دانشگاه شاهرود

دانشکده: مهندسی برق و رباتیک

گروه: الکترونیک

پایان‌نامه جهت اخذ درجه کارشناسی ارشد در رشته مهندسی برق – الکترونیک

بازشناسی اشیاء با استفاده از ویژگی‌های محلی جهت درک ربات از محیط و پیاده سازی سخت‌افزاری روی پردازنده دیجیتال TMS320DM6446

جواد جوکار

اساتید راهنما:

دکتر علیرضا احمدی فرد

دکتر حسین مروی

بهمن ۱۳۹۳

شماره : ۵۶/۱۲/آ.ت.ب
تاریخ : ۹۳/۱۱/۲۹
ویرایش : ------

بسمه تعالی

مدیریت تحصیلات تکمیلی
فرم شماره (۶)

فرم صورتجلسه دفاع پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) جلسه دفاع از پایان نامه کارشناسی ارشد خانم / آقای :

جواد جوکار رشته : **برق - الکترونیک** گرایش : **دیجیتال**

تحت عنوان : **بازشناسی اشیاء با استفاده از ویژگی های محلی جهت درک ربات از محیط**

که در تاریخ **۹۳/۱۱/۲۹** با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح زیر است :

قبول (با درجه : بسیار خوب امتیاز ۱۸٫۸۷۵) ☑ دفاع مجدد ☐ مردود ☐

۱ـ عالی (۲۰ ـ ۱۹) ۲ـ بسیار خوب (۱۸/۹۹ ـ ۱۸)

۳ـ خوب (۱۷/۹۹ ـ ۱۶) ۴ـ قابل قبول (۱۵/۹۹ ـ ۱۴)

۵- نمره کمتر از ۱۴ غیر قابل قبول

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضـــاء
۱ـ استاد راهنما	۱- علیرضا [illegible]	استادیار	
	۲- حسین [illegible]	استادیار	
۲ـ استاد مشاور	ــــ	ــــ	ــــ
۳ـ نماینده شورای تحصیلات تکمیلی	[illegible]	استادیار	
۴ـ استاد ممتحن	[illegible]	استادیار	
۵ ـ استاد ممتحن	[illegible]	استادیار	

رئیس دانشکده :

تقدیم به:

پدر و مادر عزیزم

تشکر و قدردانی

نخستین سپاس و ستایش از آن خداوندی است که بنده کوچکش را در دریای بیکران اندیشه ، قطره‌ای ساخت تا وسعت آن را از دریچه اندیشه‌های ناب آموزگارانی بزرگ به تماشا نشیند . لذا اکنون که در سایه سار بنده‌نوازی‌هایش پایان‌نامه حاضر به انجام رسیده است، بر خود لازم می‌دانم مراتب سپاس و قدردانی را از بزرگوارانی به جا آورم که اگر دست یاریگرشان نبود، هرگز این پایان‌نامه به انجام نمی‌رسید.

در ابتدا از استاد عزیزم، جناب آقای دکتر **علیرضا احمدی‌فرد**، که دلسوزانه مرا در انجام این پروژه یاری کردند و از راهنمایی‌ها و تجربیاتشان بهره‌های فراوان بردم تشکر ویژه دارم.

همچنین از زحمات جناب آقای **دکتر حسین مروی** و تمام اساتیدی که از محضر درس ایشان بهره جسته‌ام تشکر و قدردانی می‌کنم.

سپاس آخر را به مهربانترین همراهان زندگیم، به **پدر و مادر عزیزم** تقدیم می‌کنم که حضورشان در فضای زندگیم مصداق بی ریای سخاوت بوده است.

تعهدنامه

اینجانب **جواد جوکار** دانشجوی کارشناسی ارشد رشته **الکترونیک دیجیتال** دانشکده **برق و رباتیک دانشگاه شاهرود**، نویسنده پایان‌نامه با عنوان:

«بازشناسی اشیاء با استفاده از ویژگی‌های محلی جهت درک ربات از محیط و پیاده‌سازی سخت‌افزاری روی پردازنده دیجیتال TMS320DM6446 »،

تحت راهنمایی **دکتر علیرضا احمدی‌فرد** و **دکتر حسین مروی** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های دیگر پژوهش‌گران، به مرجع مورد استفاده استناد شده است.
- مطالب این پایان‌نامه تا کنون توسط خود، یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- حقوق معنوی این اثر، به دانشگاه شاهرود تعلق دارد و مقالات مستخرج با نام "دانشگاه شاهرود" یا "Shahrood University" به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آوردن نتایج اصلی پایان‌نامه تأثیرگذار بوده‌اند، در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است، ضوابط و اصول اخلاقی رعایت شده است.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی دسترسی یافته (یا استفاده) شده است، اصل رازداری و اصول اخلاق انسانی رعایت شده است.

جواد جوکار

۱۳۹۳

مالکیت نتایج و حق نشر

- تمام حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه شاهرود می‌باشد. این مطلب باید به نحو مقتضی، در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان‌نامه بدون ذکر منبع مجاز نمی‌باشد.

چکیده

در این پایان‌نامه مسأله شناسایی اشیاء از پیش تعیین شده در یک تصویر از صحنه، مورد مطالعه قرار می‌گیرد. این مسأله کاربرد فراوانی در بینایی ربات دارد. از آنجا که برای یک ربات مسأله بلادرنگ بودن الگوریتم یک امر مهم است، روش ارائه شده باید دارای سرعت خوبی بوده و همچنین دقت آن هم مناسب باشد. با بررسی روش‌های موجود از توصیفگر SURF برای بازنمایی تصویر شیء استفاده شد. این روش هم دقت خوب و هم سرعت مناسبی دارد. برای انطباق توصیفگرهای استخراج شده فاصله اقلیدسی بین زوج توصیفگرهای صحنه و مدل اشیاء محاسبه می‌گردد. این روش به دلیل پیچیدگی محاسباتی کم پیشنهاد گردید. در مرحله تطبیق یک سری داده‌های پرت بوجود می‌آیند که برای حذف آن‌ها از روش RANSAC استفاده شده است.

همچنین برای پیاده‌سازی الگوریتم روی ربات باید از یک سخت‌افزار استفاده کنیم که قابل‌حمل بوده و همچنین سرعت اجرای مناسبی داشته باشد. به همین منظور ما از پردازندهِی DSP شرکت TI سری داوینچی مدل DM6446 بهره برده‌ایم. ما در این پایان‌نامه الگوریتم پیشنهادی را بر روی سخت‌افزار اشاره شده پیاده‌سازی نمودیم.

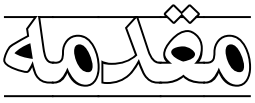
کلمات کلیدی: شناسایی اشیاء، توصیفگر SURF، عملگر RANSAC، پردازشگرهای سیگنال دیجیتال، DM6446

فهرست

فهرست شکل‌ها

فهرست جدول‌ها

فصل اول

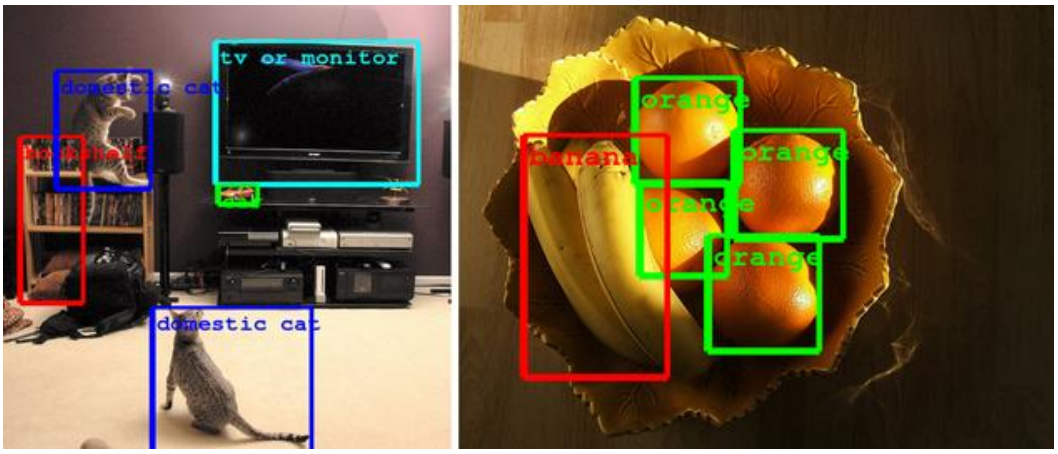


۱-۱) مقدمه

مسائل متنوعی در بینایی کامپیوتر مورد تحقیق و بررسی قرار گرفته است که حل هر کدام از آن‌ها می‌تواند تحول بزرگی در تعامل میان انسان و ماشین ایجاد کند. یکی از مسائل مطرح در حوزه بینایی کامپیوتر تشخیص اشیاء یا طبقه‌بندی آن‌ها است. هدف از طبقه‌بندی اشیاء شناسایی و یافتن نمونه‌های یک شیء در داخل تصاویر دنیای واقعی است.

همان طوری که ما در طول زندگی خود تجربه کرده‌ایم، سیستم بصری انسان دارای توانایی فوق‌العاده‌ای برای تشخیص اشیاء می‌باشد. انسان انواع مختلفی از شیءها را می‌شناسد و می‌تواند آن‌ها را در حالت‌های مختلف از جمله زوایای دید گوناگون بدون زحمت تشخیص دهد.

پیدا کردن اشیای معمولی در محیط برای انسان بسیار ساده بوده، اما برای ربات هنوز یک موضوع در حال مطالعه و تحقیق است. از آنجا که انسان‌ها در برقراری ارتباط از رؤیت و شناسایی اشیاء استفاده می‌کنند، کاربردهایی از ربات که به ارتباط انسان و ربات نیاز دارند نیز می‌توانند از قابلیت یافتن شیء استفاده فراوانی ببرند. شکل (۱-۱) یک نمونه از تشخیص اشیاء در دو تصویر را نشان می‌دهد.



شکل (۱-۱) : بازشناسی اشیاء در تصویر

۱-۲) تعریف شیء و الفاظ مرتبط با شناسایی اشیاء

اغلب الفاظ و کلماتی در حوزه بازشناسی اشیاء وارد می‌شود که تعریف مناسبی برای آن‌ها وجود ندارد و در مواردی باعث ابهام می‌شود، مشهورترین این اصطلاحات عبارت‌اند از [۱]:

- ✓ **تشخیص**[۱]: آیا یک شیء خاص در تصویر وجود دارد؟
- ✓ **مکان‌یابی**[۲]: تشخیص بعلاوه مشخص کردن محل دقیق یک شیء خاص
- ✓ **بازشناسی**[۳]: مکان‌یابی تمامی اشیاء حاضر در یک صحنه خاص
- ✓ **فهمیدن یا تحلیل**[۴]: بازشناسی و بررسی نقش هر شیء با توجه به محیط اطراف

گاهی اوقات تعریف شیء ابهام برانگیز است. در واقع یک شیء بر اساس کاربرد تعریف می‌شود. برای مثال تعریف شیء در بازشناسی کلمات دست‌نویس با تعریف شیء زمانی که هدف بازشناسی کروموزوم‌ها در تحقیقات پزشکی است متفاوت می‌باشد. به عنوان یک تعریف ساده می‌توان یک شیء را مجموعه‌ای از قالب‌های معین دانست که در شرایط مشخص می‌تواند حس شود [۱].

۱-۳) اجزا مهم یک سیستم بازشناسی

بازشناسی شامل دو قسمت مهم است: بازنمایی و تطبیق[۵]. اکثر روش‌های مورد استفاده برای بازنمایی را می‌توان بر اساس نوع ویژگی‌ها به دو بخش تقسیم‌بندی کرد. این دو بخش عبارت‌اند از ویژگی‌هایی که از بخش‌بندی کردن تصویر و استفاده از تکه‌های بدست آمده حاصل می‌شوند و ویژگی‌هایی که از

---

۱ Detection
۲ Localization
۳ Recognition
۴ Understanding
۵ Matching

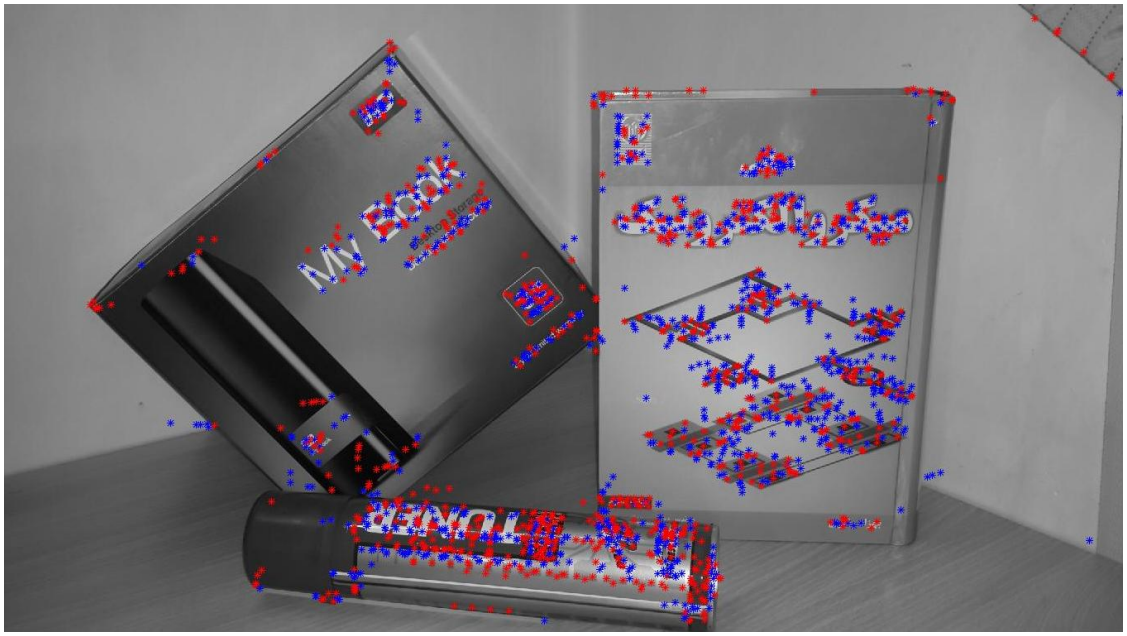
توصیفگرها استفاده می‌کنند. البته در برخی از پژوهش‌ها به صورت ترکیبی از این ویژگی‌ها برای تشخیص اشیاء استفاده شده است [۲].

استفاده ترکیبی از هر دو ویژگی در آینده بسیار شایع خواهد شد زیرا می‌توان با مقایسه قرار دادن آن‌ها در موقعیت‌های مختلف بهترین تصمیم را اتخاذ کرد که البته بار محاسباتی زیادی را طلب می‌کند. یک روش خوب روشی است که از هر دو نوع ویژگی به طور مؤثر استفاده کند و قابل پیاده‌سازی و استخراج در یک سیستم بلادرنگ[۱] باشد. از این رو مدیریت منابع امری مهم در به کارگیری سامانه‌های تشخیص اشیاء می‌باشد. با توجه به میزان کارایی و دقت توصیفگرها در این پایان‌نامه از این نوع ویژگی‌ها استفاده شده است.

توصیفگرها تصویر را با استفاده از یک سری نقاط کلیدی[۲] به صورت یکتا بازنمایی می‌کنند. این نقاط مربوط به یک تصویر را می‌توانید در شکل (۱-۲) ببینید. از انواع توصیفگرها می‌توان به لبه‌ها، گوشه‌ها و تقاطع‌ها نام برد که هر کدام دارای کاربردهای خاص خود هستند.

[۱] Real-time

[۲] Interest points



شکل (۱-۲) : پیدا کردن نقاط کلیدی در تصویر

از یک دیدگاه توصیفگرها به دو قسمت سراسری و محلی تقسیم می‌شوند. در حالی که توصیفگرهای سراسری سعی دارند که کل تصویر را به صورت یکپارچه توصیف کنند، توصیفگرهای محلی، سعی دارند تا نواحی مختلف تصویر را به صورت محلی توصیف کنند. به دلیل پیچیدگی‌هایی که تصویر شامل شیء (اشیاء) دارد و همچنین عدم توزیع یکنواخت این پیچیدگی‌ها در کل تصویر، توصیفگرهای محلی ابزار مناسب‌تری برای بازنمایی هستند.

تا کنون توصیفگرهای بسیاری طراحی و معرفی شده است که هر کدام دارای معایب و مزایایی هستند. با توجه به تغییراتی که می‌تواند در یک شیء اتفاق بیفتد، هر توصیفگر باید بتواند در برابر این تغییرات مقاوم باشد. یکی از توصیفگرهای قدرتمند، توصیفگر SIFT است [۳]. این توصیفگر هر تصویر را با استفاده از تعدادی نقاط کلیدی توصیف می‌کند. هر نقطه کلیدی دارای یک بردار با طول ۱۲۸ است. از این بردار برای توصیف نقاط کلیدی به صورت مجزا استفاده می‌شود. این توصیفگر همچنین برای هر نقطه کلیدی یک جهت و یک مقیاس نیز در نظر می‌گیرد.

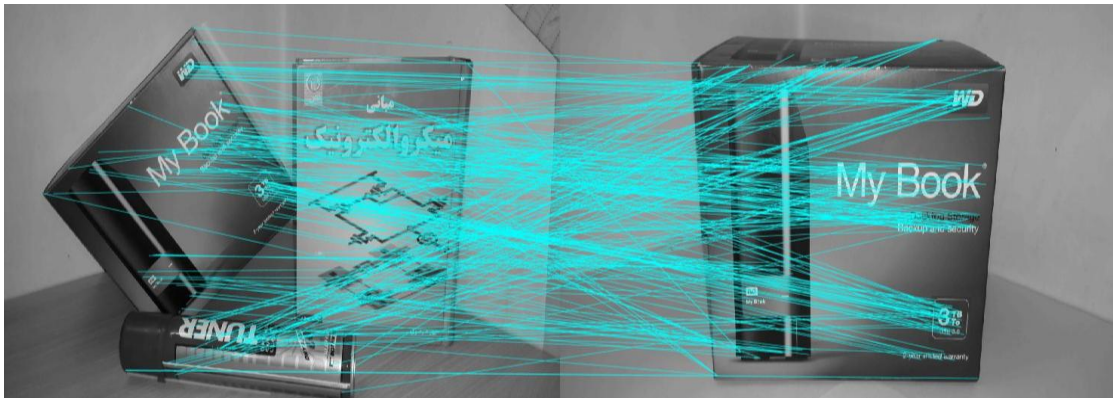
نویسندگان نوعی دیگر SIFT را ارائه کرده‌اند که GLOH نامیده می‌شود. از لحاظ عملی این توصیفگر با همان تعداد ابعاد قدرت تمایز بیشتری دارد ولی بار محاسباتی آن از SIFT بیشتر است [۴].

SIFT هنوز هم جذاب‌ترین توصیف گر برای استفاده‌های عملی به نظر می‌رسد و در حال حاضر به طور گسترده‌ای مورد استفاده قرار می‌گیرد [۴]. ولی با این حال زمان نسبتاً بالای محاسبات و همچنین ابعاد بالای توصیفگر در مرحله تطبیق از اشکالات SIFT است که باعث می‌شود برای کاربردهای بلادرنگ خیلی مناسب نباشد.

لذا برای بهبود پیچیدگی محاسباتی نسخه‌های سریع‌تری معرفی شدند. از مهم‌ترین آن‌ها می‌توان به SURF اشاره کرد که طول بردار در این توصیفگر ۶۴ است. SURF یک آشکارساز قوی از ویژگی‌های محلی می‌باشد که اولین بار توسط Herbert Bay و همکارانش در سال ۲۰۰۶ ارائه شده است [۴]. SURF تا حدودی از توصیفگر SIFT الهام گرفته و همچنین از ماتریس هسین در پیدا کردن نقاط کلیدی بهره می‌گیرد.

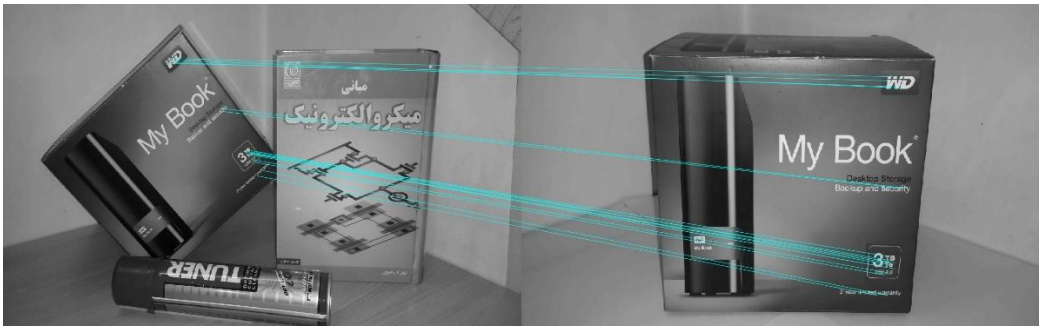
با توجه به اینکه توصیفگر SURF از لحاظ دقت و مقاوم بودن در برابر شرایط نامساعد جزء بهترین توصیفگرها بوده و همچنین دارای حجم محاسباتی کمتر و به تبع سرعت بیشتری نسبت به SIFT می‌باشد، در این پایان‌نامه از این نوع توصیفگر برای توصیف و بازنمایی استفاده شده است.

پس از بازنمایی یک شیء نوبت به تطبیق می‌رسد. روش‌های مختلفی برای این کار وجود دارد از جمله استفاده از گراف، فاصله اقلیدسی و استفاده از انواع طبقه‌بندها. با توجه به اینکه ویژگی‌هایی که توصیفگر SURF استخراج می‌کند قابلیت تمایز زیادی دارند لذا استفاده از روش‌هایی که پیچیدگی محاسباتی کمتری دارند بیشتر توصیه می‌شود، از این رو در این پایان‌نامه از روش فاصله اقلیدسی برای مرحله تطبیق استفاده شده است(شکل ۱-۳).



شکل (۱-۳) : تطبیق ویژگی‌ها

در مرحله تطبیق یک سری داده‌های پرت[۱] بوجود می‌آیند. این داده‌ها از آنجا ناشی می‌شوند که یک نقطه کلیدی در یک تصویر به نقطه‌ای به غیر از نقطه کلیدی متناظر با خودش در تصویر دیگر نسبت داده شود. برای حذف این داده‌ها روش‌های مختلفی ارائه شده که ما در این پایان‌نامه از عملگر RANSAC استفاده می‌کنیم (شکل ۱-۴).

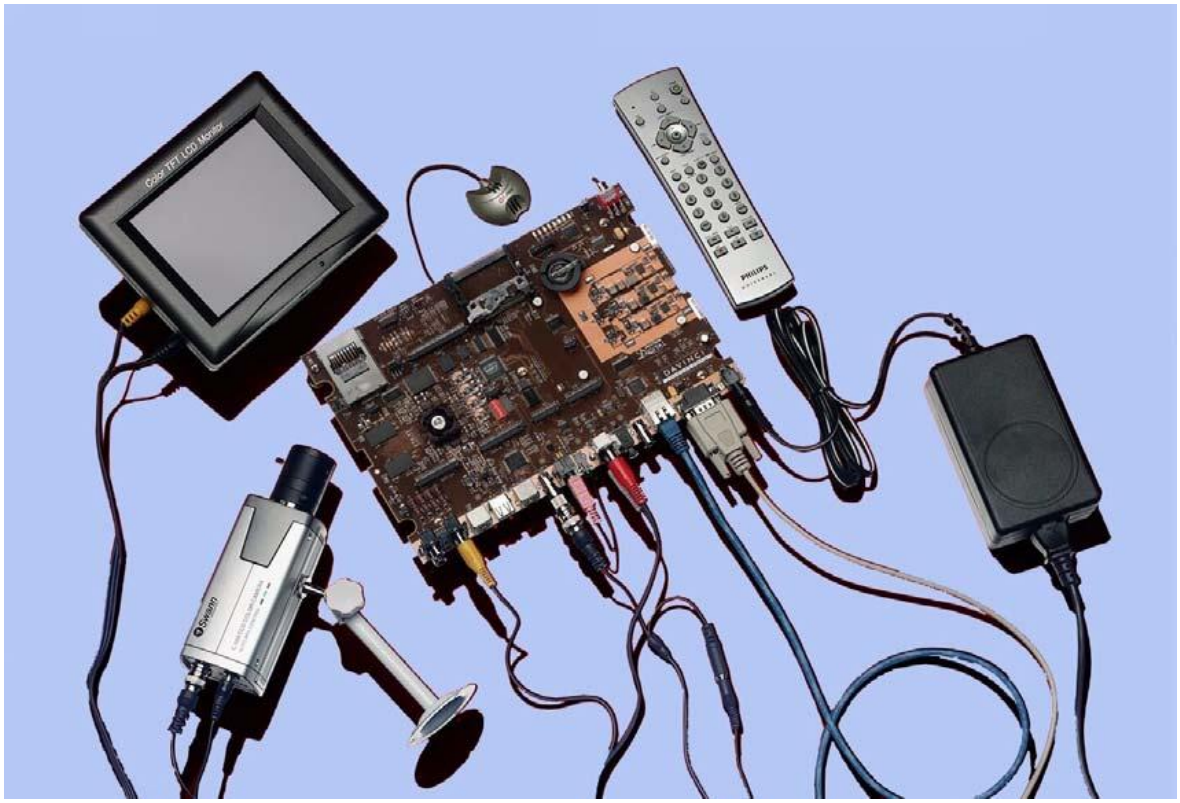


شکل (۱-۴) : حذف داده‌های پرت با استفاده از RANSAC

از آنجا که هدف این پایان‌نامه از شناسایی اشیاء استفاده در کاربردهای رباتیک می‌باشد لذا الگوریتم مورد نظر باید توانایی اجرای بلادرنگ و همچنین سخت‌افزار مربوطه امکان قابل‌حمل[۲] بودن را داشته باشد. برای حل محدودیت اول همان طور که اشاره شد از روش‌هایی استفاده شده که پیچیدگی

[۱] Outlier data
[۲] Portable

محاسباتی کمتری داشته و همچنین از دقت قابل قبولی برخوردار باشند و برای مورد دوم از پردازنده‌های سیگنال دیجیتال یا DSP[1] سری داوینچی مدل DM6446 که یک پردازنده دو هسته‌ای (یک هسته ARM و یک هسته DSP) می‌باشد استفاده شده است (برد این پردازشگر به همراه ادوات جانبی آن در شکل (۱-۵) نشان داده شده است) که این قابلیت را دارند که بتوان آن‌ها را به صورت مستقل[2] و بدون هیچ سخت‌افزار کمکی راه‌اندازی کرد.



شکل (۱-۵) : برد DSP داوینچی مدل DM6446

[1] DSP (Digital Signal Processor)

[2] Stand-alone

فصل دوم

مروری بر کارهای انجام شده

۲-۱) مقدمه

همان‌طور که گفته شد مهم‌ترین بخش هر سیستم بینایی، بخش بازشناسی اشیاء است. با توجه به اینکه کارهای بسیار زیادی در این حوزه انجام شده یا در حال انجام است، در این فصل منابع به دو دسته تقسیم شده و سپس در هر دسته مهم‌ترین و مرتبط‌ترین کارهای انجام شده مرور خواهد شد.

این دو دسته عبارت‌اند از:

- سیستم‌های مرسوم بازشناسی اشیاء[۱]
- سیستم‌های بازشناسی اشیاء که از بینایی انسان الگو می‌گیرند و سعی می‌کنند مانند انسان عمل کنند.[۲]

۲-۲) سیستم‌های مرسوم بازشناسی اشیاء

در این دسته بازشناسی را می‌توان در حالت کلی در دو مرحله انجام داد [۵]:

- بازنمایی اشیاء[۳]
- تطبیق

در مسأله بازنمایی باید بتوان یک شیء مفروض را به بهترین نحو بازنمایی کرد؛ به عبارت دیگر باید بتوان یک شیء را به فضایی دیگر نگاشت[۴] داد. این فضا باید در مقابل تغییراتی که اتفاق می‌افتد مقاوم[۵] باشد.

---

[۱] Traditional object recognition systems
[۲] Cognitive object recognition systems
[۳] Object representation
[۴] Mapping
[۵] Robust

۲-۲-۱) بازنمایی اشیاء

به صورت کلی در دسته‌بندی روش‌های بازنمایی توافق جامعی وجود ندارد و منابع مختلف این دسته‌بندی‌ها را به صورت مختلفی انجام می‌دهند.

در یک نگاه جامع می‌توان دو دسته برای بازنمایی یک شیء در نظر گرفت [۶]. در حالی که دسته اول سعی دارند که یک شیء را با استفاده از اطلاعاتی نظیر لبه، مرزها، گوشه‌ها و نقاط تقاطع نشان دهند، دسته دوم از روشنایی و رنگ برای بازنمایی استفاده می‌کنند. جدا از دسته‌بندی بالا از دیدگاه کاربردی‌تر می‌توان روش‌های متفاوتی برای بازنمایی یک شیء در نظر گرفت، از جمله [۶]:

- ✓ روش‌های مبتنی بر هیستوگرام
- ✓ روش‌های مبتنی بر فضای ویژه[۱]
- ✓ روش‌های مبتنی بر گوشه‌ها و لبه‌ها
- ✓ روش‌های مبتنی بر گراف[۲]

در میان دسته‌بندی‌های مبتنی بر هیستوگرام می‌توان به کار Swain [۷] اشاره کرد که از رنگ به عنوان یک ویژگی اصلی برای بازشناسی استفاده کرده است. همچنین می‌توان از کار Stricker [۸] که از هیستوگرام مرز شیء برای بازشناسی استفاده کرده است، نام برد. از آنجا که روش‌های مبتنی بر هیستوگرام ساده، دارای سرعت مناسب و در برابر تغییرات مقاوم هستند، روش‌های جالبی هستند اما در عین حال ضعف‌هایی دارند [۶]، از جمله عیب‌های بزرگ این روش‌ها، ناتوانی آن‌ها در مدل کردن ساختار و شکل شیء و تمرکز بر روی ویژگی‌های رنگ است.

---

[۱] Eigensapce-based

[۲] Graph-based

روش‌های مبتنی بر فضای ویژه، کل تصویر را به صورت یک بردار در نظر می‌گیرند و سعی می‌کنند با استفاده از روش‌های کاهش بعد مانند آنالیز مؤلفه‌های اصلی[۱]، بعد آن‌ها را کاهش دهند. از کارهای انجام شده در این دسته می‌توان به کار Turk [۹] اشاره کرد. این روش‌ها زمانی کارایی مناسبی دارند که برای مثال تمام پس‌زمینه دارای یک توزیع یکنواخت قابل‌شناسایی باشد [۶]. روش‌های موجود در این دسته زمانی که یک شیء کل تصویر را در بر می‌گیرد بسیار مناسب است [۱۰].

در روش‌های مبتنی بر گراف تصویر به نواحی تشکیل‌دهنده آن بخش می‌شود و هر ناحیه یک نود از گراف را تشکیل می‌دهد. سپس بین هر ناحیه و نواحی مجاور یک ارتباط[۲] در نظر گرفته می‌شود [۵]. از مزیت‌های روش‌های مبتنی بر گراف این است که ویژگی‌های ساختاری را می‌توان به راحتی به صورت گراف در نظر گرفت و سپس از یک الگوریتم تطبیق برای تطبیق زیر گراف‌ها استفاده کرد [۱۱].

روش‌های مبتنی بر ظاهر سعی می‌کنند یکی شیء را با استفاده از تصاویر آن مدل کنند به عنوان مثال در [۱۲] سعی می‌شود که یک شیء با استفاده از تعداد زیادی تصویر که از زوایای مختلف گرفته شده‌اند مدل شود (همانند آنچه در شکل (۲-۱) نشان داده شده است). سپس با استفاده از ارتباط بین نماهای تصویر شیء مدل فشرده‌ای از آن در فضای ویژه بدست آورد. از معایب این روش می‌توان به حساسیت آن‌ها نسبت به داده‌های پرت اشاره کرد. در تلاش برای ترکیب مزایای روش‌های مبتنی بر ویژگی و روش‌های مبتنی بر ظاهر روش‌های مبتنی بر توصیفگرها[۳] ارائه شد [۵].

---

[۱] Principle Component Analysis(PCA)

[۲] Edge

[۳] Descriptor

شکل (۲-۱) : یک مجموعه از تصاویر مربوط به یک شیء که از چرخش شیء مورد نظر حول یک محور بدست آمده است [۱۲].

برخلاف روش‌های مبتنی بر ویژگی، توصیفگر محلی یک توصیف یکتا[۱] از تصویر ارائه می‌دهد. از مزایای مهم روش‌های مبتنی بر توصیفگرهای محلی این است که می‌توان با استفاده تنها از این توصیفگرها به مقایسه دو تصویر از یک شیء پرداخت. علاوه بر آن از آنجا که این توصیفگرها نسبت به تغییرات هندسی مقاوم هستند برخلاف روش‌های مبتنی بر ظاهر به تعداد نماهای کمتری برای بیان یک شیء نیاز دارند [۵].

توصیفگرها را می‌توان به دو بخش سراسری[۲] و محلی تقسیم کرد. از آنجا که توصیفگرهای سراسری سعی می‌کنند تمام تصویر را به صورت ساده‌ای توصیف کنند در بسیاری از اوقات نمی‌توانند جزئیات تصویر را مشخص کنند. این جزئیات می‌تواند شامل تغییرات هندسی باشد که در بخشی از تصویر اتفاق می‌افتد [۶] [۱۳].

[۱] Unique
[۲] Global

با توجه به قدرت توصیفگرهای محلی در توصیف کامل تصویر، سعی می‌شود مرور کوتاهی به مهم‌ترین آن‌ها شود. توصیفگرهای بسیار زیادی ارائه شده است که هر کدام ویژگی‌های خاصی را مدنظر قرار می‌دهند. در [۱۴] تقسیم‌بندی کلی روی توصیفگرهای محلی می‌شود که این تقسیم‌بندی را می‌توان به صورت زیر خلاصه کرد:

- ✓ توصیفگرهای مبتنی بر توزیع[۱]
- ✓ تکنیک‌های مکانی - فرکانسی[۲]
- ✓ توصیفگرهای تفاضلی[۳]

توصیفگرهای مبتنی بر توزیع سعی می‌کنند با استفاده از هیستوگرام ویژگی‌های مختلف ظاهر یا شکل یک شیء را بیان کنند [۱۴]. یک توصیفگر ساده در این دسته، همان پیکسل‌های خود تصویر می‌باشد. توصیفگر بعدی که در این خانواده قرار دارد و شهرت بسیار دارد توصیفگر ارائه شده توسط Lowe یعنی همان SIFT می‌باشد [۳]. خانواده دوم از توصیفگرها، توصیفگرهای مکانی-فرکانسی است. این تکنیک‌ها سعی می‌کنند که محتوای تصویر را با استفاده از فرکانس‌ها بیان کنند [۱۴]. برای مثال، تبدیل فوریه[۴] محتوای یک تصویر را به توابع اولیه تجزیه می‌کند؛ اما از آنجا که ارتباط مکانی بین نقاط از بین می‌رود و همچنین تعداد این توابع اولیه بی‌نهایت هستند برای توصیف محلی مناسب نیستند [۱۴]. تبدیل گابور[۵] [۱۴] مشکل فوق را حل می‌کند اما مشکل دیگری دارد و آن این است که برای یک تغییر در فرکانس و جهت باید تعداد بسیار زیادی فیلتر محاسبه شود. این کار باعث رشد فزاینده پیچیدگی محاسباتی می‌شود. فیلتر گابور و تبدیلات موجک [۱۴] برای کلاسه‌بندی بافت[۶] مورد استفاده قرار می‌گیرد.

---

[۱] Distribution-Based Descriptors
[۲] Spatial-Frequency Techniques
[۳] Differential Descriptors
[۴] Fourier Transform
[۵] Gabor wavelet
[۶] Texture

در دسته سوم سعی می‌شود تخمینی از همسایه‌های یک نقطه یا پیکسل محاسبه شود [۱۴]. از مهم‌ترین کارهای اخیر در این دسته، الگوهای باینری محلی[۱] [۱۵] است. این روش در عین حال که ساده است ولی کاربردهای بسیار زیادی مخصوصاً در آنالیز بافت دارد. این توصیفگر یک عملگر مربعی ۳×۳ را در نظر گرفته و روی تصویر شیء حرکت می‌دهد. این مربع روی هر پیکسلی که قرار می‌گیرد ۸ پیکسل همسایه با پیکسل مرکزی مقایسه می‌شوند و اگر مقدار شدت روشنایی آن‌ها از پیکسل همسایه بیشتر بود مقدار ۱ و در غیر این صورت مقدار صفر را می‌گیرند. در نهایت مقدار شدت روشنایی پیکسل مرکزی، جمع وزن‌دار مقادیر پیکسل‌های همسایه است. سپس مربع را حرکت داده و روی پیکسل مجاور قرار می‌گیرد. این کار تا انتها ادامه می‌یابد. هیستوگرام مقادیر، نتیجه اعمال الگوی دودویی محلی می‌باشد. از این توصیفگر در کاربردهای بسیار استفاده شده است.

در مقایسه‌ای که میان روش‌های مختلف توصیف ویژگی انجام شده است [۱۴]، تبدیل ویژگی غیر حساس به مقیاس (SIFT) [۳]، بارزترین توصیف را از شیء ارائه داده است. توصیفگر SIFT روی نقاط کلیدی غیر حساس که از طریق تفاوت‌های گاوسی[۲] (DOG) استخراج شده است، تکیه می‌کند. نسخه‌های مختلفی از توصیفگر SIFT ارائه شده است که در هر کدام سعی شده تا یکی از ویژگی‌های آن تقویت شود. در برخی از موارد هم ویژگی‌های جدید مانند رنگ به آن اضافه شده است. این توصیفگر دارای دقت مناسبی می‌باشد، اما تعداد نقاط ویژگی استخراج شده در آن بالاست، که محاسبات بیشتری را در پی خواهد داشت. این مسأله در تصاویری که دارای پیچیدگی بیشتری هستند، نمود بیشتری دارد. در همین راستا روش PCA-SIFT [۱۶] جهت کاهش ابعاد بردار ویژگی SIFT پایه از ۱۲۸ به ۳۶ ارائه شد. این روش باعث کاهش پیچیدگی زمانی می‌شود اما انعطاف‌پذیری زیادی ندارد. در مواقعی که ساختار بافتی مشابهی در تصاویر اتفاق افتاده است، SIFT خصوصیات محتویات شکل را در جهت بالا بردن توان تمایز، به بردار ویژگی قبلی اضافه می‌کند [۱۷].

---

[۱] Local Binary Pattern (LBP)

[۲] Difference of Gaussian

در سال ۲۰۰۹ میچل و یو توصیفگر ASIFT[۱] [۱۸] را ارائه دادند. این توصیفگر علاوه بر داشتن تمام ویژگی‌های SIFT قدرت بسیار بالایی در تصاویری که تبدیل کشیدگی[۲] بر روی آن اعمال شده است، داشت. در توصیفگر SIFT-CCH [۱۹]، از ماتریس هم‌رخداد[۳] [۲۰] در جهت شرکت دادن ویژگی رنگی در توصیفگر SIFT استفاده شده است. ماتریس هم‌رخداد اولیه بر روی تصاویر سطح خاکستری کاربرد داشت، ولی نسخه‌های دیگری از آن روی تصاویر رنگی هم قابل استفاده می‌باشد. این روش به دلیل ترکیب دو ویژگی در یک بردار ویژگی به SIFT-CCH معروف است.

در سال ۲۰۰۶ یک توصیفگر ویژگی غیر حساس به مقیاس دیگر تحت عنوان SURF [۴] ارائه شد. توصیفگر SURF نیز همانند SIFT نقاط کلیدی را استخراج می‌نماید با این تفاوت که هر نقطه توسط یک بردار ویژگی با ۶۴ مؤلفه نمایش داده می‌شود. بر اساس ارزیابی‌های انجام شده، به نتیجه رسیده‌اند که SIFT قدرت تشخیص بیشتری نسبت به SURF دارد ولی در مقابل SURF سریع‌تر و منسجم‌تر عمل می‌نماید. به منظور مطالعه بیشتر در مورد توصیفگرهای ویژگی غیر حساس به مقیاس می‌توانید به مراجع [۴]، [۱۴] و [۲۱] مراجعه نمایید.

۲-۲-۲) تطبیق

همان‌طور که در ابتدای این فصل بیان شد پس از بیان مناسب صحنه و مدل، فاز بعدی تطبیق مناسب بین این دو صحنه می‌باشد. هدف از تطبیق بررسی میزان شباهت[۴] بین دو بردار ویژگی استخراج شده از مدل و صحنه است. وقتی صحبت از میزان شباهت یا معیار شباهت می‌شود خود به خود بحث میزان دوری یا نزدیکی دو بردار تداعی می‌شود. می‌توان گفت که کارایی یک سیستم بازشناسی وابسته به معیار شباهت است [۲۲]. در سال‌های اخیر تلاش‌های بسیار زیادی برای طراحی یک معیار شباهت

---

[۱] Affine-SIFT

[۲] Affine transformation

[۳] Co-occurrence matrix

[۴] Similarity

مناسب مطرح شده است [۲۳]- [۲۴]. برخی از این معیارهای فاصله عبارت‌اند از: فاصله اقلیدسی، هاسدورف [۲۵]، فاصله هدف[۱] [۲۶]، تقاطع هیستوگرام[۲]. در میان این معیارها فاصله اقلیدسی بسیار ساده و در عین حال پر استفاده است. فاصله هدف در محاسبه ارقام دست‌نویس بسیار کاربرد دارد [۲۲]، این نوع معیار شباهت به تغییر شکل‌های محلی حساس است. [۲۲] فاصله هاسدورف[۳] نه تنها به نویز مقاوم است بلکه این امکان را نیز می‌دهد تا بخشی از یک تصویر را با تصویر دیگر مقایسه شود. البته در تمامی این معیارهای شباهت فرض بر این است که تصویر به درستی توصیف شده است. یکی دیگر از روش‌های بازنمایی اشیاء استفاده از روش‌های مبتنی بر گراف است. این روش علی‌رغم اینکه دقت خوبی دارد ولی حجم محاسباتی آن به نسبت بیشتر می‌باشد.

پس از اینکه با دو بخش بسیار مهم سیستم‌های بازشناسی اشیاء و کارهای مهم در هر بخش به صورت مجزا آشنا شدیم. در این قسمت از مهم‌ترین و مرتبط‌ترین سیستم‌هایی که تا کنون انجام شده است چند نمونه معرفی می‌شود.

در سال ۲۰۰۵ Kostin و همکارانش [۱۱] برای اینکه مشکلات بخش‌بندی را از بین ببرند و از طرفی بتوانند ناحیه‌های تصویر شیء را به راحتی و با قدرت بیشتری از نظر مقاوم بودن در برابر تغییرات استخراج کنند، از توصیفگرSIFT برای استخراج نواحی استفاده کردند، با این نگاه هر ناحیه معادل با یک نقطه کلیدی استخراج شده توسط SIFT بود. در ادامه بین هر نقطه کلیدی و نقاط کلیدی همسایه یک همسایگی در نظر گرفتند. آن‌ها سپس این ساختار را در گراف ARG قرار دارد. در نهایت آن‌ها از مقادیر بردار ۱۲۸ تایی SIFT برای ویژگی‌های یکانی و برای ویژگی‌های باینری از مقادیر بزرگنمایی و جهت استخراج شده توسط SIFT استفاده کردند.

[۱] Target distance
[۲] Histogram intersection
[۳] Hausdorff distant

در سال ۲۰۱۲ Sanroma و همکارانش [۲۷] روش دیگری ارائه دادند که به روش قبل تا حدی نزدیک بود. آن‌ها به جای اینکه پیچیدگی را به نقاط کلیدی منتقل کنند، ابتدا با استفاده از کشف کننده هریس [۲۸] که شناسایی کننده ساده‌ای است، نقاطی را استخراج می‌کنند. سپس به جای اینکه تنها با استفاده از توصیفگرها به تطبیق بپردازند، با استفاده از تعریف دو ارتباط ساختاری و هندسی این کار را انجام می‌دهند. این روش برخلاف اینکه از مقادیر مطلق[۱] پیکسل‌ها استفاده کند از مقادیر هندسی آن‌ها استفاده کنند [۲۹] و [۳۰]. در این الگوریتم، آن‌ها با یک مدل ترکیب شده روبرو بودند که برای حل آن از روش حداکثر انتظار[۲] استفاده می‌کنند. در این نگاه هدف، موفقیت در تخصیص احتمال است که با استفاده از انتصاب نرم[۳] انجام می‌شود. نکته مهم الگوریتم ارائه شده این است که می‌تواند خطا را در حین فرآیند یافته و اصلاح کند.

همان‌طور که در ابتدای فصل گفته شد روش‌های بسیار زیادی برای حل مسئله بازشناسی ارائه شده است. ارائه تمامی آن‌ها در این فصل و این پایان‌نامه میسر نیست برای مطالعه بیشتر به [۵] و [۱۳] مراجعه شود.

۲-۳) روش‌های الهام گرفته از بینایی انسان

آلن تورینگ[۴] (۱۹۱۲-۱۹۴۵) [۳۱] هوش مصنوعی را ساخت ماشینی دانست که می‌تواند از انسان تقلید کند. با الهام از این تعریف تورینگ، دسته دوم شامل روش‌هایی است که سعی می‌کنند بینایی انسان را تقلید کنند.

---

[۱] Absolute

[۲] Expectation Maximization (EM)

[۳] Soft-assign

[۴] Alan Turing

اگر کمی به بینایی انسان توجه کنیم، مشاهده خواهیم کرد که انسان به راحتی بسیاری از اجسام را می‌تواند تشخیص دهد بدون اینکه زحمت زیادی را متحمل شود. این در حالی است که بازشناسی اشیاء مسئله پیچیده‌ای می‌باشد. هدف اصلی تمام کارهایی که در این دسته انجام می‌شود این است که انسان چگونه اشیاء را تشخیص می‌دهد؟ [۳۲]

۲-۳-۱) محدودیت‌های سیستم‌های مرسوم بازشناسی

روش‌های مرسوم بازشناسی همیشه به یک پایگاه داده وابسته هستند و واضح است که هیچ پایگاه داده‌ای کامل نیست لذا این روش‌ها عمومیت ندارند. مشکل بعدی به این برمی‌گردد که روش‌های مرسوم بازشناسی یک بار آموزش می‌بینند و سپس مورد آزمایش قرار می‌گیرند که این نیز یک نقص است زیرا وقتی به بینایی انسان بازمی‌گردیم کاملاً متوجه خواهیم شد که انسان مدام در حال یادگیری است [۳۳].

مشکل بعدی سیستم‌های مرسوم این است که وقتی یک سیستم مرسوم پس از آموزش در یک محیط ناآشنا (منظور تعریف راسل از محیط است [۳۱]) قرار داده می‌شود عملاً هیچ کاری نمی‌تواند انجام دهد. در واقع نمی‌تواند خود را با شرایط جدید وفق[۱] دهد؛ اما انسان در یک محیط جدید به راحتی پس از مدت بسیار اندکی با محیط وفق پیدا می‌کند. این مشکلات تنها بخشی از ناتوانی سیستم‌های مرسوم در مقابل سیستم بینایی انسان است. در ادامه این فصل مرور بسیار مختصری بر روش‌هایی که در این دسته قرار می‌گیرند خواهیم کرد.

۲-۳-۲) سیستم بینایی انسان

تمامی کارهایی که در این دسته قرار می‌گیرند از مقاله [۳۴] سرچشمه می‌گیرند. در این مقاله یک مدل ریاضی برای لایه کورتکس[۲] مغز پیشنهاد شد. همان‌طور که مشخص شده است عمل بازشناسی اشیاء

[۱] Adopt
[۲] Cortex

در لایه‌ای از مغز به نام کورتکس انجام می‌شود. ما بدون اینکه بدانیم انرژی بسیار زیادی برای عمل بازشناسی اشیاء مصرف می‌کنیم. این مقاله یک معماری سلسله مراتبی از عمل بازشناسی ارائه می‌دهد این مدل HMAX نام دارد.

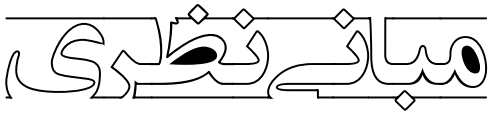
مقالات بسیار زیادی در سال‌های اخیر بر اساس این مقاله ارائه شد از جمله:

در سال ۲۰۱۲ Jeong و همکارانش [۳۳] روش سلسله مراتبی را ارائه دادند که از همان معماری کورتکس استفاده می‌کرد. این تحقیق ابتدا به مشکل یک بار آموزش دیدن سیستم‌های مرسوم بازشناسی اشاره می‌کند و سپس یک مدل افزایشی برای ارائه ویژگی‌ها ارائه می‌دهد. سپس بیان می‌کند که به جای اضافه کردن داده‌های اضافی از انعطاف‌پذیری داده‌ها استفاده شود. از طرفی اضافه کردن داده‌های اضافی فراموشی را نیز به سیستم تحمیل نخواهد کرد.

در کار دیگر siagian و همکارانش [۳۵] روشی مبتنی بر کورتکس برای ربات‌های متحرک در محیط‌های باز ارائه می‌دهند. روش‌های ارائه شده تا کنون در محیط‌های کنترل شده کارایی مناسبی داشته‌اند اما زمانی که محیط تغییر می‌کند و وارد محیط‌های باز می‌شویم کارایی سیستم‌های مرسوم به شدت کاهش می‌یابد. دلیل این کاهش کارایی به علت محیط بسیار پیچیده‌ای است که از خصوصیات ذاتی محیط‌های روباز می‌باشد.

این مقاله به ارائه معماری می‌پردازد که از لحاظ مکانیزم و همچنین پیچیدگی محاسباتی از انسان تقلید می‌کند. در این تحقیق اشاره می‌شود انسان همواره بر روی منابعی که در میدان دیدش وجود دارد تمرکز می‌کند. در واقع انسان آن قسمت که بیشتر مورد علاقه است را مورد بررسی و جستجو قرار می‌دهد. تجزیه و تحلیل بسیار سریع تصویر جامع در این استراتژی، منجر به یک مجموعه کوچک از مکان‌های کاندید مهم در صحنه می‌شود.

فصل سوم



۳-۱) مقدمه

همان طور که در فصل قبل هم اشاره شد با توجه به دقت کافی و همچنین سرعت اجرای قابل قبول SURF در این پایان‌نامه برای توصیف نقاط کلیدی از این توصیفگر استفاده شده است. به طور کلی می‌توان گفت سیستم شناسایی دارای دو مرحله می‌باشد:

در مرحله اول تصاویری (معمولاً نزدیک و از روبرو) از اشیاء مورد نظر تهیه می‌گردد، سپس در هر تصویر شیء از پس زمینه جدا گشته و از آن به کمک روش SURF نقاط مهم استخراج شده و توصیف می‌گردند. این ویژگی‌ها به عنوان مدل شیء در سیستم ذخیره می‌شوند.

در مرحله دوم که مرحله آزمون می‌باشد یک فریم[۱] تصویر از دوربین وارد سیستم می‌گردد آنگاه نقاط مهم تصویر با روش SURF استخراج شده و نواحی اطراف نقاط توسط توصیفگر SURF توصیف می‌شوند. نقاط توصیف شده با ویژگی‌های استخراج شده از مدل اشیاء (در مرحله آموزش) مقایسه می‌شوند تا شیء مورد نظر در تصویر ورودی شناسایی شوند. در این قسمت یک سری داده‌های پرت بوجود می‌آیند که برای حذف آن‌ها از RANSAC [۳۶] استفاده می‌شود. در آخرین مرحله اگر تعداد نقاط باقی مانده پس از اعمال RANSAC بیشتر از حد معینی بود مفهومش این است که شیء مورد نظر در این فریم وجود دارد که با استفاده از یک سری از پردازش‌ها آن را در تصویر شناسایی می‌کنیم.

۳-۲) توصیفگر SURF

SURF یک آشکارساز ویژگی محلی قوی می‌باشد که برای اولین با در سال ۲۰۰۶ توسط Herbert Bay و همکارانش ارائه شد [۴] و تا حدودی از توصیفگر SIFT الهام گرفته شده است. SURF بر اساس جمع پاسخ‌های موجک هار دو بعدی[۲] می‌باشد و وقتی کارآمد و مؤثر است که همراه با تصویر

---

[۱] Frame

[۲] 2D Haar wavelet

انتگرال[1] مورد استفاده قرار گیرد. این توصیفگر از یک تخمین صحیح از دترمینان هسین استفاده می‌کند که می‌توان آن را خیلی سریع با تصویر انتگرال محاسبه نمود. برای ویژگی‌ها هم که از جمع پاسخ‌های موجک هار استفاده می‌کند می‌توان دوباره از تصویر انتگرال استفاده نمود.

۳-۲-۱) تصویر انتگرال

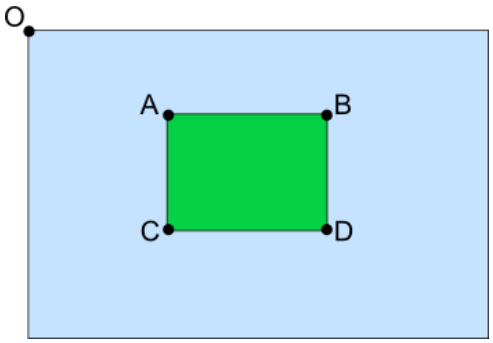
بخش عمده‌ای از افزایش بازدهی در SURF را می‌توان به تصویری واسطه که به عنوان «تصویر انتگرال» شناخته می‌شود، نسبت داد. تصویر انتگرال در ابتدا از تصویر ورودی محاسبه شده و به منظور محاسبه مجموع شدت پیکسل‌ها در یک ناحیه مستطیلی با سرعت بالا مورد استفاده قرار می‌گیرد. تصویر ورودی I گرفته شده و نقطه (x,y) در تصویر انتگرال I_Σ بوسیله جمع مقادیر بین نقطه مورد نظر و مبدأ محاسبه می‌گردد. فرمول زیر بیانگر همین موضوع است:

$$I_\Sigma(x,y) = \sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(x,y) \qquad (۱-۳)$$

با استفاده از تصویر انتگرال محاسبه مجموع شدت پیکسل‌ها در یک ناحیه مستطیلی به چهار عمل کاهش پیدا می‌کند. اگر ما یک مستطیلی که به رأس‌های A, B, C و D محدود شده است همانند شکل (۳-۱) را در نظر بگیریم، جمع شدت روشنایی‌ها برابر است با:

$$\Sigma = A + D - (C + B) \qquad (۲-۳)$$

[1] Integral image



شکل (۳-۱) : محاسبه جمع شدت روشنایی‌های محصور در ناحیه با استفاده از تصویر انتگرال [۳۷]

از آنجا که زمان محاسبه مستقل از تغییر در اندازه می‌باشد، این روش زمانی مفید است که ناحیه‌های بزرگی مورد نیاز باشد. SURF به خوبی از این ویژگی استفاده می‌کند تا در یک زمان ثابت کم، کانوولوشن‌هایی با اندازه متغیر را انجام دهد.

۳-۲-۲) هسین

مبنای کار SURF بر اساس ماتریس هسین[۱] می‌باشد. برای آشنا شدن با هسین یک تابع پیوسته از دو متغیر که مقدار تابع در (x,y) برابر f(x,y) می‌باشد را در نظر بگیرید. ماتریس هسین برابر ماتریس مشتقات جزئی تابع f می‌باشد.

$$H\big(f(x,y)\big) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \qquad (۳-۳)$$

دترمینان این ماتریس که به عنوان تفکیک‌کننده[۲] شناخته می‌شود با فرمول زیر محاسبه می‌گردد:

$$\det(H) = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 \qquad (۳-۴)$$

[۱] Hessian matrix

[۲] discriminant

از مقدار دترمینان برای طبقه‌بندی حداکثر و حداقل تابع بوسیله تست مشتق مرحله دوم استفاده می‌شود، و همچنین چون دترمینان حاصل‌ضرب مقادیر ویژه هسین می‌باشد می‌توان از علامت جواب برای طبقه‌بندی نقاط استفاده کرد؛ بدین صورت که اگر دترمینان منفی بود آنگاه مقادیر ویژه دارای علامت‌های متفاوت می‌باشند و از این رو نقطه اکسترمم محلی نمی‌باشد، اگر مثبت بود آنگاه هر دوی مقادیر ویژه منفی یا هر دو مثبت‌اند که در هر صورت نقطه به عنوان اکسترمم طبقه‌بندی می‌شود.

حال برای اینکه بتوان از نظریه بالا استفاده کرد ابتدا f(x,y) را با شدت‌های پیکسل تصویر I(x,y) جایگزین می‌کنیم. برای محاسبه مشتق‌ها هم می‌توان از کانوولوشن تصویر با یک کرنل مناسب استفاده کرد. در مورد SURF کرنل انتخابی یک کرنل گاوسی مرتبه دوم که نرمالیزه شده است می‌باشد که امکان تحلیل در حوزه مقیاس همانند حوزه مکان را می‌دهد. می‌توان کرنل‌هایی برای مشتقات جزئی در راستای x ، y و ترکیب xy ساخت که بشود با آن‌ها چهار المان ماتریس هسین را محاسبه نمود. استفاده از کرنل گاوسی این مزیت را دارد که می‌توان مقدار هموارسازی[1] هنگام مرحله کانوولوشن را تغییر داده، در نتیجه دترمینان را در مقیاس‌های متفاوت محاسبه کرد. علاوه بر این چون تابع گاوسی یک تابع ایزوتروپیک (متقارن دایروی) می‌باشد، کانوولوشن با آن مستقل از چرخش می‌باشد. حال می‌توان ماتریس هسین را به عنوان تابعی از مکان X = (x,y) و مقیاس σ بیان کرد.

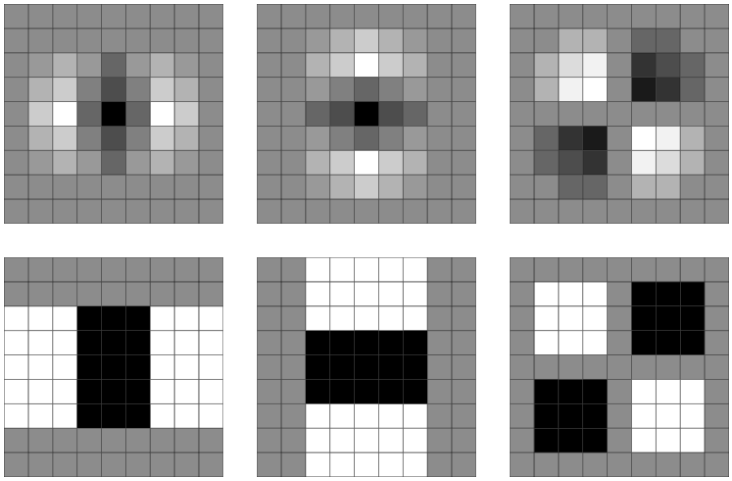
$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \qquad (۵-۳)$$

در اینجا $L_{xx}(X,\sigma)$ به کانوولوشن مشتق مرتبه دوم گاوسی $\frac{\partial^2 g(\sigma)}{\partial x^2}$ با تصویر در نقطه X = (x,y) اشاره دارد و به طور مشابه برای L_{yy} و L_{xy} هم همین طور است. این مشتقات به عنوان لاپلاسین گاوسی شناخته می‌شوند. حال می‌توانیم دترمینان ماتریس هسین برای هر پیکسل را محاسبه و از آن برای

[1] smoothing

پیدا کردن نقاط کلیدی استفاده کنیم. این قسمت آشکارساز هسین شبیه به چیزی است که Beaudet [38] ارائه داد.

Lowe [3] روشـی ارائه کرد که تقریب لاپلاسین گاوسی را با استفاده از تفاضل گاوسی‌ها محاسبه می کرد و این کار سـبب افزایش عملکرد محاسـبه شد. Bay [۴] در روشی مشابه یک تقریب از لاپلاسین گاوسی با استفاده از فیلتر جعبه‌ای[۱] به عنوان نماینده‌ای از کرنل مربوطه ارائه داد. شکل (۳-۲) شباهت بین کرنل‌های گسـسـته و فیلترهای جعبه‌ای متناظر با آن‌ها را نشان می‌دهد. افزایش عملکرد هنگامی حاصـل می‌شود که این فیلترها همراه با تصویر انتگرال که در بخش قبل توضیح داده شد به کار گرفته شوند.



شکل (۳-۲) : تقریب لاپلاسین گاوسی. سطر اول: مشتق مرتبه دوم گاوسی گسسته در راستای x، y و xy. که ما آن‌ها را به عنوان Lxx، Lyy و Lxy می‌شناسیم. سطر پایین: تقریب فیلتر جعبه‌ای وزن‌دار شده در راستای x، y و xy. که با عنوان Dxx، Dyy و Dxy شناخته می‌شوند [۳۷].

برای نشان دادن تفاوت به عنوان مثال می‌توان تعداد آرایه‌ها و همچنین عملگرهای مورد نیاز را در نظر گرفت. برای یک فیلتر ۹×۹ ما به ۸۱ عنصـر و عملگر برای فیلتری با مقادیر صـحیح نیاز داریم ولی در صـــورتی کـه این عدد برای فیلتر جعبه‌ای ۸ می‌باشـــد. حال اگر اندازه فیلتر افزایش پیدا کند هزینه

[۱] Box filter

محاسباتی به طور قابل توجهی برای لاپلاسین واقعی افزایش پیدا می‌کند در حالی که برای فیلترهای جعبه‌ای مستقل از اندازه می‌باشد.

در شکل (۳-۲) وزن‌هایی که به هر یک از قسمت‌های فیلتر اعمال می‌شود ساده انتخاب شده است. برای فیلتر D_{xy} ناحیه سیاه با مقدار ۱ وزن دهی شده، ناحیه سفید با مقدار ۱- و نواحی باقی‌مانده وزنی به آن‌ها داده نمی‌شود. فیلترهای D_{xx} و D_{yy} به طور مشابه وزن دهی می‌شوند اما نواحی سفید با مقدار ۱- و نواحی سیاه با مقدار ۲. وزن دهی ساده اجازه محاسبه سریع نواحی را می‌دهد اما یک اختلاف بین مقادیر پاسخ کرنل اصلی و تقریب زده شده بوجود می‌آید. برای کاهش این اختلاف Bay [4] فرمول زیر را به عنوان یک تقریب دقیق برای دترمینان هسین با استفاده از گاوسی تقریب زده شده ارائه کرد.

$$\mathrm{Det}\,(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \qquad (۶-۳)$$

در [4] دو فیلتر با جزئیات با هم مقایسه شدند و نتایج حاصل از فیلتر جعبه‌ای کمی کاهش در دقت داشت که در مقایسه با افزایش در عملکرد و سرعت ناچیز بود. دترمینان در اینجا اشاره به پاسخ حبابی[۱] در مکان $X = (x,y,\sigma)$ دارد. جستجوی مقدارهای بیشینه محلی این تابع روی هر دوی فضا و مقیاس باعث پیدا شدن نقاط کلیدی می‌گردد. روش دقیق برای استخراج این نقاط در قسمت بعدی توضیح داده خواهد شد.

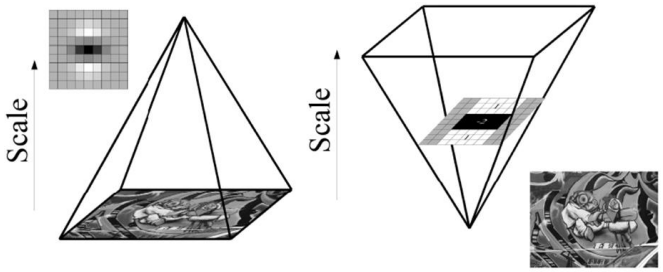
۳-۲-۳) ساخت فضای مقیاس

به منظور آشکارسازی نقاط کلیدی با استفاده از دترمینان هسین در ابتدا لازم است تا مقدمه‌ای درباره مفهوم یک فضای مقیاس[۲] داشته باشیم. یک فضای مقیاس یک تابع پیوسته است که می‌تواند برای پیدا کردن اکسترمم در تمام مقیاس‌های ممکن مورد استفاده قرار گیرد [39]. در بینایی ماشین فضای

[۱] blob response

[۲] scale-space

مقیاس به طور معمول به عنوان یک هرم تصـویر به اجرا در می‌آید که تصویر ورودی به صورت تکراری با کرنل گاوسـی کانوالو شده و به زیر نمونه تبدیل می‌شود (در اندازه کاهش پیدا می‌کند). این روش به طور مؤثر در SIFT [۳] به کار برده می‌شـود اما چون هر لایه وابسته به لایه قبلی می‌باشد و تصویر نیاز به تغییر اندازه دارد از لحاظ محاسـباتی بهینه نیسـت. از آنجا که زمان پردازش کرنل‌های استفاده شده در SURF مسـتقل از زمان اسـت، فضـای مقیاس می‌تواند بوسـیله‌ی اعمال کرنل‌هایی که اندازه آن‌ها افزایش پیدا کرده اسـت به تصـویر اصـلی سـاخته شـود. این روش این امکان را به ما می‌دهد که برای لایه‌های چندگانه از فضـای مقیاس هرم به طور همزمان پردازش شـده و عدم احتیاج به زیر نمونه‌های تصـویر باعث افزایش عملکرد می‌گردد. شکل (۳-۳) تفاوت بین ساختار فضای مقیاس سنتی و SURF متناظر با آن را نشان می‌دهد.



شکل (۳-۳) : هرم فیلتر. روش سنتی برای ساخت فضای مقیاس(سمت چپ). اندازه تصویر متغیر است و فیلتر گاوسی مکرراً به تصویر اعمال می‌شود تا لایه‌های بعدی را هموار کند. در روش SURF (سمت راست) تصویر اصلی ثابت است و فقط اندازه فیلتر تغییر می‌کند [۳۷].

یک فضـای مقیاس به تعدادی اکتاو[۱] تقسـیم می‌شود، که یک اکتاو به یک سری از نقشه‌های پاسخی[۲] که از دو برابر شـدن مقیاس بدسـت آمده، اشـاره دارد. در SURF پایین‌ترین سـطح فضـای مقیاس از خروجی فیلتر ۹×۹ که در شکل (۳-۲) نشان داده شده بدست می‌آید. این فیلترها مربوط به یک فیلتر گاوسـی با مقادیر صـحیح و $\sigma = 1.2$ می‌باشـند. لایه‌های بعدی با افزایش مقیاس فیلتر با حفظ همان

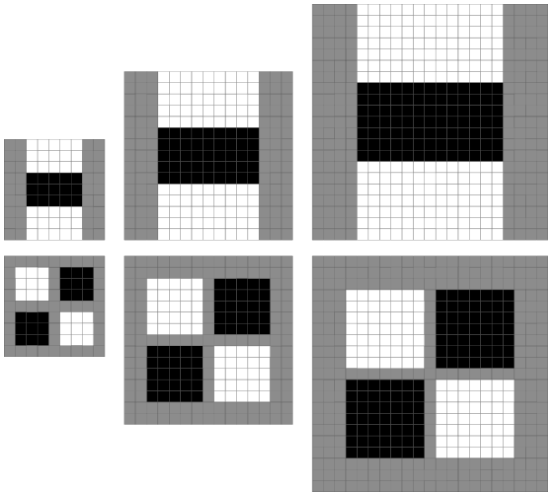
[۱] Octave
[۲] response maps

نسبت لایه فیلتر بدست می‌آید. چون اندازه فیلتر افزایش پیدا می‌کند در نتیجه مقدار مقیاس فیلتر گاوسی مربوطه نیز زیاد می‌شود و چون نسبت لایه‌ها ثابت باقی می‌ماند ما می‌توانیم این مقیاس را بوسیله فرمول زیر محاسبه کنیم.

$$\sigma_{approx} = \text{اندازه فیلتر جاری} \times \frac{\text{مقیاس فیلتر پایه}}{\text{اندازه فیلتر پایه}} = \text{اندازه فیلتر جاری} \times \frac{۱.۲}{۹} \qquad (۳-۷)$$

هنگام ساخت فیلترهای بزرگ‌تر یک سری عوامل وجود دارد که باید در نظر گرفته شود. افزایش اندازه بوسیله طول قطعه‌های مثبت و منفی بدست آمده تحت مشتقات جزئی مرتبه دوم محدود می‌شود. در فیلترهای تقریب زده شده اندازه قطعه[۱] به اندازه یک سوم طول ضلع فیلتر تنظیم شده و اشاره به طول ضلع کوتاه‌تر ناحیه‌های سیاه و سفید وزن‌دار شده دارد. از آنجا که ما نیاز به حضور پیکسل مرکزی داریم، ابعاد باید به یک اندازه اطراف این نقطه افزایش یابد و از این رو اندازه قطعه می‌تواند حداقل با مقدار ۲ افزایش یابد. چون در هر فیلتر ۳ قطعه وجود دارد که باید هم اندازه باشد، کوچک‌ترین اندازه گام بین فیلترهای پشت سر هم ۶ می‌تواند باشد. برای فیلترهای D_{xx} و D_{yy} طول ضلع بزرگ‌تر ناحیه وزن دار شده ۲ واحد در هر طرف افزایش پیدا می‌کند تا تناسب ساختار حفظ شود. شکل (۳-۴) ساختار فیلترها چنانکه افزایش در اندازه دارند را نشان می‌دهد.

[۱] Lobe size



شکل (۳-۴) :ساختار فیلتر. اندازه‌های فیلترهای بعدی باید حداقل ۶ واحد زیاد شوند تا ساختار فیلتر حفظ شود[۳۷].

با توجه به ملاحظات ذکر شـده اندازه فیلترهای بعدی ۱۵×۱۵، ۲۱×۲۱، ۲۷×۲۷ و ... می‌تواند انتخاب شود.

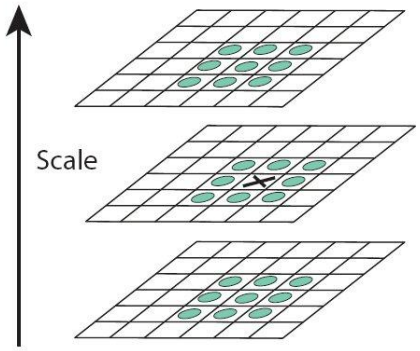
۳-۲-۴) محل دقیق نقاط کلیدی

به طور کلی مکان‌یابی نقاط کلیدی که مسـتقل از چرخش و مقیاس در تصـویر باشـند را می‌توان به ۳ گام تقسیم کرد.

در ابتدا پاسـخ‌ها آسـتانه گذاری می‌شـوند به طوری که تمام مقادیر کمتر از مقدار آسـتانه پیش‌فرض حذف می‌شـوند. افزایش مقدار آسـتانه باعث کاهش تعداد نقاط آشـکار شـده می‌گردد و فقط نقاط با قدرت بیشتر باقی می‌مانند.

بعد از آسـتانه گذاری، باید معلوم کنیم که نقطه انتخابی مقدار بیشـینه را در بین همسـایه‌های خود داشـته باشـد. برای این منظور هر پیکسـل در فضـای مقیاس با ۲۶ همسـایه خود مقایسـه می‌شود که

شـامل ۸ نقطه در مقیاس محلی و ۹ نقطه در هر کدام از مقیاس‌های بالا و پایین می‌باشد. شکل (۳-۵) این مرحله که حذف غیر بیشینه‌ها[۱] نام دارد را نشان می‌دهد.



شکل (۳-۵) : حذف غیر بیشینه. پیکسلی که با "X" نشانه گذاری شده است به عنوان بیشینه انتخاب می‌گردد اگر بزرگ‌تر از پیکسل‌های مجاور در لایه‌ی خود و لایه‌های بالا و پایین باشد [۳۷].

مرحله نهایی در مکان‌یابی نقطه‌ها شـــامل درون‌یابی داده‌های مجاور برای پیدا کردن مکان در هر دوی مقیاس و فضا برای دقت زیرپیکسل می‌باشد. این کار بوسیله‌ی انطباق[۲] با یک معادله درجه ۲ سه‌بعدی که بوسیله Brown ارائه شد انجام می‌گیرد. به منظور انجام این کار ما دترمینان تابع هسین، H(x,y,σ) را بوســیله بســط تیلور تا جمله درجه ۲ حول مکان انتخاب شده بیان می‌کنیم که به صورت زیر تعریف می‌شود:

$$H(x) = H + \frac{\partial H^T}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 H}{\partial X^2} X \qquad (۳-۸)$$

مکان درون‌یابی شـده اکسترمم $\hat{x} = (x, y, \sigma)$ بوسیله مشتق گرفتن از این معادله و برابر صفر قرار دادن بدست می‌آید:

$$\hat{x} = -\frac{\partial^2 H}{\partial \mathbf{x}^2}^{-1} \frac{\partial H}{\partial \mathbf{x}} \qquad (۳-۹)$$

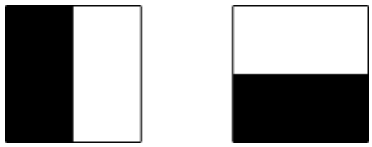
[۱] Non-Maximal Suppression

[۲] fitting

مشـتق در اینجا بوسـیله تفاضـل از پیکسل‌های همسایه تخمین زده می‌شود. اگر $\hat{x}$ بزرگ‌تر از ۰.۵ در راستاهای x، y یا σ باشد آنگاه این نقطه جز نقاط کلیدی نمی‌باشد و حذف می‌گردد.

۳-۲-۵) توصیفگر نقاط کلیدی

توصـیفگر SURF از هسـین سـریع[۱] برای توصیف نقاط کلیدی انتخاب‌شده استفاده می‌کند. این روش شـبیه SIFT می‌باشد اما از تصویر انتگرال همراه با موجک هار استفاده می‌شود که باعث افزایش قدرت و کاهش زمان محاسـبات می‌گردد. موجک هار فیلتری سـاده می‌باشـد که می‌تواند برای پیدا کردن گرادیان در راستای x و y مورداستفاده قرار گیرد (شکل ۳-۶).



شکل (۳-۶) : موجک هار. فیلتر سمت چپ جواب را در راستای x و سمت راست در راستای y محاسبه می‌کند. وزن‌ها برای ناحیه‌های سیاه ۱ و ۱- برای سفید می‌باشد. هنگامی که با تصویر انتگرال استفاده می‌شود هر موجک فقط به ۶ عمل برای محاسبه نیاز دارد [۳۷].

استخراج توصیفگر به دو مرحله تقسیم می‌شود. در ابتدا به هر نقطه کلیدی یک جهت که قابلیت تولید مجدد دارد[۲] تخصـیص داده می‌شـود، سـپس یک پنجره وابسـته به مقیاس که از یک بردار۶۴ بعدی اسـتخراج شـده، ساخته می‌شود. توجه به این نکته مهم است که تمام محاسبات برای توصیفگر باید بر اسـاس اندازه‌گیری نسـبی برای پیدا کردن مقیاس باشـد تا نتایجی مستقل از مقیاس بدست آید. روش استخراج توصیفگر در ادامه بیشتر توضیح داده شده است.

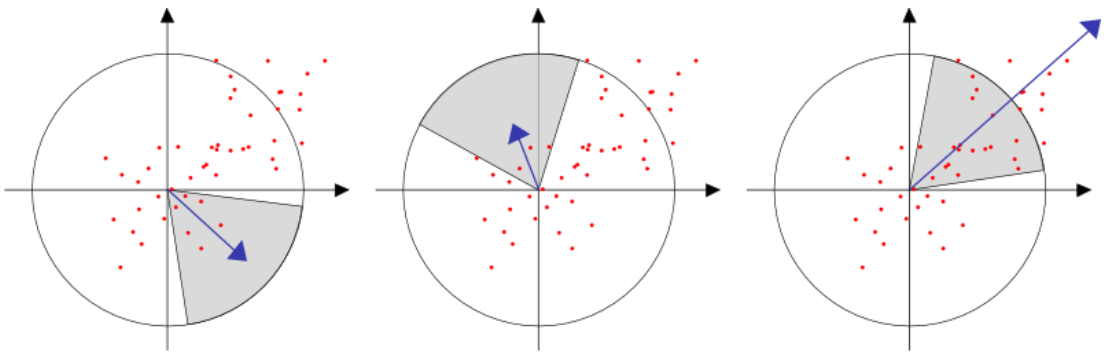
---

[۱] Fast-Hessian

[۲] reproducible orientation

۳-۲-۶) تخصیص جهت

برای اینکه تصـویر مسـتقل از چرخش باشد باید برای هر نقطه کلیدی انتخاب‌شده یک جهت با قابلیت تولید مجدد اختصـاص داده می‌شـود. اسـتخراج هر مؤلفه توصیفگر نسبت به این راستا انجام می‌پذیرد، بنابراین مهم می‌باشـد که این راسـتای انتخاب‌شده تحت شرایط مختلف قابل تکرار باشد. برای انتخاب این راستا ابتدا یک مجموعه از نقاط به مرکزیت نقطه کلیدی و شعاع 6σ گردآوری می‌شود که σ اشاره به مقیاس در نقطه انتخاب‌شـده دارد. سـپس موجک هار با اندازه 4σ برای تک تک نقاط این مجموعه محاسبه شده و پاسخ‌ها با یک فیلتر گاوسی با انحراف معیار 2.5σ و به مرکزیت نقطه کلیدی وزن دهی می‌شـوند. حال جواب‌های وزن‌دار شـده به عنوان نقاط در فضای بردار نشان داده می‌شود به طوری که پاسـخ موجک در راسـتای x در امتداد طول و پاسـخ در راسـتای y در امتداد عرض قرار گیرند. جهت غالب بوسـیله چرخش یک قطاع دایره با زاویه π/۳ حول مرکز انتخاب می‌شود. در هر مرحله پاسخ‌های x و y داخـل قطاع با هم جمع شـــده و برای تشـــکیل یک بردار جدید مورد اســـتفاده قرار می‌گیرد. طولانی‌ترین بردار جهت نقطه کلیدی را نشـان می‌دهد. این فرآیند در شـکل (۳-۷) نشـان داده شـده است.

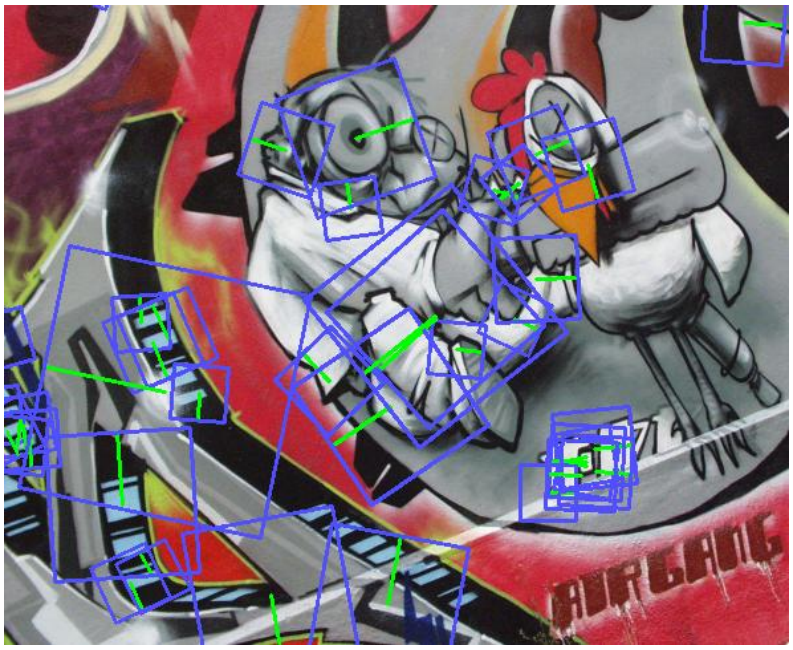


شکل (۳-۷) : تخصیص جهت: پنجره حول مرکز نقطه مورد نظر می‌چرخد و مؤلفه‌های پاسخ نقاط داخل آن با هم جمع شده و بردار آبی رنگ را بوجود می‌آورند. بزرگ‌ترین بردار جهت غالب را نشان می‌دهد [۳۷].

در بعضی کاربردها مستقل بودن از چرخش لازم نیست در نتیجه این مرحله می‌تواند حذف شود، از این رو افزایش عملکرد بیشتر می‌شود. در [4] این نسخه از توصیفگر به عنوان SURF عمودی (یا U-SURF) ارجاع داده شده و در برابر چرخش تصویر تا ۱۵-/+ درجه مقاوم می‌باشد.

۳-۲-۷) مؤلفه‌های توصیفگر

اولین گام برای استخراج ویژگی توسط توصیفگر SURF ساخت یک پنجره مربعی حول نقطه کلیدی می‌باشد. اندازه این پنجره 20σ بوده که در آن σ اشاره به مقیاس نقطه انتخاب شده دارد. علاوه بر این پنجره حول جهت پیدا شده در قسمت قبل می‌چرخد به طوری که تمام محاسبات بعدی نسبت به این جهت می‌باشد. تصویر (۳-۸) تعدادی از این پنجره‌ها را در نقاط کلیدی یک تصویر نشان می‌دهد.



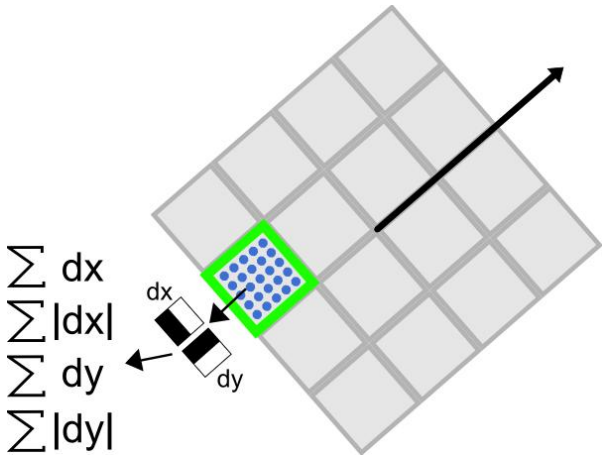
شکل (۳-۸) : پنجره‌های توصیفگر. اندازه پنجره ۲۰ برابر مقیاس نقطه کلیدی می‌باشد و سمت جهت غالب با رنگ سبز نشان داده شده است [۳۷].

پنجره توصیفگر به ۴×۴ زیر ناحیه متقارن تقسیم می‌شود. در هر یک از این زیر ناحیه‌ها موجک هار با اندازه 2σ برای ۲۵ نقطه نمونه که به صورت منظم توزیع شده‌اند، محاسبه می‌شود. اگر پاسخ‌های

موجـک در جهـت x و y را بـه ترتیب با dx و dy بیان کنیم آنگاه برای این ۲۵ نقطه نمونه یک بردار با جمع آن‌ها به صورت زیر محاسبه می‌کنیم:

$$V_{subregion} = \left[\sum dx, \sum dy, \sum |dx|, \sum |dy|\right] \qquad (۱۰-۳)$$

بنابراین هر زیر ناحیه چهار مقدار به بردار توصـیفگر اضـافه می‌کند که منجر می‌شود طول نهایی بردار برابر ۴×۴×۴=۶۴ گردد. در نتیجه توصـیفگر SURF مسـتقل از چرخش، مقیاس و روشـنایی می‌باشد. چگونگی این کار برای یک نقطه کلیدی در شکل (۳-۹) نشان داده شده است.



شکل (۳-۹) : مؤلفه‌های توصیفگر. مربعی که با رنگ سبز احاطه شده یک از ۱۶ زیر ناحیه می‌باشد و دایره‌های آبی‌رنگ نقاط نمونه را نشان می‌دهد که ما روی هر کدام پاسخ موجک را محاسبه می‌کنیم. همان طور که نشان داده شده پاسخ‌های x و y نسبت به جهت غالب محاسبه شده است [۳۷].

RANSAC (۳-۳

RACSAC مخفف عبارت Random Sample Consensus به معنی اجماع نمونه تصـادفی می‌باشـد. این روش یک الگوریتم عمومی می‌باشـد که می‌تواند با دیگر روش‌های تخمین پارامتر برای دستیابی به یک مدل با احتمال معین هنگامی که نویز روی داده‌ها شناخته شده نیست، به کار برده شود [۳۶].

۳-۳-۱) معرفی

تکنیک‌های تخمین پارامتر معمولی بر اســاس این فرض اســت که نویز روی داده‌های آموزش یک نویز گاوســی با میانگین صفر می‌باشد. بنابراین، می‌توان بعد از بهینه‌سازی پارامتر با یک میانگین‌گیری روی کل داده‌ها آن را هموار کرد. به هر حال، هنگامی که خطاهای بزرگی که ممکن اســت به خاطر مقدار زیاد نویز یا اشتباه در اندازه‌گیری موقع نمونه‌برداری اتفاق افتاده باشد، این فرض صادق نیست. این نوع از خطاها در حقیقت با مدل منطبق[۱] نمی‌شــوند و باید هنگام آموزش حذف شــوند تا انطباق داده‌ها با مدل بهتر صورت پذیرد.

۳-۳-۲) بررسی اجمالی

هـدف RANSAC این اســت که پارامترهای تابع را در زمانی که نمونه‌هایی با خطای بالا در آن وجود دارد و باعث به اشتباه افتادن در برآورد پارامترها می‌شود، تخمین بزند.

۳-۳-۳) فرضیات

RANSAC فرض می‌کند که داده‌های آموزش شـامل نقاط صـحیحی[۲] می‌باشـند که می‌توانند توسـط مدل بیان شـوند و همچنین شـامل نقاطی پرت که نمونه‌هایی با خطای بالا هسـتند و در حالت کلی با مدل منطبق نمی‌شـوند. بنابراین اسـتفاده از داده‌های پرت هنگام آموزش مدل خطای نهایی را افزایش می‌دهـد، بـه این خـاطر کـه آن‌ها غالباً اطلاعاتی در رابطه با مدل ندارند. از این گذشــته RANSAC پارامترهای مدل را تنها با نقاط صـحیح آموزش می‌دهد و از نقاط پرت چشـم‌پوشی می‌کند. به هر حال جدا کردن داده‌ها به عنوان نقاط صـحیح و نقاط پرت یک فرض نسـبتاً سنگین است، اما اختلاف اصلی بین RANSAC و دیگر روش‌هـا می‌بـاشــد. علاوه بر این، حتی اگر این فرض برای یک مجموعه داده

---

[۱] fit

[۲] inliers

صادق نباشد، RANSAC آسیب زیادی به تخمین پارامتر نمی‌زند، در این شرایط فرض می‌کند که کل مجموعه داده نقاط صحیح است و مدل را با استفاده از آن آموزش می‌دهد.

بـه منظور حـذف نقـاط پرت هنگـام آموزش، RANSAC از یـک مجموعـه کوچک از نمونه‌ها به جای اسـتفاده از کل داده‌ها برای آموزش مدل اسـتفاده می‌کند و سـپس مجموعه را به دیگر نمونه‌ها بسـط می‌دهد. هنگام اسـتفاده از مجموعه کوچک، خود به خود فرض می‌شود که مدل می‌تواند با تعداد کمی از نقاط صـحیح تخمین زده شـود. به هر حال، این فرض برای اکثر موارد صـدق می‌کند. برای مثال به منظور بدست آوردن یک تابع خطی دو نمونه داده کفایت می‌کند.

۳-۳-۴) روش کار

RANSAC به صـورت تصادفی یکنواخت[۱] یک مجموعه از نمونه‌های داده را انتخاب کرده و از آن برای تخمین پارامترها اسـتفاده می‌کند. آنگاه نمونه‌هایی که با یک حد مجاز خطا[۲] از مدل سـاخته شـده هسـتند را تعیین می‌کند. این نمونه‌ها به عنوان موافق مدل سـاخته شـده مطرح شـده و به عنوان «مجموعـه اجمـاع»[۳] از نمونـه‌های داده انتخاب شـده نامیده می‌شـود. در اینجا، داده‌هایی که درون مجموعه اجماع قرار دارند به عنوان نقاط صحیح و بقیه به عنوان نقاط پرت توسط RANSAC شناخته می‌شـوند. اگر تعداد نمونه‌های درون مجموعه اجماع به اندازه کافی زیاد باشـد، مدل نهایی اجماع را با استفاده از آن‌ها آموزش می‌دهد. این فرآیند را برای تعدادی تکرار دوباره انجام داده و مدلی که کمترین میانگین خطا بین مدل‌های ساخته شده دارد را به عنوان بهترین مدل برمی‌گرداند.

---

[۱] Uniformly random

[۲] Error tolerance

[۳] Consensus set

RANSAC به عنوان یک الگوریتم تصادفی این را تضمین نمی‌کند که بهترین مدل را نسبت به نقاط صحیح پیدا می‌کند. اما می‌توان با اختصاص مقادیر مناسب به پارامترهای الگوریتم، احتمال رسیدن به راه حل بهینه را افزایش داد.

۳-۳-۵) الگوریتم

۳-۳-۵-۱) پیش‌نیازها

در حقیقت RANSAC یک الگوریتم کامل و مستقل[۱] نیست، بلکه یک الگوریتم مکمل می‌باشد که از یک مدل و تابع فاصله آن برای تخمین قوی پارامترهای مدل هنگامی که نقاط پرت در آن وجود دارد، استفاده می‌کند. این عملگر به مجموعه داده‌ها اعمال شده و نقاط پرت را جدا کرده و مدل را تنها با نقاط صحیح بر اساس تابع فاصله مدل آموزش می‌دهد. بنابراین قبل از استفاده از RANSAC، مدل و تابع فاصله باید تعیین شده باشد. به هر حال، این قابلیت‌ها باعث می‌شود که RANSAC قابلیت توسعه را داشته باشد و این اجازه را بدهد که بتوان آن را با هر مدل تخمین و هر تابع فاصله‌ای به کار برد.

۳-۳-۵-۲) طرح

به منظور فهم بهتر اینکه RANSAC چگونه کار می‌کند، شبه کد الگوریتم به صورت معمول در زیر توضیح داده شده است.

[۱] Self-working

ورودی‌ها

- data – یک مجموعه از نقاط داده مشاهده شده
- model – یک مدل که می‌تواند با نقاط داده منطبق شود
- n – حداقل تعداد داده که برای انطباق با مدل نیاز است
- k – حداکثر تعداد تکرار مجاز در الگوریتم
- t – یک مقدار آستانه برای تعیین اینکه چه موقع یک نقطه با مدل منطبق می‌شود
- d – تعداد داده‌های موردنیاز برای اینکه یک مدل به خوبی با داده منطبق شده است

خروجی

- bestfit – پـارامترهـای مدل که بهترین انطباق را با داده‌ها دارند (یا «پوچ» اگر مدل خوبی پیدا نشد)

iterations = ۰

bestfit = پوچ

besterr = یک عدد بزرگ

تا زمانی که iterations کمتر از k باشد کارهای زیر را تکرار کن

- maybeinliers = n مقدار که به صورت تصادفی از داده‌ها انتخاب شده‌اند
- maybemodel = پارامترهای model که منطبق با maybeinliers شده‌اند
- alsoinliers = مجموعه تهی
- برای هر نقطه درون داده‌ها که درون maybeinliers نیست کارهای زیر را انجام بده
  اگر نقطه با خطای کمتر از t با maybemodel منطبق شد آنگاه:
  نقطه را به alsoinliers اضافه کن
- اگر تعداد المان‌های درون alsoinliers از d بیشتر شد آنگاه:
  (این معلوم می‌کند که ممکن است یک مدل خوب پیدا کرده باشیم)
  (حال تست می‌کنیم تا ببینیم آیا واقعاً خوب است)
  bettermodel = پـارامترهای model که با تمام نقاط درون maybeinliers و alsoinliers انطباق داده شده‌اند
  thiserr = یک اندازه‌گیری نسبت به اینکه چقدر مدل با نقاط منطبق است
  اگر thiserr کمتر از besterr بود آنگاه:
  besterr = bestfit
  thiserr = besterr
- Iterations را یکی افزایش بده

مقدار bestfit را به عنوان خروجی برگردان

۳-۳-۵-۳) پارامترها

همان طور که در ابتدای طرح توضیح داده شد، RANSAC به تعدادی پارامتر پیش‌فرض برای اجرا نیاز دارد (اندازه زیرمجموعه نمونه‌ها[۱] (n)، آستانه مجاز خطا[۲] (t)، حداقل آستانه اجماع[۳] (d) و تعداد تکرار الگوریتم (k)). به علاوه، تخمین نسبت نقاط صحیح (w) در مجموعه داده به منظور محاسبه بعضی از این پارامترها مهم می‌باشد.

از آنجا که RANSAC یک الگوریتم تصادفی می‌باشد ضروری است که پارامترها به طور صحیح تخمین زده شوند تا احتمال پیدا کردن مدل بهینه افزایش پیدا کند، با این حال هنوز هم برتری‌های محاسباتی یک الگوریتم تصادفی نسبت به الگوریتم‌هایی که به صورت کامل و جامع عمل می‌کنند، قابل انکار نیست. در اینجا به بعضی از روش‌های محاسبه پارامتر اشاره می‌کنیم.

۳-۳-۵-۴) نسبت نقاط صحیح

اگر چه این یک پارامتر مستقیم در الگوریتم نیست، ولی نسبت نقاط صحیح در محاسبه پارامترهای الگوریتم استفاده می‌شود و حتی می‌تواند پیچیدگی محاسباتی را تحت تأثیر قرار دهد. از این رو اگر اطلاعاتی در مورد نسبت نقاط پرت درون مجموعه داده‌ها قبل از اجرای RANSAC داشته باشیم مفید است.

۳-۳-۵-۵) اندازه زیرمجموعه نمونه

اندازه زیرمجموعه نمونه، تعداد نمونه‌هایی است که به صورت تصادفی توسط RANSAC برای پیکربندی مدل در هر تکرار انتخاب می‌شود. این اندازه به طور مستقیم به مدلی که می‌خواهد منطبق

---

[۱] Sample subset

[۲] Error tolerance threshold

[۳] Consensus threshold

به مجموعه داده شـود بسـتگی دارد. RANSAC از حداقل تعداد نمونه‌های موردنیاز برای معرفی مدل به عنوان اندازه زیرمجموعه نمونه اسـتفاده می‌کند. به عنوان مثال برای انطباق یک مدل خطی ۲ نمونه داده را انتخاب می‌کند یا مثلاً برای پیدا کردن تبدیل سـه‌بعدی[۱] بین دو صـفحه از ۴ نقطه اسـتفاده می‌کند.

n = حداقل تعداد نمونه برای تعیین مدل

۳-۳-۵-۶) آستانه مجاز خطا

RANSAC به این منظور از آسـتانه مجاز خطا اسـتفاده می‌کند که معلوم کند که یک داده آیا موافق یک مدل می‌باشـد یا نه. نمونه‌های زیر این آسـتانه مجموعه اجماع برای آن مدل را تشکیل می‌دهند، و اگر مدل درسـت پیدا شده باشد همان نقاط صحیح در مجموعه داده می‌شوند. از این رو این مقدار باید بر اساس خطای گاوسی در نقاط صحیح انتخاب شود.

t = خطای گاوسی در نقاط صحیح

۳-۳-۵-۷) حداقل آستانه اجماع

حداقل آسـتانه اجماع، کمترین تعداد از نمونه‌ها می‌باشد که به عنوان یک اجماع معتبر برای تولید یک مدل نهایی در آن تکرار پذیرفته می‌شـود. تعداد نمونه‌ها در یک مجموعه اجماع معتبر به طور مسـتقیم به تعداد نقاط صـحیح در مجموعه داده بسـتگی دارد. از این رو RANSAC از یک مقدار آسـتانه که کوچک‌تر یا مسـاوی تعداد نقاط صـحیح می‌باشـد اسـتفاده می‌کند. اگر تعداد کل نمونه‌ها در مجموعه داده برابر |مجموعه داده| ، تعداد نقاط صـحیح در مجموعه برابر w و حداقل آسـتانه اجماع برابر d باشد آنگاه:

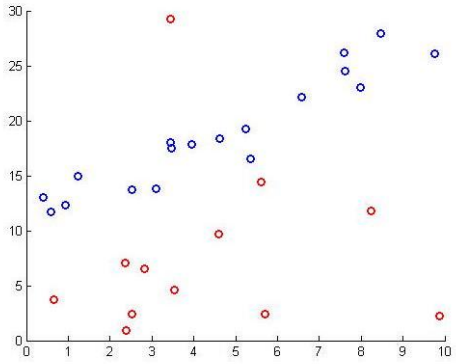
[۱] Perspective transformation

d ≈ w.|مجموعه داده|

برای فهم بهتر موضـــوع یک مثال ســـاده از تخمین یک تابع خطی که همراه با یک ســـری نقاط پرت می‌باشد در ادامه آورده خواهد شد.

۳-۳-۶) تخمین تابع خطی

شکل (۳-۱۰) مجموعه داده که در این مثال استفاده شده است را نشان می‌دهد.

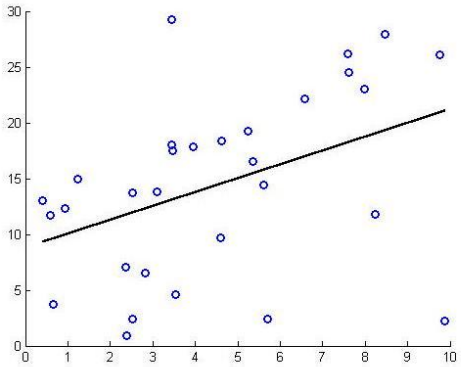


شکل (۳-۱۰) : ۳۰ نمونه داده که بوسیله معادله y = 2x + 5 و اضافه کردن نویز گاوسی به آن و همچنین اضافه کردن مقداری داده پرت تولید شده است. نقاط پرت به رنگ قرمز و نقاط صحیح به رنگ آبی می‌باشند.

۳-۳-۶-۱) رگرسیون خطی

در شـکل (۳-۱۱) نتیجه روش رگرســیون خطی[۱] بدون RANSAC آورده شـده اسـت. از آنجا که این روش نمونه‌های با خطای بالا را تشـخیص نمی‌دهد، مدل انطباق ساخته شده به شدت تحت تأثیر نقاط پرت قرار گرفته است.

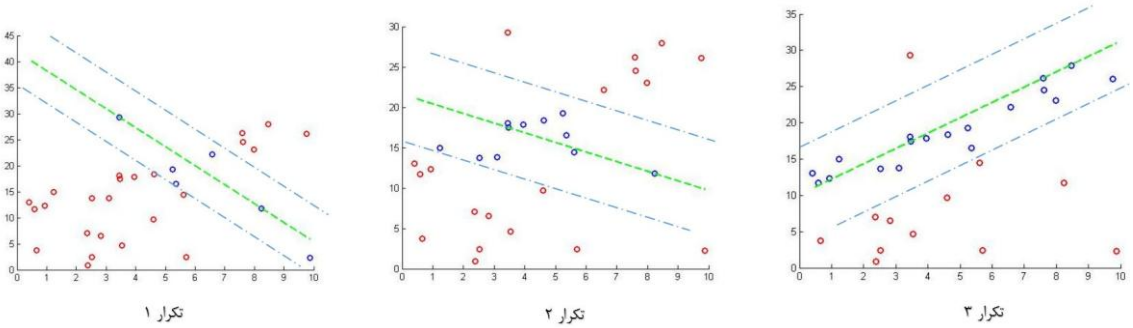
[۱] Linear regression



شکل (۳-۱۱) : بدست آوردن مدل خطی با استفاده از رگرسیون خطی

۳-۳-۶-۲) استفاده از RANSAC

در شکل (۳-۱۲) چند نمودار از مجموعه داده در چند تکرار RANSAC آورده شده است



شکل (۳-۱۲) : چند مرحله تکرار الگوریتم RANSAC

تکرار ۱

در این تکرار ۲ نقطـه انتخـابی از نقاط پرت می‌باشـــد در نتیجه نمی‌تواند یک تخمین خوب برای خط باشد. از این رو نمی‌تواند حداقل آستانه اجماع را بدست آورد.

تکرار ۲

در اینجا یکی از نقاط انتخابی از نقاط صـحیح می‌باشـد ولی هنوز یکی از انتخاب‌ها نقطه پرت است. به همین دلیل مدل ساخته شده خیلی دقیق نیست و باز هم حداقل آستانه اجماع را بدست نمی‌آورد.

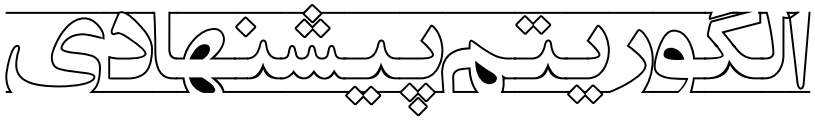
تکرار ۳

در این مرحله RANSAC به صورت تصادفی دو داده صحیح را برای زیر مجموعه داده انتخاب کرده است. از آنجا که زیر مجموعه نمونه فقط شامل نقاط صحیح همراه با یک آستانه مجاز خطای مناسب می‌باشد، تمام نقاط صحیح می‌توانند به عنوان مجموعه اجماع انتخاب شوند. با توجه به این موضوع مدل می‌تواند حداقل آستانه اجماع را کسب کرده و مدل نهایی را با استفاده از تمام نمونه‌های درون اجماع محاسبه کند.

۳-۳-۷) مدل انتخابی در این پایان‌نامه

از آنجا که بعد از استخراج ویژگی از دو تصویر (یکی تصویر شیء مورد نظر و یکی تصویر جستجو) توسط SURF و تطبیق آن‌ها یک سری داده پرت بوجود می‌آید هدف ما در اینجا حذف کردن آن‌ها و بدست آوردن یک تبدیل بین دو تصویر می‌باشد. در نتیجه حداقل نقطه‌ای که برای این تبدیل نیاز داریم ۴ عدد بوده و مدل ما هم یک تبدیل سه‌بعدی می‌باشد. روش کار بدین صورت است که ابتدا ۴ نقطه به صورت تصادفی از نقاط تطبیق داده شده انتخاب کرده و یک تبدیل برای آن‌ها بدست می‌آوریم (که یک ماتریس ۳×۳ می‌شود). حال تمام نقاط کلیدی تصویر شیء را با این تبدیل نگاشت داده و سپس فاصله آن‌ها را با نقطه کلیدی متناظر با آن‌ها در تصویر جستجو محاسبه می‌کنیم. فاصله هر نقطه که کمتر از آستانه مجاز خطا بود آن نقطه را به مجموعه اجماع اضافه می‌کنیم. در انتها این روند را به تعداد تکرار مجاز انجام می‌دهیم و در هر سری طول مجموعه اجماع را ذخیره کرده و بعد از انجام تمام تکرارها آن مجموعه‌ای که بیشترین طول را دارد بهترین مدل ساخته شده می‌باشد. حال که بهترین مدل پیدا شد تمام نقاط کلیدی تصویر شیء را با این تبدیل نگاشت می‌دهیم و دوباره فاصله آن‌ها را با نقاط متناظر با خودشان در تصویر جستجو محاسبه کرده و فاصله هر کدام که از آستانه مجاز خطا کمتر بود بدین معنی است که آن نقطه یک نقطه صحیح بوده وگرنه جز نقاط پرت می‌باشد.

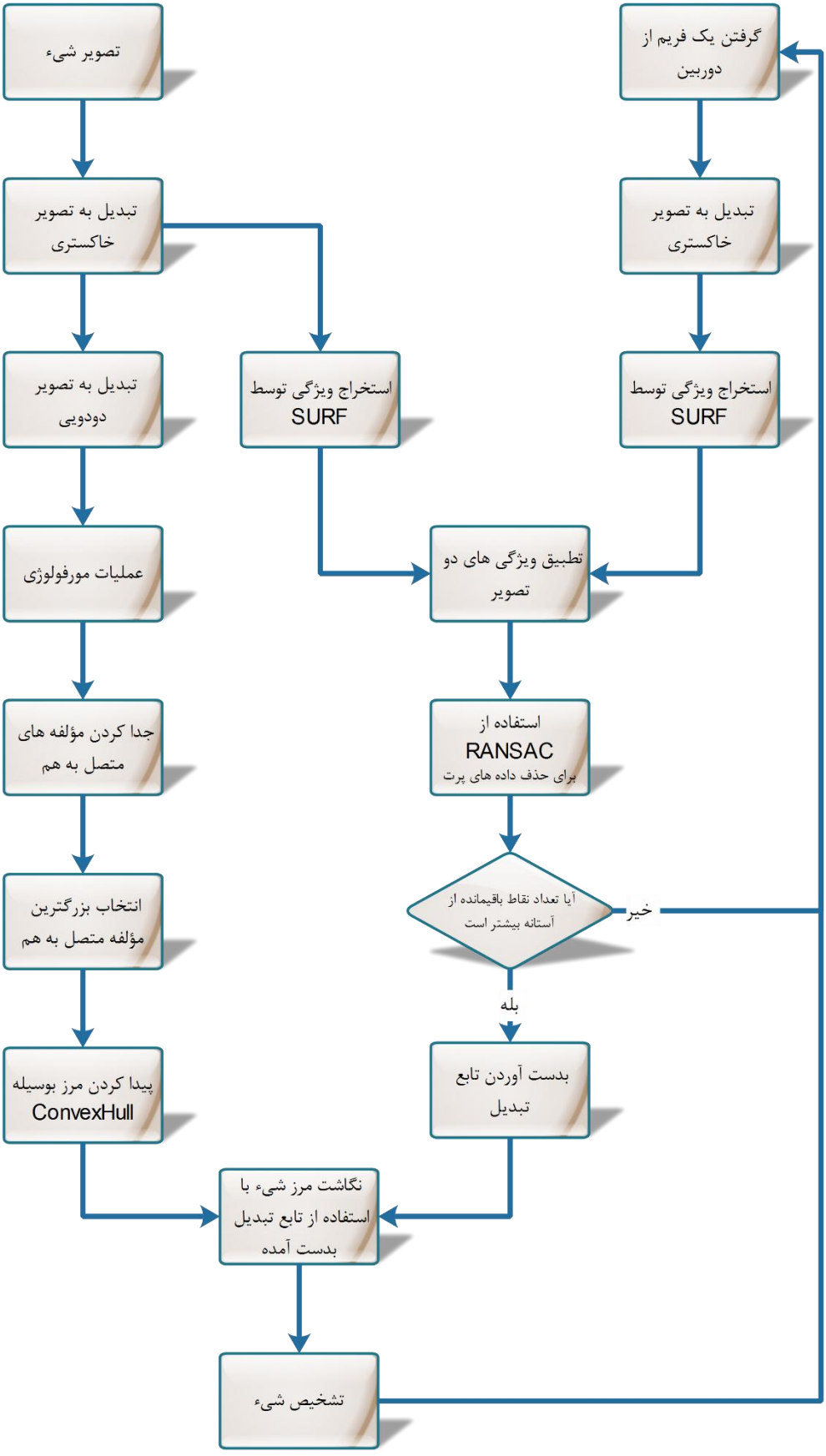
فصل چهارم



۴-۱) مقدمه

همان طور که قبلاً اشـاره شد هدف ما در این پایان‌نامه تشخیص شیء به صورت بلادرنگ و پیاده‌سازی سـخت‌افزاری آن می‌باشـد. در این فصـل ما به بیان الگوریتم ارائه شـده و نتایج پیاده‌سـازی نرم‌افزاری می‌پردازیم. روال کلی بدین صـورت اسـت که ابتدا از تصـویر شیء بوسیله SURF یک مجموعه ویژگی اسـتخراج می‌کنیم. سـپس بوسـیله یک سـری عملیات پردازش تصویر که در ادامه بیشتر توضیح داده خواهد شـد شـیء را از پس‌زمینه جدا کرده و مرز آن را بدسـت می‌آوریم. حال یک فریم از دوربین گرفته[1] شـده و از آن هم بوسـیله SURF ویژگی استخراج می‌کنیم. حال ویژگی‌های بدست آمده از دو تصـویر را با هم تطبیق داده و ویژگی‌های نزدیک به هم را هم‌ارز می‌گیریم. در این مرحله یک سـری داده پرت بوجود می‌آید که آن‌ها را بوسـیله RANSAC حذف می‌کنیم. در این مرحله اگر تعداد نقاط باقی مانده بعد از حذف بیشـتر از آسـتانه معینی بود آنگاه معلوم می‌شود که تصویر مورد بررسی حاوی شـیء موردنظر اسـت. بعد از این مرحله بوسـیله تبدیلی که از نقاط صـحیح بدسـت آمده مرز شـیء را نگاشت می‌دهیم تا شیء در تصویر جستجو شناسایی شود؛ و سپس این مراحل دوباره برای فریم بعدی تکرار می‌شود. نمای کلی الگوریتم را می‌توانید در شکل (۴-۱) مشاهده کنید.

[1] در بعضی بخش‌ها از این فریم به عنوان تصویر جستجو هم نام برده شده است.

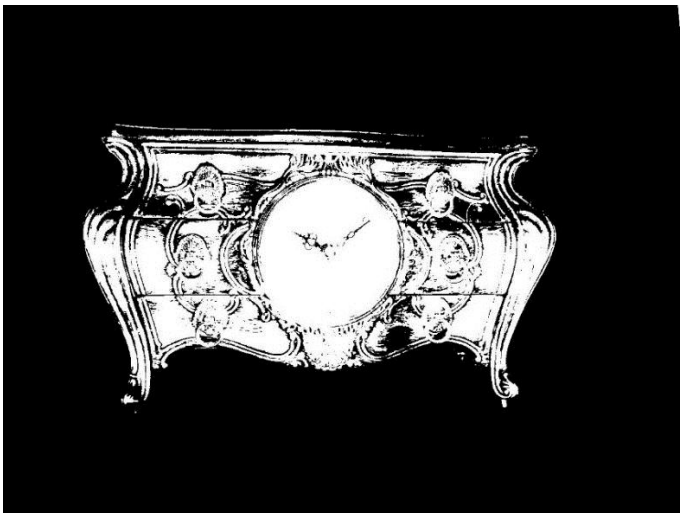


شکل (۴-۱) : نمای کلی الگوریتم پیشنهادی

۴-۲) عملیات پیش‌پردازش روی تصویر شیء

ابتدا تصـویر شـیء (شکل ۴-۲) را به تصویر خاکستری[۱] تبدیل کرده و با استفاده از یک آستانه مناسب آن را به تصویر دودویی[۲] تبدیل می‌کنیم (شکل ۴-۳).

شکل (۴-۲) : تصویر شیء موردنظر

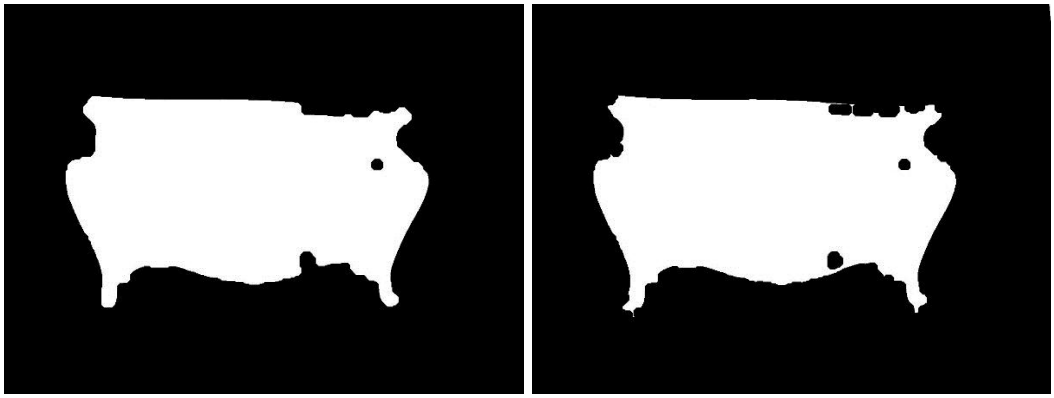


شکل (۴-۳) : تبدیل تصویر شیء به تصویر دودویی

[۱] Grayscale

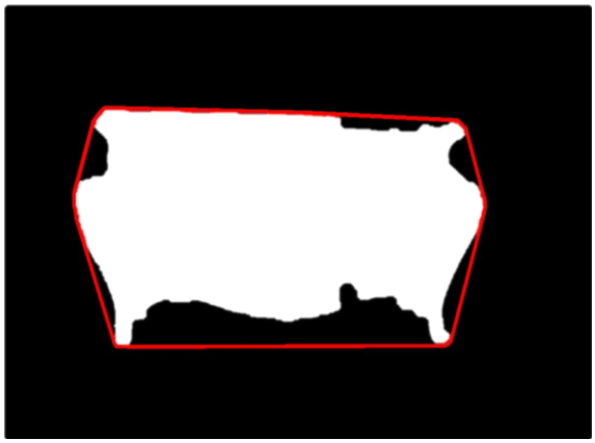
[۲] Binary

سـپس با اسـتفاده از عملیات ریخت‌شـناسـی[۱] بستن[۲] و بازکردن[۳] همان طور که در شکل (۴-۴) نشان داده شـده قسمت‌های کوچک را حذف می‌کنیم. بعد از این مرحله مؤلفه‌های متصل به هم در تصویر را بدست آورده و بزرگ‌ترین آن‌ها که شیء مورد نظر می‌باشد را انتخاب می‌کنیم.



شکل (۴-۴) : نتیجه تصویر بعد از اعمال عملیات ریخت‌شناسی بستن (شکل راست) و باز کردن (شکل چپ)

حال با استفاده از Convex Hull مانند شکل (۴-۵) مرز شیء را بدست می‌آوریم.



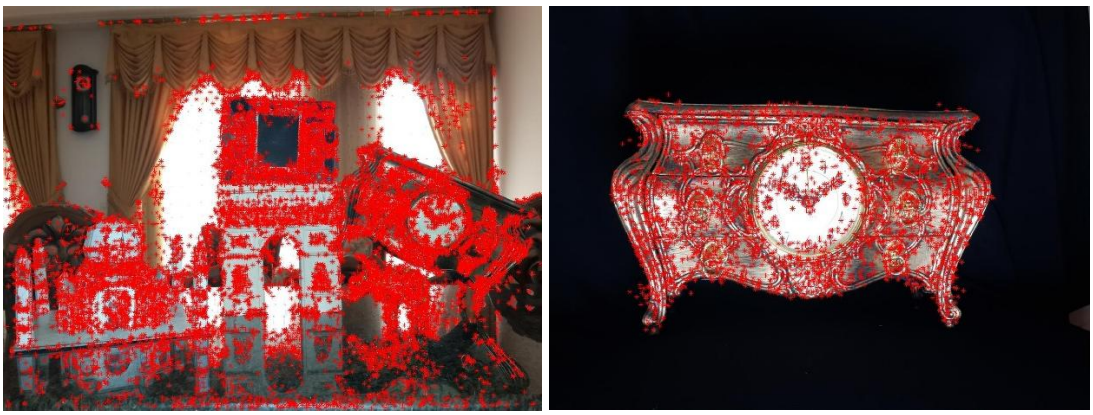
شکل (۴-۵) : بدست آوردن مرز شیء با استفاده از Convex Hull

[۱] Morphological processing
[۲] Closing
[۳] Opening

در مرحله آخر این مرز بدست آمده را ذخیره می‌کنیم تا در پیدا کردن شیء هدف در تصویر جستجو از آن استفاده کنیم.

۴-۳) استخراج ویژگی از تصاویر

همان طور که در مقدمه گفته شد و در فصل قبل هم به طور مفصل بحث شد برای استخراج ویژگی از توصیفگر SURF استفاده می‌کنیم. روال کار بدین صورت است که از تصویر حاوی شیء فقط یک بار و در ابتدای فرآیند ویژگی استخراج کرده و در حافظه ذخیره می‌کنیم ولی برای تصویر دوربین در هر فریم باید این کار انجام گردد. نقاط کلیدی پیدا شده مربوط به هر دو تصویر در شکل (۴-۶) نشان داده شده است.

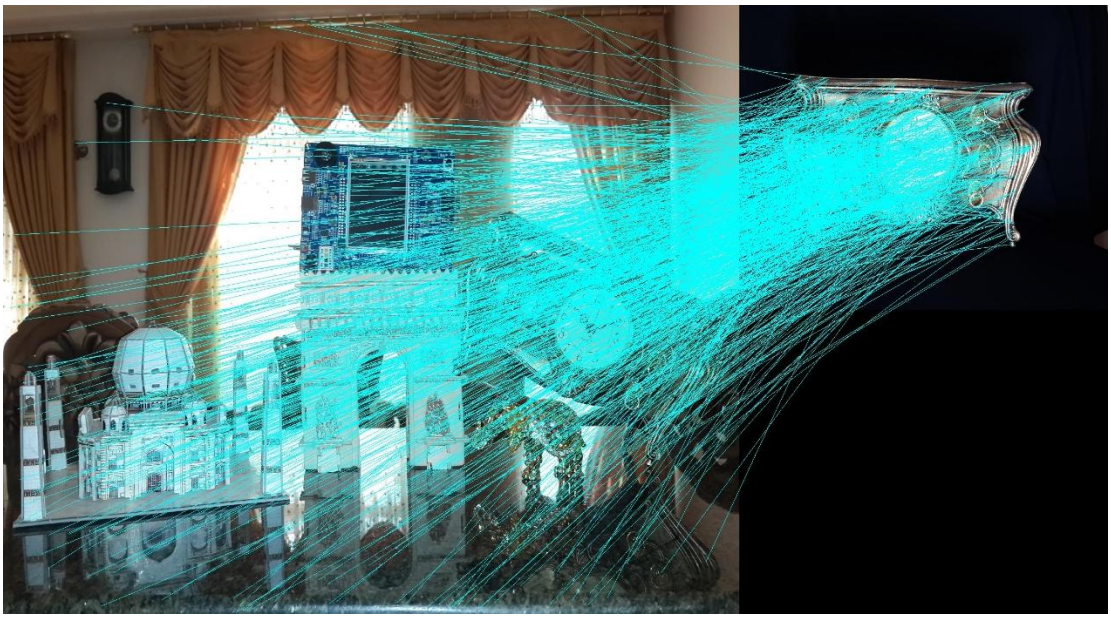


شکل (۴-۶) : پیدا کردن نقاط کلیدی در تصویر حاوی شیء (سمت راست) و فریم دوربین یا تصویر جستجو (سمت چپ)

۴-۴) تطبیق ویژگی

بعد از استخراج ویژگی باید ویژگی‌ها را با هم تطبیق دهیم تا نقاط متناظر با هم در تصویر شیء و تصویر جستجو بدست آیند. روشی که ما از آن استفاده می‌کنیم بدین صورت است که برای هر نقطه در تصویر شیء فاصله اقلیدسی آن را با تمام نقاط کلیدی در تصویر جستجو محاسبه کرده سپس آن‌ها را بر حسب اندازه‌شان مرتب می‌کنیم. حال اگر نسبت کمترین آن‌ها به یکی قبل از آن بیشتر از آستانه

معینی شـده باشـد (مثلاً در اینجا ما ۰/۶ را انتخاب کردیم) آنگاه این نقطه متناظر با نقطه‌ای اسـت که کمترین فاصـله را داشـته وگرنه حذف می‌گردد. اسـتفاده از این نسـبت به این منظور است که اگر یک نقطه کلیدی در تصـویر شـیء از لحاظ فاصـله‌ای نزدیک به حداقل دو نقطه در تصویر جستجو بود این نقطه نمی‌تواند انتخاب خوبی باشـد زیرا باعث بوجود آمدن خطا یا داده پرت می‌گردد لذا اگر آن را در ابتـدا حـذف کنیم هم بـاعـث کـاهش خطـا می‌گردد و هم در مرحلـه بعـد که داده‌های پرت را حذف می‌کنیم باعث کمتر شـدن محاسـبات می‌شود. نقاط متناظر در شکل (۴-۷) بوسیله خط به هم متصل شده‌اند.

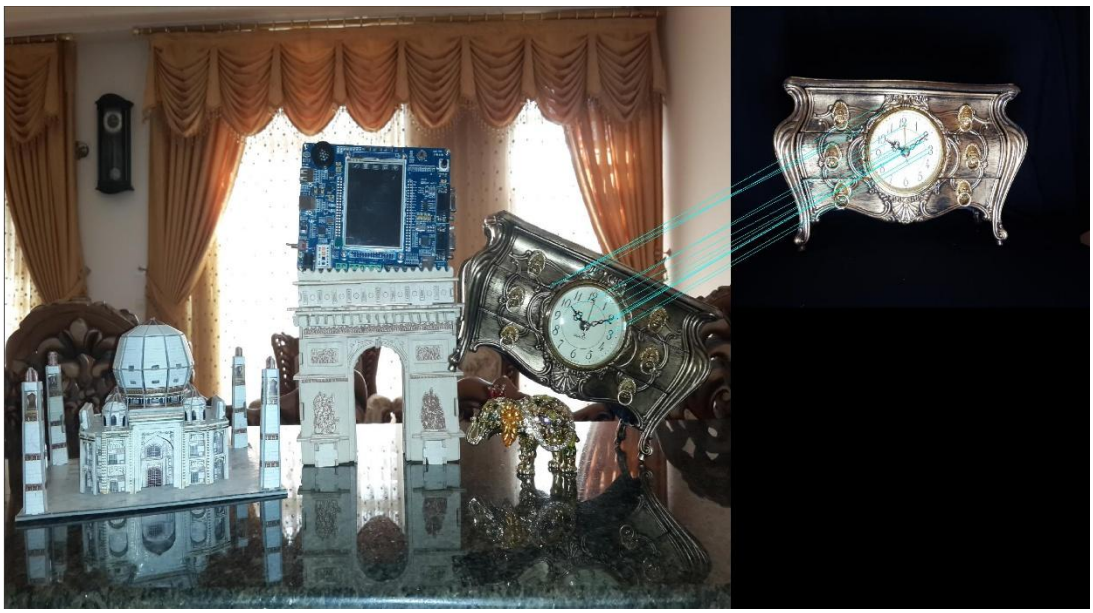


شکل (۴-۷) : تطبیق ویژگی‌ها در دو تصویر

۴-۵) حذف داده‌های پرت با استفاده از RANSAC

همان طور که در شکل (۴-۷) مشاهده می‌شود با وجود اینکه در مرحله قبل آن ویژگی‌هایی که شرایط مناسـب را نداشتند حذف کردیم ولی باز هم تعدادی نقطه پرت وجود دارد. برای حذف این داده‌ها ما از روش RANSAC که در فصـل قبل توضـیح داده شـد اسـتفاده می‌کنیم (شـکل ۴-۸). خروجی

RANSAC یک تعداد نقطه صـــحیح و یک ماتریس تبدیل می‌باشـــد که از آن در قســـمت بعد برای تشخیص محل دقیق شیء استفاده می‌کنیم.

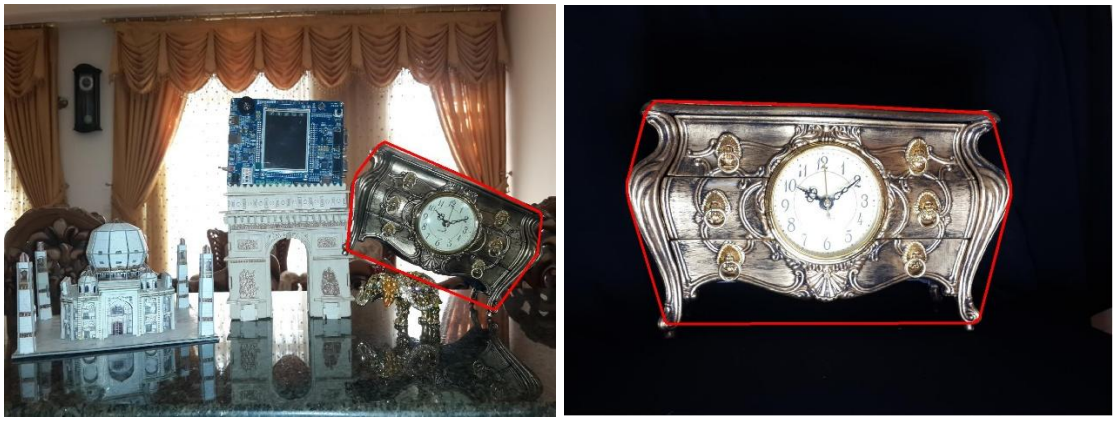


شکل (۴-۸) : استفاده از RANSAC برای حذف داده‌های پرت

حال اگر تعداد نقاط خروجی RANSAC از حد معینی کمتر بود (مثلاً در اینجا ما مقدار آســـتانه را ۴ نقطه در نظر گرفتیم) به این معنی می‌باشد که تصویر جستجو حاوی شیء موردنظر نیست.

۴-۶) تشخیص و مکان‌یابی شیء در تصویر جستجو

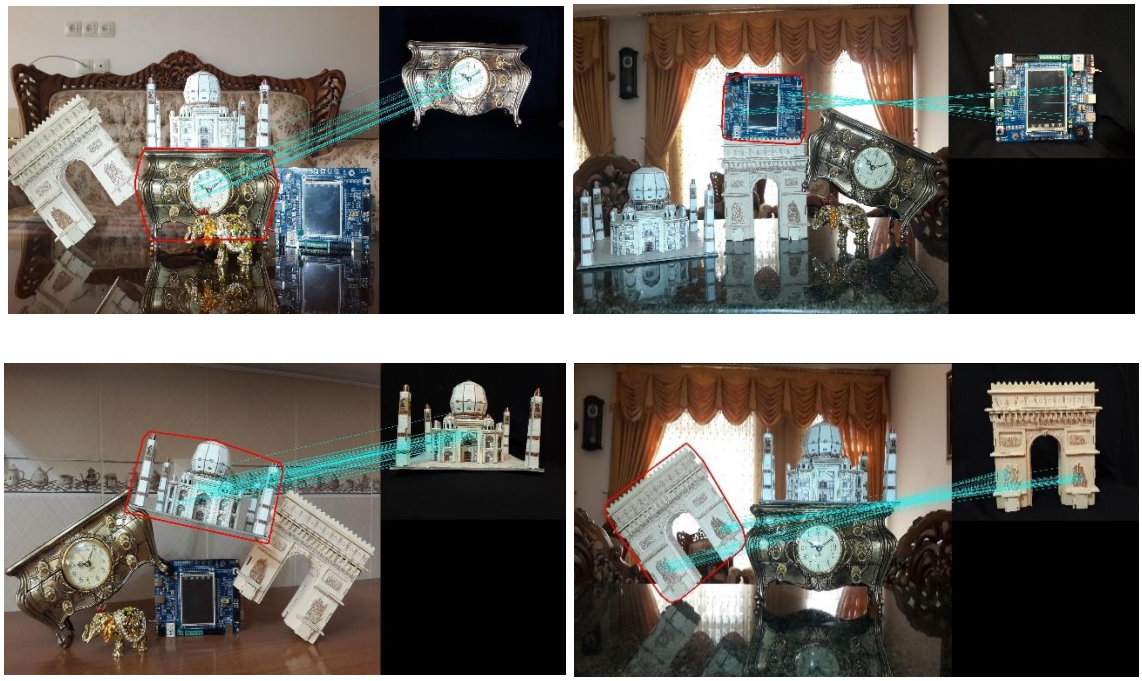
مرحله نهایی الگوریتم پیدا کردن محل دقیق شـیء در تصویر می‌باشد. برای این منظور نقاط مرزی که بوسـیله Convex Hull در مراحل قبل بدسـت آمد را توسـط ماتریس تبدیل بدست توسط RANSAC نگاشـت می‌دهیم. این کار باعث می‌شود مرز شیء در تصویر جستجو معلوم گردد. این مرحله که منجر به تشخیص نهایی شیء در تصویر جستجو می‌باشد در شکل (۴-۹) نشان داده شده است.



شکل (۴-۹) : مرز شیء در تصویر اصلی (سمت راست) و مرز نگاشت یافته آن تحت تبدیل بدست آمده توسط RANSAC (سمت چپ)

۴-۷) نتایج شبیه سازی

در شکل (۴-۱۰) می‌توانید تعدادی از نتایج شبیه سازی شده در صحنه واقعی را مشاهده کنید.



شکل (۴-۱۰) : تعدادی از نتایج شبیه سازی شده در صحنه واقعی

برای اینکه میزان سرعت و دقت روش پیشنهادی را بسنجیم آن را با توصیفگر SIFT مقایسه نمودیم. این توصیفگر همان طور که گفته شد یکی از مشهورترین و قوی‌ترین توصیفگرها برای توصیف شیء می‌باشد. برای آزمایش از پایگاه داده‌ای که بوسیله Mikolajczyk[1] تهیه شده بود و همچنین از یک رایانه با پردازنده Core™ i7 و حافظه رم ۶ GB استفاده کردیم. پایگاه داده انتخابی از تصاویری از صحنه‌های حقیقی همراه با بافت[2] تشکیل شده است که دارای هشت گروه می‌باشد. هر گروه دارای شرایط مختلف تصویربرداری مانند تغییر زاویه دید، تغییر مقیاس، میزان تاری[3]، میزان فشرده‌سازی و تغییر روشنایی می‌باشد. برخی از این تصاویر را می‌توانید در شکل (۴-۱۱) ببینید.

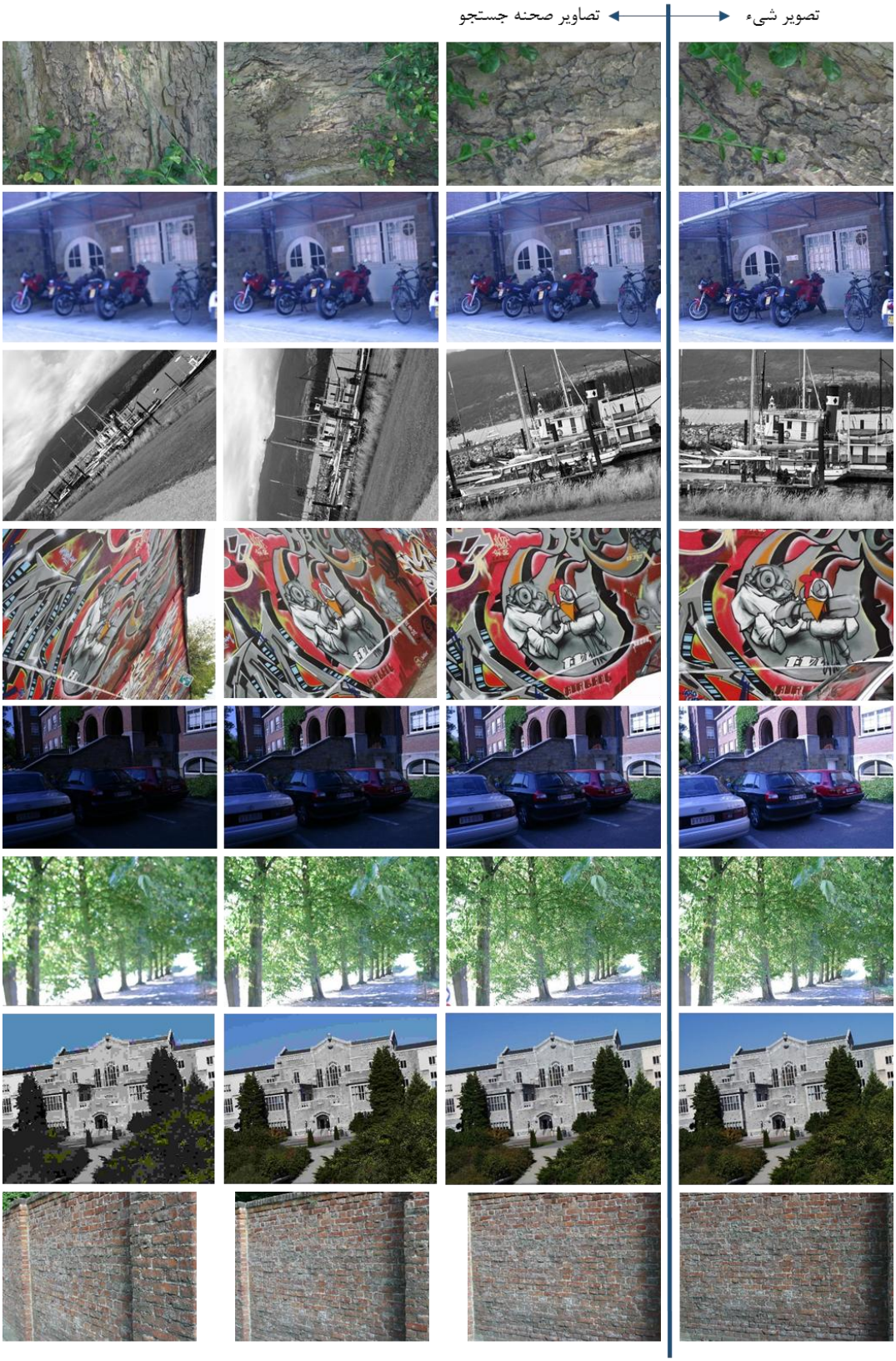
آزمایش انجام گرفته بدین صورت است که روی تصاویر هر گروه به صورت جداگانه نتایج را بدست آورده و سپس در همان گروه میانگین گیری می‌کنیم یا به بیانی دیگر نتایج مربوط به یک گروه از میانگین نتایج تصاویر آن بدست می‌آید. در پایان میزان بازشناسی روی کل تصاویر هم برای SURF و هم برای SIFT حدود ۹۰ درصد بود ولی سرعت محاسباتی SURF در مقایسه با SIFT بهتر بود که در ادامه برخی از نتایج به صورت مقایسه‌ای خواهد آمد.

در شکل (۴-۱۲) نمودار مربوط به زمان لازم برای توصیف شیء توسط این دو توصیفگر آمده است. همان طور که مشاهده می‌گردد سرعت محاسباتی SURF بسیار بهتر از SIFT می‌باشد.

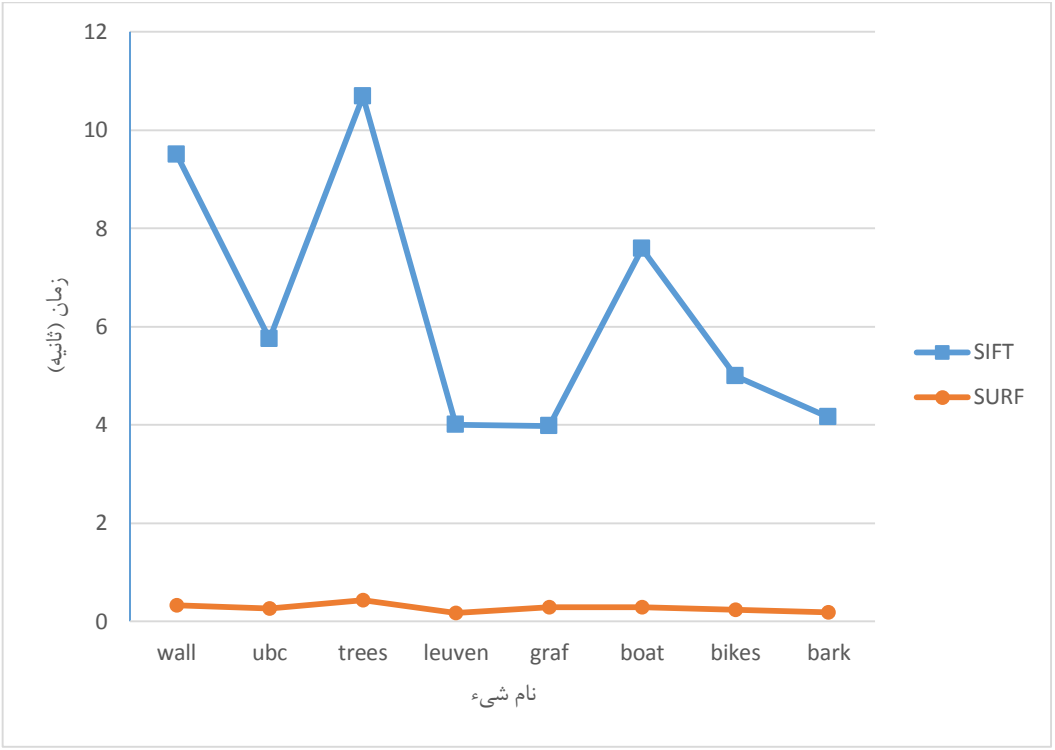
[1] http://www.robots.ox.ac.uk/~vgg/research/affine/index.html

[2] Textured scenes

[3] Blur

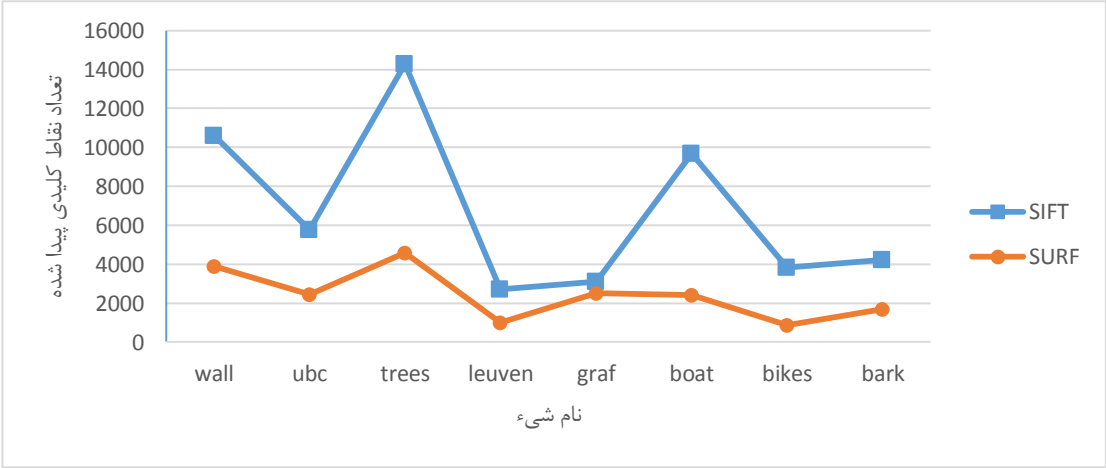


شکل (۴-۱۱) : برخی از تصاویر مربوط به پایگاه داده Mikolajczyk . ستون اول سمت راست مربوط به اشیاء مورد نظر بوده و بقیه ستون‌ها تصاویر صحنه جستجو می‌باشند. هر سطر متعلق به یک گروه است که نام آنها به ترتیب از سطر اول bark، bikes، boat، graf، leuven، trees، ubc و wall می‌باشد.



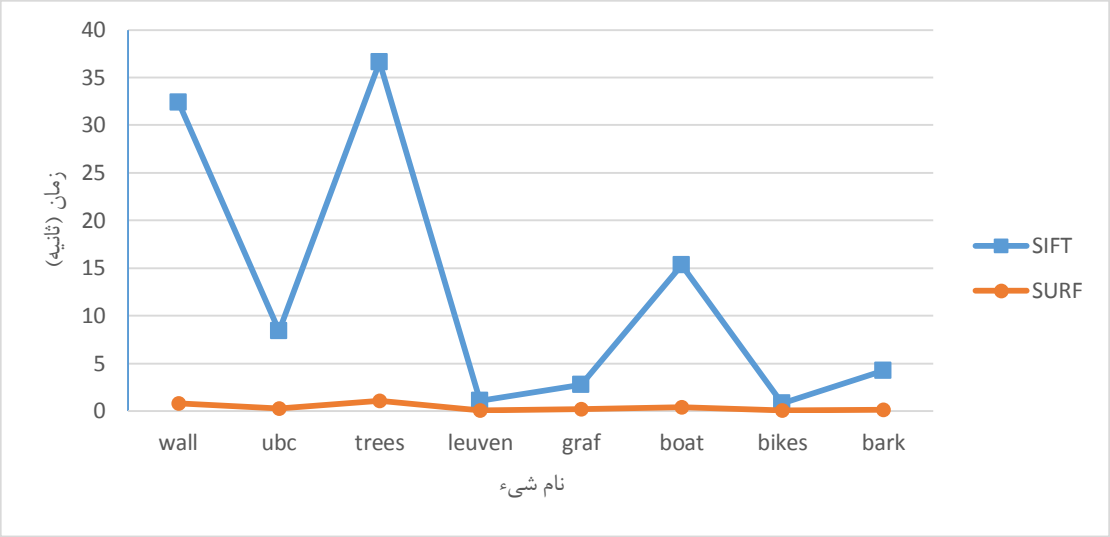
شکل (۴-۱۲) : نمودار مقایسه‌ای زمان لازم برای توصیف شیء توسط دو توصیفگر SIFT و SURF بر حسب ثانیه

در شـکل (۴-۱۳) می‌توانید تعداد نقاط کلیدی پیدا شـده توسـط دو توصیفگر را ببینید. همان طور که مشاهده می‌شود تعداد نقاط پیدا شده توسط SIFT بیشتر از SURF می‌باشد که برای پیدا کردن شیء در تصاویر پیچیده بهتر می‌باشد ولی بار محاسباتی بیشتری را طلب می‌کند.

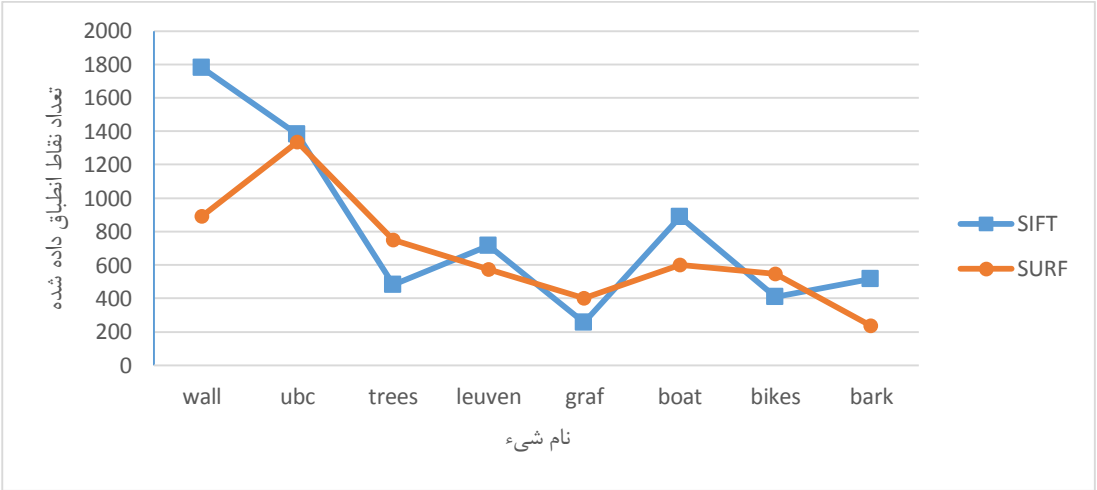


شکل (۴-۱۳) : نمودار مقایسه‌ای تعداد نقاط کلیدی پیدا شده توسط دو توصیفگر SIFT و SURF

شـکل (۴-۱۴) و (۴-۱۵) مربوط به زمان محاسـبه و تعداد نقاط در مرحله تطبیق می‌باشند. همان طور که مشـاهده می‌گردد با توجه به این که زمان محاسـبه انطباق ویژگی‌ها برای توصیفگر SIFT به خاطر تعداد زیاد نقاط کلیدی از SURF بیشـتر می‌باشـد ولی در بعضـی موارد تعداد نقاط تطبیق داده شـده توسط SURF بیشتر است.

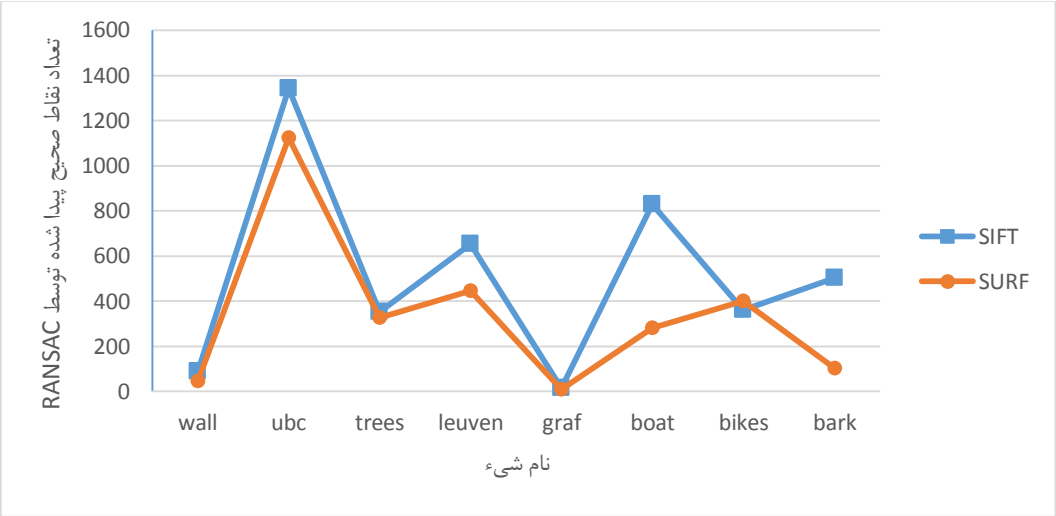


شکل (۴-۱۴) : نمودار مقایسه‌ای زمان لازم برای انطباق ویژگی‌های بدست آمده از تصویر شیء و تصویر جستجو توسط دو توصیفگر SIFT و SURF بر حسب ثانیه

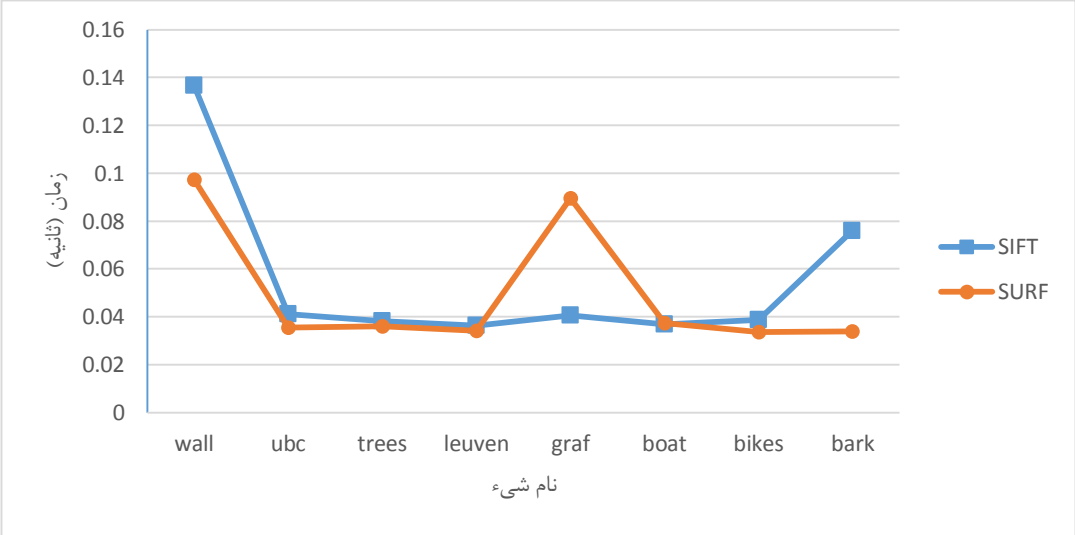


شکل (۴-۱۵) : نمودار مقایسه‌ای تعداد نقاط انطباق داده شده توسط دو توصیفگر SIFT و SURF

شـکل (۴-۱۶) و (۴-۱۷) زمان محاسـبه و تعداد نقاط صـحیح پیدا شـده توسـط RANSAC را نشـان می‌دهد که تقریباً برای هر دو توصیفگر در یک محدوده می‌باشد.



شکل (۴-۱۶) : نمودار مقایسه‌ای تعداد نقاط صحیح پیدا شده توسط RANSAC بعد از اعمال دو توصیفگر SIFT و SURF

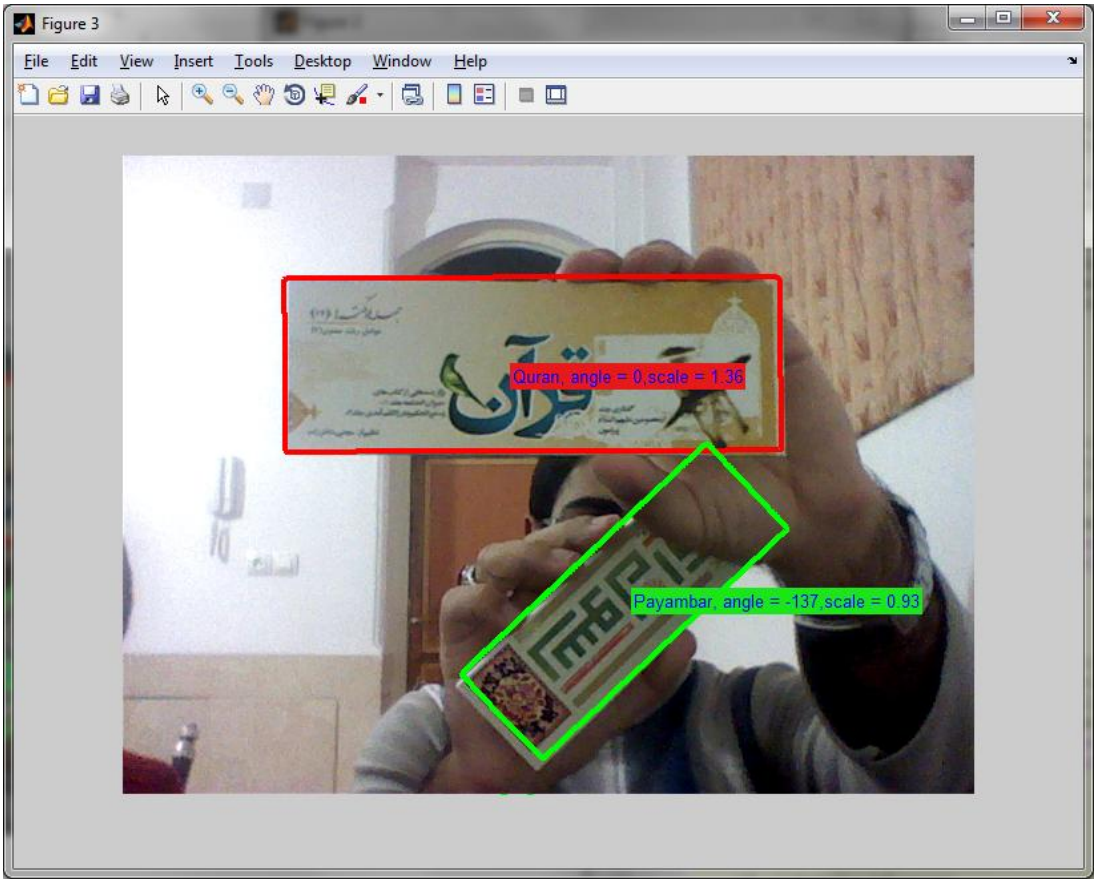


شکل (۴-۱۷) : نمودار مقایسه‌ای زمان لازم برای محاسبه RANSAC بعد از پیدا شدن نقاط کلیدی توسط دو توصیفگر SIFT و SURF

از مقایسـه نتایج بدسـت آمده می‌توان به این جمع بندی رسـید که از لحاظ پیچیدگی و سـرعت محاسـباتی توصـیفگر SURF بسـیار بهتر از SIFT می‌باشـد. از لحاظ دقت هم هر دو توصـیفگر نتایج

خوبی داشتند و واقعاً نمی‌توان گفت که کدام‌یک بهترند و همان طور که در شکل (۴-۱۵) مشاهده شد هر کدام در بعضی موارد تعداد نقاط بیشتری برای تطبیق پیدا کردند.

برای پیاده‌سازی نهایی الگوریتم از نرم‌افزار متلب[۱] به عنوان بستر پیاده‌سازی و از وب‌کم[۲] به عنوان تصویر ورودی استفاده نمودیم. نمایی از اجرای برنامه را می‌توانید در شکل (۴-۱۸) مشاهده کنید. سرعت اجرای برنامه برای اندازه تصویر ۴۸۰×۶۴۰ با توجه به میزان شلوغی تصویر حدود ۱۰ تا ۱۲ فریم بر ثانیه[۳] و برای تصویری با اندازه ۲۴۰×۳۲۰ این نرخ حدود ۳۰ تا ۴۰ فریم بر ثانیه است که می‌توان گفت تقریباً به صورت بلادرنگ عمل می‌کند.



شکل (۴-۱۸) : تصویری از اجرای بلادرنگ الگوریتم با استفاده از نرم‌افزار متلب و وب‌کم به عنوان تصویر ورودی

[۱] MATLAB
[۲] Webcam
[۳] Frame per second

فصل پنجم

مقدمه‌ای بر نرم‌افزار

۵-۱) مقدمه

برد پردازنده‌ای که ما از آن استفاده کردیم EVMDM6446 نام دارد که محصول شرکت Spectrum Digital می‌باشد. پردازنده اصلی این برد از پردازنده‌های DSP شرکت TI خانواده DaVinci مدل DM6446 می‌باشد که از دو هسته DSP و ARM تشکیل شده است. این معماری دو هسته‌ای بودن این مزیت را دارد که هم بتوان از قابلیت‌های ARM برای استفاده از سیستم‌عامل و همچنین ارتباط قوی با پورت‌ها و ادوات جانبی بهره برد و هم قدرت پردازشی DSP را برای پردازش‌های سنگین به کار برد. به عنوان مثال ما در اینجا از پردازنده ARM برای گرفتن[۱] و نمایش دادن[۲] تصویر، ارتباط با شبکه و ارتباط با پورت سریال استفاده کرده و از سیستم‌عامل لینوکسی که روی آن نصب می‌باشد به عنوان یک واسط کاربری قوی بهره می‌بریم؛ پردازنده DSP هم کار اصلی پردازش تصویر را به عهده دارد. یکی از مهم‌ترین کارها برای کار با این پردازنده ایجاد یک بستر ارتباطی بین این دو پردازنده می‌باشد که بتوانند بین یکدیگر تبادل اطلاعات داشته باشند. برای این منظور شرکت TI مجموعه از رابط‌های کاربری برنامه به اسم Codec Engine (که در این پایان‌نامه برای راحتی کار از لفظ معادل فارسی آن به نام "کدک‌انجین" استفاده می‌کنیم) را ارائه داده است. کدک‌انجین این قابلیت را دارد که بتواند در هر دو هسته اجرا شده و قابلیت کنترل پردازنده‌ها را به عهده گیرد. این رابط کاربری برای ایجاد کانال ارتباطی از واسطه‌ای دیگر به نام DSP/BIOS Link بهره می‌برد. از آنجا که این واسط فقط برای لینوکس ارائه شده است لذا برنامه اصلی باید در این محیط نوشته شود. همچنین ابزارهای ذکر شده برای اجرا روی پردازشگر به یک سیستم‌عامل بلادرنگ که توسط TI به نام DSP/BIOS ارائه شده نیاز دارند. در این فصل درباره این موارد بیشتر توضیح داده خواهد شد.

---

[۱] Capture

[۲] Display

۵-۲) پردازشگرهای دیجیتال

در دهه ۷۰ میلادی همزمان با ساخت اولین پردازنده‌ها توسط شرکت‌های مختلف، شرکت Texas Instrument تصمیم گرفت پردازنده‌هایی را مخصوص سیگنال طراحی و روانه بازار کند. این پردازنده‌ها که بیشتر با نام پردازنده‌های DSP[۱] معروف هستند، همگی با نام TMS320 شروع می‌شوند. پردازنده‌های DSP در طول ۴۰ سال از حضورشان بسیار تکامل یافته و امروزه وارد حوزه‌های مختلفی شده‌اند. اولین سری این پردازنده‌ها با نام TMS320C10 به بازار عرضه شد. پس از چند سال، حضور سری TMS320C25 باعث معروف شدن DSP ها گردید. این پردازنده‌ها که در اواخر دهه ۷۰ وارد بازار شد می‌توانست یک تبدیل فوریه را با سرعتی انجام دهد که ۲۰ سال بعد اولین سری‌های پردازنده‌های پنتیوم ساخت شرکت اینتل به آن سرعت رسیدند [۴۰].

۵-۲-۱) پردازنده‌های مهم شرکت TI

با ورود JTAG[۲] به عرصه پردازنده‌ها، شرکت TI نیز یک نرم‌افزار یکپارچه به نام CCS[۳] به بازار عرضه نمود. این نرم‌افزار با کمک JTAG به کامپیوتر وصل شده و کار با پردازنده‌های DSP را آسان می‌نماید. نام سری‌های جدید ساخت شرکت TI چهار رقمی شده و در سه دسته اصلی تقسیم‌بندی شدند [۴۰].

الف- سری ۵۰۰۰ (یا 5XXX): این سری شامل دو خانواده اصلی 54XX و 55XX می‌باشد. سری 55XX کم‌مصرف‌ترین پردازنده‌های ساخت شرکت TI می‌باشد که در بسیاری از تجهیزاتی که نیاز به قدرت پردازش بالا و جریان مصرفی کم (مانند موبایل‌ها) دارند مورد استفاده قرار گرفته‌اند. در حال حاضر عملاً طراحی سری‌های جدید برای خانواده 54XX متوقف شده و هر روز پردازنده‌های کم‌مصرف‌تری بر اساس سری 55XX به بازار عرضه می‌گردد. در سری ۵۰۰۰ سرعت پردازنده‌ها بین

---

[۱] Digital signal processing
[۲] Joint Test Action Group
[۳] Code Composer Studio

۱۰۰ تا ۳۰۰ مگاهرتز می‌باشد و در سری 55XX قدرت محاسبات ریاضی دو برابر فرکانس کاری پردازنده است. یعنی سری 55XX می‌تواند تا ۶۰۰ میلیون ضرب را در یک ثانیه انجام دهد.

کاربرد اصلی پردازنده‌های سری ۵۰۰۰ در پردازش صوت و الگوریتم‌هایی که نیاز به پردازش‌های پرسرعت دارند می‌باشد. از بعضی از سری‌ها که حجم حافظه داخلی آن‌ها بیشتر از ۱۲۸ کیلوبایت است می‌توان برای برخی کاربردهای پردازش تصویر نیز استفاده نمود.

ب- سری ۲۰۰۰ (2XXX): این سری شامل دو خانواده اصلی 24XX و 28XX می‌باشد. سری 28XX یک خانواده با عملکردی نزدیک به میکروکنترلرها می‌باشد. این سری تنها سری است که در آن حافظه فلش وجود دارد. وجود حافظه فلش داخلی، برنامه‌ریزی این پردازنده‌ها را نسبت به دیگر خانواده‌ها آسان‌تر نموده است. در این خانواده‌ها حافظه داخلی از نوع SRAM و حجم آن کمتر از ۳۲ کیلو می‌باشد. به همین دلیل این سری‌ها برای پردازش تصویر مناسب نمی‌باشند. کاربرد اصلی این سری بیشتر به عنوان یک میکروکنترلر پرسرعت می‌باشد.

ج- سری ۶۰۰۰ (6XXX): این سری شامل سه خانواده اصلی 62XX، 64XX و 67XX هستند. این خانواده‌ها پیشرفته‌ترین پردازنده‌های ساخت شرکت TI می‌باشند. در این خانواده‌ها فرکانس کاری پردازنده بین ۱۵۰ مگاهرتز تا ۱/۲ گیگاهرتز می‌باشد اما سرعت واقعی این پردازنده‌ها ۸ برابر کلاک کاری آن‌ها است. در این پردازنده‌ها در هر کلاک تا حداکثر ۸ دستور به شکل همزمان قابل اجرا بوده، به همین دلیل این پردازنده‌ها می‌توانند تا حدود ۱۰ گیگا دستورالعمل در ثانیه (GIPS)[1] را اجرا نمایند. این خانواده برای تمامی انواع پردازش‌های پرسرعت مناسب هستند اما سری 64XX با قابلیت‌های خاص آن مناسب‌ترین سری برای پردازش تصویر می‌باشد. در بین خانواده‌های مختلف، پیچیده‌ترین سری از نظر طراحی سخت‌افزار، سری ۶۰۰۰ می‌باشد. سری‌های ۲۰۰۰ و ۵۰۰۰ از نظر طراحی سخت‌افزار تقریباً پیچیدگی یکسانی دارند.

[1] Giga Instruction Per Second

البته تقسیم‌بندی بالا یک تقسیم‌بندی کلی می‌باشد و هر کدام از سری‌های ذکر شده خود دارای مدل‌های مختلفی می‌باشند که در زیر به برخی از آن‌ها اشاره می‌کنیم:

- **پردازنده‌های چند هسته‌ای KeyStone** . این سری شامل دو خانواده اصلی C665X و C667X می‌باشد. این خانواده‌ها در یک محدوده وسیع از دستگاه‌هایی که نیاز به بازدهی بالا و مصرف و هزینه کم دارند، می‌تواند مورد استفاده قرار گیرد. پردازنده‌های این سری از معماری چند هسته‌ای (۱، ۲، ۴ و ۸ هسته) با سرعتی بیش از ۱/۲۵ گیگاهرتز بهره می‌برند. کاربرد این پردازشگرها در بینایی ماشین، اندازه‌گیری‌ها، کارهای امنیتی، محاسبات ابری، محاسبات با عملکرد بالا و ... می‌باشد.[۱]
- **پردازنده‌های ویدئو DaVinci** . این پردازنده‌ها که یک ترکیب از پردازنده‌های DSP و ARM می‌باشند برای استفاده در سیستم‌های ویدئو دیجیتال مانند پردازش ویدئو، تصویر، کاربردهایی بینایی مانند سیستم نظارت و دیدبانی منازل و ... بهینه شده‌اند. این پردازنده‌ها شامل خانواده‌های DM3X، DM37X، DM38X، DM64X، DM64X DSP و DM81X است. پردازنده‌ای که در این پایان‌نامه از آن استفاده شده است از این سری و از خانواده DM64X می‌باشد. این پردازنده که TMS320DM6446 نام دارد از نوع دو هسته‌ای به صورت یک هسته DSP سری C64X و یک هسته ARM9 می‌باشد.[۲] در ادامه با این نوع پردازنده و قابلیت‌های آن بیشتر آشنا خواهید شد.

[۱] http://www.ti.com/lsds/ti/dsp/keystone/products.page

[۲] http://www.ti.com/lsds/ti/dsp/video_processors/products.page

۵-۲-۲) پردازنده TMS320DM6446

TMS320DM6446 (که بیشتر با نام DM6446 مطرح می‌شود) اوج فناوری DaVinci™ شرکت TI جهت پاسخگویی به پردازش سیستم‌های تحت شبکه و کُدگذاری[۱] و کُدگشایی[۲] اطلاعات که نیاز دستگاه‌های نسل بعدی هستند را نشان می‌دهد [۴۱].

DM6446 این توانایی را به سازندگان دستگاه‌های پردازشی می‌دهد تا بتوانند با سرعت، دستگاه‌هایی به بازار عرضه کنند که دارای ویژگی‌هایی مانند پشتیبانی از سیستم‌عامل‌های قدرتمند، رابط کاربری غنی، عملکرد پردازشی بالا و عمر باتری طولانی همراه با حداکثر انعطاف‌پذیری باشند.

معماری دو هسته‌ای DM6446 مزایای استفاده از DSP و فناوری کاهش مجموعه دستورالعمل کامپیوتر (RISC)[۳] را فراهم می‌کند و این کار را با ادغام هسته TMS320C64x+™ DSP با کارایی بالا و هسته ARM926EJ-S انجام می‌دهد.

ARM926EJ-S یک هسته پردازنده RISC ۳۲ بیتی است که دستورالعمل‌های ۳۲ بیتی یا ۱۶ بیتی را اجرا و داده‌های ۳۲ بیتی، ۱۶ بیتی یا ۸ بیتی را پردازش می‌کند. هسته از خط لوله[۴] استفاده می‌کند به طوری که تمام قسمت‌های پردازنده و سیستم حافظه می‌توانند به طور پیوسته فعالیت کنند.

هسته ARM شامل موارد زیر است:

- یک پردازنده کمکی ۱۵(CP15) و ماژول حفاظت.
- واحدهای مدیریت حافظه برنامه و داده‌ها (MMUs)[۵].

---

[۱] Encode
[۲] Decode
[۳] Reduced Instruction Set Computer
[۴] Pipeline
[۵] Data and program Memory Management Units

- دستورالعمل‌های ۱۶ کیلوبایتی جداگانه و حافظه‌های نهان[۱] داده ۸ کیلوبایتی. هر دو چهار-راه شرکت‌پذیر[۲] با شاخص‌ها و برچسب‌های مجازی هستند (VIVT)[۳].

DSP های TMS320C64x+™، بالاترین عملکرد DSP ممیز ثابت[۴] را در پلتفرم‌های سری TMS320C6000™ دارند. مبنای این پردازنده‌ها مبتنی بر نسخه بهبودیافته از نسل دوم با عملکرد بالا و معماری پیشرفته VLIW[۵] توسعه‌یافته توسط TI می‌باشد که هسته‌های این DSP ها را به گزینه‌ای عالی برای کاربردهای رسانه‌های دیجیتال تبدیل می‌کند. C64x یک عضو از پلتفرم DSP سری C6000™ است که از لحاظ نرم‌افزاری و کد کاملاً با آن سازگار می‌باشد. پردازنده TMS320C64x+ یک DSP بهبودیافته C64x+™ DSP با قابلیت‌های اضافه شده و مجموعه دستورالعمل گسترش یافته است.

هسته C64x+ با عملکردی تا سقف ۶۴۸۰ میلیون دستورالعمل در ثانیه (MIPS)[۶] در یک نرخ کلاک ۸۱۰ مگاهرتزی، راه چاره‌ای برای چالش‌های برنامه‌نویسی DSP که نیاز به عملکرد بالا دارند را ارائه می‌دهد. هسته DSP دارای انعطاف‌پذیری عملیاتی کنترلرهای با سرعت بالا و همچنین قابلیت شمارشی[۷] پردازنده‌های آرایه‌ای می‌باشد. پردازنده C64x+ با هسته DSP دارای ۶۴ ثبات همه‌منظوره[۸] با طول کلمه ۳۲ بیت و هشت واحد عملکردی مستقل (شامل دو ضرب کننده برای یک نتیجه ۳۲ بیتی و شش واحد محاسبات منطقی (ALUs)[۹]) بوده و همچنین شامل دستورالعمل‌هایی برای سرعت

---

[۱] Caches
[۲] four-way associative
[۳] Virtual index virtual tag
[۴] Fixed-point
[۵] very-long-instruction-word
[۶] Million Instruction Per Second
[۷] Numerical capability
[۸] General-purpose registers
[۹] Arithmetic Logic Units

بخشیدن به عملکرد در کاربردهای تصویری و ویدئویی می‌باشد. هسته DSP می‌تواند چهار ضرب و جمع (MACs)[۱] ۱۶ بیتی را در هر چرخه[۲] انجام دهد که منجر به اجرای ۳۲۴۰ میلیون MACs در هر ثانیه (MMACS)[۳] شود، یا اینکه هشت MACs ۸ بیتی را در هر چرخه انجام دهد که این بار می‌تواند ۶۴۸۰ MMACS تولید کند.

DM6446 همچنین دارای واحد سخت‌افزار مختص کاربردهای خاص[۴]، حافظه بر روی تراشه[۵] و لوازم جانبی بر روی تراشه اضافی شبیه به دیگر دستگاه‌های پلتفرم DSP سری C6000 می‌باشد. هسته DM6446 از معماری مبتنی بر حافظه نهان دو سطحی استفاده می‌کند. حافظه برنامه سطح ۱ (L1P) یک حافظه نهان نگاشت شده مستقیم ۲۵۶ کیلوبیتی و حافظه داده سطح ۱ (L1D) یک حافظه نهان مجموعه شرکت‌پذیر دو راهی ۶۴۰ کیلوبیتی است. سطح ۲ حافظه/کش (L2) شامل فضای حافظه ۵۱۲ کیلوبیتی می‌باشد که بین برنامه و فضای داده به اشتراک گذاشته شده است. حافظه L2 می‌تواند به عنوان حافظه نگاشت شده، حافظه نهان یا ترکیبی از هر دو پیکربندی شود.

مجموعه جانبی شامل این موارد می‌باشد: پورت‌های ویدئویی با قابلیت تنظیم و پیکربندی؛ یک پورت شبکه MAC (EMAC)[۶] با سرعت ۱۰۰/۱۰ مگابیت بر ثانیه همراه با ماژول مدیریت داده‌های ورودی/خروجی (MDIO)[۷]؛ یک رابط کاربری گذرگاه ارتباط بین مداری (I2C)[۸]؛ یک پورت سریال صوتی (ASP)؛ دو زمان‌سنج[۹] همه‌منظوره ۶۴ بیتی که هر کدام می‌توانند به عنوان ۲ زمان‌سنج ۳۲ بیتی مستقل تنظیم شوند؛ یک زمان‌سنج نگهبان[۱۰] ۶۴ بیتی؛ بیش از ۷۱ پین همه‌منظوره

---

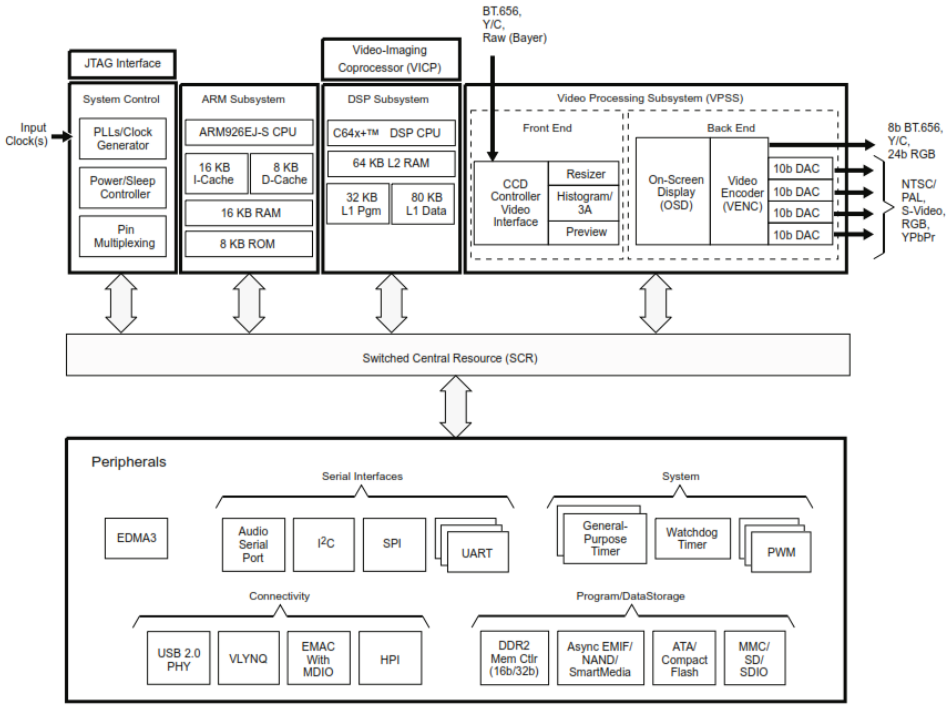
[۱] Multiply-accumulates
[۲] Cycle
[۳] Million MACs per second
[۴] application-specific hardware logic
[۵] on-chip memory
[۶] Ethernet MAC
[۷] Management Data Input/output
[۸] Inter-integrated circuit
[۹] Timer
[۱۰] Watchdog timer

ورودی/خروجی (GPIO)[۱] بـا قـابلیـت تولیـد وقفـه/رویداد[۲] قابل‌برنامه‌ریزی، ترکیب شـــده[۳] با دیگر دســتگاه‌های جانبی؛ سه UART با پشتیبانی سخت‌افزاری دست دهی[۴] روی یک UART؛ دستگاه‌های جانبی مدولاسیون پهنای پالس (PWM)[۵]؛ و دو رابط کاربری حافظه خارجی: یکی رابط حافظه خارجی ناهم‌زمان[۶] (EMIFA)[۷] برای حافظه/دســتگاه‌های جانبی کندتر و یکی رابط حافظه همزمان[۸] ســرعت بالاتر برای DDR2. بلوک دیاگرام کلی این پردازنده را می‌توانید در شکل (۵-۱) مشاهده کنید.

DM6446 دارای مجموعه کاملی از ابزارهای توسـعه برای هر دو پردازنده ARM و DSP می‌باشـد. این موارد شـامل کامپایلرهای C، یک بهینه‌ساز[۹] اسمبلی DSP برای ساده‌سازی برنامه‌نویسی و برنامه‌ریزی و یک رابط کاربری اشکال‌زدایی[۱۰] ویندوز برای مشاهده نحوه اجرای کدها است.

---

[۱] general-purpose input/output
[۲] interrupt/event
[۳] Multiplexed
[۴] handshaking
[۵] Pulse width modulator
[۶] Asynchronous
[۷] External memory interface
[۸] Synchronous
[۹] optimizer
[۱۰] debugger

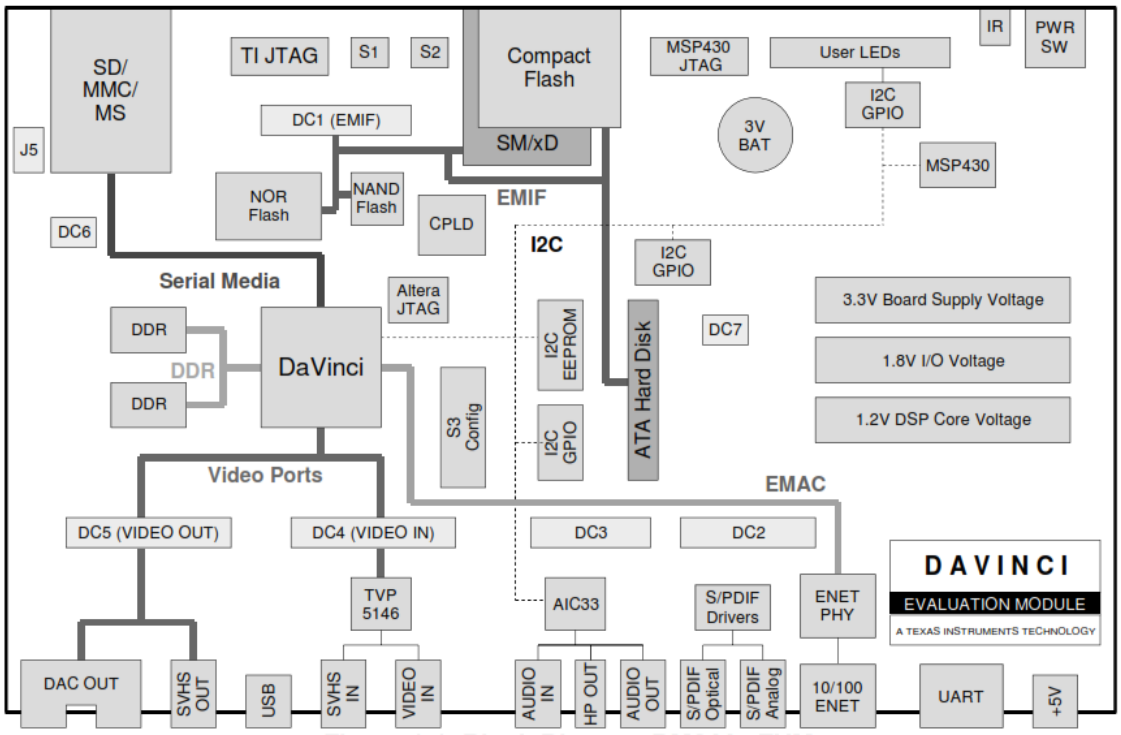


شکل (۵-۱) : بلوک دیاگرام TMS320DM6446 [41]

DM644x EVM (۳-۵

DM644x Evaluation Module یا به‌اختصار DM644x EVM یک پلتفرم توسعه مستقل[۱] است که به کاربر این امکان را می‌دهد که برنامه خود را برای خانواده پردازنده‌های DaVinci شرکت TI توسعه دهد. مبنای این EVM بر اساس پردازنده DM644x می‌باشد [۴۲].

[۱] Standalone



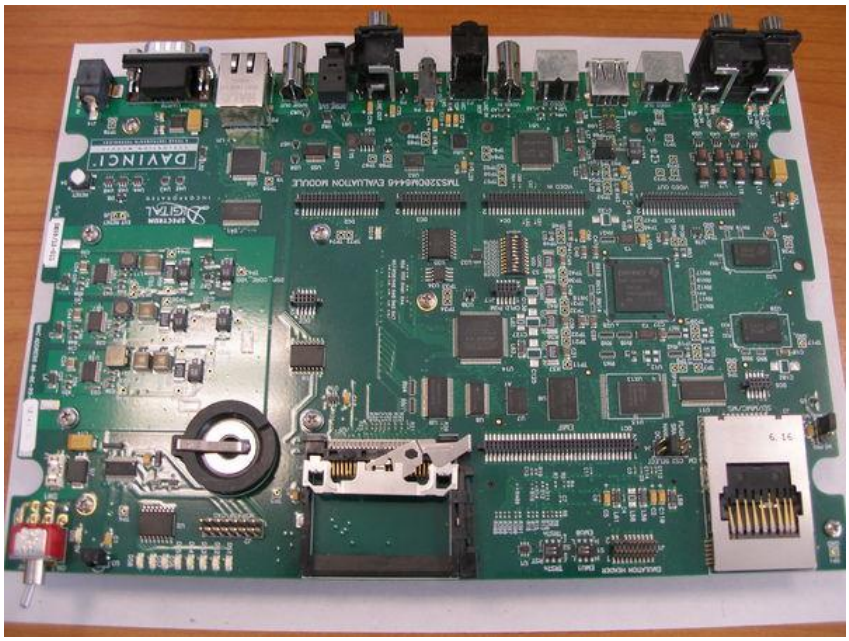
شکل (۵-۲) : بلوک دیاگرام DM644x EVM [42]

برد EVM با مؤلفه‌های کاملی از تجهیزات قابل نصب روی دستگاه‌ها به بازار آمده که مناسب محدوده وسیعی از کاربردها می‌باشد. ویژگی‌های کلیدی این برد عبارت‌اند از:

- ۱ پورت ورودی ویدئو که از قالب‌های composite و S video پشتیبانی می‌کند.
- ۴ خروجی مبدل دیجیتال به آنالوگ ویدئو،RGB و composite
- ۲۵۶ مگابایت حافظه DDR2
- UART، واسط کارت‌های حافظه (کارت SD، کارت xD، کارت SM، کارت MS، MMC)
- کدک استریو AIC33
- رابط کاربری USB2
- رابط کاربری شبکه ۱۰۰/۱۰ مگابیت بر ثانیه

- رابط کاربری کنترل از را دور توسط کنترل[۱]، ساعت دقیق، همراه با MSP430
- تنظیمات قابل پیکربندی هنگام راه‌اندازی[۲]
- واسط کاربری JTAG
- ۸ LED قابل‌برنامه‌ریزی

در شـکل (۵-۲) می‌توانید نمای کلی این برد به صـورت بلوک دیاگرام و در شکل (۵-۳) شکل ظاهری آن را مشاهده کنید.



شکل (۵-۳) : برد EVMDM6446

DSP/BIOS (۴-۵

DSP/BIOS یک کرنل بلادرنگ مقیاس‌پذیر[۳] است. این کرنل به منظور استفاده در برنامه‌های کاربردی که نیازمند برنامه‌ریزی و هماهنگ‌سازی بلادرنگ هستند، ارتباط‌های میزبان-به-هدف[۴] یا اندازه‌گیری

[۱] IR Remote Interface
[۲] Configurable boot load
[۳] Scalable
[۴] Host-to-target

بلادرنگ طراحی شده است. DSP/BIOS خصوصیات multi-threading، مستقل از سخت‌افزار بودن، تجزیه و تحلیل بلادرنگ و ابزارهای پیکربندی را فراهم می‌کند [۴۳].

۵-۴-۱) ویژگی‌ها و مزایای DSP/BIOS

DSP/BIOS برای به حداقل رساندن میزان حافظه و CPU مورد نیاز بر روی هدف[۱] طراحی شده است. این هدف طراحی از طریق روش‌های زیر حاصل می‌شود:

- تمامی اشیاء DSP/BIOS می‌توانند به صورت ثابت[۲] و محدود به فایل یک برنامه اجرایی پیکربندی شوند؛ که این باعث کاهش اندازه کد و بهینه‌سازی ساختار داده‌های داخلی می‌شود.
- داده اندازه‌گیری (مانند log ها و trace ها) روی میزبان تشکیل می‌شوند.
- رابط‌های کاربری برنامه متشکل از بخش‌هایی[۳] هستند که می‌توان آن‌ها را کم یا زیاد کرد، بنابراین تنها رابط‌های کاربری که توسط برنامه استفاده می‌شوند، لازم است که در برنامه اجرایی تعیین شوند.
- کتابخانه طوری بهینه‌سازی شده است تا کوچک‌ترین عدد ممکن از چرخه دستورالعمل[۴]، همراه با پیاده‌سازی بخش قابل ملاحظه‌ای در زبان اسمبلی را بدست آورد.
- ارتباط‌های میان هدف و ابزارهای تجزیه و تحلیل DSP/BIOS در حلقه استراحت پس‌زمینه[۵] انجام می‌شوند. این تضمین می‌کند که ابزارهای تجزیه و تحلیل DSP/BIOS با وظایف برنامه تداخلی پیدا نمی‌کنند. اگر CPU هدف برای انجام وظایف پس‌زمینه بیش از حد مشغول باشد،

---

۱ Target
۲ Statically
۳ modularized
۴ Instruction cycles
۵ Background idle loop

ابزارهای تجزیه و تحلیل DSP/BIOS دریافت اطلاعات از هدف را تا زمانی که CPU در دسترس باشد، متوقف می‌کنند.

- بررسی خطا که حافظه و پردازنده مورد نیاز را افزایش می‌دهد در حداقل میزان خود نگه داشته شده است. در عوض، مستندات مرجع API محدودیت‌هایی را برای فراخوانی توابع API مشخص می‌کند که پاسخی از توسعه‌دهندگان نرم‌افزار برای این محدودیت است.

علاوه بر این، رابط‌های کاربری برنامه DSP/BIOS گزینه‌های بسیاری را برای توسعه نرم‌افزار فراهم می‌کند:

- برنامه می‌تواند به صورت پویا[۱] اشیائی که در شرایط خاص استفاده می‌شوند را ایجاد و حذف کند. همان برنامه می‌تواند از هر دوی شیءهای ایجاد شده به صورت پویا و ثابت استفاده کند.
- مدل threading انواع thread ها را برای انواع شرایط فراهم می‌کند. وقفه‌های سخت‌افزاری، وقفه‌های نرم‌افزاری، وظیفه‌ها[۲]، توابع استراحت[۳] و توابع متناوب به صورت کامل پشتیبانی می‌شوند. شما می‌توانید اولویت‌ها[۴] و مشخصات مسدود کردن thread ها را از طریق انتخاب انواع آن‌ها کنترل کنید.
- ساختارهایی برای پشتیبانی از ارتباط‌ها و هماهنگی میان thread ها ارائه شده است که شامل semaphore ها، صندوق‌های پستی[۵] و قفل‌های منابع می‌باشند.
- دو مدل I/O برای حداکثر انعطاف‌پذیری و قدرت پشتیبانی می‌شود. لوله‌ها[۶] برای ارتباط‌های هدف/میزبان و برای پشتیبانی موارد ساده که در آن یک thread درون لوله می‌نویسد و دیگری

---

[۱] Dynamically
[۲] Tasks
[۳] Idle functions
[۴] priorities
[۵] Mailboxes
[۶] Pipes

آن را از طریق لوله می‌خواند، استفاده می‌شوند. جریان‌ها[1] برای I/O های پیچیده‌تر و پشتیبانی از درایورهای دستگاه استفاده می‌شوند.

- سیستم‌های سطح پایین اولیه برای آسان‌سازی رسیدگی به خطاها، ایجاد ساختار داده‌های رایج و مدیریت استفاده از حافظه ارائه شده است.

رابط‌های کاربری برنامه DSP/BIOS برنامه‌نویسی DSP را برای تعدادی از دستگاه‌های TI استانداردسازی کرده و ابزارهای توسعه برنامه قدرتمندی با قابلیت استفاده آسان را فراهم می‌کنند. این ابزارها زمان مورد نیاز برای ایجاد برنامه‌های DSP را با استفاده از روش‌های زیر کاهش می‌دهند:

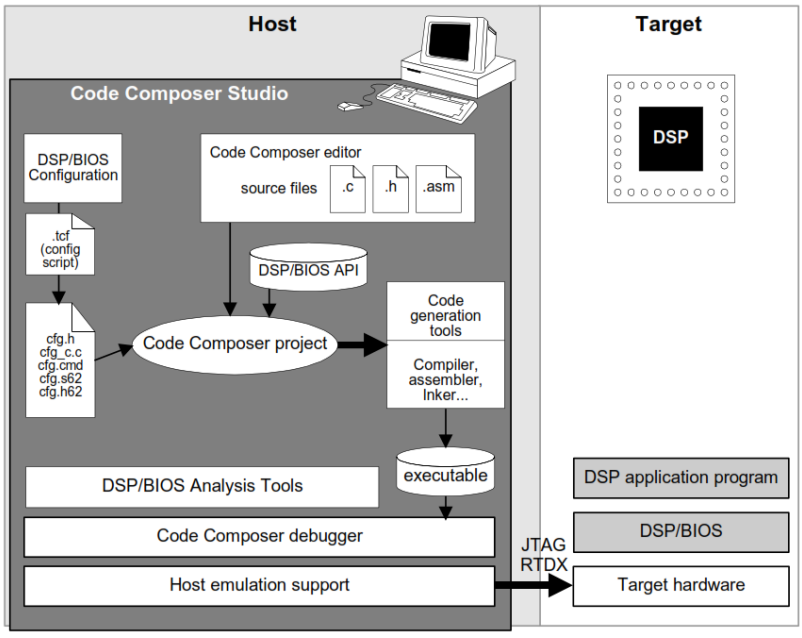
- اسکریپت پیکربندی Tconf کد مورد نیاز برای تعریف ثابت اشیاء استفاده شده در برنامه را تولید می‌کند.
- پیکربندی تشخیص خطاها توسط اعتبار سنجی ویژگی‌ها قبل از انجام برنامه ساخته شده است.
- اسکریپت پیکربندی می‌تواند در یک ویرایشگر متن شامل انشعاب، حلقه، تست آرگومان خط فرمان و غیره اصلاح شود.
- ورود و آمارگیری[2] اشیاء DSP/BIOS در زمان اجرا بدون برنامه‌نویسی اضافی در دسترس هستند. دستورالعمل‌های اضافی در صورت نیاز می‌توانند برنامه‌نویسی شوند.
- ابزارهای تجزیه و تحلیل DSP/BIOS اجازه نظارت بلادرنگ بر رفتار برنامه را می‌دهند.
- DSP/BIOS یک رابط کاربری برنامه استاندارد را ارائه می‌دهد که این به توسعه‌دهندگان الگوریتم DSP اجازه ارائه کدی را می‌دهد که به راحتی می‌تواند با دیگر توابع برنامه یکپارچه شود.

[1] Streams
[2] statistics

- DSP/BIOS در CCS یکپارچه شده است طوری که نیاز به هیچ مجوزی برای اجرا ندارد و به طور کامل توسط TI پشتیبانی می‌شود. DSP/BIOS یک جز از مؤلفه‌های فناوری نرم‌افزاری بلادرنگ eXpressDSP^{TM} شرکت TI می‌باشد.

۵-۴-۲) مؤلفه‌های DSP/BIOS

شکل (۵-۴) اجزای DSP/BIOS همراه با تولید برنامه و محیط اشکال‌زدایی Code Composer Studio را نشان می‌دهد:



شکل (۵-۴) : مؤلفه‌های DSP/BIOS [43]

رابط کاربری DSP/BIOS. شما روی کامپیوتر میزبان برنامه‌هایی (در C، C++، یا اسمبلی) را می‌نویسید که توابع رابط کاربری DSP/BIOS را فراخوانی می‌کنند.

ابزار پیکربندی DSP/BIOS. شما یک پیکربندی ایجاد می‌کنید که اشیاء ثابت استفاده شده در برنامه خودتان را معرفی می‌کند. پیکربندی فایل‌هایی را تولید می‌کند که شما آن‌ها کامپایل و با برنامه لینک می‌کنید.

ابزارهای تجزیه و تحلیل DSP/BIOS. این ابزارها در Code Composer Studio به شما اجازه تست برنامه روی دستگاه هدف در حال مشاهده بار CPU، زمان‌بندی، اجرای Thread و غیره را می‌دهد. (thread به تمامی thread های در حال اجرا اشاره دارد: وقفه سخت‌افزاری، وقفه نرم‌افزاری، وظیفه، یا توابع استراحت.)

در زیر این مؤلفه‌های DSP/BIOS بیشتر توضیح داده خواهد شد.

۵-۴-۳) کرنل بلادرنگ و رابط کاربری DSP/BIOS

DSP/BIOS یک کرنل بلادرنگ مقیاس‌پذیر است که برای برنامه‌هایی که نیازمند برنامه‌ریزی و هماهنگ‌سازی بلادرنگ، ارتباطات میزبان به هدف و یا ابزارهای اندازه‌گیری بلادرنگ می‌باشند، طراحی شده است. DSP/BIOS حالت multi-threading، مستقل از سخت‌افزار بودن، تجزیه و تحلیل بلادرنگ و ابزارهای پیکربندی را فراهم می‌کند.

رابط کاربری DSP/BIOS به ماژول‌هایی تقسیم می‌شود. اندازه DSP/BIOS با توجه به اینکه چه ماژول‌هایی توسط برنامه پیکربندی و استفاده می‌شوند، می‌تواند در حدود ۵۰۰ تا ۶۵۰۰ کلمه کد متغیر باشد. تمامی عملیات‌های لازم برای شروع کار با یک ماژول در شکل (۵-۴) نشان داده شده است.

برنامه‌های کاربردی از DSP/BIOS به وسیله فراخوانی API استفاده می‌کنند. همه ماژول‌های API رابط کاربری قابل فراخوانی در C[1] را ارائه می‌کنند. اغلب رابط‌های کاربری قابل فراخوانی در C می‌توانند از زبان اسمبلی نیز فراخوانی شوند، به شرطی که قراردادهای فراخوانی C تشریح شده باشد. برخی از رابط‌های C درواقع ماکروهای C هستند و در نتیجه، نمی‌توانند زمانی که از زبان اسمبلی فراخوانی می‌شوند، مورد استفاده قرار گیرند.

[1] C-callable

۵-۴-۴) ابزار پیکربندی DSP/BIOS

پیکربندی DSP/BIOS به شما اجازه بهینه‌سازی برنامه‌های کاربردی‌تان با ایجاد اشیاء و تنظیمات ویژگی‌هایشان به صورت ثابت، به جای زمان اجرا را می‌دهد. هر دو این موارد سبب بهبود عملکرد زمان اجرا و کاهش اثرات[۱] برنامه می‌شود.

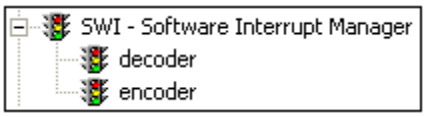
منبع فایل برای پیکربندی یک اسکریپت DSP/BIOS Tconf است که دارای پسوندی با فرمت tcf. می‌باشد. دو راه برای دسترسی به یک پیکربندی DSP/BIOS وجود دارد:

- **متنی.** شما می‌توانید متن اسکریپت را با استفاده از Code Composer Studio یا یک ویرایشگر متن جداگانه ویرایش کنید. شما پیکربندی را با استفاده از ترکیب جاوا اسکریپت[۲] رمزگذاری می‌کنید (همانند چیزی که در شکل (۵-۵) نشان داده شده است).

```
prog.module("SWI").create("encoder");
prog.module("SWI").create("decoder");
```

شکل (۵-۵) : پیکربندی DSP/BIOS به صورت متنی [43]

- **گرافیکی.** شما می‌توانید پیکربندی‌ها را در حالت فقط خواندنی با ابزار پیکربندی DSP/BIOS مشاهده کنید که این ابزار یک ویرایشگر گرافیکی می‌باشد. رابط کاربری شبیه به ویندوز اکسپلورر است. اسکریپت نشان داده شده در شکل (۵-۶) معادل کدی است که در شکل (۵-۵) ایجاد شده است.



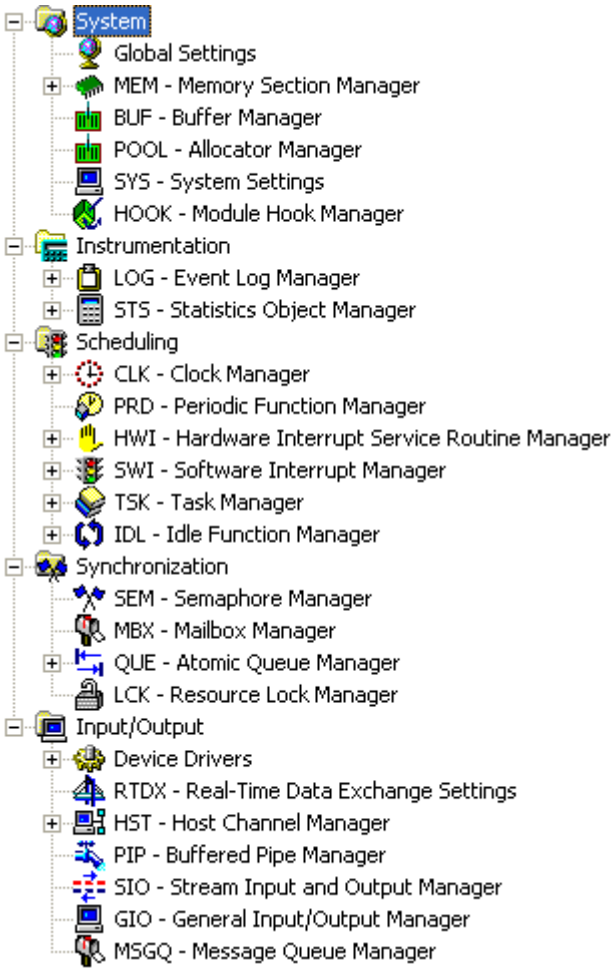
شکل (۵-۶) : پیکربندی DSP/BIOS به صورت گرافیکی [43]

---

[۱] Footprint

[۲] JavaScript syntax

شما می‌توانید طیف گسترده‌ای از پارامترهای استفاده شده توسط DSP/BIOS در زمان اجرا را تنظیم کنید. اشیائی که شما ایجاد کرده‌اید توسط فراخوانی واسط کاربری DSP/BIOS برنامه استفاده می‌شوند؛ که این اشیاء شامل وقفه‌های نرم‌افزاری، وظیفه‌ها، جریان‌های I/O و رویدادها می‌باشند. شکل (۵-۷) یک نمای درختی از ابزارهای در دسترس را نشان می‌دهد.



شکل (۵-۷) : نمودار درختی ابزارهای پیکربندی [43]

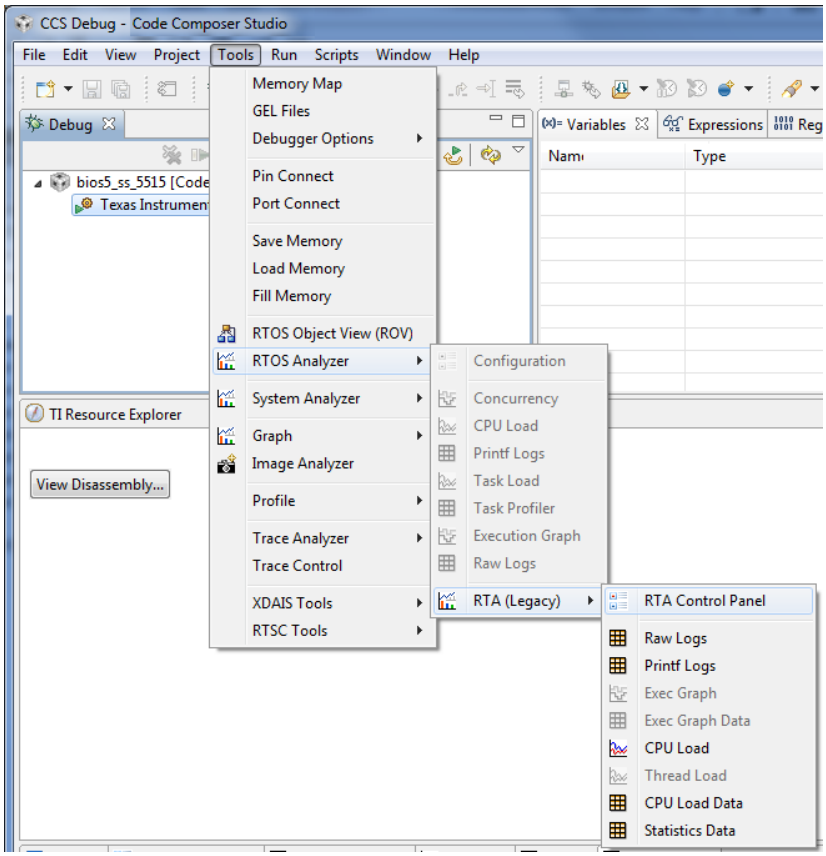
هنگامی که شما یک پیکربندی را ذخیره کنید، Tconf فایل‌هایی را به منظور استفاده در پروژه تولید می‌کند. با استفاده از پیکربندی ثابت، اشیاء DSP/BIOS می‌توانند پیش از اجرا پیکربندی و محدود به

یک فایل برنامه اجرایی شوند. همچنین، یک برنامه DSP/BIOS می‌تواند اشیاء خاصی را در زمان اجرا ایجاد و حذف کند.

۵-۴-۵) ابزارهای تجزیه و تحلیل DSP/BIOS

ابزارهای تجزیه و تحلیل DSP/BIOS به انضمام محیط Code Composer Studio بوسیله فعال کردن تجزیه و تحلیل برنامه‌های بلادرنگ DSP/BIOS در دسترس هستند. شما می‌توانید به صورت دیداری یک برنامه DSP را با کمترین تأثیر در کارایی بلادرنگ آن نظارت کنید.

در CCS ابزارهای تجزیه و تحلیل DSP/BIOS در منوی Tools>RTOS Analyzer>RTA(Legacy) قرار گرفته‌اند که در شکل (۵-۸) نشان داده شده است.



شکل (۵-۸) : ابزارهای تجزیه و تحلیل DSP/BIOS در محیط CCS [43]

بر خلاف اشکال‌زدایی سنتی که بیرون از برنامه اجرایی انجام می‌شود، تجزیه و تحلیل برنامه احتیاج به برنامه‌ی هدفی دارد که شامل سرویس‌های دستورالعمل بلادرنگ باشد. با استفاده از رابط کاربری DSP/BIOS و اشیاء آن توسعه‌دهندگان به صورت خودکار هدف را برای گرفتن و بارگذاری اطلاعات بلادرنگ روی میزبان از طریق ابزارهای تجزیه و تحلیل DSP/BIOS نرم‌افزار CCS، آماده می‌کنند.

۵-۵) کدک‌انجین

از دید توسعه‌دهندگان نرم‌افزار، کدک‌انجین[۱] مجموعه‌ای از رابط‌های کاربری برنامه[۲] است که از آن برای معرفی و اجرای الگوریتم‌های xDAIS استفاده می‌شود. یک رابط کاربری VISA برای تعامل با الگوریتم xDAIS سازگار با xDM ارائه شده است [۴۴].

API برای تمام شرایط زیر یکسان است:

- الگوریتم ممکن است به صورت محلی[۳] (در GPP)[۴] یا از راه دور[۵] (در DSP) اجرا شود.
- سیستم ممکن است یک GPP+DSP، DSP تنها، یا GPP تنها باشد.
- تمام GPP ها و DSP های پشتیبانی شده دارای API یکسان هستند.
- تمام سیستم‌عامل‌های پشتیبانی شده همانند Linux، PrOS، VxWorks، DSP/BIOS و WinCE دارای API یکسان هستند.

هر الگوریتم xDM با استاندارد رابط کاربری الگوریتم eXpressDSP (xDAIS)[۶] سازگار می‌باشد. علاوه بر این، رابط کاربری xDAIS-DM (xDM) را به عنوان یک بسط از استاندارد xDAIS به منظور فراهم

---

۱ Codec Engine
۲ APIs
۳ Local
۴ General purpose processor (در اینجا منظور هسته آرم می‌باشد)
۵ Remote
۶ eXpressDSP Algorithm Interface Standard

کردن بستری برای پشتیبانی از کُدبندها[۱]، کُدگشاها[۲] و کدک‌های رسانه‌های دیجیتال ایجاد کرده است. xDM رابط‌های کاربری برنامه برای کدک‌های رسانه دیجیتال با پسوندهای تعریف شده برای ویدیو، تصویربرداری، گفتار و کدک‌های صوتی را توسط کلاس تعریف می‌کند.

رابط‌های کاربری xDM الگوریتم‌های کدک را به چهار کلاس تقسیم می‌کنند: ویدئو، تصویر، گفتار و صوت (VISA)[۳]. VISA مربوط به این رابط‌های کاربری xDM می‌باشد. یک مجموعه از API ها به ازای هر کلاس کدک ارائه شده است؛ بنابراین، MP3 می‌تواند با WMA بدون تغییر کد برنامه جایگزین شود. بلکه تنها پیکربندی نیاز به تغییر دارد.

کدک‌انجین همچنین اجرای بلادرنگ کدک را پشتیبانی می‌کند؛ که این رابط‌های کاربری برنامه را برای دسترسی به حافظه و به طور کلی آمارهای استفاده شده توسط CPU و ردیابی اطلاعات[۴] هنگام اجرا، فراهم می‌کند.

کدک‌انجین به صورت باینری عرضه می‌شود؛ بنابراین کتابخانه‌های برنامه که با همان نسخه کدک‌انجین ارائه می‌شوند همیشه قابلیت سازگاری با آن را دارند.

۵-۵-۱) چرا باید از کدک‌انجین استفاده کنیم؟

کدک‌انجین برای حل برخی مشکلات رایج مرتبط با توسعه برنامه‌های کاربردی سیستم‌های قرار گرفته روی یک تراشه (SoC)[۵] طراحی شده است. مهم‌ترین مشکلات شامل موارد زیر است:

---

[۱] Encoders

[۲] Decoders

[۳] Video, Image, Speech, and Audio

[۴] Trace information

[۵] system-on-a-chip

- اشکال‌زدایی[1] در یک محیط پردازنده ناهمگن می‌تواند رنج‌آور باشد. اشکال‌زداهای متعدد و با راه‌اندازی پیچیده وجود دارد.
- پیاده‌سازی‌های مختلف از الگوریتم مشابه، مانند MP3، دارای رابط‌های کاربری برنامه مختلفی است. تغییر به یک الگوریتم کارآمدتر مستلزم تغییر قابل‌توجه در آن می‌باشد.
- مسأله انتقال پذیری یا قابل‌حمل بودن[2] که با دو پردازنده ترکیب شده است. شما ممکن است بخواهید از بردهای مختلفی با DSP جدیدتر یا GPP جدیدتر استفاده کنید.
- برخی الگوریتم‌ها ممکن است یا روی DSP و یا GPP بخواهند اجرا شوند. برای حفظ تعادل میزان بار سیستم، الگوریتم‌هایی با پیچیدگی کم می‌توانند روی یک GPP اجرا شوند، اما این در صورتی است که زمان کمی برای این تغییر صرف شود. اگر تغییر مکانی که الگوریتم در آن اجرا می‌شود آسان باشد، شما مجبور به ارزیابی موضوع عملکرد در برابر سختی تغییر برنامه نیستید.
- برای موفقیت در بازار، اغلب برنامه‌های کاربردی نیاز به پشتیبانی از کدک‌های متعدد برای به کار بردن در همان نوع از رسانه را دارند. برای مثال، یک برنامه ممکن است به پشتیبانی از سه یا چهار فرمت صوتی نیاز داشته باشد.
- برنامه نویسان با دید GPP (پردازنده همه‌منظوره) به طور معمول نمی‌خواهند یاد بگیرند که یک برنامه‌نویس DSP شوند و همچنین نمی‌خواهند که درباره مدیریت حافظه پیچیده DSP و مشکلات بلادرنگ DSP نگران باشند.

کدک‌انجین این مشکلات را با فراهم کردن معماری نرم‌افزاری استاندارد و رابط‌های کاربری برای اجرای الگوریتم‌ها حل کرده است. کدک‌انجین دارای خصوصیات زیر می‌باشد:

---

[1] Debugging

[2] Portability

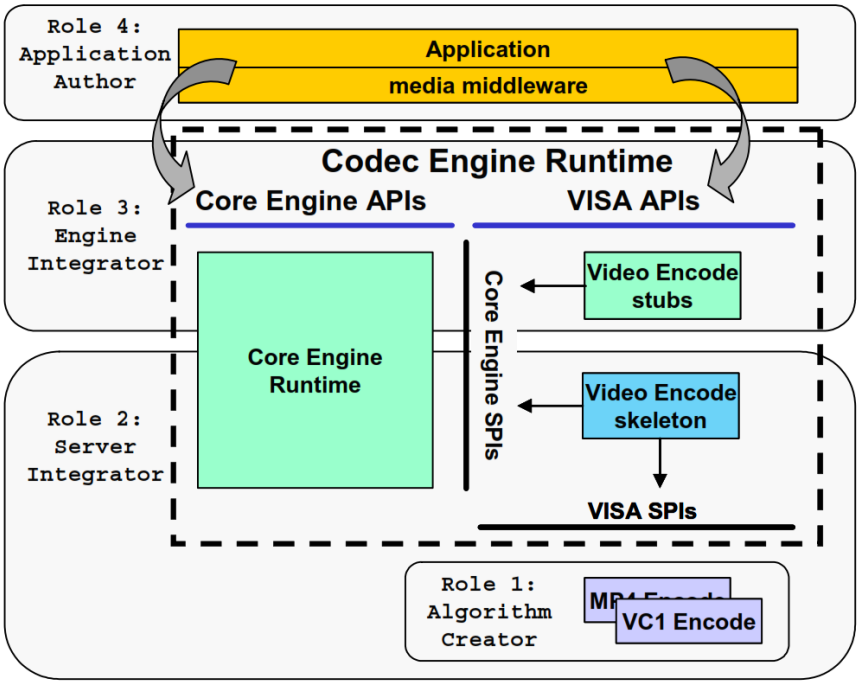
- **آسان برای استفاده**. توسعه‌دهندگان برنامه مشخص می‌کنند که چه الگوریتمی نیاز است که اجرا شود، اما به چگونگی انجام کار و در کجا اجرا شدن برنامه کاری ندارند.
- **قابلیت گسترش و پیکربندی**. هر کسی می‌تواند با استفاده از ابزارها و فن‌های[1] استاندارد الگوریتم‌های جدید اضافه کند.
- **قابل‌حمل بودن**. رابط‌های کاربری برنامه مستقل از نوع برد، پلتفرم و حتی کدک‌ها می‌باشد.

۵-۵-۲) کدک‌انجین کجا مناسب معماری من است؟

کد برنامه API کدک‌انجین را فراخوانی می‌کند. در کدک‌انجین، واسط‌های کاربری VISA از stub ها و skeleton ها برای دسترسی به هسته انجین و کدک‌های واقعی که ممکن است محلی یا از راه دور باشد، استفاده می‌کنند.

شکل (۵-۹) معماری عمومی برنامه کاربردی که از کدک‌انجین استفاده می‌کند را نشان می‌دهد. همچنین نقش‌های کاربر در ایجاد بخش‌های مختلف نرم‌افزار را نشان می‌دهد.

[1] Technique

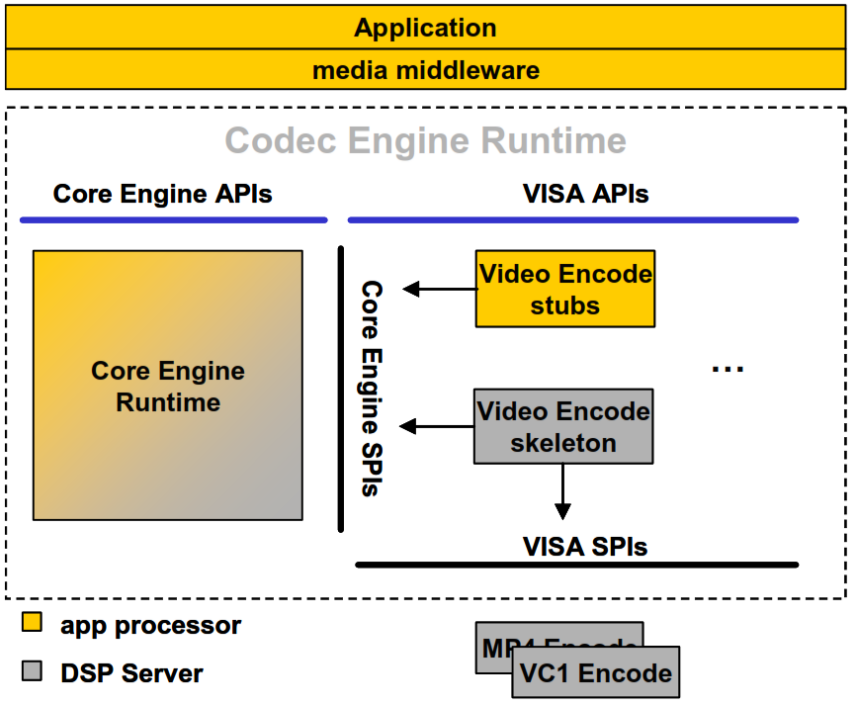


شکل (۵-۹) : نحوه استفاده برنامه‌ها از کدک‌انجین [44]

برنامه، رابط‌های کاربری کدک‌انجین و رابط‌های کاربری VISA را فراخوانی می‌کند. رابط‌های کاربری VISA از stub ها برای دسترسی به SPI های (رابط‌های برنامه‌نویسی سیستم)[۱] هسته انجین استفاده می‌کند. skeleton ها به SPI های هسته انجین و SPI های VISA دسترسی دارند. SPI های VISA به الگوریتم اصلی دسترسی دارد.

شکل (۵-۱۰) یک مدل اصلاح‌شده از نمودار قبلی است که نشان می‌دهد چگونه این معماری در یک سیستم GPP+DSP توزیع می‌شود. در این مثال، بخش زرد روی GPP و بخش خاکستری روی DSP اجرا می‌شود. از این رو، skeleton کُدبند ویدئو و کدک‌های کُدبند ویدئو روی DSP و برنامه اجرایی و stub کُدگشا روی GPP می‌باشد.

[۱] System Programming Interfaces



شکل (۵-۱۰) : نمای دیگری از نحوه ارتباط برنامه‌ها با کدک‌انجین [44]

از آنجا که کدک‌انجین انعطاف‌پذیر است، نمودار آن می‌تواند برای سیستم‌های فقط DSP و فقط GPP نشان داده شود.

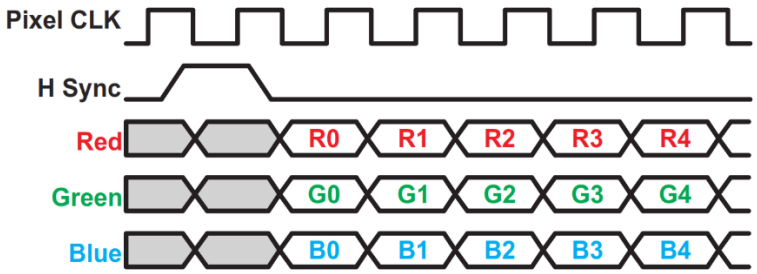
۵-۶) مقدمه‌ای بر تئوری رنگ‌ها در سخت‌افزارهای TI

سیستم‌های روی تراشه داوینچی DM6446/DM6443 شرکت TI سازگار با خروجی ویدئو دیجیتال YCbCr یا RGB می‌باشند؛ خروجی‌های YCbCr می‌توانند به یک کُدبند[۱] ویدئو یا خروجی‌های RGB می‌توانند به LCD متصل شوند. در ادامه این درباره این دو نوع استاندارد بیشتر توضیح داده خواهد شد [۴۵].

[۱] Encoder

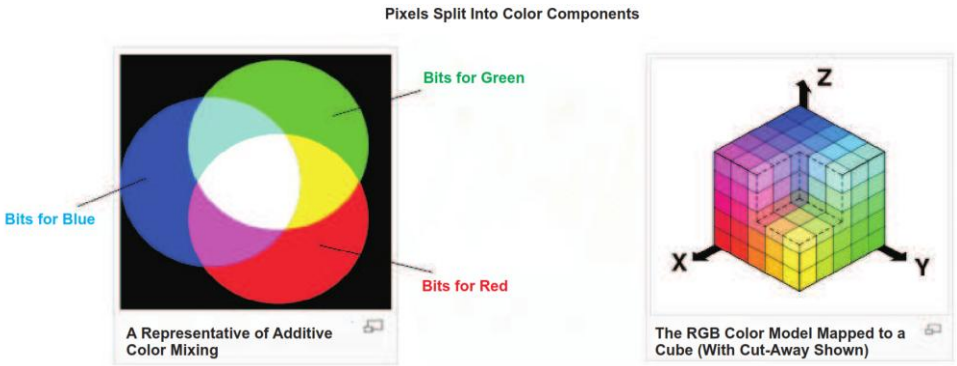
۵-۶-۱) قالب پیکسل RGB

قبل از ارائه قالب Y/C 4:2:2، ابتدا قالب پیکسل RGB را به عنوان یک حسگر دوربین در نظر گرفته و چگونگی کار در فضای رنگ RGB را نشان می‌دهیم. شکل (۵-۱۱) یک نمایش معمولی از یک پیکسل RGB در یک گذرگاه داده سخت‌افزاری می‌باشد.



شکل (۵-۱۱) : نمایش پیکسل RGB [45]

اساساً، هر پیکسل ویدئو شامل یک جزء از قرمز، سبز و آبی داده‌های ویدئویی است. رنگ حقیقی را می‌توان از طریق ترکیب این رنگ‌های اصلی همان‌طور که در شکل (۵-۱۲) نشان داده شده است، بدست آورد.



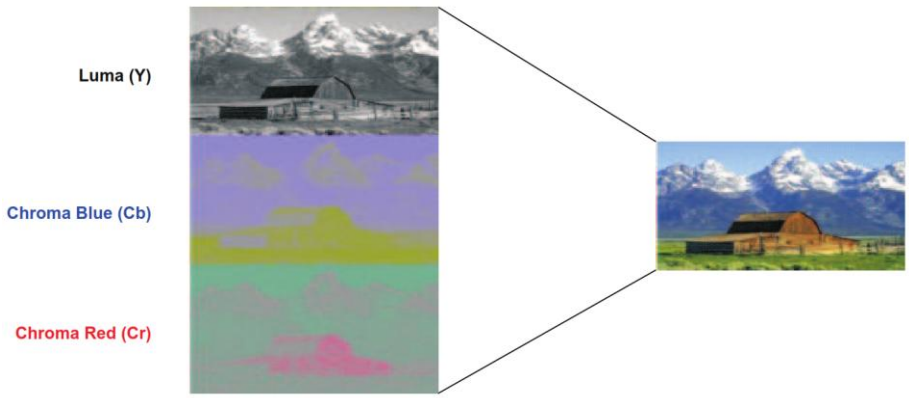
شکل (۵-۱۲) : مؤلفه‌های رنگ RGB [45]

استانداردهای رایج بسیاری برای داده پیکسل RGB، متغیر از RGB332 تا RGB888 وجود دارد. اعداد هر استاندارد مطابق با تعداد بیت اختصاص داده شده برای هر مؤلفه رنگ داده ویدئویی می‌باشند. برای

مثال، RGB332 یک استاندارد پیکسل ۸ بیتی رایج است که دارای سه بیت قرمز، سه بیت سبز و دو بیت داده آبی برای هر پیکسل می‌باشد. RGB565 استاندارد ۱۶ بیتی با پنج بیت قرمز، شش بیت سبز و پنج بیت آبی است. سرانجام، RGB888 یک استاندارد ۲۴ بیتی با هشت بیت از هر مؤلفه رنگ می‌باشد.

۵-۶-۲) سبک رنگ Y/C و قالب پیکسل ۴:۲:۲

یکی از مزایای استفاده از فضای رنگ YCbCr نسبت به RGB داشتن مؤلفه Y یا luma است که نمایانگر روشنایی بوده و سیگنال‌ها را به نمایشگرهای سیاه و سفید انتقال می‌دهد، در حالی که U و V، یا مؤلفه‌های رنگی[۱]، رنگ تصویر را نشان می‌دهند. شکل (۵-۱۳) مثالی از فضای رنگی YCbCr را نشان می‌دهد.

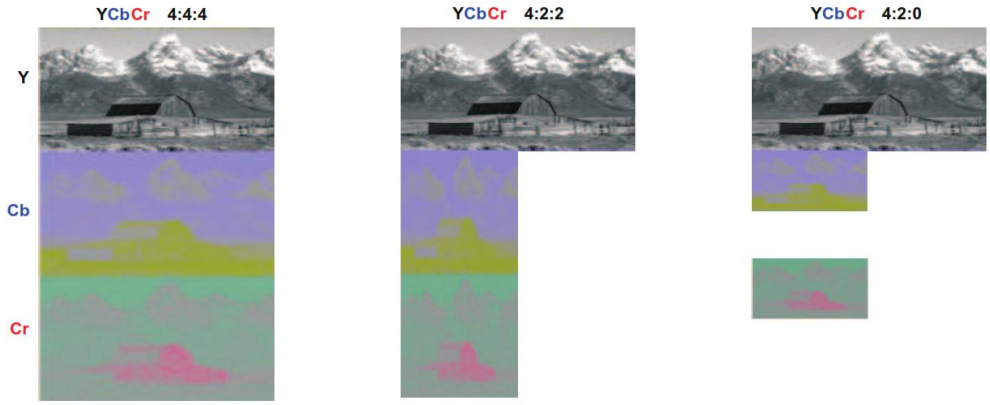


شکل (۵-۱۳) : فضای رنگ YCbCr [45]

علاوه بر این، چشم انسان به مؤلفه Y حساس‌تر از مؤلفه‌های Cb و Cr است. مانند آنچه از تصویر در شکل (۵-۱۳) دیده می‌شود، شما با نگاه کردن به تنهایی به Y یا مؤلفه luma، به راحتی تشخیص می‌دهید که تصویر یک انبار با کوه در پس‌زمینه است. با این حال، تشخیص تصویر با مؤلفه‌های رنگی آبی[۲] و رنگی قرمز[۳]، دشوار است. با در نظر گرفتن این ویژگی‌ها، شما می‌توانید با خیال راحت برخی از

[۱] Chroma components
[۲] Chroma blue
[۳] Chroma red

داده‌های رنگی را حذف و هم روی پهنای باند و هم روی فضای دیسک صرفه‌جویی داشته باشید. تنوع ارائه پیکسل YCbCr از این ویژگی بهره می‌برد (برای مثال، YCbCr 4:4:4, 4:2:2, and 4:2:0).

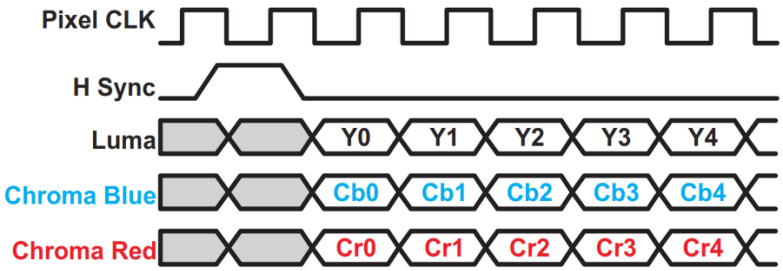


شکل (۵-۱۴) : نمایش حالت‌های مختلف پیکسل YCbCr [45]

شکل (۵-۱۴) نسبت‌های مختلف از داده luma به داده‌های رنگی که در برخی از مهم‌ترین استانداردهای Y/C رایج وجود دارد را نشان می‌دهد. در مورد YCbCr 4:4:4، نسبت داده روشنایی Y به هر کدام از مؤلفه‌های رنگی ۴:۴ یا ۱:۱ است. در مورد YCbCr 4:2:2، مقدار داده رنگی در هر خط به نصف کاهش می‌یابد. YCbCr 4:2:0 یک مرحله دیگر کاهش مقدار داده رنگی بوسیله نصف کردن در فضای عمودی، علاوه بر حالت قبل دارد. از آنجا که تمایز بین یک تصویر ساخته شده با هر یک از این سه استاندارد دشوار است، این حس را برای توسعه‌دهندگان ایجاد می‌کند که هر زمان که می‌توانند از YCbCr 4:2:0 استفاده کنند. با این حال قالب‌های دیگر مانند درایورهای ویدئو DaVinci که از YCbCr 4:2:2 استفاده می‌کنند نیز مفید هستند.

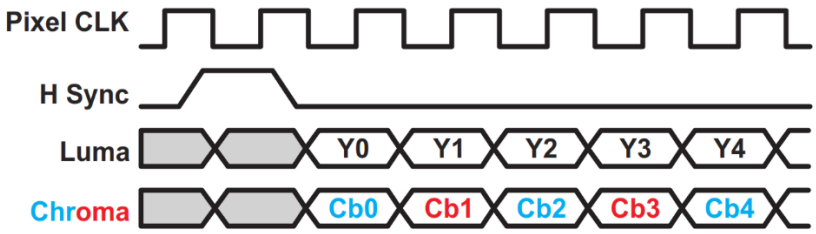
مشاهده چگونگی فرمت‌های مختلف پیکسل YCbCr به طور معمول در یک گذرگاه سخت‌افزار نشان می‌دهد که چرا محصولات DaVinci شرکت TI از YCbCr 4:2:2 به عنوان انتخاب استاندارد پورت‌های ویدئو دیجیتال استفاده می‌کنند.

شکل (۵-۱۵) یک گذرگاه داده YCbCr 4:4:4 را نمایش می‌دهد. توجه داشته باشید که این گذرگاه بسیار شبیه به گذرگاه داده RGB می‌باشد. همانند RGB، YCbCr 4:4:4 نیز برای هر کدام از سه مؤلفه به یک گذرگاه نیاز دارد.



شکل (۵-۱۵) : نمایش پیکسل YCbCr 4:4:4 [45]

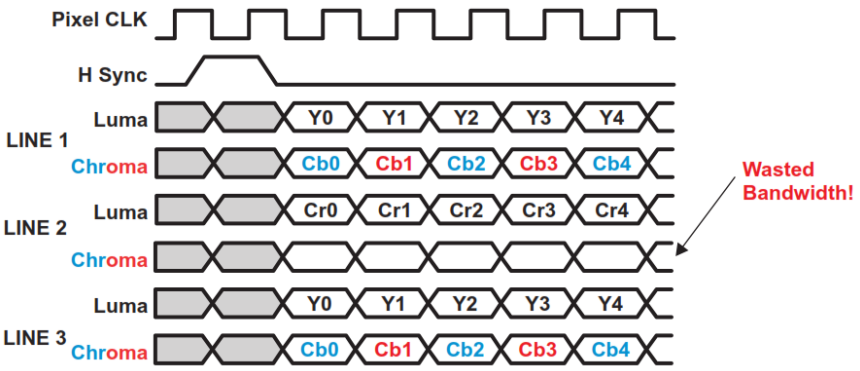
اگر برخی از داده‌های رنگی حذف شوند، مزایای آن بیشتر نمایان می‌شود (شکل ۵-۱۶). توجه داشته باشید که پیکسل‌های ۴:۲:۲ به جای سه گذرگاه، تنها نیازمند دو گذرگاه داده هستند، بدین صورت که گذرگاه‌های Cb و Cr ترکیب[۱] شده و به عنوان یک گذرگاه مشترک به اشتراک گذاشته می‌شوند (این کار در سخت‌افزار بسیار رایج است).



شکل (۵-۱۶) : نمایش پیکسل YCbCr 4:2:2 [45]

به هر حال اگر داده رنگی بیشتری حذف شود، مانند آنچه در شکل (۵-۱۷) مشاهده می‌گردد، شما می‌توانید ببینید که هیچ مزایای اضافی از لحاظ گذرگاه داده سخت‌افزار وجود ندارد.

[۱] multiplexed



شکل (۵-۱۷) : نمایش پیکسل YCbCr 4:2:0 [45]

بنابراین، YCbCr 4:2:2 یک قالب پیکسل مشهور مورد استفاده در حوزه دیجیتال است و توسط پورت‌های ویدئو در تمام پردازنده‌های DaVinci شرکت TI پشتیبانی می‌شود.

فصل ششم

پیاده‌سازی سخت‌افزاری

۶-۱) مقدمه

همان طور که در فصل‌های گذشته اشاره شد برای پیاده‌سازی الگوریتم تشخیص اشیاء از برد پردازشگر سیگنال EVMDM6446 استفاده شده است. در این فصل قصد داریم مروری بر چگونگی راه‌اندازی این برد داشته باشیم و همچنین راهکاری برای این که بتوان الگوریتم دلخواه خود را روی این برد پیاده‌سازی کرد، ارائه دهیم.

با توجه به معماری دو هسته‌ای این پردازنده و بنا به دلایلی که قبلاً ذکر شد و همچنین بهره بردن از یک ساختار استاندارد و قابل توسعه از کدک‌انجین به منظور ایجاد یک بستر ارتباطی بین دو هسته و پیاده‌سازی الگوریتم مورد نظر و همچنین از DSP/BIOS به عنوان یک سیستم‌عامل بلادرنگ استفاده کرده‌ایم. کدک‌انجین این امکان را به ما می‌دهد که بتوانیم الگوریتم پردازشی که قرار است روی هسته DSP اجرا شود را به صورت مستقل بنویسیم و سپس با تبدیل آن به یک کدک آن را از طریق هسته ARM فراخوانی کرده و برای پردازش از آن استفاده کنیم. کدک‌انجین از DSPLink به منظور ارتباط بین DSP و ARM استفاده می‌کند و با توجه به اینکه نسخه‌ای که با این برد ارائه شده مخصوص سیستم‌عامل لینوکس[۱] می‌باشد کلیه‌ی الگوریتم‌های مربوطه باید در این محیط نوشته و کامپایل[۲] شوند. از آنجا که نوشتن برنامه‌ها در لینوکس به خاطر نبود یک محیط کدنویسی و اشکال‌زدایی[۳] مناسب بسیار مشکل می‌باشد سعی شده با استفاده از یک سری ترفندها تا آنجا که ممکن است الگوریتم‌ها را در محیط ویندوز[۴] نوشته و سپس آن‌ها را تبدیل به الگوریتمی کنیم که قابلیت اجرا روی لینوکس را دارند.

---

[۱] Linux
[۲] Compile
[۳] Debugging
[۴] Windows

۶-۲) لینوکس

همان طور که در مقدمه اشـاره شد با توجه به یک سری از محدودیت‌هایی که استفاده از این برد برای ما بوجود آورد مجبور شـدیم تا علی‌رغم همه مشکلات و سختی‌های کار با سیستم‌عامل لینوکس از این سـیسـتم‌عامل به عنوان بستر اصلی پیاده‌سازی الگوریتم استفاده کنیم. برخی از این مشکلات عبارت‌اند از:

- ارائه شدن تمام مثال‌ها، کدک‌ها و ابزارهای مورد نیاز همراه برد در محیط لینوکس.
- نصب یک لینوکس روی هسته ARM و نیاز به یک محیط لینوکس دیگر برای ارتباط با آن.
- ارائه شدن DSPLink که برای کار با کدک‌انجین به آن نیاز است فقط در محیط لینوکس.

۶-۲-۱) آماده‌سازی محیط لینوکس

نسـخه لینوکس پیشـنهادی که درون راهنمای برد از آن نام برده شـده نسـخه Red Hat Enterprise Linux v4 (Server Edition) می‌باشـد؛ اما بعد از نصـب یک نسـخه خاص از آن متوجه شدیم که این توزیع لینوکس فقط شـامل یک پنجره ترمینال برای کار با آن می‌باشـد و هیچ محیط گرافیکی دیگری ندارد. از این رو از نسخه CentOS V6.4 که یک توزیع از RedHat می‌باشد، استفاده کردیم. این توزیع علاوه بر سازگار بودن و داشتن تمام خصوصیات RedHat دارای محیط گرافیکی و رابط کاربری بهتری می‌باشد.

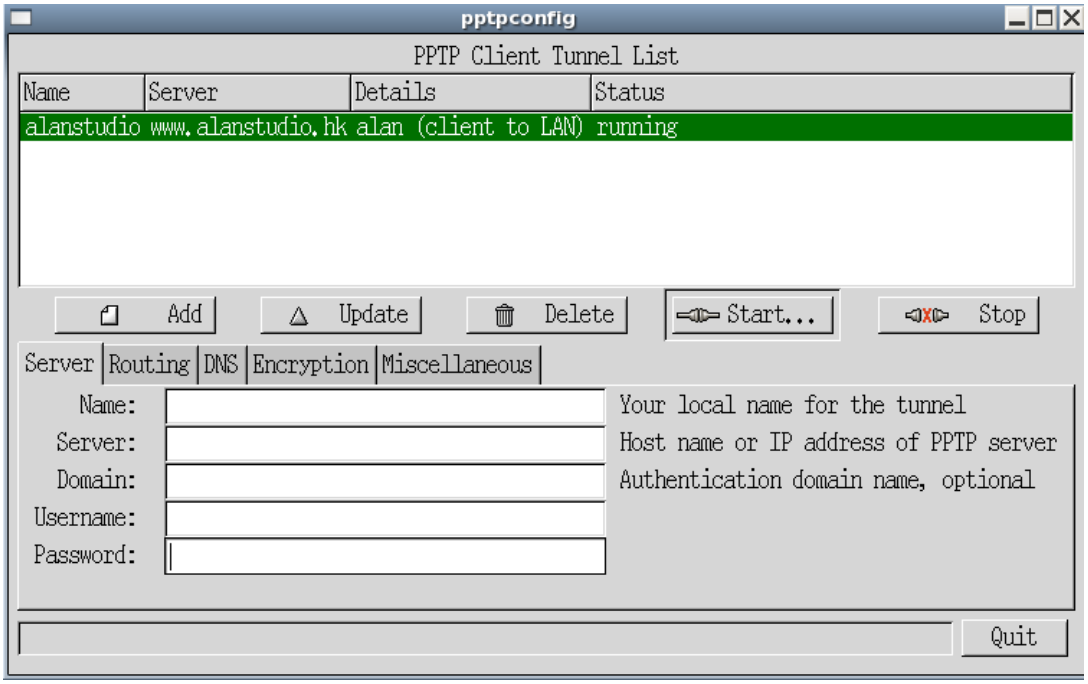
۶-۲-۲) وصل کردن لینوکس به اینترنت

علی‌رغم نصـــب شـــدن اکثر ابزارهـای موردنیاز همراه با لینوکس ولی باز هم تعدادی ابزار که به آن‌ها احتیاج بود (مانند سـرویس DHCP، vpn، minicom و ...) در این نسـخه وجود نداشـت که برای نصب آن‌ها حتماً باید به اینترنت وصـــل شـــد تا خود لینوکس علاوه بر دانلود نرم‌افزار مربوطه، نرم‌افزارهای

وابســته به آن[1] هم دانلود کند. در اینجا یک مشــکل بوجود آمد که برای رفع آن زمان زیادی گذاشــته شــد. مشــکل از این قرار بود که برای وصل شدن به اینترنت در فضای دانشگاه به یک نرم‌افزار vpn نیاز داشــتیم ولی برای نصــب خود این نرم‌افزار به اینترنت احتیاج بود لذا به اصــطلاح فلســفی یک "دور" بوجود می‌آمد که حل آن را مشــکل می‌کرد. بعد از راهنمایی خواســتن از مسئولین سایت دانشگاه و به نتیجه‌ای نرســیدن و مطالعه و امتحان کردن اکثر روش‌های ممکن به این نتیجه رســیدیم که با نصــب یک ســرویس pptp (که نرم‌افزار وابســته‌ی خاصی نداشت و توانستیم آن را توسط یک سیستم وصل به اینترنت گرفته و در این محیط نصــب کنیم) و پیکربندی آن به صــورت دســتی در یک فایل متنی به اینترنت وصــل شویم؛ که این کار به درستی انجام شد و به اینترنت هم ظاهراً وصل شدیم ولی در عمل دســترسی به سایت خاصی نداشتیم که باعث سردرگمی شد. بعد از بررسی‌هایی که انجام گرفت متوجه شدیم که لینوکس به تمام IP[2] های موجود در شبکه اینترنت اجازه وصل شدن را نمی‌دهد که با اعمال تنظیمات خاصــی به آن مبنی بر ایجاد تبادل اطلاعات با همه IP ها این مشکل هم حل شد و توانستیم به اینترنت وصل شده و برنامه‌های لازم مخصوصاً نرم‌افزار pptpconfig که یک نرم‌افزار برای کار با vpn می‌باشد را نصب کنیم (شکل ۶-۱).

[1] Dependency

[2] Internet protocol



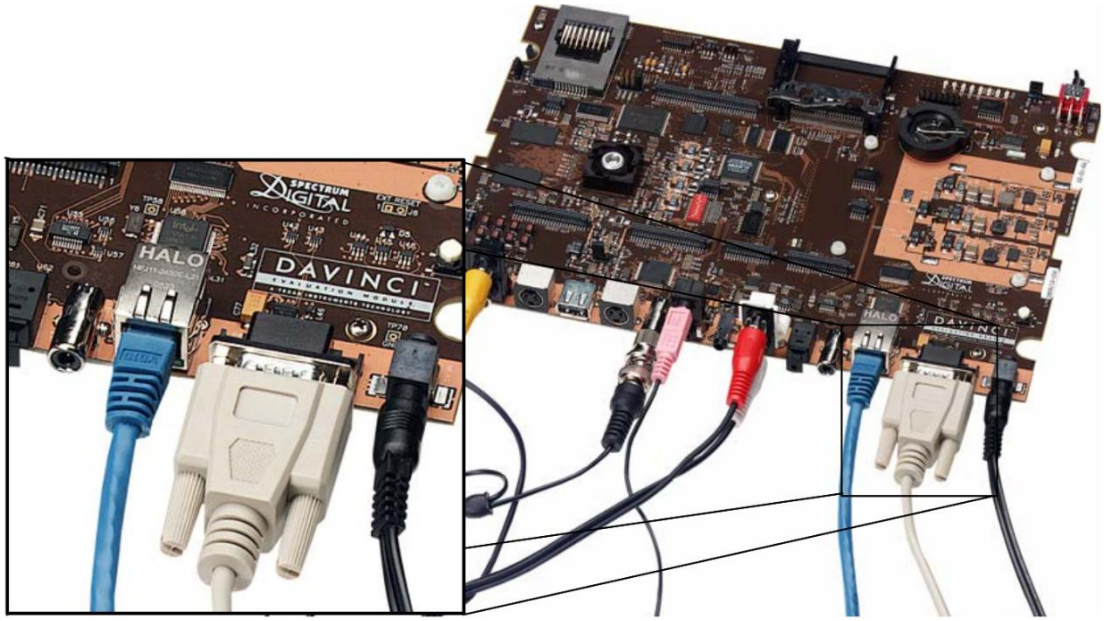
شکل (۶-۱) : نرم‌افزار pptpconfig برای وصل شدن به اینترنت از طریق vpn

۶-۲-۳) نصب نرم‌افزارهای موردنیاز

برای کار با لینوکس و ایجاد یک بستر ارتباطی بین آن و برد پردازشگر به یک سری نرم‌افزار نیاز داریم که به برخی از آن‌ها اشاره می‌کنیم.

۶-۲-۳-۱) نرم‌افزار کار با پورت سریال

کنترل کردن و اجرای دستورهای روی برد به وسیله پورت سریال روی آن انجام می‌شود، بدین منظور بعد از وصل کردن پورت موردنظر به کامپیوتر همان طور که در شکل (۶-۲) نشان داده شده است و نصب نرم‌افزار minicom تنظیمات زیر را روی آن انجام می‌دهیم تا بتوانیم یک ارتباط کنترلی بین برد و لینوکس برقرار کنیم (شکل ۶-۳).



شکل (۶-۲) : وصل نمودن پورت سریال به برد [۴۲]

- Bits per Second: 115200
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

شکل (۶-۳) : تنظیمات پورت سریال نرم‌افزار minicom در محیط لینوکس

۶-۲-۳-۲) سرویس DHCP

برای تبادل اطلاعات بین برد و لینوکس از پورت شبکه استفاده می‌شود. در محیط ویندوز اگر بخواهیم دو کامپیوتر را به هم شـــبکه کنیم و به یکی IP دهیم آنگاه خودش به صـــورت خودکار به آن کامپیوتر

دیگر یک IP اختصـــاص می‌دهد و به آســـانی به هم لینک می‌شـــوند، ولی در لینوکس این قابلیت که DHCP[1] نام دارد حتی نصـب هم نیست. بدین منظور باید ابتدا سرویس DHCP را نصب کرده و سپس در هر بار راه‌اندازی مجدد هم آن را فعال کنیم.

۶-۲-۳-۳) نصب لینوکس مربوط به برد پردازشگر

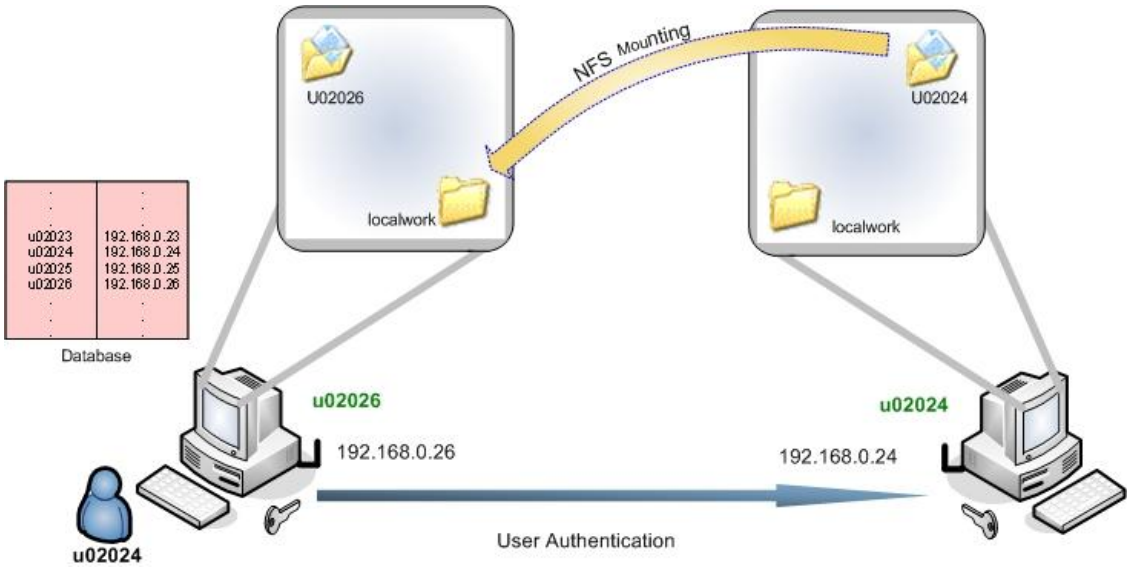
همان طور که اشـــاره شـــد روی هارددیســـک ۴۰ گیگابایتی موجود روی برد پردازشـــگر یک لینوکس نسخه نمایشی MontaVista Linux Pro v5.0 نصب می‌باشد. ارتباط با این سیستم‌عامل فقط از طریق پورت ســـریال بوده و محدود به اجرای دســـتورات برای راه‌اندازی برنامه‌ها می‌شـــود و برای کار ما که احتیاج به تغییرات در فایل‌ها را دارد مناسـب نیسـت. برای حل این مشـکل ما یک نسخه از این توزیع لینوکس را روی لینوکس CentOS نصـب شـده روی کامپیوتر نصـب کرده و سـپس بوسـیله ارتباط با شـبکه و روشـی که در قسـمت بعدی توضـیح داده خواهد شد آن را به برد انتقال داده و از آن استفاده می‌کنیم.

۶-۲-۳-۴) به اشتراک‌گذاری یک سیستم مشترک برای دسترسی برد

برای ایجاد یک ارتباط بین برد و کامپیوتر باید ابتدا در یک مسیر مشخص سیستم‌عامل لینوکس نسخه نمایشـی مخصـوص برد و فایل‌های برنامه را کپی کنیم. سپس بوسیله سرور NFS[2] این مسیر مشخص شـده را به اشـتراک می‌گذاریم. در مرحله بعد باید با اعمال تغییراتی در تنظیمات راه‌اندازی[3] برد، آن را به‌گونه‌ای تنظیم کنیم که به جای اینکه هنگام راه‌اندازی از روی هارددیسـک بارگذاری[4] شود، از طریق وصل شدن به شبکه و آدرس آن مسیر مشخص که به آن داده‌ایم به فایل‌های لینوکس نصب شده روی کامپیوتر دسـترسـی پیدا کند و با آن‌ها پردازشـگر را راه‌اندازی کند. انجام این کارها به ما این امکان را

[1] Dynamic Host Configuration Protocol
[2] Network File System
[3] Boot
[4] Load

می‌دهـد کـه بتوانیم به راحتی فایل‌ها را تغییر داده یا برنامه‌های مورد نظر خود را به آن اضـــافه کنیم. نکته‌ای که باید به آن توجه داشـــت این اســـت که لینوکس در هر بار راه‌اندازی مجدد ســـیســـتم این ســـرویس را غیر فعال می‌کند و باید هنگام روشـــن شـــدن ســـیســـتم خودمان دوباره ســـرویس NFS را راه‌اندازی کنیم. نحوه به اشتراک گذاری این سرویس در شکل (۶-۴) نشان داده شده است.



شکل (۶-۴) : نحوه به اشتراک گذاری فایل‌ها در لینوکس بوسیله NFS

۶-۲-۳-۵) نصب بسته نرم‌افزاری DVSDK

برای کار کردن با مثال‌ها و ســاختن پروژه‌های جدید به یک ســری نرم‌افزار نیاز داریم که اکثر آن‌ها در یک بســـته به نام DVSDK[۱] وجود دارد. این بســـته شـــامل ابزارهای موردنیاز برای کار با کدک‌انجین، DSP/BIOS Link، فـایـل‌های هدر xDAIS و xDM و همچنین یک اختصـــاص دهنده حافظه برای لینوکس (CMEM) می‌باشد.

بعد از نصـب DVSDK دیگر نرم‌افزارهای موردنیاز که درون این بسـته نیسـت را باید خودمان نصب به صورت مجزا نصب کنیم. این نرم‌افزارها شامل XDCTools، سیستم‌عامل DSP/BIOS و همچنین یک

[۱] Digital Video Software Development Kit

کامپایلر که با نام Code Generation Tools شـــناخته می‌شـــود و برای کامپایل کردن برنامه‌ها به آن احتیاج داریم، می‌شـود. (البته توصـیه می‌شـود که از نسـخه جدیدتر این کامپایلر اسـتفاده شـود زیرا نسخه‌ای که همراه با این برد ارائه شده همه ویژگی‌های زبان C++ را پشتیبانی نمی‌کند.)

۶-۳) راه‌اندازی برنامه‌های نمونه

با پشـت سـر گذاشـتن تمام مراحل بالا به این نقطه رسـیدیم که بتوانیم برنامه‌های نمایشی[۱] که همراه برد ارائه شـده‌اند را اجرا کنیم که همان طور که در شـکل (۶-۵) نشـان داده شـده شـامل برنامه‌های encode، decode و encodedecode می‌باشند.

شکل (۶-۵) : راه‌اندازی برنامه‌های نمونه برد داوینچی [42]

این برنامه‌ها فقط برای ضبط و پخش ویدئو کاربرد دارند. بدین صورت که تصویری که از دوربین می‌آید توسـط برنامه encode ذخیره شـده و سـپس توسـط برنامه decode آماده برای پخش روی نمایشـگر می‌گردد.

حال که توانسـتیم برنامه‌های نمایشـی را اجرا کنیم نوبت آن اسـت که بتوانیم کار اصلی یعنی پردازش روی ویدئو را انجام دهیم. متأسفانه به علت نبود یک راهنمای جامع در این مورد، نبود یک برنامه نمونه

[۱] Demo

و پشـتیبانی نکردن شرکت سازنده DSP در مورد سؤالات نرم‌افزاری در این قسمت به مشکلات فراوانی برخورد کردیم کـه ذکر همـه آن‌ها در قالب این پایان‌نامه نمی‌گنجد اما به برخی که اهمیت بیشـــتری دارد اشاره‌ای کرده و در ادامه روش پیاده‌سازی شده را توضیح خواهیم داد.

۶-۴) پیاده‌سازی الگوریتم پردازشی روی DSP

برای اینکه بتوانیم الگوریتم خود را روی این برد پیاده‌ســازی کنیم راه‌های زیادی را آزمایش کردیم که به برخی از آن‌ها اشاره‌ای خواهیم کرد.

- در ابتدا تصـمیم گرفته شد که از همین برنامه نمایشی که روی خود برد وجود دارد و تصویر را گرفتـه و دوبـاره پخش می‌کند، اســـتفاده کنیم. بعد از وقت زیادی که برای تحلیل کد برنامه گذاشـته شـد (چون در لینوکس امکان اشکال‌زدایی[۱] وجود ندارد و نوشتن برنامه در یک فایل متنی صـورت می‌گیرد لذا تحلیل برنامه یک کار بسیار مشکل و زمان‌بر می‌باشد) به این نتیجه رسـیدیم که این برنامه از دو کدک encode و decode درون کدک‌انجین اسـتفاده می‌کند که بـه علت بســـته[۲] بودن آن‌ها امکان تغییر در آن‌ها وجود ندارد و در نتیجه نمی‌توان از آن برای تبدیل به الگوریتم دلخواه استفاده کرد.
- در ســـایـت Spectrum Digital یـک مجموعـه مثال برای کار با برد وجود دارد که مربوط به راه‌اندازی ادوات جانبی پردازنده مانند کار با حافظه جانبی eeprom، لامپ‌های led، پورت‌های usb و uart و ... می‌بـاشـــد. از میـان این مثال‌ها تنها دو مورد به نام‌های video_colorbars و video_loopback مربوط به ویدئو می‌باشـند. برنامه اول که فقط یک سـری سـتون‌های رنگی در تصـــویر ایجـاد می‌کرد و بدرد کار ما نمی‌خورد ولی برنامه دوم با تغییراتی که در آن ایجاد کردیم می‌توانســـت تصـــویر را گرفته و پخش کند و به نظر می‌رســـید که می‌توان از آن برای

---

[۱] Debugging

[۲] Package

رسـیدن به هدفمان که پردازش تصویر بین این دو مرحله می‌باشد، بهره برد. با بررسی‌های که انجام گرفت به این نتیجه رسیدیم که این برنامه فقط در هسته ARM اجرا می‌شود و با اعمال یک الگوریتم پردازشـی با محاسبات کم به آن به این نتیجه رسیدیم که سرعت پردازش بسیار ضـعیف اسـت به طوری که مثلاً برای منفی[۱] کردن تصـویر یک کار بسیار ساده می‌باشد (فقط کافی اسـت که تک‌تک پیکسـل‌های تصویر از حداکثر مقدار آن‌ها برای مثال از ۲۵۵ کم شود) زمانی حدود چند ثانیه برای یک فریم تصـویر صـرف می‌شـد. همچنین هنگام اسـتفاده از کتابخانه‌های پرکاربرد و مفید ارائه شده توسط شرکت TI برای پردازش تصویر و ویدئو (vlib و imglib) متوجه شـدیم که این کتابخانه‌ها مخصوص هسته DSP بوده و این امکان وجود ندارد کـه بخواهیم از آن‌هـا در این برنـامـه اسـتفـاده کنیم. با توجه به این موارد از این برنامه هم نتوانستیم استفاده کنیم.

- با مطالعه راهنماها، پایگاه‌ها[۲] و تالار گفتگوهای[۳] TI به این نتیجه رسـیدیم که تنها راه برای کار با این نوع معماری و خواسـته‌ای که ما از آن داریم اسـتفاده از کدک‌انجین می‌باشـد. در مثال‌های ارائه شـده همراه کدک‌انجین مثالی برای خواندن تصویر از دوربین و پخش آن وجود ندارد لذا تصـمیم گرفتیم که از همان برنامه video_loopback که در محیط ویندوز و نرم‌افزار CCS آن را امتحان کرده بودیم اسـتفاده کنیم (مشکل اصلی برنامه video_loopback سرعت اجرای پایین آن بود که اگر می‌توانسـتیم کدک‌انجین را به آن اضـافه کنیم، با توجه به توانایی اجرا شـدن کدک‌انجین در هسـته DSP این مشـکل حل می‌شد). پس لازم بود کدک‌انجین را که نسـخه مربوط به لینوکس آن را داشـتیم به نحوی در محیط ویندوز راه‌اندازی کنیم. برای این منظور نسخه سازگار با محیط ویندوز و اکثر برنامه‌های وابسته به آن را از سایت TI گرفته و نصـب کردیم. بعد از وقت زیادی که گذاشـته شد تا بتوانیم برنامه را به نحوی با کدک‌انجین

---

[۱] Negative

[۲] Sites

[۳] Forums

ادغام کنیم متوجه شدیم که DSPLink که کدک‌انجین از آن برای ایجاد ارتباط بین دو هسته استفاده می‌کند فقط در محیط لینوکس ارائه شده است و علی‌رغم تلاش‌های زیادی که برای آوردن آن به محیط ویندوز انجام شد اما نتیجه‌ای نداشت و این روش هم جواب نداد.

بعد از امتحان کردن اکثر روش‌های ممکن به این نتیجه رسیدیم که تنها راه، استفاده از کدک‌انجین در محیط لینوکس می‌باشد. برای این منظور خود کدک‌انجین را که در این محیط امتحان کرده بودیم و جواب هم داده بود و تنها به یک برنامه نیاز داشتیم که کار ضبط و پخش تصویر را انجام دهد. بعد از بررسی‌های انجام گرفته روی اکثر مثال‌ها در برنامه‌های نصب شده، یک مثال در برنامه DMAI[1] به اسم video_loopback_resize پیدا کردیم که این برنامه تصویر ورودی را می‌گرفت و بعد از تغییر اندازه آن دوباره آن را پخش می‌کرد. تنها مشکل این برنامه این بود که روی هسته ARM اجرا می‌شد و همان مشکلات ذکر شده در برنامه‌های قبل برای آن صدق می‌کرد، اما با توجه به اینکه در لینوکس کامپایل شده بود این امکان وجود داشت که بتوان کدک‌انجین را در آن به کار برد. با تلاش‌هایی که در این زمینه صورت گرفت توانستیم یک کدک همه‌منظوره به نام UNIVERSAL را در برنامه ذکر شده پیاده‌سازی کنیم. کاری که این کدک انجام می‌داد کپی کردن بافر[2] ورودی در خروجی بود لذا خروجی برنامه همانند حالت اولیه بود که تصویر را از دوربین می‌خواند و در نمایشگر نشان می‌داد ولی با این کار این امکان بوجود آمد که اگر بتوان یک کدک درست کرد که الگوریتم پردازشی خودمان را درون آن پیاده‌سازی کرد و به جای کدک ذکر شده گذاشت آنگاه به نتیجه دلخواهمان برسیم؛ که همین طور هم شد و با روشی ترکیبی از محیط ویندوز و لینوکس که در ادامه توضیح داده خواهد شد

[1] Davinci Multimedia Application Interface

[2] Buffer

توانستیم یک کدک برای کدک‌انجین طراحی کنیم و الگوریتم پردازشی خود را درون آن پیاده‌سازی کرده و سپس با تبدیل آن به کدک سرور[1] و فراخوانی آن از طریق برنامه ARM به هدفمان برسیم.

۶-۴-۱) ساخت یک کدک قابل‌استفاده در کدک‌انجین

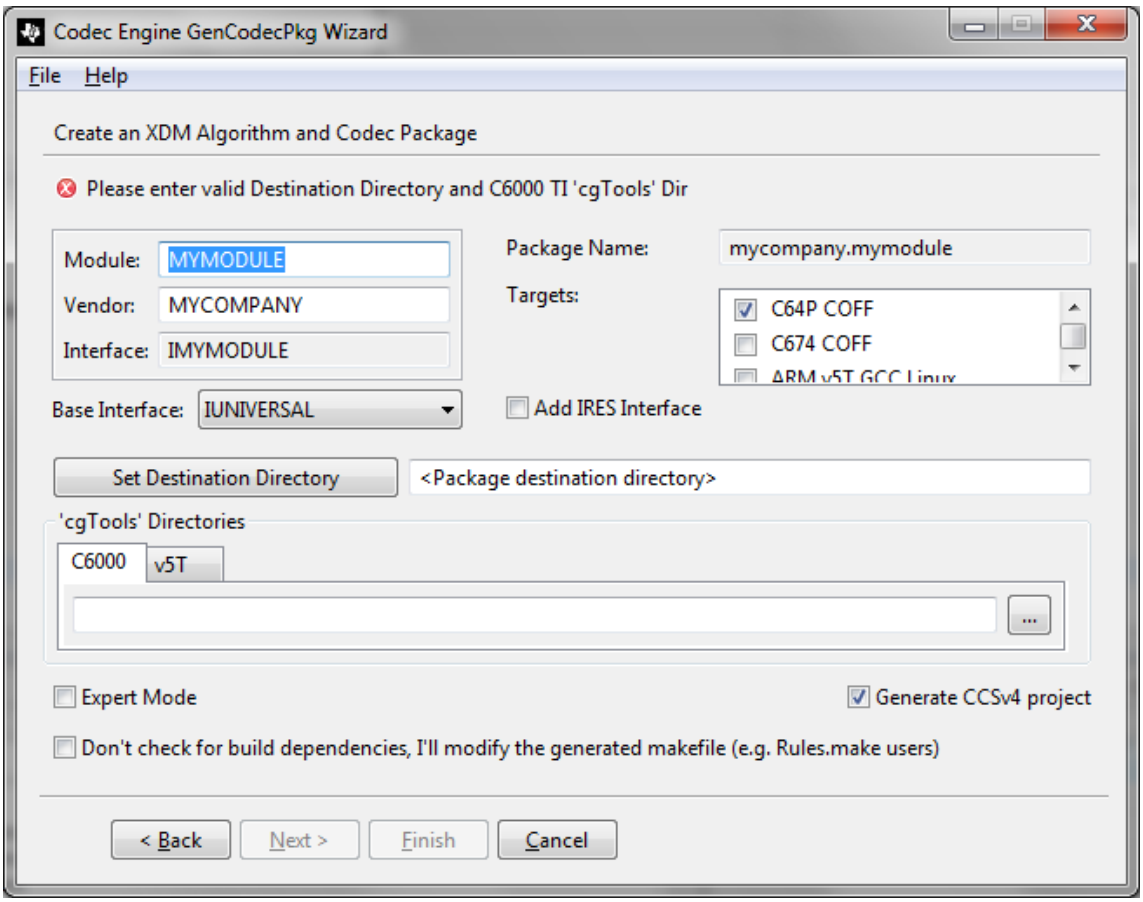
همان طور که اشاره شد برنامه‌ای که بتواند با هر دو هسته ARM و DSP ارتباط برقرار کند در مراحل قبل نوشته شد. روال کلی این برنامه بدین صورت است که کد اصلی روی هسته ARM و در محیط لینوکس اجرا می‌شود و این هسته تصویر ورودی را از دوربین گرفته سپس بوسیله کدک‌انجین هسته DSP را روشن کرده تصویر را به آن ارسال می‌کند و بعد پردازش کردن آن دوباره آن را می‌گیرد و روی نمایشگر نشان می‌دهد. تنها نقص این برنامه این بود که از کدک‌های آماده کدک‌انجین استفاده می‌کرد و قادر نبود پردازش خاصی روی تصویر انجام دهد لذا در این قسمت قصد داریم روشی ارائه کنیم که بتوان به سادگی یک کدک ساخت و از آن استفاده کرد.

اگر بخواهیم کدک را در محیط لینوکس بسازیم یک کار بسیار زمان‌بر است و احتمال خطا در آن بسیار زیاد است (چون روال کار بدین صورت است که باید یک سری فایل متنی درست کرد و طبق قاعده‌های خاص و بسیار زیادی آن‌ها را مقداردهی کرد).

یک روش دیگر این است که کدک‌انجین مخصوص ویندوز را در محیط CCS نصب کرده و از یک نرم‌افزار جانبی که با آن نصب می‌شود و در شکل (۶-۶) نشان داده شده برای ساخت کدک استفاده کرد و سپس آن را به محیط لینوکس انتقال داد. یک مشکل کوچک در اینجا بوجود می‌آید و از این قرار است که مسیرهای درون فایل Makefile کدک همگی برای نرم‌افزارهای ویندوزی تنظیم می‌شوند که می‌توان به صورت دستی بعد از منتقل کردن به لینوکس و متناسب با مسیر نرم‌افزارها آن‌ها را تنظیم کرد. علاوه بر مزیت گفته شده یک خوبی دیگر که این روش دارد این است که می‌توان از

[1] Codec Server

قابلیت اشـکال‌زدایی CCS استفاده کرد و بعد از جواب گرفتن و مطمئن شدن از الگوریتم مورد نظر آن را به محیط لینوکس انتقال داد.

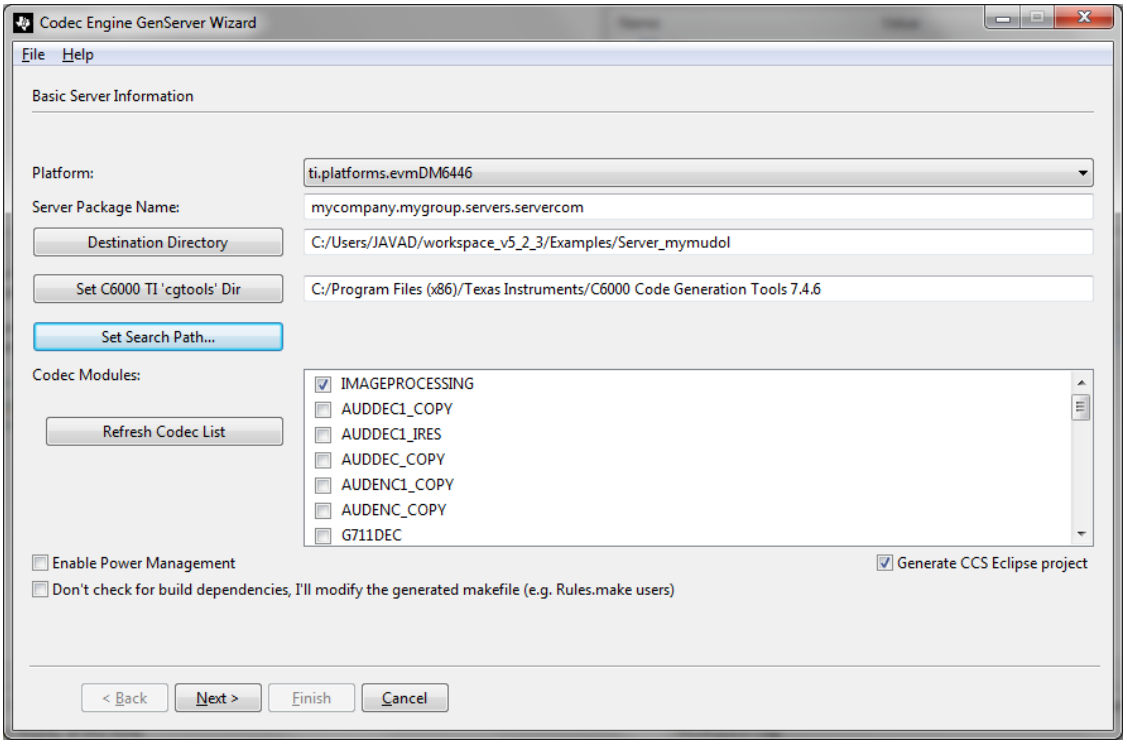


شکل (۶-۶) : نرم‌افزار مربوط به ساختن کدک در CCS

۶-۴-۲) ساخت یک کدک سرور

حال که توانستیم یک کدک با الگوریتم خودمان بسازیم باید آن را به یک بسته تبدیل کنیم و آن را کنار برنامه ARM قرار دهیم تا بتواند آن را خوانده و از آن به روشی که توضیح داده شد استفاده کند (این قسمت شبیه به فایل‌های DLL در برنامه‌نویسی می‌باشد که برنامه باید هنگام اجرا به این فایل‌ها دسترسی داشته باشد تا بتواند از توابع آن‌ها استفاده کند). در اینجا به این بسته "کدک سرور" گفته می‌شود. برای ساخت این فایل هم همانند روشی که برای ساخت کدک به کار بردیم از یک نرم‌افزار

کمکی دیگر همراه با کدک‌انجین نشـان داده شـده در شـکل (۶-۷) مخصـوص محیط ویندوز استفاده کرده و سپس تغییرات لازم را به آن اعمال می‌کنیم و در محیط لینوکس از آن بهره می‌بریم.



شکل (۶-۷) : نرم‌افزار مربوط به ساختن کدک سرور در CCS

۶-۴-۳) تخصیص حافظه

هنگام آزمایش الگوریتم‌های متفاوت به این مشـــکل برخوردیم که در بعضـــی از آن‌ها که به حافظه‌ی بیشـتری برای اجرا نیاز داشـتند، برنامه اجرا نمی‌شد. بعد از بررسی کردن موضوع معلوم شد که هنگام سـاخت برنامه باید میزان تخصـیص حافظه برای متغیرهای پویا[۱] که به اندازه Heap معروف اسـت را تعیین کرد. این کار در محیط CCS به دلیل محیط گرافیکی که دارد بسـیار سـاده می‌باشد ولی هنگام کـار با کدک‌انجین در محیط لینوکس و همچنین به خاطر اسـتفاده کدک‌انجین از سـیستم‌عامل DSP/BIOS که تنظیمات آن کلاً فایل متنی می‌باشد کمی مشکل است ولی خوشبختانه با روش سعی

[۱] Dynamic memory allocation

و خطا و تغییر در بعضی فایل‌ها این مشکل هم برطرف شد و توانستیم میزان تخصیص حافظه لازم را تعیین کنیم

۶-۵) نتایج پیاده سازی

به منظور پیاده سازی الگوریتم مورد نظر ابتدا آن را به یک کد C++ که هیچ وابستگی به کتابخانه خاصی نداشت تبدیل کردیم و سپس از روی آن یک کدک ساخته و با تبدیل آن به کدک سرور آن را روی پردازنده اجرا نمودیم. بعد از اجرای الگوریتم متوجه شدیم به خاطر اینکه بعضی از محاسبات ما به صورت اعشاری بود و این پردازنده به علت ماهیت ممیز ثابتی بودن (و با توجه به اینکه هر محاسبه اعشاری را در چند سیکل کلاک انجام می‌داد) نسبت به حالت شبیه‌سازی نرم‌افزاری سرعت مناسبی نداشت. برای حل این مشکل از کتابخانه پردازش تصویر ارائه شده توسط شرکت TI به نام VLIB استفاده کردیم. این کتابخانه به علت اینکه از لحاظ اسمبلی بهینه شده است بسیار سرعت مناسبی دارد اما یک مشکل عمده آن محدود بودن تابع‌های آن می‌باشد به طوری که به طور مثال هیچ تابعی برای کار با توصیفگر SURF درون آن وجود ندارد. ولی یک سری تابع برای شناسایی اشیاء درون آن وجود داشت که ما برای پیاده سازی از آن‌ها استفاده کردیم و در بهبود سرعت بسیار مؤثر بودند.

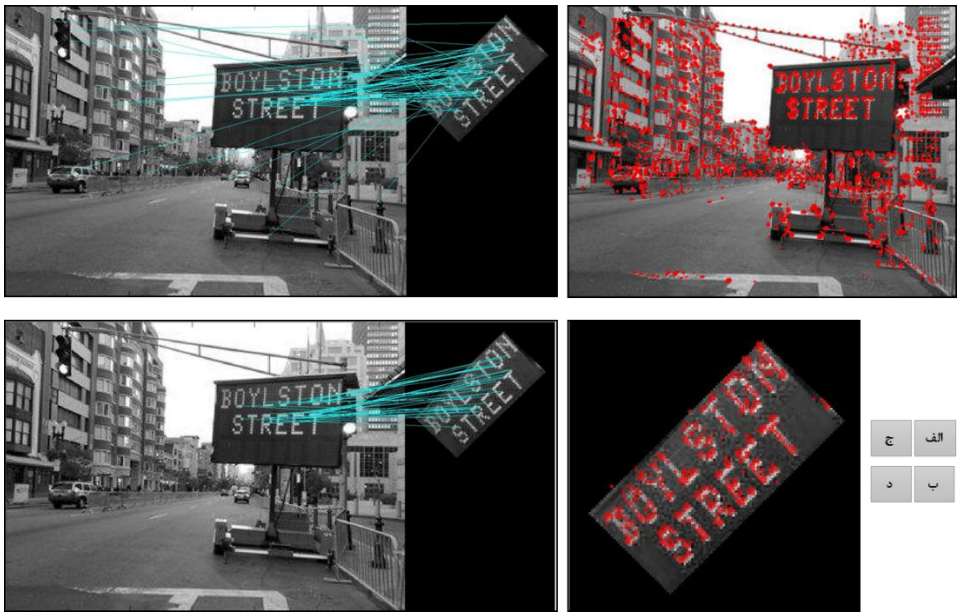
روند الگوریتم پیاده سازی شده بدین صورت است که ابتدا توسط آشکار ساز گوشه FAST[1] نقاط کلیدی تصویر را بدست می‌آوریم و سپس آن‌ها را توسط توصیفگر BRIEF توصیف می‌کنیم. نام دیگر این الگوریتم ORB یا Oriented FAST and Rotated BRIEF می‌باشد. این روش نسبت به چرخش مقاوم بوده ولی نسبت به مقیاس مقاوم نیست. برای اینکه بتوانیم سرعت اجرای برنامه را محاسبه کنیم از محیط شبیه‌سازی CCS استفاده کردیم. تعداد کلاک‌های لازم برای اجرای قسمت‌های مختلف برنامه برای دو تصویر شیء مورد نظر و تصویر صحنه را می‌توانید در جدول (۶-۱) ببینید.

[1] FAST Corner detection

جدول (۶-۱) : تعداد کلاک‌های لازم برای اجرای توابع مورد استفاده در الگوریتم پیاده سازی شده

نام تابع مورد نظر	تعداد کلاک برای اجرای تابع مورد نظر روی تصویر صحنه با اندازه ۳۲۰×۲۴۰	تعداد کلاک برای اجرای تابع مورد نظر روی تصویر شیء با اندازه ۱۲۰×۱۲۰
VLIB_afast9_detectCorners	۱،۶۷۹،۸۸۴	۴۵۰،۱۴۲
VLIB_ORB_computeOrientation	۴۸،۴۷۹،۰۲۵	۲۴،۴۰۰،۷۱۳
VLIB_integralImage8	۳۶۰،۲۸۴	۲۹۲،۱۳۴
VLIB_ORB_computeRBrief	۶۶،۳۷۵،۱۸۶	۲۷،۰۳۲،۵۳۲
جمع کل	۱۱۶,۸۹۴,۳۷۹	۵۲,۱۷۵,۵۲۱

همان طور که مشاهده می‌شود تعداد کل کلاک برای توصیف تصویر صحنه حدود ۱۱۶ میلیون می‌باشد و با توجه به اینکه پردازنده قادر به اجرای ۶۴۸۰ میلیون دستورالعمل در ثانیه در یک نرخ کلاک ۸۱۰ مگاهرتزی می‌باشد [۴۱]، سرعت اجرا برای توصیف تصویر حدود ۵۵ فریم بر ثانیه است که نسبتاً مناسب می‌باشد. در شکل (۶-۸) می‌توانید نقاط کلیدی پیدا شده توسط الگوریتم و همچنین انطباق آن‌ها را ببینید.



شکل (۶-۸) : نتایج حاصل از پیاده سازی سخت‌افزاری. الف) نقاط کلیدی پیدا شده در صحنه (گوشه‌ها)، ب) نقاط کلیدی پیدا شده در شیء مورد نظر، ج) انطباق ویژگی در دو تصویر، د) حذف داده‌های پرت با استفاده از RANSAC

فصل هفتم

نتیجه‌گیری و پیشنهاد برای کارهای آینده

۷-۱) نتیجه گیری

در این پایان‌نامه روشی برای شناسایی اشیاء ارائه شد که دارای دقت مناسب و سرعت اجرای خوبی باشد. برای این منظور روش‌های مرسوم استخراج ویژگی که اولین گام در شناسایی اشیاء می‌باشد بررسی شدند و از بین آن‌ها توصیفگر SURF که هم سرعت اجرای مناسب و هم کارایی خوبی داشت انتخاب شد. در مرحله بعد، از تصویر مدل شیء مورد نظر و تصویر صحنه‌ای که می‌خواهیم شیء را در آن پیدا کنیم بوسیله توصیفگر انتخابی ویژگی استخراج کرده و بوسیله فاصله اقلیدسی ویژگی‌های متناظر با هم را مشخص کردیم. در این مرحله که تطبیق ویژگی‌ها نام دارد یک سری داده‌های پرت بوجود آمد که از عملگر RANSAC برای حذف آن‌ها استفاده کردیم. برای پیاده‌سازی اولیه الگوریتم از نرم‌افزار متلب استفاده کردیم و پس از جواب گرفتن از برنامه به خاطر اینکه قصد پیاده‌سازی سخت‌افزاری آن را داشتیم الگوریتم را به C++ تبدیل کردیم؛ در این مرحله چون کمی از برنامه به کتابخانه OpenCV وابسته بود و امکان پیاده‌سازی این کتابخانه به علت محدودیت‌های حافظه در سخت‌افزار مورد نظر وجود نداشت دوباره کد را بازنویسی کردیم به طوری که کاملاً به صورت مستقل اجرا شود.

برای پیاده‌سازی سخت‌افزاری از برد پردازشگر EVMDM6446 استفاده کردیم. برای آشنایی با این سخت‌افزار ابتدا قسمت‌های مختلف آن را توضیح دادیم و سپس روش راه‌اندازی و پیکربندی اولیه را تشریح کردیم به طوری که بتوان برنامه‌های نمایشی همراه با برد را اجرا نمود و همچنین بتوان در آن‌ها تغییراتی اعمال کرد. سپس برای اینکه بتوانیم الگوریتم پردازشی خودمان را روی آن پیاده کنیم روش‌های متعددی را مورد بررسی قرار دادیم که هر کدام به علتی مناسب خواسته ما نبود. سرانجام به این نتیجه رسیدیم که باید از کدک‌انجین در محیط لینوکس استفاده کنیم. ساخت یک کدک دلخواه که الگوریتم مورد نظر درون آن پیاده شده باشد در محیط لینوکس کار نسبتاً پیچیده‌ای است به همین منظور با ارائه ترفندی کدک و همچنین کدک سرور مربوط به آن را در محیط ویندوز ساختیم

و بعد اعمال تنظیمات خاصی آن را به لینوکس انتقال داده و آن را در برنامه مورد نظر استفاده کردیم. کدک‌انجین این امکان را برای ما فراهم آورد که بتوانیم از هر دو هسته پردازنده به نحو احسن استفاده کنیم بدین صورت که ابتدا بوسیله هسته ARM تصویر ورودی را از دوربین خوانده سپس پردازنده DSP را روشن کرده و تصویر را به آن می‌فرستیم و بعد از اعمال پردازش مورد نظر روی آن دوباره آن‌را گرفته و روی نمایشگر نشان می‌دهیم.

۷-۲) پیشنهادها برای کارهای آینده

- از آنجا که هدف از شناسایی اشیاء در این پایان‌نامه استفاده در حوزه‌های رباتیک بوده پیشنهاد می‌شود که سخت افزار تشریح شده بر روی یک ربات پیاده‌سازی شود تا بتوان در محیط واقعی برای جابجایی اشیاء مورد نظر به کار رود.
- استفاده از برد پیشرفته‌تر و با سرعت پردازشی بالاتر DM816x/AM389x Evaluation Module که قابلیت پشتیبانی از کتابخانه OpenCV هم دارد برای پردازش تصویر بسیار مناسب‌تر است. پیشنهاد بعدی راه اندازی این برد جهت الگوریتم شناسایی اشیاء می‌باشد.

منابع

[1] A. Andreopoulos and J. K. Tsotsos, "50 Years of Object Recognition: Directions Forward," *Computer Vision and Image Understanding,* 2013.

[2] A. Opelt, A. Pinz and A. Zisserman, "Learning an Alphabet of Shape and Appearance for Multi-Class Object Detection," *International Journal of Computer Vision,* vol. 80, pp. 16-44, 2008.

[3] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[4] H. Bay, T. Tuytelaars and L. V. Gool, "SURF: Speeded Up Robust Features," in *Computer Vision – ECCV 2006,* Austria, 2006, pp. 404-417.

[5] A. Ahmadyfard and J. Kittler, "Using relaxation technique for region-based object recognition," *Image and Vision Computing,* vol. 20, pp. 769-781, 2002.

[6] S. K. Naik and C. A. Murthy, "Distinct Multi-Colored Region Descriptors for Object Recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 29, no. 99, pp. 1291-1296, 2007.

[7] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision* , vol. 7, no. 1, pp. 11-32, 1991.

[8] M. A. Stricker, "Color and Geometry as Cues for Indexing," Chicago, 1992.

[9] M. Turk and P. A.P, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, Maui, 1991.

[10] P. Tu, R. Hartley and T. Saxena, "Recognizing Objects Using Color-Annotated Adjacency Graphs," in *Shape, Contour and Grouping in Computer Vision*, 1999, pp. 246-263.

[11] A. Kostin, J. Kittler and W. Christmas, "Object recognition by symmetrised graph matching using relaxation labelling with an inhibitory mechanism," *Pattern Recognition Letters,* vol. 26, no. 3, pp. 381-393, 2005.

[12] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *International Journal of Computer Vision,* vol. 14, no. 1, pp. 5-24, 1995.

[13] B. McFee, C. Galleguillos and G. Lankriet, "Contextual Object Localization With Multiple Kernel Nearest Neighbor," *Image Processing, IEEE Transactions on,* vol. 20, no. 2, pp. 570-585, 2011.

[14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 27, no. 10, pp. 1615-1630, 2005.

[15] M. Pietikäinen, T. Ojala and Z. Xu, "Rotation-invariant texture classification using feature distributions," *Pattern Recognition,* vol. 33, no. 1, pp. 43-52, 2000.

[16] Y. Ke and S. R, "PCA-SIFT: a more distinctive representation for local image descriptors," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, 2004.

[17] S. Belongie, J. Malik and J. Pazicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 24, no. 4, pp. 509-522, 2002.

[18] J.-M. Morel and G. Yu, "ASIFT: A New Framework for Fully Affine Invariant Image Comparison," *SIAM Journal on Imaging Sciences,* vol. 2, no. 2, pp. 438-469, 2009.

[19] C. Ancuti and P. Bekaert, "SIFT-CCH: Increasing the SIFT distinctness by Color Co-occurrence Histograms," *Image and Signal Processing and Analysis, 2007. ISPA 2007. 5th International Symposium on,* pp. 130-135, 2007.

[20] S.-O. Shim and T.-S. Choi, "Image indexing by modified color co-occurrence matrix," in *Acoustics, Speech, and Signal Processing, 2003 IEEE International Conference on*, 2003.

[21] P. Moreno, A. Bernardino and J. Santos-victor, "Improving the SIFT descriptor with smooth derivative filters," *Pattern Recognition Letters,* vol. 30, no. 1, pp. 18-26, 2009.

[22] H. Li, Y. Wei and Y. Yuan, "Similarity learning for object recognition based on derived kernel," *Neurocomputing,* vol. 83, pp. 110-120, 2012.

[23] X. Tan, S. Chen, Z.-H. Zhou and J. Liu, "Face Recognition Under Occlusions and Variant Expressions With Partial Similarity," *Information Forensics and Security, IEEE Transactions on,* vol. 4, no. 2, pp. 217-230, 2009.

[24] C.-Y. Yen and K. J. Cios, "Image recognition system based on novel measures of image similarity and cluster validity," *Neurocomputing,* vol. 72, no. 1-3, pp. 401-412, 2008.

[25] D. P. Httenlocher, G. A. Klanderman and W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 15, no. 8, pp. 850-863, 1993.

[26] P. Simard, Y. LeCun and J. S. Denker, "Efficient Pattern Recognition Using a New Transformation Distance," *Advances in Neural Information Processing Systems,* pp. 50-58, 1992.

[27] G. Sanromàa, R. Alquézarb and F. Serratosaa, "A new graph matching method for point-set correspondence using the EM algorithm and Softassign," *Computer Vision and Image Understanding,* vol. 116, no. 2, pp. 292-304, 2012.

[28] C. Harris and M. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference,* 1988.

[29] B. Luo and E. Hancock, "A unified framework for alignment and correspondence," *Computer Vision and Image Understanding*, vol. 92, pp. 26-55, 2003.

[30] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *nternational Journal of Computer Vision*, vol. 19, no. 1, pp. 57-91, 1996.

[31] S. J. Russell and P. Norvig, Artificial intelligence a modern approach, Prentice Hall Series in Artificial Intelligence, 1995, p. 74.

[32] J. J. DiCarlo, D. Zoccolan and N. C. Rust, "How Does the Brain Solve Visual Object Recognition?," *Neuron*, vol. 73, no. 3, pp. 415-434, 2012.

[33] S. Jeong and M. Lee, "Adaptive object recognition model using incremental feature representation and hierarchical classification," *Neural Networks*, vol. 25, pp. 130-140, 2012.

[34] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, pp. 1019 - 1025, 1999.

[35] C. Siagian and L. Itti, "Rapid Biologically-Inspired Scene Classification Using Features Shared with Visual Attention," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 2, pp. 300-312, 2007.

[36] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.

[37] C. Evans, "Notes on the OpenSURF Library," 2009.

[38] P. Beaidet, "Rotationally invariant image operators," in *International Conference on Pattern Recognition*, 1978.

[39] A. P. Witkin, "Scale-space filtering: A new approach to multi-scale description," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84*, 1984.

[۴۰] ک. شـفاعی، مرجع کامل پردازنده‌های DSP سـری‌های 2000، 5000 و 6000، مؤسسه چاپ و انتشارات آستان قدس، تهران، ۱۳۸۹.

[41] "TMS320DM6446 Datasheet," Texas Instruments, 2005.

[42] "TMS320DM6446 DVEVM V2.0 Getting Started Guide," Texas Instruments, 2008.

[43] "TMS320 DSP/BIOS V5.42 User's Guide," Texas Instruments, 2012.

[44] "Codec Engine Application Developer User's Guide," Texas Instruments, 2007.

[45] "Digital Video Using DaVinci SoC," Texas Instruments, 2007.

Abstract

Main objective of this thesis is to recognize predefined objects in an image of scene, especially for robotics applications. As working with robots, it is important to have a real-time algorithm, so the proposed method must have high performance in both aspects: speed and accuracy. Therefore, common methods of feature extraction (the first step in object recognition) were studied, and the SURF descriptor, which has high accuracy and speed, has been selected. Since the extracted features are good discriminators, we used a simple method with low computational complexity to implement "matching" stage. For matching between descriptors of scene image and object's model we use Euclidean Distance. In this stage some outlier data appear, that were removed using RANSAC method.

To implement mentioned algorithm on the robot, we must use a portable hardware; so we choose the DM6446 model in DaVinci series of TI DSP's. In this thesis, we focus mainly on setting up this Evolution Module, and also on providing a general comprehensible approach to implement arbitrary algorithms on this board.

Keywords: object recognition, SURF descriptor, RANSAC, digital signal processors, DM6446

Shahrood University

Faculty of Electrical and Robotics Engineering

Object Recognition Using Local Features for Robot Perception of Environment and Hardware Implementation on TMS320DM6446 digital processor

Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.) in Electronic Engineering

By:

Javad Jowkar

Supervisors:

Dr. Alireza Ahmadyfard

Dr. Hossein Marvi

February 2015