیستم‌های همبستگی متقابل

روش همبستگی متقابل معمولاً در سیستم‌های الکترونیکی اندازه‌گیری پایش نظامی[۴۰] (ESM) و مکان‌یابی تلفن‌های همراه به هنگام تماس‌های اضطراری استفاده می‌شود. دیاگرام زیر جریان اطلاعات را به صورت ساده در این سیستم‌ها نشان می‌دهد.

در این شکل سیگنال دریافتی پس از تبدیل فرکانسی در Down Convertor توسط یک مدل سیگنال آنالوگ به دیجیتال، رقمی می‌شود. پس از دیجیتال شدن، در گام همبستگی متقابل مجموعه‌ای از محاسبات همبستگی بین جفت‌های سایت‌ها انجام می‌شود. با فرض این‌که سیگنال مشخصی، در شکل موج دریافتی توسط همه‌ی سایت‌ها وجود دارد، نتیجه‌ی محاسبات مقادیر TDoA را بین سایت‌های مختلف می‌دهد. دقت این روش به فاکتورهای متعددی من جمله نوع سیگنال بستگی دارد. الگوریتم‌هایی بایستی به کار گرفته شوند که مقادیری را که مبهم هستند یا از سیگنال‌هایی هستند که ذاتاً خصوصیات مناسبی برای همبستگی متقابل و خود همبستگی ندارند، حذف کنند.



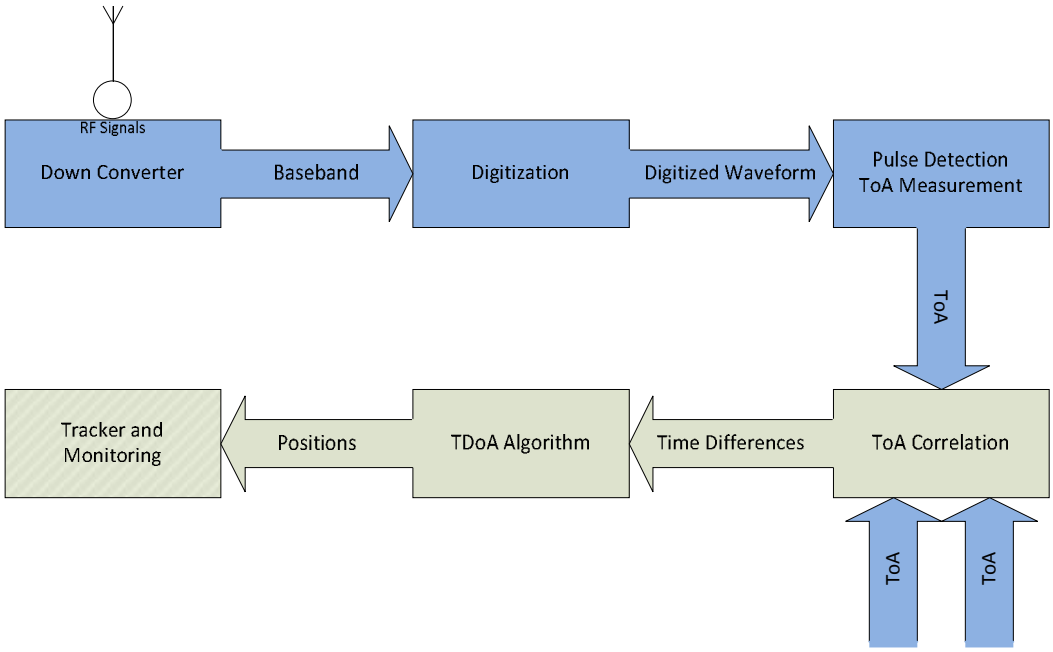

**تصویر ۷ – جریان اطلاعات در سیستم همبستگی متقابل**

[۳۹] معادل Auto Correlation

[۴۰] معادل Military Electronic Surveillance Measure System

با داشتن مقادیر TDoA، یک الگوریتم ساده (با رایاضیاتی که پیشتر شرح دادیم استفاده می‌شود تا مکان هدف را مشخص کند. در نهایت این سیستم به یک *دنبال کننده* متصل می‌شود که اطلاعات مسیر رانگهداری می‌کند و به این ترتیب داده‌های ناصحیح و خطادار در یک سطح دیگر نیز اصلاح می‌شوند.

سیستم همبستگی متقابل اساساً با یک سیستم ToA (که در بخش بعدی توضیح می‌دهیم) فرق دارد، چرا که زمان واقعی رسیدن سیگنال به هر گیرنده هیچ‌گاه محاسبه نمی‌شود و فقط مقادیر TDoA بدست می‌آیند.




**تصویر ۸ - جریان اطلاعات در سیستم زمان دریافت**

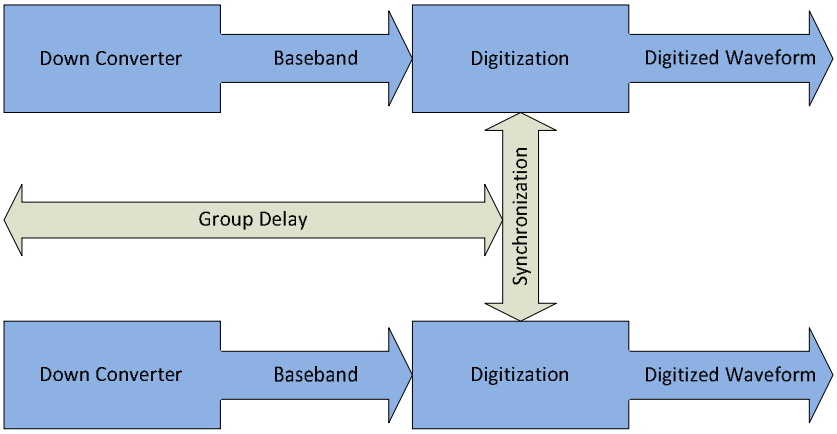
### ٤-٥-١-٢- سیستم‌های زمان دریافت (ToA)

روش ToA برای مکان‌یابی در سیستم‌های چندجانبگی که از سیگنال رادار پایش ثانویه پرنده استفاده می‌کنند به صورت گسترده‌ای مورد استفاده است. شکل زیر جریان داده را در چنین سیستمی نشان می‌دهد.

همانند روش قبل تبدیلات فرکانسی و دیجیتال کردن به همان منوال انجام می‌شود. پس از دیجیتال شدن، یک سیستم تشخیص، زمان‌های دریافت را به صورت محلی در هر گیرنده حساب می‌کند. سپس این زمان‌ها دو به دو با هم مقایسه شده و اختلاف‌های زمانی را می‌سازند. الگوریتم TDoA و دنبال کننده مانند روش همبستگی متقابل عمل می‌کنند.

### ۴-۵-۲- روش هم‌زمان‌سازی

نیاز به هم‌زمان‌سازی در هر دو روش همبستگی متقابل و زمان دریافت (و روش‌های دیگر مکان‌یابی که قبلاً اشاره شد) با این وجود که روش‌های متعددی برای هم‌زمان‌سازی وجود دارد، بسیار اساسی است. برای محاسبه‌ی مکان، ضروری است تا اختلاف زمان دریافت رسیدن سیگنال به یک آنتن را با یک آنتن دیگر از سیستم بدانیم. با این حال، زمان سیگنال در طول فاز دیجیتال‌سازی به صورت زمان‌مهر[۴۱] در آن قرار می‌گیرد که تاخیر زمانی ناشی از تبدیل فرکانس (*تاخیر گروهی*[۴۲]) نیز در آن اتفاق افتاده است. به این ترتیب برای محاسبه‌ی دقیق TDoA این زمان را باید به صورت دقیق بدانیم. به علاوه، زمان در حین فرآیند دیجیتال‌کردن بایستی در سایت‌های مختلف مقدار مشترکی باشد، در غیر این صورت سایت‌ها ساعت‌هایی با زمان متفاوت خواهند داشت و مقادیر زمان مهر قابل مقایسه نخواهد بود.




**تصویر ۹ – هم‌زمان‌سازی و تاخیر گروهی**

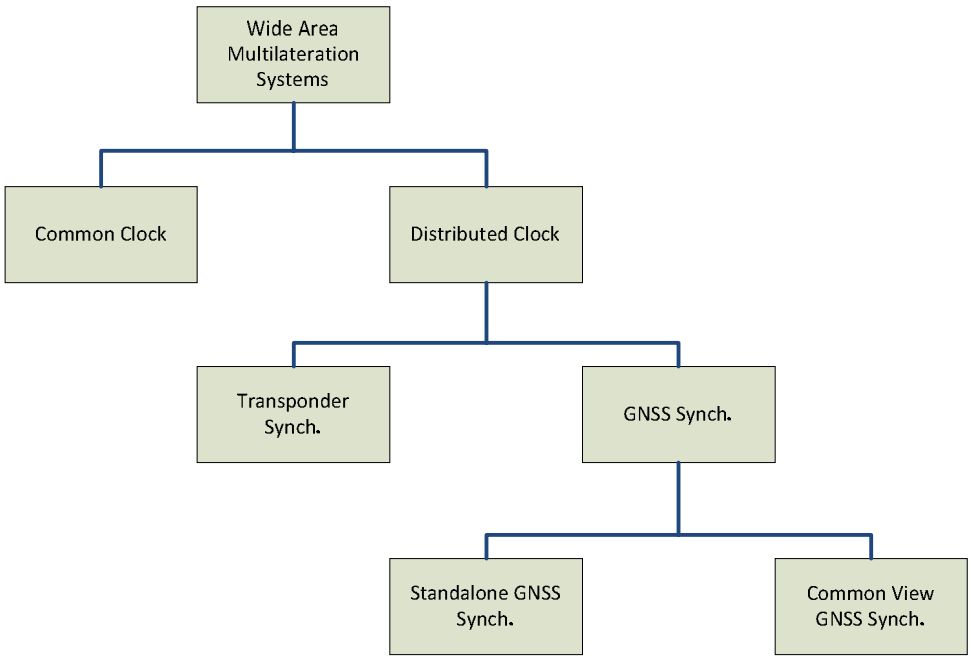
شکل ط تاخیر گروهی و المان‌های هم‌زمان‌سازی را نشان می‌دهد. هم‌زمان‌سازی روشی است که بر مبنای آن فرآیند دیجیتال‌کردن در سایت‌های مختلف با هم یکی می‌شوند.

دیاگرام زیر توپولوژی تکنولوژی‌های مختلفی را که برای هم‌زمان‌سازی مورد استفاده قرار می‌گیند، نشان می‌دهد. این روش‌ها را در ادامه به طور مختصر توضیح می‌دهیم.

---

[۴۱] Timestamp معادل کلمه

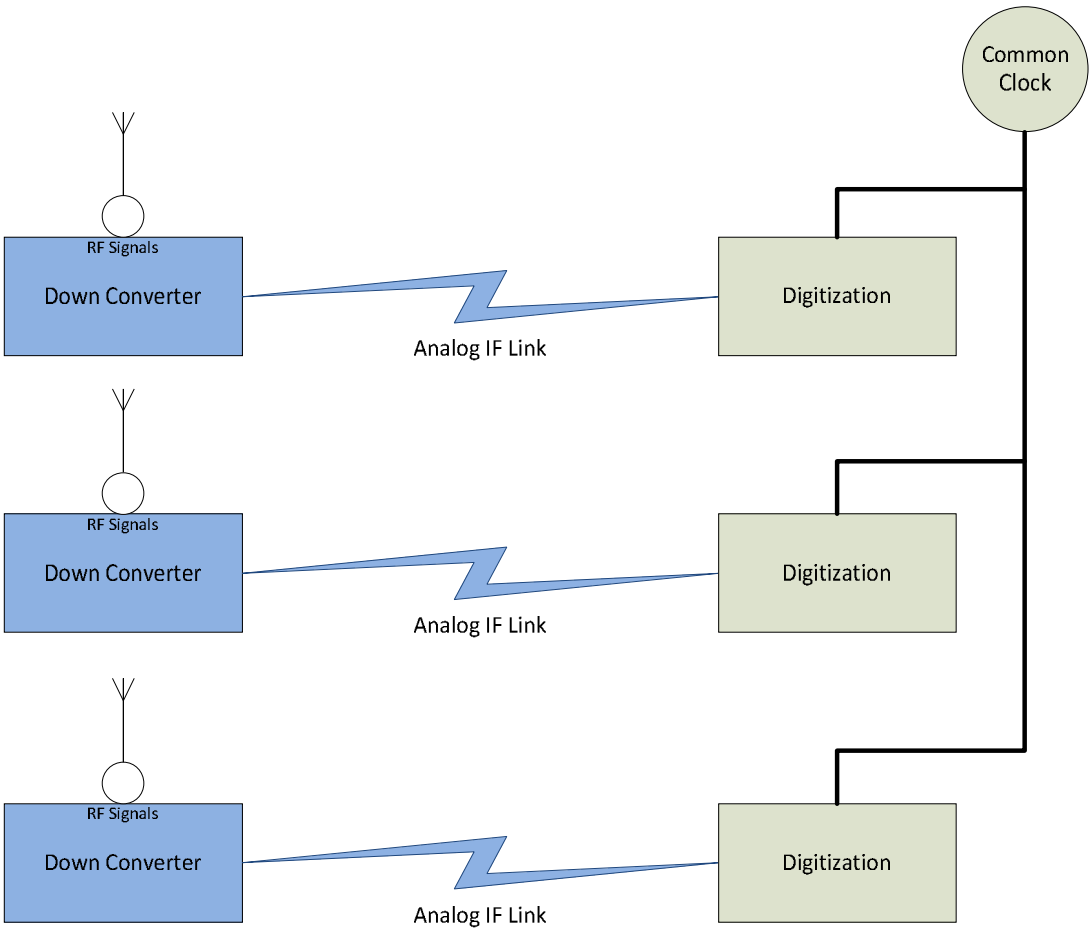
[۴۲] Group Delay معادل عبارت




**تصویر ۱۰ – توپولوژی هم‌زمان‌سازی**

### ٤-٥-٢-١- سیستم‌های با ساعت مشترک

سیستم‌های با ساعت مشترک از گیرنده‌های ساده‌ای استفاده می‌کنند و بشتر پیچیدگی کار را در واحد پردازش انجام می‌دهند. این سیستم‌ها سیگنال‌های فرکانس رادیویی را از پرنده دریافت می‌کنند و به یک فرکانس میانی تبدیل می‌کنند. این سیگنال با فرکانس میانی با استفاده از یک لینک آنالوگ به سایت مرکزی ارسال می‌شود. سایت مرکزی متصل به یک ساعت مشترک برای همه‌ی گیرنده‌ها است. با این معماری، نیازی نیست که تک تک گیرنده‌های اصلی را هم‌زمان کنیم، چرا که دیجیتال‌سای در سایت اصلی اتفاق می‌افتد. با این حال تاخیر گروهی بین دریافت سیگنال در آنتن تا دیجیتال‌سازی در سایت مرکزی طولانی است، چرا که تاخیر لینک آنالوگ میانی نیز در آن لحاظ می‌شود. این تاخیر بایستی به طور بسیار دقیق برای هر گیرنده مشخص باشد. این مسئله به این معنی است که هم زنجیره‌ی گیرنده و هم لینک‌های داده بایستی کاملاً برای حصول مقدار مشخصی تاخیر گروهی کالیبره شوند. هر چه تاخیر لینک افزایش یابد، معمولاً به دلیل افزایش فاصله یا شرایط سیستم، رسیدن به دقت پی‌فرض دشوارتر خواهد بود. بنابراین اگر، به عنوان مثال، تاخیرها را با دقت ۱٪ بدانیم و دقت ساعت ۱نانوثانیه باشد، تاخیر ۱۰۰نانوثانیه قابل تحمل است، ولی ۲۰۰نانوثانیه نیست. این روابط در شرایط مختلف برای سیستم متفاوت خواهد بود.



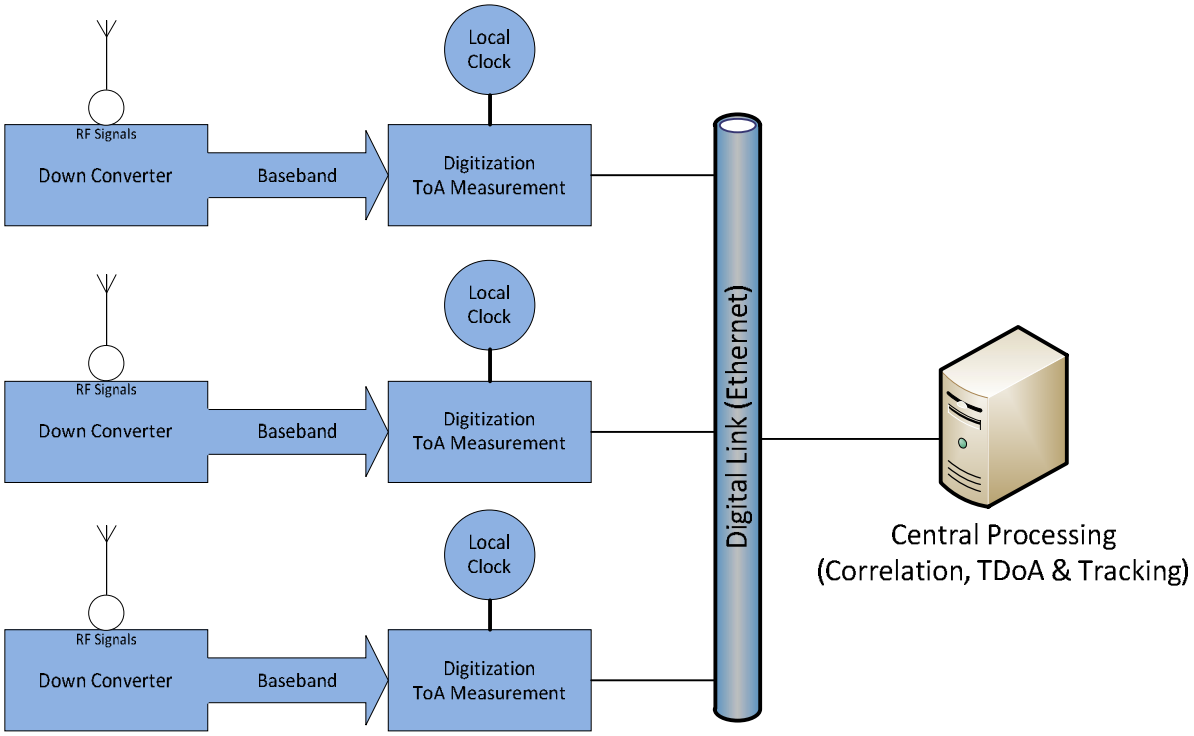

**تصویر ۱۱ – معماری ساعت مشترک**

این معماری از مزیت گیرنده‌های ساده و مصرف توان اندک و تمرکز پیچیدگی‌ها در سایت مرکزی برخوردار است. با این حال تاخیر سیگنال بین آنتن و پردازنده شرایط سخت‌گیرانه‌ای را روی نوع و برد لینک‌ها تحمیل می‌کند. معمولاً، از یک لینک مایکروویو یا فیبر نوری بین سایت‌ها استفاده می‌کنند. برای کمینه‌کردن فواصل لینک، سایت پردازنده بایستی تقریباً در مرکز توپولوژی قرار بگیرد. این معماری در محصول MSS توسط Era از زیرمجموعه‌های شرکت معتبر SRA International برای سیستم‌های مکان‌یابی هوابرد مورد استفاده قرار می‌گیرد.

### ٤-٥-٢-٢- سیستم‌های با ساعت توزیع‌شده

سیستم‌های با ساهت توزیع‌شده از گیرنده‌های پیچیده‌تری استفاده می‌کنند که ما را از تحمیل شرایط دشوار لینک رها می‌کند. دریافت سیگنال رادیویی، تبدیل فرکانسی، دیجیتال‌کردن، استخراج کد (در حالت امکان احراز هویت در سیستم‌هایی که از SSR استفاده می‌کنند کد Mode A تحلیل می‌شود) و ToA در صورت نیاز، همگی در گیرنده انجام می‌شود و سپس به سایت مرکزی ارسال

می‌شود. در این حالت از هر لینک دیجیتالی می‌توان استفاده کرد و تاخیر حیاتی نیست و نقشی در دقت سیستم ندارد (صرفاً در تاخیر کشف موقعیت می‌تواند موثر باشد).




**تصویر ۱۲ - هم‌زمان‌سازی با معماری ساعت توزیع‌شده**

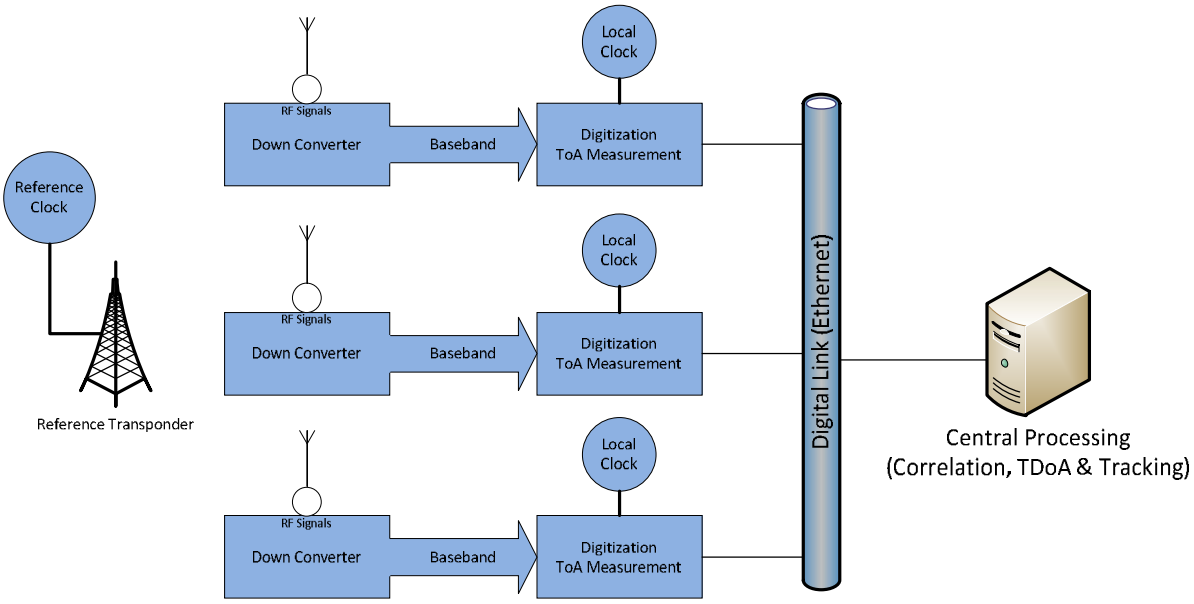
با این حال، بایستی از مکانیزمی استفاده شود تا ساعت‌های محلی سایت‌ها با هم هم‌زمان باشند. این معماری، متداول‌ترین نوع معماری برای سیستم‌های چندجانبگی است و شرکت‌هایی چون Rannoch، Roke Manor Reaserch و Sensis از این معماری استفاده می‌کنند.

### ٤-٥-٢-٣- سیستم‌های هم‌زمان‌شده به وسیله‌ی فرستنده

سیستم‌های هم‌زمان‌شده توسط فرستنده، از سیگنال‌های ارسال‌شده توسط یک فرستنده‌ی مرجع برای هم‌زمانی ساعت‌های محلی در گیرنده‌ها استفاده می‌کنند. سیگنال زمانی مرجع و سیگنال‌های ارسال‌شده توسط پرنده، از یک زنجیره‌ی یکسان دریافت آنالوگ عبور می‌کنند، به این معنی که بایاس تاخیر ناشی از اجزای آنالوگ خنثی می‌شود. این روش اجازه‌ی ساخت یک سیستم دقیق کوچک را می‌دهد.

در سیستم‌های گسترده‌تر تاخیرهای جوی باعث کاهش دقت می‌شوند. نیازی نیست که ارسال‌کننده‌ی مرجه با واحد پردازش مرکزی هم‌مکان باشد، ولی باید در خط دید مستقیم گیرنده‌ها

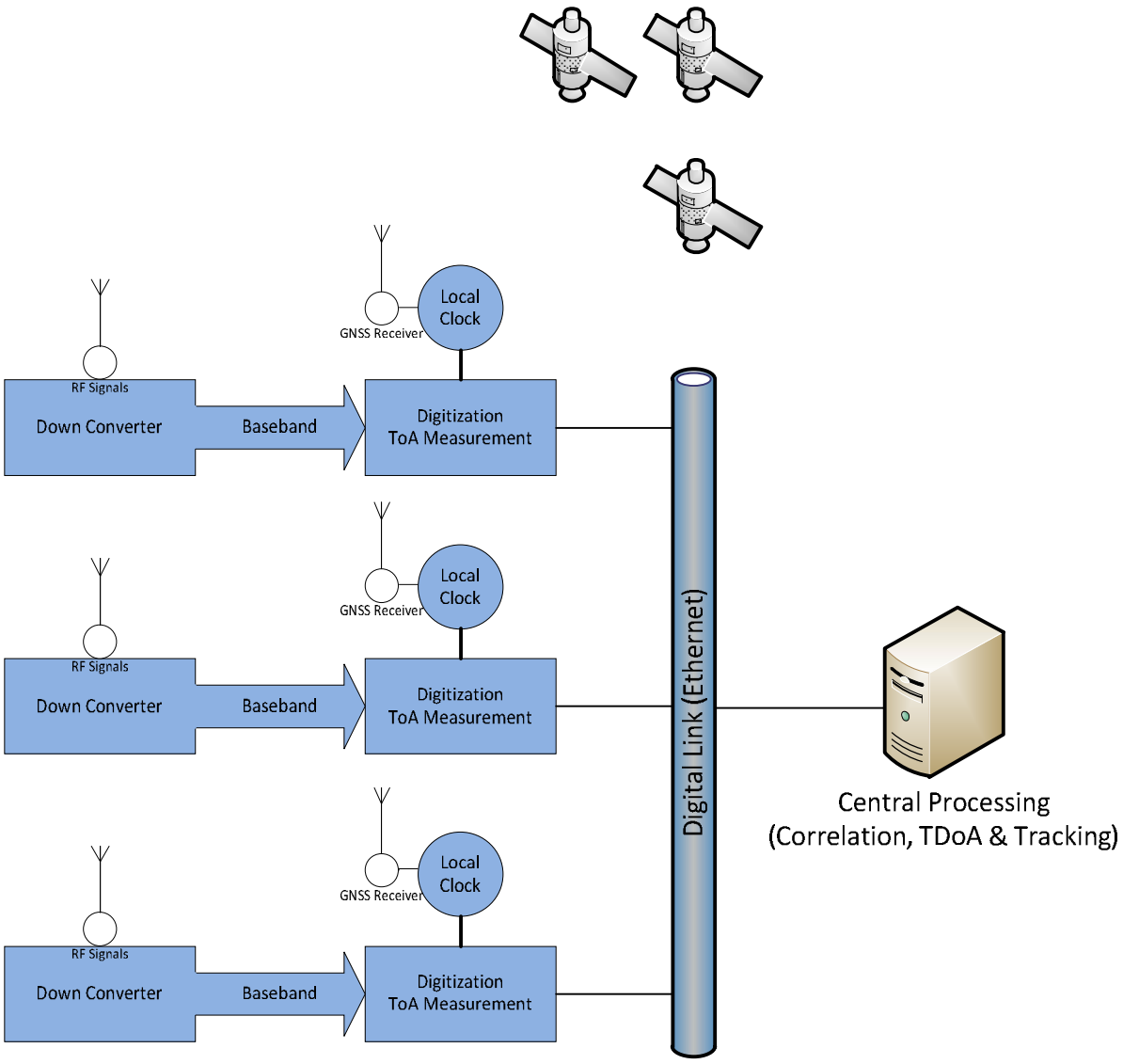
قرار داشته باشد. بر همین مبنا می‌توان سیستم‌های گسترده‌تر هم‌زمانی را نیز فراهم کرد. شرکت‌های Roke Manor Research و Sensis از این روش برای هم‌زمانی استفاده می‌کنند.




**تصویر ۱۳ – معماری هم‌زمانی با فرستنده**

### ٤-٥-٢-٤- سیستم‌های هم‌زمان‌شده با GNSS مستقل

معمولاً می‌توان در مدل قبل از یک مرجع زمانی مشترک مانند سیستم ماهواره‌ای ناوبری جهانی (GNSS) استفاده کرد تا یک مرجع مشترک برای گیرنده‌ها فراهم کرد. زمان‌های سیستم GNSS بسیار دقیق نگه‌داشته می‌شوند، چرا که برای دقت در مسیریابی و ناوبری بسیار ضروری است. برای مثال، مجموعه ماهواره‌های GPS دقت زمانی در حد ۱۰۰نانوثانیه UTC دارند. از این زمان می‌توان به عنوان مرجع مشترک گیرنده‌ها استفاده کرد.در سیستم های چندجانبگی، تنها زمانی که مهم است، اختلاف زمان بین دریافت‌ها است و زمان دقیق اهمیتی ندارد، به این ترتیب می‌توان ساعت‌های گیرنده‌ها را در حدود ۱۰ تا ۲۰نانوثانیه با استفاده از یک نوسان‌سازی GPS هم‌زمان کرد. ساخت چنین سیستم‌هایی به نسبت سیستم‌های هم‌زمان‌شده به وسیله‌ی فرستنده بسیار ساده‌تر است، چرا که نیازی به فراهم کردن ساعت مرجع با شرایطی که پیش‌تر ذکر آن رفت، نیست. شکل زیر معماری چنین سیستمی را نشان می‌دهد. شرکت Rannoch سیستم‌هایی با این معماری تولید می‌کند.

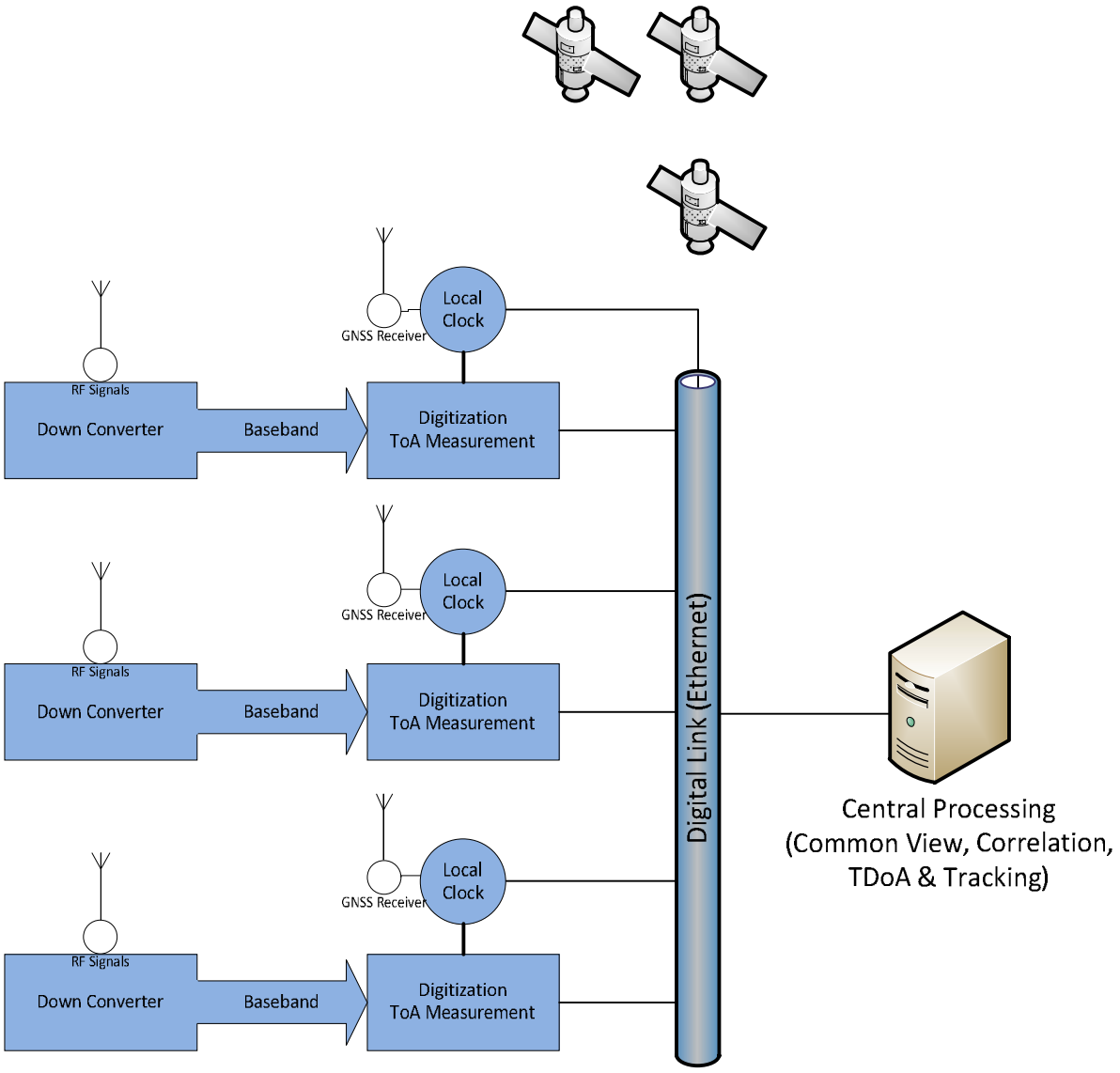



**تصویر ١٤ - معماری هم‌زمانی با GNSS**

### ٤-٥-٢-٥- سیستم‌های هم‌زمان‌شده با GNSS دید مشترک

برای شرایطی که هم‌زمان‌سازی GNSS مستقل بین گیرنده‌ها به قدر کافی دقیق نیست، از یک سیستم هم‌زمان‌سازی دید مشترک می‌توان استفاده کرد. سیستم‌های دید مشترک از ماهواره‌های GNSS که در دید گیرنده هستند، استفاده می‌کنند و داده‌های تفاضلی را محاسبه می‌کنند. این روش اجازه می‌دهد تا مقدار زیادی از منابع خطا حذف شوند، چرا که بین سیگنال‌ها مشترک هستند و بنابراین سیستم هم‌زمانی دقیق‌تری بدست می‌آید. دقت‌های حدود نانوثانیه را می‌توان با این روش به دست آورد. با استفاده از پردازش دید مشترک و روش‌های صحت‌سنجی مانند *دیده‌بانی* صحت

خودکار *گیرنده‌ها* می‌توان یکپارچگی سیستم را نیز سنجید. شرکت Roke Manor Research از این معماری برای رده‌ای از سیستم‌های چندجانبگی استفاده می‌کند.




**تصویر ۱۵ – معماری هم‌زمانی GNSS با دید مشترک**

### ۴–۵–۳– نتیجه‌گیری روش‌های هم‌زمانی

جدول زیر خصوصیات روش‌های مختلف هم‌زمانی را خلاصه می‌کند. باید توجه داشت که این نتیجه‌گیری حاصل مطالعات نگارنده و نه تجربه‌های میدانی است.

جدول ۱ – مقایسه‌ی روش‌های هم‌زمانی

| | دقت | محدوده | لینک | دکل | خط دید |
|---|---|---|---|---|---|
| **ساعت مشترک** | متوسط | متوسط | فیبر | – | – |
| **هم‌زمانی با فرستنده** | متوسط | متوسط | دلخواه | بلند | بله |
| **ساعت GNSS استاندارد** | کم | دلخواه | دلخواه | – | – |
| **ساعت GNSS با دید مشترک** | زیاد | وسیع | دلخواه | – | – |

## ۴–۶– خصوصیت بلادرنگی در سیستم‌های RTLS

همان‌طور که در معماری‌های فوق دیدیم، یکی از بخش‌های کلیدی در همه‌ی معماری‌ها (جز حالت ساعت مشترک) ارسال داده از گیرنده‌ها به سایت پردازنده است. همان‌طور که در روش‌های محاسبه‌ی اختلاف زمانی توضیح دادیم، این داده می‌تواند زمان دریافت (ToA) یا خود سیگنال دیجیتال شده باشد، که به هدف اعمال همبستگی همبستگی متقابل ارسال می‌شود.

در تعریف سیستم‌های مکان‌یابی بلادرنگ توضیح دادیم که مکان‌یابی باید در یک بازه‌ی زمانی از پیش تعیین‌شده (مهلت زمانی) انجام شود. این مفهوم این سیستم‌ها را در رده‌ی سیستم‌های بلادرنگ قرار می‌دهد. خصوصیت بلادرنگی با توجه به معماری‌های ارائه شده در نحوه‌ی مدیریت لینک رابط مابین گیرنده‌ها و سایت مرکزی تجلی می‌یابد، بنابراین پروتکل و روش مدیریت این لینک‌ها بایستی خصوصیت بلادرنگی را ارضا کند. در ادامه ابتدا به ارائه‌ی جنبه‌های مختلف ارتباطات بلادرنگ پرداخته و سپس بیان می‌کنیم که در این پروژه این خصوصیات را چگونه فراهم کردیم.

به لحاظ تئوری به ارتباطی بلادرنگ می‌گوییم که مشتری‌ها در آن بتوانند نیازمندی‌های کارایی خاصی را مشخص کنند و فراهم شدن آن نیازمندی‌ها توسط سیستم برای ایشان تضمین شود.

معماری شبکه معمولاً متشکل است از پشته‌ای از پروتکل‌های مختلف. در رابطه با یک چنین معماری‌ای، مفهوم مشتری و سرویس‌دهنده یک زوج از موجودیت‌های درگیر در سیستم هستند. در این مدل هر لایه از لایه‌ی زیرینش سرویس می‌گیرد و سرویسی به لایه‌ی بالاترش ارائه می‌کند. در این قسمت، منظور از مشتری و سرویس‌دهنده هر دو جزئی است که چنین خاصیتی داشته باشند. با این حال، عموماً منظور از این عبارات کاربر انسانی است.

## ۴-۷- نیازمندی‌های کارایی

یک مشتری ممکن است عبارتی بولینی مانند $pred$ را معیار تعریف و انتظار از کارایی قرار دهد که گزاره‌هایی در این عبارت متاثر یا در کنترل سرور هستند. مثال واضحی از این نوع تعریف کارایی یک حد آستانه است. برای مثال در یک تعریف ممکن است داشته باشیم:

$$(var \leq bound) = pred = TRUE$$

که مثلاً متغیر var یک متغیر در کنترل سرویس‌دهنده و bound مقداری است که مشتری مشخص می‌کند. مقدار فوق را *حدآستانه‌ی بالا* می‌نامیم. کافی است جهت نامساوی را تغییر دهیم تا حد آستانه‌ی پایین مشخص شود.

گاهی اوقات ممکن است، عبارتی مشابه عبارت بالا به صورت احتمالی بیان شود. برای مثال:

$$\mathcal{P}(var \leq bound) \geq prob.bound$$

نوعی دیگر از محدوده که بسیار شبیه نوع محدوده‌های احتمالی است که در بالا ذکر شد، محدوده‌ی کسری است، به این معنی که چه تعداد از مثلاً پیغام‌ها، پردازش‌ها، بسته‌ها و ... این شرایط را ارضا می‌کنند:

$$C_A(var \leq bound) \geq B$$

عبارت بالا به عنوان نمونه، بیان می‌کند که تعداد ارضا شدن شرط داخل پرانتز برای A پیغام پشت هم کمتر از B نباشد. پر واضح است که در این مثال بایستی B کمتر از A باشد.

در ادامه، تمرکزمان بیشتر بر نیازمندی‌هایی است که برای کاربردهای بلادرنگ مورد نیازند.

### ۴-۷-۱- تاخیر

بسته به نوع کاربرد، ممکن است نیاز داشته باشیم تا تاخیر را به اشکال گوناگون بیان کنیم. عموماً تاخیر بر اساس پیغام‌هایی بیان می‌شود که از طرف مشتری شناخته شده هستند. به عبارت دیگر، و به عنوان یک مثال تاخیر در رابطه با پیغام‌هایی معنی پیدا می‌کند که به گیرنده تحویل می‌شود. در رابطه با سایر پیغام‌ها مسئله‌ای که مطرح می‌شود، قابلیت اتکا است و نه تاخیر. نکته‌ی قابل توجه دیگر در رابطه با تاخیر در محیط‌های توزیع‌شده، بحث همگامی است. چرا که عموماً گیرنده تاخیر را در رابطه با ساعت محلی‌اش در نظر می‌گیرد. برخی از انواع بیان نیازمندی تاخیر به شرح زیر است:

۱- محدوده‌ی قطعی تاخیر. عبارت $D_i \leq D_{max}$ این نوع تاخیر را بیان می‌کند که در آن $D_i$ تاخیر دریافت $i$ امین پیغام است و $D_{max}$ حداکثر تاخیر مجاز شناخته شده توسط کاربر.

۲-محدوده‌ی احتمالی تاخیر. عبارت $\mathcal{P}(D_i \leq D_{max}) \geq Z_{min}$ این نوع تاخیر را بیان می‌کند که در آن $D_i$ تاخیر دریافت $i$ امین پیغام است و $D_{max}$ حداکثر تاخیر مجاز شناخته شده توسط کاربر و $Z_{min}$ کمینه‌ی احتمال مورد قبول است.

۳-محدوده‌ی قطعی تاخیر - ناپایداری. اگر ناپایداری تاخیر را به صورت $J_i = |D_i - D|$ درنظر بگیریم، آن‌گاه این نوع محدوده‌گذاری روی ناپایداری به شکل $J_i \leq J_{max}$ خواهد بود که در آن $D$ تاخیر ایده‌آل، $J_i$ ناپایداری در پیغام $i$ ام و $D_i$ تاخیر پیغام $i$ ام است.

٤-محدوده‌ی احتمالی تاخیر - ناپایداری. مشابه حالت محدوده‌ی احتمالی برای تاخیر، این حالت نیز اعمال یک شرط کمینه‌ی احتمال برای عبارت ناپایداری است، به این ترتیب این محدوده به شکل $\mathcal{P}(J_i \leq J_{max}) \geq U_{min}$ خواهد بود.

اشکال دیگری از قیود بر روی تاخیر، محدود کردن تاخیر میانگین، واریانس تاخیر و توابعی از شماره ترتیب‌های پیغام‌ها است.

### ۴-۷-۲- گذردهی

گذردهی واقعی یک انتقال اطلاعات از یک منبع به یک مقصد متناسب است با نرخی که فرستنده با آن نرخ پیغام‌ها را به سیستم تزریق می‌کند. ممکن است به دلیل رخ دادن خرابی، گم شدن بسته‌ها و نیاز به ارسال مجدد، گذردهی واقعی کمتر از این میزان باشد. از طرف دیگر گذردهی تابعی از بار شبکه نیز هست. وقتی فرستنده نرخ ارسال را افزایش می‌دهد، گذردهی تا جای مشخصی افزایش می‌یابد و سپس متوقف می‌شود. از این نقطه به بعد، گذردهی ثابت می‌ماند و با افزایش نرخ ارسال از طرف فرستنده، گذردهی مفید کاهش می‌یابد. وقتی نیازمندی گذردهی در رابطه با استفاده‌ی بلادرنگی تبیین می‌شود، نیاز است تا اطمینان حاصل کنیم که هیچ‌گاه سیستم به مرز اشباع نمی‌رسد، یا اگر برسیم، برای مدت کوتاهی در این حالت باقی می‌مانیم. این مسئله هنگامی که پهنای باند ثابت نیست شکل پیچیده‌تری پیدا می‌کند. با این حال، اشکال متداول بیان نیازمندی‌های گذردهی به شکل زیر است:

۱-محدوده‌ی قطعی گذردهی. اگر $\theta_i$ را گذردهی فراهم شده و $\theta_{min}$ حداقل گذردهی شرط شده باشد، محدوده‌ی گذردهی با $\theta_i \geq \theta_{min}$ بیان می‌شود.

۲-محدوده‌ی احتمالی گذردهی. با اعمال تابع احتمال به عبارت فوق $\mathcal{P}(\theta_i \geq \theta_{min}) \geq Y_{min}$ بیانگر محدوده‌ی احتمالی برای گذردهی است.

۳-محدوده‌ی میانگین گذردهی. اگر $\bar{\theta}$ را میانگین گذردهی در نظر بگیریم و $\theta_{avg}$ میانگین مورد انتظار گذردهی باشد و هر دو متغیر طی بازه‌ی یکسانی محاسبه شده باشند، عبارت $\bar{\theta} \leq \theta_{avg}$ این نوع بیان محدوده را نشان می‌دهد.

یک تفاوت آشکار بین تاخیر و گذردهی به عنوان یک نیازمندی بلادرنگ این است که با این که عموماً سرویس‌دهنده مسئول تاخیر است، در یک سیستم غیراشباع، گذردهی توسط مشتری‌ها تعیین می‌شود.

### ۴-۷-۳- قابلیت اطمینان

اینکه کنترل خطا با استفاده از تصدیق و ارسال مجدد در کاربردهای بلادرنگ سودمند است یا نه محل اشکال و ایراد بسیاری از متخصصان است. از طرف دیگر تاخیر اضافی ناشی از تصدیق و ارسال مجدد در بسیاری از کاربردها قابل تحمل نیست. خوشبختانه، خرابی و گم شدن تعدادی از بسته‌ها در اغلب این کارکردها تا حدی قابل قبول است.

یک پیغام ممکن است به صورت خراب به دست گیرنده برسد، یا اصلاً در شبکه گم شده و هیچ‌گاه تحویل نشود. عموماً عدم قابلیت اطمینان در زیرساخت شبکه (مثلاً نویز در شبکه‌های بی‌سیم) منشا نوع اول خرابی است در حالی که نوع دوم عموماً ناشی از ازدحام شبکه و سرریز بافرها است. از نقطه نظر گیرنده اما، تمایزی بین این دو حالت وجود ندارد. در هر دو حالت پیغام از دست رفته است. از این رو ساده‌ترین و عمومی‌ترین نوع برخورد با این نیازمنده به شکل زیر است:

$$\mathcal{P}(message\ is\ correctly\ delivered) \geq W_{min}$$

اگر مفهوم تاخیر را نیز دوباره دخیل کنیم، در کاربردهایی (مانند همین کاربرد مکان‌یابی) که پیغام‌هایی که بعد از مهلت مشخص می‌رسند، قابل استفاده نیستند، پیغام‌های قابل استفاده عبارتند از $W_{min}Z_{min}$. ممکن است در نوعی از تعریف شرایط، مشتری تنها میزان این حاصل ضرب را انتخاب کند و انتخاب در رابطه با خرابی و تاخیر را به عهده‌ی سرویس‌دهنده بگذارد.

### ۴-۷-۴- دنباله‌سازی

برای برخی کاربردها که شامل جریانی از پیغام‌ها هستند (به جای یک دیتاگرام منفرد)، احتمالاً لازم است که پیغام‌ها به ترتیب برسند، حتی با این فرض که دنباله‌ها کامل نیستند. اگر سرویس‌دهنده‌های لایه‌های پایین‌تر، قابلیت تحویل پیغام‌ها را به صورت مرتب ندارند، ترتیب باید در لایه‌های بالاتر تضمین شود. در حالاتی که قیود قابلیت اتکا ما را مجبور می‌کنند که ارسال مجدد

بسته‌ها را به کار بگیریم، ارائه‌ی ترتیب عموماً منجر به معطلی بسیاری از بسته‌ها می‌شود. معمولاً با استفاده از صف و بافر، فراهم کردن ترتیب کار دشواری نیست.

### ٤-٧-٥- نبود تکرار

اغلب جنبه‌هایی که در رابطه با ترتیب بسته‌ها از نظر گذراندیم در رابطه با نبود تکرار در بسته‌های دریافتی نیز قابل اعمال است. با این حال، فراهم کردن عدم تکرار عموماً بسیار ساده‌تر از فراهم کردن دنباله‌ی مرتب از بسته‌های ارسالی است. بدیهی است که ملاحظات این بخش برای حالاتی که تکرار به دلیل اولویت بالاتر است قابل اعمال نیست.

### ۴-۷-۶- بازیابی از خطا

تعریف یک سیستم بلادرنگ بایستی به وضوح مشخص کند که به هنگام رخ دادن خطا چه اتفاقی می‌افتد. به صورت ایده‌آل، از دید مشتری، خرابی بایستی به شکل کامل پنهان شود و سرویس بایستی به صورت کاملاً عاری از خطا و خرابی ارائه شود. همان‌طور که پیشتر بیان شد، این انتظار غیرواقعی خواهد بود. به طور کلی مشتری عموماض یک احتمال خرابی را تحمیا خواهد کرد.

$$\mathcal{P}(failure) \leq F_{max}$$

قاعدتاً کاربردهای متفاوت، نیازمندی‌های بهبود از خرابی متفاوتی دارند. عموماً در کاربردهای بلادرنگ زمانی بهبود از خطا سودی برای تعدادی از مشتری‌ها نخواهد داشت، مگر این که بسیار سریع انجام شود.

## ۴-۸- پروتکل انتقال بلادرنگ

در این بخش به پروتکل انتقال بلادرنگ می‌پردازیم که سرویس‌های انتقال انتها به انتها را برای داده‌هایی با خصوصیات بلادرنگ، مانند صوت، تصویر و رادار فراهم می‌کند. این سرویس‌ها شامل مشخص‌کردن نوع سربار، شماره‌گذاری دنباله، الصاق زمان مهر و نظارت بر انتقال است. معمولاً در عمل از RTP بر روی پروتکل UDP استفاده می‌شود تا از تسهیم و جمع مقابله‌ای که این پروتکل ارائه می‌دهد استفاده شود. با این‌حال ممکن است از RTP بر روی پروتکل سطح انتقال دیگری مانند TCP هم استفده کرد.

دقت کنید که RTP فی‌نفسه هیچ مکانیزمی را برای انتقال در بازه‌ی زمانی مشخص یا سایر تضمین‌های کیفیت سرویس فراهم نمی‌کند و وابسته به سرویس‌های لایه‌های پایین‌تر است. این پروتکل رسیدن یا رسیدن به ترتیب را به هیچ نحوی تضمین نمی‌کند و هیچ فرضی در رابطه با

قابلیت اتکای لایه‌ی پایینی نیز انجام نمی‌دهد. شماره‌ی ترتیبی که در بسته‌های RTP قرار گرفته است، به گیرنده این *امکان* را می‌دهد که ترتیب بسته‌های مبدا را بازسازی کند. مثلاً شماره‌ی ترتیب می‌تواند برای پیداکردن مکان فریم در یک ارتباط ویدیویی مورد استفاده قرار گیرد. پروتکل کنترل انتقال بلادرنگ (RTCP)، بر اساس ارسال‌های دوره‌ای بسته‌های کنترلی به همه‌ی گیرنده‌ها با همان توزیع ارسال بسته‌ها عمل می‌کند.

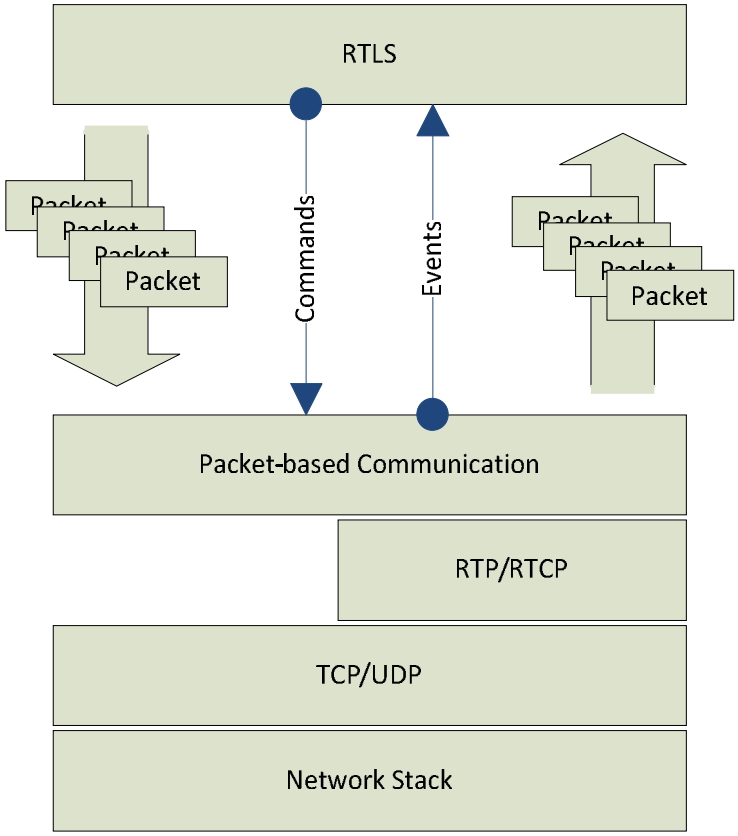
اطلاعات جامع‌تر در رابطه با فرمت بسته‌ها و سرآیندهای این پروتکل‌ها در RFC۳۵۵۰ توضیح داده‌شده است و شرح این جزئیات در این گزارش جر اطاله‌ی کلام نیست. در کد و مستندات فنی که به پیوست ارائه می‌کنیم، موارد مربوط به ساختار بسته‌ها و چینش بیت‌ها به طور مبسوط مورد بررسی قرار گرفته است.

```
  /// <summary>
  /// Updates performance counters
  /// </summary>
  /// <param name="ms">An approximate delay in since the last
update</param>
  internal void UpdatePerformanceCounters(int ms)
  {
      lock(pcLock)
      {
          if(intervalsSum != ۰)
              pc[RtcpSenderPC.ID.BandwidthAvg] = (uint)(bytes /
(intervalsSum / ۱۰۰۰.F));

          pc[RtcpSenderPC.ID.Bytes] = bytes;
          pc[RtcpSenderPC.ID.BytesPerPacketAvg] = bytesPerPacketAvg;
          pc[RtcpSenderPC.ID.BytesPerPacketMax] = bytesPerPacketMax;
          pc[RtcpSenderPC.ID.BytesPerPacketMin] = bytesPerPacketMin;

          pc[RtcpSenderPC.ID.Packets] = packets;
          pc[RtcpSenderPC.ID.PacketsPerIntervalMax] =
packetsPerIntervalMax;

          pc[RtcpSenderPC.ID.Intervals] = intervals;
          pc[RtcpSenderPC.ID.IntervalForced] = intervalForced;
          pc[RtcpSenderPC.ID.IntervalMax] = intervalMax;
          pc[RtcpSenderPC.ID.IntervalMin] = intervalMin;
          if(intervals != ۰) pc[RtcpSenderPC.ID.IntervalAvg] = intervalsSum
/ intervals;

          pc[RtcpSenderPC.ID.Ssrc] = rtpSession.SSRC;
      }
  }
```

**کد ۱ – به‌روزرسانی آماره‌های RTCP**

## ۴–۹– ساختار شیءگرا در انتقال بسته‌ها

کتابخانه‌ی تولیدی برای فراهم‌کردن انتقال بلادرنگ برای سهولت استفاده توسط کاربر بایستی واسطی را ارائه کند تا پیچیدگی‌های RTP در آن پنهان باشد. با توجه به این‌که اغلب نیروهای مرکز تحقیقات نصر با پلتفرن دات‌نت آشنا هستند، برآن شدیم تا با استفاده از مکانیزم *رخدادها* در C# این ساختار را فراهم کنیم. نهایت اطلاعاتی که استفاده‌کننده از این کتابخانه بایستی داشته باشد، ارث‌بری از یک کلاس `Packet` و پیاده‌سازی توابع مربوط به سریاله کردن در قالب آرایه‌ای از بایت‌ها و تبدیل آرایه‌ی بایت‌ها به نوع `Packet` است. به علاوه با هوک کردن رخدادها می‌توان از ارسال بسته‌ها، دریافت آن‌ها و نیز قطع‌شدن و اتصال مجدد ارتباط به صورت آسنکرون اطلاع پیاده کرد. شکل زیر نمایشی از پیاده‌سازی موجود را نشان می‌دهد.




**تصویر ۱۶ – معماری کتابخانه‌ی ارتباطات بلادرنگ**

## ۴–۱۰– بهینه‌سازی ارسال پالس

همان‌طور که در بخش‌های ابتدایی توضیح دادیم، ممکن است شرایطی وجود داشته باشد تا هدف توسط همه‌ی گیرنده‌ها دیده نشود. در این حالت اگر مطابق با روشی استاندارد کار کنیم نیاز داریم تا تمام پالس‌ها را به واحد پردازش ارسال‌کرده و تمام پردازش را در همین واحد متمرکز کنیم.

این‌کار علاوه بر بالابردن نیاز پردازشی در واحد پردازش، ترافیک شبکه را هم بسیار افزایش می‌دهد و متعاقباً پدیده‌ی ازدحام در شبکه منجر به گم‌شدن بسته‌ها، ارسال مجدد و به طور کلی افزایش تاخیر می‌شود. افزایش تاخیر در دریافت بسته‌ها توسط واحد پردازش مرکزی، می‌تواند به از دست رفتن مهلت‌ها شده و کلاً بلادرنگی سیستم را نقض کند.

ایده‌ای که در این قسمت به آن می‌پردازیم را می‌توان به نوعی ترکیب دو روش ToA و همبستگی متقابل که هریک پیشتر جداگانه شرح داده‌شد، دانست. به این ترتیب که با استفاده از یک ماژول تشخیص پالس در گیرنده‌ها، به محض دریافت پالس، اندیس زمانی آن به واحد مرکزی ارسال می‌شود. دقت کنید که این اندیس به طور بیشینه یک یا چند متغیر `long` است که حجم آن بسیار کمتر از حجم مورد نیاز برای یک طول پالس کامل است. این اندیس‌ها به محض دریافت در یک بافر قرار می‌گیرند. اندیس‌هایی که در یک بازه‌ی زمانی محدود با توجه به توپولوژی گیرنده‌ها قرار دارند، محتمل برای تولید پیک در همبستگی متقابل هستند. بنابراین به محض این‌که تعداد گیرنده‌هایی که در یک بازه‌ی محدود یک پالس را کشف کرده‌اند به سه یا بیشتر رسید، واحد پردازش مرکزی از گیرنده‌های متناظر می‌خواهد تا اصل داده‌ی پالس‌ها را هم ارسال کند. سپس همبستگی متقابل بر روی دو به دوی این پالس‌ها انجام می‌شود و اختلاف زمانی‌ها به دست می‌آید.

با توجه به گستردگی سیستم، بسیار پیش می‌آید که برخی پالس‌ها تنها توسط بخشی از گیرنده‌ها دریافت شوند که از آن‌جایی که این تعداد به میزان کافی نیست، امکان محاسبات مکانی وجود ندارد، بنابراین نیازی به ارسال داده‌ی مربوط به آن‌ها نیست. روش بالا با صرفه‌جویی در ارسال بسته‌ها و یه کار گیری یک مکانیزم کندرو[۴۳] ترافیک شبکه را به شدت کاهش می‌دهد.

[۴۳] که به مکانیزم‌هایی اطلاق می‌شود که پردازش (در این‌جا ارسال داده) را تا جایی که نیازی به آن وجود ندارد به تاخیر Eager در مقابل Lazy معادل کلمه می‌اندازند.

# فصل ۵–  مفهوم نقشه

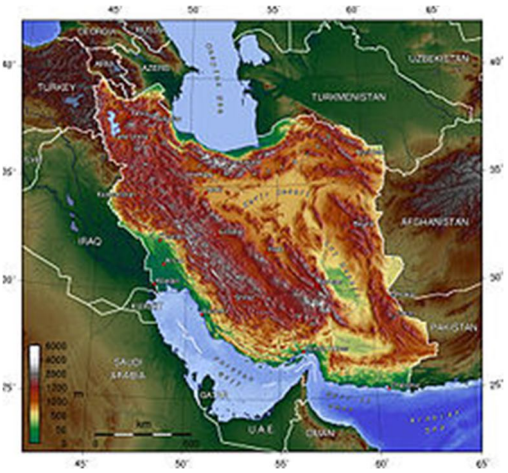
## ۵–۱– انواع نقشه

نقشه تجریدی از واقعیت جهان است، خصوصیات جغرافیایی یا اطلاعات مکانی جغرافیایی را ارایه کرده و دربردارنده‌ی اطلاعاتی درباره‌ی موقعیت ویژگی‌هاست.

هزاران سال است که از نقشه‌ها جهت ارائه‌ی اطلاعات در مورد جهان واقعی استفاده می‌شود. مفهوم و طرح آن‌ها تبدیل به علمی با درجه بالایی از پیچیدگی شده است. ثابت شده که نقشه‌ها به شدت برای بسیاری کاربردها در قلمروهای گوناگون مفید هستند. کارتوگرافی به عنوان یک علم و هنر ساخت نقشه، به عنوان مترجم پدیده‌های دنیای واقعی، با بازنمایی درست، روشن و قابل فهم برای استفاده‌ی ما عمل می‌کند. هم‌چنین نقشه‌ها می‌توانند منبع اطلاعات برای نقشه‌های دیگر باشند. با ظهور سیستم‌های کامپیوتری، نقشه‌کشی آنالوگ تبدیل به نقشه‌کشی دیجیتال شد. یادآوری این نکته مهم است که امروزه هرگاه در مورد کارتوگرافی سخن می‌گوییم، به طور ضمنی همان نقشه‌کشی دیجیتال را در نظر می‌گیریم. استفاده از کامپیوترها در ساخت نقشه، بخشی جدایی ناپذیر از نقشه‌کشی مدرن است. نقش نقشه بر همین اساس تغییر یافته است.

به طور فزاینده نقشه‌ها در حال از دست دادن نقش خود به عنوان منبع نگهداری اطلاعات هستند. این نقش تاکنون بیش از پایگاه داده گرفته شده است. چیزی که باقی می‌ماند همان کارکرد تصویری نقشه‌هاست.

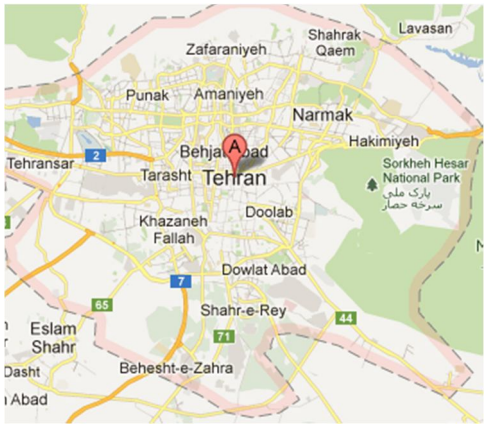
دو دسته نقشه بر مبنای هدف استفاده وجود دارد: (۱) نقشه‌های مرجع یا عمومی (۲) نقشه‌های موضوعی یا خاص منظوره




**نمونه ای از یک نقشه**

### ۵-۱-۱- نقشه‌های مرجع یا عمومی

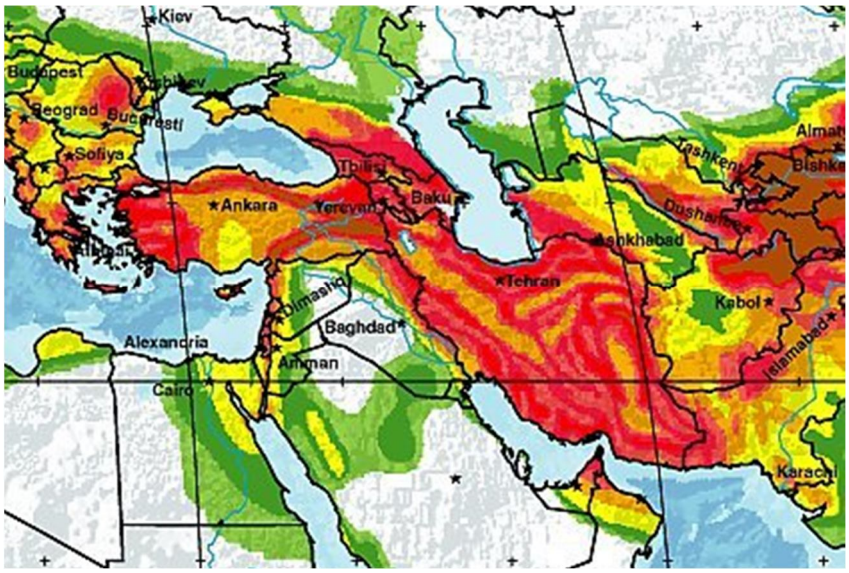
این نقشه‌ها برای هیچ هدف خاصی طراحی نشده‌اند. این دسته از نقشه‌ها روی مکان‌هایی متمرکز شده و گوناگونی از خصوصیات فیزیکی و فرهنگی مانند شکل زمین، زه‌کشی، رودخانه- دریاچه، جاده‌ها، راه‌آهن، فرودگاه، مناطق ساخته‌شده، جنگل‌ها، و مناطق زیر کشت را نشان می‌دهد. آن‌ها به عنوان یک مبنای مرجع جغرافیایی برای توسعه، و ادغام دیگر فرم‌های اطلاعات جغرافیایی عمل می‌کنند. مثلاً در کامبوج، ما اغلب از نقشه‌های توپوگرافی ۱:۵۰,۰۰۰، ۱:۱۰۰,۰۰۰ و ۱:۲۵۰,۰۰۰ استفاده می‌کنیم که توسط ایالات متحده بین سال‌های دهه‌ی ۱۹۶۰ و ۱۹۷۰ تولید شده‌اند.




**تصویر ۱۷ – یک نمونه نقشه‌ی عمومی**

### ۵-۱-۲- نقشه‌های موضوعی یا خاص منظوره

این‌ها نقشه‌هایی هستند که جهت به تصویر کشیدن نوع خاصی از ویژگی یا اندازه‌گیری طراحی شده‌اند. آن‌ها در مقیاس‌های گوناگون بر مبنای نقشه‌ی کلی تولید شده‌اند. بنابراین نقشه‌های توزیع جمعیت، اعضا، جنگل/ زمین، ناحیه‌ی ماهیگیری، جامعه‌ی شیلات/ جنگل، مناطق حفاظت شده، توزیع بارش باران، رودخانه‌ها، شامل زمین و بزرگراه‌ها در کامبوج مثال‌های خوبی هستند. در کاربردهای GIS، این نقشه‌ها پدیده‌ها فرآیندهای جغرافیایی را به تصویر می‌کشند.




**تصویر ۱۸ – نقشه‌ی سیزمولوژی (زلزله‌شناسی) خاورمیانه**

## ۵–۲– عناصر و ویژگی‌های نقشه

نقشه‌ها به عنوان تصاویری از جهانی که با آن‌ها مدل‌سازی می‌کنیم، موقعیت‌های اشیا را در مکان و نیز کیفیت یا بزرگی آن‌ها را نشان می‌دهند. این دو مورد مرتبط به ترتیب نهادها و ویژگی‌های نامیده شده و هر دو به همان اندازه‌ای در اسناد کارتوگرافی لازم هستند که در هنگام جمع‌آوری اطلاعات در مکان اولیه به آن‌ها توجه می‌کنیم. اما بدون توجه به اینکه آبا نهادها و اشیا ارایه دهنده‌ی نقاط، خطوط، نواحی یا سطوحی در جهان واقعی هستند یا نه، نمی‌توانند به عنوان واقعیت‌های کوچک‌سازی شده نشان داده شوند چون در مقیاس محدودیت‌هایی وجود دارد. به جای آن، بایستی آن‌ها را در حافظه‌ی کامپیوتر ذخیره نمود و سپس *روی برون داد*، یک سری سمبل‌ها جهت بازنمایی آن‌ها اختصاص دهیم. به نوبه‌ی خود، این سمبل‌ها بایستی دارای کلیدی تفسیری به نام افسانه‌ی نقشه باشند که بشود توسط کاربر به آن‌ها اشاره کرد. این یک ویژگی واقعی با خصوصیات قابل اندازه‌گیری فهمیده شود. در این طریق خواننده‌ی نقشه قادر به پیش‌بینی آنچه که عملاً دیده شده و به عنوان داده‌های اصلی‌ای خواهد بود که از طریق روش‌های گوناگون جمع‌آوری شده است.

این نهادها قابل اندازه‌گیری بوده و خصوصیات آن‌ها می‌تواند با استفاده از تعدادی سطوح مختلف اندازه‌گیری داده‌ها تحدید شود. اطلاعات ویژگی خصوصیات ویژگی‌های جغرافیایی ارایه شده را توصیف می‌کنند، از قبیل نوع ویژگی، نام آن و اطلاعات کمی از قبیل ناحیه‌ی آن یا طول.

- ویژگی نقطه‌ای[٤٤]. یک ویژگی نقطه‌ای نمایانگر یک نقطه‌ی واحد است. یک شی در نقشه را برای نشان دادن یک خط یا یک ویژگی منطقه بیش از حد کوچک تعریف می‌کنند. نماد خاصی از برچسب معمولاً یک مکان نقطه را به تصویر می‌کشد. مثلاً مکان یک صحنه‌ی تصادف پایگاه نظامی و غیره.
- ویژگی خطی[٤٥]. یک ویژگی خطی یک سری مختصات متصل و مرتب است که نمایانگر شکل خطی شی در نقشه‌ای هستند که ممکن است برای نشان دادن به عنوان ناحیه‌ای مانند یک جاده، جریان رودخانه و یا خصوصیتی، هیچ عرضی از جمله خط کانتور، بسیار باریک باشد.
- ویژگی ناحیه‌ای[٤٦] (چندضلعی). ویژگی ناحیه‌ای، شکلی بسته است که مرزهای آن ناحیه‌ی همگن را در بر می‌گیرد، مانند نوع خاک یک ایالت، ناحیه‌ی دریاچه یا جنگل یک گونه‌ی معین.

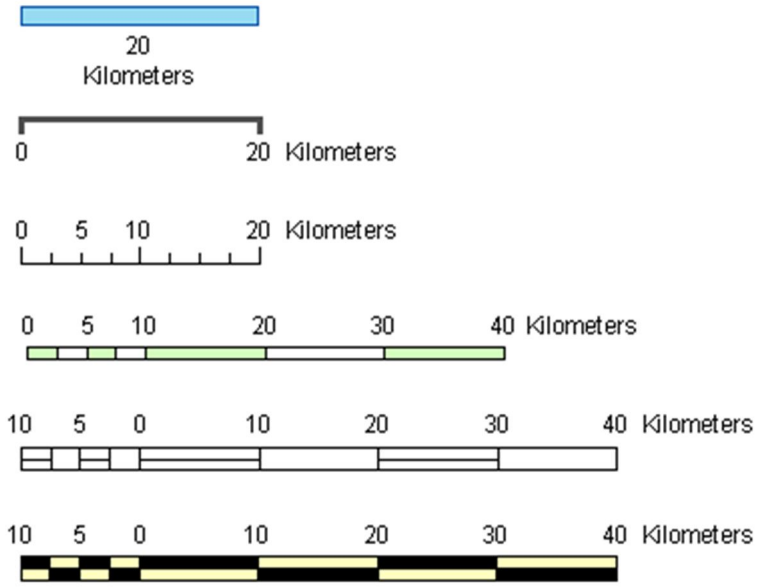
## ۵-۳- مقیاس و تفکیک پذیری نقشه

مقیاس نقشه مقدار کاهش تعریف شده به عنوان ضریب فاصله روی نقشه با فاصله‌ی مشابه است. همان طوری که روی زمین به نظر می‌رسد. واحد موجود در سمت چپ فاصله را روی نقشه نشان می‌دهد و عدد روی سمت راست فاصله روی زمین را نشان می‌دهد. مورد آخری به عنوان کسر نمایش (RP) شناخته شده، زیرا مقادیر در هر دو سمت دو نقطه با هم، هم ارز هستند، یعنی ۱:۲٤,۰۰۰ یعنی یک اینچ برابر است با ۲٤۰۰۰ اینچ یا یک فوت برابر است با ۲٤,۰۰۰ پا و یا ۱ متر برابر است با ۲٤,۰۰۰ متر و غیره.

[٤٤] Point Feature معادل عبارت

[٤٥] Line Feature معادل عبارت

[٤٦] Area Feature معادل عبارت




**تصویر ۱۹ – مثال‌هایی از نمایش مقیاس در نقشه**

مقیاس نقشه نشان می‌دهد که منطقه‌ی مورد نظر تا چه حدی کاهش یافته است. برای نقشه یا سایز مشابه، ویژگی‌ها روی نقشه‌ی مقیاس کوچک (۱:۱۰,۰۰۰,۰۰۰) از آن‌هایی که روی نقشه‌ی سایز بزرگ هستند (۱:۱۲۰۰) کوچک‌تر خواهد بود.

از نقشه‌ای با جزئیات کمتر انتظار می‌رود که نسبت به نقشه‌ای که دارای جزئیات بیشتری است، کوچک‌تر باشد، نقشه‌کش‌ها اغلب مقیاس‌ها را به سه دسته‌ی مختلف تقسیم می‌کنند. نقشه‌های مقیاس کوچک دارای مقیاس‌هایی کوچک‌تر از ۱:۱۰۰۰,۰۰۰ بوده و برای نقشه‌هایی از نواحی وسیع استفاده می‌شوند که جزئیات زیادی مورد نیاز نیست.

تفکیک‌پذیری نقشه به این مورد اشاره دارد که چگونه موفقیت و شکل خصوصیات نقشه می‌تواند یک مقیاس نقشه‌ی مورد نظر را به تصویر بکشد. مقیاس روی تفکیک پذیری اثر می‌گذارد. در یک نقشه با مقیاس بزرگ‌تر، تفکیک‌پذیری خصوصیات، بیشتر با خصوصیات دنیای واقعی انطباق دارد زیرا حدی از تفکیک‌پذیری از زمین تا نقشه کمتر است. همین‌طور که مقیاس نقشه کاهش می‌یابد، تفکیک‌پذیری نقشه تخریب می‌شود زیرا خصوصیات بایستی صاف و ساده‌ای‌شند یا اصلاً نشان داده نشوند.

## ۵-۴- دقت و صحت نقشه

یک نقشه‌ی غیر دقیق قابل اتکا نیست. مکان X ممکن است نشاندهنده‌ی جایی باشد که گنج در آن مدفون شده اما تا وقتی که جستجوگر نتواند X را نسبت به نشانه‌های واقعی شناخته شده موقعیت‌یابی کنند، نقشه زیاد به درد نمی‌خورد.

بررسی زمین‌شناسی ایالات متحده (USGS) نقشه‌ها و دیگر محصولات را در سطوح بالایی از دقت و صحت منتشر می‌کند. قابلیت اتکا بسیار حیاتی است، مثلاً برای مهندسان، مسئولان بزرگراه و طراحان استفاده از زمین که از نقشه‌های توپوگرافی USGS به عنوان ابزار نقشه‌کشی اساسی استفاده می‌کنند. در نتیجه، USGS هر تلاشی می‌کند تا به سطح بالایی از صحت در تمام محصولات منتشر شده‌ی خود برسد. یک هدف مهم صحت در برنامه‌ی کنترل انطباق آن با استاندارد دقت نقشه‌ی ملی ایالات متحده است.

### ۵-۴-۱- استانداردهای دقت نقشه‌ی ملی ایالات متحده

جهت یافتن روش‌ها برای اطمینان از صحت موقعیت (عرض و طول یک نقطه) و بلندی (ارتفاع از سطح دریا) انجمن امریکایی برای فتوگرامتری و سنجش از دور- سازمانی که فعالانه درگیر علم ساختن اندازه‌گیری‌های دقیق از عکس‌هاست (فتوگرامتری) و در حال کسب اطلاعات از عکس‌های هوایی و اطلاعات خاص از تصاویر ماهواره (سنجش از دور) است- کمیته‌ای را در سال ۱۹۳۷ جهت تهیه پیش نویس مشخصات صحت و دقت تشکیل داد. آژانس‌های دولت‌مندان، مثل USGS، تحقیقات خود را از استانداردهای دقت نقشه آغاز کردند. در سال ۱۹٤۱ دفتر امریکایی بودجه استانداردهای دقت نقشه‌های ایالات متحده را منتشر کرد که در تمام آژانس‌های فدرال که تولید کننده‌ی نقشه بودند صدق می‌کرد. این استانداردها چندمین بار اصلاح شد و نسخه‌ی کنونی در سال ۱۹٤۷ منتشر شد.

از آن جایی که در نقشه‌ی توپوگرافی USGS صدق می‌کند، استاندارد دقت افقی نیازمند این است که موفقیت‌های ۹۰ درصد تمام نقاط ثبت شده بایستی ظرف $\frac{۱}{۵۰}$ یک اینچ ( ۰.۰۵ سانتی متر) روی نقشه دقت داشته باشند. در مقیاس ۱:۲٤,۰۰۰، $\frac{۱}{۵۰}$ اینچ ٤۰ فوت (۱۲.۲ متر) است. صحت عمودی نیازمند این است که ارتفاع ۹۰ درصد تمام نقاط ثبت شده بایستی در نصف یک دقیقه‌ی کانتور صحیح باشند. روی نقشه‌ای با وقفه‌ی کانتور ۱۰ فوت، این نقشه باید ۹۰ درصد تمام نقاط ثبت شده را در ۵ فوت نشان دهد. (۱.۵ متر)

تمام نقشه‌های تولید شده توسط USGS در مقیاس ۱:۲۵۰،۰۰۰ و بزرگ‌تر با روش‌هایی آماده شده‌اند که جهت پاسخ گویی به لزوم استانداردهای صحیح طراحی شده و حامل این پیام باشند: این نقشه مطابق استانداردهای دقت نقشه‌ی ملی است. انتظارات در مورد این مورد شامل نواحی پوشش داده شده توسط درختان بسیار متراکم یا پوشیده از مه و ابر است.

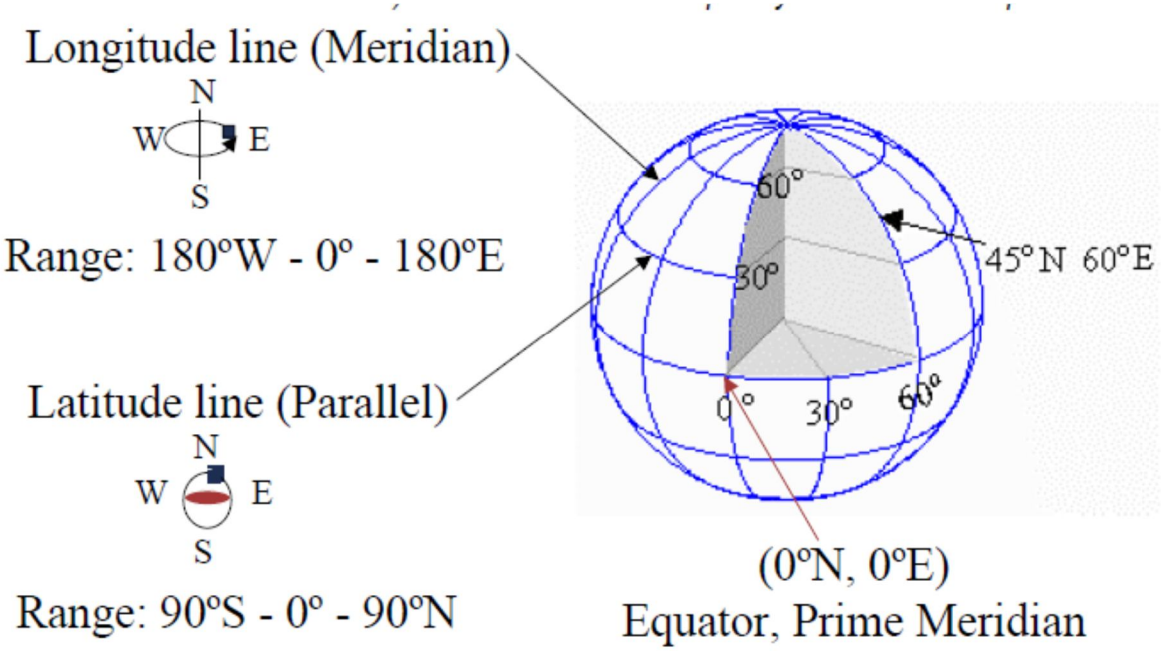
در این نواحی، عکس‌های هوایی نمی‌توانند جزئیات مورد نیاز را جهت نقشه برداری دقیق فراهم کنند. تست‌های USGS به قدر کافی نقشه‌های خود را تست می‌کند تا مطمئن شود وسایل و روندهای بررسی مورد استفاده در تولید نقشه مطابق استانداردهای دقت نقشه‌ی ملی ایالات متحده است.

## ۵–۵– مختصات نقشه و سیستم‌های طرح‌ریزی

به منظور مشخص نمودن مکان‌ها روی زمین، یک سیستم مرجع هماهنگ مختصات سه بعدی بایستی ایجاد شود که شکل آن را نیز به حساب آورد. سیستم مختصات کروی بیش از دو هزار سال است به عنوان سیستم مختصات جغرافیایی استفاده می‌شود. و از شبکه‌ای از عرض و طول جغرافیایی (به نام چهارخانه کردن) جهت ثابت کردن موقعیت‌های نقاط روی زمین استفاده می‌کند.

مفهوم عرض و طول جغرافیایی را می‌توان به کارهای ستاره شناسی یونانی هیپارکوس در میانه‌ی قرن دوم میلاد نسبت داد که توسط کلادیوس تولمی در سیصد سال پس از آن فرم‌دهی شد. این دو نقطه روی سطح زمین توسط محور چرخش تقسیم می‌شوند. در نیمه راه میان دو قطب خطی فرضی به نام استوا قرار دارد. محور قطبی و دایره‌ی حاوی استوا در مرکز زمین با زاویه‌ای قائمه یکدیگر را قطع می‌کنند که به عنوان منشأ این سیستم مختصات جغرافیایی دانسته می‌شود.

موقعیت یک نقطه روی سطح کروی زمین فقط در زاویه تعیین می‌شود» در صفحه‌ی چند وجهی که در منشأ یا مرکز زمین یکدیگر را قطع می‌کنند. یکی از این صفحات، صفحه‌ی استوا است که به عنوان صفحه‌ی مرجع جهت اندازه‌گیری اولین زاویه (زاویه عمودی به نام عرض جغرافیایی) استفاده می‌شود. صفحه‌ی دیگر که از نصف‌النهار انتخاب می‌شود، به عنوان طول صفر در نظر گرفته می‌شود (به نام نصف‌النهار مبدأ)، که به عنوان مبدأ جهت اندازه‌گیری زاویه‌ی دوم (زاویه‌ی افقی) در طول استوا به نصف‌النهاری دیگر که از روی نقطه‌ای روی سطح زمین که باید ثابت شود می‌گذرد. این زاویه را به عنوان طول جغرافیایی (شکل ط) می‌شناسیم.




**تصویر ۲۰ – سیستم مختصاتی زمین**

این را در زاویه‌ی شمال یا جنوب خط استوا اندازه‌گیری می‌کنند به طوری که عرض استوا صفر درجه است، عرض قطب شمال نود درجه‌ی شمالی و قطب جنوب نود درجه‌ی جنوبی است.

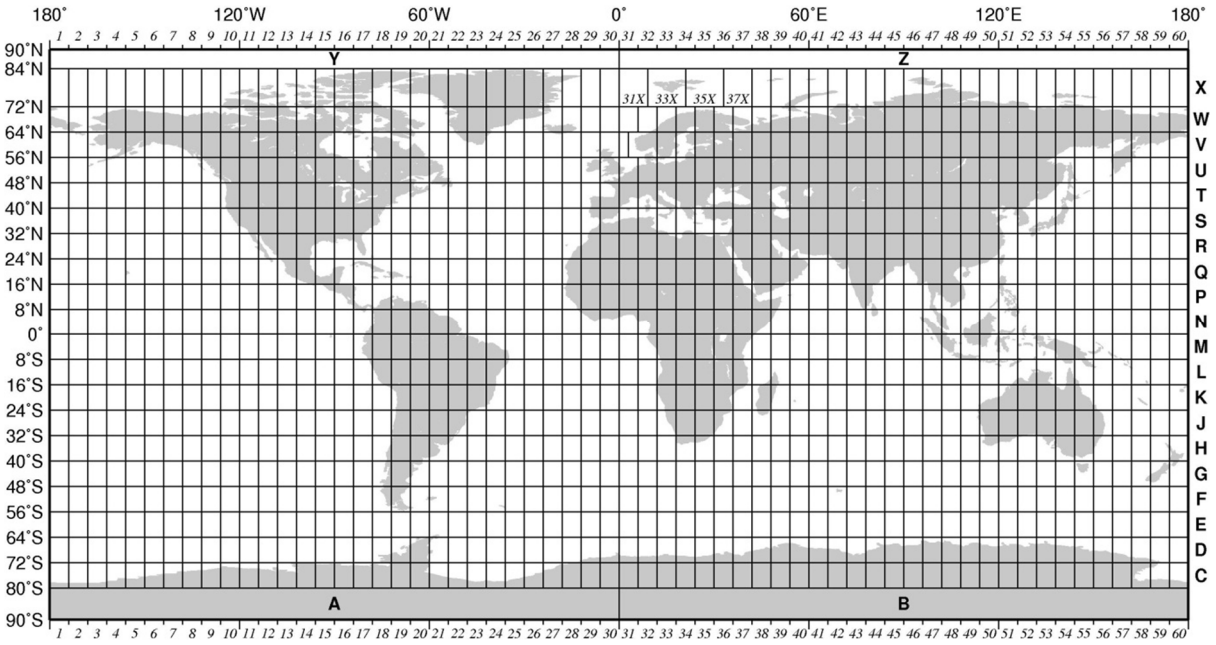
در سال ۱۸۸٤، جامعه نقشه‌کشی جهان در استفاده از نصف‌النهار گرین‌ویچ در لندن انگلستان موافقت کرد که به جای نصف‌النهار مبدأ مورد استفاده قرار بگیرد (یعنی با عرض صفر درجه). طول جغرافیایی از شرق و غرب نصف‌النهار گرین‌ویچ تا ۱۸ درجه اندازه‌گیری می‌شود. نشانه‌ی قراردادی این است که طول شرقی مثبت (+) و طول غربی منفی (–) باشد.

روی سطح زمین، یک سری تشابهات از عرض جغرافیایی را می‌توان به موازات خط استوا کشید و یک سری را می‌توان از نصف‌النهارهای طول جغرافیایی را از قطب تا قطب کشید که هر خط موازی خطوط عرض جغرافیایی را با زاویه قائم قطع کند اما در چندین نقطه روی زمین ضربه می‌زند. این سری از خطوط فرضی یک شبکه‌ی خطوط موازی مدار نصف‌النهار را تشکیل می‌دهند که در بردارنده‌ی یک سیستم جغرافیایی مختصات است که قبلاً شرح داده شده است.

### ۵–۵–۱– سیستم مختصات جهانی متقاطع مرکاتور (UTM)

سیستم مختصات جهانی متقاطع مرکاتور یک سیستم بر مبنای پروجکشن مرکاتور متقاطع است (که در اروپا به عنوان پروجکشن گاوس–کروگر شناخته می‌شود) و توسط یوهان هاینریش لمبرت اختراع شده است. سیستم مختصاتی UTM با قرار دادن یک مربع معمولی و منظم روی هر ناحیه‌ی UTMبا ٦ درجه عرض جغرافیایی تشکیل شده است. این شبکه قرار در وسط قرار دارد به طوری که

خطوط عمودی به موازات نصف‌النهار مرکزی هستند. نواحی UTM از ۱ تا ٦٠ شماره‌گذاری شده‌اند که از خط تاریخ بین‌المللی (که هم چنین به عنوان آنتی مری دین= $^{o}$١٨٦نیز شناخته می‌شود) و تا شرق ادامه می‌یابد. بنابراین ناحیه‌ی ۱ از ۱۸۰ درجه‌ی غربی تا ١٧٤ درجه غربی با نصف‌النهار مبدأ در ۱۷۷ درجه‌ی غربی ادامه می‌یابد. کامبوج در ناحیه‌ی ٤٨ قرار دارد. هر ناحیه‌ی UTM به باند افقی پوشای ۸ درجه طول جغرافیایی تقسیم شده است. این باندها با حروف، جنوب تا شمال شناسایی شده و از ۸۰ درجه‌ی جنوبی با حرف C آغاز و با حرف X در ٨٤ درجه‌ی شمالی پایان می‌یابد (شکل ط).




**تصویر ۲۱ - سیستم مختصات جهانی متقاطع مرکاتور**

## ۵-۵-۲- پروجکشن نقشه

چون به تولید نقشه‌ای از زمین روی سطح صاف نیاز داریم، برخی روش‌های تبدیل مختصات جغرافیایی همان‌طور که قبلاً شرح داده شده‌اند از سه بعد به دو بعد، مورد نیاز است. پروجکشن نقشه بازنمایی سیستماتیک تمام یا قسمتی از سطح یک جسم مدور، خاصه زمین است که روی یک صفحه انجام می‌شود. سطح این نقشه می‌تواند یک صفحه، یک استوانه، یا یک مخروط باشد، یعنی چیزی که بتوان تای آن را باز کرد و آن را تبدیل به یک صفحه کرد. هر نقشه توسط اصلاح روش بازنمایی آن طوری بدست می‌آید که خصوصیات مورد نظر خاصی بتوانند باقی بمانند، مانند ناحیه‌ای برابر فاصله‌ی برابر یا شکلی صحیح. یک مثال خوب همان پروجکشن مرکاتور است.

### ۵-۵-۲-۱- خصوصیات پروجکشن‌های نقشه

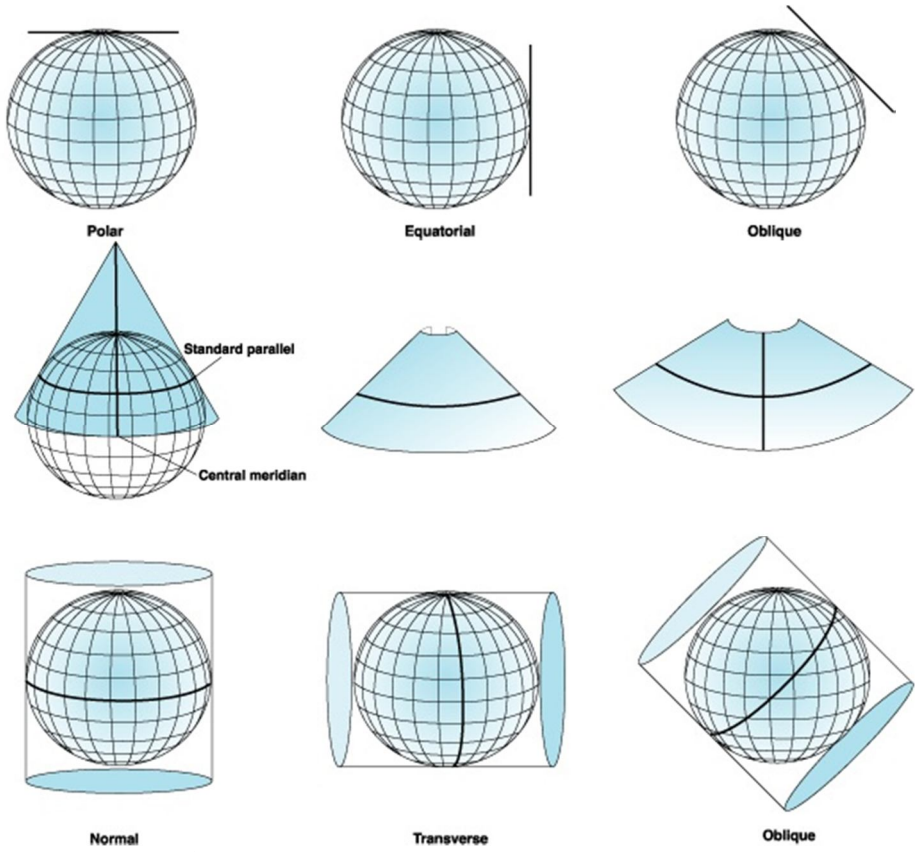
اگر یک پرتقال را روی یک سطح صاف له کنید، پوست پرتقال جدا شده و در تمام جهات پخش می‌شود و نیز خواص اصلی پرتقال مانند شکل آن تغییر خواهد کرد. در پروجکشن نقشه، زمین می‌تواند به عنوان یک کره‌ی کامل در نظر گرفته شود.

باز نمایی آن روی یک صفحه‌ی صاف مانند همان له کردن پرتقال روی یک سطح صاف است، که برخی از خصوصیات کروی زمین نیز با آن از بین خواهد رفت. چهار خاصیت است که باید در نظر گرفت: (۱) ناحیه (۲) شکل (۳) فاصله (٤) جهت. برای یک زمین کروی تمام این چهار خاصیت کامل است گرچه اگر یک‌بار زمین به یک صفحه منتقل شود، تنها برخی از خصوصیات آن باقی خواهد ماند. بنابراین تمام طرح نقشه‌های مختلف برای تولید یک شبکه از کانال‌های نصف‌النهار و مدار طراحی شده‌اند که می‌تواند یک یا دو تا از این ویژگی‌ها را در نقشه‌ی نهایی برای هدف خاصی از نحوه‌ی بازنمایی ارایه کند.

- ناحیه. یک پروجکشن نقشه ممکن است برای یک ناحیه‌ی برابر طراحی شده باشد به طوری که هر ناحیه‌ی اندازه‌گیری شده روی نقشه با همان اندازه‌گیری روی سطح زمین برابر باشد. به این« ناحیه‌ی برابر یا پروجکشن معادل» می‌گویند. ویژگی‌های دیگر تحریف می‌شوند.
- شکل. می‌تواند شکل صحیحی از ویژگی‌های ارایه شده از خصوصیات فضایی را ارایه دهد. این امر تنها با ساختن مقیاس در طول نصف‌النهار و مداراتی مشابه از هر دو جهت امکان‌پذیر است. این نوع از طراحی را به عنوان «مطابقتی یا ارتومورنیک» می‌شناسیم. به طور اجتناب ناپذیری هم ناحیه و هم فاصله را تخریب می‌کند.
- فاصله. بهتر است فاصله‌ای بین دو نقطه‌ی اندازه‌گیری شده روی نقشه‌ای باشد که فاصله از آن‌ها مساوی است و این نیز روی زمین نیز صدق کند. به آن « پروجکشن مساوی از نظر فاصله» می‌گوییم.
- جهت. احیای اندازه‌گیری جهتی که روی نقشه است، مشابه همان جهت‌های روی زمین است. خصوصیات دیگر روی نقشه را نمی‌توان نگاه داشت.

### ۵-۵-۲-۲- رده بندی پروجکشن نقشه

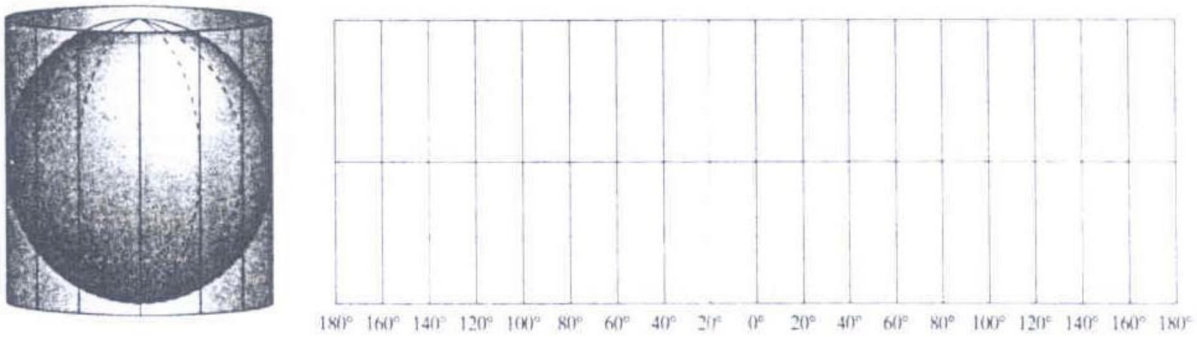
پروجکشن‌های متفاوتی جهت رده‌بندی وجود دارد. یکی از پروجکشن‌های ساده برای طبقه بندی پیش بینی نقشه با توجه به نوع سطح قابل توسعه است که بر روی شبکه‌ای از کانال‌های نصف‌النهار و مدار بنا شده است. سطح قابل توسعه سطحی است که بدون تخریب بتوان آن را صاف کرد. سه نوع سطح قابل توسعه داریم: (۱) استوانه‌ای (۲) قیفی (۳) دو وجهی




**تصویر ۲۲ – روش اعمال پروجکشن‌های استوانه‌ای، قیفی و آزیموتال**

### ۵-۵-۲-۳- پروجکشن استوانه‌ای

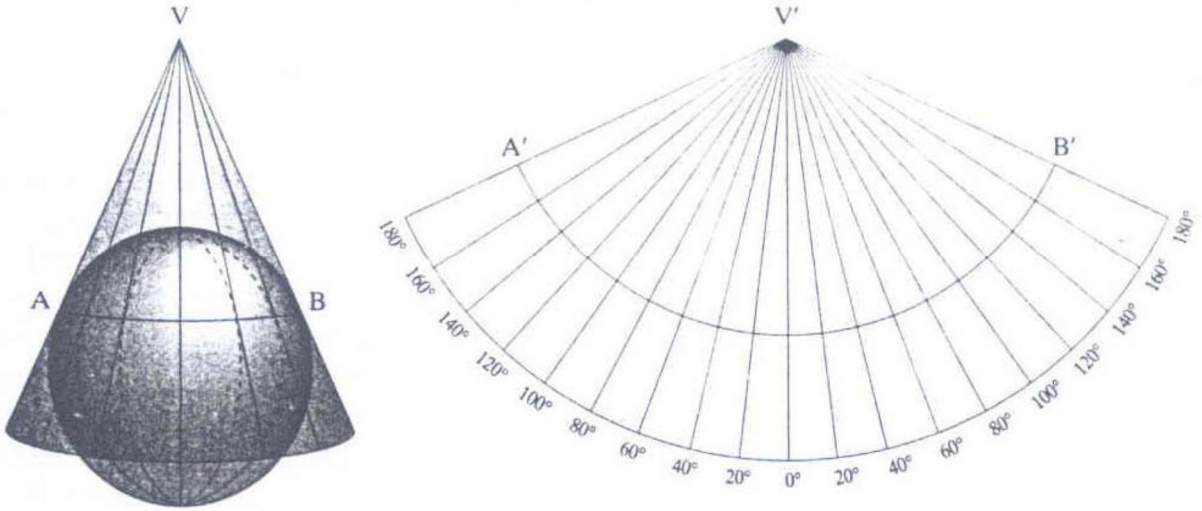
از یک استوانه انتظار می‌رود که یک کره‌ی شفاف را طوری عمود و مشخص کند که این استوانه در سراسر محیط پیرامون به استوا دسترسی داشته باشد.(شکل ط) و این با نصف‌النهارها و مدارها مشخص شده است. با فرض اینکه یک لامپ برق در مرکز این کره نهاده شده است، چهارخانه کردن کره روی استوانه انجام می‌شود. با برش استوانه و باز کردن آن در امتداد نصف‌النهار یک طرح استوانه‌ای مستطیل شکل بدست می‌آید. نصف‌النهارها عمودی‌اند و مدارها خطوطی مستقیم، که استوا را با زوایای قائم برش داده و آن را به ۳۶۰ قسمت مساوی تقسیم می‌کند. مدارها همان خطوط صاف افقی هستند که در فواصل انتخاب شده از استوا و از یکدیگر قرار دارند.




**تصویر ۲۳ – نمونه‌ای از پروجکشن استوانه‌ای**

### ۵-۵-۲-٤- پروجکشن قیفی

یک قیف روی یک کره طوری قرار می‌گیرد که رأس قیف دقیقاً روی محور قطبی (شکل ط) قرار می‌گیرد. یک قیف بایستی کره را در طول مدار عرض جغرافیایی که همان مدار استاندارد است لمس کند که توسط کار توگرافر می‌تواند انتخاب شود. در طول این مدار استاندارد، مقیاس صحیح بوده و تخریب در پایین‌ترین سطح قرار دارد. هنگامی که قیف برش خورده و باز می‌شود و در طول نصف‌النهار به صورت صاف قرار می‌گیرد، یک نقشه‌ی پنکه مانند تولید می‌شود، با نصف‌النهارهایی که مثل خطوط صافی هستند که از رأس با زاویه مساوی تشعشع می‌کند، در حالی که مدارها قوس‌های دایره‌ای هستند که تمام آن‌ها با استفاده از رأس در مرکز کشیده شده‌اند.

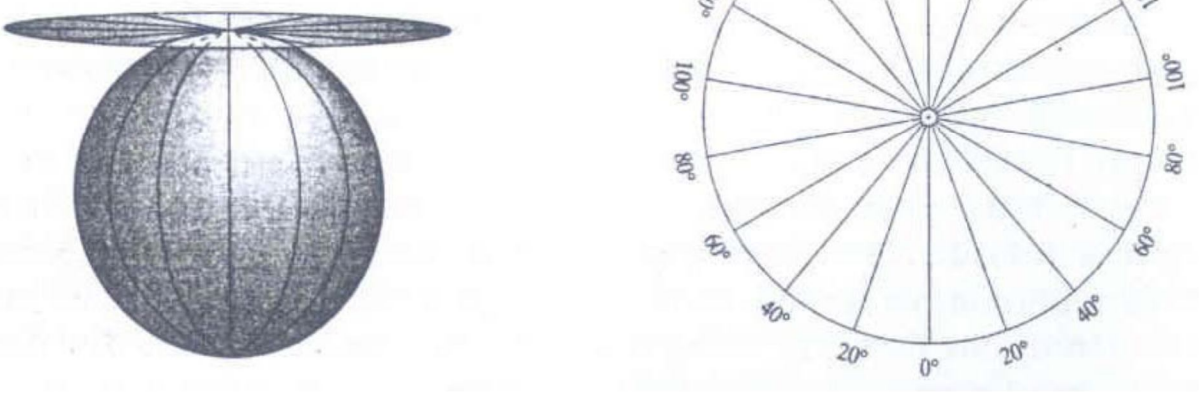



**تصویر ٢٤ – نمونه‌ای از پروجکشن قیفی**

### ۵-۵-۲-۵- پروجکشن دو وجهی یا آزیموتال

یک صفحه چنان قرار داده می‌شود که کره را در قطب‌های شمال و جنوب می‌کند (شکل ط). این می‌تواند این گونه اثبات شود که قیف به طور فزاینده‌ای به گونه‌ای صاف شده که رأس آن به حد ۱۸۰° می‌رسد. این طرح نتیجه، با نام طرح آزیموتال قطبی بهتر شناخته می‌شود. شکل آن دایره‌ای

است با نصف‌النهارهایی که به عنوان خطوط مستقیمی کشیده شده‌اند که از مرکز دایره انشعاب یافته‌اند (نقطه‌ی تانژانت صفحه) که همان قطب است. این نصف‌النهارها در زوایای صحیح خود قرار گرفته‌اند. مدارها دوایر کاملی هستند که مرکز آن‌ها قطب است. شعاع این دایره‌ها ممکن است با مراجعه به برخی فرمول‌های ریاضی محاسبه شود، مانند طرح مسافت مساوی آزیموتال یا از نظر هندسی از برخی نقطه نظرها در جایی روی محور قطبی یا ادامه‌ی آن مانند طرح استریوگرافیک قرار گرفته است (نقطه نظر در مرکز کره)




**تصویر ۲۵ – نمونه‌ای از پروجکشن مسطح یا آزیموتال**

# فصل ۶– معرفی سیستم‌های اطلاعات جغرافیایی

## ۶–۱– مقدمه

برنامه‌ریزی جهت هر نوع کاری نیازمند داشتن اطلاعات مربوط به آن کار است که این نیازمندی در مورد برنامه‌ریزی برای استفاده‌های انسان از سرزمین نیز صادق است. زیرا بدون داشتن اطلاعات در مورد ویژگی‌های منابع موجود در سرزمین نمی‌توان برای استفاده و آمایش سرزمین برنامه‌ریزی کرد. از بدو به وجود آمدن نیاز برای برنامه‌ریزی و استفاده از سرزمین، گردآوری اطلاعات مهمترین مشکل برنامه‌ریزان بوده است. گردآوری اطلاعات ابتدا با آماربرداری و نمونه برداری از منابع به انجام می‌رسید، با این حال به تدریج نیاز به اطلاعات مکانی از منابع یا اطلاعات فضایی جهت برنامه‌ریزی بهتر و دقیق‌تر به وجود آمد؛ به این ترتیب برای رفع این مشکل، برنامه‌ریزان شروع به ساخت نقشه از منابع موجود در سرزمین کردند. پس از جنگ جهانی دوم و پیشرفت دانش تفسیر عکسهای هوایی، تولید نقشه از منابع آسان‌تر شد و برنامه‌ریزان دریافتند که می‌توانند با یاری نقشه‌برداران و فن‌آوری مورد استفاده توسط آنها به نتیجه بهتری دست پیدا کنند. همچنین این برنامه‌ریزان دریافتند که برای جمع‌بندی داده‌های موضوعی تک بعدی با روی هم‌گذاری نقشه‌های موضوعی تک بعدی، تقابل‌ها، تداخل‌ها و وابستگی‌های به هم پیچیده را به شکل بسیار بهتری می‌توان مشاهده، ارزیابی و تحلیل کرد. نحوه پیشرفت و تکامل استفاده از دانش نقشه‌برداری و استفاده از آن برای برنامه‌ریزی را می‌توان به صورت خلاصه در مراحل زیر نشان داد:

۱–تفسیر اتوماتیک عکس‌های هوایی برای نقشه‌سازی موضوعی منابع زمینی

۲–ساماندهی اطلاعات نقشه‌ای در گروه‌های مختلف موقعیت جغرافیایی

۳–ارزیابی توان اکولوژیکی[۴۷] پهنه‌های جغرافیایی برای کاربری‌های استفاده از سرزمین و محیط زیست

٤–مدل‌سازی اکولوژیکی برای ارزیابی رایانه‌ای قابلیت‌ها و منابع سرزمین

به این ترتیب مفهوم سامانه‌های اطلاعات جغرافیایی[۴۸] یا به اختصار GIS به وجود آمد و به سرعت مورد استقبال قرار گرفت. سامانه‌های اطلاعات جغرافیایی یک سامانه رایانه‌ای برای مدیریت

[۴۷] Ecologic

[۴۸] Geographical Information Systems

و تجزیه و تحلیل اطلاعات مکانی بوده که قابلیت جمع‌آوری، ذخیره، تجزیه و تحلیل و نمایش اطلاعات جغرافیایی (مکانی) را دارد. دقیق‌ترین تعریف برای این سامانه‌ها، مربوط به موسسه تحقیقات سامانه‌های محیطی[۴۹] - یکی از فروشندگان اصلی این سامانه‌ها در جهان - است: " سامانه‌های اطلاعات جغرافیایی مجموعه‌ای از سخت‌افزار، نرم‌افزار، داده‌های جغرافیایی و منابع انسانی است که به منظور کسب، ذخیره، به‌روزرسانی، به‌کارگیری، تحلیل و نمایش کلیه اشکال اطلاعات مرجع جغرافیایی طراحی می‌شود."

هدف نهایی یک سامانه اطلاعات جغرافیایی، پشتیبانی جهت تصمیم‌گیری‌های پایه‌گذاری‌شده بر اساس داده‌های مکانی می‌باشد و عملکرد اساسی آن بدست آوردن اطلاعاتی است که از ترکیب لایه‌های متفاوت داده‌ها با روش‌های مختلف و با دیدگاه‌های گوناگون بدست می‌آیند. این سامانه‌ها را می‌توان به یک پازل شبیه دانست که با کنار هم قرار دادن اجزای آن معنی و مفهومی پیدا می‌کند. مکان بیمارستان‌ها، پمپ بنزین‌ها، سینماها و ... تکه‌های این پازل‌اند که با کنار هم قرار دادن آنها نقشه‌ای کامل و با معنی از یک منطقه جغرافیایی بدست می‌آید. به زبان ساده هر گونه توضیحات مربوط به هر چیزی که در هر مکان متغیر یا ثابت جغرافیایی، در یک سامانه اطلاعاتی یا پایگاهی موجود است را سامانه اطلاعات جغرافیایی یا استفاده از سامانه اطلاعات جغرافیایی‌گویند. در عبارت " سامانه اطلاعات جغرافیایی" به ترتیب:

- واژه "جغرافیایی" عبارت است از موقعیت موضوع‌های داده‌ای برحسب مختصات جغرافیایی
- واژه "اطلاعات" نشان می‌دهد که داده‌ها در یک سامانه اطلاعات جغرافیایی برای ارائه دانسته‌های مفید، نه تنها به صورت نقشه‌ها و تصاویر رنگی بلکه بصورت گرافیک‌های آماری، جداول و پاسخ‌های تصویری به منظور جستجوهای عملی سازماندهی می‌شوند.
- واژه "سامانه" نیز نشان دهنده این است که یک سامانه اطلاعات جغرافیایی از چندین قسمت متصل و وابسته به یکدیگر برای کارکردهای گوناگون ساخته شده است.

[۴۹] Environmental Systems Research Institute (ESRI)

## ۶-۲- تاریخچه

برای اولین بار در اواسط دهه ۱۹۶۰ در آمریکا کار بر روی اولین سیستم اطلاعات جغرافیایی آغاز شد. در این سیستم‌ها عکس‌های هوایی، اطلاعات کشاورزی، جنگلداری، خاک، زمین شناسی و نقشه‌های مربوطه مورد استفاده قرار گرفتند.

اولین سیستم اطلاعات جغرافیایی، سیستم اطلاعات جغرافیایی ملی کانادا است که در اواخر دههٔ ۱۹۶۰ به وجود آمد و تاکنون مورد استفاده است. در ایران، اولین مرکزی که به طور رسمی استفاده از این نوع سامانه‌ها را در کشور آغاز کرد، سازمان نقشه‌برداری کشور بود که در سال ۱۳۶۹ براساس مصوبه مجلس شورای اسلامی، عهده‌دار طرح به‌کارگیری این سامانه شد.

## ۶-۳- ویژگی‌ها و وظایف سامانه اطلاعات جغرافیایی

همه سامانه‌های اطلاعات جغرافیایی دارای مجموعه مشترکی از ویژگی‌ها هستند که عبارتند از:

- این سامانه علاوه بر اطلاعات توصیفی، امکان ورود اطلاعات پیکسلی و یا برداری را از منابع مختلفی از قبیل نقشه، تصاویر هوائی و ماهواره‌ای، سامانه موقعیت‌یاب جهانی[۵۰]، تجهیزات نقشه‌برداری و غیره دارد.
- این سامانه امکان انجام تحلیل، پردازش پرسش و پاسخ‌های[۵۱] مکانی مورد نیاز کاربر را دارد.
- این سامانه امکان ارائه نتایج در قالب نقشه، گزارش، جدول و نمودار را دارد.
- در طراحی و تولید این سامانه‌ها از مجموعه فناوری‌های مهندسی نرم‌افزار، مهندسی اطلاعات (مدل داده) و مهندسی سامانه‌های اطلاعات جغرافیایی برای نیل به خصوصیات فوق استفاده می‌شود.

وظایف یک سامانه اطلاعات جغرافیایی را می‌توان در چهار گروه کلی دسته بندی نمود :

۱- کسب و جمع‌آوری داده‌ها

۲- نگهداری و مدیریت داده‌ها

۳- دستکاری[۵۲] و تجزیه و تحلیل

---

[۵۰] Global Positionening System (GPS)

[۵۱] Query

[۵۲] Manipulation

٤ – ارائه

روش رقومی[٥٣] تشریح هندسی و موضوعی عوارض امکان انتخاب و دسته‌بندی پ عوارض را از دیدگاه‌های مختلف میسر می‌سازد. داده‌ها می‌توانند از دیدگاههای مختلف مورد پرسش و تجزیه و تحلیل قرار گیرند؛ نظیر مالکیت ، شبکه آب رسانی ، فضای سبز، آمار جمعیت ، کاربری زمین ، طراحی راههای عبور و مرور. نتایج این پرسش و تجزیه و تحلیل‌ها می‌توانند به صورت نقشه، جدول یا گزارش ارائه شوند. جدول ارکان و وظایف یک سامانه اطلاعات جغرافیایی را نمایش می‌دهد.

**جدول ۲ : ارکان و وظایف سامانه اطلاعاتی جغرافیایی**

| رکن | وظایف |
|---|---|
| سخت افزار | کسب و جمع‌آوری داده‌ها |
| نرم‌افزار | نگهداری و مدیریت داده‌ها |
| داده | دستکاری و تجزیه و تحلیل داده‌ها |
| کاربر | ارائه |

## ۶–۴– انواع داده در سامانه اطلاعات جغرافیایی

از آنجایی که ارتباط مستقیمی بین چگونگی نمایش و پردازش یک نوع داده خاص وجود دارد می‌توان سه نوع داده در سامانه اطلاعات جغرافیایی در نظر گرفت: داده‌های هندسی، گرافیکی و توصیفی.

### ۶–۴–۱– داده‌های هندسی

موقعیت و شکل عوارض از طریق داده‌های هندسی و در یک سامانه مختصات معین تشریح می‌شوند. دو ساختار یا مدل داده برای ارائه داده‌های هندسی وجود دارند:

- ساختار برداری[٥٤]

در این ساختار موقعیت هر نقطه به طور دقیق با یک جفت مختصات (x,y) در یک سامانه مختصات معین ارائه می‌گردد، ضمن آنکه روابط همسایگی را نیز میتوان به آن افزود. منظور از

[٥٣] Digital

[٥٤] Vector

روابط همسایگی به طور مثال این است که نقاط آغاز و پایان یک خط و همچنین سطوح مجاور آنها کدامند. به دلیل دقت هندسی بسیار بالای ساختار برداری در ارائه موقعیت عوارض، این ساختار برای تشریح موقعیت مکانی عوارض نقطه‌ای، خطی و همچنین نقشه‌های بزرگ مقیاس (۱۰۰۰۰: ۱ – ۱۰۰ : ۱) بسیار مناسب می‌باشد. داده‌های هندسی برداری عمدتاً از طریق رقومی‌گر[۵۵]، برداشت‌های نقشه برداری و GPS کسب می‌شوند. شکل ۱ نمونه‌ای از انواع داده‌های برداری را نمایش می‌دهد.

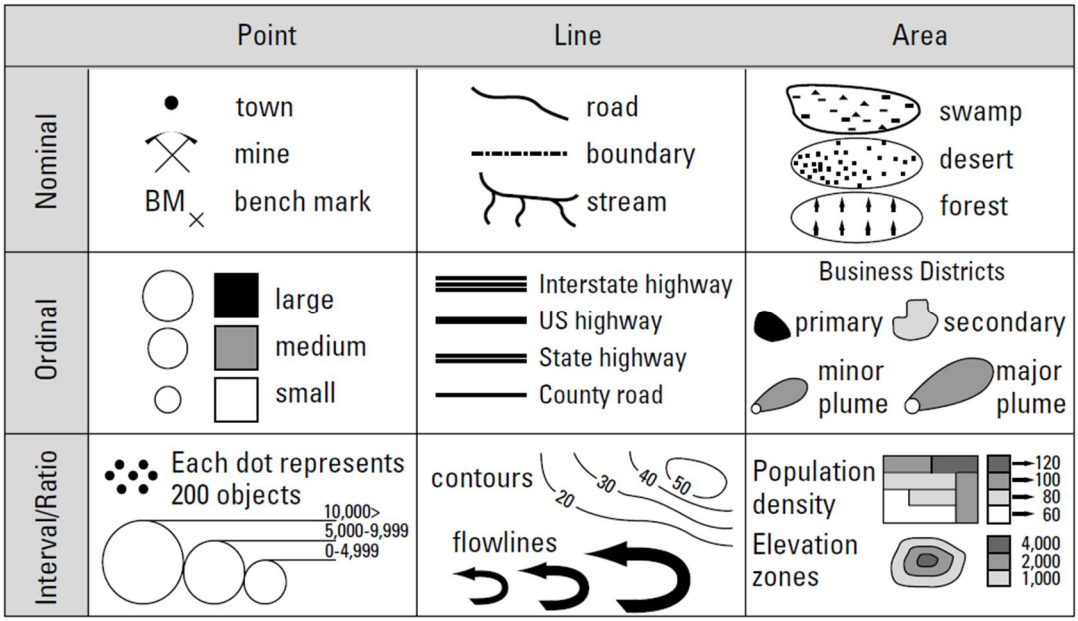
• ساختار رستری[۵۶]

روش تشریح و ارائه موقعیت مکانی عوارض با ساختار رستری جدیدتر از روش برداری می‌باشد و بر خلاف آن بر اساس سطح (به جای نقطه) استوار است. کوچکترین جزء پایه هندسی در این ساختار سلول می‌باشد که معمولاً به شکل مربع و به صورت ستون و ردیف‌هایی در یک ماتریس همسان ارائه می‌گردد. بین سلول‌های یک داده رستری هیچگونه ارتباط منطقی وجود ندارد. هر سلول تنها می‌تواند دارای یک ارزش باشد که نمایانگر یک ویژگی نظیر ارتفاع، نوع خاک و پوشش گیاهی خواهد بود. ساختار رستری عمدتاً برای ارائه نقشه‌هایی در مقیاس‌های کوچکتر از ۱۰۰۰۰ : ۱ بکار می‌رود. کسب داده‌های رستری با استفاده از اسکنر[۵۷] صورت می‌گیرد. تصاویر ماهواره‌ای دارای ساختار رستری می‌باشند و از همین روست که ارتباط بسیار نزدیکی با سامانه اطلاعات جغرافیایی پیدا نموده است چرا که این تصاویر یا نتایج حاصل از تفسیر آنها می‌توانند مستقیماً وارد این سامانه‌ها شوند. شکل ۲ نمونه‌ای از انواع داده رستری را نمایش می‌دهد.

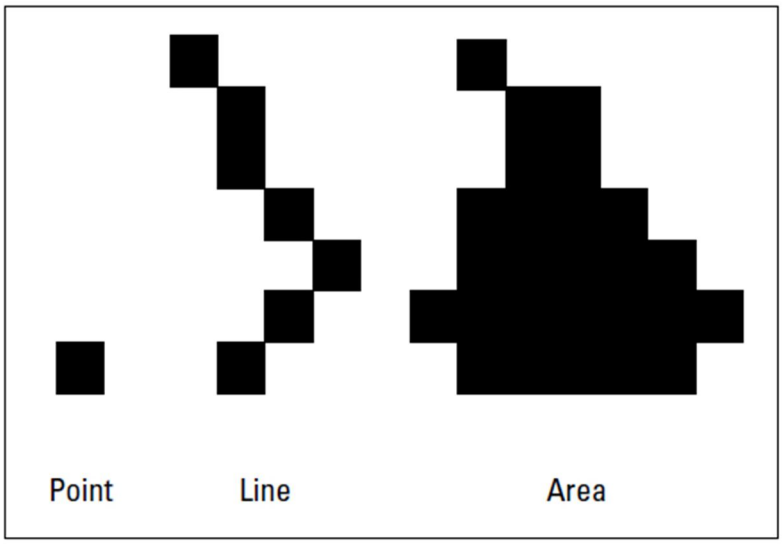
[۵۵] Digitalizer

[۵۶] Raster

[۵۷] Scanner




**شکل ۱ : انواع داده برداری و استفاده‌های آن**




**شکل ۲ : انواع داده رستری**

### ۶-۴-۲- داده‌های گرافیکی

داده‌ای تشریحی گرافیکی مربوط به چگونگی و نحوه نمایان‌سازی عوارض نقشه‌های موضوعی مختلف (داده‌های هندسی و توصیفی) بر روی سخت‌افزارهای بخش خروجی سامانه اطلاعات جغرافیایی می‌باشند. رنگ، طیف خاکستری، علائم، هاشور، نقطه چین، خط نقطه، خط و تعیین موقعیت متن از جمله داده‌های گرافیکی می‌باشند. داده‌های گرافیکی فوق و داده‌های هندسی (برداری یا رستری) با هم درآمیخته می‌شوند تا داده‌های موضوعی را به همراه موقعیت مکانی آنها

چه به صورت آنالوگ بر روی نقشه و چه به صورت رقومی بر روی صفحه نمایش رایانه به نمایش درآورند. ترکیب داده‌های گرافیکی با داده‌های هندسی از نوع برداری را گرافیک برداری و با داده‌های رستری را گرافیک رستری می‌نامند. گرافیک برداری ضمن آنکه از حجم فایل کمتری نسبت به گرافیک رستری برخوردار است، توانایی‌های بسیار زیادی را در راستای تهیه یک خروجی مناسب نظیر ضخامت خطوط و استفاده از علامت و هاشورهای مختلف در اختیار کاربر قرار می‌دهد.

### ۶-۴-۳- داده‌های توصیفی

اداره و تجزیه و تحلیل همزمان داده‌های هندسی و توصیفی، مشخصه بارز سامانه‌های اطلاعات جغرافیایی می‌باشد. داده‌های توصیفی که تشریحی و موضوعی نیز نامیده می‌شوند ارائه دهنده تمامی اجزای غیر هندسی نظیر نام (مالک، شهر)، شماره (پارسل، خانه)، اندازه‌های کمی و کیفی (شوری خاک، حجم، تعداد و کیفیت درختان جنگل)، نوع (سنگ مادر و خاک) و خلاصه هر نوع مشخصه مرتبط با کاربرد نقشه می‌باشد.

تهیه داده‌های توصیفی امری است جدا از وظایف سامانه اطلاعات جغرافیایی و باید از طریق روش‌های موضوع مربوطه صورت پذیرد. این داده‌ها می‌توانند به صورت آنالوگ (نقشه‌های کاغذی، یادداشت و گزارش) و رقومی (بانک داده سامانه اطلاعاتی، پرونده داده) در اختیار کاربر قرار گیرند. به نقشه‌های موضوعی رقومی با ساختار برداری داده‌های برداری اطلاق می‌شود. این چنین نقشه‌ها چنانچه دارای ساختار رستری باشند ، داده رستری نامیده می‌شوند.

## ۶-۵- کسب و وارد سازی داده در سامانه اطلاعات جغرافیایی

کسب و وارد سازی داده در سامانه اطلاعات جغرافیایی را به لحاظ منبع و مأخذ می‌توان به دو بخش تقسیم نمود:

### ۶-۵-۱- کسب اولیه ( مستقیم ) داده

کسب مستقیم داده شامل آن دسته از روش‌هایی می‌شوند که به تشریح مکانی و توصیفی عوارض به طور بلاواسطه در دنیای حقیقی و یا بر اساس تصاویر آنها می‌پردازند. مهمترین این روش‌ها عبارتند از: نقشه برداری، GPS، فتوگرامتری، تفسیر عکس‌های هوایی، سنجش از دور، ثبت دائم

مشخصات (نظیر میزان و کیفیت منابع آب)، کار میدانی، پرسش و مصاحبه ( بویژه در زمینه‌های اجتماعی – اقتصادی )

### ۶-۵-۲- کسب ثانویه داده‌ها

منظور از کسب ثانویه داده رقومی‌سازی داده‌هایی است که به روشهای گوناگون از پیش تهیه شده و معمولاً به صورت نقشه‌های کاغذی و آماری در دسترس می‌باشند. بررسی کیفیت داده‌های موجود و رقومی‌سازی آنها بسیار متداول و نقش بزرگی را در سامانه‌های اطلاعات جغرافیایی بازی می‌کند. روشهای رقومی‌سازی عبارتند از: رقومی سازی دستی، نیمه خودکار، خودکار (اسکن)، واردسازی داده‌های حرفی و عددی.

- رقومی سازی دستی

این روش متداول‌ترین شیوه رقومی‌سازی است. اپراتور نقش بسیار مهمی در انجام این کار دارد. او یکایک نقاط، خطوط و سطوح را شناسایی و ضمن کدگذاری هندسی آنها اطلاعات تشریحی شان را نیز به سامانه اطلاعات جغرافیایی وارد می‌نماید. این روش در مورد نقشه‌هایی با تنوع زیاد عوارض، هندسه نامنظم عوارض و علائم زیاد، بسیار کارا خواهد بود. نکته منفی این روش نسبت به سایر روش‌ها نیاز زیاد به زمان و نیروی انسانی و یکنواختی آن برای اپراتور می‌باشد.

- رقومی‌سازی نیمه خودکار

این روش بر اساس فرآیند تعقیب و پی‌جویی خطوط طرح‌ریزی شده است. به این منظور ابتدا باید نقشه را اسکن و بر روی میکروفیلم منتقل نمود. اپراتور در نقطه آغاز یک خط، ارزش توصیفی مربوط به آن را مشخص می‌نماید و از سامانه می‌خواهد تا بر اساس این روش اجزاء خط را پی‌جویی و شناسایی نماید. به‌کارگیری این روش نیازمند سرمایه‌گذاری سنگین می‌باشد، ضمن آنکه اپراتور نیز باید به طور پیوسته و دائم فرآیند را هدایت نماید.

- رقومی‌سازی خودکار (اسکن)

یک نقشه را می‌توان کاملاً به طور خودکار و بدون نیاز به اپراتور با استفاده از اسکنر رقومی نمود. حاصل این فرآیند نقشه‌ای خواهد بود با ساختار رستری که مجموعه‌ای است از سلول‌ها و بدون توپولوژی و هر گونه ارتباط با سلول‌های مجاور و ساختار عوارض. اندازه سلول یا به بیانی دیگر تعداد سلول در واحد طول یا سطح را می‌توان به هنگام انجام عمل

اسکن تعیین نمود. عمل رقومی‌سازی با سرعت بالایی انجام و نقشه رستری حاصله از دقت مکانی بسیار بالایی برخوردار است. این چنین نقشه‌هایی می‌توانند به خوبی به عنوان زمینه در نمایش داده‌ها به کار روند.

## ۶-۶- خروجی تجزیه و تحلیل

یکی از اصلی‌ترین خروجی‌ها و اهداف یک سامانه اطلاعات جغرافیایی انجام یک تجزیه و تحلیل بر روی داده‌ها است . در پایان هر تجزیه و تحلیل باید تلاش گردد تا نتایج حاصله به بهترین شکل ممکن ارائه شوند. یک تجزیه و تحلیل بدون ارائه خروجی مناسب عملاً ناموفق قلمداد خواهد شد. خروجی‌های سامانه اطلاعات جغرافیایی را می‌توان بر اساس ماهیت شان به دو دسته تقسیم نمود : خروجی برای رویت انسان و خروجی سازگار با ماشین. دسته اول خروجی‌ها از نوع کارتوگرافی بوده و بیشتر مد نظر هستند. این دسته از خروجی‌ها می‌توانند به صورت رقومی (کپی نرم) که تنها قابل نمایش بر روی صفحه رایانه‌اند و همچنین به صورت کپی سخت (بر روی کاغذ) باشند. خروجی‌های سازگار با ماشین از نوع غیرکارتوگرافی بوده و عمدتاً شامل تولیدات عملیات کسب داده، تغییر و تبدیل و تجزیه و تحلیل‌های اولیه می‌باشند. این دسته از خروجی‌ها مورد پردازش‌های کارتوگرافی قرار نمی‌گیرند و به عنوان داده‌های ورودی در دیگر تجزیه و تحلیل‌ها کاربرد خواهند داشت. خروجی‌های نقشه‌ای عمدتاً از نوع نقشه‌های موضوعی می‌باشند. نقشه موضوعی خروجی سامانه اطلاعات جغرافیایی، نقشه راه حل نیز نامیده می‌شود چرا که نقشه خروجی حاصل از یک تجزیه و تحلیل در این سامانه پاسخی به یک سوال مشخص یا حاصل یک فرآیند تصمیم‌گیری می‌باشد. به عنوان مثال می‌توان از نقشه مناطق مناسب دفن زباله و همچنین نقشه مسیر مناسب انتقال نیرو نام برد.

گام اصلی در تهیه یک نقشه موضوعی به عنوان خروجی سامانه اطلاعات جغرافیایی از نوع کارتوگرافی، طراحی نقشه می‌باشد. این طراحی شامل تعیین مقیاس مناسب، مشخص نمودن عوارض که باید بر روی نقشه ظاهر شوند، انتخاب نشانه‌ها و عوارض گرافیکی، جایابی آنها بر روی نقشه به منظور نشان دادن هر چه بهتر عوارض مهم و روابط مکانی آنها در منطقه مورد مطالعه می‌باشد. تمامی عوارض که بر روی این چنین خروجی‌هایی آورده می‌شوند باید گویای مطلبی در رابطه با موضوع نقشه باشند و از درج عوارض غیرضروری بر روی نقشه باید اجتناب گردد. به

علاوه باید راهنمایی[۵۸] نیز برای نقشه طراحی نمود. حتی در مواردی که از یک تصویر سنجش از دوری به عنوان زمینه در نقشه خروجی استفاده گردد، ترجیحاً باید راهنمایی برای تفسیر ساده و اولیه آن تعیین نمود، چرا که بینندگان نقشه خروجی ممکن است قادر به درک مفهوم طیف‌های خاکستری و یا رنگ‌های این تصاویر نباشند.

## ۶-۷- کیفیت داده‌ها

کیفیت تابعی است از صحت، دقت و نامعلومی. دقت مربوط به فرآیند اندازه‌گیری و ابزارهای آن می‌باشد و توانایی کسب نتایج یکسان در اندازه گیری‌های مکرر است. دقت بالا تضمین کننده صحت زیاد نمی‌باشد و بالعکس.

نا معلومی معادل نامشخص بودن داده از نظر مکانی، توصیفی و زمانی (تاریخ) است. هر چه میزان خطا در داده کمتر باشد آن داده از کیفیت بالاتری برخوردار خواهد بود. ایده‌آل است که در منطقه مورد مطالعه پوششی یکنواخت و کامل از تمامی داده‌های مورد نیاز وجود داشته باشد. به این مفهوم که اولاً انواع داده مورد نیاز برای کل منطقه مورد نظر در دسترس باشد. ثانیاً تمامی داده‌ها از ویژگی‌های همسانی برخوردار باشند. منظور از ویژگی مواردی نظیر صحت مکانی و توصیفی، سن، قدرت تفکیک، مقیاس و ... می‌باشد.

نقشه‌های بزرگ مقیاس مقدار بیشتری از جزییات موضوعی را با دقت مکانی بیشتری در مقایسه با نقشه‌های کوچک مقیاس ارائه می‌دهند. یک پروژه در مقیاس متناسب با اهداف آن اجرا می‌شود. استفاده از داده‌هایی با مقایس کوچکتر از مبنا موجب بروز خطا می‌گردد. ضمن آنکه استفاده از داده‌هایی با مقیاس بزرگ‌تر از مقیاس مبنا بر حجم پردازش‌ها می‌افزاید.

## ۶-۸- تفسیر و جمع بندی داده‌ها

سامانه‌های اطلاعات جغرافیایی از این مزیت برخوردارند که می‌توان داده را به صورت خام وارد سامانه نمود. اما این امر به این معنا نیست که داده‌های خام را بتوان به همان صورت و یا حتی پردازش شده برای ارزیابی و برنامه‌ریزی به کار گرفت . داده‌ها چه به صورت نقشه و یا چه به صورت جدول از چنان گستردگی و گوناگونی برخوردارند که بدون جمع‌بندی آنها تصمیم‌گیری راجع به آنها اگر محال نباشد گیج کننده و بغرنج است. بنابراین داده‌های پردازش شده باید نخست

[۵۸] Legend

تجزیه و تحلیل شوند سپس بر حسب هدف تصمیم‌گیری تفسیر گردند و در نهایت به صورت گروه‌های منظم قابل استفاده در تصمیم‌گیری درآیند .

بسیاری از استفاده‌کنندگان سامانه‌های اطلاعات جغرافیایی منطقه مورد مطالعه را در یک ماتریس شبکه‌بندی می‌نمایند و پس از ورود و پردازش داده‌ها در ماتریس شبکه‌ها، انتظار دارند که برای هر شبکه که آمیخته‌ای از داده‌های گوناگون و ناهمگن است با پرسش و پاسخ و یا حتی مدل به ارزیابی و برنامه‌ریزی و تصمیم‌گیری برای آن شبکه و منطقه مورد بررسی دست یابند. ناهمگنی داده‌ها در شبکه دلیل اصلی عدم موفقیت بوده است .

پهنه‌بندی برای دستیابی به ساختارهای مشابه و همگن از داده‌ها بر روی نقشه انجام می‌پذیرد. بدین ترتیب بعد از جمع بندی داده‌ها با پرسش و پاسخ و یا از همه مهم‌تر با مدل‌سازی آسان‌تر، دقیق‌تر و زودتر می‌توان به ارزیابی توان و نیاز، برنامه‌ریزی و در آخر به تصمیم‌گیری دست یافت.

## ۶-۹- ارتباط سنجش از دور و سامانه اطلاعات جغرافیایی

تمامی مطالعات و تجزیه و تحلیل‌ها در منابع طبیعی بر پایه و اساس اطلاعات محیطی استوارند. از آنجایی که این مطالعات به طور فزاینده‌ای با استفاده از سامانه‌های اطلاعات جغرافیایی انجام می‌شوند و بخش مهمی از اطلاعات مورد نیاز نیز از طریق عملیات دورسنجی کسب می‌شوند، این دو فن‌آوری نزدیکی زیادی با یکدیگر دارند. در واقع سنجش از دور به عنوان منبع تأمین بسیاری از داده‌های مورد نیاز و GIS به عنوان سامانه‌ای که عمدتاً مدیریت تحلیل و ارائه مجدد اطلاعات را بر عهده دارند ارتباط تنگاتنگی پیدا نموده‌اند. ماهیت رقومی داده‌های سنجش از دور و اطلاعات حاصل از آنها که قابلیت ورود مستقیم آنها به سامانه‌های اطلاعات جغرافیایی را میسر ساخته است این ارتباط را تسهیل نموده است. ضرورت به کارگیری داده‌های به هنگام در مطالعات و برنامه‌ریزی‌ها و امکاناتی که سنجش از دور در این ارتباط فراهم می‌سازد، پیوند این دو فن‌آوری را مستحکم‌تر می‌سازد.

علاوه بر دلایل و زمینه‌های یاد شده برای ارتباط بین سنجش از دور و سامانه اطلاعات جغرافیایی تشابه و هم پوشانی‌هایی نیز بین قابلیت و توانایی‌های (توابع) این دو سامانه وجود دارد که زمینه‌های تلفیق، جمع‌بندی و یکپارچگی این دو مقوله را فراهم می‌سازد. از جمله این تشابه‌ها می‌توان موارد ذیل را برشمرد:

- خروجی

خروجی به عنوان رکن چهارم وظایف سامانه اطلاعات جغرافیایی نیز به نوعی در سامانه‌های تجزیه و تحلیل سنجش از دور وجود دارد. در واقع در سنجش از دور نیز همواره این ضرورت محسوس بوده است که تصاویر و نتایج تجزیه و تحلیل آنها با کیفیتی مناسب و مطابق با اصول نقشه‌برداری[۵۹] ارائه شوند.

- تهیه مدل رقومی ارتفاع

هر چند مدل رقومی زمین معمولاً از طریق درون‌یابی داده‌های ارتفاعی در سامانه اطلاعات جغرافیایی تهیه می‌شوند ولی امروزه امکان آن از طریق تحلیل تصاویر ماهواره‌ای ۳ بعدی[۶۰] فراهم شده است .

- توابع

بعضی از توابع نظیر طبقه‌بندی، فیلتر، تبدیل، تغییر پروژکسیون و تطابق هندسی در سنجش از دور و سامانه اطلاعات جغرافیایی به کار گرفته می‌شوند.

## ۶-۱۰- کاربردهای سامانه اطلاعات جغرافیایی

کمتر بانک اطلاعاتی را می‌توان نام برد که حداقل بخشی از اطلاعات آن به نوعی به مکان وابسته نباشد. فهرست زیر شامل تعدادی از شناخته شده‌ترین کاربردهای سامانه اطلاعات جغرافیایی در دیگر زمینه‌های صنعت است:

- نقشه برداری

کاربرد پایه این استانداردها و ابزارها در تهیه نقشه‌های شهری و برون شهری بوده‌است.

- علوم زمین

یکی از مهم‌ترین مسائل کاربردی سامانه‌های اطلاعات جغرافیایی در علوم زمین، انجام هم‌پوشانی‌ها در راستای تعیین پهنه‌بندی‌ها می‌باشد؛ به عنوان مثال در یک پهنه‌بندی هدف تعیین مناطق با خطر رانش زمین می‌باشد. برای رسیدن به این هدف، ابتدا باید عوامل موثر در رانش را شناسایی کرد، از جمله این عوامل می‌توان به تراکم پوشش گیاهی، جنس زمین، شیب توپوگرافی، شیب لایه‌های زمین و شدت بارندگی اشاره کرد.

---

[۵۹] Cartography

[۶۰] Stereoscopic

پس از شناسایی عوامل، باید نقشه‌های مورد نیاز تهیه گردد و سپس با هم‌پوشانی این لایه‌ها در نرم‌افزارهای سامانه اطلاعات جغرافیایی می‌توان به نقشه‌ای رسید که در آن منطقه مورد مطالعه خود به پهنه‌های ریز تقسیم شده و از طرفی در بانک اطلاعاتی مشخصات هر پهنه یعنی اجداد سازنده آن پهنه از نظر عوامل مختلف مشخص می‌باشد. حال باید در بانک اطلاعاتی به هر عامل وزن مربوطه را داده و سپس با جمع و ضرب کردن عوامل به اعداد نهایی رسید. در مرحله بعد باید تعیین نمود که چه عددی نشان دهنده پهنه‌های پرخطر می‌باشد. پس از دسته‌بندی[۶۱] نقشه، اینک وظیفه تیم زمین‌شناس است که با بازدید میدانی به بررسی مناطق پرداخته و در صورت لزوم با اصلاح وزن‌ها و ضرایب به نقشه نهایی دست یابند.

- جغرافیا
- مهندسی معدن
- مسائل اکتشاف معادن، تهیه نقشه و مدل از ذخایر معدنی و محاسبات آن و...
- منابع طبیعی
- سنجش از دور
- هواشناسی
- محیط زیست
- مخابرات
- شهرسازی
- کشاورزی دقیق
- جغرافیای سلامت

## ٦-١١- فن‌آوری‌های مرتبط با سامانه اطلاعات جغرافیایی

برخی از فن‌آوری‌هایی که به نوعی با سامانه‌های اطلاعات جغرافیایی ارتباط دارند عبارتند از:

- سامانه موقعیت‌یاب جهانی[۶۲]

[۶۱] Classification

[۶۲] Global Positioning System (GPS)

سامانه موقعیت‌یاب جهانی یک سامانه ناوبری مبتنی بر ماهواره است که مکان و زمان را در هر شرایط جوی بر روی زمین یا نزدیک زمین ارائه می‌دهد؛ تنها شرط عملکرد این سامانه وجود یک خط مستقیم مسدود نشده بین کاربر و ماهواره‌های این سامانه است.

- سنجش از دور[۶۳]

به عملیات جمع‌آوری اطلاعات در مورد یک شی یا پدیده بدون برقراری تماس فیزیکی با آن را سنجش از دور گویند. امروزه این اصطلاح به شناسایی و دسته‌بندی اشیای روی زمین با استفاده از حس‌گرهای[۶۴] هوایی و انتشار سیگنال‌ها[۶۵] می باشد.

- سامانه‌های اطلاعات جغرافیایی وبی[۶۶]

یکی از در حال رشدترین زمینه‌ها در زمینه سامانه‌های اطلاعات جغرافیایی است و همانطور که از نام آن برمی‌آید به سامانه‌های اطلاعات جعغرافیایی گفته می‌شود که اطلاعات خود را از طریق وب ارائه می‌دهند؛ این اطلاعات می‌توانند در قالب یک سایت، تصاویر و یا سرویس‌های وبی ارائه شوند.

- ناوبری[۶۷]

ناوبری به عملیات نظارت و کنترل حرکت هواپیماها و وسایل نقلیه از نقطه‌ای به نقطه دیگر گفته می‌شود. تمامی روش‌های استفاده شده در ناوبری شامل تشخیص مکان ناوبر در مقایسه با مکان‌ها و الگوهای شناخته شده می‌باشد.

## ۶-۱۲- ابزارهای نرم افزاری مطرح سامانه اطلاعات جغرافیایی

در دنیای نرم‌افزار ابزارها و نرم‌افزارهای مختلفی درحوزه سامانه‌های اطلاعات جغرافیایی ایجاد و تولید شده است. این نرم‌افزارها دارای طیف وسیعی از مولفه‌های نرم‌افزاری[۶۸] کوچک، تا نرم‌افزارهای مستقل برای استفاده شخصی و کاربردهای کوچک، تا نرم‌افزارهای بسیار بزرگ و تشکیل شده از چندین زیر سیستم می‌باشد. همچنین بیشتر سامانه‌های مدیریت بانک اطلاعاتی[۶۹]

---

[۶۳] Remote Sensing

[۶۴] Sensors

[۶۵] Propagated Signals

[۶۶] WebGIS

[۶۷] Navigation

[۶۸] Software Components

[۶۹] Database Management Systems (DBMS)

موجود از نوع داده‌های مکانی و جغرافیایی پشتیبانی می‌کنند و سیستم‌های اطلاعات جغرافیایی می‌توانند داده‌های با حجم بالای خود را در این سامانه‌ها ذخیره کنند. در برخی موارد سامانه های مدیریت بانک اطلاعاتی بسیاری از توابع و امکانات یک سیستم اطلاعات جغرافیایی را به تنهایی پیاده‌سازی کرده‌اند و کاربران بدون نیاز به نرم‌افزاری مجزا می‌توانند عملیات جغرافیایی و مکانی خود را توسط این سامانه‌ها انجام دهند.

برخی از سامانه‌های اطلاعات جغرافیایی معروف و شناخته شده عبارتند از:

- ArcView
- ArcGIS : شناخته‌شده‌ترین مجموعه نرم‌افزار در حوزه سامانه اطلاعات جغرافیایی می‌باشد.
- ILWIS
- CARIS
- GRASS : معروف‌ترین نرم‌افزار منبع‌باز[۷۰] در زمینه سامانه اطلاعات جغرافایی می‌باشد.
- IDRISI
- LANDSERF
- SAGA
- SAMT
- QGIS

شناخته‌شده‌ترین سامانه‌های مدیریت بانک اطلاعاتی که از داده‌های مکانی و جغرافیایی پشتیبانی می‌کنند عبارتند از:

- Oracle Database
- Microsoft SQL Server : نسخه ۲۰۰۸ به بعد
- **PostGIS** : این سامانه مدیریت بانک اطلاعاتی که مبتنی بر سامانه مدیریت بانک اطلاعاتی PostGreSQL می‌باشد منبع‌باز بوده و از طرف بسیاری از کارشناسان به عنوان قوی‌ترین پایگاه داده در زمینه سامانه اطلاعات جغرافیایی شناخته می‌شود.

برخی از ابزارهای معروفی که برای تولید سامانه‌های اطلاعات جغرافیایی تحت وب عبارتند از:

- **ArcIMS** : یکی از زیر مجموعه‌های ArcGIS
- **MapGuide**
- **MapServer**
- **GeoServer**

[۷۰] Open Source

در فضای اینترنت سرویس‌های وبی[۷۱] فراوانی برای ارائه نقشه و تصاویر ماهواره‌ای وجود دارد. این سرویس‌ها به نوعی در مجموعه سامانه‌های اطلاعات جغرافیایی قرار می‌گیرند. برخی از این سرویس‌ها عبارتند از:

- **GoogleMaps**
- **Yahoo Maps**
- **Bing Maps**
- **MapQuest**
- **OpenStreetMap**
- **Nokia Maps**

[۷۱] Web Services

# فصل ۷- معرفی نرم افزار ArcGIS

مجموعه نرم‌افزار ArcGIS یک چهارچوب[۷۲] مقیاس‌پذیر[۷۳] برای پیاده‌سازی سامانه‌های اطلاعات جغرافیایی برای کاربردهای تک کاربره[۷۴] و چند کاربره[۷۵]، سامانه‌های قرار گرفته بر روی سرور[۷۶] برای دسترسی در یک سازمان یا وب و استفاده‌های میدانی[۷۷] می‌باشد. این مجموعه خانواده به هم پیوسته‌ای[۷۸] از نرم‌افزارها می‌باشد که برای پیاده‌سازی یک سامانه اطلاعات جغرافیایی کامل مورد استفاده قرار می‌گیرد. شکل مجموعه نرم‌افزارهای ArcGIS و ارتباطات آنها با یکدیگر نمایش می‌دهد. مجموعه ArcGIS از چهار چهارچوب اصلی تشکیل شده است:

- **ArcGIS Desktop**

مجموعه به هم پیوسته‌ای از نرم‌افزارهای تخصصی در زمینه سامانه اطلاعات جغرافیایی که از سه نرم‌افزار ArcView، ArcEditor و ArcInfo تشکیل شده است.

- **Server GIS**

مجموعه‌ای از سه نرم‌افزار ArcIMS، ArcGIS Server و ArcGIS Image Server می‌باشد که برای اجرا به عنوان سرور تهیه شده‌اند.

- **Developer GIS**

شامل اجزای نرم‌افزاری جاسازی شده‌ای[۷۹] می‌شود که امکان گسترش[۸۰] نرم‌افزارهای مجموعه ArcGIS Desktop، ایجاد نرم‌افزارهای سفارشی[۸۱]، سرویس‌های نرم‌افزاری[۸۲] مبتنی بر سامانه‌های

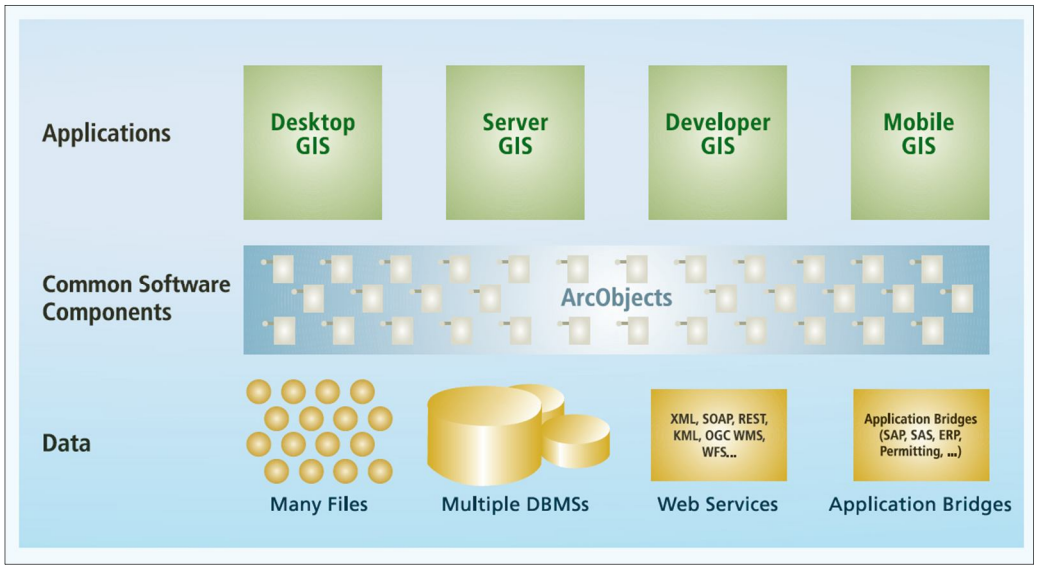
---

[۷۲] Framework
[۷۳] Scalable
[۷۴] Single User
[۷۵] Multi User
[۷۶] Server Applications
[۷۷] Field
[۷۸] Integrated
[۷۹] Embedded
[۸۰] Extend
[۸۱] Custom Applications
[۸۲] Software Services

اطلاعات جغرافیایی، نرم‌افزارهای تحت وب و نرم افزارهای تلفن‌های همراه را در اختیار توسعه‌دهندگان قرار می‌دهد.

- **Mobile GIS**

شامل نرم‌افزارهای ArcPad و ArcGIS Mobile برای پردازش‌های میدانی[۸۳] می‌شود.




**شکل ۳ : مجموعه نرم‌افزارهای ArcGIS**

## ۷-۱- ArcGIS Desktop

نرم‌افزارهای ArcGIS Desktop مجموعه فراگیر و جامعی از نرم‌افزارهای تخصصی در زمینه سامانه‌های اطلاعات جغرافیایی می‌شود که برای حل مسائل، اجرای ماموریت‌ها، بالا بردن کارایی، تصمیم‌گیری بهتر و مبادله، تجسم[۸۴] و درک اطلاعات جغرافیایی[۸۵] مورد استفاده قرار می‌گیرد.

برای انجام این فعالیت‌ها کاربران با استفاده از ArcGIS Desktop مجموعه‌ای از وظایف را که شامل موارد زیر می‌شود انجام می‌دهند:

- کار با نقشه‌ها
- گردآوری، ویرایش و نگهداری داده های جغرافیایی
- خودکارسازی وظایف کاری با استفاده از پردازش‌های مکانی[۸۶]

---

[۸۳] Field Computing

[۸۴] Visualize

[۸۵] Geographical Information

- تحلیل و مدل‌سازی یا استفاده از پردازش‌های مکانی
- تجسم و نمایش نتایج بر روی نقشه‌ها، نماهای سه بعدی[۸۷] و صفحات پویای مبتنی بر زمان
- مدیریت و نگهداری بانک‌های اطلاعات جغرافیایی چند کاربره
- ارائه نتایج و منابع[۸۸] مکانی به طیف وسیعی از کاربران و نرم‌افزارها
- مستندسازی[۸۹] و فهرست‌بندی[۹۰] نتایج، مجموعه‌داده‌های مکانی، نقشه‌ها، کره‌ها[۹۱]، نوشتارهای پردازش مکانی[۹۲]، سرویس‌های سامانه اطلاعات جغرافیایی[۹۳]، نرم‌افزارها و غیره
- ایجاد نرم‌افزارهای سفارشی برای با اشتراک‌گذاری[۹۴] سامانه اطلاعات جغرافیایی

نرم‌افزارهای ArcGIS Desktop سکوی کاری اولیه متخصصان GIS برای مدیریت گردش‌کارها و پروژه‌های مبتنی بر سامانه‌های اطلاعات جغرافیایی و ایجاد داده‌ها، نقشه‌ها، مدل‌ها و نرم‌افزارهای مورد استفاده آنها می‌باشد. این مجموعه نقطه شروع و زیربنای[۹۵] راه‌اندازی سامانه اطلاعات جغرافیایی در یک سازمان و بر روی وب می‌باشد.

مجموعه ArcGIS Desktop شامل نرم‌افزارهای ArcCatalog، ArcMap، ArcGlobe، ArcScene، ArcToolbox و ModelBuilder می‌باشد. کاربران با استفاده از این نرم‌افزارها در کنار یکدیگر می‌توانند هرگونه وظیفه‌ای را در حوزه سامانه‌های اطلاعات جغرافیایی از ساده تا پیچیده انجام دهند. این مجموعه نرم‌افزاری بسیار مقیاس‌پذیر بوده و نیازهای کاربران مختلف را پوشش می‌دهد. این نرم‌افزار در سه رده عملیاتی وجود دارد:

---

[۸۶] Geoprocessing

[۸۷] ۳d Views

[۸۸] Resources

[۸۹] Documenting

[۹۰] Cataloging

[۹۱] Globes

[۹۲] Geoprocessing Scripts

[۹۳] GIS Services

[۹۴] Share

[۹۵] foundation

۱. **ArcView**

تمرکز این رده به صورت جامع بر استفاده از داده‌ها، کار با نقشه‌ها و تحلیل اطلاعات می‌باشد.

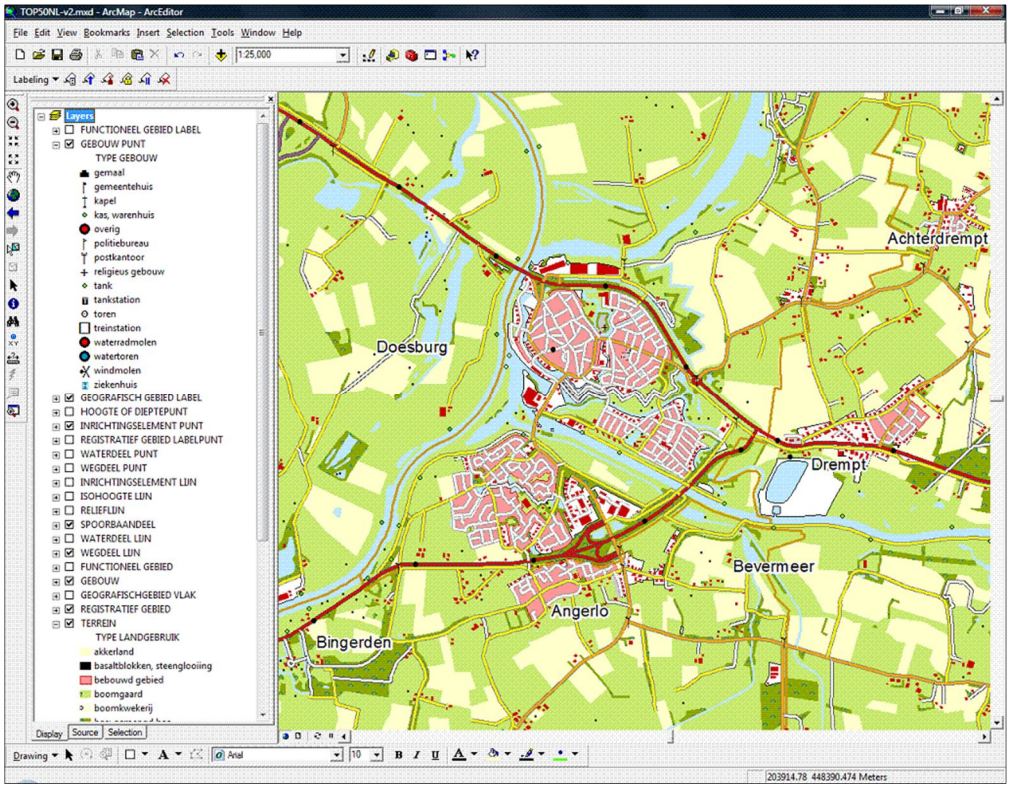
۲. **ArcEditor**

این رده علاوه بر قابلیت‌های رده قبل، دارای امکانات پیشرفته‌ای برای ویرایش و ایجاد داده برای بانک های اطلاعات جغرافیایی می‌باشد.

۳. **ArcInfo**

این رده دارای کامل‌ترین و تخصصی‌ترین ابزارها برای کار با سامانه‌های اطلاعات جغرافیایی از جمله مجموعه غنیی از ابزارهای پردازش مکانی می‌باشد.

الحاقات فراوانی برای نرم‌افزارهای مجموعه ArcGIS Desktop توسط شرکت‌های مختلف از جمله شرکت ESRI ارائه شده‌اند که قابلیت‌های جدیدی به این نرم‌افزارها اضافه می‌کنند. همچنین توسعه‌دهندگان می‌توانند با استفاده از ArcObjects (اجزای نرم‌افزاری ArcGIS Desktop) الحاقات سفارشی خود را تولید و استفاده کنند. توسعه‌دهندگان می‌توانند الحاقات و ابزارهای مورد نظر خود را با استفاده از ابزارهای توسعه استاندارد سیستم عامل ویندوز مانند Visual Basic، Net. و Visual C++ ایجاد کنند.




**شکل ٤ : نمونه‌ای از اجرای نرم‌افزار ArcMap**

### ۷-۱-۱- ArcMap

نرم‌افزار ArcMap یک نرم‌افزار جامع برای تولید و ایجاد نقشه می‌باشد. این نرم‌افزار، نرم‌افزار اصلی مجموعه ArcGIS می‌باشد و برای تمامی عملیات مربوط به نقشه‌ها و همچنین پرس و جوها و تحلیل‌های مرتبط با نقشه استفاده می‌شود. این نرم‌افزار ابزار اصلی برای انجام تمامی وظایف مرتبط با نقشه، شامل نقشه‌کشی[۹۶]، تجزیه تحلیل نقشه و ویرایش می‌باشد. نرم‌افزار ArcMap اطلاعات مکانی را به صورت مجموعه‌ای از لایه‌های نقشه و عناصر دیگر در نمای نقشه نمایش می‌دهد. عناصر معمول در نمای نقشه شامل لایه‌ها موجود در محدوده انتخاب شده، نوار مقیاس‌بندی[۹۷]، فلش شمال[۹۸]، متن‌های توضیحی و راهنمای علائم[۹۹] می‌شوند. شکل ۳ نمونه‌ای از یک اجرای نرم‌افزار ArcMap را نمایش می‌دهد.

### ۷-۱-۲- ArcCatalog

نرم‌افزار ArcCatalog به کاربران امکان مدیریت انواع اطلاعات مکانی - از قبیل نقشه‌ها، کره‌ها، فایل‌های داده، بانک‌های اطلاعات مکانی، جعبه ابزارهای پردازش مکانی[۱۰۰]، ابرداده‌ها[۱۰۱] و سرویس‌های اطلاعات مکانی[۱۰۲] - را به کاربران می‌دهد. این نرم‌افزار شامل ابزارهای برای انجام عملیاتی نظیر موارد زیر می‌باشد:

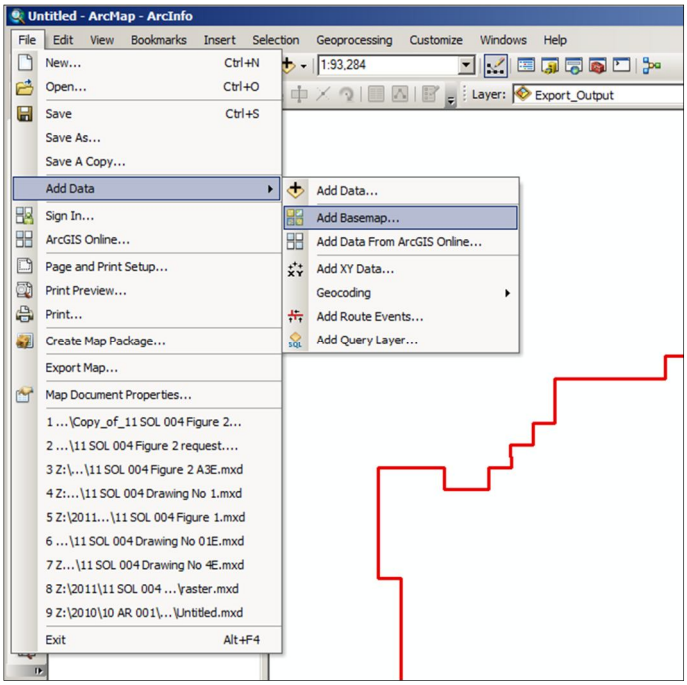
- مرور[۱۰۳] و یافتن اطلاعات مکانی
- ذخیره، مشاهده و مدیریت ابرداده‌ها
- تعریف، بیرون‌دهی[۱۰۴] و درون‌ریزی[۱۰۵] مدل‌های بانک‌های اطلاعات مکانی
- جستجو و کشف[۱۰۶] داده‌های مکانی در شبکه‌های محلی و وب

---

[۹۶] Cartography
[۹۷] Scale Bar
[۹۸] North Arrow
[۹۹] Symbol Legend
[۱۰۰] Geoprocessing Toolbox
[۱۰۱] Metadata
[۱۰۲] GIS Services
[۱۰۳] Browse
[۱۰۴] Export
[۱۰۵] Import

- مدیریت و اداره بانک‌های اطلاعات مکانی ArcSDE قرار گرفته بر روی SQL Server Express
- مدیریت و اداره بانک‌های اطلاعات مکانی فایلی و شخصی[۱۰۷]
- مدیریت و اداره سرویس‌های اطلاعات مکانی

کاربران با استفاده از ArcCatalog می‌توانند علاوه بر یافتن، سازمان دادن و استفاده از اطلاعات مکانی، با استفاده از ابرداده‌های استاندار به مستندسازی آنها بپردازند. مدیران بانک‌های اطلاعاتی با استفاده از این نرم‌افزار می‌توانند بانک‌های اطلاعات مکانی را تعریف و ایجاد کنند و مدیران شبکه نیز می‌توانند با استفاده از قابلیت‌های نرم‌افزار ArcCatalog به اداره اجزای سروری نرم‌افزار ArcGIS بپردازند. شکل ۵ نمونه‌ای از صفحه اصلی نرم‌افزار ArcCatalog را نمایش می‌دهد.




**شکل ۵ : نمونه‌ای از صفحه اصلی نرم‌افزار ArcCatalog**

### ۷-۱-۳- ModelBuilder

رابط کاربری ModelBuilder که جزئی از نرم‌افزار ArcMap می‌باشد یک چهارچوب مدل‌سازی گرافیکی برای طراحی و پیاده‌سازی مدل‌های پردازش مکانی ارائه می‌دهد که می‌توانند شامل ابزارها، نوشتارها[۱۰۸] و داده‌ها باشند. کاربران با کشیدن و انداختن ابزارها و مجموعه داده‌ها بر روی مدل و

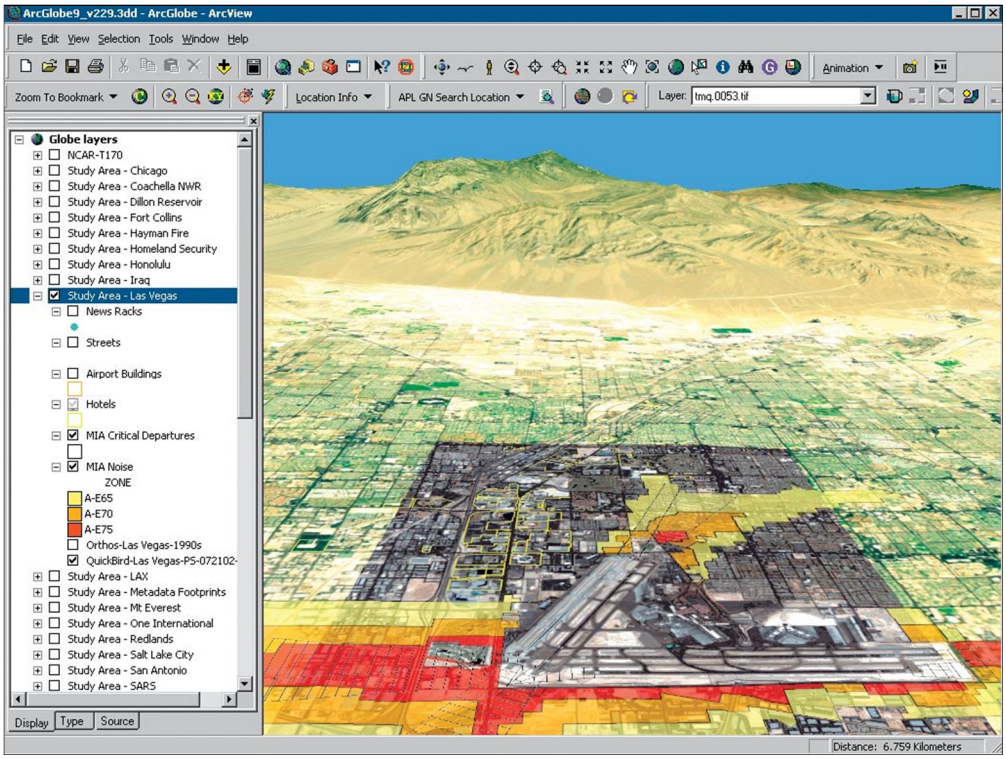
---

[۱۰۶] Discover

[۱۰۷] Personal Geodatabase

[۱۰۸] Script

برقرار کردن ارتباط بین آنها می‌توانند توالی کاری پیچیده‌ای از وظایف مرتبط با سامانه اطلاعات جغرافیایی ایجاد کنند. به عبارت دیگر ModelBuilder یک مکانیزم تعاملی[۱۰۹] را برای ایجاد رویه‌های پیچیده‌ای برای سامانه اطلاعات جغرافیایی و اجرای آنها ارائه می‌دهد. از این نرم‌افزار می‌توان برای اشتراک‌گذاری روش‌ها و رویه‌های موجود بین افراد درون و بیرون یک سازمان استفاده کرد.




**شکل ٦ : نمونه‌ای از اجرای نرم‌افزار ArcGlobe**

### ۷-۱-٤- ArcGlobe

نرم‌افزار ArcGlobe امکان نمایش و مشاهده داده‌های مکانی را به صورت پیوسته، با تفکیک‌پذیری‌های[۱۱۰] مختلف و به شکل تعاملی ارائه می‌دهد. همانند ArcMap، نرم‌افزار ArcGlobe نیز توانایی کار با لایه‌های مختلف اطلاعات مکانی و نمایش محتویات بانک‌های اطلاعات مکانی و تمامی قالب‌های نگهداری داده‌های مکانی را دارد. این نرم‌افزار دارای یک نمای سه بعدی برای نمایش داده‌های مکانی است. نرم‌افزار ArcGlobe تمامی لایه‌های اطلاعاتی را درون یک زمینه[۱۱۱] عمومی نمایش می‌دهد و به این ترتیب داده‌های موجود در تمامی مجموعه داده‌ها را به صورت

[۱۰۹] Interactive

[۱۱۰] Resolutaion

[۱۱۱] Context

یکپارچه[۱۱۲] در یک چهارچوب سراسری نمایش می‌دهد. همچنین نرم‌افزار ArcGlobe قابلیت پردازش داده‌ها با انواع تفکیک‌پذیری را داشته و می‌تواند مجموعه داده‌ها را به شکل مناسبی با انواع سطوح تفکیک‌پذیری نمایش دهد. شکل ٦ نمونه‌ای از اجرای نرم‌افزار ArcGlobe را نمایش می‌دهد.

### ۷-۱-۵- ArcScene

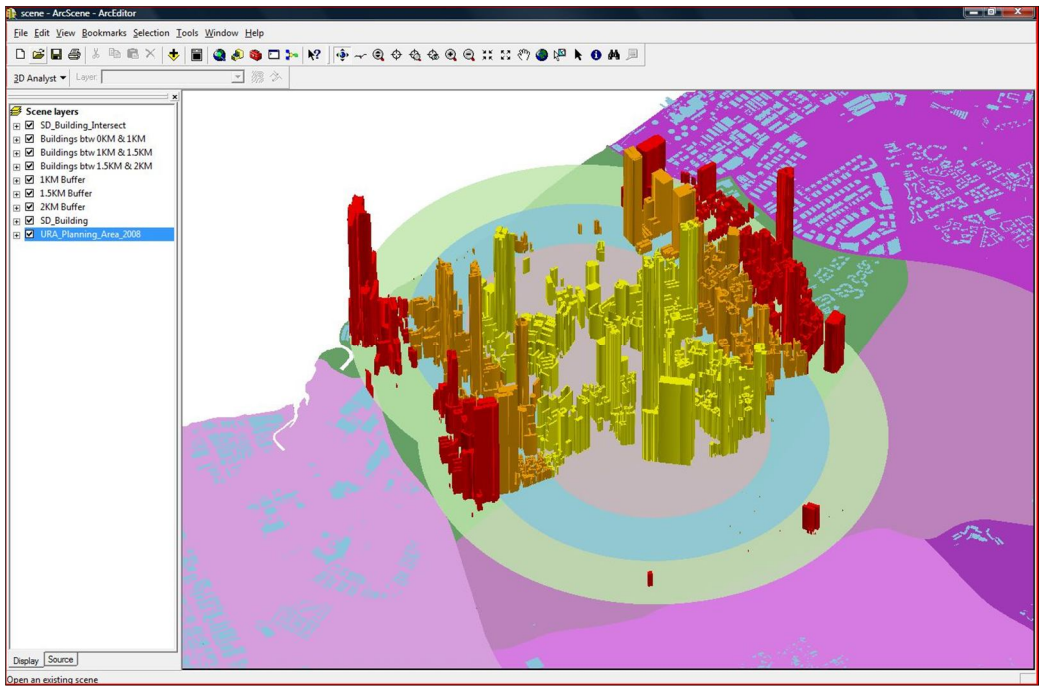
نرم‌افزار ArcScene دومین نرم‌افزار سه بعدی در مجموعه ArcGIS است. این نرم‌افزار توانایی امکان تولید و کار با منظره‌های سه بعدی را در اختیار کاربران قرار می‌دهد؛ تفاوت نماهای تولید شده توسط این نرم‌افزار و نرم‌افزار ArcGlobe در این است که نرم‌افزار ArcGlobe برای نمایش چندین مجموعه داده و با وسعت دید زیاد استفاده می‌شود در حالی که نرم‌افزار ArcScene برای نمایش بخش‌های کوچکی از پروژه‌ها به صورت متمرکز مورد استفاده قرار می‌گیرد. شکل ۷ نمونه‌ای از یک اجرای نرم‌افزار ArcScene را نشان می‌دهد. نرم‌افزار ArcScene با ارائه نمایش تنگ ماهی مانند امکان کار با داده‌های مکانی سه بعدی را به کاربران ارائه می‌دهد.

## ۷-۲- Server GIS

امروزه استفاده از سامانه‌های اطلاعات جغرافیایی به عنوان سرور به شکل روز افزونی در حال رشد است؛ علت این امر مزیت‌های تجاری و امکان ارائه اطلاعات و منابع مکانی با ارزش که توسط متخصصان GIS تولید شده‌اند با نفوذ پزیری بالا می‌باشد. به همین دلیل موسسات و سازمان‌ها برای ارائه اطلاعات و توابع مکانی خود در کل مجموعه خود، Desktop GISها موجود را به وسیله سرورهای اطلاعات جغرافیایی و سرویس‌ها وبی[۱۱۳] گسترش می‌دهند. به این ترتیب کاربران در کل سازمان از یک پیاده‌سازی سامانه اطلاعات جغرافیایی سود می‌برند. به اشتراک‌گذاری دانش مکانی به این ترتیب موجب استاندارد و یک‌دست شدن تکنیک‌های پردازش‌های مکانی و سناریوهای گردش‌کار شده، هزینه‌های نصب و راه‌اندازی نرم‌افزار را کاهش داده و راحتی حجم پیاده‌سازی می‌شود.

[۱۱۲] Integrated

[۱۱۳] Web Services




**شکل ۷ : نمونه‌ای از اجرای نرم‌افزار ArcScene**

متخصصان GIS نه تنها از سرورهای اطلاعات مکانی به عنوان یک سکو برای انتشار و ترویج کارهای خود به صورت نقشه‌ها، کره‌ها، مدل‌های پردازش مکانی و نرم‌افزارها به اشتراک گذاشته شده استفاده می‌کنند، بلکه می‌توانند از دیگر سرویس‌های مکانی منتشر شده توسط دیگر متخصصان GIS استفاده کنند.

سرورهای اطلاعات مکانی مزیت‌های زیر را به کاربران ارائه می‌دهد:

- صرفه اقتصادی بسیار بالا در راه‌اندازی یک سامانه اطلاعات جغرافیایی در کل سازمان و استفاده از آن
- به اشتراک‌گذاری منابع پردازشی در زمینه GIS و مدیریت مرکزی آن
- طیف وسیعی از نرم‌افزارها و ابزارها برای کلاینت‌ها[۱۱۴] که انعطاف‌پذیری لازم برای انجام هر گونه وظیفه و هدف را دارند و از سرویس‌های ارائه شده توسط سرورهای اطلاعات مکانی استفاده می‌کنند؛ به عنوان مثال دسترسی به سامانه اطلاعات مکانی از طریق مرورگرهای وب و دستگاه‌های تلفن همراه، نرم‌افزارهای ویرایش اطلاعات مکانی مختلف، مجموعه نرم‌افزارهای ArcGIS Desktop و همچنین نرم‌افزاری مصرفی نظیر سرویس‌های ارائه شده توسط Google و Microsoft.

[۱۱۴] Client

- سرویس‌های ارائه شده توسط سرورهای اطلاعات مکانی قابلیت یکپارچه شدن با دیگر سامانه‌های سازمانی نظیر نرم‌افزارهای مدیریت ارتباط با مشتری[۱۱۵] و نرم‌افزارهای مدیریت منابع سازمانی[۱۱۶] به وسیله رابط‌ها و رویه‌های استاندارد نرم‌افزاری را دارند. همچنین این سرورها زیر بنای لازم برای استفاده در معماری‌های سرویس‌گرا[۱۱۷] را دارند.
- قابلیت تولید نرم‌افزارهای شخصی‌سازی شده به وسیله محیط‌ها و استانداردهای توسعه نرم‌افزار شناخته شده نظیر .Net، Java، SOAP، REST، JavaScript و Adobe Flex
- مجموعه واحدی از نقشه‌ها و سرویس‌های اطلاعات مکانی به اشتراک گذاشته شده که استحکام و پیوستگی را در مدیریت اطلاعات و نمایش آنها تضمین می‌کند.
- پشتیبانی از استانداردهای همکاری[۱۱۸] هم در قلمرو سامانه‌های اطلاعات مکانی (مانند OGC[۱۱۹] و ISO) و هم در قلمروهای وسیع‌تری نظیر وب جهان‌گستر[۱۲۰].

جزء سرور مجموعه نرم افزار ArcGIS یکی از کامل‌ترین سرورهای اطلاعات مکانی می‌باشد که تمامی قابلیت‌ها و کاربردها اشاره شده را به کاربران ارائه می‌دهد. این مجموعه نرم‌افزار دارای سه محصول اصلی می‌باشد:

- **ArcIMS**[۱۲۱]

این نرم‌افزار یک سرور مقیاس‌پذیر[۱۲۲] برای ارائه نقشه‌ها از طریق وب است که برای نشر نقشه‌های پویا[۱۲۳]، داده‌ها و فهرست‌هایی از ابرداده‌های به وسیله پروتکل‌های باز اینترنت[۱۲۴] مورد

---

[۱۱۵] Customer Relationship Management (CRM)

[۱۱۶] Enterprise Resource Planning (ERP)

[۱۱۷] Service Oriented Architecture (SOA)

[۱۱۸] Interoperatablity Standards

[۱۱۹] Open GIS Consortium

[۱۲۰] World Wide Web (WWW)

[۱۲۱] Arc Internet Map Server

[۱۲۲] Scalable

[۱۲۳] Dynamic

[۱۲۴] Open Internet Protocols

استفاده قرار می‌گیرد. تمرکز اصلی ArcIMS برای ارائه داده‌ها و نقشه‌ها از طریق وب به تعداد زیادی کاربر می‌باشد. این نرم‌افزار برای ارائه نقشه‌های باز با کارایی[۱۲۵] بالا طراحی شده است.

- **ArcGIS Server**

این نرم‌افزار یک سامانه اطلاعات جغرافیایی مبتنی بر وب جامع است که طیفی از نرم‌افزارها و سرویس‌های حاضر و آماده را به کاربر نهایی[۱۲۶] برای عملیات نقشه‌کشی، تحلیل، جمع‌آوری اطلاعاتی، ویرایش و مدیریت بانک‌های اطلاعات مکانی ارائه می‌دهد. نرم‌افزار ArcGIS Server یک سکوی مقرون به صرفه و مبتنی بر استانداردها را در اختیار کاربران ArcGIS Desktop قرار می‌دهد تا آنها بتوانند به راحتی دانش خود در زمینه اطلاعات جغرافیایی را در اجتماع وسیع‌تری منتشر کنند. این نرم‌افزار از طیف گسترده‌ای از روش‌های اتصال را در اختیار کامپیوترهای رومیزی، گوشی‌های تلفن همراه و مرورگرهای وب قرار می‌دهد.

- **ArcGIS Image Server**

این نرم‌افزار یک سرور پردازش تصویر می‌باشد که با سرعت و به راحتی راه‌اندازی می‌شود. این سرور دسترسی سریع به مجموعه بزرگی از تصاویر را فراهم می‌کند و به این ترتیب زمان بین تصویربرداری و استفاده از تصاویر را به شدت کاهش می‌دهد.

مجموعه سرورهای اطلاعات جغرافیایی ArcGIS از استانداردهای موجود در حوزه فن‌آوری اطلاعات، وب و صنعت GIS پشتیبانی می‌کند. همچنین آنها با دیگر نرم‌افزارها با استفاده از فن‌آوری‌های شناخته شده نظیر وب سرویس‌ها و سامانه‌های مدیریت بانک اطلاعاتی همکاری می‌کند. همچنین با سکوی‌های توسعه نرم‌افزاری نظیر J۲EE[۱۲۷] و Microsoft.Net می‌توان با این نرم‌افزارها ارتباط برقرار کرد و بر اساس آنها نرم‌افزارهای جدید و سفارشی شده‌ای طراحی کرد.

## ۳-۷ Developer GIS

توسعه‌دهندگان نرم‌افزار موجب گسترش سامانه‌های اطلاعات جغرافیایی بین طیف وسیعی از کاربران شده‌اند. توسعه‌دهندگان سامانه‌های اطلاعات جغرافیایی با ساخت و توسعه نرم‌افزارهای سفارشی و متمرکز شده بر موضوعی خاص که به بسیاری از کاربران نهایی امکان استفاده از تمامی

---

[۱۲۵] Performance

[۱۲۶] End User

[۱۲۷] Java ۲ Enterprise Edition

قدرت موجود در سامانه‌های اطلاعات مکانی را توسط رابط‌های کاربری ساده می‌دهند موجب گسترش وسیع این سامانه‌ها شده‌اند. توسعه‌دهندگان همچنین به توسعه نرم‌افزارهای وبی با قابلیت‌های اطلاعات مکانی و همچنین سرویس‌های وبی خاص منظوره[۱۲۸] می‌پردازند.

یکی از اهداف اصلی ArcGIS ارائه چهارچوب‌های توسعه جامع به توسعه‌دهندگان برای تولید نرم‌افزارهای سفارشی و گسترش سامانه‌های اطلاعات جغرافیایی می‌باشد. به این منظور مجموعه نرم‌افزارهای ArcGIS اجزای نرم‌افزاری، ابزارها و روش‌هایی را برای توسعه‌دهندگان در برای چهارچوب‌های زیر نظر گرفته است:

- سفارشی‌سازی[۱۲۹] و بسط[۱۳۰] ArcGIS Desktop
- ایجاد نرم‌افزارهای سفارشی به استفاده از اجزای GISی جاسازی شده به وسیله ArcGIS Engine
- ایجاد نرم‌افزارهای اینترنتی غنی[۱۳۱] به وسیله ArcGIS Server؛ به عنوان مثال ایجاد یک نرم‌افزار سفارشی به وسیله JavaScript
- استفاده و سفارشی‌سازی نرم‌افزارها و سرویس‌های وبی مرتبط با نقشه به وسیله ArcIMS و ArcGIS Server
- بسط انواع داده، دسترسی به محتویات بانک‌های اطلاعات مکانی، بسط ArcSDE و دسترسی به سامانه‌های مدیریت بانک‌های اطلاعاتی رابطه‌ای به وسیله SQL
- توسعه نرم‌افزارهای تلفن‌های همراه برای استفاده از ArcGIS

قلب بسط و توسعه نرم‌افزارها برای ArcGIS، کتابخانه اجزای نرم‌افزاری[۱۳۲] ArcObjects می‌باشد. مجموعه ArcObjects مجموعه یکپارچه‌ای از اجزای نرم‌افزاری بین‌سکویی[۱۳۳] برای GIS است که هم برای کلاینت و هم برای سرور تولید شده‌اند.

---

[۱۲۸] Specialized Web Services

[۱۲۹] Customizing

[۱۳۰] Extending

[۱۳۱] Rich Internet Application

[۱۳۲] Software Component Library

[۱۳۳] Cross-Platform

کتابخانه ArcObjects که بین تمامی اجزای ArcGIS مشترک می‌باشد، یک تجربه توسعه واحد را برای ArcGIS Desktop، ArcGIS Engine و ArcGIS Server ارائه می‌دهد. این کتابخانه بر اساس یک معماری واحدی، مقیاس‌پذیر و بین‌سکویی ایجاد شده و رابط‌های برنامه‌سازی[۱۳۴] استانداردی را برای C++، Net. و Java ارائه می‌دهد. شکل ۸ رابطه ArcObjects و دیگر اجزای ArcGIS را نمایش می‌دهد.

## Mobile GIS – ۴–۷

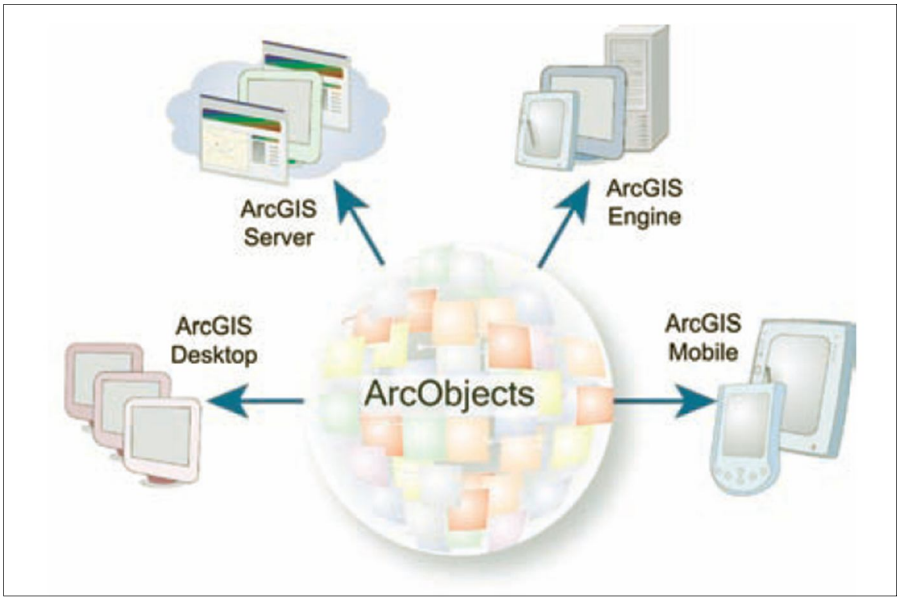
پردازش سیار[۱۳۵] با اضافه کردن قابلیت استفاده میدانی از سامانه‌های اطلاعات جغرافیایی و تعامل با دنیا تغییرات بنیادینی را در دنیای این سامانه‌ها به وجود آورده است. سامانه‌های مبتنی بر Mobile GIS با یکپارچه‌سازی چندین فن‌آوری به وجود می‌آیند:

- سامانه‌های اطلاعات جغرافیایی
- سخت‌افزارهای سیار به صورت دستگاه‌های سبک‌وزن و کامپیوترهای مقاوم
- سامانه موقعیت‌یاب جهانی (GPS)
- ارتباط بی‌سیم برای دسترسی به سامانه‌های اطلاعات جغرافیایی از طریق اینترنت
- همگام‌سازی[۱۳۶] اطلاعات با استفاده از سرورهای اطلاعات جغرافیایی

[۱۳٤] Application Programming Interface (API)

[۱۳٥] Mobile Computing

[۱۳٦] Synchronization




**شکل ۸ : رابطه ArcObjects و دیگر اجزای ArcGIS**

در شکل سنتی، اطلاعات مکانی به وسیله نقشه‌های کاغذی معمولا به صورت نقشه کتاب به زمین و مکانی مورد نظر برده می‌شد. اطلاعات به وسیله نقشه کتاب به صورت طرح‌های ساده[۱۳۷] بر روی نقشه جمع‌آوری شده و در زمان بازگشت به دفتر کار وارد سامانه اطلاعات جغرافیایی می‌شدند. بازرسی‌ها نیز در قالب فرم‌های کاغذی استانداردی که به زمین برده می‌شد جمع‌آوری و سپس در دفتر کار به بانک اطلاعاتی منتقل می‌شدند. جمع‌آوری داده به روش کاغذی عملیاتی ناکارآمد، تکراری و مستعد خطا می‌باشد.

امروزه سازمان‌ها روش‌های مبتنی بر کاغذ خود را با نرم‌افزارهای سیار جایگزین می‌کنند. نرم‌افزارهای سیار مبتنی اطلاعات جغرافیایی، سامانه‌های اطلاعات جغرافیایی را وارد تعداد زیادی از گردش‌کارهای مرتبط با نقشه‌برداری و ویرایش آن کرده‌اند. در این راستا ArcGIS Server و ArcGIS Engine ابزارهای قدرت‌مند، آسان و کارآمدی را برای تولید سامانه‌های اطلاعات جغرافیایی سیار ارائه می‌دهند.

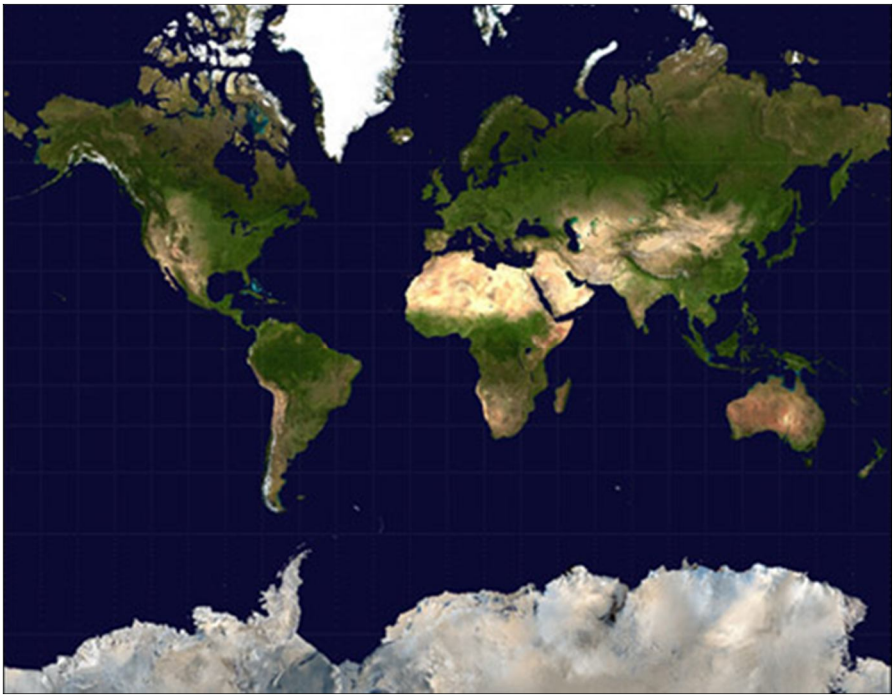
[۱۳۷] Sketch

# فصل ۸– معرفی Google Maps

نقشه‌های گوگل یا Google Maps یک سرویس وبی از شرکت گوگل است. این سرویس نقشه‌های دقیق و کاملی از زمین ارائه می‌گردد. سرویس نقشه‌های گوگل از فن‌آوری‌هایی چون Tele Atlas استفاده می‌کند.

این سرویس یکی از ده‌ها خدمات رایگان گوگل در وب است و زیر بنای خدمات بزرگ دیگری نظیر وب سایت Google Maps، Google Transit، Google Ride Finder و دیگر سایت‌های بنا شده با کمک Google Maps API است.

سرویس نقشه‌های گوگل از استاندارد Mercator Projection برای محاسبه مختصات نقشه‌های خود استفاده می‌کند. این استاندارد در سال ۱۵۶۹ میلادی توسط جغرافی دان بلژیکی جراردوس مرکاتور[۱۳۸] ارائه شد. در این استاندارد با محاسباتی که انجام می‌شود، می‌توان سطح یک کره را به طور دقیق و با حفظ مختصات به روی یک صفحه کاغذ آورد؛ شکل ۹ نمونه‌ای از نقشه ایجاد شده توسط این استاندارد را نمایش می‌دهد. به همین خاطر سرویش نقشه‌های گوگل فاقد نقشه‌ها و تصاویر دو قطب کره زمین است (برای دیدن تصاویر از قطب‌ها باید از نرم افزار Google Earth استفاده نمود).



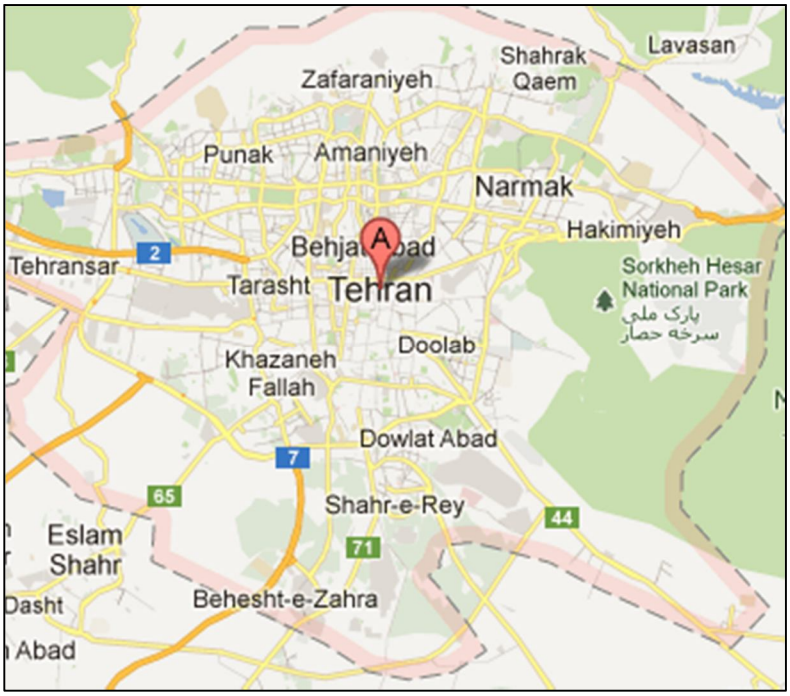
[۱۳۸] Gerardus Mercator

**شکل ۹ : نمونه نقشه تولید شده توسط استاندار Mercator Projection**

## ۸-۱- تاریخچه

نقشه‌های گوگل ابتدا به صورت یک نرم‌افزار توسط لارس راسموسن[۱۳۹] و ینس راسموسن[۱۴۰] در شرکت Where ۲ Technologies نوشته شد. در اکتبر سال ۲۰۰٤ این شرکت توسط گوگل خریداری شد و در آخر نرم‌افزار به صورت یک Application تحت وب با عنوان GoogleMaps در سال ۲۰۰۵ عرضه شد. سیر تحولات این سرویس تا امروز ادامه دارد. شکل ۱۰ نمونه‌ای از صفحه وب سایت GoogleMaps را نمایش می‌دهد.

نقشه‌های گوگل امکان جستجوی نقشه، قرار دادن نظرات، عکس‌ها و فیلم‌های کاربران برای یک مکان و همچنین امکان تغییر نوع نمایش از نقشه به تصاویر ماهواره یا هم نقشه و هم تصاویر ماهواره و یا نقشه توپوگرافی[۱۴۱] (پستی و بلندی) منطقه را در اختیار کاربر قرار می‌دهد.




**شکل ۱۰ : نمونه‌ای از صفحه وب سایت GoogleMaps**

[۱۳۹] Lars Rasmussen

[۱٤۰] Jens Rasmussen

[۱٤۱] Topography

## ۸-۲- نماهای مختلف در نقشه‌های گوگل

نقشه‌های گوگل دیدن نقشه را در چندین حالت امکان‌پذیر کرده‌است:

- **نمای نقشه**[۱۴۲]

در این نما نقشه که به سبک نقشه‌های گرافیکی و معمول، از نمای بالا امکان رویت را فراهم می‌کند.

- **نمای ماهواره‌ای**[۱۴۳]

این نما تصاویر ماهواره‌ای واقعی از تمام نقاط سطح زمین را به کاربران نمایش می‌دهد.

- **نمای زمین**[۱۴۴]

با نصب الحاق گوگل ارث[۱۴۵] امکان رویت تصاویر ماهواره‌ای به صورت سه‌بعدی مانند نرم‌افزار گوگل ارث برای مرورگر[۱۴۶] فراهم می‌شود.

- **نمای خیابان**[۱۴۷]

در این نما که تنها در مکان‌هایی خاص قابل دسترسی است، تصاویری سه بعدی از درون خیابان‌های مورد نظر قابل رویت است. این تصاویر قبلا به کمک دوربین‌هایی سوار بر خودروهایی مخصوص از سطح خیابان‌ها گرفته شده‌است. شکل ۱۱ نمونه‌ای از این نما را نمایش می‌دهد.

---

[۱۴۲] Map View

[۱۴۳] Satellite View

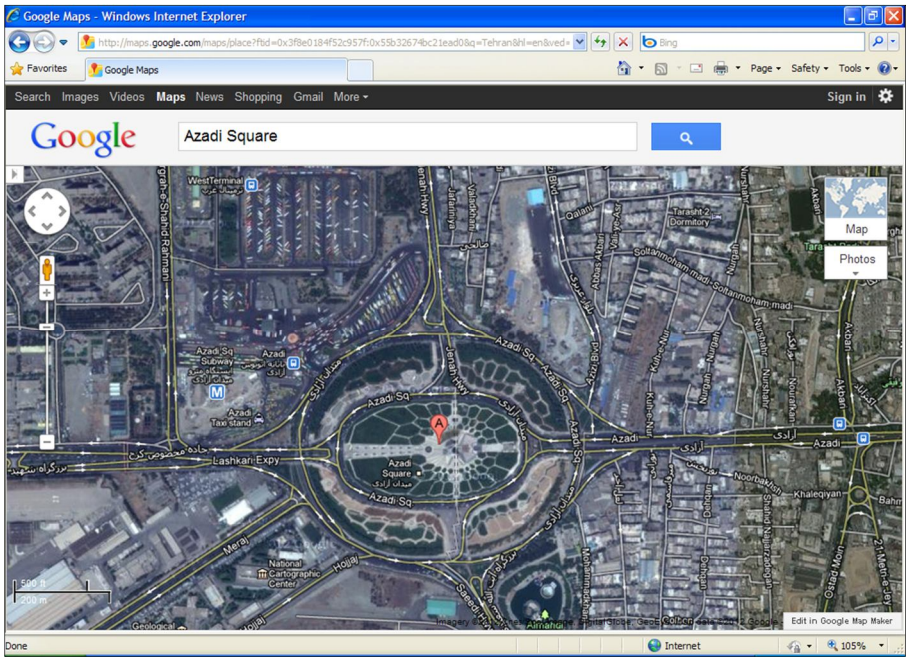
[۱۴۴] Earth View

[۱۴۵] Google Earth

[۱۴۶] Browser

[۱۴۷] Street View


**شکل ۱۱ : نمونه‌ای از نمای خیابان**




**شکل ۱۲ : نقشه‌ای ترکیبی از نمای نقشه و نمای ماهواره**

- **نمای سه بعدی**[۱۴۸]

این نما که به نمای هلیکوپتری معروف است امکان رویت مسیرها را به صورت سه‌بعدی از آسمان را فراهم می‌آورد.

- **نمای درون**[۱۴۹]

[۱۴۸] 3D View

در این نما نقشه ساختمان‌های چند طبقه به تفکیک طبقات آن نمایش داده می‌شود.

- **نمای هوایی**[۱۵۰]

این نما تصاویری از ماهواره‌ها را نشان می‌دهد اما با این تفاوت که تصاویر عمود گرفته نشده‌اند، یعنی از جهات دیگری به جز زاویه عمود گرفته شده‌اند.

شکل ۱۲ نمونه‌ای از نقشه ترکیبی از نمای نقشه و نمای ماهواره را نمایش می‌دهد و شکل ۱۳ نمونه‌ای از نمای هوایی را در مقایسه با نمای ماهواره نمایش می‌دهد.


**شکل ۱۳ : مقایسه نمای هوایی و نمای ماهواره**

## ۸-۳-تهیه نقشه‌های نمای نقشه

نقشه‌های به نمایش در آمده در سرویس نقشه‌های گوگل در ابتدا فقط به کشور ایالات متحده و چند کشور اروپایی محدود می‌شد. نقشه‌ها توسط خود گوگل درست می‌شدند و امکان تغییرات در آن توسط کس دیگری نبود. نقشه‌ها رویه طولانی و پر هزینه‌ای را طی می‌کردند تا بالاخره اجازه نمایش عمومی پیدا کنند. با گذشت زمان، پروژه‌ای به نام سازنده نقشه گوگل[۱۵۱] معرفی شد که وظیفه آن تسهیل در ساخت و به روز رسانی نقشه‌ها بود. این نرم‌افزار در ژوئن ۲۰۰۸ از آزمایشگاه‌های گوگل[۱۵۲] بیرون داده شد و همزمان ٦۰ کشور را برای ساخت نقشه پشتیبانی می‌کرد که ایران نیز جز آنها بود. در نرم‌افزار سازنده نقشه گوگل کاربران از هر جای زمین می‌توانند نقشه‌ها را توسعه دهند، ویرایش کنند و یا حذف کنند. اطلاعاتی که کاربران عادی وارد می‌کنند ممکن است

---

[۱۴۹] Indoor View

[۱۵۰] Aerial View

[۱۵۱] Google Map Maker

[۱۵۲] Google Labs

قابل اطمینان نباشد، بنابراین گوگل قوانینی برای اطمینان حاصل کردن از اطلاعات وارد شده وضع کرد. کاربرانی که بیشتر از همه اطلاعات تایید شده دارند رتبه بالاتری دارند. به این ترتیب کاربرانی که رتبه پایین‌تری دارند برای انتشار نقشه‌های کشیده شده باید منتظر تایید کاربران رده بالاتر باشند. در حال حاضر سرویس نقشه‌های گوگل به واسطه نقشه‌های کشیده شده توسط کاربران در بیش از ۱۶۰ کشور جهان، نقشه‌های قابل اطمینانی دارد. نقشه‌ها و عکس‌های این سرویس در قسمت ایران نیز دارای کیفیت مطلوبی هستند.

## ۸-۴- تهیه تصاویر ماهواره‌ای

تا قبل از سال ۲۰۰۸ گوگل با شرکت‌هایی که کار آنها صرفا تهیه و فروش عکس‌های ماهواره‌ای است قرارداد داشت و از تصاویر آنها در سرویس نقشه‌های گوگل استفاده می‌کرد و البته هنوز هم اکثر تصاویر این سرویس توسط شرکت‌های مذکور تهیه می‌شوند. برخی از این شرکت‌ها عبارت‌اند از:

- **GeoEye**
- **Digital Globe**
- **ناسا**

در سال ۲۰۰۸ گوگل اولین ماهواره خود را به فضا فرستاد. کار این ماهواره صرفا گرفتن عکس‌های با کیفیت فوق العاده بالا و قرار دادن روی گوگل ارث و سرویس نقشه‌های گوگل بود. به علت بالا بودن بیش از اندازه این عکس‌ها (پیکسل از عکس ۲ اینچ مربع از صفحه است که در مقایسه با با کیفیت‌ترین عکس‌های GeoEye که ۱۸ اینچ است، رقم قابل مقایسه‌ای است.) دادگاه فدرال آمریکا گوگل را مجبور به کم‌کردن کیفیت تصاویر و سپس قرار دادن آن‌ها روی اینترنت کرد. این ماهواره که ساخت شرکت GeoEye است هم اکنون در حال عکس برداری با کیفیت بالا از زمین است.

## ۸–۵– استفاده در ناوبری

بسیاری از سرویس‌های ناوبری[۱۵۳] امروزه از نقشه‌های گوگل استفاده می‌کنند و با آن هماهنگی دارند. همچنین سرویس نقشه‌های گوگل با ارائه الگوی ترافیک لحظه‌ای به مسیر یابی کمک می‌کند.

## ۸–۶– قابلیت‌های توسعه نرم‌افزار

علاوه بر سایت‌های مرتبط با نقشه‌های گوگل، شرکت گوگل رابط‌های برنامه‌سازی[۱۵۴] ویژه‌ای را نیز ارائه داده است که با استفاده از آنها می‌توان از قابلیت‌های نقشه‌های گوگل در وب سایت‌ها و نرم‌افزارها بهره برد. با استفاده از این رابط‌های برنامه‌سازی می‌توان نقشه‌ها را از این سرویس گوگل دریافت کرده و در نرم‌افزارهای خود نمایش داد؛ در این روش با فرستادن مختصات جغرافیایی مکان مورد نظر و همچنین مقدار بزرگ‌نمایی تصاویر مربوط به آن منطقه از سرویس گوگل دریافت می‌شود.

[۱۵۳] Navigation

[۱۵۴] Application Programming Interfaces (API)

# فصل ۹–  معرفی نرم‌افزار NRDF

نرم‌افزار NRDF[۱۵۵] نرم‌افزاری برای جمع‌آوری و ذخیره اطلاعات راداری و پردازش و نمایش آنها می‌باشد. این نرم‌افزار دارای قابلیت‌ها و عملکردهای مختلفی است، که اصلی‌ترین عملکرد آن تنظیم سخت‌افزار جهت‌یاب راداری، دریافت اطلاعات از آن و نمایش و تحلیل آن است. شکل ١٤ نمای کلی نرم‌افزار را نمایش می‌دهد.

همانطور که در شکل مشاهده می‌شود، فضای کلی نرم‌افزار زبانه‌های Map، Google Maps، Analyse، Search و Offline تشکیل شده است. این زبانه‌ها را می‌توان در ٤ دسته طبقه‌بندی کرد:

١–زبانه‌های مربوط به نقشه و نمایش اجسام یافت شده بر روی نقشه شامل: Map و Google Maps

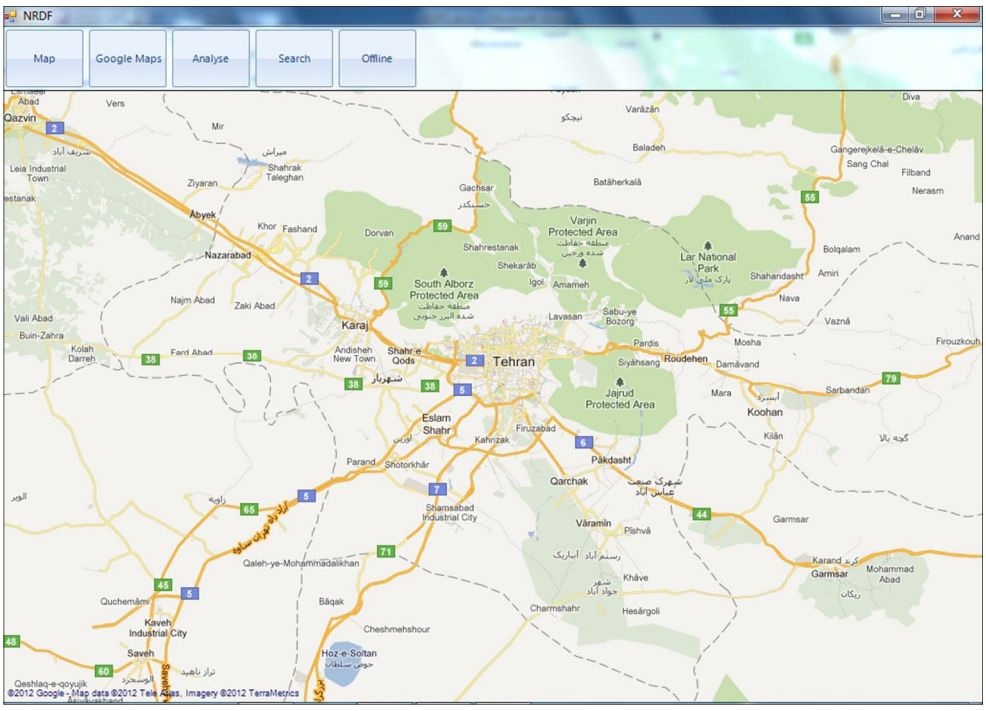
٢–زبانه تحلیل پالس با نام Analyse

٣–زبانه جستجو با نام Search

٤–زبانه Offline

در ادامه هر کدام از این دسته‌ها معرفی شده و نحوه عملکرد نرم‌افزار در هرکدام از آنها شرح داده می‌شوند.

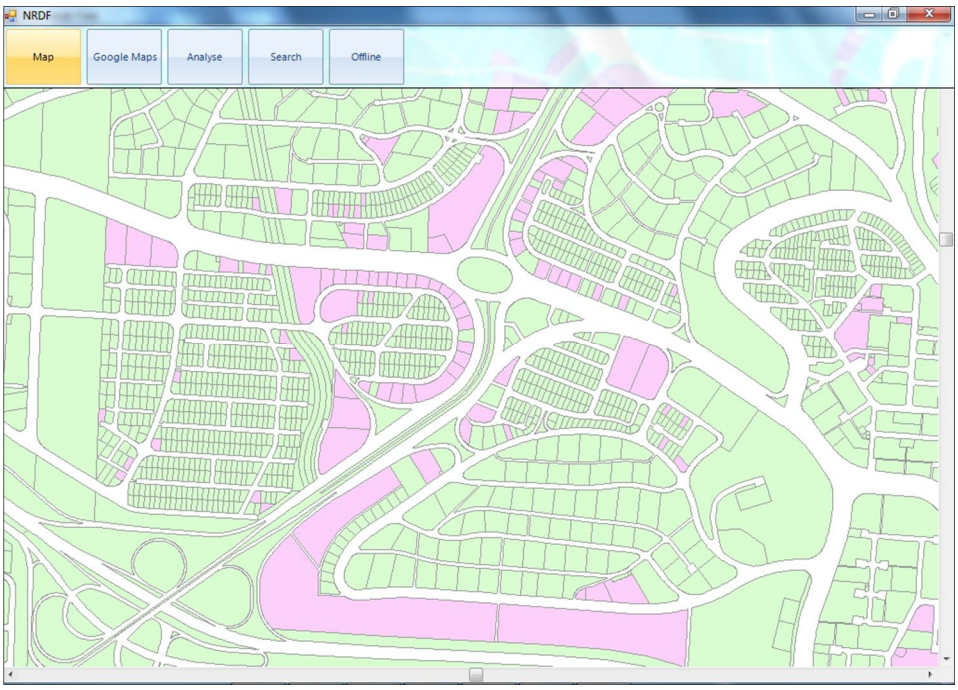
[۱۵۵] Nasr Radar Direction Finder




**شکل ۱٤ : نمای کلی نرم‌افزار NRDF**

## ۹-۱- زبانه‌های مربوط به نقشه

در این عملکرد، نرم‌افزار با کنترل حرکت آنتن جهت‌یاب راداری و تنظیم پارامترهای کارت پردازش‌گر، سخت‌افزار جهت‌یاب راداری را آماده جستجوی رادار کرده و دادن فرمان لازم این سیستم را راه‌اندازی می‌کند. سخت‌افزار جهت‌یاب راداری با دریافت دستور شروع و با توجه به پارامترهای تعیین شده شروع به کار می‌کند. آنتن‌های این دستگاه سیگنال‌های ارسال شده توسط رادارهای موجود در محدوده عملکرد دستگاه را جمع آوری می‌کند و آنها را برای پردازش به کارت پردازش‌گر ارسال می‌کند.

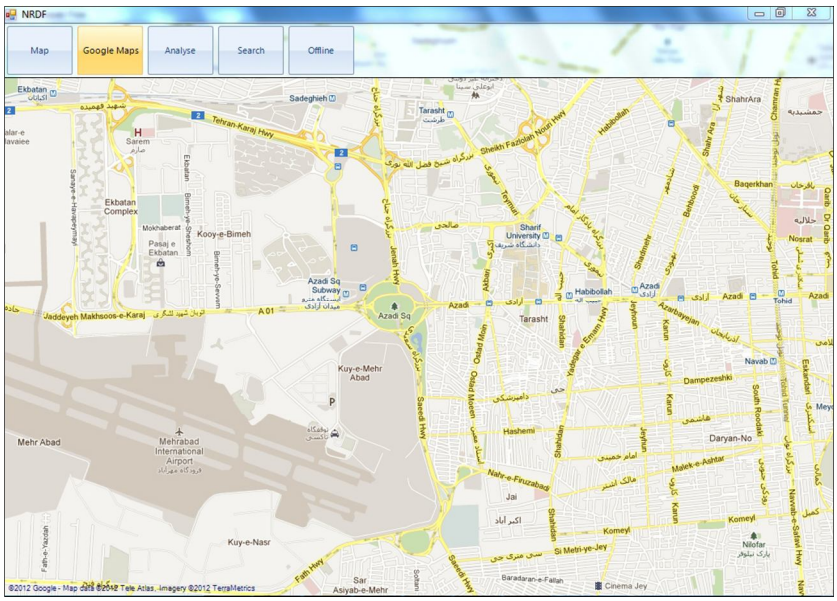



**شکل ۱۵ : نمایی از صفحه نقشه تولید شده توسط ArcGIS**

کارت پردازش‌گر پس از دریافت سیگنال‌ها از آنتن‌ها، پالس‌های موجود در آنها را تشخیص داده و اطلاعات هر یک از پالس‌ها را به یک کلمه توصیف کننده پالس[۱۵۶] – که به شکل خلاصه PDW نامیده می‌شود – تبدیل می‌کند. در نهایت نرم‌افزار PDWهای تولید شده در کارت پردازش‌گر را دریافت می‌کند و آنها در نمودارهای مربوطه نمایش می‌دهد. نمودارهای واقع در زبانه‌های مربوط به این بخش، از یک نقشه و یک نمودار قطبی تشکیل شده است.

نقشه که توسط یکی از دو ابزار Google Maps یا ArcGIS تولید می‌شود محدوده عملیات جهت‌یاب راداری را نمایش می‌دهد؛ این نقشه بسته با اینکه از کدامیک از دو ابزار نام برده استفاده شود شامل داده‌ها و لایه‌های اطلاعاتی مختلفی خواهد بود. به عنوان مثال نقشه می‌تواند شامل خیابان‌های یک شهر، موقعیت ساختمان‌های دولتی، بلوک‌های شهری و تاسیسات نظامی باشد. شکل ۱۵ و شکل ۱۶ نمای نرم‌افزار را در حالت نمایش نقشه توسط ArcGIS (زبانه Map) و Google Maps نمایش می‌دهند.

[۱۵۶] Pulse Description Word




**شکل ۱۶ : نمایی از صفحه نقشه تولید شده توسط Google Maps**

نمودار قطبی که با توجه به انتخاب کاربر بر روی نقشه نمایش داده می‌شود، به دو منظور مورد استفاده قرار می‌گیرد:

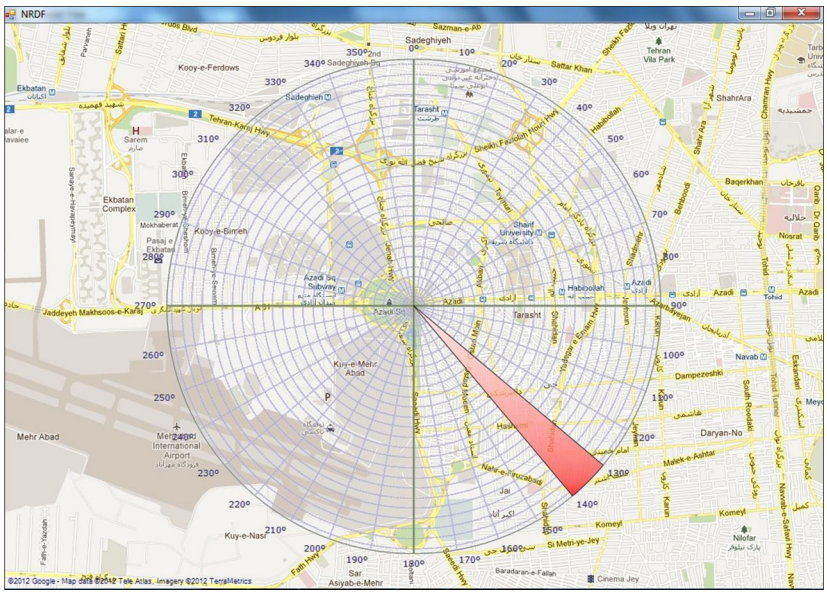
۱-نمایش موقعیت آنتن متحرک جهت‌یاب راداری

همانطور که در شکل ۱۷ نشان داده شده است، موقعیت آنتن متحرک که در واقع زاویه این آنتن می‌باشد به صورت یک مخروط با مقطع منحنی شکل نمایش داده شده است. زاویه این آنتن در هر ۲۰ میلی ثانیه از سخت‌افزار جهت‌یاب راداری و از طریق درگاه سریال[۱۵۷] سیستم دریافت می‌شود.

۲-نمایش موقعیت رادارهای تشخیص داده شده

همانطور که اشاره شد PDWهای دریافت شده از کارت پردازش‌گر شامل اطلاعات رادارهای تشخیص داده شده می‌باشد؛ این رادارها توسط نقاطی بر روی نمودار قطبی نمایش داده می‌شوند.

[۱۵۷] Serial Port




**شکل ۱۷ : نمودار قطبی نمایش دهنده جهت آنتن رادار**

در صورتی که کاربر گزینه آنالیز را در این بخش انتخاب کند، PDWهای دریافت شده از کارت پردازش‌گر برای تحلیل به واحد پردازش و خوشه‌بندی[۱۵۸] ارسال می‌کند. در واحد پردازش و خوشه‌بندی، نرم‌افزار پالس‌هایی که از اجسام یکسانی ارسال شده‌اند را تشخیص می‌دهد و پالس‌هایی که اطلاعات مناسبی ندارند را حذف می‌کند؛ در پایان این این مرحله اطلاعات خوشه‌بندی شده در جدول نتایج که در شکل ۱۸ نشان داده شده است نمایش داده می‌شود.

ستون‌های جدول نتایج عبارتند از:

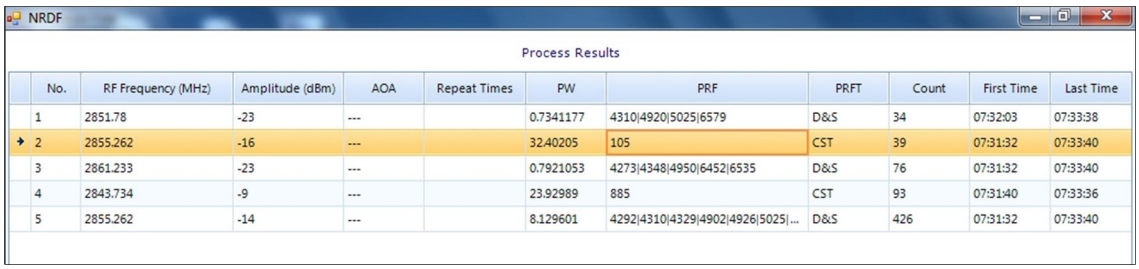
- **NO :** شماره ردیف سطر
- **RF Frequency :** فرکانس RF سیگنال رادار مربوط به سطر در واحد مگاهرتز
- **Amplitude :** دامنه سیگنال دریافتی در واحد dBm
- **AOA**[۱۵۹] **:** زاویه رادار پیدا شده را نسبت به جهت‌یاب راداری در واحد درجه نمایش می‌دهد.
- **Repeat Times :** تعداد دفعاتی که رادار شناسایی شده است.
- **PW**[۱۶۰] **:** پهنای پالس در واحد میکروثانیه

[۱۵۸] Clustering

[۱۵۹] Angle Of Arrival

[۱۶۰] Pulse Width

- **PRF**[۱۶۱] **:** مقدار PRFهای محاسبه شده برای رادار
- **PRFT**[۱۶۲] **:** نوع آرایش PRF
- **Count :** تعداد پالس‌هایی از رادار است که در زمان اسکن توسط سیستم جهت‌یاب راداری دریافت شده است.
- **First Time :** زمان دریافت اولین پالس از رادار
- **Last Time :** زمان دریافت آخرین پالس از رادار


NRDF

Process Results

| No. | RF Frequency (MHz) | Amplitude (dBm) | AOA | Repeat Times | PW | PRF | PRFT | Count | First Time | Last Time |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2851.78 | -23 | --- | | 0.7341177 | 4310\|4920\|5025\|6579 | D&S | 34 | 07:32:03 | 07:33:38 |
| 2 | 2855.262 | -16 | --- | | 32.40205 | 105 | CST | 39 | 07:31:32 | 07:33:40 |
| 3 | 2861.233 | -23 | --- | | 0.7921053 | 4273\|4348\|4950\|6452\|6535 | D&S | 76 | 07:31:32 | 07:33:40 |
| 4 | 2843.734 | -9 | --- | | 23.92989 | 885 | CST | 93 | 07:31:40 | 07:33:36 |
| 5 | 2855.262 | -14 | --- | | 8.129601 | 4292\|4310\|4329\|4902\|4926\|5025\|... | D&S | 426 | 07:31:32 | 07:33:40 |

**شکل ۱۸ : جدول نتایج**

### ۹-۱-۱- پارامترها

شکل ۱۹ صفحه مربوط به پارامترهای زبانه‌های مربوط به نقشه را نمایش می‌دهد. این پارامترها برای هر دو زبانه Map و Google Maps مشترک می‌باشد و عبارتند از:

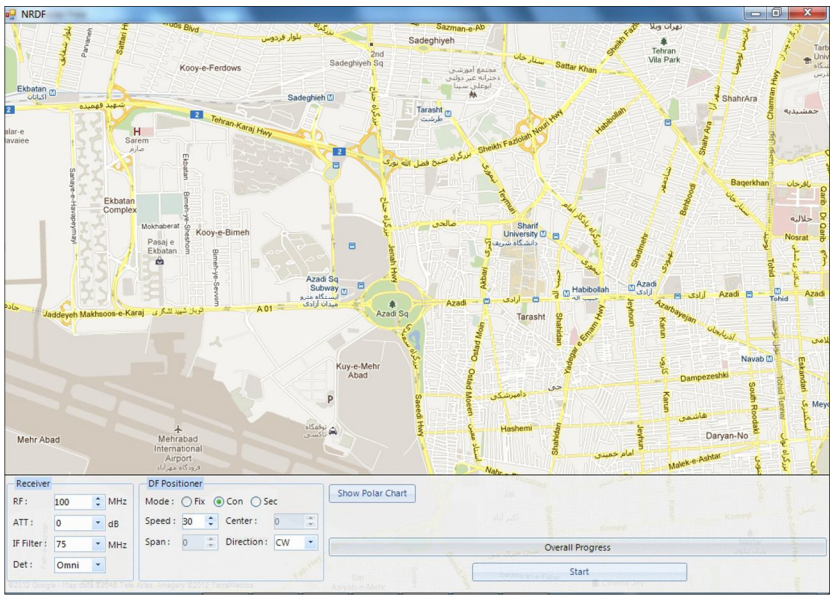
- **Receiver :** این بخش حاوی پارامترهای تنظیم نحوه دریافت سیگنال می‌باشد. تنظیمات دقیق و مناسب پارامترهای این بخش موجب بالا رفتن دقت سیگنال‌گیری و در نتیجه پردازش دقیق‌تر اطلاعات خواهد شد. همچنین با استفاده از پارامترهای این بخش می‌توان سیگنال RF ورودی را تا ۳۱ dBm تضعیف کرد. پارامترهای این بخش عبارتند از:
  - **RF :** مقدار فرکانس سیگنال RF که برای دریافت پالس مورد جستجو قرار می‌گیرد در این بخش با واحد مگاهرتز تعیین می‌شود.

[۱۶۱] Pulse Repetition Frequency

[۱۶۲] PRF Type

- **ATT :** در صورتی که گزینه فعال کننده این پارامتر انتخاب شود می‌توان با وارد کردن مقدار تضعیف تا ۳۱ dBm مقدار تضعیف اعمال شده بر روی سیگنال RF را تعیین کرد.
- **IF Filter :** پهنای باند فیلتر مورد استفاده برای سیگنال RF در این بخش تعیین می‌شود. فرکانس مرکزی این فیلتر ۱۶۰ مگاهرتز بوده و می‌توان مقادیر ۱، ۵، ۱۰، ۲۰، ۵۰ و ۷۵ مگاهرتز را به عنوان پهنای باند برای آن انتخاب کرد.
- **Det :** آنتن معیار برای تشخیص رادار را مشخص می‌کند.

- **DF Positioner :** در این قسمت تنظیمات مربوط به آنتن متحرک اعمال می‌شود. آنتن متحرک از سه حالت حرکتی که توسط پارامتر Mode تعیین می‌شود برای حرکت پشتیبانی می‌کند که عبارتند از:
  - **Continuous :** آنتن متحرک به میزان سرعت تعریفی در پارامتر Speed مسیر۳۶۰ درجه را به صورت چرخشی طی کرده و هدف را در هر جهتی جاروب کند؛ جهت چرخش آنتن در حالت Continuous توسط پارامتر Direction تعیین می‌گردد. حداکثر سرعت چرخش در این حالت بستگی به نوع آنتن متحرک دارد.
  - **Sectorial :** آنتن یک بازه حرکتی که توسط پارامتر Span تعریف شده است را با حرکت رفت و برگشت پیموده و هدف را در یک بازه بسته جاروب کند.
  - **Fixed :** آنتن در زاویه تعیین شده توسط پارامتر Center قرار گرفته و ثابت می‌ماند؛ در این حالت گین آنتن جهتی بیشترین مقدار را دارد.
- کلید Start : این کلید برای راه‌اندازی جهت‌یاب راداری و جستجو برای رادار استفاده می‌شود. پس از راه‌اندازی سیستم، این کلید به Stop تبدیل شده که با زدن آن عملیات متوقف می‌شود.

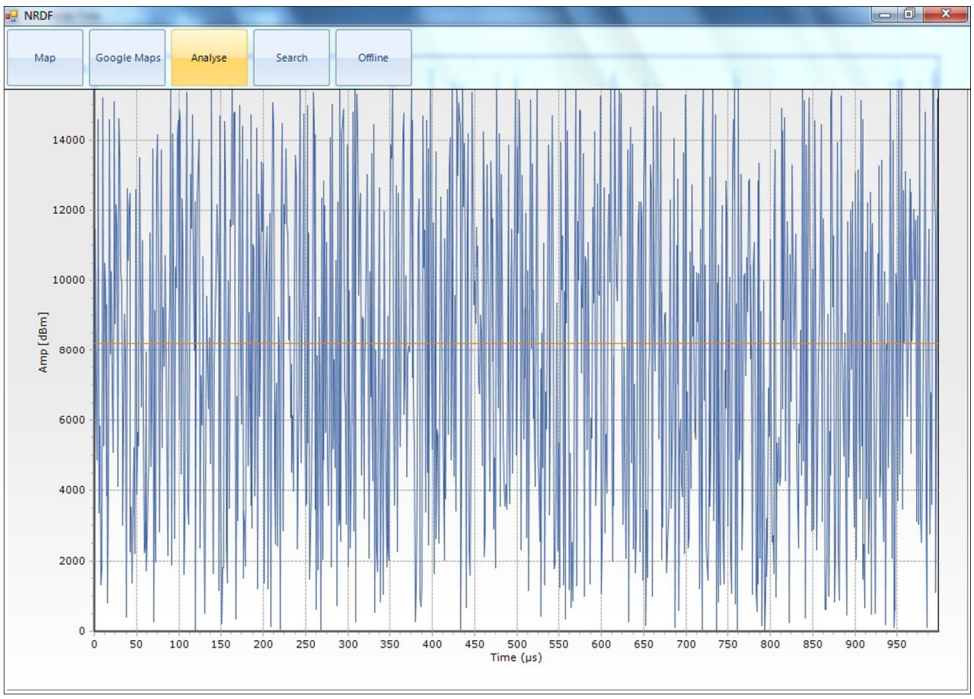



**شکل ۱۹ : پارامترهای زبانه‌های نقشه**

## ۹-۲- تحلیل پالس[۱۶۳]

این قسمت از نرم‌افزار برای مشاهده و آنالیز سیگنال‌های دریافت شده از کارت پردازش‌گر توسط کاربر طراحی شده است. در این قسمت با تنظیم پارامترهای کارت، دستور پردازش به کارت ارسال می‌شود؛ پس از شروع پردازش سیگنال‌های دریافت شده از کارت در نمودار موجود در این بخش نمایش داده می‌شوند. شکل ۲۰ فضای کلی این بخش را نمایش می‌دهد.

[۱۶۳] Pulse Analyzer




**شکل ۲۰ : نمایی از زبانه تحلیل پالس**

### ۹-۲-۱- پارامترها

شکل ۲۱ صفحه تنظیم پارامترهای این زبانه را نمایش می‌دهد. پارامترهای این بخش عبارتند از:

- **Card Settings :** این قسمت مربوط به تنظیمات کارت پردازش‌گر می‌شود. پارامترهای این بخش عبارتند از :
  - **IF Band :** فرکانس IF مورد استفاده کارت را مشخص می‌کند.
  - **Analog Mode :** در این بخش نحوه نمایش سیگنال دریافتی در نمودار تعیین می‌شود.
  - **Determinant :** آنتنی که سیگنال آن به عنوان سیگنال مرجع در نظر گرفته می‌شود را مشخص می‌کند.
  - **Manual Thr :** در صورتی که این گزینه انتخاب شده باشد مقدار حد آستانه[۱۶۴] در نظر گفته شده در کارت پردازش‌گر برابر مقدار این پارامتر قرار داده می‌شود، در غیر این صورت حد آستانه به صورت خودکار تعیین خواهد شد.

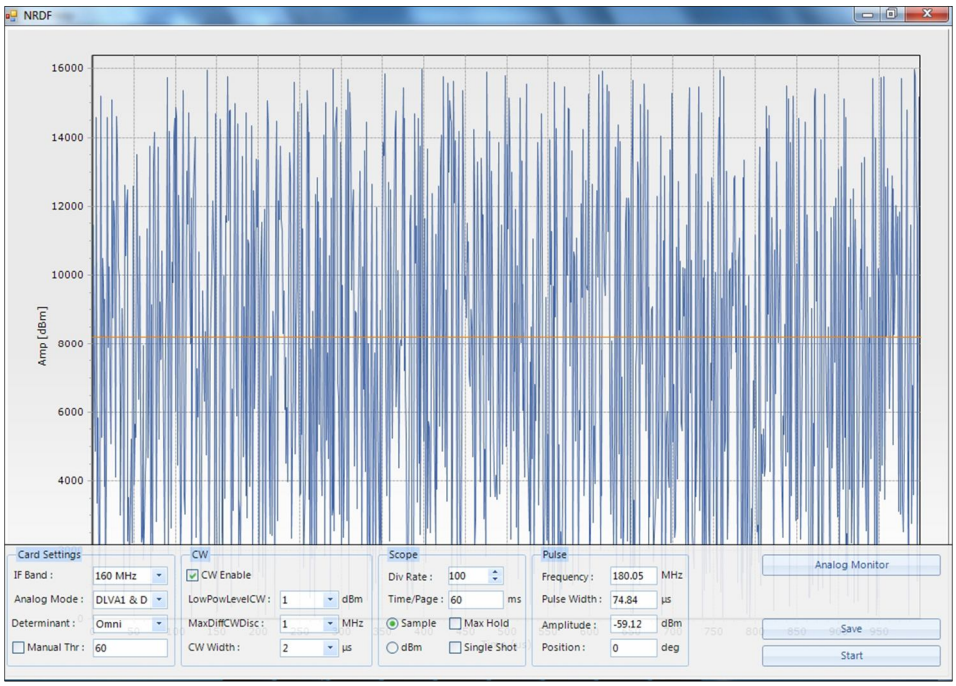
[۱۶۴] Threshold

- **CW**[۱۶۵] **:** در این بخش تنظیمات مربوط به بخش تشخیص موج پیوسته کارت پردازش‌گر انجام می‌گیرد:
  - **CW Enable :** با انتخاب این گزینه کارت پردازش‌گر در صورت داشتن قابلیت دریافت سیگنال‌هایCW، این سیگنال‌ها را دریافت می‌کند و این سیگنال‌ها در نمودار این بخش نمایش داده خواهند شد.
  - **LowPowLevelCW :** کمترین قدرت برای تشخیص سیگنال CW را مشخص می‌کند.
  - **MaxDiffCWDisc :** بیشترین تغییرات فرکانسی مورد تایید در سیگنال CW را مشخص می‌کند.
  - **CW Width :** این پارامتر حداکثر مقدار پهنای پالس قابل دریافت توسط کارت پردازش‌گر را تعیین می‌کند؛ سیگنال‌های دریافتی که پهنای پالسی بیشتر از مقدار این پارامتر ضرب در ۸۰ میکرو ثانیه داشته باشند به عنوان سیگنال CW شناخته خواهند شد.
- **Scope :** این قسمت حاوی تنظیمات مربوط به نحوه نمایش سیگنال‌های دریافتی در نمودار می‌باشد. پارامترهای این بخش عبارتند از:
  - **Div Rate :** این پارامتر برای مشخص کردن سرعت نمونه‌برداری کارت پردازش‌گر و در نتیجه محاسبه زمانبندی مناسب برای نمایش سیگنال‌های دریافتی استفاده می‌شود.
  - **Time/Page :** این پارامتر اندازه محور زمان (محور افقی) نمودار را به واحد میلی ثانیه مشخص می‌کند.
  - **Sample :** در صورتی که این گزینه انتخاب شده باشد، دامنه سیگنال دریافتی (محور عمودی) برحسب تعداد نمونه نمایش داده می‌شود.

[۱۶۵] Continuous Wave

- **dBm :** در صورتی که این گزینه انتخاب شده باشد، دامنه سیکنال دریافتی (محور عمودی) بر حسب dBm نمایش داده می‌شود.
- **Max Hold :** با انتخاب این گزینه سیگنال‌های دریافتی در حالت Max Hold نمایش داده می‌شوند.
- **Single Shot :** با انتخاب این گزینه نمایش سیگنال در نمودار پس از سپری شدن زمان مشخص شده در پارامتر Time/Page متوقف می‌شود.

- **Pulse :** در این بخش مشخصات سیگنال دریافتی پس از محاسبه نمایش داده می‌شود. این بخش حوای مشخصات زیر می‌باشد:
  - **Frequency :** نشان‌دهنده فرکانس سیگنال دریافتی به مگاهرتز می‌باشد.
  - **Pulse Width :** نشان‌دهنده پهنای پالس سیگنال به میکرو ثانیه دریافتی می‌باشد.
  - **Amplitude :** دامنه سیگنال دریافتی را به dBm نمایش می‌دهد.
  - **Position :** زاویه سیگنال دریافتی را در هر لحظه به واحد درجه نمایش می‌دهد.
- **کلید Start :** با استفاده از این کلید کارت پردازش‌گر شروع به کار کرده و سیگنال‌های دریافت می‌شوند.
- **کلید Save :** با فشردن این کلید، سیگنال دریافت شده در فایلی ذخیره می‌شود.
- **کلید Analog Monitor :** با فشردن این کلید سیگنال‌های دریافتی در نمودار نمایش داده می‌شوند.




**شکل ۲۱ : پارامترهای زبانه تحلیل پالس**

## ۹-۳- جستجو

این قسمت از نرم‌افزار NRDF برای جستجوی سیگنال‌های رادار در کل باند دستگاه جهت‌یاب راداری استفاده می‌شود. دو روش برای جستجوی سیگنال‌های رادار در نرم‌افزار پیش‌بینی شده است: جستجوی سوپر هترودین[۱۶۶] و جستجوی IFM[۱۶۷]. هر کدام از این دو روش با استفاده از گیرنده مربوط به خود عملیات جستجو را به دو روش کاملا متفاوت انجام می‌دهند. در روش سوپر هترودین، گیرنده مربوطه با گرفتن بازه فرکانسی مورد نظر و مقدار هر قدم[۱۶۸] به صورت رفت و برگشت[۱۶۹] به جستجوی فرکانس‌ها می‌پردازد؛ در این روش در هر لحظه یک فرمانس مورد بررسی قرار می‌گیرد. در روش IFM در هر لحظه کل باند ۱ تا ۲۰ گیگاهرتز مورد بررسی قرار می‌گیرد. نتیجه جستجوی سیگنال‌های هر کدام از دو روش بر روی نمودار مربوط به آن نمایش داده می‌شود؛ شکل ۲۲ فضای کلی این بخش را نمایش می‌دهد.

[۱۶۶] Super Heterodyne

[۱۶۷] Instantaneous Frequency Measurement

[۱۶۸] Step

[۱۶۹] Sweep


**شکل ۲۲ : نمای اصلی زبانه جستجو**

## ۹-۳-۱- پارامترها

شکل نمای صفحه تنظیم پارامترهای زبانه جستجو را نمایش می‌دهد. در ادامه پارامترهای مربوط به تنظیمات این بخش معرفی و شرح داده می‌شوند:

- **Super Heterodyne Search :** این قسمت حاوی پارامترهای مربوط به تنظیم گیرنده سوپر هترودین می‌باشد. این پارامترها عبارتند از:

  - **Start :** این پارامتر نشان‌دهنده فرکانس شروع جستجو می‌باشد. این پارامتر بر حسب مگاهرتز بوده و عددی بین ۱۰۰۰ تا ۲۰۰۰۰ مگاهرتز می‌تواند باشد.
  - **Stop :** این پارامتر فرکانس پایان جستجو را بر حسب مگاهرتز نشان می‌دهد. این پارامتر باید عددی بین ۱۰۰۰ تا ۲۰۰۰۰ مگاهرتز بوده و بزرگ‌تر از پارامتر Start باشد.
  - **Step :** مقدار قدم هر مرحله از جستجوی سوپر هترودین را نشان می‌دهد. مقدار این پارامتر بر حسب مگاهرتز می‌باشد.
  - **Dwell Time :** این پارامتر مقدار زمانی را که گیرنده بر روی هر فرکانس مکس کرده و به جمع‌آوری سیگنال می‌پردازد مشخص می‌کند. واحد این پارامتر میلی ثانیه می‌باشد.

- **Search Mode :** این پارامتر نحوه جستجوی فرکانس‌ها را نمایش می‌دهد؛ در صورتی که گزینه Automatic انتخاب شود، فرکانس‌ها به صورت خودکار به روشی که توضیح داده شد جستجو می‌شوند. در صورتی که گزینه Manual انتخاب شود فرکانس‌ها با توجه به فرکانس‌های مشخص شده در فهرست فرکانس‌های مورد جستجو در تنظیمات پیشرفته جستجو می‌شوند.
- **Display Mode :** این پارامتر نحوه نمایش سیگنال‌های جدید در نمودار را تعیین می‌کند. چنانچه گزینه Clear Write این پارامتر انتخاب شود نمودار آخرین مقدار دریافت شده به ازای هر فرکانس را نمایش خواهد داد؛ به عبارت دیگر در هر تکرار مقدار قبلی نمودار به ازای فرکانس جاری پاک شده و مقدار جدید نمایش داده می‌شود. اگر گزینه Hold Max برای این پارامتر انتخاب شود، همواره بزرگترین مقدار مربوط به هر فرکانس در نمودار نمایش داده می‌شود و با دریافت سیگنال جدید لزوما مقادیر قبلی پاک نمی‌شوند.
- **Sweep Type :** در صورتی که مقدار Forward برای این پارامتر انتخاب شود جستجو از ابتدای بازه تعیین شده شروع و با رسیدن به پایان بازه متوقف می‌شود، در حالی که اگر مقدار Repeat انتخاب شود جستجو بعد از رسیدن به پایان بازه مجددا از ابتدا بازه از سر گرفته می‌شود.

- **IFM Search :** این بخش مربوط به تنظیم پارامترهای گیرنده IFM می‌باشد. پارامترهای مربوط به این بخش عبارتند از:

- **Refresh Time :** این پارامتر مدت زمانی که اطلاعات جدید از گیرنده IFM دریافت شده و نمودار بر اساس آن به روز رسانی می‌شود را مشخص می‌کند. واحد این پارامتر میلی ثانیه می‌باشد.
- **Display Mode :** این پارامتر نحوه نمایش اطلاعات در نمودار مربوط به گیرنده IFM را تعیین می‌کند. در صورتی که گزینه Auto برای این پارامتر انتخاب شود مقیاس محور

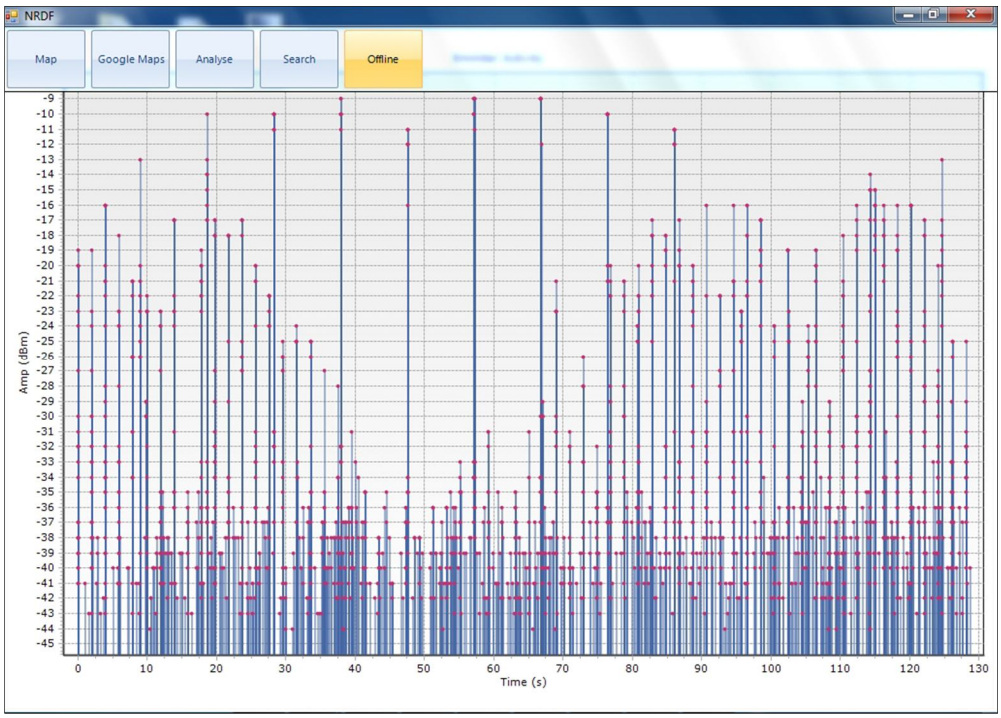
عمودی نمودار با توجه به تعداد پالس‌های دریافتی تغییر می‌کند. در صورتی که گزینه Fix برای این پارامتر انتخاب شود، مقیاس محور عمودی با توجه به مقدار مشخص شده برای پارامتر تعیین خواهد شد و با دریافت سیگنال‌های جدید مقیاس نمودار تغییر نمی‌کند.


**شکل ۲۳ : پارامترهای زبانه جستجو**

## ۹-٤- Offline

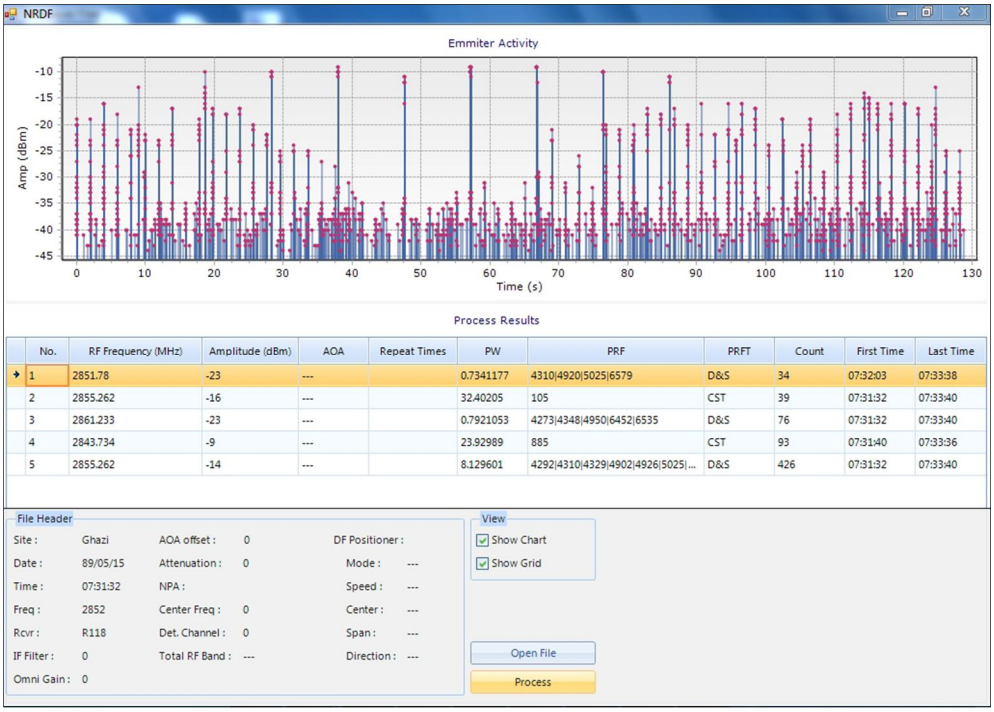
در این قسمت از نرم‌افزار، کاربر با انتخاب فایل‌های حاوی PDW با پسوند[۱۷۰] bsr. که توسط نرم‌افزار NRDF یا نرم‌افزارهای دیگر – با قالب مشخص– تولید شده‌اند از قابلیت‌های تحلیلی نرم‌افزار استفاده می‌کند. در این بخش پس از مشخص شدن فایل مورد نظر کاربر، محتویات آن در نمودار این بخش نمایش داده می‌شوند. این نمودار اطلاعات دامنه و زمان دریافت PDW موجود در فایل را نمایش می‌دهد و به کاربر اجازه می‌دهد به راحتی داده‌های فایل را بررسی و تحلیل کند. نمونه‌ای از این نمودار در شکل ٢٤ نشان داده شده است. یکی از ویژگی‌های این بخش از نرم‌افزار امکان بزرگنمایی در نمودار می‌باشد که به کاربر این اجازه را می‌دهد که علاوه بر بررسی دقیق‌تر اطلاعات موجود در فایل محدوده مورد نظر خود برای انجام پردازش و خوشه‌بندی را مشخص کند.

[۱۷۰] Extension




**شکل ۲٤ : نمایی از زبانه Offline**

پس از مشخص شدن محدوده مورد نظر کاربر و پس از فشردن کلید پردازش، PDWهای مشخص شده به واحد پردازش نرم‌افزار ارسال می‌شوند. در این واحد همانطور که در بخش‌های قبل توضیح داده شد، PDWهای دارای اطلاعات معیوب حذف شده و همچنین PDWهای مربوط به یک منبع تشخیص داده شده و به اصطلاح خوشه‌بندی می‌شوند. خروجی این واحد در جدولی که در شکل ۲۵ نشان داده شده است نمایش داده می‌شود.




**شکل ۲۵ : نتایج حاصل از خوشه‌بندی**

# فصل ۱۰– معرفی زبان C#

## ۱۰–۱–مقدمه

در ابتدا نام این زبان COOL بود که مخفف C like Object Oriented Language بود، زمانی که ماکروسافت پروژه را عمومی اعلام کرد، اسم آن به#C تغییر پیدا کرد.

زبان C# یک زبان جدید برنامه نویسی است که در ژوئن ۲۰۰۰ به عموم عرضه شد. این زبان توسط مایکروسافت ابداع و برای استانداردی به ECMA ارسال شده است. طراحی این زبان توسط تیمی از متخصصین مایکروسافت با مدیریت Andres Hejlsberg انجام شده است. این شخص از مهندسینی است که در تولید سایر زبان‌ها و محصولات مایکروسافت نظیر بورلند توربو C++ و بورلند دلفی نقش داشته است. تمرکز وی در C#، حفظ جنبه‌های مثبت و افزودن امکاناتی برای ایجاد زبانی بهتر بوده است.

C# یک زبان قوی و در عین حال انعطاف‌پذیر است و می‌تواند همانند سایر زبان‌های برنامه-نویسی برای تولید انواع برنامه‌های کاربردی بکار رود.در C# توان شما به قوه تخیل شما محدود می-شود و خود زبان هیچگونه قیدی را به شما تحمیل نمی‌کند. این زبان در حال حاضر برای ایجاد وب سایت‌های پویا، ابزار طراحی و تولید و حتی کامپایلرها بکار می‌رود.

C# یک زبان برنامه‌نویسی شی‌گرا (OOP) است. سایر زبان‌های برنامه‌نویسی نیز ویژگی‌های شی-گرایی را دارا هستند اما تعداد معدودی از آنها شی‌گرای کامل محسوب می‌شوند.

## ۱۰–۲– چرا C#؟

بسیاری از مردم اعتقاد دارند با وجود زبانهای نظیر C++، جاوا، پرل، ویژوال بیسیک و زبانهای موجود دیگر، نیازی به یک زبان برنامه‌نویسی جدید وجود ندارد زیرا زبان‌های موجود کلیه امکانات مورد نیاز را فراهم می‌کنند.

C# زبانی مشتق از C و C++ است اما کاملا از پایه ایجاد شده است. در این زبان جدید، امکانات دو زبان فوق بکار رفته و ویژگیهای جدیدی به آن افزوده شده است. بسیاری از ویژگی‌ها در جاوا نیز وجود دارند. بطور کلی اهداف مایکروسافت در ساخت این زبان عبارتند از :

۳– C# ساده است.

٤– C# امروزیست.

۵ – C# شی‌گراست.

علاوه بر دلایل فوق، دلایل دیگری نیز برای استفاده از C# وجود دارد :

۱ – C# قدرتمند و انعطاف‌پذیرست.

۲ – C# زبانی با حداقل کلمات کلیدی است.

۳ – C# ماجولارست.

٤ – C# محبوبیت عام را کسب خواهد کرد.

### ۱۰–۲–۱– C# ساده است

در این زبان بسیاری از مشکلات و پیچیدگی‌های سایر زبان‌ها نظیر جاوا و C++ حذف شده‌اند. برای نمونه می‌توان به حذف ماکروها، قالب‌ها (Template) ، وراثت چندگانه ( Multiple Inheritance) و کلاس‌های مجازی (Virtual) اشاره کرد. این امکانات معمولاً مشکلاتی را برای طراحان و برنامه‌نویسان C++ بوجود می‌آورند.

زبان C# ساده است زیرا براساس C و C++ بنا شده است. آشنایی با این دو زبان و یا حتی جاوا یادگیری C# را به مراتب تسهیل می‌کند. دستورات (Statements)، عبارات، عملگرها و بسیاری از توابع مستقیماً از C و C++برگرفته شده است اما بهبودهایی نیز برای تسهیل کار در آنها ایجاد شده است. برخی از این بهبودها در جهت حذف افزونگی‌ها و موارد تکراری و برخی دیگر در ایجاد تغییرات در نگارش دستورات بوده است. برای نمونه در C++ سه عملگر :: ، . و <- برای کار با اعضاء وجود دارد. بدیهی است درک زمان استفاده از این عملگرها کمی گیج کننده است. در C# تمامی این سه عملگر به عملگر . تبدیل شده‌اند بنابراین برای برنامه‌نویسان تازه کارتر مناسبتر است.

### ۱۰–۲–۲– C# امروزی است

چه چیزی یک زبان را امروزی می‌سازد؟ ویژگی‌هایی نظیر مدیریت استثناها ( Exception Handling)، پاکسازی حافظه (Garbage Collection) ، انواع داده‌ای قابل بسط و ایمنی کد از ویژگی-های قابل انتظار زبانهای امروزی هستند و C# تمامی آنها را داراست.

### ۱۰–۲–۳– C# شی‌گراست

کلید زبانهای شی‌گرا در مفاهیم کپسوله کردن (Encapsulation) [بسته‌بندی صفحات مشخصه و متدها برای ایجاد شیء] ، وراثت (Inheritance) و پلی مورفیسم (Polymorphism) [قابلیت تعریف مجدد یک روال] نهفته است. کپسوله کردن به معنی استقرار عملکرد و توانایی‌ها در یک بسته واحد

است. وراثت روش ساخت یافته‌ای برای بسط کد و توانایی‌های موجود در برنامه‌ها و بسته‌های جدید است. پلی مورفیسم نیز توانایی تطبیق با نیازمندیهای قابل انجام است.

### ۱۰-۲-۴- C# قدرتمند و انعطاف‌پذیرست

تنها محدودیت کار با C#قوه تخیل شماست، به عبارت دیگر خود زبان قیدی را به شما تحمیل نمی‌کند. این زبان برای ایجاد واژه پردازها، برنامه‌های گرافیکی، صفحات گسترده و حتی کامپایلرهای زبان‌های دیگر بکار می‌رود.

### ۱۰-۲-۵- C# زبانی با کلمات کلیدی محدود

تعداد کلمات کلیدی زبان C#انگشت شمار است.اکثر ین کلمات برای توصیف اطلاعات بکار می‌روند. ممکن است این ابهام برایتان بوجود آید که زبانی با کلمات کلیدی بهتر است، اما اینطور نیست.

در این زبان کلمات دیگری نیز وجود دارند که جزو کلمات کلیدی محسوب نمی‌شوند. برای نمونه می‌توان به get، set و value اشاره کرد.

### ۱۰-۲-۶- C# ماجولار است

کد C#می‌تواند (و باید) در بخش‌هایی با عنوان کلاس (Class) نوشته شود. این کلاس‌ها از روال-هایی با نام متدهای عضو (Member Method) تشکیل می‌شوند. کلاس‌ها و روال‌های آنها در سایر برنامه‌ها قابل استفاده مجدد می‌باشند. با انتقال بخشی از اطلاعات به کلاس و متدها می‌توانیم کدهای مفید و قابل استفاده ایجاد نماییم.

### ۱۰-۲-۷- C# محبوب عام

اگر چه C#یکی از جدیدترین زبانهای برنامه‌نویسی است اما به دلایل متعدد یکی از زبانهای بسیار محبوب خواهد بود. یکی از دلایل کلیدی، مایکروسافت و تکنولوژی دات نت (.NET) آنست.

مایکروسافت خواهان محبوبیت این زبان است. اگرچه این خواسته به تنهایی کافی نیست اما یک عامل مثبت است. C#زبانی است که در کارهای روزمره مایکروسافت بکار رفته است و بخش‌های بسیاری از محصولات مایکروسافت با این زبان برنامه‌نویسی شده‌اند. استفاده مایکروسافت از C#، نشاندهنده قابلیت این زبان در تحقق نیازهای برنامه نویسان است.

تکنولوژی NET.مایکروسافت دلیل دیگری برای موفقیت C# است. NET.روشی برای تولید و پیاده‌سازی برنامه‌های کاربردی است. از نظر مجازی کلیه زبان‌های برنامه‌نویسی قابلیت کار با NET. را دارند اما C# مناسبترین گزینه است.

دلایل دیگر محبوبیت C#ویژگی‌هایی است که قبل از این ذکر کردیم: سادگی، شیءگرا بودن، ماجولار بودن و انعطاف‌پذیری.

یکی از مزایای c#.net قدرت دسترسی فوق العاده بالای آن به سیستم است.

## ۱۰-۳- C# و سایر زبان‌های برنامه‌نویسی

به عقیده مایکروسافت C#زبانی است که توان C++ را با سادگی ویژوال بیسیک ارائه می‌کند. در توانمندی C#بحثی نیست اما آیا این زبان واقعاً به سادگی ویژوال بیسیک است؟ در پاسخ باید گفت این زبان بسادگی ویژوال بیسیک ویرایش شش نیست اما با سادگی ویژوال بیسیک NET.(ویرایش هفت) که بطور کامل بازنویسی شده است قابل قیاس است. به عبارت دیگر نوشتن بسیاری از برنامه‌ها در C# به کدنویسی کمتری نیاز دارد.

اگر چه در C#برخی از ویژگی‌های پیچیده حذف شده‌اند اما در عمل چیزی از قدرت یا توانایی-های آن کاسته نشده است. برخی از خطاهای برنامه‌نویسی رایج در C++ در C# اتفاق نمی‌افتند به همین دلیل برنامه‌نویسی با C#سریعتر انجام می‌شود.

برنامه‌های C#برای اجرا در Common Language Runtime (CLR)ایجاد می‌شوند. به عبارت دیگر برنامه اجرایی تولید شده C#برروی کامپیوتر فاقد CLRیا مشابه آن اجرا نمی‌شوند. منظور از برنامه اجرایی برنامه‌ایست که قابلیت اجرا در کامپیوتر را داراست.

مزیت تولید برنامه‌هایی برای محیط اجرا، قابلیت انتقال آنهاست. در زبانهای قدیمیتر نظیر C یا C++ ، اجرای فایل اجرایی تولید شده در یک سیستم عامل در سیستم عامل دیگر امکان‌پذیر نیست مگر آنکه در آن محیط مجددا کامپایل شود. برای مثال ایجاد یک برنامه Cقابل اجرا برروی دو سیستم عامل لینوکس و ویندوز نیاز به دو فایل اجرایی متفاوت دارد: یکی برای لینوکس و دیگری برای ویندوز اما در C#تنها یک فایل اجرایی تولید شده و در هر دو محیط اجرا می‌شود.

برای اجرای یک برنامه در سریعترین وضعیت ممکن لازم است لازم است فایل اجرایی واقعی آن را تولید کنید. یک کامپیوتر به دستورات دیجیتال، باینری یا همان زبان ماشین نیاز دارد. بنابراین برنامه‌ها باید از کد منبع به زبان ماشین ترجمه شوند. این ترجمه را برنامه‌ای با نام مترجم (کامپایلر)

انجام می‌دهد. در برنامه‌های C یا C++فایل حاصل بدون هیچ کار اضافی قابل اجراست.اما کامپایلرهای C#بجای کد زبان ماشین یک فایل زبان میانی (Intermediate language (IL تولید می‌کنند. این فایل بطور مستقیم قابل اجرا نیست و نیاز به ترجمه یا کامپایل دیگری برای اجرا دارد. بنابراین CLR یا برنامه زمان اجرای C#، کامپایل نهایی را مطابق نیازمندی‌های محیط اجرا انجام می‌دهد.

ترجمه نهایی یکی از نخستین کارهایی است که CLRبرروی فایل ILانجام می‌دهد. در این فرآیند، CLRکد ILقابل انتقال را به کد زبان ماشین کامپیوتر اجراکننده برنامه تبدیل می‌کند. CLRتنها بخش‌های مورد استفاده برنامه را ترجمه کرده و در نتیجه در زمان صرفه‌جویی می‌کند. پس از تبدیل ILبه زبان ماشین نیازی به ترجمه مجدد نیست و بخش تبدیل شده برنامه پس از ذخیره، در اجراهای متفاوت برنامه بکار می‌رود.

در شکل‌های زیر یک مقایسه‌ی بین زبان‌های مختلف را نشان می‌دهد.

| Feature | D | C | C++ | C# | Java | Delphi | Eiffel | Ada | Haskell | Sather | Common Lisp | Smalltalk | Perl | Python | Ruby |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Static typing | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Optional (Lib) | No | No | No | No |
| Dynamic typing/ polymorphism | | No | Polymorphic | | | | | | Polymorphic | | Dynamic | Dynamic | | Dynamic | Dynamic |
| Garbage Collection | Yes | No | No | Yes | Yes | No | Yes | Unspec | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Strong typedefs | Yes | No | No | No | No | Yes | | Yes | Yes | Yes | N/A | N/A | N/A | N/A | N/A |
| Aliases | Yes | Yes | Yes | Yes | For Objects Only | Yes | Yes | Yes | Yes | Yes | Yes | | Yes | Yes | Yes |
| Constants | Yes | No | Yes | Yes | For Non-Objects? Only | Yes | | | | | | | | | |
| **Value types** | **D** | **C** | **C++** | **C#** | **Java** | **Delphi** | **Eiffel** | **Ada** | **Haskell** | **Sather** | **Common Lisp** | **Smalltalk** | **Perl** | **Python** | **Ruby** |
| Typed Enumeration/ Boolean | Yes | Yes (booleans) | Yes | Yes | Yes (v1.5) | Yes | No | Yes | Yes | | Keywords | Lib | | | Yes (booleans) |
| Long double floating point (80bit) | Yes | Yes | Yes | Lib | No | Yes | | Yes | Yes | | Most | Most | | Yes (no single precision floats) | |
| Complex and Imaginary | Yes | Yes | Lib (std.) | Lib | No | No | | Yes | Std. Lib. | Yes | Yes | Most | Lib | Yes | Lib |
| Rational | | No | No | | Lib | | | | Std Lib | | Yes | Yes | Lib | No | Lib |
| Arbitrary Precision Arithmetic | | No | No | | Lib (std.) | | | | Yes | | Yes | Yes | | Stdlib | |
| String | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes |
| Regular Expression | Lib | Lib | Lib | Lib | Lib (std., v1.5) | Lib | Lib | Lib | Std. Lib. | Lib | Lib | Lib | Yes | Lib | Yes |
| **Arrays** | **D** | **C** | **C++** | **C#** | **Java** | **Delphi** | **Eiffel** | **Ada** | **Haskell** | **Sather** | **Common Lisp** | **Smalltalk** | **Perl** | **Python** | **Ruby** |
| Lightweight array | Yes | Yes | Yes | No | No | Yes | | Yes | Lib | No | Yes | Yes | No | Lib | No |
| Array bounds checking | Yes | No | No | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Resizable array / Vector | Yes | No | Lib (std.) | Yes (v2) | Yes (v1.6) | Yes | Yes | No | Lib | Yes | Yes | Yes | Yes | Yes | Yes |
| Array slicing | Yes | No | No | No | Lib (std.) | Yes | Yes | Yes | Lib | No | Yes | Yes | Yes | Yes | Yes |
| Array of bits / Finite set | Yes | No | Yes | No | Lib (std.) | Yes | Yes | Yes | Lib | | Yes | Yes | Lib | No | No |

## ۱۰-۴- ویژگی‌های زبان C#

برخی از تفاوت‌های زبان سی شارپ با C و C++ عبارتند از:

هیچ تابع یا متغیر سراسری (Global) وجود ندارد، تمام متدها و اعضا بایستی در داخل کلاس‌ها تعریف شوند. این امر ممکن است، هر چند برای استفاده از متغیرها و توابع عمومی باید از متدها و متغیرها در کلاس‌های عمومی استفاده کرد.

- متغیرهای عمومی، بر خلاف C و C++ ، نمی‌توانند بلاک‌های پیوستی را در بر بگیرند.
- C#دارای یک نوع داده بولی است.(bool) برخی از عبارت‌ها مانند while و if که شرطی هستند، نیازمند یک عبارت نوع بولی هستند.همان طور که C++ نیز دارای نوع داده بولی است،این نوع داده به راحتی میتواند به یا ازInteger ها تبدیل شود، و عبارتی مانند if(a) نیازمند این امر است که a از یک نوع قابل تبدیل به bool یا اشاره گر باشد.کامپایلر سی شارپ برنامه نویس را در این شرایط مجبور به استفاده از عباراتی می‌کند که به درستی یک مقدار bool را برمیگردانند. بنابراین دستوری مانند if(a = b) باعث بروز خطا می‌شوند.(به جای = بایستی از = = استفاده شود).
- در سی شارپ ، اشاره گرهای به حافظه بایستی فقط در داخل بلوکهای unsafe استفاده شوند و برنامه در این حالت برای اجرا نیاز به اجازه از کاربر دارد. بیشتر دسترسی شی از طریق شی امن است که یا همیشه در حال اشاره به شی صحیح موجود است یا یک مقدار Null دارد. اشاره گری به شی به درد نخور یا بلاک حافظه رندم غیر ممکن است.اشاره گر نا امن می‌تواند به نمونه‌ای از value-type، آرایه، رشته یا بلاکی که حافظه به آن داده شده‌است اشاره نماید.کدی که به عنوان نا امن علامت نخورده باشد، هنوز می‌تواند اشاره گرها را از سیستم بازیابی یا در آن ذخیره کند ولی نمی‌تواند مرجع جدیدی به آنها اختصاص دهد.

- حافظه ساماندهی شده نمی‌تواند صریحا آزاد شود، ولی به طور خودکار به عنوان به درد نخور تلقی می‌شود.انتخاب آدرس‌های به درد نخور حافظه نفوذ ناپذیر است. هم چنین C# با استفاده از عبارات، پشتیبانی مستقیمی از پایان اجباری می‌کند(پشتیبانی از اصطلاح Resource Acquisition Is Initialization).
- وراثت چندگانه از کلاس‌ها در این زبان پشتیبانی نمی‌شود.البته یک کلاس امکان ارث بری از تعداد نامحدود واسط‌ها را دارد. پشتیبانی نکردن از وراثت چندگانه به دلیل اهداف معماری این زبان در CLI و برای جلوگیری از پیچیدگی است.
- سی شارپ بسیار type safe تر از C++ است. تنها تبدیلات ضمنی مثل تبدیل نوع داده کوچکتر به بزرگتر یا تبدیل نوع مشتق شده به نوع پایه به طور پیش فرض و بدون خطا صورت می‌پذیرد.هیچ تبدیل ضمنی ای میانBooleanها وInteger ها وجود ندارد و هر تبدیل user-definedبایستی به صراحت با یکی از کلمات explicit یا implicit نشانه گذاری شود. تبدیل b به a در حالتی که a یک Integer و b یک double باشد در زبان C++ مجاز است اما در سی شارپ به یک خطای زمان کامپایل منجر می‌شود(بایستی به صورت explicit تعریف شود)
- اعضای Enumeration در داخل محدوده شخصی خود قرار دارند.

## ۱۰–۵– متدهای برنامه در C#

برنامه‌های C#متشکل از قسمت‌های متعددی از جمله متدها و کلاس‌ها هستند. برنامه‌نویس اقدام به ایجاد و ترکیب متد و کلاس‌های جدید با کلاس‌های از قبل آماده شده موجود در FCL[۱۷۱]می‌کند.

[۱۷۱] .NET Framework Class Library

FCL حاوی کلکسیون با ارزشی از کلاس‌ها و متدها به منظور انجام محاسبات ریاضی، کار با رشته‌ها، کار با کارکترها، عملیات ورودی/خروجی، تست و بررسی خطا و بسیاری از کاربردهای مناسب دیگر است. این Framework بدلیل اینکه نیازهای متعدد یک برنامه‌نویس را تدارک دیده است، کار برنامه‌نویس را آسانتر کرده است.

فرمت کلی تعریف یک متد بصورت زیر است

**(لیست پارامترها)  نام متد  نوع مقدار برگشتی**

{

**اعلان‌ها و عبارات**

}

نام متد می‌تواند هر شناسه معتبری باشد. لیست پارامترها یک لیست جدا شده با کاما از یکدیگر است که حاوی اسامی پارامترهای دریافتی متد به هنگام فراخوانی آن است. بایستی برای هر پارامتر موجود در تعریف متد، یک آرگومان متناظر با آن در بخش فراخوان متد وجود داشته باشد. اگر متدی داشته باشیم که هیچ مقداری دریافت نمی‌کند، لیست پارامتری آن خالی خواهد بود (پس از نام متد یک جفت پرانتز خالی قرار داده می‌شود). بخش اعلان و عبارات، بدنه متد را تشکیل می‌دهند.

## ۱۰-۶- Garbage Collection

کلمه کلیدی New حافظه مورد نیاز برای شی را اخذ کرده و سپس سازنده شی را فراخوانی می-کند. ممکن است سازنده به منابع دیگری از سیستم نظیر اتصالات شبکه و پایگاه داده دسترسی داشته باشد. شی‌ها باید با استفاده از یک روش اصولی حافظه و منابع در اختیار گرفته خود را پس از اتمام کار خود به سیستم بازگردانند. عدم آزاد سازی چنین منابعی موجب فقدان خواهد شد.

برخلاف C و C++، که برنامه‌نویسان بایستی خود مدیریت حافظه را بصورت صریح انجام دهند. C#خد مدیریت حافظه را انجام می‌دهد. NET Frathiswork. عمل Garbage Collection حافظه را به منظور بازگرداندن حافظه‌ای که دیگر به آن نیاز نیست، انجام می‌دهد. زمانی که عمل Garbage Collection اجرا می‌شود شی‌هایی که در برنامه مورد مراجعه نیستند، تشخیص داده می‌شوند. چنین شی‌هایی می‌توانند در زمان اجرا یا در توالی اجرای جمع کننده Garbage جمع‌آوری شوند. از اینرو فقدان حافظه در زبان‌های نظیر C و C++به نسبت C#بسیار بیشتر رخ می‌دهد. با اینکه C#به صورت اتوماتیک عمل Garbage Collection را انجام می‌دهد، اما بهترین روش برای مدیریت منابع نیست. منابع مشخص همانند اتصالات شبکه، پایگاه داده، انتقال فایل نیاز دارند تا بصورت صریح و توسط

برنامه‌نویس مدیریت شوند. یکی از تکنیک‌های بکار رفته در مورد این منابع تعریف کردن یک متد پایان دهنده است که عمل پاکسازی شی را قبل از اخذ حافظه شی توسط جمع کننده Garbage انجام دهد.

هر کلاس فقط می‌تواند یک نابود کننده داشته باشد نام نابودکننده پس از کارکتر ~ نام کلاس آورده می‌شود. برای مثال نابودکننده کلاس Time می‌تواند ()Time ~ باشد. نابودکننده‌ها، آرگومانی دریافت نمی‌کنند و از این رو نمی‌توانند OverLoad شوند. زمانی که جمع‌کننده Garbage مبادرت به حذف یک شی از حافظه می‌کند، ابتدا نابودکننده شی را به منظور آزاد کردن منابع بکار رفته توسط کلاس، فعال می‌کند. با این همه، دقیقاً نمی‌توان از زمان فراخوانی نابودکننده اطلاع پیدا کرد، چرا که نمی‌توانیم دقیقاً بگوییم که در چه زمانی عمل Garbage Collection رخ خوهد داد. هنگامی که برنامه خاتمه می‌پذیرد، نابودکننده برای هر شی که تابحال در اختیار Garbage Collection قرار نگرفته، فراخوانی می‌شود.

کپسوله کردن :

کپسوله کردن به مفهوم ایجاد بسته‌هایی متشکل از تمامی چیزهای مورد نیاز است. به عبارت دیگر در برنامه‌نویسی شیءگرا می‌توانیم یک شی یا بسته نظیر یک دایره را بگونه‌ای ایجاد کنید که تمامی عملیات مورد نظر شما بر روی دایره را انجام دهد. (برای مثال نگهداری اطلاعات یک دایره نظیر مختصات مرکز و اندازه شعاع، نحوه رسم آن، محاسبه محیط و مساحت و ... )

بسته‌بندی دایره به کاربر امکان می‌ئهد بدون توجه به چگونگی کارکرد دایره، تعامل لازم با آن را انجام دهد. به این ترتیب کارکرد درونی دایره از چشم کاربر محفوظ می‌ماند. علاوه بر آن مادامیکه کاربر می‌تواند دایره مورد نظر را بدست بیاورد نیازی به اطلاعات داخلی ذخیره آن ندارد.

پلی مورفیسم

پلی مورفیسم توانایی بکارگیری چند شکل یا فرم است. این توانایی در برنامه‌نویسی شیءگرا حداقل در دو زمینه کاربرد دارد. نخست آنکه می‌توانیم یک شکل یا روال را به اشکال متفاوت فراخوانی کنید و هر بار نتیجه واحدی را بدست آورید. برای مثال فرض کنید می‌خواهید شیء دایره را برای محاسبه مساحت آن فراخوانی کنید. اینکار می‌تواند با دردست داشتن سه نقطه متفاوت از دایره و یا یک نقطه و اندازه شعاع دایره انجام شود. در هر دو حالت دریافت نتایج یکسانی را انتظار دارید. در زبانهای روالی نظیر Cبرای اینکار نیاز به دو روال متفاوت با اسامی متفاوت وجود دارد. در #Cکماکان این دو روال وجود دارند اما می‌توانیم نام واحدی را به آنها تخصیص دهیم. کلیه برنامه-

های ما و دیگران روال دایره را فراخوانی کرده و اطلاعات مورد نیاز را به آن منتقل می‌کنند. برنامه دایره بطور خودکار و براساس داده‌های دریافتی، روال قابل استفاده را پنهان از چشم کاربران فراخوان روال تعیین می‌کند.

وراثت

وراثت پیچیده‌ترین مفهوم شیءگرایی است. داشتن یک دایره خوب است اما ایا کره بهتر نیست؟ کره نوع خاصی از دایره است که تمامی ویژگی‌های دایره را به انضمام بعد سوم داراست. به عبارت دیگر کره نوع خاصی از دایره را گرفته و سپس چیزهایی را به آن اضافه کرده است. با این وصف کره می‌تواند خصوصیات دایره را به ارث ببرد. توانایی به ارث بردن از مشخصه‌های وراثت است.

استفاده مجدد

یکی از نکات کلیدی در زبان‌های شیءگرا، مفهوم استفاده مجدد است. در اینگونه مفاهیم زبان‌ها می‌توانیم هر کلاس را بدفعات و برای ایجاد اشیاء متعدد بکار بریم. با در نظر داشتن مفاهیم وراثت و سایر ویژگی‌های بررسی شده قبلی می‌توانیم روال‌هایی با قابلیت استفاده مجدد در انواع برنامه‌ها و اشکال مختلف ایجاد کنیم. کپسوله کردن ایجاد روال‌های تست شده و تائید شده را امکان‌پذیر می‌سازد بنابراین در استفاده‌های بعدی نیازی به تست جزئیات عملکرد روال‌ها نداریم. کافیست آنها را به شکل صحیح فراخوانی کنیم و از سرعت و سهولت کاربری آنها بهره‌مند شویم.

## ۱۰-۷- داده انتزاعی

با وجود کلاس‌ها در #Cایجاد داده‌های خاص که داده‌های انتزاعی ADT (Abstract data Type) نامیده می‌شوند، ساده‌تر شده است، که اقدام به پنهان‌سازی خود از دید سرویس گیرنده‌ها می‌کنند. مشکلی که در زبان‌های برنامه‌نویسی روالی وجود دارد این است که غالباً کد سرویس گیرنده برای استفاده از داده‌های خاصی نوشته شده و در صورتیکه کد واسط تغییر یابد، بایستی مجدداً نوشته شود. در صورتیکه ADTاین مشکل را با تدارک دیدن پیاده‌سازی مستقل واسط‌ها از سرویس‌گیرنده‌های خود برطرف کرئه است. ایجاد کننده یک کلاس می‌تواند پیاده‌سازی کلاسی را بدون اینکه مجبور به تغییر سرویس‌گیرنده‌های کلاس نماید، تغییر دهد.

ثوابت :

ثوابت مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند.

اگر متغیری هرگز نباید تغییر کند، آن را ثابت در نظر بگیرید. این امر به رفع خطاهایی کمک می کند که ممکن است در صورتی روی دهند که مقدار متغیر تغییر یابد.

نامگذاری برای ثوابت از قانون نامگذاری برای متغیرها تبعیت می کند.

۱. استفاده از دستور const

برای تعریف ثوابت با دستور const به صورت زیر عمل می شود:

const <مقدار> = <نام ثابت> <نوع داده>

مانند:

```
const int n= ۱۰۰, count=۵۰;
const char x = 'a';
```

اعضای داده که به طور ضمنی به صورت const معرفی می شوند، static هستند و باید در معرفی خود مقداردهی شوند.

معرفی یک عضو داده کلاس به صورت const و عدم مقداردهی آن در معرفی آن کلاس، یک خطای دستوری است.

انتساب مقدار به یک متغیر const بعد از مقداردهی آن متغیر، یک خطای کامپایل است.

معرفی یک عضو const به صورت static یک خطای دستوری است، زیرا اعضای const به طور ضمنی static هستند.

۲. استفاده از دستور readonly

برای تعریف ثوابت با دستور readonly به صورت زیر عمل می شود:

Readonly <نوع داده> <نام ثابت> = <مقدار>

مانند:

```
class classX
 {
      readonly int x = ۱۰۰;
      readonly int y;
public classX (int i)
      {
         y = i;
      }
}
```

به اعضای داده ای که به صورت readonly معرفی می شوند تنها یک بار می توان مقداری را نسبت داد، یا در معرفی خود و یا در سازنده کلاس.

متغیرها

به بیان بسیار ساده، مکانهایی جهت ذخیره اطلاعات هستند. شما اطلاعاتی را در یک متغیر قرار می‌دهید و از این اطلاعات بوسیله متغیر در عبارات C# استفاده می‌نمایید. کنترل نوع اطلاعات ذخیره شده در متغیرها بوسیله تعیین کردن نوع برای هر متغیر صورت می‌پذیرد.

C# زبانی بسیار وابسته به انواع است، بطوریکه تمامی عملیاتی که بر روی داده‌ها و متغیرها در این زبان انجام می‌گیرد با دانستن نوع آن متغیر میسر می‌باشد. قوانینی نیز برای تعیین اینکه چه عملیاتی بر روی چه متغیری انجام شود نیز وجود دارد.(بسته به نوع متغیر)

## ۱۰-۸- زبان واسط IL[۱۷۲]

هنگامی که برنامه‌ای که در آن توابع در کتابخانه کلاس .NET استفاده شده است را کامپایل می-کنیم، بلافاصله کد قابل فهم سیستم عامل و یا کد محلی تولید نمی‌شود. در عوض کد ما به زبانی به نام زبان سطح میانی مایکروسافت و یا به اختصار ILتبدیل می‌شود. این کد برای سیستم‌عامل خاصی نیست و همچنین منحصر به زبان C#نیز نیست. به عبارت دیگر کد زبانهای دیگر نیز می‌تواند به IL تبدیل شود. کدهای زبان‌های دیگر که از چارچوب .NETاستفاده می‌کنند نیز (مانند ویژوال بیسیک) هنگام کامپایل ابتدا به زبان ILتبدیل می‌شوند.به زبا ساده‌تر می‌توان گفت که IL زبان واسطی است که در مرحله اول کامپایل برنامه ها، تمامی کدهای نوشته شده با زبان های دات نت (اعم از سی شارپ، وی بی و ..) به این زبان ترجمه می شوند. (باز هم جهت سازگاری بین زبان های مختلف در دات نت).

اما برای اجرای یک برنامه توسط سیستم‌عامل یک مرحله دیگر نیز مورد نیاز است. این مرحله وظیفه‌ی کامپایلر Just-In-Time و یا به اختصار JIT کامپایلر است. این کامپایلر کد ILیک برنامه را دریافت کرده و آن را به کدی تبدیل می‌کند که به وسیله سیستم‌عامل قابل اجرا باشد. به از اینکه این تبدیل توسط JIT انجام شد، سیستم عامل می‌تواند برنامه را اجرا کند.

تکه کدی به زبان C#

```
public int add( int num۱, int num۲)
```

---

[۱۷۲] Language Runtime

```
{
return num١ + num٢;
}
```

همان تکه کد به زبان VB

```
Public Function add (ByVal num١ As Integer, ByVal num٢ As Integer) As Integer
Return num١ + num٢
End Function
```

تکه کد فوق به زبان IL

```
method public hidebysig instance int٣٢ add(int٣٢ num١, int٣٢ num٢) cil managed.
{
maxstack ٢.
(locals init .
 [٠]int٣٢ CS□١□٠٠٠٠)
L_٠٠٠٠: nop
L_٠٠٠١: ldarg.١
L_٠٠٠٢: ldarg.٢
L_٠٠٠٣: add
L_٠٠٠٤: stloc.٠
L_٠٠٠٥: br.s L_٠٠٠٧
L_٠٠٠٧: ldloc.٠
L_٠٠٠٨: ret
}
```

**نکته** : شما می توانید کل برنامه خود را به زبان IL بنویسید و سپس آن را اجرا کنید. ولی اصولا هیچ آدم عاقلی این کار را نمیکنه.

## ۱۰-۹-معرفی Language-Integrated Query (LINQ)

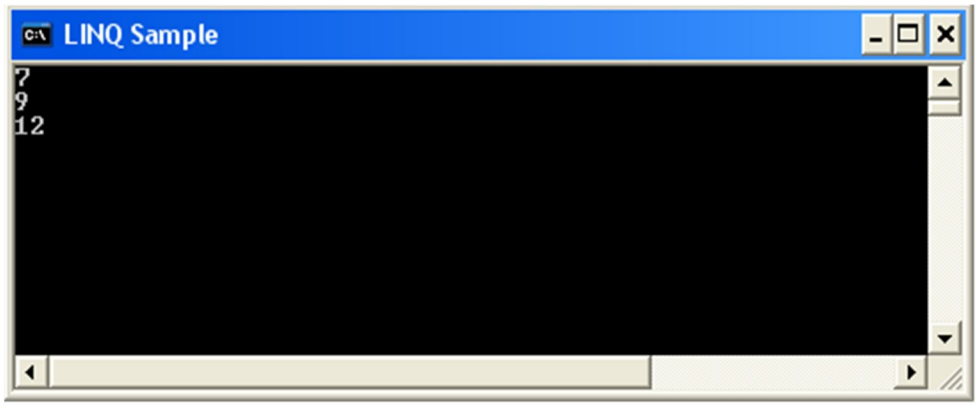
LINQ، یک زبان پرس و جوی قابل انعطاف و همه منظوره برای بسیاری از انواع منبع داده ها است( مثل انتخاب اشیا شناور، سندهایXML ، بانک‌های اطلاعاتی و . . .)، به زبان ساده، پروژه LINQبه عنوان یک راه حل "نگاشت شی‌گرا - رابطه‌ای" توکار عمل می‌کند. بنابراین، گستره وسیعی را در بر می‌گیرد و دارای انواع زیر است:

§ **LINQ**-to-Objects - talks to in-memory objects
§ **LINQ**-to-SQL - talks to SQL Server databases
§ **LINQ**-to-XML - talks to hierarchical data represented in XML
§ **LINQ**-to-DataSets - talks to DataSet objects and underlying DataTables with their relationships
§ **LINQ**-to-Entities - talks to "entities", part of ADO.NET ۳.۰

نمونه ای ازکاربر LINQ در شکل زیر نشان داده شده است :

```
static void Main(string[] args)
{
    var int_array = new int [] { 1, 2, 7, 9, 12 };

    var selective_array = from c in int_array where c > 5 select c;

    foreach (var selected in selective_array)
    {
        Console.WriteLine(selected);
    }
}
```

توضیح : در مثال بالا ابتدا یک آرایه‌ی intی با مقداردهی اولیه تعریف شده است. سپس با استفاده از دستورات LINQ ( که جز کلمات کلیدی سی شارپ محسوب می شوند)، آرایه ای با اعضای بزرگتر از ۵ انتخاب و در متغیر ضمنی محلی selective array ذخیره می شود. در نهایت اعضای selective array به شکل زیر چاپ می شوند:



# فصل ۱۱– داده کاوی

## ۱۱–۱–مقدمه

به صورت سنتی، تحلیل گران فرآیند استخراج اطلاعات مفید از داده های ذخیره شده را اجرا می کنند. اما افزایش حجم داده ها در علوم و تجارت مدرن، به نام شیوه های مبتنی بر کامپیوتر نامیده می شود. مادامی که اندازه و پیچیدگی مجموعه داده ها رشد می کند، یک تغییر و جابجایی غیرقابل تصوری از تحلیل داده های دستی به سمت تحلیل داده های خودکار و تا حدودی غیر مستقیم با استفاده از ابزار پیچیده و پیشرفته حاصل می شود. در حقیقت، فناوری های جدید کامپیوتر، شبکه و سنسورها، جمع آوری داده ها و سازماندهی آنها را تقریبا به یک امر بدون تلاش و زحمت تبدیل نموده اند. هرچند، داده های به دست آمده برای آنکه مفید باشند، باید با استفاده از داده های ذخیره شده به اطلاعات و دانش مفید تبدیل شود. داده کاوی در حقیقت فرایند کاملی از اعمال متولوژی مبتنی بر کامپیوتر می باشد که شامل تکنیک های جدید برای اکتشاف دانش از داده است.

باید توجه داشت، دنیای مدرن در حقیقت دنیای داده گرا می باشد و ما در محاصره داده ها، چه عددی و چه انواع دیگری قرار گرفته ایم. این داده ها باید تحلیل و پردازش شوند تا تبدیل به اطلاعاتی گردند که به عنوان ابزاری در جهت اطلاع رسانی، آموزش و سایر روش ها به درک و تصمیم گیری ما کمک کنند. در واقع این دوران، عصر اینترنت، انترانت انبار داده ها و مراکز داده ای و بلاخره پارادایم مبانی تحلیل داده های کلاسیک به تغییرات مناسب در این مبحث می باشد. مجموعه های بسیار بزرگی از داده ها (گاهی اوقات صد ها میلیون رکورد منفرد) در انبار های داده ای متمرکز ذخیره می شوند.

در سال های اخیر، فناوری داده کاوی به یکی از داغ ترین موضوعات برای تصمیم سازان تبدیل شده است، زیرا فناوری داده کاوی یک هوش علمی و تجاری با ارزش و تا حدودی پنهان را برای حجم زیادی از داده ها و سوابق کاربرد فراهم می سازد – هرچند اساسا داده کاوی یک فناوری جدید محصوب نمی شود. استخراج اطلاعات و دانش از داده های ذخیره شده یک مفهوم کاملا تعریف شده در مباحث علمی و پزشکی است. آن چه که جدید است، همگرایی چندین نظام و تکنولوژی است که فرصت کاملا ممتازی را برای داده کاوی در دنیای معاصر ایجاد کرده است.

داده‌کاوی امروزه در حوزه‌ی فناوری اطلاعات توجه زیادی را به خود جلب کرده است. این محبوبیت زیاد به سبب مواجه بودن با حجم بسیار بالای داده و نیاز قطعی به تبدیل این داده‌ها به اطلاعات و دانش‌های مفید است. اطلاعات و دانش‌های به دست آمده از این طریق، آنگاه جهت کاربردهای گوناگون – از تحلیل بازار، تشخیص کلاهبرداری و حفظ مشتری تا کنترل تولید و کاوش علمی – می‌تواند مورد استفاده قرار گیرد.

تصمیمات مهم سازمان‌ها اغلب نه بر اساس داده‌های ذخیره شده در مخازن داده ـ که خود حاوی اطلاعات فراوانی هستند ـ بلکه براساس درک شهودی تصمیم‌گیرندگان صورت می‌گیرد، و علت آن هم عدم در اختیار داشتن ابزارهایی جهت استخراج دانش‌های ارزشمند پنهان شده در میان مقادیر فراوان داده‌هاست. به علاوه، فناوری‌های مرتبط با سیستم‌های خبره که نوعاً در آن‌ها باید کاربران یا خبرگانِ دامنه به صورت دستی دانش را وارد پایگاه دانش کنند، با توجه به وجود چنین رویه‌ی دستی از خطای بالایی برخوردار بوده و در عین حال زمان‌بَر و پرهزینه می‌باشد.

شکاف گسترده بین داده و اطلاعات، توسعه‌ی سیستماتیک ابزارهای داده‌کاوی را می‌طلبد تا بدین وسیله گورستان‌های داده به قطعه‌های طلایی دانش تبدیل شوند. بنابراین در یک تعریف ساده می‌توان داده‌کاوی را استخراج دانش از مقادیر زیاد داده دانست، که امروزه مورد توجه زیادی قرار گرفته و موضوع پژوهش‌های گسترده‌ای در میان محققان گشته است. در این گزارش ما در ابتدا به بیان تعدادی از حوزه‌های کاری مهم مرتبط با داده‌کاوی که تحت عنوان کارکردهای[۱۷۳] داده‌کاوی مطرح می‌شود، خواهیم پرداخت. سپس از میان این کارکردها توضیحات بیشتری را در مورد دو کارکرد رده‌بندی[۱۷٤] و خوشه‌بندی[۱۷۵] بیان می‌کنیم. امید است در گزارش بعدی، با نگاهی عمیق‌تر تمرکز اصلی خود را به سمت یکی از این دو کارکرد معطوف کنیم.

## ۱۱–۲–کارکردهای داده کاوی

کارکردهای داده‌کاوی، انواع الگوهایی را که در عملیات داده‌کاوی باید یافت شوند، مشخص می‌کنند. به طور کلی می‌توان عملیات داده‌کاوی را در دو گروه دسته‌بندی کرد: *توصیفی* و *پیشگویانه*. عملیات کاوش توصیفی، ویژگی‌های عمومی داده‌های موجود در پایگاه داده را توصیف می‌کند و عملیات کاوش پیشگویانه، با انجام استنتاج بر روی داده‌های موجود به پیشگویی می‌پردازد.

---

[۱۷۳] functionality

[۱۷٤] classification

[۱۷۵] clustering

در ادامه به بیان کارکردهای مختلف داده‌کاوی و انواع مختلف الگوهایی که می‌توانند استخراج کنند، می‌پردازیم.

## ۱۱-۳- توصیف مفهوم یا رده: توصیف ویژگی ها و بیان وجوه تمایز

داده‌ها را می‌توان با رده‌ها یا مفاهیمی مرتبط دانست. توصیف رده‌ها و مفاهیم خاص در قالب واژه‌های مختصر و مفید و در عین حال دقیق می‌تواند مفید باشد. چنین توصیفاتی را از دو طریق می‌توان به دست آورد؛ یکی توصیف ویژگی‌های داده‌ها که با خلاصه کردن داده‌های مربوط به رده-های مورد بررسی در قالب واژگان عمومی قابل انجام است و دیگری بیان وجوه تمایز، که با مقایسه‌ی رده‌ی مورد بررسی با یک یا چند رده‌ی دیگر صورت می‌گیرد. همچنین می‌توان از هر دو روش نیز با هم بهره گرفت.

## ۱۱-۴- کاوش الگوهای پربسامد، وابستگی ها[۱۷۶] و همبستگی ها[۱۷۷]

الگوهای پربسامد، الگوهایی هستند که به میزان زیادی در میان داده‌ها وجود دارند. انواع مختلفی از چنین الگوهایی وجود دارد که از آن جمله می‌توان به مجموعه اقلام، زیردنباله‌ها و زیرساختارها اشاره کرد. یک *مجموعه از اقلام پربسامد* به مجموعه اقلامی گفته می‌شود که غالباً در مجموعه‌ای از داده‌ها با هم ظاهر می‌شوند؛ مثل شیر و نان. یک *زیردنباله‌ی پربسامد*، مثلاً این الگو که مشتریان معمولاً ابتدا یک کامپیوتر شخصی سپس یک دوربین دیجیتال و پس از آن یک کارت حافظه را می-خرند، در واقع یک الگوی ترتیبی است که زیاد تکرار می‌شود. منظور از *زیرساختار*، شکل‌های ساختاری گوناگونی همچون گراف‌ها، درخت‌ها یا لاتیس‌هاست که ممکن است به صورت ترکیبی با مجموعه اقلام یا زیردنباله‌ها به کار روند. کاوش الگوهای پربسامد منجر به کشف وابستگی‌ها و همبستگی‌های جالبی میان داده‌ها می‌شود.

## ۱۱-۵- رده بندی و پیشگویی[۱۷۸]

رده‌بندی عبارتست از فرآیند یافتن یک مدل (یا تابع) جهت توصیف و تمیز رده‌ها یا مفاهیم، با این هدف که به کمک این مدل بتوان به پیشگویی رده‌ی مربوط به اشیائی پرداخت که برچسب رده-

[۱۷۶] associations
[۱۷۷] correlations
[۱۷۸] prediction

بندی آن‌ها نامشخص است. مدل بدست آمده بر اساس تحلیل مجموعه‌ای از داده‌های آموزش‌دهنده (یعنی اشیاء داده‌ای که برچسب رده‌بندی آن‌ها مشخص است) می‌باشد. جهت نمایش مدل استخراج شده، شکل‌های گوناگونی را می‌توان مورد توجه قرار داد که از آن جمله قوانین رده‌بندی (IF-THEN)، درخت‌های تصمیم، فرمول‌های ریاضی یا شبکه‌های عصبی را می‌توان ذکر کرد. درخت تصمیم یک ساختار درختی شبیه نمودار گردش کار[۱۷۹] است که برگ‌های آن نشان دهنده‌ی رده‌ها یا توزیع‌های رده‌ای می‌باشند و با شروع از ریشه‌ی درخت و عبور از گره‌های میانی، با رسیدن به برگ‌ها تخمینی از رده‌ی مورد جستجو به دست می‌آید. درخت‌های تصمیم را می‌توان به راحتی به قوانین رده‌بندی تبدیل کرد. یک شبکه‌ی عصبی، هنگامی که برای رده‌بندی استفاده می‌شود، نوعاً مجموعه‌ای است از واحدهای پردازشی شبیه به سلول‌های عصبی و دارای اتصالات وزن‌دار بین واحدها. روش‌های بسیار دیگری نیز برای ساختن مدل‌های رده‌بندی وجود دارد، مانند رده‌بندی بیزی ساده[۱۸۰]، ماشین‌های برداری پشتیبانی و رده‌بندی از نوع k تا نزدیک‌ترین همسایه.

در حالیکه در عملیات رده‌بندی، برچسب‌های (گسسته و غیرترتیبی) مربوط به رده‌ها پیش‌بینی می‌شود، در روش‌های پیشگویانه، توابع دارای مقادیر پیوسته مدل می‌شوند؛ یعنی در اینجا به جای برچسب رده‌ها، این مقادیر عددی داده‌هاست که پیش‌بینی می‌شود.

تحلیل پسرفت[۱۸۱] نوعی روش آماری است که اغلب برای پیشگویی عددی مورد استفاده قرار می‌گیرد، هرچند سایر روش‌ها نیز همچنان موجودند. در روش‌های پیشگویانه همچنین شناسایی روندهای توزیعی بر اساس داده‌های موجود نیز انجام می‌شود.

ممکن است قبل از رده‌بندی و پیشگویی نیاز به تحلیل ارتباط[۱۸۲] وجود داشته باشد، که به منظور شناسایی صفاتی که در فرآیند رده‌بندی یا پیشگویی نقشی ندارند، انجام می‌شود. آنگاه می‌توان چنین صفاتی را کنار گذاشت.

## ۱۱-۶- تحلیل خوشه

بر خلاف رده‌بندی و پیشگویی که به تحلیل اشیاء داده‌ای برچسب خورده می‌پرداختند، در مبحث خوشه‌بندی، اشیاء داده‌ای بدون در نظر گرفتن برچسب‌های شناخته شده مورد تحلیل قرار می‌گیرند.

---

[۱۷۹] flowchart
[۱۸۰] naïve Bayesian classification
[۱۸۱] regression analysis
[۱۸۲] relevance analysis

در حالت کلی برچسب‌های مشخص‌کننده‌ی رده‌ها در داده‌های آموزش‌دهنده وجود ندارد، زیرا چنین برچسب‌هایی هنوز شناخته نشده‌اند. هدف از خوشه‌بندی نیز اساساً تولید چنین برچسب‌هایی است. در اینجا جهت خوشه‌بندی اشیاء، برمبنای قواعد *بیشینه‌سازی شباهت‌های داخل رده‌ها* و *کمینه‌سازی شباهت‌های بین رده‌ها* انجام می‌شود؛ یعنی خوشه‌های اشیاء به گونه‌ای تشکیل می‌شوند که اشیاء داخل یک خوشه در مقایسه با خوشه‌ی دیگر دارای شباهت زیاد به یکدیگر باشند و در عین حال بسیار بی‌شباهت به اشیاء داخل سایر خوشه‌ها. هر خوشه‌ای را که ساخته می‌شود، می‌توان به عنوان رده‌ای از اشیاء مورد توجه قرار داد، که آنگاه یکسری قوانین را می‌توان از روی آن بدست آورد. خوشه‌بندی همچنین می‌تواند آرایش طبقه‌بندی[۱۸۳] را - یعنی سازماندهی مشاهدات در قالب یک ساختار سلسله‌مراتبی که رخدادهای مشابه را در کنار هم گروه‌بندی می‌کند - نیز تسهیل کند.

## ۱۱-۷- تحلیل اقلام نامربوط[۱۸٤]

یک پایگاه داده ممکن است حاوی اشیاء داده‌ای باشد که هماهنگ با رفتار یا مدل عمومی داده‌ها نباشند. این اشیاء داده‌ای به عنوان اقلام نامربوط شناخته می‌شوند. بسیاری از روش‌های داده‌کاوی این اقلام را به عنوان پارازیت یا استثنا در نظر گرفته و آنها را کنار می‌گذارند. با این وجود، در بعضی از کاربردها مثل تشخیص کلاه‌برداری، رخدادهای نادر ممکن است بیش از مواردی که معمولاً اتفاق می‌افتند، جالب توجه باشند. تحلیل داده‌های نامربوط در مباحث کاوش اقلام نامربوط (outlier mining) مورد بحث قرار می‌گیرد.

برای تشخیص اقلام نامربوط می‌توان از آزمون‌های آماری استفاده کرد که یک توزیع یا مدل احتمالی را برای داده‌ها در نظر می‌گیرد. همچنین می‌توان معیارهای فاصله‌ای را مورد توجه قرار داد و بر اساس آن اشیائی را که دارای فاصله‌ی قابل توجهی از همه‌ی خوشه‌ها هستند، به عنوان اقلام نامربوط در نظر گرفت. علاوه بر روش‌های آماری و فاصله‌ای، روش‌های مبتنی بر انحراف (deviation) نیز در اینجا قابل استفاده هستند که شناسایی اقلام نامربوط در آنها با بررسی تفاوت‌های موجود در ویژگی‌های اصلی اشیاء داخل یک گروه انجام می‌شود.

[۱۸۳] taxonomy formation
[۱۸٤] outlier

## ۱۱-۸- تحلیل تکامل تدریجی[۱۸۵]

تحلیل تکامل تدریجی داده‌ها به توصیف و مدل‌سازی قاعده‌ها یا روندها در مورد اشیائی می-پردازد که رفتار آنها در طول زمان تغییر می‌کند. هرچند این کار ممکن است به کمک توصیف ویژگی‌ها، بیان وجوه تمایز، تحلیل وابستگی و همبستگی، رده‌بندی، پیشگویی یا خوشه‌بندی داده-های مرتبط با زمان انجام شود، اما روش‌های خاصی نیز برای این تحلیل وجود دارد که شامل تحلیل داده‌ای سری‌های زمانی، تطبیق الگوهای دنباله‌ای یا تناوبی و تحلیل داده‌ای مبتنی بر شباهت می‌شود.

## ۱۱-۹- رده بندی

همچنان که پیشتر گفته شد رده‌بندی عبارتست از فرآیند یافتن یک مدل (یا تابع) جهت توصیف و تمیز رده‌ها یا مفاهیم، با این هدف که به کمک این مدل بتوان به پیشگویی رده‌ی مربوط به اشیائی پرداخت که برچسب رده‌بندی آن‌ها نامشخص است. رده‌بندی دارای کاربردهای گوناگونی هم در تجارت و هم در تحقیقات می‌باشد. نمونه‌هایی از این کاربردها عبارتند از: تشخیص اینکه یک تراکنش کارت اعتباری، معتبر است یا تقلبی؛ تعیین اینکه یک بیماری خاص وجود دارد یا نه؛ تخمین زدن اینکه یک عملیات خرید خانه از نظر خطرپذیری مناسب است یا خیر؛ تشخیص ایمیل-های اسپم در سیستم‌های مدیریت پست الکترونیک.

به طور کلی رده‌بندی داده‌ها طی دو مرحله انجام می‌شود. در مرحله‌ی اول یک رده‌بند ساخته می‌شود که مجموعه‌ی از پیش تعیین شده‌ای از رده‌ها یا مفاهیم داده‌ای را توصیف می‌کند. این مرحله، مرحله‌ی آموزش یا یادگیری نامیده می‌شود و طی آن یک الگوریتم رده‌بندی با تحلیل و یادگیری از مجموعه‌ی آموزش‌دهنده که شامل یکسری تاپل و برچسب رده‌بندی مربوط به آنهاست، رده‌بند را می‌سازد. هر تاپل توسط یک بردار ویژگی‌های n بعدی نمایش داده می‌شود. به هر تاپل یک برچسب رده‌بندی نسبت داده می‌شود که دارای مقداری گسسته است.

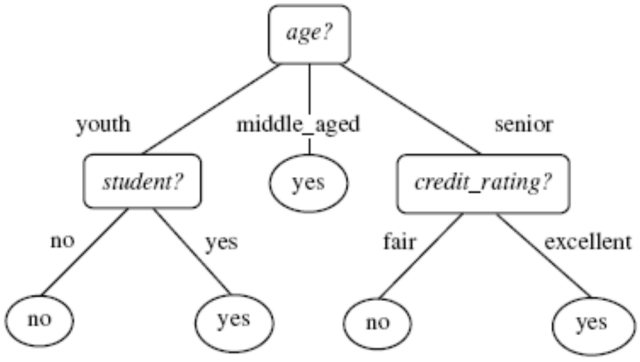
در مرحله‌ی دوم، از اطلاعات مرحله‌ی اول برای رده‌بندی استفاده می‌شود. برای ارزیابی درستی یک رده‌بند، یک مجموعه‌ی آزمون به آن داده می‌شود که رده‌بند باید با اجرای الگوریتم رده‌بندی مربوطه، برچسب‌هایی را به تاپل‌های این مجموعه نسبت دهد. اما برای ارزیابی، برچسب رده‌بندی واقعی تاپل‌های این مجموعه از قبل مشخص می‌باشد. درستی یک رده‌بند بر روی یک مجموعه‌ی آزمون داده شده عبارت است از درصدی از تاپل‌های مجموعه‌ی آزمون که به درستی توسط رده‌بند،

[۱۸۵] evolution

دسته‌بندی شده‌اند. به منظور انجام ارزیابی روش‌ها و معیارهای گوناگونی وجود دارد که در جای خود باید مورد بحث و بررسی قرار گیرند. در ادامه ما به عنوان نمونه به معرفی اجمالی دو روش رده‌بندی می‌پردازیم: یکی رده‌بندی به کمک استنتاج با درخت تصمیم و دیگری رده‌بندی بیزی[۱۸۶].

## ۱۱-۱۰- استنتاج با درخت تصمیم

استنتاج با درخت تصمیم عبارت است از یادگیری درخت تصمیم از تاپل‌های آموزش‌دهنده‌ی رده‌بندی شده. درخت تصمیم یک ساختار درختی شبیه نمودار گردش کار است که هر گره آن معرف یک آزمون بر روی یک مقدار صفت است، هر شاخه نمایانگر یک برآمد از آزمون و برگ‌های درخت نشان دهنده‌ی رده‌ها یا توزیع‌های رده‌ای می‌باشند. یک نمونه از درخت تصمیم در شکل ۱ نشان داده شده است. این درخت برای خرید کامپیوتر ترسیم شده است و پیش‌بینی می‌کند که که آیا محتمل است که یک مشتری یک کامپیوتر را بخرد یا نه. بعضی از الگوریتم‌های درخت تصمیم تنها درخت‌های دودویی تولید می‌کنند، اما ممکن است یک الگوریتم درخت‌های غیردودیی هم تولید کند.




شکل ۱ – یک درخت تصمیم برای مفهوم خرید کامپیوتر

یک سؤال که در اینجا مطرح می‌شود این است که چگونه می‌توان از درخت‌های تصمیم برای رده‌بندی استفاده کرد. فرض می‌کنیم که می‌خواهیم برچسب رده‌بندی یک تاپل داده شده را تعیین کنیم. بدین منظور مقادیر صفات آن تاپل در درخت تصمیم مورد آزمون قرار می‌گیرند؛ مسیری از

[۱۸۶] Bayesian

ریشه‌ی درخت تا یک گره برگ دنبال می‌شود و بدین ترتیب پیش‌بینی رده‌ی آن تاپل انجام می‌شود. درخت‌های تصمیم را می‌توان به سادگی به قوانین رده‌بندی تبدیل کرد.

## ۱۱-۱۱- رده بندی بیزی

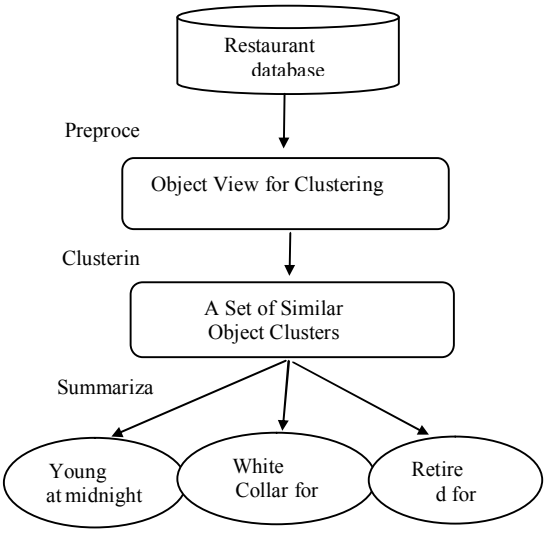
رده‌بندهای بیزی نوعی از رده‌بندهای آماری هستند. در این روش احتمالات مربوط به عضویت در رده‌های مختلف تخمین زده می‌شود؛ مثلاً احتمال اینکه یک تاپل داده شده به یک رده‌ی خاص تعلق داشته باشد. رده‌بندی بیزی برمبنای قضیه‌ی بیز در ریاضیات می‌باشد. یعنی به کمک رابطه‌ی بیز در مورد احتمال شرطی، احتمال اینکه یک تاپل به یک رده‌ی خاص تعلق داشته باشد، محاسبه می‌شود. شکل ساده‌ای از روش بیزی تحت عنوان رده‌بند بیزی ساده[۱۸۷] وجود دارد که از نظر کارآیی قابل مقایسه با رده‌بندهای مبتنی بر درخت تصمیم است. در این نوع از رده‌بندهای بیزی چنین فرض می‌شود که تأثیر مقداری از یک صفت بر روی یک رده‌ی داده شده مستقل از مقادیر سایر صفات می‌باشد. هدف از در نظر گرفتن این شرط ساده‌سازی محاسبات روش بیزی است.

## ۱۱-۱۲- خوشه بندی

خوشه‌بندی از تکنیک‌های محبوب و مورد علاقه در داده‌کاوی می‌باشد که هدف از آن قرار دادن داده‌ها در گروه‌های مجزاست. تمرکز ما در این بخش بیشتر در زمینه خوشه‌بندی کردن پایگاه داده می‌باشد. هدف از خوشه‌بندی پایگاه داده این است که ما بتوانیم اطلاعات یک پایگاه داده که انواع خاصی از اشیاء داده‌ای را نگهداری می‌کند، بگیریم و زیرگروه‌های این اشیا را مشخص کنیم به طوری که اشیاء متعلق به یک زیرگروه، بسیار مشابه هم باشند و اشیا متعلق به زیرگروه‌های متفاوت بسیار متفاوت از یکدیگر.

به طور مثال یک رستوران را در نظر می‌گیریم که دارای یک پایگاه داده شامل اطلاعات مشتریانش می‌باشد و به دنبال گروه بندی مشتریان خود به منظور درک بهتری از مشتریان اصلی خود برای اهداف تجاری می‌باشد، برای رسیدن به این هدف، همانطور که در شکل ۲ مشاهده می‌شود، قبل از خوشه‌بندی ابتدا پیش‌پردازش‌های لازم بر روی پایگاه داده‌ی رستوران صورت می‌گیرد و سپس الگوریتم خوشه‌بندی بر روی مجموعه داده‌ی پیش پردازش شده اجرا می‌شود.

[۱۸۷] naive




شکل ۲ ـ مثالی از خوشه‌بندی یک پایگاه داده

برای مثال الگوریتم ممکن است سه خوشه از مشتریان را در پایگاه داده آشکار کند، در نهایت مشخصات دانشی که از خلاصه هر خوشه تولید می‌شود به مالک رستوران می‌گوید که گروه‌های عمده مشتریانش شامل جوانانی هستند که نیمه شب می‌آیند، کارمندانی که برای شام می‌آیند، و بازنشستگانی که برای نهار می‌آیند. این دانش قطعاً برای اهداف بازاریابی و طراحی منوی غذایی مفید می‌باشد.

## ۱۱-۱۳- مراحل خوشه بندی پایگاه داده

به طور کلی، خوشه‌بندی پایگاه داده به عنوان فعالیتی در نظر گرفته می‌شود که از هفت مرحله زیر عبور می‌کند:

۱. تعریف نمای شیء[۱۸۸]
۲. انتخاب صفت‌های مرتبط[۱۸۹]
۳. تولید قالب ورودی مناسب برای ابزار خوشه‌بندی
٤. تعریف مقیاس شباهت[۱۹۰]
۵. انتخاب مجموعه پارامترها برای الگوریتم خوشه‌بندی بکارگرفته‌شده
٦. اجرای الگوریتم خوشه‌بندی

[۱۸۸] Object view
[۱۸۹] relevant attributes
[۱۹۰] similarity

۷. توصیف خوشه‌های محاسبه شده

سه مرحله‌ی اول پیشنهاد شده برای خوشه‌بندی در دسته‌ی پیش پردازنده‌ها قرار می‌گیرند و به منظور تولید مجموعه داده‌ای که قرار است الگوریتم خوشه‌بندی بر روی آن اجرا شود، مورد استفاده قرار می‌گیرند. در این مراحل، باید مشخص شود که چه اشیایی از پایگاه داده و کدام یک از خصوصیت‌هایشان به منظور اهداف خوشه‌بندی انتخاب شوند، علاوه براین اطلاعات مرتبط باید به قالبی که می‌تواند به وسیله ابزار یا ابزارهای خوشه‌بندی مورد استفاده قرار گیرد، تبدیل شود. در مرحله چهارم مقیاس و اندازه‌گیری‌های شباهت اشیایی که خوشه‌بندی می‌شوند باید تعیین شود. در نهایت، طی مراحل پنج تا هفت، الگوریتم خوشه‌بندی اجرا شده و خلاصه‌ی خوشه‌بندی به دست آمده تولید می‌شود.

## ۱۱-۱۴- تفاوت‌های بین خوشه بندی پایگاه داده و خوشه بندی معمولی

مجموعه‌های داده در قالب‌های متفاوت زیادی همچون فایل‌های مسطح[۱۹۱]، پایگاه‌های داده‌ی شیءگرا یا رابطه‌ای ذخیره می‌شوند. قالب فایل مسطح، ساده‌ترین و پر استفاده‌ترین قالب در قلمرو سنتی تحلیل داده می‌باشد. وقتی که از قالب فایل مسطح استفاده می‌کنیم، اشیاء داده به صورت بردارهایی از فضای n بعدی می‌باشند که هر بردار، یک شیء را توصیف می‌کند و هر شیء با n صفت مشخص می‌شود و هر صفت دارای یک مقدار می‌باشد. تقریباً همه‌ی تحلیل‌های داده و ابزارهای داده‌کاوی فرض می‌کنند که مجموعه داده‌هایی که باید تحلیل شوند، در قالب فایل‌های مسطح نمایش داده شده‌اند.

بر مبنای این واقعیت که پایگاه‌های داده خیلی پیچیده‌تر از فایل‌های مسطح هستند، خوشه‌بندی پایگاه داده با مسائل گسترده‌تری مواجه می‌شود که در خوشه‌بندی فایل‌های مسطح وجود ندارد. این مسائل شامل موارد زیر است:

- پایگاه‌های داده شامل اشیایی هستند که متعلق به انواع متفاوتی می‌باشند، بنابراین باید اشیایی که در یک پایگاه داده نیاز به خوشه‌بندی دارند، تعیین شوند.
- پایگاه‌های داده شامل ارتباطات بین شیئی ۱:۱، ۱:n، و n:m می‌باشند که انواع اشیاء مرتبط می‌تواند مشابه یا متفاوت باشد.

[۱۹۱] flat files

- تعریف شباهت اشیاء به دلیل وجود گستره‌ی وسیع مقادیر یا اطلاعـات مرتبطـی کـه یـک شـیء را توصیف می‌کند، بسیار پیچیده‌تر است.
- صفات اشیاء از انواع متفاوتی هستند که این خود انتخاب معیار همسانی را مشکل‌تر می‌کند.

تاکنون الگوریتم‌های خوشه‌بندی مختلفی مانند CLARAN، DBSCAN، BIRCH، و STING برای پایگاه‌های داده بزرگ ارائه شده است، اما اهداف بیشتر این الگوریتم‌ها پایگاه‌های داده‌ی فضایی[۱۹۲] می‌باشد. علاوه براین مانند تعداد زیادی از الگوریتم‌های مرسوم خوشه‌بندی فرض این الگوریتم‌ها نیز برمبنای یک تاپل، یک شیء[۱۹۳] می‌باشد. بکار بردن مستقیم الگوریتم‌های سنتی[۱۹٤] خوشه‌بندی در پایگاه‌های داده، به دلیل اینکه پایگاه‌های داده معمولا دارای ساختار منظم و مرتبط می‌باشند راه مناسبی نیست، بنابراین یکی از موضوعات پژوهشی این است که چطور می‌توان الگوریتم‌های خوشه‌بندی را مستقیما در پایگاه‌های داده حقیقی بکار برد.

[۱۹۲] spatial
[۱۹۳] one-tuple one-object
[۱۹٤] traditional

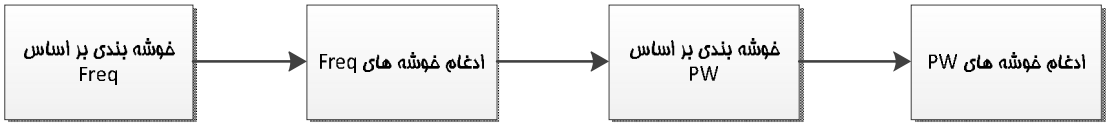
# فصل ۱۲– الگوریتم خوشه بندی

## ۱۲–۱– مقدمه

در این فصل سعی بر توضیح کامل الگوریتم خوشه بندی به منظور تشخیص اجسام داده شده است. ابتدا کلیات این الگوریتم بیان شده است و در ادامه الگوریتم به همراه کد آن با توضیحات بیشتر آورده شده است.

در کلیه قسمت های الگوریتم منظور از اطلاعات ورودی، اطلاعاتی می باشد که از جهت‌یاب راداری دریافت می شود.

## ۱۲–۲– کلیات الگوریتم

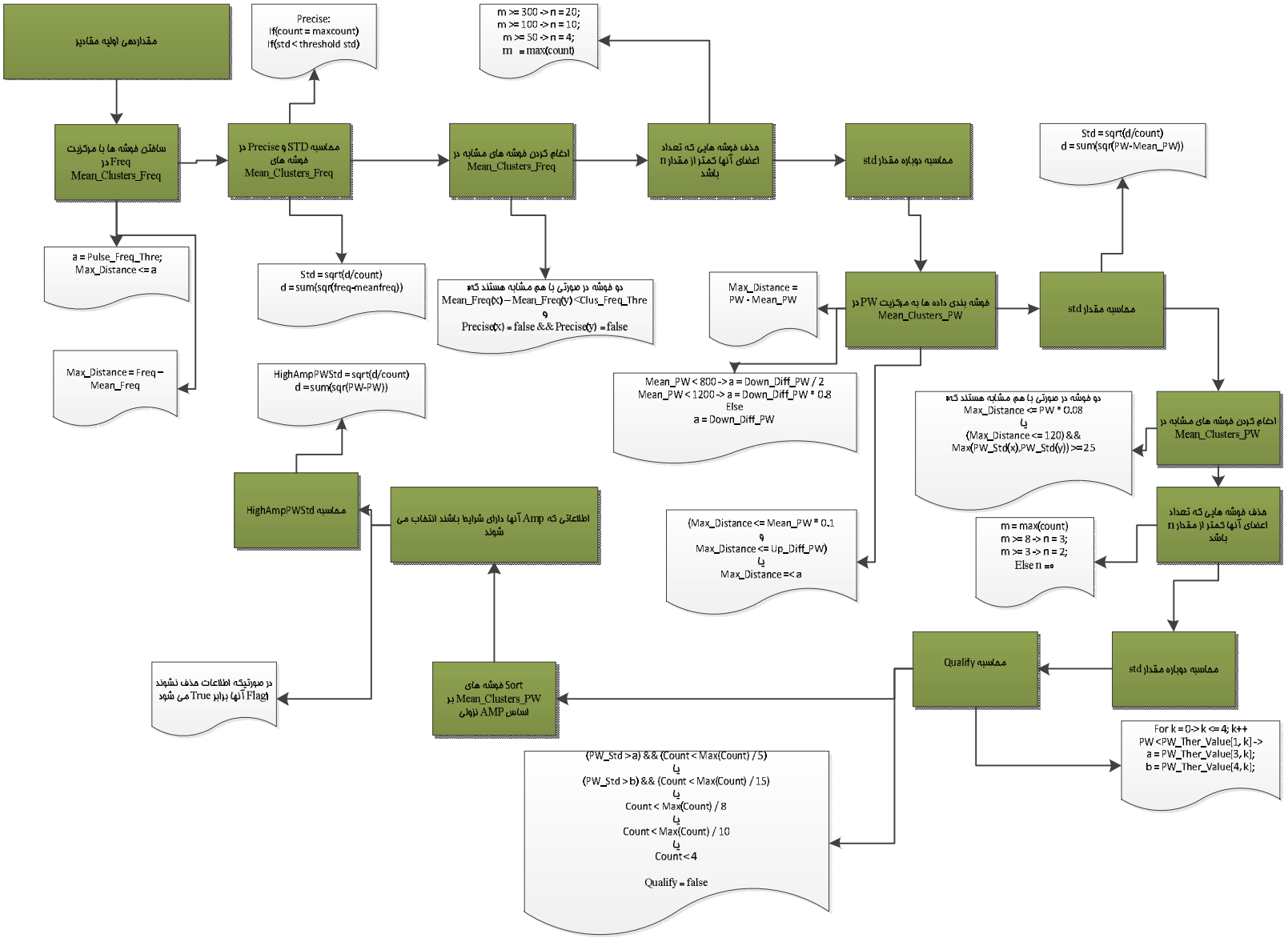



همانطور که در شکل بالا نمایش داده شده است ابتدا اطلاعات ورودی بر اساس freq خوشه بندی می شوند. برای این منظور مرکز هر خوشه میانگین Freq می باشد.

در مرحله بعد، به علت آنکه خوشه های تولید شده ممکن است پراکندگی داده کم یا زیاد داشته باشند، می بایست سعی بر ادغام خوشه ها کنیم.

پس از ادغام خوشه های Freq، حال با استفاده از PW داده ها را خوشه بندی می کنیم. برای این منظور مرکز هر خوشه میانگین PW می باشد.

همانند ادغام خوشه های Freq می بایست خوشه های جدید را نیز با هم ادغام نماییم تا هم اطلاعات بسیار پراکنده حذف و هم اطلاعاتی که بسیار به هم شبیه می باشند با هم ادغام گردند.


مقداردهی اولیه مقادیر
Precise:
If(count = maxcount)
If(std < threshold std)
m >= 300 -> n = 20;
m >= 100 -> n = 10;
m >= 50 -> n = 4;
m = max(count)
ساختن خوشه ها با مرکزیت در Freq
Mean_Clusters_Freq
محاسبه Precise و STD در خوشه های
Mean_Clusters_Freq
ادغام کردن خوشه های مشابه در
Mean_Clusters_Freq
حذف خوشه هایی که تعداد اعضای آنها کمتر از مقدار n باشد
محاسبه دوباره مقدار std
Std = sqrt(d/count)
d = sum(sqr(PW-Mean_PW))
a = Pulse_Freq_Thre;
Max_Distance <= a
Std = sqrt(d/count)
d = sum(sqr(freq-meanfreq))
دو خوشه در صورتی با هم مشابه هستند که
Mean_Freq(x) – Mean_Freq(y) <Clus_Freq_Thre
و
Precise(x) = false && Precise(y) = false
Max_Distance =
PW - Mean_PW
خوشه بندی داده ها به مرکزیت PW در
Mean_Clusters_PW
محاسبه مقدار std
Max_Distance = Freq – Mean_Freq
HighAmpPWStd = sqrt(d/count)
d = sum(sqr(PW-PW))
Mean_PW < 800 -> a = Down_Diff_PW / 2
Mean_PW < 1200 -> a = Down_Diff_PW * 0.8
Else
a = Down_Diff_PW
دو خوشه در صورتی با هم مشابه هستند که
Max_Distance <= PW * 0.08
یا
(Max_Distance <= 120) &&
Max(PW_Std(x),PW_Std(y)) >=2.5
ادغام کردن خوشه های مشابه در
Mean_Clusters_PW
HighAmpPWStd محاسبه
اطلاعاتی که Amp آنها دارای شرایط باشند انتخاب می شوند
(Max_Distance <= Mean_PW * 0.1
و
Max_Distance <= Up_Diff_PW)
یا
Max_Distance =< a
m = max(count)
m >= 8 -> n = 3;
m >= 3 -> n = 2;
Else n =0
حذف خوشه هایی که تعداد اعضای آنها کمتر از مقدار n باشد
در صورتیکه اطلاعات حذف نشوند Flag آنها برابر True می شود
Sort خوشه های
Mean_Clusters_PW بر اساس AMP نزولی
Qualify محاسبه
محاسبه دوباره مقدار std
For k = 0-> k <= 4; k++
PW <PW_Ther_Value[1, k] ->
a = PW_Ther_Value[3, k];
b = PW_Ther_Value[4, k];
(PW_Std > a) && (Count < Max(Count) / 5)
یا
(PW_Std > b) && (Count < Max(Count) / 15)
یا
Count < Max(Count) / 8
یا
Count < Max(Count) / 10
یا
Count < 4
Qualify = false

## ۱۲-۳- تشریح الگوریتم

همانطور که در شکل بالا دیده می‌شود الگوریتم به طور کاملتری نمایش داده شده است. حال به توضیح شکل بالا می پردازیم:

### ۱۲-۳-۱- مقدار دهی اولیه مقادیر

در این قسمت که در کد متدی به نام InitializeParam می باشد مقادیر ثابتی که در کل برنامه استفاده می شود مقدار دهی می شود از آن جمله می توان به موارد زیر اشاره کرد:

- PW thresholds
- Pulse Frequency Threshold

```
private static void InitializeParam()
{
    //Processing = new ProcessingCls();
    ProcessingCls.Clustering_Limits = new Clustering_LimitsCls();
    ProcessingCls.Merging_Limits = new Merging_LimitsCls();
    ProcessingCls.Scan_Params = new Scan_ParamsCls();
    ClusterCounter_Freq_P = 0;
    ClusterCounter_Freq_CW = 0;
    ClusterCounter_P = 0;
    ClusterCounter_CW = 0;
    TotalClusters_P = null;
    TotalClusters_CW = null;
    PW_Margin = null;
    Scan_Data = null;
    MaxAOADiff = 5;
    ClusteringParams = new ClusteringParamsCls
                        {
                            Min_PW = 100,
                            Max_PW = (float) (800*1e3),
                            Up_Diff_PW = 2000,
                            Down_Diff_PW = 150,
                            PW_Defult = 1,
                            Num_PW_Clus_Step = 5
                        };
    #region ClusteringParams
    ClusteringParams.PW_Ther_Value[0, 0] = 100;
    ClusteringParams.PW_Ther_Value[0, 1] = 1000;
    ClusteringParams.PW_Ther_Value[0, 2] = 5000;
    ClusteringParams.PW_Ther_Value[0, 3] = 30000;
    ClusteringParams.PW_Ther_Value[0, 4] = 200000;
    ClusteringParams.PW_Ther_Value[1, 0] = 1000;
    ClusteringParams.PW_Ther_Value[1, 1] = 5000;
    ClusteringParams.PW_Ther_Value[1, 2] = 30000;
    ClusteringParams.PW_Ther_Value[1, 3] = 200000;
    ClusteringParams.PW_Ther_Value[1, 4] = 800000;
    ClusteringParams.PW_Ther_Value[2, 0] = 100;
    ClusteringParams.PW_Ther_Value[2, 1] = 200;
    ClusteringParams.PW_Ther_Value[2, 2] = 500;
    ClusteringParams.PW_Ther_Value[2, 3] = 1000;
    ClusteringParams.PW_Ther_Value[2, 4] = 2000;
    ClusteringParams.PW_Ther_Value[3, 0] = 100;
    ClusteringParams.PW_Ther_Value[3, 1] = 200;
    ClusteringParams.PW_Ther_Value[3, 2] = 500;
```

```
            ClusteringParams.PW_Ther_Value[3, 3] = 1200;
            ClusteringParams.PW_Ther_Value[3, 4] = 3000;
            ClusteringParams.PW_Ther_Value[4, 0] = 50;
            ClusteringParams.PW_Ther_Value[4, 1] = 120;
            ClusteringParams.PW_Ther_Value[4, 2] = 400;
            ClusteringParams.PW_Ther_Value[4, 3] = 1000;
            ClusteringParams.PW_Ther_Value[4, 4] = 2000;
            ClusteringParams.Pulse_Freq_Thre = 1;
            ClusteringParams.Clus_Freq_Thre = 3;
            ClusteringParams.Clus_Freq_Std = 0.2f;
            ClusteringParams.Freq_Defult = 1;
            ClusteringParams.Freq_Ther_Value[0, 0] = 2950;
            ClusteringParams.Freq_Ther_Value[0, 1] = 3070;
            ClusteringParams.Freq_Ther_Value[0, 2] = 9350;
            ClusteringParams.Freq_Ther_Value[0, 3] = 9500;
            ClusteringParams.Freq_Ther_Value[0, 4] = 41000;
            ClusteringParams.Freq_Ther_Value[1, 0] = 3;
            ClusteringParams.Freq_Ther_Value[1, 1] = 1;
            ClusteringParams.Freq_Ther_Value[1, 2] = 3;
            ClusteringParams.Freq_Ther_Value[1, 3] = 1;
            ClusteringParams.Freq_Ther_Value[1, 4] = 3;
            #endregion
            SCT = new Dictionary<int, byte>(7);

            for (i = -1; i <= 5; i++)
            {
                SCT.Add(i, 0);
            }

            MaxAOADiff = 5;
            TScanType = new TScanType {TOA = new double[65535]};
            ProcessingCls.Merging_Limits.Freq = 3;
            ProcessingCls.Merging_Limits.PW_Min = 300;
            ProcessingCls.Merging_Limits.PW_Max = 3000;
            ProcessingCls.Merging_Limits.PW_Per_Max = 5;
            ProcessingCls.Merging_Limits.PRF = 10;
            ProcessingCls.Merging_Limits.AOA = 5;
            SetLength(ref ScanPeriodsClus, 10);
        }
```

## ۱۲-۴- ساختن خوشه ها با مرکزیت Freq در آرایه Mean_Clusters_Freq

در این قسمت از الگوریتم که در متد GenerateMeanClustersFreqArrays می باشد، اطلاعات بر اساس Freq خوشه بندی شده و در Mean_Clusters_Freq اطلاعات ذخیره می شود.

در واقع Mean_Clusters_Freq لیستی می باشد که هر رکورد از این لیست دارای اطلاعات زیر می باشد

- Mean_Freq: این رکرود در واقع مرکز خوشه می باشد و مقدار میانگین Freq در آن قرار می گیرد. برای محاسبه Mean_Freq از فرمول A استفاده می شود

A) $if\ count = ۰\ then\ Mean_{Freq}: Freq$

$$else\ Mean_{Freq} = \frac{Mean_{Freq} * Count + Freq}{Count + ۱} - ۵$$

- Count: تعداد رکورد هایی که در این خوشه قرار گرفته اند را نشان می دهد. به عبـارتی اگر رکرودی در این خوشه اضافه شود، می بایست مقدار Count اضافه شود.
- Std: مقدار Std هر خوشه در این متغیر ذخیره می شود. در ادامه توضیح داده می شود که این مقدار چگونه محاسبه می شود.
- Precise: نشان دهنده آن است که این خوشه قوی می باشد یا خیر. در ادامه توضیح داده می شود که این مقدار چگونه محاسبه می شود.

```
        private static void GenerateMeanClustersFreqArrays(int First_Index, long
PDW_Count)
        {
            for (i = First_Index; i < First_Index + PDW_Count; i++)
            {
                #region !PDWElements[i].CW
                if (!Online_PDWProcess.PDWElements[i].CW)              //Pulse
                {
                    if ((Online_PDWProcess.PDWElements[i].PW <
ClusteringParams.Min_PW) || (Online_PDWProcess.PDWElements[i].PW >
ClusteringParams.Max_PW))
                    {
                        Cluster_Members_P[i - First_Index] = -1;
                        Cluster_Members_CW[i - First_Index] = -1;
                        continue;
                    }
                    found = false;
                    if (ClusterCounter_Freq_P != 0)
                    {
                        for (j = 0; j < ClusterCounter_Freq_P; j++)
                        {
                            Max_Distance =
Math.Abs(Online_PDWProcess.PDWElements[i].Freq - Mean_Clusters_Freq_P[j].Mean_Freq);

                            if (ClusteringParams.Freq_Defult == 0)
                            {
                                for (k = 0; k <= 4; k++)
                                {

                                    if (Mean_Clusters_Freq_P[j].Mean_Freq <
ClusteringParams.Freq_Ther_Value[0, k])
                                    {
                                        a = ClusteringParams.Freq_Ther_Value[1, k];
                                        break;
                                    }
                                }
                            }
                            else
                            {
                                a = ClusteringParams.Pulse_Freq_Thre;
                            }

                            if (Max_Distance <= a)
                            {
                                Mean_Clusters_Freq_P[j].Mean_Freq =
(Mean_Clusters_Freq_P[j].Mean_Freq * Mean_Clusters_Freq_P[j].Count +
Online_PDWProcess.PDWElements[i].Freq) / (Mean_Clusters_Freq_P[j].Count + 1);
```

```
                                Mean_Clusters_Freq_P[j].Count =
Mean_Clusters_Freq_P[j].Count + 1;
                                Cluster_Members_P[i - First_Index] = j;
                                Cluster_Members_CW[i - First_Index] = -1;
                                found = true;
                                break;
                            }
                        }
                    }
                    if (!found)
                    {
                        SetLength(ref Mean_Clusters_Freq_P, ClusterCounter_Freq_P +
1);
                        Mean_Clusters_Freq_P[ClusterCounter_Freq_P].Mean_Freq =
Online_PDWProcess.PDWElements[i].Freq;
                        Mean_Clusters_Freq_P[ClusterCounter_Freq_P].Count = 1;
                        Mean_Clusters_Freq_P[ClusterCounter_Freq_P].Last_Index = -1;
                        Cluster_Members_P[i - First_Index] = ClusterCounter_Freq_P;
                        Cluster_Members_CW[i - First_Index] = -1;
                        ClusterCounter_Freq_P++;
                    }
                }
                #endregion
                #region PDWElements[i].CW
                else                                                    //CW
                {
                    found = false;
                    if (ClusterCounter_Freq_CW != 0)
                    {
                        for (j = 0; j < ClusterCounter_Freq_CW; j++)
                        {
                            Max_Distance =
Math.Abs(Online_PDWProcess.PDWElements[i].Freq -
Mean_Clusters_Freq_CW[j].Mean_Freq);

                            if (ClusteringParams.Freq_Defult == 0)
                            {
                                for (k = 0; k <= 4; k++)
                                {
                                    if (Mean_Clusters_Freq_CW[j].Mean_Freq <
ClusteringParams.Freq_Ther_Value[0, k])
                                    {
                                        a = ClusteringParams.Freq_Ther_Value[1, k];
                                        break;
                                    }
                                }
                            }
                            else
                            {
                                a = ClusteringParams.Pulse_Freq_Thre;
                            }

                            if (Max_Distance <= a)
                            {
                                Mean_Clusters_Freq_CW[j].Mean_Freq =
(Mean_Clusters_Freq_CW[j].Mean_Freq * Mean_Clusters_Freq_CW[j].Count +
Online_PDWProcess.PDWElements[i].Freq) / (Mean_Clusters_Freq_CW[j].Count + 1);
                                Mean_Clusters_Freq_CW[j].Count =
Mean_Clusters_Freq_CW[j].Count + 1;
                                Cluster_Members_CW[i - First_Index] = j;
                                Cluster_Members_P[i - First_Index] = -1;
                                found = true;
                                break;
```

```
                        }
                    }
                }
                if (!found)
                {
                    SetLength(ref Mean_Clusters_Freq_CW, ClusterCounter_Freq_CW
+ 1);
                    Mean_Clusters_Freq_CW[ClusterCounter_Freq_CW].Mean_Freq =
Online_PDWProcess.PDWElements[i].Freq;
                    Mean_Clusters_Freq_CW[ClusterCounter_Freq_CW].Count = 1;
                    Mean_Clusters_Freq_CW[ClusterCounter_Freq_CW].Last_Index = -
1;
                    Cluster_Members_CW[i - First_Index] =
ClusterCounter_Freq_CW;
                    Cluster_Members_P[i - First_Index] = -1;
                    ClusterCounter_Freq_CW++;
                }
            }
            #endregion
        }
    }
```

## ۱۲–۵– محاسبه Std و Precise در خوشه های Mean_Clusters_Freq

پس از آنکه اطلاعات را بر اساس Freq خوشه بندی کردیم حال به محاسبه Std و Precise هر خوشه می پردازیم. محاسبه این دو متغیر در متدی به نام SetMeanClustersVariable آمده است.

### ۱۲–۵–۱– محاسبه Std

برای این منظور از فرمول زیر استفاده می شود

$$d = \sum (Freq - MeanFreq)^{۲}$$

$$Std = \sqrt{\frac{d}{Count}}$$

### ۱۲–۵–۲– محاسبه Precise

برای این منظور از فرمول زیر استفاده می شود

$$if(Std[i] < ClusFreqStd)\ \&\&\ if(Count[i] \geq ClusterCount) \rightarrow Precise = True$$

در فرمول بالا ClusFreqStd متغیر ثابت برنامه می باشد که برابر ۰.۰۲ می باشد و ClusterCount برابر تعداد خوشه هایی می باشد که با استفاده از مرکزیت Freq ایجاد شده است.

```
        private static void SetMeanClustersVariables(int First_Index, long
PDW_Count)
        {
            #region Set Mean_Clusters_Freq_P[i].First_Index
            m = 0;
```

```
            for (i = 0; i < ClusterCounter_Freq_P; i++)
            {
                Mean_Clusters_Freq_P[i].First_Index = m;
                m += Mean_Clusters_Freq_P[i].Count;
            }
            m = 0;
            for (i = 0; i < ClusterCounter_Freq_CW; i++)
            {
                Mean_Clusters_Freq_CW[i].First_Index = m;
                m += Mean_Clusters_Freq_CW[i].Count;
            }
            #endregion

            #region Fill Cluster_Members_Freq_P && set
Mean_Clusters_Freq_P.LastIndex
            if (ClusterCounter_Freq_P != 0)
                for (i = 0; i < PDW_Count; i++)
                {
                    j = Cluster_Members_P[i];
                    if (j == -1)
                        continue;
                    m = Mean_Clusters_Freq_P[j].Last_Index;
                    m++;
                    Cluster_Members_Freq_P[Mean_Clusters_Freq_P[j].First_Index + m]
= i;
                    Mean_Clusters_Freq_P[j].Last_Index = m;
                }
            if (ClusterCounter_Freq_CW != 0)
                for (i = 0; i < PDW_Count; i++)
                {
                    j = Cluster_Members_CW[i];
                    if (j == -1)
                        continue;
                    m = Mean_Clusters_Freq_CW[j].Last_Index;
                    m++;
                    Cluster_Members_Freq_CW[Mean_Clusters_Freq_CW[j].First_Index +
m] = i;
                    Mean_Clusters_Freq_CW[j].Last_Index = m;
                }
            #endregion

            #region Set m,n to Max(Mean_Clusters_Freq_P.Count && fill
Mean_Clusters_Freq_P.Std
            m = 0;
            for (i = 0; i < ClusterCounter_Freq_P; i++)
            {
                if (m < Mean_Clusters_Freq_P[i].Count)
                    m = Mean_Clusters_Freq_P[i].Count;
                Mean_Clusters_Freq_P[i].Precise = false;
                d1 = 0;
                for (j = Mean_Clusters_Freq_P[i].First_Index; j <=
Mean_Clusters_Freq_P[i].First_Index + Mean_Clusters_Freq_P[i].Last_Index; j++)
                {
                    d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[j] +
First_Index].Freq - Mean_Clusters_Freq_P[i].Mean_Freq);
                }
                Mean_Clusters_Freq_P[i].Std = Math.Sqrt(d1 /
Mean_Clusters_Freq_P[i].Count);
            }
            n = 0;
            for (i = 0; i < ClusterCounter_Freq_CW; i++)
            {
                if (n < Mean_Clusters_Freq_CW[i].Count)
```

```
                    n = Mean_Clusters_Freq_CW[i].Count;
                Mean_Clusters_Freq_CW[i].Precise = false;
                d1 = 0;
                for (j = Mean_Clusters_Freq_CW[i].First_Index; j <=
Mean_Clusters_Freq_CW[i].First_Index + Mean_Clusters_Freq_CW[i].Last_Index; j++)
                {
                    d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_Freq_CW[j] +
First_Index].Freq - Mean_Clusters_Freq_CW[i].Mean_Freq);
                }
                Mean_Clusters_Freq_CW[i].Std = Math.Sqrt(d1 /
Mean_Clusters_Freq_CW[i].Count);
            }

            #endregion

            #region Set Mean_Clusters_Freq_P[i].Precise
            for (i = 0; i < ClusterCounter_Freq_P; i++)
            {
                if ((Mean_Clusters_Freq_P[i].Std < ClusteringParams.Clus_Freq_Std)
&&
                    (Mean_Clusters_Freq_P[i].Count >= m / ClusterCounter_Freq_P))
                {
                    Mean_Clusters_Freq_P[i].Precise = true;
                }
            }
            for (i = 0; i < ClusterCounter_Freq_CW; i++)
            {
                if ((Mean_Clusters_Freq_CW[i].Std < ClusteringParams.Clus_Freq_Std)
&&
                    (Mean_Clusters_Freq_CW[i].Count >= n / ClusterCounter_Freq_CW))
                {
                    Mean_Clusters_Freq_CW[i].Precise = true;
                }
            }
            #endregion
        }
```

## ۱۲–۶– ادغام کردن خوشه های مشابه در Mean_Clusters_Freq

همانطور که قبل از این به آن اشاره شد، به علت آنکه امکان دارد پراکندگی داده ها به طوری باشد که خوشه ها بسیار نزدیک به هم ایجاد شده باشند، می بایست این خوشه ها را با هم ادغام کرد. این امر در متدی به نام MerginClusters در برنامه انجام می شود.

برای ادغام دو خوشه ابتدا باید بررسی نماییم که دو خوشه Precise برابر True نداشته باشند سپس در صورتیکه مرکز این دو خوشه از یک threshold کمتر بود می توان این دو خوشه را با هم ادغام کرد. در برنامه متغیر ثابتی با نام Clus_Freq_Thre می باشد که برابر ۳ است. در صورتی که فرمول زیر مقدار true برگرداند دو خوشه x و y با هم ادغام می شوند.

$$|MeanFreq(x) - MeanFreq(y)| \leq ClusFreqThre$$

در صورتیکه دو خوشه x و y با هم ادغام شدند می بایست دو متغیر count و mean_Freq این دو خوشه نیز به روشی با هم ادغام شوند. برای این منظور ابتدا رکوردهای خوشه y را به خوشه x منتقل می کنیم، سپس با استفاده از فرمول زیر مقدار count و mean_freq خوشه x را دوباره محاسبه می کنیم.

$$MeanFreq(x) = \frac{\big(MeanFreq(x) * Count(x)\big) + (MeanFreq(y) * Count(y))}{Count(x) + Count(y)}$$

$$Count(x) = Count(x) + Count(y)$$

پس از ادغام دو خوشه x , y ، خوشه y حذف می گردد

```
        private static void MergingClustersPW()
        {
            t = ClusterCounter_Freq_P;
            i = 0;
            while (i < ClusterCounter_Freq_P - 1)
            {
                if (Mean_Clusters_Freq_P[i].Count == 0)
                {
                    i++;
                    continue;
                }
                found = false;
                for (j = i + 1; j < ClusterCounter_Freq_P; j++)
                {
                    if ((!Mean_Clusters_Freq_P[i].Precise) &&
                        (!Mean_Clusters_Freq_P[j].Precise) &&
                       (Math.Abs(Mean_Clusters_Freq_P[i].Mean_Freq -
Mean_Clusters_Freq_P[j].Mean_Freq) <= ClusteringParams.Clus_Freq_Thre) &&
                       (Mean_Clusters_Freq_P[j].Count != 0))
                    {
                        for (k = 0; k < ClusterCounter_Freq_P; k++)
                        {
                            TempClusterIndexP[i, k] = TempClusterIndexP[i, k] ||
TempClusterIndexP[j, k];
                        }

                        k = Mean_Clusters_Freq_P[i].Count +
Mean_Clusters_Freq_P[j].Count;
                        Mean_Clusters_Freq_P[i].Mean_Freq =
(Mean_Clusters_Freq_P[i].Mean_Freq * Mean_Clusters_Freq_P[i].Count +
Mean_Clusters_Freq_P[j].Mean_Freq * Mean_Clusters_Freq_P[j].Count) / k;

                        Mean_Clusters_Freq_P[i].Count = k;
                        Mean_Clusters_Freq_P[j].Count = 0;
                        found = true;
                        break;
                    }
                }
                if (!found)
                    i++;
            }
        }
```

## ۱۲-۷- حذف خوشه هایی که تعداد اعضای آنها کمتر از مقدار n باشد

به علت وجود نویز و یا پراکندگی بیش از حد اطلاعات امکان دارد خوشه هایی اضافه تولید شده باشد. به همین علت بعد از خوشه بندی، معمولا خوشه ها مورد ارزیابی قرار می گیرند و در صورتیکه خوشه ای بعد از ارزیابی مورد تایید نبود می بایست آن خوشه حذف شود. برای این منظور در برنامه از متد GenerateMeanClustersAfterMergin استفاده شده است.

در این برنامه برای خوشه هایی که بر اساس Freq ساخته شده اند تعداد اعضای آنها را مورد ارزیابی قرار می دهیم.

برای این منظور در صورتیکه تعداد خوشه ها از یک عدد محاسبه شده n کمتر بود، آن خوشه را حذف می نماییم. برای محاسبه n از فرمول زیر محاسبه می شود

$$m = \max(Count)$$
$$if\ m \geq ۳۰۰ \rightarrow n = ۲۰$$
$$if\ m \geq ۱۰۰ \rightarrow n = ۱۰$$
$$if\ m \geq ۵۰ \rightarrow n = ۴$$

```
        private static void GenerateMeanClustersAfterMergin(int First_Index, long
PDW_Count)
        {
            #region Set m to Max(Mean_Cluisters_Freq_P.Count
            m = 0;                                          //Pulse
            for (i = 0; i < ClusterCounter_Freq_P; i++)
            {
                if (m < Mean_Clusters_Freq_P[i].Count)
                    m = Mean_Clusters_Freq_P[i].Count;
            }
            #endregion

            #region Remove Mean_Clusters_Freq_P than have less than n item
            if (m >= 50)
            {
                if (m >= 300)
                    n = 20;
                else if (m >= 100)
                    n = 10;
                else if (m >= 50)
                    n = 4;

                j = 0;
                for (i = 0; i < ClusterCounter_Freq_P; i++)
                {
                    if (Mean_Clusters_Freq_P[i].Count >= n)
                    {
                        if (i != j)
                        {
                            Mean_Clusters_Freq_P[j] = Mean_Clusters_Freq_P[i];
                            for (int colIndex = 0; colIndex <
TempClusterIndexP.GetLength(1); colIndex++)
                            {
                                TempClusterIndexP[j, colIndex] =
TempClusterIndexP[i, colIndex];    //here is new
                            }
                        }
```

```
                        j++;
                    }
                }
                ClusterCounter_Freq_P = j;
                //Mean_Clusters_Freq_P = new
TMean_Clusters_Freq[ClusterCounter_Freq_P];
                SetLength(ref Mean_Clusters_Freq_P, ClusterCounter_Freq_P);
                ResizeArray(ref TempClusterIndexP, ClusterCounter_Freq_P,
TempClusterIndexP.GetLength(1));
            }
            #endregion

            #region Set m to Max(Mean_Cluisters_Freq_CW.Count
            m = 0;                                      //CW
            for (i = 0; i < ClusterCounter_Freq_CW; i++)
            {
                if (m < Mean_Clusters_Freq_CW[i].Count)
                    m = Mean_Clusters_Freq_CW[i].Count;
            }
            #endregion

            #region Remove Mean_Clusters_Freq_CW than have less than n item
            if (m >= 50)
            {
                if (m >= 300)
                    n = 20;
                else if (m >= 100)
                    n = 10;
                else if (m >= 50)
                    n = 4;

                j = 0;
                for (i = 0; i < ClusterCounter_Freq_CW; i++)
                {
                    if (Mean_Clusters_Freq_CW[i].Count >= n)
                    {
                        if (i != j)
                        {
                            Mean_Clusters_Freq_CW[j] = Mean_Clusters_Freq_CW[i];
                            for (int colIndex = 0; colIndex <
TempClusterIndexCW.GetLength(1); colIndex++)
                                TempClusterIndexCW[j, colIndex] =
TempClusterIndexCW[i, colIndex];    //here is new
                        }
                        j++;
                    }
                }
                ClusterCounter_Freq_CW = j;
                //Mean_Clusters_Freq_CW = new
TMean_Clusters_Freq[ClusterCounter_Freq_CW];
                SetLength(ref Mean_Clusters_Freq_CW, ClusterCounter_Freq_CW);
                ResizeArray(ref TempClusterIndexCW, ClusterCounter_Freq_CW,
ClusterCounter_Freq_CW);
            }
            #endregion

            #region Sorting Mean_Clusters_Freq_P Desc On Count
            ResizeArray(ref TempClusterIndex, t);
            for (i = 0; i <= ClusterCounter_Freq_P - 1; i++)
                for (j = 0; j <= ClusterCounter_Freq_P - 2; j++)
                {
                    if (Mean_Clusters_Freq_P[j].Count < Mean_Clusters_Freq_P[j +
1].Count)
                    {
```

```
                        Temp_Mean_Freq_Clusters = Mean_Clusters_Freq_P[j];
                        Mean_Clusters_Freq_P[j] = Mean_Clusters_Freq_P[j + 1];
                        Mean_Clusters_Freq_P[j + 1] = Temp_Mean_Freq_Clusters;
                        for (k = 0; k <= t - 1; k++)
                        {
                            TempClusterIndex[k] = TempClusterIndexP[j, k];
                            TempClusterIndexP[j, k] = TempClusterIndexP[j + 1, k];
                            TempClusterIndexP[j + 1, k] = TempClusterIndex[k];
                        }
                    }
                }

            Array.Resize(ref TempClusterIndex, u);
            for (i = 0; i <= ClusterCounter_Freq_CW - 1; i++)
                for (j = 0; j <= ClusterCounter_Freq_CW - 2; j++)
                {
                    if (Mean_Clusters_Freq_CW[j].Count > Mean_Clusters_Freq_CW[j +
1].Count)
                    {
                        Temp_Mean_Freq_Clusters = Mean_Clusters_Freq_CW[j];
                        Mean_Clusters_Freq_CW[j] = Mean_Clusters_Freq_CW[j + 1];
                        Mean_Clusters_Freq_CW[j + 1] = Temp_Mean_Freq_Clusters;
                        for (k = 0; k <= u - 1; k++)
                        {
                            TempClusterIndex[k] = TempClusterIndexCW[j, k];
                            TempClusterIndexCW[j, k] = TempClusterIndexCW[j + 1, k];
                            TempClusterIndexCW[j + 1, k] = TempClusterIndex[k];
                        }
                    }
                }
            #endregion

            #region Filling MapOldToNewTempClustersFalse &&
MapOldToNewTempClustersTrue
            Array.Resize(ref MapOldToNewTempClustersFalse, t);
            for (j = 0; j <= t - 1; j++)
            {
                found = false;
                for (i = 0; i <= ClusterCounter_Freq_P - 1; i++)
                {
                    if (TempClusterIndexP[i, j])
                    {
                        MapOldToNewTempClustersFalse[j] = i;
                        found = true;
                        break;
                    }
                }
                if (!found)
                {
                    MapOldToNewTempClustersFalse[j] = -1;
                }
            }

            Array.Resize(ref MapOldToNewTempClustersTrue, u);
            for (j = 0; j <= u - 1; j++)
            {
                found = false;
                for (i = 0; i <= ClusterCounter_Freq_CW - 1; i++)
                {
                    if (TempClusterIndexCW[i, j])
                    {
                        MapOldToNewTempClustersTrue[j] = i;
                        found = true;
                        break;
```

```
                    }
                }
                if (!found)
                {
                    MapOldToNewTempClustersTrue[j] = -1;
                }
            }
            #endregion

            #region Setting Mean_Clusters_Freq_P[i].First_Index
            m = 0;
            for (i = 0; i <= ClusterCounter_Freq_P - 1; i++)
            {
                Mean_Clusters_Freq_P[i].First_Index = m;
                Mean_Clusters_Freq_P[i].Last_Index = -1;
                m += Mean_Clusters_Freq_P[i].Count;
            }
            m = 0;
            for (i = 0; i <= ClusterCounter_Freq_CW - 1; i++)
            {
                Mean_Clusters_Freq_CW[i].First_Index = m;
                Mean_Clusters_Freq_CW[i].Last_Index = -1;
                m += Mean_Clusters_Freq_CW[i].Count;
            }
            #endregion

            #region Filling Cluster_Members_Freq_P && Setting
Mean_Clusters_Freq_P[j].Last_Index
            if (ClusterCounter_Freq_P != 0)
                for (i = 0; i <= PDW_Count - 1; i++)
                {
                    if (Cluster_Members_P[i] == -1)
                        continue;
                    j = MapOldToNewTempClustersFalse[Cluster_Members_P[i]];
                    if (j == -1)
                        continue;
                    m = Mean_Clusters_Freq_P[j].Last_Index;
                    m++;
                    Cluster_Members_Freq_P[Mean_Clusters_Freq_P[j].First_Index + m]
= i;
                    Mean_Clusters_Freq_P[j].Last_Index = m;
                }
            if (ClusterCounter_Freq_CW != 0)
                for (i = 0; i <= PDW_Count - 1; i++)
                {
                    if (Cluster_Members_CW[i] == -1)
                        continue;
                    j = MapOldToNewTempClustersTrue[Cluster_Members_CW[i]];
                    if (j == -1)
                        continue;
                    m = Mean_Clusters_Freq_CW[j].Last_Index;
                    m++;
                    Cluster_Members_Freq_CW[Mean_Clusters_Freq_CW[j].First_Index +
m] = i;
                    Mean_Clusters_Freq_CW[j].Last_Index = m;
                }
            #endregion

            #region Setting Mean_Clusters_Freq_P[i].Std
            m = 0;
            for (i = 0; i <= ClusterCounter_Freq_P - 1; i++)
            {
                d1 = 0;
```

```
                for (j = Mean_Clusters_Freq_P[i].First_Index; j <=
Mean_Clusters_Freq_P[i].First_Index + Mean_Clusters_Freq_P[i].Last_Index; j++)
                {
                    d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[j] +
First_Index].Freq - Mean_Clusters_Freq_P[i].Mean_Freq);
                }
                Mean_Clusters_Freq_P[i].Std = Math.Sqrt(d1 /
Mean_Clusters_Freq_P[i].Count);
            }
            n = 0;
            for (i = 0; i <= ClusterCounter_Freq_CW - 1; i++)
            {
                d1 = 0;
                for (j = Mean_Clusters_Freq_CW[i].First_Index; j <=
Mean_Clusters_Freq_CW[i].First_Index + Mean_Clusters_Freq_CW[i].Last_Index; j++)
                {
                    d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_Freq_CW[j] +
First_Index].Freq - Mean_Clusters_Freq_CW[i].Mean_Freq);
                }
                Mean_Clusters_Freq_CW[i].Std = Math.Sqrt(d1 /
Mean_Clusters_Freq_CW[i].Count);
            }
            #endregion

        }
```

## ۱۲–۸– خوشه بندی داده ها به مرکزیت PW در Mean_Clusters_PW

پس از خوشه بندی اطلاعات توسط Freq، آنها را مورد ارزیابی قرار دادیم و اطلاعاتی که در ارزیابی ما مورد قبول نبودند را حذف کردیم. حال رکورد های باقیمانده را بر اساس PW خوشه بندی می نماییم. در برنامه این خوشه بندی در متدی با نام ClusteringPW انجام می شود.

لیست Mean_Clusters_PW دارای اطلاعات خوشه هایی با مرکزیت میانگین PW می باشد. هر خانه در این لیست دارای دو متغیر زیر می باشد:

- Count: تعداد رکورد های هر خوشه را نمایش می دهد.
- Mean_PW: این متغیر در واقع مرکز خوشه می باشد که میانگین PW کلیه رکوردهای آن خوشه می باشد که از فرمول زیر محاسبه می شود.

$$if\ count = ۰\ then\ Mean_{PW}: PW$$
$$else\ Mean_{PW} = \frac{Mean_{PW} * Count + PW}{Count + ۱}$$

```
        private static void ClusteringPW(int First_Index)
        {
            for (i = 0; i <= ClusterCounter_Freq_P - 1; i++)
            {
                ClusterCounter_PW = 0;
                t = 0;
                for (k = Mean_Clusters_Freq_P[i].First_Index; k <=
Mean_Clusters_Freq_P[i].First_Index + Mean_Clusters_Freq_P[i].Last_Index; k++)
                {
```

```
                    PW = Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[k] +
First_Index].PW;
                    found = false;
                    if (ClusterCounter_PW != 0)
                    {
                        for (j = 0; j <= ClusterCounter_PW - 1; j++)
                        {
                            Max_Distance = Math.Abs(PW -
Mean_Clusters_PW[j].Mean_PW);
                            if (ClusteringParams.PW_Defult == 0)
                            {
                                for (m = 0; m <= 4; m++)
                                {
                                    if (PW < ClusteringParams.PW_Ther_Value[1, m])
                                    {
                                        a = ClusteringParams.PW_Ther_Value[2, m];
//Pulse_Thre
                                        break;
                                    }
                                }
                            }
                            else
                            {
                                if (Mean_Clusters_PW[j].Mean_PW < 800)
                                    a = ClusteringParams.Down_Diff_PW / 2;
                                else if (Mean_Clusters_PW[j].Mean_PW < 1200)
                                    a = ClusteringParams.Down_Diff_PW * 0.8f;
                                else
                                    a = ClusteringParams.Down_Diff_PW;
                            }

                            if (((ClusteringParams.PW_Defult == 0) && (Max_Distance
<= a)) ||
                                ((ClusteringParams.PW_Defult == 1) &&
                                (((Max_Distance <= Mean_Clusters_PW[j].Mean_PW *
0.1) &&
                                (Max_Distance <= ClusteringParams.Up_Diff_PW)) ||
(Max_Distance <= a))))
                            {
                                Mean_Clusters_PW[j].Mean_PW =
(Mean_Clusters_PW[j].Mean_PW * Mean_Clusters_PW[j].Count + PW) /
(Mean_Clusters_PW[j].Count + 1);
                                Mean_Clusters_PW[j].Count =
Mean_Clusters_PW[j].Count + 1;
                                Cluster_Members_PW[t] = j;
                                found = true;
                                break;
                            }
                        }
                    }
                    if (!found)
                    {
                        //Mean_Clusters_PW = new TMean_Clusters_PW[ClusterCounter_PW
+ 1];
                        SetLength(ref Mean_Clusters_PW, ClusterCounter_PW + 1);
                        Mean_Clusters_PW[ClusterCounter_PW].Mean_Freq =
Mean_Clusters_Freq_P[i].Mean_Freq;
                        Mean_Clusters_PW[ClusterCounter_PW].Mean_PW = PW;
                        Mean_Clusters_PW[ClusterCounter_PW].Count = 1;
                        Mean_Clusters_PW[ClusterCounter_PW].Last_Index = -1;
                        Cluster_Members_PW[t] = ClusterCounter_PW;
                        ClusterCounter_PW++;
                    }
                    t++;
```

```
                }

                // we have pw clusters         PW Indexs of Freq Indexes of
PDWElements
                m = 0;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                {
                    Mean_Clusters_PW[k].First_Index = m;
                    m += Mean_Clusters_PW[k].Count;
                }

                n = Mean_Clusters_Freq_P[i].First_Index;
                if (ClusterCounter_PW != 0)
                    for (k = 0; k <= t - 1; k++)
                    {
                        j = Cluster_Members_PW[k];
                        m = Mean_Clusters_PW[j].Last_Index;
                        m++;

                        Cluster_Members_Freq_PW[n + Mean_Clusters_PW[j].First_Index
+ m] = k;   // =Cluster_Members_Freq_P[Mean_Clusters_Freq_P[i].First_Index+k]
                        Mean_Clusters_PW[j].Last_Index = m;
                    }

                m = 0;
                n = Mean_Clusters_Freq_P[i].First_Index;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                {

                    d1 = 0;
                    for (j = n + Mean_Clusters_PW[k].First_Index; j <= n +
Mean_Clusters_PW[k].First_Index + Mean_Clusters_PW[k].Last_Index; j++)
                    {
                        d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[n +
Cluster_Members_Freq_PW[j]] + First_Index].PW - Mean_Clusters_PW[k].Mean_PW);
                    }
                    Mean_Clusters_PW[k].PW_Std = Math.Sqrt(d1 /
Mean_Clusters_PW[k].Count);
                }

                // pre operations for merging pw clusters

                ResizeArray(ref TempClusterIndexP, ClusterCounter_PW,
ClusterCounter_PW);
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                    for (j = 0; j <= ClusterCounter_PW - 1; j++)
                    {
                        if (k != j)
                            TempClusterIndexP[k, j] = false;
                        else
                            TempClusterIndexP[k, j] = true;
                    }
                t = ClusterCounter_PW;
                k = 0;
                while (k < ClusterCounter_PW - 1)
                {
                    if (Mean_Clusters_PW[k].Count == 0)
                    {
                        k++;
                        continue;
                    }
                    found = false;
                    for (j = k + 1; j <= ClusterCounter_PW - 1; j++)
```

```
                    {
                        PW = Mean_Clusters_PW[k].Mean_PW;
                        Max_Distance = Math.Abs(Mean_Clusters_PW[j].Mean_PW - PW);
                        if (ClusteringParams.PW_Defult == 0)
                        {
                            for (m = 0; m <= 4; m++)
                            {
                                if (PW < ClusteringParams.PW_Ther_Value[1, m])
                                {
                                    a = ClusteringParams.PW_Ther_Value[2, m];
//Pulse_Thre
                                    break;
                                }
                            }
                        }

                        if ((((ClusteringParams.PW_Defult == 1) &&
                            ((Max_Distance <= PW * 0.08) ||
                            ((Max_Distance <= 120) &&
                            (Math.Max(Mean_Clusters_PW[k].PW_Std,
Mean_Clusters_PW[j].PW_Std) >= 25))))) ||
                            ((ClusteringParams.PW_Defult == 0) && (Max_Distance <=
a)))
                        {
                            if (Mean_Clusters_PW[j].Count == 0)
                                continue;

                            for (m = 0; m <= ClusterCounter_PW - 1; m++)
                            {
                                TempClusterIndexP[k, m] = TempClusterIndexP[k, m] ||
TempClusterIndexP[j, m];
                            }

                            n = Mean_Clusters_PW[k].Count +
Mean_Clusters_PW[j].Count;
                            Mean_Clusters_PW[k].Mean_PW =
(Mean_Clusters_PW[k].Mean_PW * Mean_Clusters_PW[k].Count +
Mean_Clusters_PW[j].Mean_PW * Mean_Clusters_PW[j].Count) / n;

                            Mean_Clusters_PW[k].Count = n;
                            Mean_Clusters_PW[j].Count = 0;
                            found = true;
                            break;
                        }
                    }
                    if (!found)
                        k++;
                }

                j = 0;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
//Delete Empty Clusters
                {
                    if (Mean_Clusters_PW[k].Count != 0)
                    {
                        if (k != j)
                        {
                            Mean_Clusters_PW[j] = Mean_Clusters_PW[k];
                            for (int colIndex = 0; colIndex <
TempClusterIndexP.GetLength(1); colIndex++)
                                TempClusterIndexP[j, colIndex] =
TempClusterIndexP[k, colIndex];
                        }
                        j++;
```

```
                    }
                }
                ClusterCounter_PW = j;
                SetLength(ref Mean_Clusters_PW, ClusterCounter_PW);
                ResizeArray(ref TempClusterIndexP, ClusterCounter_PW,
TempClusterIndexP.GetLength(1));

                //*****************************************************************

                m = 0;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                {
                    if (m < Mean_Clusters_PW[k].Count)
                        m = Mean_Clusters_PW[k].Count;
                }
                if (m >= 3)
                {
                    if (m >= 8)
                        n = 3;
                    else if (m >= 3)
                        n = 2;

                    j = 0;
                    for (k = 0; k <= ClusterCounter_PW - 1; k++)
                    {
                        if (Mean_Clusters_PW[k].Count >= n)
                        {
                            if (k != j)
                            {
                                Mean_Clusters_PW[j] = Mean_Clusters_PW[k];
                                for (int colIndex = 0; colIndex <
TempClusterIndexP.GetLength(1); colIndex++)
                                    TempClusterIndexP[j, colIndex] =
TempClusterIndexP[k, colIndex];    //here is new
                            }
                            j++;
                        }
                    }
                    ClusterCounter_PW = j;
                    SetLength(ref Mean_Clusters_PW, ClusterCounter_PW);
                    ResizeArray(ref TempClusterIndexP, ClusterCounter_PW,
TempClusterIndexP.GetLength(1));
                }

                //*****************************************************************

                ResizeArray(ref TempClusterIndex, t);
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
//Sort Pulse Clusters
                    for (j = 0; j <= ClusterCounter_PW - 2; j++)
                    {
                        if (Mean_Clusters_PW[j].Count < Mean_Clusters_PW[j +
1].Count)
                        {
                            Temp_Mean_PW_Clusters = Mean_Clusters_PW[j];
                            Mean_Clusters_PW[j] = Mean_Clusters_PW[j + 1];
                            Mean_Clusters_PW[j + 1] = Temp_Mean_PW_Clusters;
                            for (m = 0; m <= t - 1; m++)
                            {
                                TempClusterIndex[m] = TempClusterIndexP[j, m];
                                TempClusterIndexP[j, m] = TempClusterIndexP[j + 1,
m];
                                TempClusterIndexP[j + 1, m] = TempClusterIndex[m];
                            }
```

```
                        }
                    }

                if (ClusterCounter_PW != 0)          //moved to here
                {
                    if (Mean_Clusters_PW[0].Count < Math.Min(4,
(Mean_Clusters_Freq_P[i].Count) / 3))
                    {
                        ClusterCounter_PW = 0;
                        break;
                    }
                }

                Array.Resize(ref MapOldToNewTempClustersFalse, t);
                for (j = 0; j <= t - 1; j++)
                {
                    found = false;
                    for (k = 0; k <= ClusterCounter_PW - 1; k++)
                    {
                        if (TempClusterIndexP[k, j])
                        {
                            MapOldToNewTempClustersFalse[j] = k;
                            found = true;
                            break;
                        }
                    }
                    if (!found)
                    {
                        MapOldToNewTempClustersFalse[j] = -1;
                    }
                }
                m = 0;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                {
                    Mean_Clusters_PW[k].First_Index = m;
                    Mean_Clusters_PW[k].Last_Index = -1;
                    m += Mean_Clusters_PW[k].Count;
                }

                n = Mean_Clusters_Freq_P[i].First_Index;
                if (ClusterCounter_PW != 0)
                    for (k = 0; k <= Mean_Clusters_Freq_P[i].Count - 1; k++)
                    {
                        j = MapOldToNewTempClustersFalse[Cluster_Members_PW[k]];
                        if (j == -1) continue;
                        m = Mean_Clusters_PW[j].Last_Index;
                        m++;
                        Cluster_Members_Freq_PW[n + Mean_Clusters_PW[j].First_Index
+ m] = k;   //
                        Mean_Clusters_PW[j].Last_Index = m;
                    }

                m = 0;
                n = Mean_Clusters_Freq_P[i].First_Index;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                {
                    d1 = 0;
                    for (j = n + Mean_Clusters_PW[k].First_Index; j <= n +
Mean_Clusters_PW[k].First_Index + Mean_Clusters_PW[k].Last_Index; j++)
                    {
                        d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[n +
Cluster_Members_Freq_PW[j]] + First_Index].PW - Mean_Clusters_PW[k].Mean_PW);
                    }
```

```
                    Mean_Clusters_PW[k].PW_Std = Math.Sqrt(d1 /
Mean_Clusters_PW[k].Count);
                }
                //Qualification

                n = 0;
                for (j = 0; j <= ClusterCounter_PW - 1; j++)
                {
                    PW = Mean_Clusters_PW[j].Mean_PW;
                    for (k = 0; k <= 4; k++)
                    {
                        if (PW < ClusteringParams.PW_Ther_Value[1, k])
                        {
                            a = ClusteringParams.PW_Ther_Value[3, k];
                            b = ClusteringParams.PW_Ther_Value[4, k];
                            break;
                        }
                    }
                    if (((Mean_Clusters_PW[j].PW_Std > a) &&
(Mean_Clusters_PW[j].Count < Mean_Clusters_PW[0].Count / 5)) ||
                       ((Mean_Clusters_PW[j].PW_Std > b) &&
(Mean_Clusters_PW[j].Count < Mean_Clusters_PW[0].Count / 15)) ||
                       (Mean_Clusters_PW[j].Count < Mean_Clusters_PW[0].Count / 8)
||
                       (Mean_Clusters_PW[j].Count < Mean_Clusters_Freq_P[i].Count /
10) ||
                       (Mean_Clusters_PW[j].Count < 4))
                    {
                        Mean_Clusters_PW[j].Qualify = false;
                    }
                    else
                    {
                        Mean_Clusters_PW[j].Qualify = true;
                        n++;   //Number of Qualified PW Clusters
                    }
                }

                //Thresholding
                m = 0;
                t = Mean_Clusters_Freq_P[i].First_Index;
                for (k = 0; k <= ClusterCounter_PW - 1; k++)
                {
                    for (j = t + Mean_Clusters_PW[k].First_Index; j <= t +
Mean_Clusters_PW[k].First_Index + Mean_Clusters_PW[k].Last_Index; j++)
                    {
                        Cluster_Amp[m] =
Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[t + Cluster_Members_Freq_PW[j]]
+ First_Index].AMP;
                        Cluster_PWClusterIndex[m] = k;   //Cluster Number
                        Cluster_Members_PW[m] = Cluster_Members_Freq_P[t +
Cluster_Members_Freq_PW[j]] + First_Index; //=j
                        m++;
                    }
                    Mean_Clusters_PW[k].Flag1 = false;
                    Mean_Clusters_PW[k].HighAmpCount = 0;
                    Mean_Clusters_PW[k].HighAmpMeanPW = 0;
                }


                if (m > 0)
                    Sorting.QuickSortAid(Cluster_Amp, Cluster_PWClusterIndex,
Cluster_Members_PW, 0, m - 1);

                if ((n > 5) || (ClusterCounter_PW > 7))
```

```
                {
                    b = Cluster_Amp[0] - 2;  //Threhsold
                    j = 1;                   //Counter
                    n = -1;                  //Last index with Amp greater than
threshold
                    u = 0;                   //Number of Amps greater than threshold
                    while (true)
                    {
                        t = n + 1;
                        for (k = t; k <= m - 1; k++)
                        {
                            if (Cluster_Amp[k] < b)
                                break;
                            Mean_Clusters_PW[Cluster_PWClusterIndex[k]].Flag1 =
true;

Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpMeanPW =
(Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpMeanPW *
Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount +
Online_PDWProcess.PDWElements[Cluster_Members_PW[k]].PW) /
(Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount + 1);

Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount++;
                            n = k;
                            u++;
                        }

                        while (((n + 1 < m / 10) && (n + 1 < 5)) || ((m > 150) && (n
+ 1 < 15)))
                        {
                            b = b - 1;
                            t = n + 1;
                            for (k = t; k <= m - 1; k++)
                            {
                                if (Cluster_Amp[k] < b)
                                    break;
                                Mean_Clusters_PW[Cluster_PWClusterIndex[k]].Flag1 =
true;

Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpMeanPW =
(Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpMeanPW *
Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount +
Online_PDWProcess.PDWElements[Cluster_Members_PW[k]].PW) /
(Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount + 1);

Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount++;
                                n = k;
                                u++;
                            }
                        }

                        b = b - 2;

                        k = 0;
                        for (t = 0; t <= ClusterCounter_PW - 1; t++)
                        {
                            if (Mean_Clusters_PW[t].Flag1)
                            {
                                if (k < Mean_Clusters_PW[t].HighAmpCount)
                                    k = Mean_Clusters_PW[t].HighAmpCount;
                            }
                        }

                        if (k >= 3)
```

```
                        {
                            if (k >= 8)
                                t = 3;
                            else if (k >= 3)
                                t = 2;

                            for (k = 0; k <= ClusterCounter_PW - 1; k++)
                            {
                                if ((Mean_Clusters_PW[k].Flag1) &&
                                    (Mean_Clusters_PW[k].HighAmpCount < t))
                                    Mean_Clusters_PW[k].Flag1 = false;
                            }
                        }

                        k = 0;
                        for (t = 0; t <= ClusterCounter_PW - 1; t++)
                        {
                            if (!Mean_Clusters_PW[t].Flag1)
                                continue;
                            Cluster_Amp[m + k] = Mean_Clusters_PW[t].HighAmpCount;
                            Cluster_PWClusterIndex[m + k] = t;
                            k++;
                        }
                        if (k != 0)
                        {
                            Sorting.QuickSortAid(Cluster_Amp,
Cluster_PWClusterIndex, Cluster_Members_PW, m, m + k - 1);
                        }
                        for (t = 0; t <= Math.Min(2, k - 1); t++)
                        {
                            if (t == 0)
                                a = (float)Mean_Clusters_PW[Cluster_PWClusterIndex[m
+ t]].HighAmpMeanPW;
                            if (t == 1)
                                e = (float)Mean_Clusters_PW[Cluster_PWClusterIndex[m
+ t]].HighAmpMeanPW;
                            if (t == 2)
                                d = (float)Mean_Clusters_PW[Cluster_PWClusterIndex[m
+ t]].HighAmpMeanPW;
                        }

                        if ((j > 2) &&
                            ((k == 2) || ((k > 2) && (d < Math.Max(a, e)))) &&
                                (Math.Abs(e - a) > 20))
                        {
                            break;
                        }

                        if ((k > 3) || (j > 3) || (b < Cluster_Amp[m - 1] + 2))
                        {
                            break;
                        }

                        j = j + 1;
                    }
                }
                else
                {
                    b = Cluster_Amp[0] + (Cluster_Amp[m - 1] - Cluster_Amp[0]) / 4;
//Threhsold
                    j = 1;                  //Counter
                    n = -1;                 //Last index with Amp greater than
threshold
                    u = 0;                  //Number of Amps greater than threshold
```

```
                    while (true)
                    {
                        t = n + 1;
                        for (k = t; k <= m - 1; k++)
                        {
                            if (Cluster_Amp[k] < b)
                                break;
                            Mean_Clusters_PW[Cluster_PWClusterIndex[k]].Flag1 =
true;

Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpMeanPW =
(Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpMeanPW *
Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount +
Online_PDWProcess.PDWElements[Cluster_Members_PW[k]].PW) /
(Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount + 1);

Mean_Clusters_PW[Cluster_PWClusterIndex[k]].HighAmpCount++;
                            n = k;
                            u++;
                        }
                        k = 0;
                        for (t = 0; t <= ClusterCounter_PW - 1; t++)
                        {
                            if (Mean_Clusters_PW[t].Flag1)
                            {
                                if (k < Mean_Clusters_PW[t].HighAmpCount)
                                    k = Mean_Clusters_PW[t].HighAmpCount;
                            }
                        }
                        if (k >= 3)
                        {
                            if (k >= 8)
                                t = 3;
                            else if (k >= 3)
                                t = 2;
                            for (k = 0; k <= ClusterCounter_PW - 1; k++)
                            {
                                if ((Mean_Clusters_PW[k].Flag1) &&
(Mean_Clusters_PW[k].HighAmpCount < t))
                                    Mean_Clusters_PW[k].Flag1 = false;
                            }
                        }

                        k = 0;
                        for (t = 0; t <= ClusterCounter_PW - 1; t++)
                        {
                            if (!Mean_Clusters_PW[t].Flag1)
                                continue;
                            Cluster_Amp[m + k] = Mean_Clusters_PW[t].HighAmpCount;
                            Cluster_PWClusterIndex[m + k] = t;
                            k++;
                        }
                        if (k != 0)
                        {
                            Sorting.QuickSortAid(Cluster_Amp,
Cluster_PWClusterIndex, Cluster_Members_PW, m, m + k - 1);
                        }

                        if ((j < 3) && (k < 2))
                            b = b - 1;
                        else if (k < 5)
                            break;
                        else if (b + 1.8 < Cluster_Amp[0])
                            b = b + 1;
```

```
                        else
                            break;
                        j = j + 1;
                    }
                }

                //Qualification
                u = 0;
                t = Cluster_PWClusterIndex[m];    //Major Cluster
                for (j = 0; j <= ClusterCounter_PW - 1; j++)
                {
                    if ((Mean_Clusters_PW[j].Flag1) &&
(Mean_Clusters_PW[j].HighAmpCount != 0))
                    {
                        PW = Mean_Clusters_PW[j].HighAmpMeanPW;
                        if (PW > 200000)
                            a = 1500;
                        else if (PW > 30000)
                            a = 750;
                        else if (PW > 5000)
                            a = 250;  //200
                        else if (PW > 3000)
                            a = 100;
                        else
                            a = 80;

                        if ((Math.Abs(PW - Mean_Clusters_PW[j].Mean_PW) <= a) ||
(Mean_Clusters_PW[j].Qualify))
                        {
                            d1 = 0;
                            m = 0;
                            for (k = 0; k <= n; k++)
                            {
                                if (Cluster_PWClusterIndex[k] == j)
                                {
                                    d1 = d1 +
MathMethods.Sqr(Online_PDWProcess.PDWElements[Cluster_Members_PW[k]].PW - PW);
                                    m++;
                                }
                            }
                            Mean_Clusters_PW[j].HighAmpPWStd = Math.Sqrt(d1 / m);

                            for (k = 0; k <= 4; k++)
                            {
                                if (PW < ClusteringParams.PW_Ther_Value[1, k])
                                {
                                    a = ClusteringParams.PW_Ther_Value[3, k];
                                    b = ClusteringParams.PW_Ther_Value[4, k];
                                    break;
                                }
                            }
                            if (((Mean_Clusters_PW[j].HighAmpPWStd > a) &&
(Mean_Clusters_PW[j].HighAmpCount < Mean_Clusters_PW[t].HighAmpCount / 5)) ||
                                ((Mean_Clusters_PW[j].HighAmpPWStd > b) &&
(Mean_Clusters_PW[j].HighAmpCount < Mean_Clusters_PW[t].HighAmpCount / 15)) ||
                                (Mean_Clusters_PW[j].HighAmpCount <
Mean_Clusters_PW[t].HighAmpCount / 8) ||
                                (Mean_Clusters_PW[j].HighAmpCount < (n + 1) / 10) ||
                                (Mean_Clusters_PW[j].HighAmpCount < 4))
                            {
                                Mean_Clusters_PW[j].Qualify = false;
                            }
                            else
                            {
```

```
                            Mean_Clusters_PW[j].Qualify = true;
                            u++;   //Number of Qualified PW Clusters
                        }
                    }
                    else
                    {
                        Mean_Clusters_PW[j].Qualify = false;
                    }
                }
                else
                {
                    Mean_Clusters_PW[j].Qualify = false;
                }
            }

            SetLength(ref TotalClusters_P, ClusterCounter_P + u);
            SetLength(ref W_TotalClusters_P, ClusterCounter_P + u);
            k = ClusterCounter_P;
            ClusterCounter_P += u;
            for (j = 0; j <= ClusterCounter_PW - 1; j++)        //build current
pulse clusters
            {
                if (!Mean_Clusters_PW[j].Qualify)
                    continue;

                TotalClusters_P[k].First_Index =
Mean_Clusters_Freq_P[i].First_Index + Mean_Clusters_PW[j].First_Index;
                TotalClusters_P[k].First_FIndex =
Mean_Clusters_Freq_P[i].First_Index;
                TotalClusters_P[k].Member = Mean_Clusters_PW[j].Count;
                TotalClusters_P[k].AdditiveCount = Mean_Clusters_PW[j].Count;
                TotalClusters_P[k].PW = (float)Mean_Clusters_PW[j].Mean_PW;
                TotalClusters_P[k].Freq =
(float)Mean_Clusters_Freq_P[i].Mean_Freq;
                TotalClusters_P[k].Repeat_ = 1;
                TotalClusters_P[k].Update_Flag = true;
                TotalClusters_P[k].CW_Flag = false;
                TotalClusters_P[k].ScanType = Constants.ScanMaxIndex;
                TotalClusters_P[k].ScanPeriod = -1;
                TotalClusters_P[k].LastScanIndex = -1;
                TotalClusters_P[k].FreqStd = (float)Mean_Clusters_Freq_P[i].Std;
                TotalClusters_P[k].PWStd = (float)Mean_Clusters_PW[j].PW_Std;
                for (m = 0; m <= 9; m++)
                {
                    TotalClusters_P[k].ScanTypes[m] = Constants.ScanMaxIndex;
                    TotalClusters_P[k].ScanPeriods[m] = 0;
                }

                TotalClusters_P[k].AMP = -200;
                TotalClusters_P[k].DFAmp = -200;
                TotalClusters_P[k].HasAOA = false;
                TotalClusters_P[k].DiffAmp = -200;
                TotalClusters_P[k].Tilt = -1000;

                for (m = 0; m <= Constants.AOAArrayMaxIndex; m++)
                {
                    TotalClusters_P[k].AOAMeans[m] = 0;
                    TotalClusters_P[k].AOACounts[m] = 0;
                    TotalClusters_P[k].AOAMaxDiffs[m] = -200;
                    TotalClusters_P[k].AOAAOAMaxDiffs[m] = -1000;
                }
                TotalClusters_P[k].AOALastIndex = -1;
                TotalClusters_P[k].TempMaxDiff = -200;
                TotalClusters_P[k].TempAOAMaxDiff = -1000;
```

```
                    TotalClusters_P[k].HasTempAOA = false;
                    t = TotalClusters_P[k].First_FIndex;
                    n = 0;
                    for (m = TotalClusters_P[k].First_Index; m <=
TotalClusters_P[k].First_Index + TotalClusters_P[k].Member - 1; m++)
                    {
                        PDWElement =
Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[t + Cluster_Members_Freq_PW[m]]
+ First_Index];

                        TotalClusters_P[k].AMP = Math.Max(TotalClusters_P[k].AMP,
PDWElement.AMP);

                        if (PDWElement.HasAOA)
                        {
                            a = PDWElement.DF2 - PDWElement.DF1;
                            found = false;
                            for (n = 0; n <= TotalClusters_P[k].AOALastIndex; n++)
                            {
                                b = TotalClusters_P[k].AOAMeans[n];
                                if ((Math.Abs(PDWElement.AOA - b) <= MaxAOADiff) ||
                                    (360 + Math.Min(PDWElement.AOA, b) -
Math.Max(PDWElement.AOA, b) <= MaxAOADiff))
                                {
                                    if (Math.Abs(PDWElement.AOA - b) <= MaxAOADiff)
                                        TotalClusters_P[k].AOAMeans[n] =
(TotalClusters_P[k].AOAMeans[n] * TotalClusters_P[k].AOACounts[n] + PDWElement.AOA)
/ (TotalClusters_P[k].AOACounts[n] + 1);
                                    else
                                    {
                                        if (PDWElement.AOA < b)
                                            TotalClusters_P[k].AOAMeans[n] =
(TotalClusters_P[k].AOAMeans[n] * TotalClusters_P[k].AOACounts[n] + PDWElement.AOA +
360) / (TotalClusters_P[k].AOACounts[n] + 1);
                                        else
                                            TotalClusters_P[k].AOAMeans[n] = ((360 +
TotalClusters_P[k].AOAMeans[n]) * TotalClusters_P[k].AOACounts[n] + PDWElement.AOA)
/ (TotalClusters_P[k].AOACounts[n] + 1);
                                        if (TotalClusters_P[k].AOAMeans[n] >= 360)
                                            TotalClusters_P[k].AOAMeans[n] =
TotalClusters_P[k].AOAMeans[n] - 360;
                                    }
                                    TotalClusters_P[k].AOATimes[n] = PDWElement.TOA;
                                    TotalClusters_P[k].AOACounts[n] =
TotalClusters_P[k].AOACounts[n] + 1;
                                    if (a > TotalClusters_P[k].AOAMaxDiffs[n])
                                    {
                                        TotalClusters_P[k].AOAAOAMaxDiffs[n] =
PDWElement.AOA;
                                        TotalClusters_P[k].AOAMaxDiffs[n] = a;
                                    }
                                    found = true;
                                    break;
                                }
                            }
                            if (!found)
                            {
                                if (TotalClusters_P[k].AOALastIndex <
Constants.AOAArrayMaxIndex)
                                {
                                    TotalClusters_P[k].AOALastIndex++;
                                    u = TotalClusters_P[k].AOALastIndex;
                                }
                                else
```

```
                                {
                                    v = TotalClusters_P[k].AOACounts[0];
                                    for (n = 1; n <= Constants.AOAArrayMaxIndex;
n++)
                                    {
                                        if (TotalClusters_P[k].AOACounts[n] < v)
                                        {
                                            v = TotalClusters_P[k].AOACounts[n];
                                            u = n;
                                        }
                                    }
                                }
                                TotalClusters_P[k].AOAMeans[u] = PDWElement.AOA;
                                TotalClusters_P[k].AOATimes[u] = PDWElement.TOA;
                                TotalClusters_P[k].AOACounts[u] = 1;
                                TotalClusters_P[k].AOAAOAMaxDiffs[u] =
PDWElement.AOA;
                                TotalClusters_P[k].AOAMaxDiffs[u] = a;
                            }
                            TotalClusters_P[k].HasTempAOA = true;

                            if (a > TotalClusters_P[k].DiffAmp)
                            {
                                TotalClusters_P[k].DiffAmp = a;
                                TotalClusters_P[k].Tilt = PDWElement.Tilt;
                            }
                        }
                    }
                    if (TotalClusters_P[k].HasTempAOA)
                    {
                        a = TotalClusters_P[k].AOAMaxDiffs[0];
                        u = 0;
                        for (n = 1; n <= TotalClusters_P[k].AOALastIndex; n++)
                        {
                            if (a < TotalClusters_P[k].AOAMaxDiffs[n])
                            {
                                a = TotalClusters_P[k].AOAMaxDiffs[n];
                                u = n;
                            }
                        }
                        TotalClusters_P[k].AOATimes[0] =
TotalClusters_P[k].AOATimes[u];
                        TotalClusters_P[k].AOAMeans[0] =
TotalClusters_P[k].AOAMeans[u];
                        TotalClusters_P[k].AOACounts[0] =
TotalClusters_P[k].AOACounts[u];
                        TotalClusters_P[k].AOAAOAMaxDiffs[0] =
TotalClusters_P[k].AOAAOAMaxDiffs[u];
                        TotalClusters_P[k].AOAMaxDiffs[0] =
TotalClusters_P[k].AOAMaxDiffs[u];
                        TotalClusters_P[k].AOA =
TotalClusters_P[k].AOAAOAMaxDiffs[0];
                        TotalClusters_P[k].AOALastIndex = 0;

                        TotalClusters_P[k].TempAOATime =
TotalClusters_P[k].AOATimes[0];
                        TotalClusters_P[k].TempMaxDiff =
TotalClusters_P[k].AOAMaxDiffs[0];
                        TotalClusters_P[k].TempAOAMaxDiff =
TotalClusters_P[k].AOAAOAMaxDiffs[0];
                        TotalClusters_P[k].AOALastIndex = -1;
                    }
```

```
                    TotalClusters_P[k].MinTime =
Online_PDWProcess.PDWElements[Cluster_Members_Freq_P[t +
Cluster_Members_Freq_PW[TotalClusters_P[k].First_Index]] + First_Index].TOA;
                    TotalClusters_P[k].MaxTime = PDWElement.TOA;
                    TotalClusters_P[k].Duration = TotalClusters_P[k].MaxTime -
TotalClusters_P[k].MinTime;
                    TotalClusters_P[k].LobeCount = 0;

                    W_TotalClusters_P[k].PW = TotalClusters_P[k].PW;
// Just for Waterfall
                    W_TotalClusters_P[k].Freq = TotalClusters_P[k].Freq;
                    W_TotalClusters_P[k].AOA = TotalClusters_P[k].AOA;
                    W_TotalClusters_P[k].HasAOA = TotalClusters_P[k].HasTempAOA;
                    W_TotalClusters_P[k].Member = TotalClusters_P[k].Member;
                    W_TotalClusters_P[k].MaxTime = TotalClusters_P[k].MaxTime;
                    k++;
                }

                //////////
            }
        }
```

# فصل ۱۳– جمع‌بندی

در این مستند نرم‌افزار NRDF که یک نرم‌افزار برای کار و تعامل با یک سخت‌افزار جهت‌یاب راداری می‌باشد معرفی شده و نحوه پیاده سازی الگوریتم تشخیص اجسام در آن شرح داده شده است. مهمترین ویژگی این نرم‌افزار استفاده از الگوریتم خوشه بندی برای یافتن اجسام و نمایش اطلاعات دریافت شده از جهت‌یاب راداری توسط سامانه‌های اطلاعات جغرافیایی به صورت بر خط و بدون تاخیر می باشد. در مرحله پیاده‌سازی نیز محاسبه اطلاعات با استفاده از الگوریتم و نمایش این اطلاعات بدون تاخیر مشکل‌ترین قسمت این بخش بود؛ علت این مساله سرعت بالای ارسال اطلاعات از سخت‌افزار راداریاب (ارسال اطلاعات در هر ۲۰ میلی‌ثانیه) بود که در صورتی که از روش‌های معمولی برنامه‌نوبسی استفاده می‌شد استفاده از نرم‌افزار به سختی انجام می‌شد. برای حل این مشکل از netframework ٤.٠. و زبان برنامه نویسی C# روش‌های برنامه‌نویسی چند رسیمانی[۱] و الگوریتم هوشمند خوشه بندی استفاده شده است.

دقت شود که به علت آنکه اطلاعات بسیاری از سمت راداریاب دریافت می شد، می بایست الگوریتم طراحی شده قالیت فیلتر کردن اطلاعات نامفید[۲] را داشته باشد که در الگوریتم گفته شده این قابلیت دیده شده است، همچنین می بایست الگوریتم بسیار سریع باشد که با استفاده از تعاریف داده کاوی و همچنین بهینه طراحی و پیاده سازی شدن الگوریتم به این امر دست یافتیم.

از آنجایی که سخت‌افزارهای مختلفی به عنوان جهت‌یاب راداری وجود دارد که هر کدام از روش ارتباطی منحصر به فردی استفاده می‌کنند و همچنین از آنجایی که امکان توسعه الگوریتم‌های پردازشی و خوشه‌بندی جدیدی در آینده وجود داشت، معماری نرم‌افزار NRDF به شکلی طراحی شده است که به راحتی قابل توسعه است و به راحتی می‌توان برای سخت‌افزارها و الگوریتم‌های جدید اجزای جدیدی تولید و به نرم‌افزار متصل کرد.

---

[۱] Multi-Thread

[۲] Noise

# فصل ۱۴– مراجع


[۱] *How Radar Works*, www.ig.utexas.edu/research/projects/mars/education/

٦- radar_works.htm

[۲] Merrill Skolnik, *Radar Handbook*, McGraw-Hill Professional, ۱۹۹۰

[۳] *Radar*, en.wikipedia.org/wiki/Radar

[٤] Michael N. DeMers, *GIS for Dummies*, John Wiley & Sons, ۲۰۰۹

[٥] *What is ArcGIS*, ESRI, ۲۰۰۸

[٦] *Google Maps*, maps.google.com

[۷] *Microsoft MSDN Library*, msdn.microsoft.com/en-us/library/ms۱۲۳٤۰۱.aspx

[۸] *GMap.NET*, greatmaps.codeplex.com