
بسم الله الرحمن الرحيم

دانشکده مهندسی برق و رباتیک

پایان نامه کارشناسی ارشد مهندسی سیستم های الکترونیک دیجیتال

ردیابی بلادرنگ اشیاء با استفاده از پردازش موازی

نگارنده: مهدی مقیمی

استاد راهنما

دکتر حسین خسروی

بهمن ۱۳۹۵

شماره: ۱۴۵۶/آ.ت.ب
تاریخ: ۱۳۹۵/۱۱/۱۲
ویرایش: - - - - -

بسمه تعالی

مدیریت تحصیلات تکمیلی

فرم شماره ۷: صورتجلسه دفاع از پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد خانم / آقای **مهدی مقیمی** به شماره دانشجویی **۹۳۱۶۴۱۴** رشته **مهندسی برق** گرایش **الکترونیک** که در تاریخ **۹۵/۱۱/۱۲** تحت عنوان :

ردیابی بلادرنگ اشیاء با استفاده از پردازش موازی

با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح ذیل اعلام می‌گردد:

قبول (با درجه بسیار خوب) امتیاز ۱۸/۵ ☒ دفاع مجدد ☐ مردود ☐

نوع تحقیق: نظری ☒ عملی ☐

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	امضاء
۱ـ استادراهنمای اول	حسین صدری	استادیار	
۲- استادراهنمای دوم	ـــــ	ـــ	ـــ
۳- استاد مشاور	ـــــ	ـــ	ـــ
۴- نماینده شورای تحصیلات تکمیلی	محمدرضا الهی	استادیار	
۵- استاد ممتحن اول	عادل دادخواه	دانشیار	
۶ـ استاد ممتحن دوم	هادی گرایلو	استادیار	

نام و نام خانوادگی رئیس دانشکده:

تاریخ و امضاء و مهر دانشکده:

تقدیم به

پدرم

استوارترین تکیه گاهم

مادرم

غمخوار تمام زندگی‌ام،

باشد که حاصل تلاشم
نسیم گونه غبار خستگی‌شان را بزداید.

تشکر و قدردانی

ستایش مخصوص خداست چرا که بهره ای از معرفت خودرا به ما ارزانی فرمود و از نعمت شکر خویش به ما الهام فرمود و برخی از درهای ناتناهی علم به ربوبیتش را به سوی ما گشود و ما را به مرتبه ی اخلاص در توحید خویش راهنمایی نمود و از شائبه ی الحاد و شک در امرش دور کرد.

خرد هر کجا گنجی آرد پدید

ز نام خدا سازد آن را کلید

نخست، حمد و سپاس خداوند رحمان و رحیم را که تمامی موفقیت های این پژوهش در سایه ی لطف او رقم خورد.

سپس، از استاد گرامی ام **جناب آقای دکتر حسین خسروی** که همواره با متانت و خوش رویی هدایت این پایان نامه را بر عهده داشتند و در جهت گیری پایان نامه نقش مؤثری ایفا کردند و

همواره مشوق بنده بودند تشکر می‌نمایم. همچنین بر خود واجب می‌دانم **از جناب آقای دکتر حسین مروی** که از محضر ایشان نیز بهره‌ها بردم، تشکر نمایم.

و در پایان، سپاس و قدر دانی ویژه، از **پدر و مادر مهربانم**، که در تمامی مراحل زندگی یار و یاور من بوده‌اند.

تعهد نامه

اینجانب **مهدی مقیمی** دانشجوی دوره کارشناسی ارشد رشته **مهندسی برق- الکترونیک** دانشکده **مهندسی برق و رباتیک** دانشگاه صنعتی شاهرود نویسنده پایان نامه **ردیابی بلادرنگ اشیاء با استفاده از پردازش موازی** تحت راهنمائی جناب آقای **دکتر حسین خسروی** متعهد می شوم:

- تحقیقات در این پایان نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهشهای محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است.
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می باشد و مقالات مستخرج با نام « دانشگاه صنعتی شاهرود » و یا «**Shahrood University of Technology**» به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه ، در مواردی که از موجود زنده (یا بافتهای آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل راز داری ، ضوابط و اصول اخلاق انسانی رعایت شده است .

تاریخ

امضای دانشجو

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه های رایانه ای، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه صنعتی شاهرود می باشد. این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در پایان نامه بدون ذکر مرجع مجاز نمی باشد.

چکیده

ردیابی اشیا را می‌توان، نمایش تغییرات موقعیت یک شی و دنبال کردن آن در یک دنباله از تصاویر ویدئویی تعریف کرد. باید توجه داشت در اکثر کاربردهای عملی مرتبط با موضوع ردیابی، بلادرنگ بودن روند پردازش دارای اهمیت ویژه‌ای است. همچنین قابل توجه است که اهمیتِ مسئله بلادرنگ بودنِ فرآیند ردیابی هدف در مسائل امنیتی و نظامی نمود بیشتری داشته و در عملکرد اینگونه سیستم‌ها تاثیر بسزایی را خواهد داشت.

اهمیت بلادرنگ بودنِ فرآیند ردیابی اشیاء در حالی است که، در سال‌های اخیر شاهد استفاده از دوربین‌های با درجه تفکیک بالا در سیستم‌های ردیابی هستیم؛ در بسیاری از روش‌های ردیابی، این مسئله باعث افزایش میزان اطلاعاتی که باید در هر قاب از دنباله تصاویر پردازش شود، می‌گردد. همچنین با توجه به پیشرفت روز افزون تکنولوژی و افزایش انتظارات از سیستم‌های هوشمند، در سال‌های اخیر روندهای پردازشی پیچیده‌تر شده‌اند. لذا صرفا با انجام عملیات نرم افزاری دستیابی به پردازش‌های بلادرنگ، در اینگونه مسائل، قابل حصول نیست و استفاده از قابلیت‌های سخت افزاری و پردازش موازی برای حصول نتیجه مطلوب ضروری می‌نماید.

در این پایان نامه به منظور ردیابی بلادرنگ اشیاء، با فرض ثابت بودن دوربین، از پردازش موازی به وسیله واحد پردازنده گرافیکی در کنار واحد پردازنده مرکزی استفاده می‌کنیم. برخی از الگوریتم‌های موجود در زمینه ردیابی اشیاء، مانند تفاضل قاب، تفاضل قاب سه‌تایی و فیلتر ذره‌ای را بر روی واحد پردازنده گرافیکی و واحد پردازنده مرکزی پیاده‌سازی می‌کنیم؛ و نهایتا با مقایسه عملکرد زمانی الگوریتم‌های موازی شده بر روی ویدئوهای با ابعاد مختلف و پیچیدگی‌های متفاوت، نشان می‌دهیم موازی سازی فرآیند ردیابی اشیاء با بهره گیری از واحد پردازنده گرافیکی در کنار واحد پردازنده مرکزی به منظور دستیابی به ردیابی بلادرنگ رویکردی مناسب خواهد بود؛

نتایج بدست آمده نشان می‌دهد در صورت رفع مشکل زمان زیاد انتقال داده‌ها بین پردازنده مرکزی و پردازنده گرافیکی، سرعت پردازش تا بیش از هزار برابر بهبود پیدا می‌کند. بطوریکه در فرایند فیلتر ذره‌ای، اجرای الگوریتم توسط پردازنده گرافیکی برای دوازده هزار ذره نسبت به اجرای الگوریتم توسط پردازنده مرکزی، منجر به افزایش سرعت ۱۱۳۰ برابری می‌گردد. همچنین با توجه به ابعاد ویدئو، استفاده از پردازنده گرافیکی، منجر به افزایش سرعت برای فرایندهای تفاضل قاب و تفاضل قاب سه-تایی به ترتیب تا ۱۳۹۰ و ۵۴۸ برابر، خواهد شد.

کلمات کلیدی: ردیابی بلادرنگ اشیاء، پردازش موازی، واحد پردازنده گرافیکی، OpenCL، فیلتر ذره‌ای، تفاضل قاب

فهرست مطالب

فصل اول: مقدمه ۱

۱-۱ مقدمه ۲

۱-۲ بیان اهداف و تعریف مسئله ۳

۱-۳ ساختار پایان نامه ۴

فصل دوم: پیشینه پژوهش ۵

۲-۱ مقدمه ۶

۲-۲ تشخیص شئ ۹

۲-۲-۱ تفاضل قاب ۱۰

۲-۲-۲ تفریق پس‌زمینه ۱۰

۲-۲-۳ جریان نوری ۱۱

۲-۳ دسته‌بندی شئ ۱۱

۲-۳-۱ دسته‌بندی مبتنی بر شکل ۱۲

۲-۳-۲ دسته‌بندی مبتنی بر حرکت ۱۲

۲-۳-۳ دسته‌بندی مبتنی بر رنگ ۱۳

۲-۳-۴ دسته‌بندی مبتنی بر بافت ۱۳

۲-۴ ارائه و ردیابی شئ مورد نظر ۱۴

۲-۴-۱ نمایش مبتنی بر نقطه ۱۵

۲-۴-۲ نمایش مبتنی بر هسته ۱۶

۲-۴-۳ نمایش مبتنی بر سایه ۱۶

۲-۵ مروری بر پژوهش‌های ردیابی اشیاء ۱۷

۲-۶ جمع‌بندی ۱۹

فصل سوم: مبانی نظری پژوهش ۲۱

۳-۱ مقدمه ۲۲

۳-۲ تفاضل قاب ۲۲

۳-۳ تفاضل قاب سه‌تایی ۲۴

۳-۴ فیلتر ذره‌ای ۲۶

۳-۵ واحد پردازنده گرافیک ۲۸

۳-۵-۱ عملکرد پردازنده‌های گرافیکی ۳۱

۳-۵-۲ زبان‌های برنامه نویسی پردازنده گرافیکی ۳۲

۳-۵-۳ معماری پردازنده‌های گرافیکی ۳۳

۳-۶ جمع‌بندی ۳۴

فصل چهارم: روش پیشنهادی ۳۵

۴-۱ مقدمه ۳۶

۴-۲ شرح مسئله ۳۶

۴-۳ روش پیشنهادی ۳۷

۴-۳-۱ تفاضل قاب ۴۴

۴-۳-۲ تفاضل قاب سه‌تایی ۴۵

۴-۳-۳ فیلتر ذره‌ای ۴۶

۴-۴ جمع‌بندی ۴۸

فصل پنجم: نتایج تجربی ۵۱

۵-۱ مقدمه ۵۲

۵-۲ تشخیص اشیاء متحرک ۵۲

۵-۳ ردیابی اشیاء ۵۷

۵-۴ بررسی نتایج و تحلیل یافته‌ها ۵۹
۵-۵ جمع‌بندی ۶۱
فصل ششم: جمع‌بندی ۶۵
۶-۱ جمع‌بندی ۶۶
۶-۲ پیشنهادات ۶۷
مراجع ۶۸

فهرست اَشکال

شکل ۲-۱: دیاگرام مراحل ردیابی شیء ۷

شکل ۲-۲: روش‌های متداول تشخیص شئ ۹

شکل ۲-۳: روش‌های متداول دسته‌بندی شئ ۱۲

شکل ۲-۴: روش‌های متداول ردیابی شئ ۱۴

شکل ۲-۵: نمونه روش‌های ارائه شئ ۱۵

شکل ۳-۱: نمایش تاثیر انتخاب حدآستانه‌های متفاوت در روش تفاضل قاب ۲۴

شکل ۳-۲: تفاوت عملکردی تفاضل قاب و تفاضل قاب سه‌تایی ۲۵

شکل ۳-۳: مدل عمومی فضای حالت ۲۶

شکل ۳-۴: مقایسه عملکرد محاسباتی بر اساس رشد عملیات ممیز شناور ۲۹

شکل ۳-۵: مقایسه عملکردی بر اساس پهنای باند ۳۰

شکل ۳-۶: تغییرات مشخصه‌های کلیدی پردازنده‌های گرافیکی ۳۱

شکل ۳-۷: ساختار واحد پردازنده گرافیک در مقابل واحد پردازنده مرکزی ۳۴

شکل ۴-۱: مدل حافظه در معماری OpenCL ۳۹

شکل ۴-۲: معماری کارت گرافیک توسط OpenCL در مقایسه با یک مدرسه ۴۰

شکل ۴-۳: نمونه‌ای از یک مسئله‌ی محاسباتی مناسبِ موازی سازی ۴۱

شکل ۴-۴: نمونه کرنل نوشته شده برای حل مسئله شکل (۴-۳) با پردازش موازی توسط OpenCL ۴۲

شکل ۴-۵: نحوه اجرای برنامه موازی سازی شده توسط OpenCL ۴۲

شکل ۴-۶: مثالی از مراحل لازم برای بکارگیری واحد پردازنده گرافیک در کنار پردازنده مرکزی در سمت میزبان ۴۴

شکل ۴-۷: مرحله پیش‌بینی در فرآیند ردیابی اشیاء با استفاده از فیلتر ذره‌ای ۴۷

شکل ۴-۸: مرحله بروزرسانی در فرآیند ردیابی اشیاء با استفاده از فیلتر ذره‌ای ۴۷

شکل ۵-۱: چند قاب از ویدئو تهیه شده برای آزمایشات ۵۳

شکل ۵-۲: نسبت زمان کل پردازش غیر موازی بر زمان کل پردازش موازی با توجه به ابعاد ویدئو .. ۵۶

شکل ۵-۳: نسبت زمان پردازش پردازنده مرکزی بر زمان پردازش پردازنده گرافیکی با توجه به ابعاد ویدئو ۵۶

شکل ۵-۴: نسبت زمان کل پردازش غیر موازی بر پردازش موازی با توجه به تعداد ذرات در فیلتر ذره‌ای ۵۹

شکل ۵-۵: نسبت زمان پردازش پردازنده مرکزی بر پردازنده گرافیکی با توجه به تعداد ذرات در فیلتر ذره‌ای ۵۹

شکل ۵-۶: محل قرارگیری دوربین‌ها برای تهیه پایگاه داده PETS از سال ۲۰۱۴ به بعد ۶۲

شکل ۵-۷: ردیابی اشیا در نمونه ویدئوهای مورد استفاده ۶۴

فهرست جداول

جدول ۵-۱: مشخصات رایانه مورد استفاده ۵۲

جدول ۵-۲: زمان متوسط پردازش هر قاب توسط پردازنده مرکزی در روش‌های تشخیص اشیاء متحرک (میلی‌ثانیه) ۵۳

جدول ۵-۳: زمان متوسط پردازش هر قاب توسط پردازنده گرافیکی در روش‌های تشخیص اشیاء متحرک (میلی‌ثانیه) ۵۴

جدول ۵-۴: زمان متوسط انتقال داده در هر قاب بین پردازنده مرکزی و پردازنده گرافیکی (میلی-ثانیه) ۵۵

جدول ۵-۵: زمان متوسط کل پردازش حالت موازی هر قاب در روش‌های تشخیص اشیاء متحرک (میلی‌ثانیه) ۵۵

جدول ۵-۶: زمان متوسط پردازش هر قاب بر اساس تعداد ذرات در فیلتر ذره‌ای (میلی‌ثانیه) ۵۸

فصل اول: مقدمه

۱-۱ مقدمه

تشخیص حرکت و ردیابی هدف، یکی از مسائل مهم در زمینه پژوهش‌ها و برنامه‌های مرتبط با حوزه بینایی ماشین است که کاربردهای فراوانی دارد. بطور کلی ردیابی هدف را می‌توان فرآیند تخمین مسیر یک شیء در قاب‌های متوالی عنوان کرد.

امروزه با توجه به پیشرفت روز افزون فناوری و جهت گیری کلی جوامع بشری به سمت آن، و همچنین نیاز روز افزون به خودکار سازی و افزایش سرعت و کارایی در کاربردهای مختلف، استفاده از فناوری در زمینه‌های گوناگون اجتناب ناپذیر است. استفاده از سخت افزارهای پیشرفته و افزایش انتظارات مصرف کنندگان از سیستم‌های هوشمند، از جمله مسائلی است که در سال‌های اخیر، فرآیندهای پردازشی را پیچیده‌تر کرده است.

اگرچه سابقه موضوع ردیابی اشیا به مسائل نظامی برمی‌گردد ولی امروزه به دلیل کاربردهای بسیار گسترده آن در زمینه‌های مختلفی مانند کنترل ترافیک، تشخیص حرکات غیر معمول، سیستم‌های نظارتی و رباتیک و ... این موضوع مورد توجه ویژه‌ای قرار گرفته است [۱].

از جمله مسائلی که همواره عملکرد فرآیندهای ردیابی را با مشکل مواجه ساخته است، تعامل آن با روش‌های تشخیص هدف، ظاهر متغیر اهداف و همچنین ردیابی همزمان چندین هدف است [۲]. با توجه به پیشرفت روز افزون فناوری و استفاده از دوربین‌هایی با قابلیت فیلم برداری با درجه تفکیک[1] بالا، پردازش ویدئو به صورت بلادرنگ[2] را مشکل‌تر می‌کند.

بنابراین برای دست‌یابی به نتیجه‌ای مطلوب در سیستم‌های ردیابی هدف، و برطرف ساختن مسائل مطرح شده نیازمند استفاده از فرآیندهایی با دقت بالا هستیم که عمدتا زمان بر می‌باشند. این در

[1] Resolution
[2] Real time

حالی است که اکثر سیستم‌های کاربردی در این حوزه نیازمند پردازش‌های بلادرنگ هستند. باید توجه داشت که انجام عملیات‌های پردازشی پیچیده در پردازش‌های بلادرنگ، صرفا با انجام عملیات نرم افزاری قابل حصول نیست و لذا استفاده از قابلیت‌های سیستم‌های سخت افزاری و پردازش موازی برای حصول نتیجه مطلوب ضروری می‌نماید [۳,۴].

۱-۲ بیان اهداف و تعریف مسئله

در این پایان نامه به منظور کاهش زمان محاسباتی عملیات پردازشی، در فرآیند ردیابی اشیاء از واحد پردازنده گرافیکی[1] استفاده می‌کنیم. با توجه به امکانات پردازش موازی که این پردازشگرها در اختیار ما قرار می‌دهند، انتظار می‌رود سرعت انجام محاسبات در مقایسه با پردازش توسط واحد پردازنده مرکزی[2] بهبود یابد.

واحد پردازنده گرافیک به طور قابل ملاحظه‌ای دارای واحدهای پردازشی موازی بیشتری نسبت به واحد پردازنده مرکزی می‌باشد. لذا استفاده مناسب از امکانات پردازش موازی این نوع پردازنده‌ها می‌تواند باعث تسریع انجام محاسبات گردد. بنابراین با توجه به ساختار واحد پردازنده گرافیک که از هسته‌های متعددی تشکیل شده و بعضا تعداد این هسته‌ها به دو هزار و پانصد عدد هم می‌رسد [۵]، برآنیم که با بکارگیری این ویژگی در قالب پردازش موازی با پیاده سازی الگوریتم‌های موجود در زمینه ردیابی اشیا برروی بستر پردازنده گرافیکی به ردیابی بلادرنگ اشیا در دنباله‌ای از تصاویر ویدئویی بپردازیم.

[1] Graphics processing unit (GPU)
[2] Central processing unit (CPU)

نهایتا تعدادی از الگوریتم‌های موجود برای طی روند ردیابی اشیاء را، با استفاده از واحد پردازنده گرافیکی در کنار واحد پردازنده مرکزی، پیاده سازی می‌کنیم. با مقایسه عملکرد زمانی فرآیند موازی شده در مقابل فرآیند در حالت غیر موازی، در ویدئوهای با ابعاد متفاوت و پیچیدگی‌های مختلف، تاثیر موازی سازی و بهره گیری از واحد پردازنده گرافیکی را در بلادرنگ سازی فرآیند بررسی کرده و مورد سنجش قرار می‌دهیم.

۱-۳ ساختار پایان نامه

مطالب این پایان نامه در شش فصل تنظیم گردیده است. در ادامه، در فصل دوم به بیان پیشینه و اهمیت ردیابی اشیا می‌پردازیم. همچنین تعدادی از روش‌های متداول که به منظور ردیابی اشیا استفاده می‌شوند را بیان می‌کنیم. در فصل سوم مباحث نظری را که در مرحله پیاده سازی به آنها احتیاج داریم، به تفصیل مورد بررسی قرار می‌دهیم. در فصل چهارم روشی متدوال و ترکیبی از روش-های ردیابی اشیاء را ارائه می‌دهیم و به بیان پیاده سازی و موازی سازیِ روشِ پیشنهاد شده با استفاده از واحد پردازنده گرافیکی می‌پردازیم. فصل پنجم به نتایج حاصل از روش پیشنهادی و تاثیر موازی سازی در بلادرنگ کردن فرآیند اختصاص یافته است. و نهایتا در فصل ششم نتیجه‌گیری و پیشنهاد برای کارهای آینده بیان شده است.

فصل دوم: پیشینه پژوهش

۲-۱ مقدمه

ردیابی بلادرنگ شیء به فرآیند تخمین مسیر یک شیء در قاب‌های متوالی گفته می‌شود [۶]. هدف اصلی ردیابی شیء پردازش بخش‌های مورد نظر یک یا چند شیء در ویدئو است، بطوریکه با تغییرات حرکتی، انسداد[1] شئ و تغییر جهت شئ، ردیابی دچار مشکل نشود. ردیابی شیء مورد نظر با استفاده از تکنیک‌های پردازش تصویر برای کاربردهای مختلف جالب و چالش برانگیز است.

اخیرا پردازش تصویر به یک حوزه بزرگ تحقیقاتی تبدیل شده است. سیستم ردیابی بلادرنگ شئ، یکی از این حوزه‌ها است. دشواری‌ها و چالش‌هایی در طول مراحل مختلف ردیابی شیء ظاهر می‌شوند. تکنیک‌های زیادی در پردازش تصویر وجود دارند که بر این مشکلات غلبه می‌کنند. استفاده صحیح و بجای این تکنیک‌ها در انواع مختلف برنامه‌های بینایی کامپیوتر نظیر کنترل ترافیک، هدایت ربات، ردیابی افراد، حرکت و امنیت اهمیت ویژه‌ای پیدا می‌کنند. مراقبت ویدئویی مهم ترین و مفیدترین کاربرد ردیابی بلادرنگ شئ است.

مراقبت ویدئویی باید در برابر پارازیت و اختلال‌های محیطی مقاوم باشد. مراقبت ویدئویی، از فعالیت‌های مجرمانه و تصادفات جلوگیری می‌کند و وضعیت‌های مشکوک را تحت نظر می‌گیرد، بنابراین بلادرنگ بودن سیستم در این حوزه بسیار با اهمیت است.

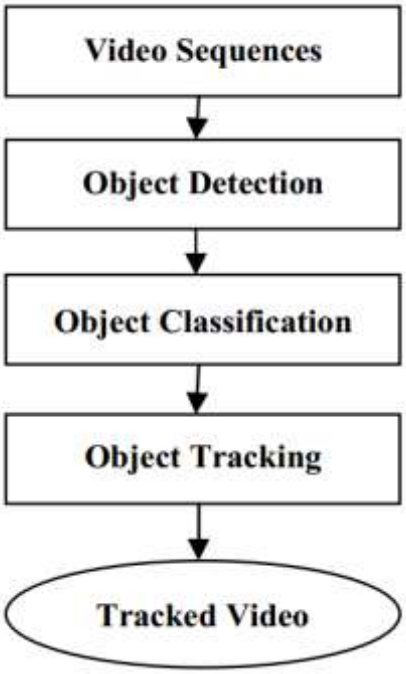
در مسئله ردیابی اشیاء، قبل از آنکه فرآیند ردیابی آغاز شود، ابتدا نیاز به تشخیص اشیاء متحرک و جداسازی از پس زمینه[2] می‌باشد [۷]. برای جداسازی شئ از پس زمینه مشکلاتی از قبیل وجود سایه، تغییر شدت روشنایی محیط، نویز و ... وجود دارد؛ که شئ به خوبی از تصویر پس زمینه استخراج

[1] Occlusion
[2] Background

نمی‌شود. برای این منظور باید پردازش‌هایی روی تصویر و پیش زمینه[1] استخراج شده انجام داد تا کیفیت شئ استخراج شده بهبود پیدا کند.

بعد از استخراج اشیاء متحرک (پیش زمینه)، که به عنوان ورودی سیستمِ ردیابی می‌باشد باید عمل تشخیص و ردیابی هدف انجام شود [۸]. بطور کلی ردیابی را می‌توان عمل بدست آوردن، استنتاج و تخمین تغییرات زمانی-مکانی هدف در طول دنباله ویدئویی بر اساس اندازه‌گیری‌ها و مشاهدات تعریف کرد.

مراحل مختلف یا گام های اصلی ردیابی شیء در شکل (۲-۱) نشان داده شده است [۸]. هـر فرآینـد ردیابی اشیاء برای ردیابی کردن، نیاز به طیِ این مراحل اصلی دارد. شرح کامل هر مرحلـه در بخـش-های بعدی ارائه شده است.



شکل ۲-۱: دیاگرام مراحل ردیابی شیء [۸]

[1] Foreground

بعضی از مفاهیم که در ردیابی بسیار کاربرد دارد به صورت زیر بیان می‌گردد:

- **تشخیص یا شناسایی اشیاء:** قطعه‌بندی تصویر به اشیاء و پس زمینه اولین و یکی از مهم-ترین گام‌ها برای ردیابی است. شناسایی اشیاء از پس زمینه، تحت عواملی مثل تغییر شدت روشنایی، شباهت رنگ اشیاء با پس زمینه، کیفیت پایین تصاویر، حرکت پس زمینه مثل برگ درختان و سایه اشیاء، کاری نسبتا دشوار می‌باشد.

- **ظاهر اشیاء:** ظاهر اشیاء ممکن است بسته به شرایط و زاویه دید متفاوت باشد. مهم‌ترین دلیل آن می‌تواند، تنوع در شکلِ جسم باشد. به عنوان مثال ظاهر متغیر شئ می‌تواند ناشی از حرکت قسمت‌های مختلف بدن انسان باشد. به طور مشابه، زاویه دید و فاصله جسم نسبت به دوربین می‌تواند دلیل دیگری، برای انطباق ناصحیح اشیاء باشد. ظاهر شئ در زمانی که مشاهدات از جلو، پشت و سمت‌های دیگر باشد، بسیار متفاوت است. به طور مشابه، ظاهر جسمِ نزدیک به یک دوربین ممکن است بسیار متفاوت نسبت به زمانی که جسم فاصله دارد، باشد. مشکلات انسداد یکی از مشکلات بزرگ در استفاده از ظاهر افراد می‌باشد چون زمانی که بخشی از قسمت‌های یک جسم قابل روئیت نباشد، ظاهر نمی‌تواند برای شناسایی افراد موثر باشد.

- **شناسایی مجدد شئ:** زمانی که یک شئ برای لحظاتی از تصویر ناپدید می‌شود و مجددا در تصویر ظاهر می‌شود، سیستم باید تصمیم بگیرد که این شئ در مشاهدات گذشته وجود داشته و یا یک شئ جدید می‌باشد. از جمله دلایل متداول ناپدید شدن شئ را در دنباله تصاویر، می‌توان انسداد اشیاء ذکر کرد.

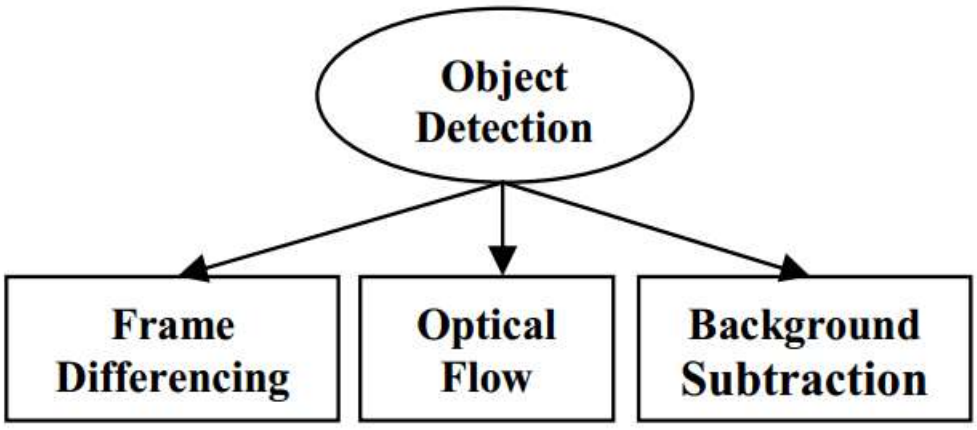
در ادامه این فصل به بیان مراحل فرآیند ردیابی شئ می‌پردازیم؛ هریک از این مراحل را به تفصیل بیان کرده و روش‌های متداول هر مرحله را مرور می‌کنیم. در بخش (۲-۲) روش های تشخیص شیء

را شرح می دهیم. بخش (۲-۳) مربوط به دسته بندی اشیاء است. بخش (۲-۴) تکنیک‌های مختلف ردیابی شیء را ارائه می دهد. و نهایتا در این فصل، تعدادی از منابع و پژوهش‌هایی که تا کنون در مورد ردیابی اشیاء انجام شده است را مرور خواهیم کرد.

۲-۲ تشخیص شئ

هر فرآیند ردیابی شئ، باید در مرحله اولِ خود، تشخیص شئ را داشته باشد. چرا که اشیاء مورد نظر را در توالی‌های ویدئویی بتواند تشخیص دهد.

تشخیص شئ، به مکان یابی شئِ درحال حرکت در قاب‌های ویدئویی متوالی گفته می‌شود [۹]. این مرحله از فرآیند را می‌توان با تکنیک‌های مختلفی انجام داد. متداول‌ترین این روش‌ها که بر اساس حرکت دسته‌بندی می‌شوند در شکل (۲-۲) نشان داده شده است.



شکل ۲-۲: روش‌های متداول تشخیص شئ [۸]

۲-۲-۱ تفاضل قاب[1]

در این روش، قاب‌های متوالی از دنباله تصاویر به عنوان ورودی هستند. و خروجی با توجه بـه تفـاوت پیکسل‌های ورودی ایجاد می‌شود. ساده‌ترین و متداول‌ترین ایده بـرای ایـن روش اسـتفاده از دو قـاب متوالی از دنباله تصاویر است و استفاده از یک حد آستانه[2] ثابت که خروجی را ایجاد می‌کنند. خروجی نتیجه تفریق مقادیر پیکسل‌های متناظر قاب‌های ورودی است، که حاصل این تفریق با توجه بـه حـد آستانه، پیکسل‌ها را به عنوان پس زمینه یا پیش زمینه دسته بندی می‌کند. این روش در کاربردهـایی با دوربین ثابت مناسب است. پیاده‌سازی این روش نسبت به دیگر روش‌ها، آسان است و بار محاسباتی زیادی ندارد. اما زمان پردازشی این فرآیند کاملا با ابعاد ویدئو در ارتباط است و با افزایش ابعاد ویـدئو، بار محاسباتی آن و زمان پردازش افزایش می‌یابد. در این روش تشخیص شیء نتیجـه دقیقـی را ارائـه نمی‌دهد و معمولا فقط نقاط مربوط به لبه‌های هدف را آشکار می‌سازد [۱۰،۱۱].

۲-۲-۲ تفریق پس‌زمینه

روش تفریق پس زمینه در ابتدا نیازمند تقریب مدلی برای پس زمینه و پس از آن بروزرسانی این مدل است. به عنوان مثال، در یک روش ساده می‌توان یک تصویر میانگین از قاب‌ها را، به عنوان مدل پـس زمینه انتخاب نمود [۱۲].

در این روش پس از تهیه مدل پس زمینه، با مقایسه هر قاب و پس زمینه‌ی تقریب زده شـده، اشـیاء متحرک شناسایی می‌شوند [۱۳]. به بیان ساده‌تر، برای این کار تصویر پس زمینه‌ی مرجع، بـا تصـویر حاضر مقایسه شده و تفاوت در مقادیر پیکسل‌ها بین هر قاب پیدا می‌شود. زمانی که تفاوت تشـخیص

[1] Frame difference
[2] Threshold

داده شد، شیء به عنوان یک شیء درحال حرکت تشخیص داده می‌شود. کارایی این روش برای ویدئو-های با دوربین ثابت تا حد قابل قبولی اثبات شده است.

۲-۲-۳ جریان نوری[1]

جریان نوری به معنی توزیع سرعت جابجایی روشنایی الگوها بین چند تصویر می‌باشد که در آن اجسام متحرک به صورت مجموعه‌ای از بردارهای سرعت نشان داده می‌شوند. در دنباله تصاویر ویدئویی، با ردیابی و جست و جوی نقاط متناظر از تصویر اصلی می‌توان جابجایی نقاط متحرک را تحلیل کرد. برای محاسبه جریان نوری نیاز به استفاده از حداقل دو قاب است. جریان نوری یک نقطه از تصویرِ ویدئو، بردار جابجایی مکان و سرعت آن نقطه را نتیجه می‌دهد.

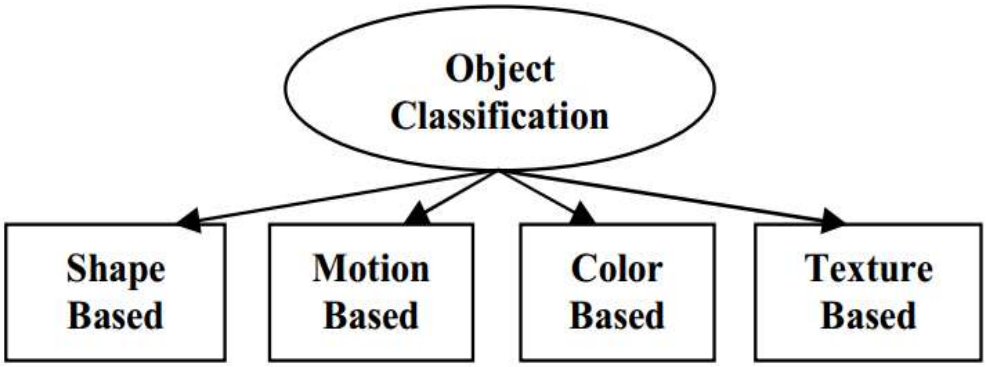
در این روش فرض می‌شود اشیاء متحرک در صحنه، دارای حرکت مستقلی از حرکت پس زمینه هستند. حرکت پس زمینه می‌تواند، در اثر حرکت دوربین ایجاد شده باشد. از جمله مزیت‌های این روش نسبت به دو روش قبلی این است که برای کاربردهای با دوربین متحرک هم قابل استفاده است. اما در صورتی که شئ دارای حرکت صلب نباشد یا به عبارت دیگر، اجزاء مختلف یک شئ دارای بردارهای مختلف حرکتی باشند، دسته بندی براساس شباهت بردارها، منجر به جداسازی و تشخیص صحیح شئِ متحرک نخواهد شد [۱۴].

۲-۳ دسته‌بندی شئ

پس از اولین مرحله‌ی فرآیند ردیابی شئ، تشخیص اشیاء متحرک کامل می‌شود. این اشیاء تشخیص داده شده ممکن است هرچیزی مثل خودرو، انسان، درخت، پرنده یا اشیاء دیگری باشند. برای دسته

[1] Optical flow

بندی این اشیاء تشخیص داده شده، دسته‌بندی شئ به عنوان دومین مرحله در فرآیند ردیـابی شـئ، اعمال می‌شود. دسته بندی شئ روش‌های زیادی دارد که در شکل (۲-۳) متداول‌تـرین ایـن روش‌هـا نشان داده شده است.



شکل ۲-۳: روش‌های متداول دسته بندی شئ [۸]

۲-۳-۱ دسته‌بندی مبتنی بر شکل

در این نوع دسته‌بندی، ویژگی‌های شکل برای شناسایی و ردیابی اشیاء استفاده می‌شود. تشخیص لبه، تطبیق مرز شئ، مساحت و اندازه، از جمله ویژگی‌های شکل شئ محسوب می‌شوند که برای ردیابی مورد استفاده قرار می‌گیرند. عموما این روش برای شرایط پویا که شئ از شکل ثابتی برخوردار نیست (مانند حرکت انسان که دارای حرکت دست‌ها و پاها است) به خوبی عمل نمی‌کند [۸].

۲-۳-۲ دسته‌بندی مبتنی بر حرکت

در این نوع دسته بندی، در قاب‌های متوالی از دنباله ویدئویی، از ویژگی‌هایی از شئ استفاده می‌شـود که بیانگر حرکت شیء باشند؛ مانند مکان، سرعت و شتاب شئ. شار نوری از جمله معروف‌ترین روش-هایی است که از اطلاعات مبتنی بر حرکت شئ استفاده می‌کند.

استفاده کردن از دسته بندی مبتنی بر حرکت به تنهایی، شناسایی شئ را در لحظات سکون با مشکل مواجه می‌کند. در ردیابی بر پایه حرکت به دلیل پیچیدگی این نوع از سیستم‌ها، فرض‌هایی را برای ردیابی در نظر می‌گیرند. به عنوان نمونه در [۱۵] فرض شده است که نرخ قاب[1] ویدئو به اندازه کافی بالا می‌باشد. بنابراین موقعیت مکانی شئ به میزان قابل توجهی بین قاب‌های متوالی از دنباله ویدئویی تغییر نخواهد کرد.

۲-۳-۳ دسته‌بندی مبتنی بر رنگ

اگر قاب‌های دنباله ویدئویی، تصاویر رنگی باشد در این صورت روش دسته بندی مبتنی بر رنگ جهت دسته بندی کردن شیء با استفاده از ویژگی رنگ، می‌تواند اعمال شود. برای کاربردهای بلادرنگ، تکنیک مبتنی بر هیستوگرام رنگ به علت بار پردازشی کم و پیاده سازی نسبتا آسان، در ردیابی شیء مبتنی بر رنگ متداول است. از جمله فضاهای رنگی مورد استفاده می‌توان فضاهای رنگی RGB و HSV را نام برد.

۲-۳-۴ دسته‌بندی مبتنی بر بافت

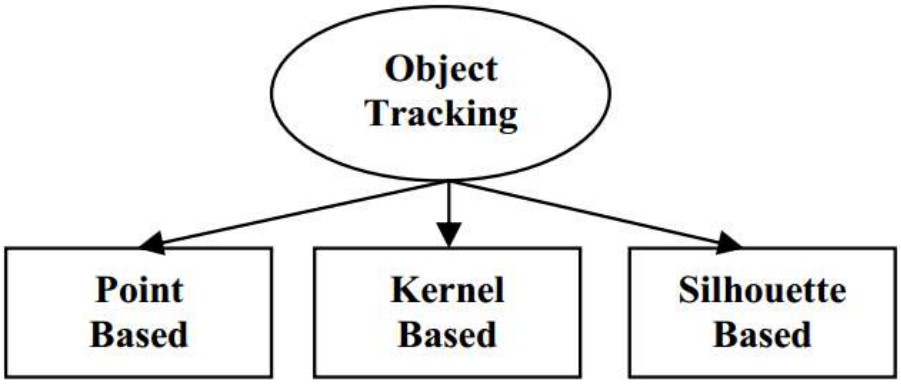
بافت[2]، نوسان شدت روشنایی سطح است. و یک ویژگی مهم برای انواع مختلف اشیاء در تصویر است. دسته بندی بافت شامل دو فاز است؛ یادگیری و تشخیص. فاز یادگیری منجر به مجموعه‌ای از ویژگی-های بافت برای شیء می‌شود. فاز تشخیص ویژگی، بافتی را که بهترین مطابقت را دارد، انتخاب می-کند. از جمله نقاط ضعف این روش‌ها بالا بودن زمان پردازش است که برای کارهای بلادرنگ مناسب نیستند. به عنوان نمونه، الگوریتم‌های SIFT و SURF در منبع [۱۶] برای دسته بندی مبتنی بر بافت مورد استفاده قرار گرفته‌اند.

---

[1] Frame rate
[2] Texture

۲-۴ ارائه[1] و ردیابی شئ مورد نظر

نحوه نمایش شئ ردیابی شده با توجه به خصیصه شئ قابل تعیین است. به عنوان مثال کوچک یا بزرگ بودن شئ، صلب یا غیر صلب بودن و ... . انواع اصلی روش های ارائه شئ، در شکل (۲-۴) نشان داده شده است. اهمیت انتخاب نوع نمایش شئ آنجاست که این نمایش فضای استخراج ویژگی را نیز مشخص می‌کند. به عنوان مثال در نمایش نقطه‌ای از مجموعه نقاط و در نمایش مبتنی بر هسته از ناحیه‌ای که شکل هندسیِ شئِ مورد نظر مشتمل بر آن است استخراج ویژگی می‌شود. بنابراین مرحله تشخیص شئ و ارائه شئ (ردیابی شئ) به یکدیگر وابسته هستند.



شکل ۲-۴: روش‌های متداول ردیابی شئ [۸]

به عنوان نمونه روش‌های متداول نمایش شئ، در شکل (۲-۵) نشان داده شده است. در این شکل شئ مورد ردیابی، انسان فرض شده است.

[1] Representation

(الف) نمایش مبتنی بر نقطه (ب) نمایش مبتنی بر هسته (ج) نمایش مبتنی بر هسته

(د) نمایش مبتنی بر سایه-شکل (ه) نمایش مبتنی بر سایه-کرانه

شکل ۲-۵: نمونه روش‌های ارائه شئ. [۱۷]

۲-۴-۱ نمایش مبتنی بر نقطه

شئ را می‌توان با یک نقطه یا مجموعه‌ای از نقاط نمایش داد. در شکل (۲-۵-الف) نمونه‌ای از این نوع نمایش قابل مشاهده است. عموما و در حالت کلی، نمایش شئ با مجموعه نقاط برای ردیابی اشیائی که ناحیه کوچکی از تصویر را اشغال می‌کنند مناسب است [۱۹،۱۸]. روش های مختلفی وجود دارند که از نمایش مبتنی بر نقطه برای شئ استفاده می کنند. از معروف‌ترین روش‌های ردیابی که نمایش شئ به صورت نقطه‌ای است می‌توان به فیلتر ذره‌ای[1] اشاره کرد.

[1] Particle filter

۲-۴-۲ نمایش مبتنی بر هسته

در این نوع نمایش شئ، شکل شئ با یکی از اشکال هندسی پایه مانند مستطیل یا بیضی قابل نمایش است [۱۷]. شکل (۲-۵-ب) و (۲-۵-ج) نمونه‌ای از این نوع نمایش را نشان می‌دهند. اگرچه این نوع نحوه نمایش برای اشیاء صلب و غیر مفصلی مناسب هستند، اما بعضا برای نمایش اشیاءِ غیر صلب و مفصلی هم مورد استفاده قرار می‌گیرند. از جمله شناخته شده‌ترین روش‌های ردیابی مبتنی بر هسته، می‌توان به جابجایی میانگین[۱] و ماشین بردار پشتیبان[۲] اشاره کرد.

۲-۴-۳ نمایش مبتنی بر سایه

در این روش نمایش شئ به صورت مرزی از شئ نمایش داده می‌شود. در شکل (۲-۵-د) و (۲-۵-ه) نمایشی از این نوع ارائه شئ قابل مشاهده است. اشیاء ممکن است شکل‌های پیچیده نظیر شانه‌ها، انگشت و دست داشته باشند که نمی‌توان با شکل های هندسی ساده آنها را به شکل مناسبی توصیف کرد. به همین منظور ردیابی مبتنی بر سایه جهت تعریف دقیق شکل شیء استفاده می‌شود. به طور کلی می‌توان گفت این نوع از ارائه شئ برای اشیاءِ پیچیده‌ی غیر صلب و مفصلی مناسب است [۸]. این نوع نمایش به دو زیر شاخه تقسیم می‌شود:

- **نمایش الگو یا کرانه**[۳]: در این نوع نمایش، مرز یا به عبارتی نقاط روی محیط شئ انتخاب و نمایش داده می‌شود. و شئ مورد نظر ما با یک مرز مدل می‌شود.
- **نمایش شکل:** در این نوع نمایش ناحیه‌ی درون مرز، سایه نمای شئ نامیده می‌شود و این سایه، نماینده شئ مورد نظر خواهد بود.

---

[1] Mean shift
[2] SVM (Support Vector Machine)
[3] Contour

۲-۵ مروری بر پژوهش‌های ردیابی اشیاء

ردیابی بلادرنگ اشیاء، یک حوزه گسترده تحقیقاتی است و کاربردهای فراوانی دارد. واضح است که، کاربری سیستم ردیابی و ویژگی‌های هدف، تعیین کننده‌ی روند فرآیند بوده و با توجه به انتظاراتی که از سیستم وجود دارد، هریک از روش‌های ذکر شده مورد استفاده قرار خواهند گرفت. در ادامه بطور خلاصه به معرفی برخی از پژوهش‌های انجام شده در حوزه ردیابی اشیاء خواهیم پرداخت.

منبع [۱۷] یکی از بهترین منابع مروری در زمینه ردیابی اشیاء است. در این پژوهش به بررسی روش-های ردیابی و بیان آنها پرداخته شده است. در این پژوهش چالش‌ها و مشکلات مسئله ردیابی و پیشنهاداتی برای رفع آنها ارائه شده است؛ بنابراین با توجه به آنها می‌توان روش مناسب ردیابی را با توجه به شرایط مسئله، ویدئو و شئ مورد ردیابی برگزید.

در مطالعه‌ای که در [۲۰] انجام شده است، به بررسی عملکرد تعدادی از روش‌های ردیابی پرداخته شده و مقایسه‌ای بین آنها بر اساس میزان دقت و مقاوت در مقابل ردیابی انجام شده است. در این پژوهش روش‌های فیلتر ذره‌ای، فیلتر کالمن، جابجایی میانگین و ردیابی با استفاده از ماشین بردار پشتیبان بررسی شده‌اند.

در پژوهش [۲۱] که در سال ۲۰۱۵ انجام شده، در ابتدا به مرور روش‌های ردیابی اشیاء اشاره شده است. در ادامه این پژوهش، بر اساس طرحِ ارائه هدف، روش بروزرسانی مدل و فرآیند جست وجوی شئ، روش‌های ردیابی اشیاء دسته‌بندی شده‌اند. همچنین در انتها بررسی گسترده‌ای از عملکرد سی و یک روش ردیابی ارائه شده است که طی دهه اخیر در مقالات و پژوهش‌ها ذکر شده‌اند. آزمایش‌های مختلفی از جمله دقت و سرعت فرآیند و همچنین عملکرد روش‌های ردیابی در مقابل چالش‌های انسداد و مواجه با ناحیه رنگی مشابه هدف برای ارزیابی این روش‌ها انجام شده است. عملکرد این

روش‌ها براساس پایگاه‌های داده مختلف نیز، بررسی شده و در جمع‌بندی نهایی روش‌های موثر ردیابی برای چالش‌های متفاوت معرفی شده است.

قابل ذکر است که، رویکرد این پایان نامه استفاده از واحد پردازنده گرافیکی برای تسریع فرآیند ردیابی است. واحد پردازنده گرافیکی در سال‌های اخیر در پژوهش‌های متعددی به منظور تسریع روند محاسبات استفاده شده است. واحد پردازنده گرافیکی، در حوزه پردازش تصویر و بینایی ماشین نیز کاربرد داشته است. در ادامه به تعدادی از پژوهش‌های مرتبط اشاره خواهیم کرد.

اولین مقالات در زمینه کاربرد واحد پردازنده گرافیکی در پردازش تصویر و بینایی ماشین در سال ۱۹۹۹ منتشر شدند. در مقاله [۲۲] مزایای استفاده از پردازنده‌های گرافیکی همه منظوره[1] (GPGPU) در پردازش تصویر مورد مطالعه و بررسی قرار گرفته است.

در مقاله [۲۳] با استفاده از پردازنده‌های گرافیکی و با بهره‌گیری از تکنیک موازی سازی، تصاویر به قطعاتی تقسیم‌بندی شده و هر قطعه به یک هسته[2] برای پردازش واگذار شده است. نتایج بدست آمده از این مقاله نشان می‌دهد که به سرعتی حدود ۶/۸ برابرِ حالتی خواهیم رسید که اطلاعات یک تصویر، تنها با بهره‌گیری از پردازنده مرکزی[3] و به صورت سری پردازش شود.

در مقاله [۲۴] الگوریتم معروف تشخیص لبه کنی[4] بر روی واحد پردازنده گرافیکی پیاده سازی شده است. با توجه به اینکه این الگوریتم در کاربردهای زیادی از پردازش تصویر، مورد استفاده قرار می-گیرد. نتایج بدست آمده از این پژوهش در کاربردهای این حوزه نیز قابل استفاده خواهد بود.

از جمله مقالات مرتبط به ردیابی اشیا با استفاده از واحد پردازنده گرافیک می‌توان به [۲۵] و [۲۶] اشاره کرد که با استفاده از الگوریتم شار نوری و بکارگیری واحد پردازنده گرافیکی به ردیابی بلادرنگ

---

[1] General-purpose computing on graphics processing units
[2] Thread
[3] CPU
[4] Canny

اشیاء پرداخته‌اند. از دیگر کارهای انجام شده در سال‌های اخیر می‌توان بهبود بخشیدن سرعت الگوریتم‌های جابجایی میانگین [۲۷] را نیز ذکر نمود.

۲-۶ جمع‌بندی

در این فصل موضوع ردیابی اشیاء و مراحل اصلی فرآیند آن را بیان کردیم. این مراحل بطور کلی شامل تشخیص شئ، دسته‌بندی شئ و ردیابی شئ می‌باشد. در ادامه این فصل تعدادی از روش‌های متداول هریک از مراحل فرآیند ردیابی ذکر شد و مهم‌ترین معایب و مزایای آنها را بیان کردیم. سپس چالش‌هایی که در مسئله ردیابی اشیاء با آن‌ها مواجه می‌شویم و راهکارهای مقابله با آنها را بیان کردیم و نهایتا تعدادی از پژوهش‌هایی که در سال‌های اخیر در ارتباط با موضوع ردیابی اشیاء انجام شده را مرور کردیم. در ادامه، در فصل بعد مباحث نظری مرتبط با پژوهش انجام شده در این پایان-نامه را بیان خواهیم کرد.

فصل سوم: مبانی نظری پژوهش

۳-۱ مقدمه

پیش از پرداختن به روش پیشنهادی و تحلیل نتایج، لازم است بعضی از مباحث که در ادامه به آنها نیاز داریم، بیان شوند.

همانطور که در فصل دوم بیان شد، در مسئله ردیابی اشیاء، قبل از آنکه فرآیند ردیابی آغاز شود، ابتدا نیاز به تشخیص اشیاء متحرک و جدا سازی از پس زمینه می‌باشد، پس از آن باید اشیاء تشخیص داده شده طبقه‌بندی شوند و نهایتا ردیابی انجام گردد. برای هریک از این مراحل روش‌های زیادی وجود دارد که با توجه به نوع شئ مورد نظر و ویژگی‌های ویدئو، قابل تعیین هستند.

در ادامه به بیان مبانی نظری روش‌هایی که در این پایان نامه استفاده کرده‌ایم خواهیم پرداخت. همچنین در نهایت با توجه به اینکه هدف اصلی این پایان‌نامه، که استفاده از پردازش موازی برای بلادرنگ کردن فرآیند ردیابی است، به بیان نکاتی که مربوط به فرآیند موازی سازی و سخت افزار مورد نیاز می‌باشد، خواهیم پرداخت.

بنابراین در این فصل در ابتدا به بیان روش‌های تفاضل قاب، تفاضل قاب سه‌تایی از بخش تشخیص اشیاء متحرک خواهیم پرداخت. سپس مبانی نظری مرتبط با فیلتر ذره‌ای را که برای ردیابی اشیاء استفاده کردیم را بیان خواهیم کرد. و نهایتا با توجه به اینکه فرآیند موازی سازی با استفاده از واحد پردازنده گرافیکی انجام شده است، به بیان نکات مورد نیاز در مورد واحد پردازنده گرافیکی و زبان‌های برنامه نویسی آن خواهیم پرداخت.

۳-۲ تفاضل قاب

تشخیص شئ متحرک در دنباله‌ای از تصاویرِ گرفته شده از یک دوربینِ ثابت، با استفاده از روش

تفاضل قاب بسیار متداول است. در این روش، ابتدا نویز دنباله تصاویر را با نرم کردن قاب‌ها کاهش می‌دهیم و سپس تفاضل آن‌ها را باینری می‌کنیم.

اکنون فرض کنید F_k، قابِ K-ام دنباله تصاویر، و F_{k+1}، قابِ k+1-ام باشد. بنابراین تفاضل این دو قاب برای ما تصویر باینریِ D(x,y) را مطابق با رابطه (۳-۱) ایجاد می‌کند [۲۸].

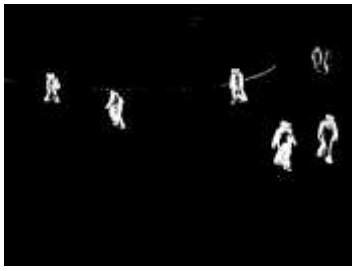
$$D(x,y) = \begin{cases} 1\ , & |F_k(x,y) - F_{k+1}(x,y)| \geq T \\ 0\ , & |F_k(x,y) - F_{k+1}(x,y)| < T \end{cases} \qquad (۳-۱)$$

که در آن T بیانگر یک حد آستانه است که به صورت تجربی حاصل می‌شود. باید توجه شود که انتخاب مقدار مناسب برای این حد آستانه بسیار مهم است؛ چراکه اگر حد آستانه بزرگ انتخاب شود در شئ متحرک فضاهای توخالی مشاهده می‌شود؛ مخصوصا برای قسمت‌هایی که یک ناحیه رنگی مشابه وجود داشته باشد. اگر این حد آستانه کوچک انتخاب شود باعث می‌شود قسمت‌هایی از پس-زمینه نیز به عنوان شئ متحرک شناسایی شود و یا به عبارتی نویز تصویر افزایش می‌یابد.

این روش در کاربردهایی با دوربین ثابت مناسب است. پیاده‌سازی این روش نسبت به دیگر روش‌ها، آسان است و بار محاسباتی زیادی ندارد. اما تشخیص شئ نتیجه دقیقی را ارائه نمی دهد و معمولا فقط نقاط مربوط به لبه‌های هدف را آشکار می‌سازد [۱۰،۱۱]. در شکل (۳-۱) می‌توان نتایج متفاوت را حاصل از انتخاب حدآستانه‌های متفاوت برای یک قاب مشخص مشاهده کرد. در تصویر (۳-۱-ب) حد آستانه از مقدار مناسب بیشتر انتخاب شده و در نتیجه قسمتی از اشیاء متحرک از دست رفته‌اند. در تصویر (۳-۱-د) حد آستانه بسیار کوچک‌تر از مقدار مناسب انتخاب شده و باعث ایجاد نویز در تصویر خروجی شده است. تصویر (۳-۱-ج) نمونه‌ای از خروجی فرآیند تفاضل قاب متوالی با انتخاب حد آستانه‌ای مناسب را نمایش می‌دهد.

(الف)

(د)



(ج)

(ب)

شکل ۳-۱: نمایش تاثیر انتخاب حدآستانه‌های متفاوت در روش تفاضل قاب

اگر در دنباله تصاویر، حرکت شئِ متحرک به صورت یکنواخت باشد، روش تفاضل قاب با انتخاب یک حدآستانه‌ی ثابتِ مناسب، برای یافتن پس زمینه و شی متحرک عملکرد خوبی از خود به نمایش می-گذارد؛ اما اگر حرکت شئِ متحرک به صورت شتاب دار باشد و مدام تغییر سرعت داشته باشد تعداد تشخیص‌های صحیحِ (بدون حفره یا بدون اضافات) شئِ متحرک کم می‌شود، چراکه مقدار حد آستانه را ثابت فرض کردیم و سرعت شئ متحرک با تغییر همراه است. برای رفع این مشکل می‌توان از تفاضل قاب سه‌تایی استفاده نمود.

۳-۳ تفاضل قاب سه تایی

در این روش پس از کاهش نویزِ تصاویر، قاب k-ام دنباله تصاویر، از قاب k-1-ام دنباله تصاویر کم شده و مطابق با رابطه (۳-۲) تصویر $D_1(x,y)$ ساخته می‌شود. سپس قابِ k+1-ام از قابِ k-م کم می‌شود

و تصویر $D_2(x,y)$ طبق رابطه (۳-۳) ساخته می‌شود. در نهایت با توجه به مقادیر $D_1(x,y)$ و $D_2(x,y)$ تصویر $D(x,y)$ طبق رابطه (۳-۴) بدست می‌آید و بیانگر تفاضل قاب سه‌تایی است. اینگونه تفاضل قاب سه‌تایی توسط ووچیک[1] و کامینسکی[2] ارائه شده است [۲۹].

$$D_1(x,y) = \begin{cases} 1 \; ; |F_k(x,y) - F_{k-1}(x,y)| \geq T \\ 0 \; ; |F_k(x,y) - F_{k-1}(x,y)| < T \end{cases} \quad (۳-۲)$$

$$D_2(x,y) = \begin{cases} 1 \; ; |F_{k+1}(x,y) - F_k(x,y)| \geq T \\ 0 \; ; |F_{k+1}(x,y) - F_k(x,y)| < T \end{cases} \quad (۳-۳)$$

$$D(x,y) = \begin{cases} 1 \; ; D_1(x,y) \cap D_2(x,y) = 1 \\ 0 \; ; D_1(x,y) \cap D_2(x,y) = 0 \end{cases} \quad (۳-۴)$$

در این روش اگرچه بازهم حد آستانه‌ی ثابتی در نظر گرفته می‌شود، ولی نتایج تجربی نشان می‌دهد که این روش نسبت به روش تفاضل قاب بهتر عمل می‌کند. بنابراین استفاده از تفاصل قاب سه تایی در مقابل تفاضل قاب، تعدادِ تشخیصِ صحیح برای شئِ متحرک با حرکتِ شتابدار را بیشتر می‌کند.

در شکل (۳-۲) تفاوت عملکردی این دو روش قابل مشاهده است. قابل ذکر است شکل (۳-۲) نشان دهنده نتایج در حالتی است که با آزمون و خطا بهترین حدآستانه‌ها برای دو روش انتخاب شده باشد.

(الف) قاب k-ام دنباله تصاویر ویدئو

(ب) تفاضل قاب

(ج) تفاضل قاب سه‌تایی

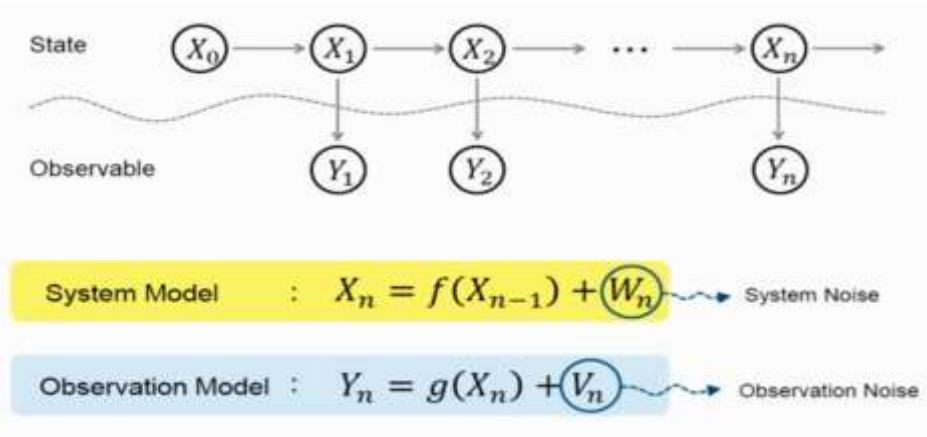
شکل ۳-۲: تفاوت عملکردی تفاضل قاب و تفاضل قاب سه‌تایی.

[1] Grzegorz M. Wojcik
[2] Wieslaw A. Kaminski

۳-۴ فیلتر ذره‌ای

ردیابی می‌تواند به عنوان تخمینی از مکان یک شئ در یک سکانس محسوب شود [۲۱]. فرض کنید شخصی را که از مقابل دوربین عبور می‌کند، ردیابی می‌کنیم. در هر قاب، موقعیت این شخص را تعیین می‌کنیم. اما باید توجه داشت که در هر قاب با یک تخمین موقعیت شخص در هر قاب روبرو هستیم. این تخمین احتمالا به دلایل بسیاری کاملا دقیق نیست. از جمله عوامل دقیق نبودن تخمین مکان هدف را می‌توان، عواملی مانند دقیق نبودن اندازه‌گیری‌ها، تقریب‌های مراحل قبلی پردازش، مسائل ناشی از انسداد، سایه‌ها و یا تغییر ظاهری اشیاء، ذکر کرد. مثلا زمانی که شخص در حال راه رفتن، پاها و دست‌هایش با حرکت او تاب می‌خورند. دلیل آن هر چیزی که باشد، انتظار داریم که این اندازه گیری‌ها حول مقادیر واقعی به صورت تصادفی تغییر کنند. می‌توانیم همه این عدم دقت‌ها را به شکل اضافه شدن نویز به فرآیند ردیابی خود ببینیم.

هدف ردیابی با استفاده از فیلتر ذره‌ای حدس زدن حالت شئ با یک سری از مشاهدات تا آن لحظه است [۳۰]. به عبارتی می‌توان گفت حالت فعلی یک سیستم، تابعی است از حالات قبلی سیستم به همراه احتمالا مقداری خطا. این مسئله در شکل (۳-۳) بطور ساده بیان شده است.



شکل ۳-۳: مدل عمومی فضای حالت [۳۱]

ایده اصلی استفاده از نظریه فیلتر ذره‌ای، بدست آوردن تابع چگالی احتمال مشروط بردار حالت، با توجه به بردار اندازه‌گیری‌ها و استفاده از تئوری بیزین[1] بدون هرگونه خطی سازی و صرفا با استفاده از مدل دینامیکی سیستم می‌باشد، با هدف بیشینه کردن احتمالِ صحتِ اندازه‌گیری‌های قبلی. یعنی مدل جدیدی که بعد از یک اندازه‌گیری می‌سازیم، با در نظر گرفتن عدم قطعیت مدل قبلی و هم خطا دار بودن اندازه‌گیری‌های جدید، مدلی است که بیشترین احتمال درست بودن را دارد. این روش جزء روش‌های آماری مونت کارلو[2] محسوب می‌شود و تابع توزیع احتمال شرطی را به صورت مجموع وزن دار تعدادی تابع گسسته تقریب می‌زند [۳۰].

اگرچه صحت و کاربرد این روش در مسائل ردیابی اثبات شده است، اما مشکلاتی نیز دارد. از جمله این مشکلات، در دسترس نبودن اطلاعاتی درباره تعداد نمونه‌های مورد نیاز برای دقت درخواست شده است.

هدف ردیابی به روش فیلتر ذره‌ای این است که بصورت برگشتی چگالی پیشین $P(x_t|z_{1:t})$ را برای وضعیت شئ حاضر X_t که بر طبق مشاهدات $Z_t=(Z_1,\ldots,Z_t)$ تا لحظه t عوض شده است را محاسبه کند. تابع چگالی احتمال[3] $P(x_t|z_{1:t})$ می‌تواند در دو مرحله به صورت بازگشتی بدست آورده شود: پیش‌بینی و بروزرسانی. اگر موقعیت X_t که با زمان تغییر می‌کند، به عنوان یک فرآیند درجه اول مارکوف مدل شود، برای تابع چگالی احتمال روابط (۳-۵) و (۳-۶) را خواهیم داشت [۳۲]:

$$P(x_t|z_{1:t})=KP(z_t|x_t)P(x_t|z_{1:t-1}) \quad (۳-۵)$$

$$P(x_t|z_{1:t-1})=\int P(x_t|x_{t-1})P(x_{t-1}|z_{1:t-1})dx_{t-1} \quad (۳-۶)$$

[1] Bayes' theorem
[2] Monte Carlo methods (or Monte Carlo experiments)
[3] PDF (Probability Density Function)

بـطوریکه K یک ضریب ثابت نرمالیزه کننده و مستقل از X_t است، $P(z_t|x_t)$ تابع شباهت است، $P(x_t|x_{t-1})$ مدل دینامیکی سیستم است، و $P(x_{t-1}|z_{1:t-1})$ از مشاهدات قبلی حاصل شده است. در فیلتر ذره‌ای الزامی نیست که $P(x_t|x_{t-1})$ و $P(z_t|x_t)$ توزیع گوسی داشته باشند.

در فیلتر ذره‌ای، چگالی پیشین مورد نیاز بوسیله یک مجموعه ذرات وزن دار شده، در هر زمان t تخمین زده می‌شود ($\{ (s_t^{(n)},\Pi_t^{(n)}) \}_{n=1:N}$). که هر ذره $s_t^{(n)}$ یک موقعیت فرضی شئ را نشان می‌دهد و بوسیله یک احتمال نمونه برداری گسسته وزن دهی شده‌اند ($\Pi_t^{(n)}= p(z_t|x_t=s_t^{(n)})$). مجموعه ذره‌ها توسط مدل دینامیکی سیستم در زمان منتشر می‌شوند. نهایتا موقعیت در هر لحظه توسط این ذرات و وزن‌ها طبق رابطه (۳-۷) تخمین زده می‌شود [۳۰].

$$x_t = \sum_{n=1}^{N} \Pi_t^{(n)} \times s_t^{(n)} \qquad (۳-۷)$$

۳-۵ واحد پردازنده گرافیک

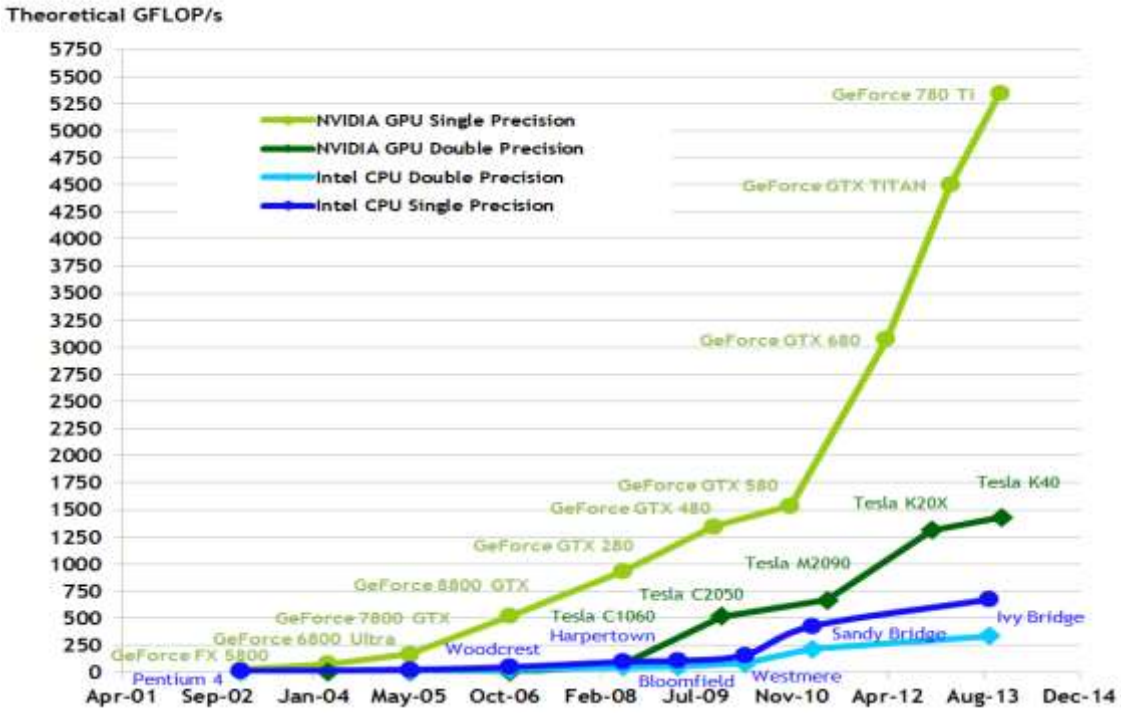
پردازنده‌های گرافیکی معمولا بر روی کارت گرافیک و یا مستقیما بر روی بُرد اصلی[1] قرار می‌گیرند؛ و توان پردازشی بسیار بالایی را نسبت به پردازنده‌های مرکزی ارائه می‌دهند، این موضوع موجب گسترش کاربردهای این پردازنده‌ها در حوزه‌هایی فراتر از بازی‌های کامپیوتری گشته است.

پردازنده‌های گرافیکی به دلیل معماری ویژه و امکان موازی سازی که دارند، علاوه بر انجام عملیات گرافیکی، قدرت انجام محاسبات و پردازش‌های عمومی را در انجام محاسبات پیچیده در کاربردهای خاص نیز دارند. جدیدترین کاربردهای عمومی پردازنده‌های گرافیکی در مراجع [۳۳،۳۴] قابل ملاحظه است. بهره برداری از این توان پردازشی برای محاسبات عمومی، منوط به شناخت مشخصات

[1] Main board

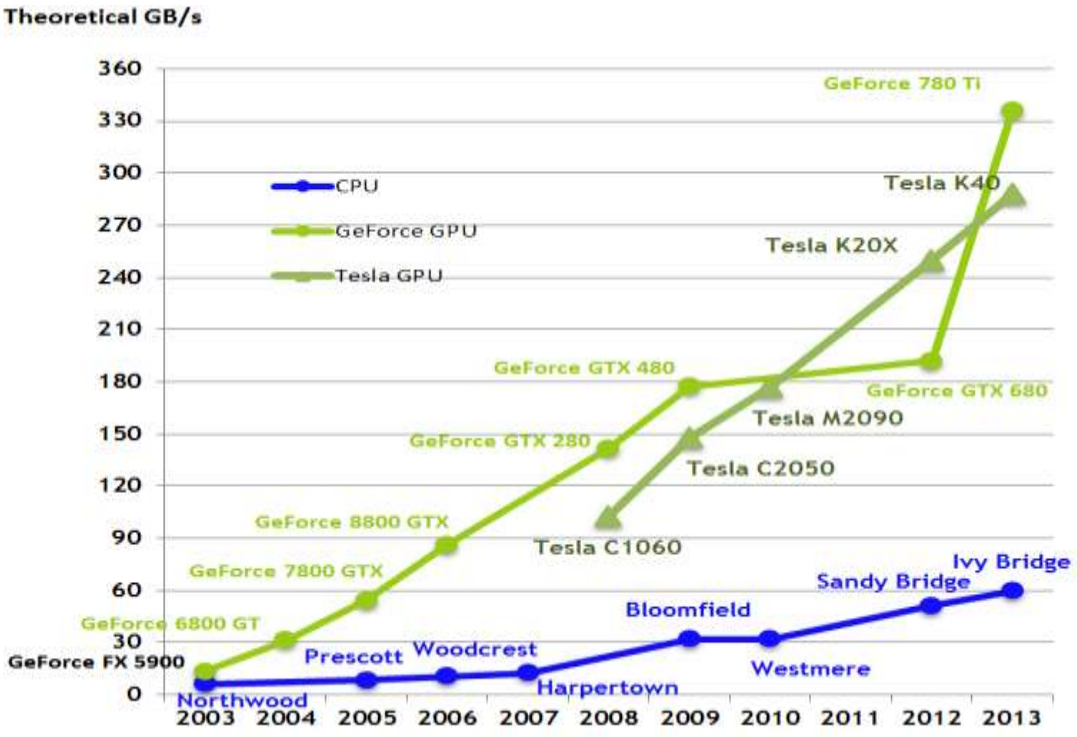
بنیادین سخت افزاری و نرم افزاری این پردازنده‌ها و ناتوانایی‌های ذاتی برخاسته از این طراحی منحصر بفرد است. به علاوه لازم است کارکردهای مناسب، در کنار نقاط ضعف و قوت آنها در مقایسه با پردازنده‌های سنتی نیز مدنظر قرار داده شود.

پردازنده‌های مرکزی، چند هسته‌ای هستند؛ در حالیکه پردازنده‌های گرافیکی از هسته‌های متعددی تشکیل شده که بعضا تعداد این هسته‌ها به بیش از دو هزار عدد هم می‌رسد [۳۵]. میان ممیز شناور در پردازنده مرکزی و پردازنده گرافیکی نیز تفاوت‌های قابل توجهی وجود دارد. شکل‌های (۳-۴) و (۳-۵) نمایانگر مقایسه‌ی رشد عملیات ممیز شناور بر ثانیه[1] (FLOPS) و پهنای باند میان پردازنده-های مرکزی و گرافیکی می‌باشند.



شکل ۳-۴: مقایسه عملکرد محاسباتی بر اساس رشد عملیات ممیز شناور [۳۶]

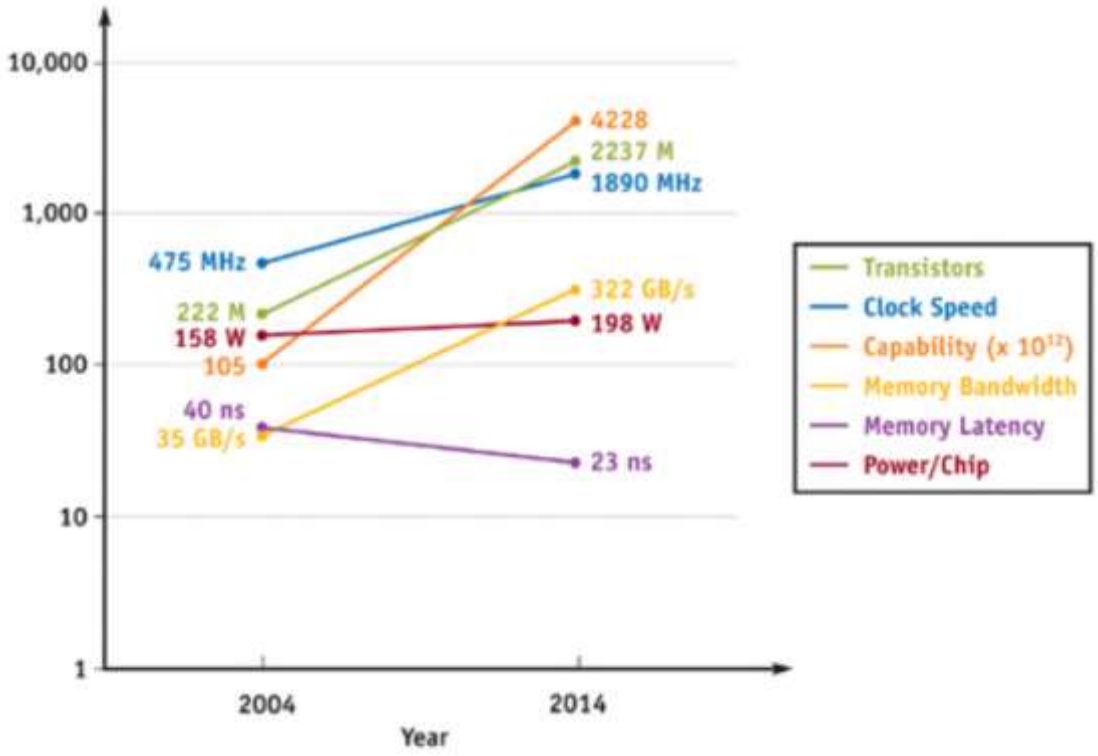
[1] Floating point operations per second



شکل ۳-۵: مقایسه عملکردی بر اساس پهنای باند [۳۶]

در سال‌های اخیر رقابت بر سر قدرت محاسبه، علاوه بر محدودیت گنجاندن هسته‌های بیشتر یک مانع دیگر نیز دارد: مصرف انرژی. در این زمینه به نظر می‌رسد پردازنده‌های مرکزی به سقف کارایی خود رسیده باشند [۳۷] درحالی که پردازنده‌های گرافیکی پیشرفت قابل توجهی را در زمینه‌ی سرعت ساعت[1]، پهنای باند حافظه، کارایی و کاهش مصرف انرژی [۳۷] داشته‌اند. شکل (۳-۶) شاهدی بر این مدعی است.

[1] Clock speed



شکل ۳-۶: تغییرات مشخصه‌های کلیدی پردازنده‌های گرافیکی [۳۷]

با توجه به مسائل مطرح شده می‌توان مهم‌ترین علل محبوبیت پردازنده‌های گرافیکی و استفاده از آنها را در پژوهش‌های اخیر، چنین ذکر کرد: قدرت پردازشی عظیم، قیمت مناسب، قابلیت برنامه نویسی انعطاف پذیر و از همه مهمتر، در دسترس بودن و توسعه‌ی روز افزون این پردازنده‌ها از حیث معماری و امکانات نرم‌افزاری. این فاکتورها، پردازنده‌های گرافیکی را برای طیف وسیعی از برنامه‌های کاربردی، با محاسبات عددی، مناسب می‌سازد [۳۸].

۳-۵-۱ عملکرد پردازنده‌های گرافیکی

پردازنده‌های گرافیکی در حقیقت جدا از پردازنده‌های مرکزی و به عنوان رقیبی برای پردازنده‌های مرکزی بشمار نمی‌روند. بلکه این پردازنده‌ها با همکاری پردازنده مرکزی به تسریع عملیات کمک می-

کنند. روش کار بدین صورت است که قسمتی از دستورات، که به آن هسته[1] گفته می‌شود، از سمت پردازنده مرکزی (میزبان[2]) به سمت پردازنده گرافیکی ارسال می‌شود و برخی از محاسبات در پردازنده گرافیکی انجام می‌شود و نتایج به سمت پردازنده مرکزی ارسال می‌شود. در حقیقت مدیریت کارها در سیستم توسط پردازنده مرکزی انجام می‌شود.

در رایانه‌ها، پردازنده مرکزی و پردازنده گرافیکی هریک فضای حافظه‌ی مجزایی دارند. هنگامی که داده‌ها از سمت پردازنده مرکزی به سمت پردازنده گرافیکی ارسال می‌شود، در طی این فرآیند ارسال و دریافت نیز، زمانی علاوه بر زمان صرف شده برای محاسبات تلف می‌شود. در حقیقت پردازنده مرکزی اطلاعات را به حافظه‌ای از پردازنده گرافیکی به نام حافظه سراسری[3] ارسال می‌کند. علاوه بر آن یک سری از حافظه‌های دیگر روی کارت گرافیک به نام‌های حافظه محلی[4] و حافظه اختصاصی[5] نیز وجود دارد. تفاوت این حافظه‌ها به سبب سرعت دسترسی و حجم اطلاعاتی که در خود ذخیره می‌کنند، بوده و برای انجام محاسبات با حجم داده‌های مختلف، در پردازنده گرافیکی کاربرد دارند.

۳-۵-۲ زبان‌های برنامه نویسی پردازنده گرافیکی

زبان‌های برنامه نویسی استفاده شده در این مبحث را می‌توان به دو گروه تقسیم کرد. زبان‌های برنامه نویسی در قسمت میزبان که در واقع بر روی پردازنده مرکزی قابل استفاده هستند؛ و زبان‌های برنامه نویسی در قسمت دستگاه که بر روی پردازنده‌های گرافیکی قابل استفاده هستند. زبان‌های برنامه نویسی در سمت پردازنده مرکزی، زبان‌های C و C++ هستند و متداول‌ترین زبان‌های استفاده شده در سمت پردازنده گرافیکی عبارتند از:

---

[1] Kernel
[2] Host
[3] Global memory
[4] Local memory
[5] Private memory

- **OpenGL**[1]**:** این زبان شامل مجموعه‌ای از توابع کتابخانه‌ای است که امکان برنامه نویسی بر روی پردازنده گرافیکی را بصورت سری وبرای کارهای گرافیکی فراهم می‌کند [۳۹].

- **OpenCL**[2]**:** این زبان شبیه زبان برنامه نویسی C است و امکان نوشتن برنامه‌ها را بطور موازی بر روی پردازنده گرافیکی فراهم می‌کند [۳۹].

- **CUDA**[3]**:** این زبان شبیه زبان برنامه نویسی C است و توسط شرکت انویدیا[4] تنها برای کار بر روی پردازنده‌های گرافیکی ساخته شده توسط این شرکت ارائه شده است. این زبان نیز همانند OpenCL برای نوشتن برنامه‌ها بطور موازی بر روی پردازنده گرافیکی کاربرد دارد [۳۹].

۳-۵-۳ معماری پردازنده‌های گرافیکی

تعداد هسته‌های واحد پردازنده گرافیکی بسته به سازنده‌ی آن متفاوت است. انویدیا و ای‌ام‌دی[5] دو تولید کننده‌ی بزرگ تراشه‌ی گرافیکی برای کامپیوترها هستند و دو سبک متفاوت در طراحی واحد پردازش گرافیکی انتخاب کرده‌اند. انویدیا سعی می‌کند توان بیشتری را درهسته‌های کمتری متمرکز کند در حالیکه ای‌ام‌دی برای افزایش قدرت پردازش سعی کرده از هسته‌های بیشتر با توان کمتراستفاده کند. یک کارت گرافیک معمولی انویدیا دارای شصت و هشت هسته است در حالی که یک کارت گرافیک معمولی ای‌ام‌دی حدود هزار و پانصد هسته دارد اما در عین حال قدرت پردازش این دو تراشه‌ی گرافیکی تقریباً مشابه است. شکل (۳-۷) تفاوت ساختاری واحد پردازنده مرکزی و واحد پردازنده گرافیکی را به نمایش می‌گذارد.

---

[1] Open graphic language

[2] Open computing language

[3] Compute Unified Device Architecture

[4] NVIDIA

[5] AMD

شکل ۳-۷: ساختار واحد پردازنده گرافیک در مقابل واحد پردازنده مرکزی [۴۰]

همانطور که مشخص است واحد پردازنده گرافیک دارای تعداد زیادی واحد محاسبات منطقی[1] است. این درحالی است که یک پردازنده مرکزی مدرن نوعا دارای چهار هسته‌ی واقعی است. بطور کلی می-توان گفت که پردازنده‌های مرکزی برای انجام محاسبات پیچیده و پردازش‌های متنوع مناسب هستند. در حالی که واحد پردازنده گرافیکی برای پردازش و محاسبات ساده به تعداد زیاد و بصورت موازی بهینه شده است [۴۰].

۳-۶ جمع بندی

در این فصل به مباحثی که در طی فصول آینده به آنها نیاز داریم، پرداختیم. مبانی نظری برخی از روش‌های استفاده شده در طی فرآیند ردیابی اشیاء را بیان کردیم؛ توضیحاتی راجع به پردازنده‌های گرافیکی، عملکرد، معماری و زبان‌های متداول برنامه نویسی آنها را ارائه دادیم. در فصل بعد روش ارائه شده در این پایان نامه را برای ردیابی بلادرنگ اشیاء بیان خواهیم کرد.

[1] ALU (arithmetic logic unit)

فصل چهارم: روش پیشنهادی

۴-۱ مقدمه

هدف از انجام این پایان نامه تسریع بخشیدن عملیات ردیابی به وسیله پردازش موازی با بهره‌گیری از واحد پردازنده گرافیکی است. با توجه به ساختار واحد پردازنده گرافیکی، که از هسته‌های متعددی تشکیل شده و بعضا تعداد این هسته‌ها به بیش از دو هزار عدد هم می‌رسد [۳۵] قصد داریم که با بکارگیری این ویژگی، در قالب پردازش موازی با پیاده سازی الگوریتم‌های مطرح شده، بر روی بستر کارت گرافیکی به ردیابی بلادرنگ اشیاء در دنباله‌ای از تصاویر ویدئویی بپردازیم.

در این فصل، به منظور ردیابی بلادرنگ اشیاء، فرآیند پیشنهادی و نحوه استفاده از واحد پردازنده گرافیکی به منظور موازی سازی هر یک از روش‌های ذکر شده را بیان خواهیم کرد.

۴-۲ شرح مسئله

بسیاری از برنامه‌های کاربردی در زمینه ردیابی اشیاء به دلیل طبعیت مسئله، نیازمند پردازش‌های بلادرنگ هستند. بلادرنگ بودن پردازش در مسائلی مانند مسائل نظامی و سیستم‌های نظارتی و امنیتی اهمیت ویژه‌ای پیدا می‌کند. این در حالی است که با پیشرفت تکنولوژی و متعاقب آن ابزار فیلم برداری، با افزایش ابعاد تصاویر و درجه تفکیک[1] ویدئوها مواجه هستیم. همچنین افزایش انتظارات کاربران از کارایی این سیستم‌ها نیز باعث شده در سال‌های اخیر شاهد استفاده از الگوریتم-های پیچیده‌تری باشیم. این درحالی است که مسائل مطرح شده، دستیابی به پردازش بلادرنگ را دشوارتر می‌کند. با توجه به مسائل مطرح شده نیاز به یک الگوریتم ردیابی بلادرنگ اشیاء احساس می‌شود، در حالی که از دقت و پایداری کافی نیز برخوردار باشد.

[1] Resolution

۳-۴ روش پیشنهادی

همانطورکه بیان شد یک فرآیند ردیابی برای برخورداری از پایداری و دقت کافی نیازمند طی کردن چندین مرحله است. این مراحل به ترتیب عبارتند از:

- تشخیص اشیاء متحرک و جداسازی پیش‌زمینه و پس‌زمینه
- دسته‌بندی و استخراج ویژگی از اشیاء
- ارائه و ردیابی شئ مورد نظر

در این پژوهش قصد داریم با توجه به ویدئوهای مورد استفاده که توسط دوربین ثابت تهیه شده‌اند، بوسیله روش‌های تفاضل قاب و تفاضل قاب سه‌تایی اشیاءِ متحرک تصویر را تشخیص دهیم؛ پس از آن با استفاده از ویژگی رنگی اشیاء و استفاده از فیلتر ذره‌ای، روند ردیابی را طی نماییم. دلیل اصلی انتخاب الگوریتم‌های ذکر شده، علاوه بر ویژگی ویدئوها و اشیاء (که به تفصیل در فصل دوم مورد بررسی قرار گرفت) را، می‌توان ماهیت مناسب موازی سازی آنها ذکر کرد.

کارت‌های گرافیکی امروزی، کاربردهایی فراتر از کنترل آن چیزی که برروی نمایشگر نقش می‌بندد، دارند. کاربردهایی نظیر محاسبات در کنار پردازنده‌های مرکزی و در نتیجه کاهش بار محاسباتی پردازنده‌های مرکزی از مهمترین کاربردهای کارت‌های گرافیکی امروزی است [۴۱].

بعضی مواقع که نیاز به پردازش‌های گرافیکی سنگین نداریم، واحد پردازنده گرافیک عملا کار زیادی برای انجام ندارد. این در حالی است که، ممکن است پردازنده اصلی زیر بار پردازشی فراوانی قرار گرفته باشد. بنابراین عقلانی است در صورت امکان مقداری از اطلاعات و پردازش آنها را به صورت هم زمان و موازی با واحد پردازنده مرکزی، به واحد پردازنده گرافیکی منتقل کنیم. اما انتقال این اطلاعات و هماهنگ سازی واحد پردازنده گرافیکی با واحد پردازنده مرکزی امری مهم است. این هماهنگ

سازی و انتقال اطلاعات به لطف وجود سکوهای برنامه‌نویسی[1] CUDA و OpenCL و ... برای ما فراهم می‌شود.

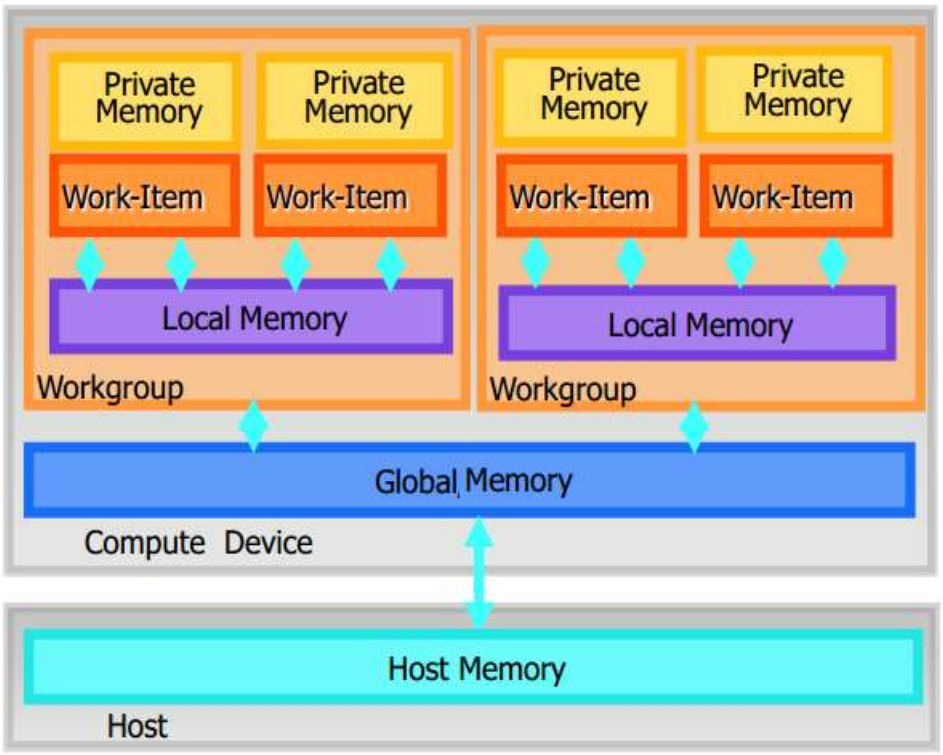
کتابخانه‌ی OpenCL در ابتدا توسط شرکت اَپِل[2] معرفی شد و بعدها توسط شرکت‌های ای‌ام‌دی، آی-بی‌ام[3] و انویدیا گسترش یافت. در حالی که CUDA منحصرا توسط شرکت انویدیا و مخصوص کارت-های گرافیکی این شرک توسعه یافته است. بنابراین مزیت استفاده از OpenCL در مقابل CUDA را می‌توان، قابل استفاده بودن آن برای تمامی کارت‌های گرافیکی برشمرد.

در ادامه و به منظور کاهش زمان پردازشِ روش پیشنهادی، از موازی سازی الگوریتم‌های ذکر شده، به کمک OpenCL بر روی واحد پردازنده گرافیکی بهره می‌بریم. انجام یک عملیات بر روی چندین داده بصورت موازی، و انجام چندین عملیات بصورت موازی بر روی مجموعه‌ای از داده‌ها، دو شیوه‌ای است که با استفاده از OpenCL برای پردازش موازی می‌توان استفاده نمود [۳۶].

همانطور که در فصل سوم بیان شد، قسمتی از پردازش که نیاز است بوسیله پردازنده گرافیکی روی داده‌ها انجام شود، از سمت پردازنده مرکزی به سمت پردازنده گرافیکی ارسال شده، محاسبات در پردازنده گرافیکی صورت می‌گیرد و نتایج به سمت پردازنده مرکزی ارسال می‌شود. این دستورات تحت عنوان کرنل[4] به پردازنده گرافیکی ارسال می‌شوند. بطور کلی کرنل را می‌توان معادل با تابع در زبان‌های برنامه نویسی دانست؛ با این تفاوت که کرنل فقط قابل اجرا توسط هسته‌های پردازنده گرافیکی است.

[1] Software platforms
[2] Apple company (a multinational technology company)
[3] IBM company (International Business Machines Corporation)
[4] Kernel

داده‌های مورد نیاز برای پردازش بوسیله واحد پردازنده گرافیک، بر روی حافظه‌های تعبیه شده برای پردازنده کارت گرافیک ذخیره می‌شوند. برای درک بهتر، شکل (۴-۱) مدل حافظه در OpenCL را نشان می‌دهد.

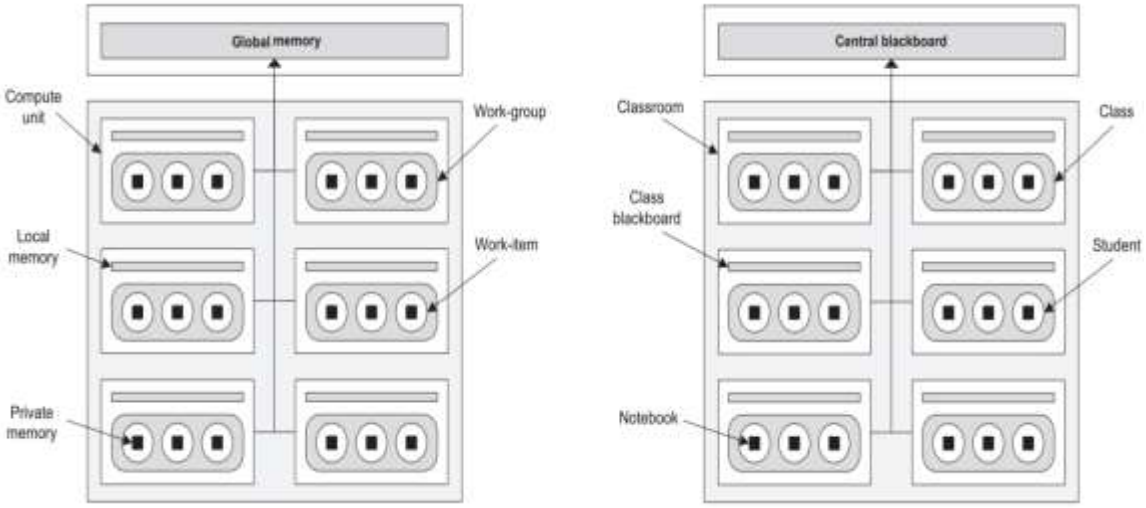


شکل ۴-۱: مدل حافظه در معماری OpenCL [۳۶]

در مدل نشان داده شده در شکل (۴-۱) تبادل داده‌ها بین پردازنده گرافیکی و پردازنده مرکزی از طریق حافظه مرتبط با میزبان (پردازنده مرکزی) و حافظه‌ی سراسریِ پردازنده‌ی کارت گرافیک صورت می‌پذیرد. در پردازنده گرافیکی حافظه‌های دیگری نیز تعبیه شده است که تفاوت این حافظه‌ها به سبب حجم حافظه‌ها، سرعت و سطح دسترسی پردازنده‌های گرافیک به آن‌ها است.

برای توضیح معماری حافظه‌ی کارت گرافیک توسط OpenCL می‌توان آن را با یک مدرسه مقایسه کرد. مدرسه‌ای را فرض کنید که، یک تابلو اعلانات داریم که تمامی دانش آموزان به آن دسترسی

دارند و حجم داده‌های زیادی را می‌تواند در خود جای دهد. برای هر کلاس یک تخته داریم که دانش آموزانِ آن کلاس فقط به آن تخته دسترسی دارند و هر دانش آموزِ این مدرسه دفترچه‌ای مخصوص به خود دارد. هر کلاس درس مانند یک واحد محاسبه[۱] بوده و درست مانند کلاس که می‌تواند به وسیله دانش آموزان اشغال شود، هر واحد محاسبه نیز می‌تواند بوسیله گروه کاری[۲] اشغال شود. منظور از اِلمانِ کاری[۳] کوچکترین واحد داده‌ای است که باید پردازش شود؛ و زمانی که این اِلمان‌های کاری کنار یگدیگر قرار گیرند، تشکیل یک گروه کاری را می‌دهند. شکل (۴-۲) این موضوع را به صورت گرافیکی شرح می‌دهد [۴۲].

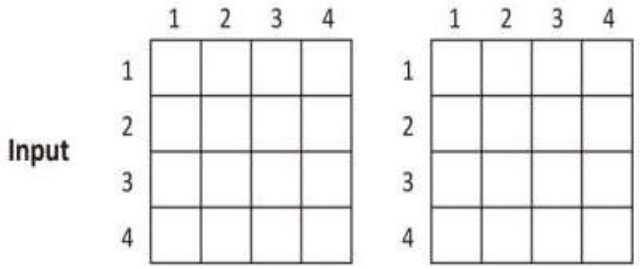


شکل ۴-۲: معماری کارت گرافیک توسط OpenCL در مقایسه با یک مدرسه [۴۲].

به عنوان نمونه و درک بهتر از نوشتنِ یک کرنل، فرض کنید مطابق شکل (۴-۳) دو ماتریس چهار در چهار، با داده‌های از نوع شناور[۴] داریم و ماتریس خروجی مطابق شکل (۴-۳-ب) از این دو ماتریس ساخته می‌شود. در برنامه نویسی متداول با نوشتنِ تعدادی حلقه و تعدادی دستور شرطی باید

[1] Compute unit
[2] Work group
[3] Work item
[4] Float values

خروجی را محاسبه کنیم. کاملا واضح است که این نوع از برنامه نویسی برای پردازش چنین مسائلی بهینه نیست؛ چراکه عناصر ماتریس خروجی به یکدیگر وابستگی ندارند و نیازی نیست به صورت سری و یکی پس از دیگری محاسبه شوند.



(الف)

Output

	1	2	3	4
1	A(1,1) + B(1,1)	A(1,2) - B(1,2)	A(1,3) × B(1,3)	A(1,4) ÷ B(1,4)
2	A(2,1) + B(2,1)	A(2,2) - B(2,2)	A(2,3) × B(2,3)	A(2,4) ÷ B(2,4)
3	A(3,1) + B(3,1)	A(3,2) - B(3,2)	A(3,3) × B(3,3)	A(3,4) ÷ B(3,4)
4	A(4,1) + B(4,1)	A(4,2) - B(4,2)	A(4,3) × B(4,3)	A(4,4) ÷ B(4,4)

(ب)

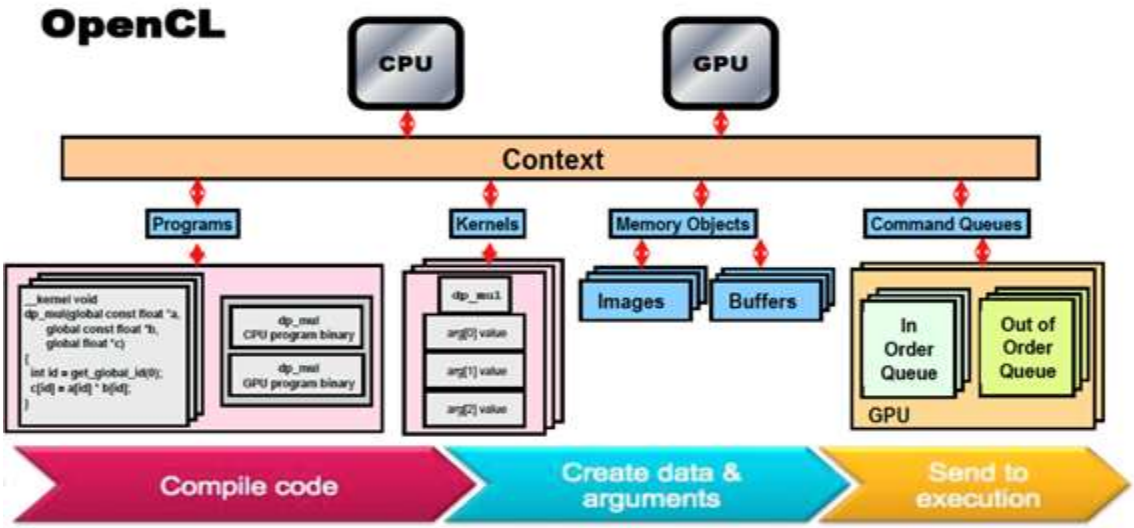
شکل ۴-۳: نمونه‌ای از یک مسئله‌ی محاسباتی مناسبِ موازی سازی

به منظور بدست آوردن ماتریس خروجی مطرح شده در شکل (۴-۳) و استفاده از پردازش موازی، کرنلی که برای این محاسبات می‌توان نوشت در شکل (۴-۴) نشان داده شده است. در این کرنل هر عنصر از ماتریس‌ها به عنوان یک المان کاری فرض می‌شود. در مرحله بعد باید تعداد این اِلمان‌های کاری مشخص شود؛ تعداد اِلمان‌های کاری در سمت میزبان مشخص می‌شود و با استفاده از دستور get_global_id در سمت پردازنده گرافیک دریافت می‌شود. در این مثال تعداد المان‌های کاری مقدار چهار فرض شده است. بنابراین متغیر base در کرنل، اعداد صفر، یک، دو و سه را اختیار می‌کند و ماتریس خروجی طبق دستورات نوشته شده ایجاد می‌گردد.

```
__kernel void dataParallel(__global float* A, __global float* B, __global float* C)
{
int base = 4*get_global_id(0);
C[base+0] = A[base+0] + B[base+0];
C[base+1] = A[base+1] - B[base+1];
C[base+2] = A[base+2] * B[base+2];
C[base+3] = A[base+3] / B[base+3];
}
```

شکل ۴-۴: نمونه کرنل نوشته شده برای حل مسئله شکل (۴-۳) با پردازش موازی توسط OpenCL

در حالت کلی در معماری OpenCL پس از پردازش داده‌ها توسط واحد پردازنده گرافیک، نهایتا داده-های پردازش شده برای واحد پردازنده مرکزی ارسال می‌شوند و روند اصلی برنامه طی خواهد شد. در حقیقت مدیریت کارها در سیستم توسط پردازنده مرکزی انجام می‌شود و تنها بخشی از محاسبات بر روی واحد پردازنده گرافیکی انجام می‌گردد. به طور کلی روند اجرای یک برنامه با همکاری پردازنده مرکزی و پردازنده گرافیکی و ایجاد ارتباط و هماهنگی بین آنها توسط OpenCL، در شکل (۴-۵) قابل مشاهده است.



شکل ۴-۵: نحوه اجرای برنامه موازی سازی شده توسط OpenCL [۳۶].

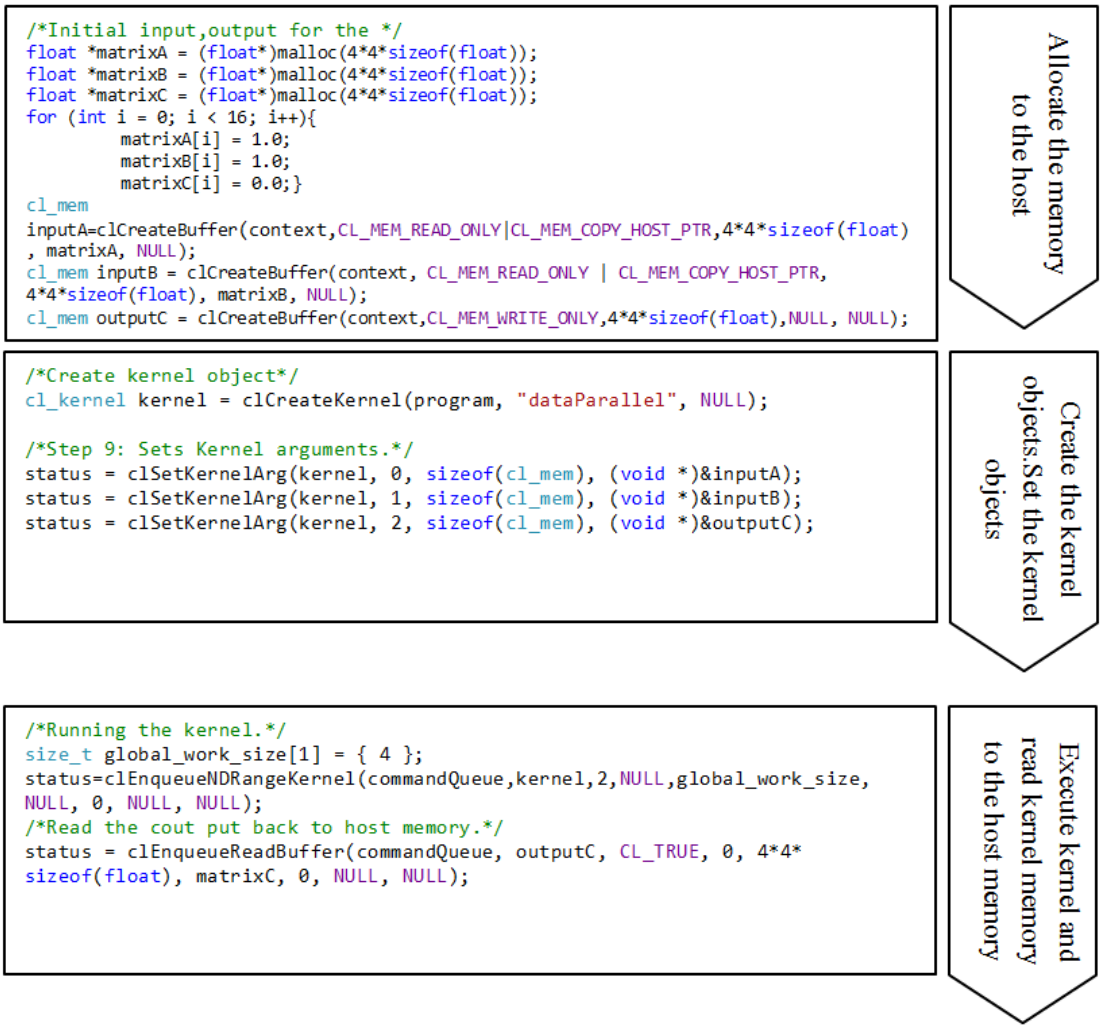
بنابراین برای پردازش موازی لازم است در ابتدا، دستگاه یا دستگاه‌هایی که در طی روند پردازشی در کنارِ واحد پردازنده مرکزی، باید کار کنند مشخص شود، پس از آن دستگاهی که نیاز است به کمک پردازنده مرکزی بیاید انتخاب می‌شود. در مرحله بعد مجموعه عملیات لازم برای بکارگیری دستگاه در کنار پردازنده مرکزی بصورت صفی از دستورات مرتب می‌شوند؛ این دستورات می‌توانند شامل دریافت و یا ارسال داده‌ها، اجرای توابع عملیاتی و مجموعه دستورات هماهنگ ساز باشند. در مرحله بعد پس از اختصاص حافظه‌های مناسب برای انجام عملیات پردازشی و مشخص کردن توابع و ورودی و خروجی‌های آن، دستورات مورد نظر بر روی دستگاه بصورت موازی با پردازنده مرکزی انجام می‌شوند و نهایتا داده‌های پردازش شده برای واحد پردازنده مرکزی ارسال می‌شوند. به منظور واضح‌تر شدن موضوع، برنامه نوشته شده در سمت میزبان، برای حلِ مسئله‌ی مطرح شده در شکل (۴-۳) و استفاده از کرنل مطرح شده، در شکل (۴-۶) قابل مشاهده است.

Getting available platforms and select the desired device

```
/*Getting platforms and choose an available one.*/
cl_uint numPlatforms;
cl_platform_id platform = NULL;
cl_int   status = clGetPlatformIDs(0, NULL, &numPlatforms);
cl_platform_id* platforms = (cl_platform_id*)malloc(numPlatforms*
sizeof(cl_platform_id));
status = clGetPlatformIDs(numPlatforms, platforms, NULL);
platform = platforms[0];
free(platforms);
/*choose the GPU device.*/
cl_uint numDevices = 0;
cl_device_id *devices;
status = clGetDeviceIDs(platform, CL_DEVICE_TYPE_GPU, 0, NULL, &numDevices);
devices = (cl_device_id*)malloc(numDevices * sizeof(cl_device_id));
status =clGetDeviceIDs(platform,CL_DEVICE_TYPE_GPU,numDevices,devices,NULL);
```

Create context and command queue.
Build the program

```
/*Create context and command queue associate with the context.*/
cl_context context = clCreateContext(NULL, 1, devices, NULL, NULL, NULL);
cl_command_queue commandQueue=clCreateCommandQueue(context,devices[0],0,NULL)
/*Create program object and build program */
const char *filename = "MatrixOperations.cl";
string sourceStr;
status = convertToString(filename, sourceStr);
const char *source = sourceStr.c_str();
size_t sourceSize[] = { strlen(source) };
cl_program program=clCreateProgramWithSource(context,1,&source,sourceSize,NULL);
status = clBuildProgram(program, 1, devices, NULL, NULL, NULL);
```



شکل ۴-۶: مثالی از مراحل لازم برای بکارگیری واحد پردازنده گرافیک در کنار پردازنده مرکزی در سمت میزبان

۴-۳-۱ تفاضل قاب

همانطورکه در فصل سوم، مبانی نظری این روش را بیان کردیم، خروجی این روش قاب‌های دنباله تصاویر را به دو ناحیه ثابت (اغلب پس زمینه) و متحرک (اغلب پیش زمینه) تفکیک می‌کند. تفاضل قاب در حقیقت تفاضل نقطه به نقطه بین دو قاب متوالی است. مقدار قاب تفاضل شده در یک نقطه بیانگر تغییر در آن نقطه می‌باشد. این تغییر می‌تواند، ناشی از حرک شئ باشد و یا تغییری غیر از حرک شئ، که برای ما حکم نویز را دارد. برای تمییز این دو حالت از یک حد آستانه استفاده می‌شود.

با توجه به اینکه باید پیکسل‌های متناظر دو قاب متوالی نظیر به نظیر از یکدیگر کم شوند، می‌توان گفت موازی سازی این فرآیند توسط پردازنده گرافیکی قابل انجام است. بنابراین برای پیاده‌سازی روش تفاضل قاب بر روی واحد پردازنده گرافیک نیاز داریم مقادیر پیکسل‌های دو قاب متوالی از دنباله تصاویر ویدئویی را به سمت پردازنده گرافیکی ارسال کنیم و طبق رابطه (۳-۱)، قاب تفاضل شده برگردانده شود.

با توجه به اینکه داده‌های ارسالی به پردازنده گرافیکی از نوع تصویر هستند، لذا برای افزایش سرعت دسترسی پردازنده گرافیکی به حافظه، می‌توان از حافظه محلی پردازنده گرافیکی به منظور ذخیره داده‌ها استفاده کرد. و نهایتا هر پیکسل را به عنوان یک المان کاری برای پردازش در نظر می‌گیریم. بدین ترتیب وظیفه تفاضل قاب‌های متوالی در روش تفاضل قاب را به واحد پردازنده گرافیکی محول می‌کنیم.

۴-۳-۲ تفاضل قاب سه‌تایی

همانطور که در مبانی نظری این روش بیان شد، برای کاهش نویز قاب تفاضل شده نسبت به روش تفاضل قاب متوالی، از سه قاب متوالی برای طی فرآیند استفاده می‌شود.

برای کاهش بار پردازشی واحد پردازنده مرکزی، وظیفه تفاضل قاب‌ها و ساخت قاب تفاضل شده‌ی نهایی، که طبق روابط (۳-۲) تا (۳-۴) بیان شده در فصل سوم ایجاد می‌شود را به واحد پردازنده گرافیکی واگذار می‌کنیم.

برای پیاده‌سازی بر روی پردازنده گرافیکی به روش بیان شده، نیاز است قاب‌های ویدئویی موردِ نیاز، به پردازنده گرافیکی منتقل شوند. بنابراین با توجه به اینکه داده‌های منتقل شده به پردازنده گرافیکی از نوع تصویر است، از حافظه محلی پردازنده گرافیکی برای ذخیره‌ی مقادیرِ پیکسل‌های قاب‌ها استفاده می‌کنیم و هر پیکسل را به عنوان یک اِلمانِ کاری در نظر می‌گیریم.

۴-۳-۳ فیلتر ذره‌ای

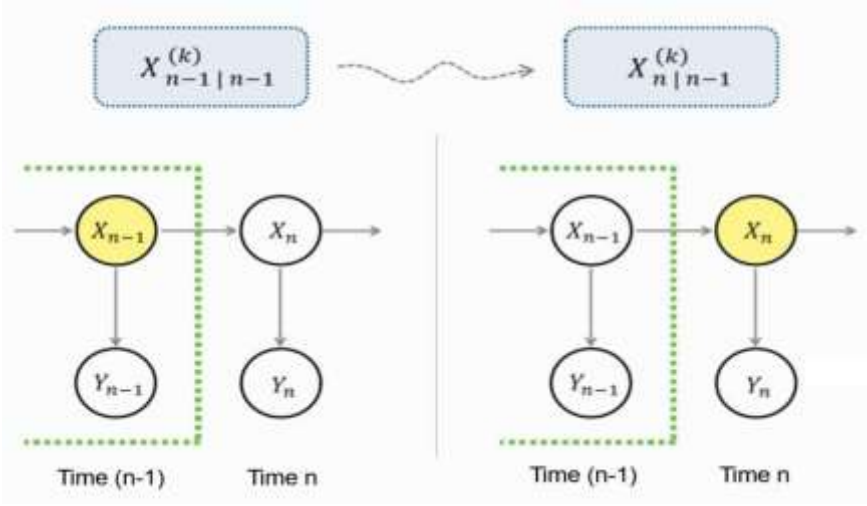
در ردیابی با استفاده از فیلتر ذره‌ای باید توجه داشت که هرچند صحت الگوریتم در حل مسئله ردیابی به نحو قابل قبولی اثبات شده است؛ اما یکی از بزرگترین مشکلات فیلتر ذره‌ای، عدم وجود اطلاعات برای تشخیص تعداد ذرات مورد نیاز برای دستیابی به دقت کافی در ردیابی شئ مورد نظر است [۴۳،۴۴].

با افزایش تعداد این ذرات به منظور افزایشِ دقتِ تخمینِ ردیابی، و مقاومت در برابر از دست رفتن هدف، این روش با افزایش زمان پردازش مواجه می‌شود [۴۳-۴۵]؛ که گاهی به قیمت از دست رفتن پردازش بلادرنگ تمام می‌شود. بنابراین یک مصالحه بین زمان پردازش و دقت ردیابی، تعیین کننده‌ی تعداد ذرات خواهد بود [۴۳].

بنابراین با افزایش سرعت زمان پردازش در این الگوریتم می‌توانیم از تعداد ذرات بیشتری برای ردیابی هدف استفاده کنیم و در نتیجه دقت ردیابی را افزایش دهیم. به منظور افزایش سرعت روند پردازشی این روش از واحد پردازنده گرافیکی در کنار واحد پردازنده مرکزی بهره خواهیم برد.

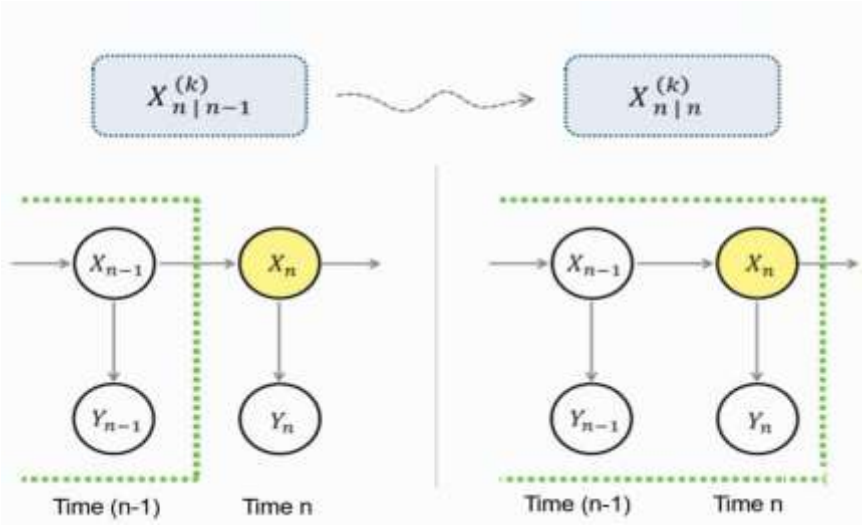
به منظور پیاده سازی فیلتر ذره‌ای برای ردیابی اشیاء به طور کلی نیاز به طی دو مرحله پیش‌بینی و بروزرسانی داریم. برای اجرای این مراحل نیاز است حالت سیستم و مشاهدات را نیز تعیین کنیم. بنابراین مختصات و سرعت هدف را به عنوان حالت و ویژگی رنگی پیکسل‌هایی که ذرات روی آن‌ها قرار گرفته‌اند را به عنوان مشاهده در نظر می‌گیریم.

منظور از مرحله پیش‌بینی، تخمین حالت سیستم در لحظه n بر اساس اطلاعات(حالت و مشاهدات) سیستم تا لحظه n-1 بر اساس مدل سیستم می‌باشد. این مسئله در شکل (۴-۷) به زبان ساده بیان شده است.



شکل ۴-۷: مرحله پیش‌بینی در فرآیند ردیابی اشیاء با استفاده از فیلتر ذره‌ای [۳۱]

منظور از مرحله بروزرسانی، تعیین وزن هر ذره بر اساس معیار شباهت[1] و نهایتا تعیین حالت سیستم در لحظه n براساس اطلاعات(حالت و مشاهدات) سیستم تا همان لحظه n می‌باشد. این مسئله در شکل (۴-۸) به زبان ساده بیان شده است.



شکل ۴-۸: مرحله بروزرسانی در فرآیند ردیابی اشیاء با استفاده از فیلتر ذره‌ای [۳۱]

[1] likelihood

بنابر مسائل مطرح شده با در نظر گرفتن هر ذره بـه عنـوان یـک اِلمـانِ کـاری و اسـتفاده از حافظـه سراسری واحد پردازنده گرافیک و روابط بیان شده در بخش (۳-۴)، مراحل پـیش‌بینـی و بروزرسـانی فیلتر ذره‌ای را بر روی بستر واحد پردازنده گرافیکی پیاده‌سازی می‌کنیم.

برای تعیین میزان شباهت در فیلتر ذره‌ای، از ویژگی رنگی اشیاء استفاده می‌کنیم؛ ویژگی رنگی مـی-تواند بسته به خصوصیات شئ مورد نظر و ویدئوی مورد استفاده شـامل پارمترهـای رنگـی در محـیط رنگی RGB یا HSV باشد و یا این ویژگیِ رنگی، هیستوگرامِ رنگیِ ناحیه‌ای از قاب باشد که مرکز این ناحیه منطبق بر مرکز ثقل جسم است و اندازه‌ی آن متناسب با اندازه جسم است. همچنین قابل ذکـر است که برای مرحله پیش‌بینی در فیلتر ذره‌ای، مدل خطی با سرعت ثابت برای شئ را به عنوان مدل دینامیکی سیستم استفاده کرده‌ایم.

۴-۴ جمع‌بندی

در این فصل یک روند جامع و متداول برای بسیاری از مسائل ردیابی را بیان کردیم. در این روش پیشنهاد کردیم که از روش‌های تفاضل قاب و یا تفاضل قاب سه‌تایی، برای تشخیص اشیاء متحرک از پس زمینه استفاده شود. پس از آن با توجه به ویژگی رنگی هدف، و استفاده از روش فیلتر ذره‌ای به ردیابی شئ مورد نظر پرداختیم.

قابل توجه است که زمان پردازش روش‌های تشخیص اشیاء متحرک عموما با ابعاد دنباله تصاویر ویدئویی رابطه دارد؛ و با افزایش ابعاد ویدئو زمان پردازش نیز افزایش می‌یابد. همچنین در ردیابی با استفاده از فیلتر ذره‌ای با افزایش تعداد ذرات به منظور افزایش دقتِ تخمینِ ردیابی، زمان پردازش افزایش می‌یابد که گاها منجر به از دست رفتن پردازش بلادرنگ می‌شود.

بنابراین به منظور کاهش زمان پردازش در کنار دستیابی به دقت کافی در فرآیند ردیابیِ پیشنهادی، روش‌هایِ ذکر شده برای طی فرآیند را، با بهره‌گیری از واحد پردازنده گرافیکی در کنار پردازنده مرکزی موازی سازی کردیم. در فصل بعد، نتایج حاصل از این پیاده سازی را بیان می‌کنیم و مقایسه زمانی پردازش را در حالت موازی شده و موازی نشده بیان خواهیم کرد.

فصل پنجم: نتایج تجربی

۵-۱ مقدمه

با توجه به اینکه هدف، ارزیابی و تاثیر موازی سازی با استفاده از واحد پردازنده گرافیکی، بر سرعت پردازش در فرآیند ردیابی اشیاء است، در ادامه آزمایشاتی به منظور سنجش عملکرد روش پیشنهاد شده را ارائه می‌کنیم و نتایج آن‌ها را مورد بررسی قرار خواهیم داد.

لازم به ذکر است، پیاده سازی‌های بیان شده با استفاده از نرم افزار Visual studio و زبان‌های C و C++ و بهره‌گیری از OpenCV 2.4.8 و OpenCL 1.1 انجام شده است؛ و نتایج بیان شده، حاصلِ انجامِ آزمایشات بر روی رایانه‌ای با مشخصات ذکر شده در جدول (۵-۱) می‌باشد.

جدول ۵-۱: مشخصات رایانه مورد استفاده

Processor	Intel® Core™ i7-2670QM
Graphic card	NVIDIA GeForce GT 540M
RAM	8.0 GB
System type	64 bit
Operating system	Windows 7 Home Premium

۵-۲ تشخیص اشیاء متحرک

در مرحله تشخیص اشیاء متحرک، از روش‌های تفاضل قاب و تفاضل قاب سه‌تایی استفاده کردیم. این روش‌ها از لحاظ زمان پردازش کاملا وابسته به ابعاد ویدئوی مورد نظر هستند. لذا به منظور نشان دادن این مسئله اقدام به تهیه یک ویدئو با ابعاد ۴۰۹۶×۲۱۶۰ پیکسل کردیم. و برای بررسی تاثیر ابعاد ویدئو بر سرعت پردازش، این ویدئو را به ویدئوهای با ابعاد کوچکتر تبدیل کردیم. چراکه فقط

تغییر ابعاد ویدئو در تغییر زمان پردازش تاثیر داشته باشد. به عنوان نمونه، چند قاب از این ویدئو در شکل (۵-۱) قابل مشاهده است.

شکل ۵-۱: چند قاب از ویدئو تهیه شده برای آزمایشات

نتایج زمانی حاصل از پردازش ویدئوهای تهیه شده، بدون موازی سازی و تنها با استفاده از واحد پردازنده مرکزی برای روش‌های تفاضل قاب و تفاضل قاب سه‌تایی در جدول (۵-۲) قابل مشاهده است. زمان‌ها بر حسب میلی‌ثانیه بوده و متوسط زمانِ پردازش هر قاب از ویدئو را بیان می‌کند.

جدول ۵-۲: زمان متوسط پردازش هر قاب توسط پردازنده مرکزی در روش‌های تشخیص اشیاء متحرک (میلی‌ثانیه)

تفاضل قاب سه‌تایی	تفاضل قاب	ابعاد ویدئو
۸۳/۴۰	۳۱/۱۴	۴۰۹۶×۲۱۶۰
۱۹/۱۲	۷/۳۵	۱۹۲۰×۱۰۸۰
۸/۱۹	۳/۰۴	۱۲۸۰×۷۲۰
۳/۸۵	۱/۳۰	۸۵۴×۴۸۰
۱/۴۰	۰/۵۴	۴۸۴×۲۷۲
۰/۳۰	۰/۱۱	۲۴۰×۱۳۶

نتایج بیان شده در جدول (۵-۲) نشان می‌دهد، در روش‌های تفاضل قاب و تفاضل قاب سه‌تایی، سرعت پردازش با ابعاد ویدئوی مورد استفاده رابطه عکس دارد؛ و این افزایش ابعاد ویدئو ممکن است باعث خارج شدن پردازش از حالت بلادرنگ شود. به عنوان مثال اگر در روش تفاضل قاب سه‌تایی در حالت غیر موازی، از ویدئویی با ابعاد ۴۰۹۶x۲۱۶۰ استفاده شده باشد، برای پردازش هر قاب نیازمند ۸۳ میلی‌ثانیه زمان هستیم و این بدین معنا است که که اگر نرخ قاب ویدئو بیش از ۱۲ باشد، پردازش از حالت بلادرنگ خارج خواهد شد.

نتایج زمانی حاصل از اجرای روش‌های تفاضل قاب و تفاضل قاب سه‌تایی موازی شده توسط واحد پردازنده گرافیک در جداول (۵-۳)، (۵-۴) و (۵-۵) بیان شده است. زمانی که بخواهیم پردازشی را توسط واحد پردازنده گرافیکی در کنار واحد پردازنده مرکزی انجام دهیم، علاوه بر صرف زمانی برای پردازش، زمانی هم صرف انتقال اطلاعات بین این دو پردازنده خواهد شد. جدول (۵-۳) بیانگر زمان پردازش اطلاعات توسط پردازنده گرافیکی است. جدول (۵-۴) زمانی که برای تبادل داده‌ها بین دو پردازنده صرف می‌شود را نشان می‌دهد. و نهایتا جدول (۵-۵) بیانگر زمان کلی پردازش هر قاب از ویدئو در حالت موازی سازی شده (شامل زمان پردازش، انتقال داده‌ها و ...) است.

جدول ۵-۳: زمان متوسط پردازش هر قاب توسط پردازنده گرافیکی در روش‌های تشخیص اشیاء متحرک (میلی‌ثانیه)

تفاضل قاب سه‌تایی	تفاضل قاب	ابعاد ویدئو
۰/۰۶	۰/۰۵	۴۰۹۶×۲۱۶۰
۰/۰۵	۰/۰۴	۱۹۲۰×۱۰۸۰
۰/۰۴	۰/۰۳	۱۲۸۰×۷۲۰
۰/۰۳	۰/۰۲	۸۵۴×۴۸۰
۰/۰۲	۰/۰۲	۴۸۴×۲۷۲
۰/۰۲	۰/۰۲	۲۴۰×۱۳۶

جدول ۵-۴: زمان متوسط انتقال داده در هر قاب بین پردازنده مرکزی و پردازنده گرافیکی (میلی‌ثانیه)

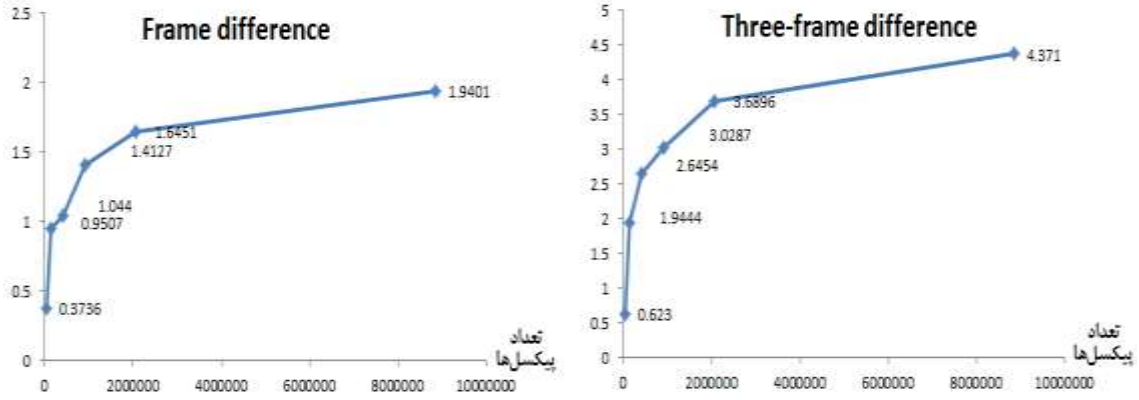
تفاضل قاب سه‌تایی	تفاضل قاب	ابعاد ویدئو
۱۸/۹۷	۱۵/۶۶	۴۰۹۶×۲۱۶۰
۵/۱۵	۴/۴۳	۱۹۲۰×۱۰۸۰
۲/۶۶	۲/۱۲	۱۲۸۰×۷۲۰
۱/۴۳	۱/۲۲	۸۵۴×۴۸۰
۰/۷۰	۰/۵۵	۴۸۴×۲۷۲
۰/۴۸	۰/۲۹	۲۴۰×۱۳۶

جدول ۵-۵: زمان متوسط کل پردازش حالت موازی هر قاب در روش‌های تشخیص اشیاء متحرک (میلی‌ثانیه)

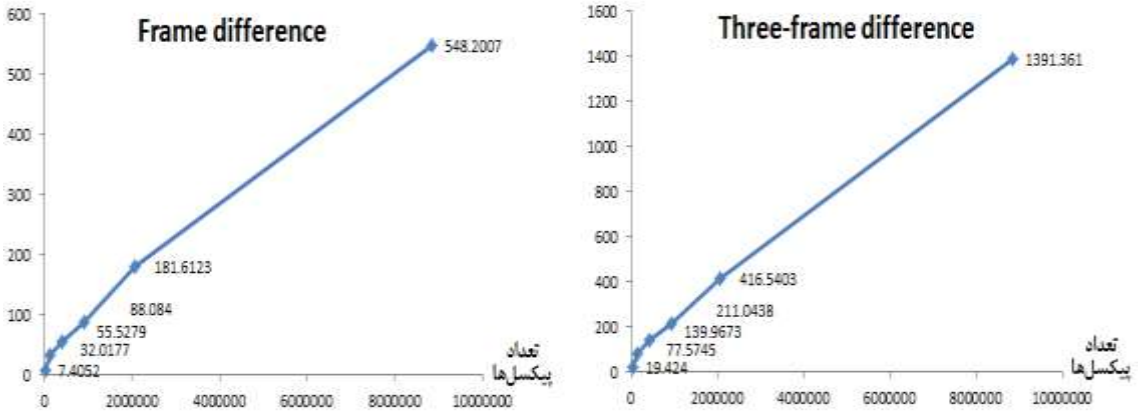
تفاضل قاب سه‌تایی	تفاضل قاب	ابعاد ویدئو
۱۹/۰۳	۱۶/۰۵	۴۰۹۶×۲۱۶۰
۵/۱۸	۴/۴۷	۱۹۲۰×۱۰۸۰
۲/۷۰	۲/۱۵	۱۲۸۰×۷۲۰
۱/۴۶	۱/۲۴	۸۵۴×۴۸۰
۰/۷۲	۰/۵۷	۴۸۴×۲۷۲
۰/۴۹	۰/۳۰	۲۴۰×۱۳۶

به منظور درک بهتر میزان تاثیر موازی سازی در روش‌های ذکر شده، بهبودِ زمانیِ پردازش در شکل-های (۵-۲) و (۵-۳) نشان داده شده است. اعداد بیان شده در شکل (۵-۲)، بیانگر نسبت زمانِ

فرآیند پردازشِ هر قاب در حالتِ غیر موازی به زمانِ پردازشِ هر قاب در حالتِ پردازشِ موازی است. اعداد بیان شده در شکل(۵-۳) ناشی از تقسیم زمان پردازش توسط پردازنده مرکزی بر زمان پردازش توسط پردازنده گرافیکی است و نمایانگر نسبت افزایش سرعت پردازش است.



شکل ۵-۲: نسبت زمان کل پردازش غیر موازی بر زمان کل پردازش موازی با توجه به ابعاد ویدئو



شکل ۵-۳: نسبت زمان پردازش پردازنده مرکزی بر زمان پردازش پردازنده گرافیکی با توجه به ابعاد ویدئو

۵-۳ ردیابی اشیاء

بررسی معیارهای موثر در دقت ردیابی با استفاده از فیلتر ذره‌ای نشان می‌دهد، در فیلتر ذره‌ای با پارامترهای تنظیمِ مناسب و یکسان، افزایش تعداد ذرات باعث افزایش دقتِ تخمینِ مکان هدف و بهبود کارایی ردیاب می‌شود؛ این در حالی است که با افزایش تعداد این ذرات زمان محاسبات افزایش می‌یابد [۴۵-۴۳].

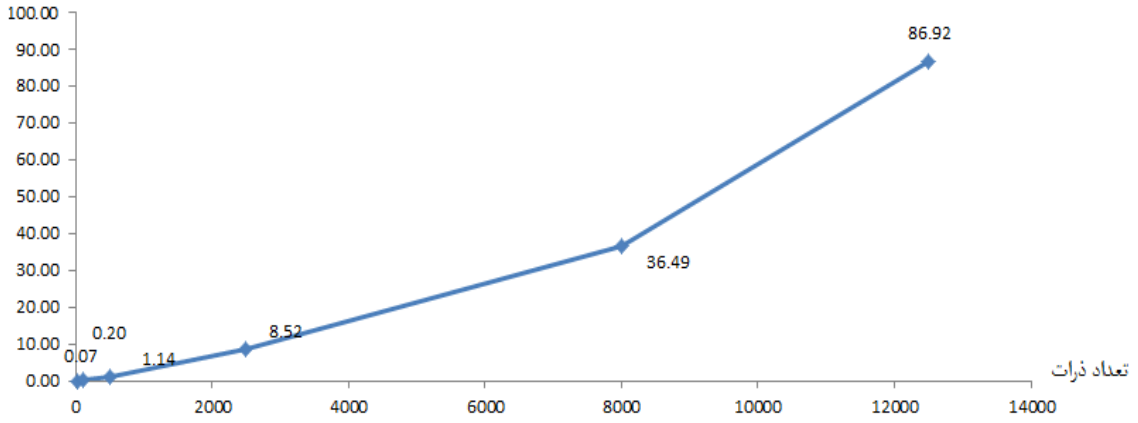
به منظور بررسی موضوع مطرح شده، در یکی از ویدئوهای مورد آزمایش با تغییر تعداد ذرات، زمانِ پردازشِ هر قاب را برای حالت موازی شده و موازی نشده‌ی فیلتر ذره‌ای بررسی می‌کنیم.

جدول (۵-۶) بیان کننده‌ی نتایجِ زمانی بر اساس تعداد ذرات در فرآیند فیلتر ذره‌ای می‌باشد. در این جدول، زمان متوسط پردازش هر قاب: توسط پردازنده مرکزی (در حالت موازی سازی نشده)، زمانی که پردازنده گرافیکی پردازش را انجام می‌دهد (در حالت موازی سازی شده)، زمان انتقال داده‌ها بین دو پردازنده (در حالت موازی سازی شده) و زمان کل پردازش در حالت موازی شده (شامل زمان انتقال داده، پردازش پردازنده گرافیکی، تعیین پارامترهای کرنل‌ها، اختصاص حافظه و ...) بیان شده است.

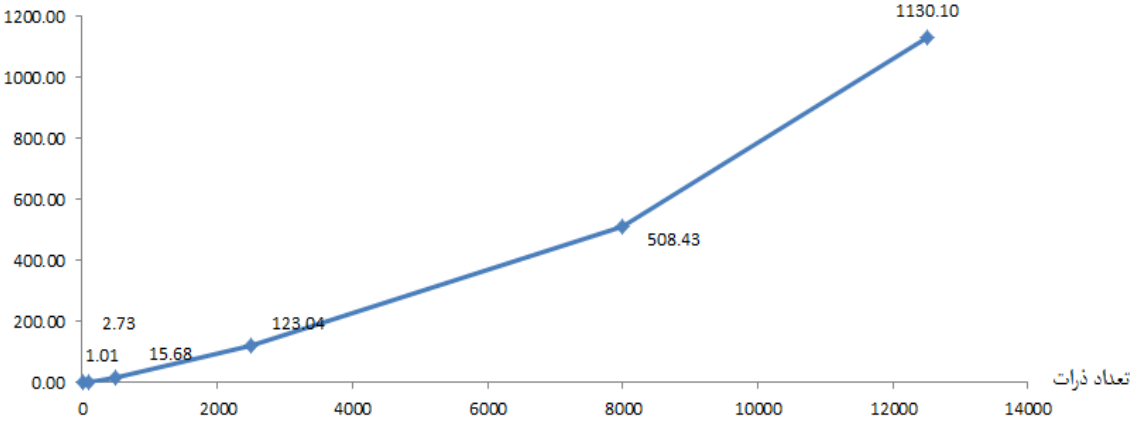
جدول ۵-۶: زمان متوسط پردازش هر قاب بر اساس تعداد ذرات در فیلتر ذره‌ای (میلی‌ثانیه)

فیلتر ذره‌ای موازی سازی شده			فیلتر ذره‌ای موازی نشده	
زمان کل پردازش موازی	انتقال داده بین پردازنده‌ها	پردازنده گرافیکی	پردازنده مرکزی	تعداد ذرات
۳/۶۱	۲/۱۸	۰/۲۳	۰/۲۳	۲۰
۳/۷۵	۲/۷۳	۰/۲۶	۰/۷۳	۱۰۰
۳/۸۵	۲/۸۹	۰/۲۸	۴/۳۹	۵۰۰
۴/۸۲	۳/۷۲	۰/۳۳	۴۱/۰۹	۲۵۰۰
۴/۸۸	۴/۲۲	۰/۳۵	۱۷۸/۰۵	۸۰۰۰
۴/۹۵	۴/۲۹	۰/۳۶	۳۹۹/۱۵	۱۲۵۰۰

به منظور درک بهتر تاثیر موازی سازی و عملکرد پردازنده گرافیکی در فرآیند فیلتر ذره‌ای، در شکل (۵-۴) نسبت زمان متوسط فرآیند فیلتر ذره‌ای در حالت موازی سازی نشده بر زمان متوسط این فرآیند در حالت موازی سازی شده، برای هر قاب نشان داده شده است. همچنین شکل (۵-۵) نشان دهنده نسبت زمان متوسط پردازش پردازنده مرکزی بر زمان متوسط پردازش پردازنده گرافیکی، برای هر قاب را نمایش می‌دهد.



شکل ۵-۴: نسبت زمان کل پردازش غیر موازی بر پردازش موازی با توجه به تعداد ذرات در فیلتر ذره‌ای



شکل ۵-۵: نسبت زمان پردازش پردازنده مرکزی بر پردازنده گرافیکی با توجه به تعداد ذرات در فیلتر ذره‌ای

۵-۴ بررسی نتایج و تحلیل یافته‌ها

همانطور که می‌دانیم یک فرآیند ردیابی مطلوب با کارایی مناسب از بخش‌های متعددی تشکیل شده است. اساسی‌ترین مراحل فرآیند ردیابی را می‌توان تشخیص اشیاء متحرک و در نتیجه تفکیک پس زمینه و پیش زمینه، و نهایتا مرحله ردیابی و ارائه شئ دانست.

در دو بخش قبلی در این فصل نتایج زمانی حاصل از موازی سازی این دو مرحله را بیان کردیم. با مراجعه به جداول نتایج و شکل‌های بیان شده در این دو بخش کاملا واضح است که نمی‌توان گفت موازی سازی برای هر شرایطی مناسب است و موجب بهبود سرعت پردازش می‌شود. به عنوان نمونه در بخشِ تشخیصِ اشیاءِ متحرک، موازی سازی برای ویدئوهای با ابعاد پایین موجب افزایش زمان پردازش می‌شود (به شکل (۵-۳) مراجعه شود). همچنین برای فرآیند فیلتر ذره‌ای، پردازش موازی به منظور پردازش داده‌ها در حالتی که تعداد ذرات زیاد نباشند نیز موجب بهبود سرعت پردازش نمی-شود (به شکل (۵-۴) مراجعه شود). بنابراین استفاده از موازی سازی با استفاده از پردازنده گرافیکی، برای افزایش سرعت پردازش، زمانی مناسب خواهد بود که حجم داده‌های پردازشی بالا باشد؛ و به منظور پردازشِ فرآیند با حجمِ داده‌های کم، استفاده از پردازش تک هسته‌ای از سرعت بیشتری برخوردار خواهد بود. این مسئله در سال‌های اخیر در پژوهش‌های مشابهی [۴۶] نیز مطرح شده است.

با دقت در نتایج زمانی بیان شده در جداول (۵-۳) تا (۵-۶) می‌توان متوجه شد، مهمترین عامل افزایش زمان پردازش فرآیند با حجم داده‌های نه چندان زیاد، در حالت موازی شده توسط پردازنده گرافیکی، زمانِ زیادِ مربوط به انتقال داده‌ها، تعیین پارامترهای کرنل‌ها، اختصاص حافظه و ... می‌باشد؛ و نه زمان پردازش توسط واحد پردازنده گرافیک. بنابراین می‌توان نتیجه گرفت که مشترک بودن حافظه‌ها، و یکسان بودن سطح دسترسی به حافظه‌ها، برای پردازنده گرافیکی و پردازنده مرکزی، سبب کاهش زمان انتقال داده‌ها شده و در نتیجه سرعت پردازش در حالت موازی بهبود خواهد یافت. این مسئله در پژوهش‌های مرتبطِ انجام شده در سال‌های اخیر [۵۰-۴۷] نیز بیان شده است. بنابراین انتظار داریم در سیستم‌هایی با این خاصیت (مشترک بودن حافظه پردازنده گرافیکی و پردازنده مرکزی)، مانند بُردهای پردازشیِ فشرده[1] (مانند رسپبری پای[1]، بیگل بن[2] و ...) مشکلِ زمانِ زیادِ انتقالِ داده بین دو پردازنده تا حد زیادی بر طرف گردد [۵۱،۵۲].

[1] Embedded Boards

۵-۵ جمع‌بندی

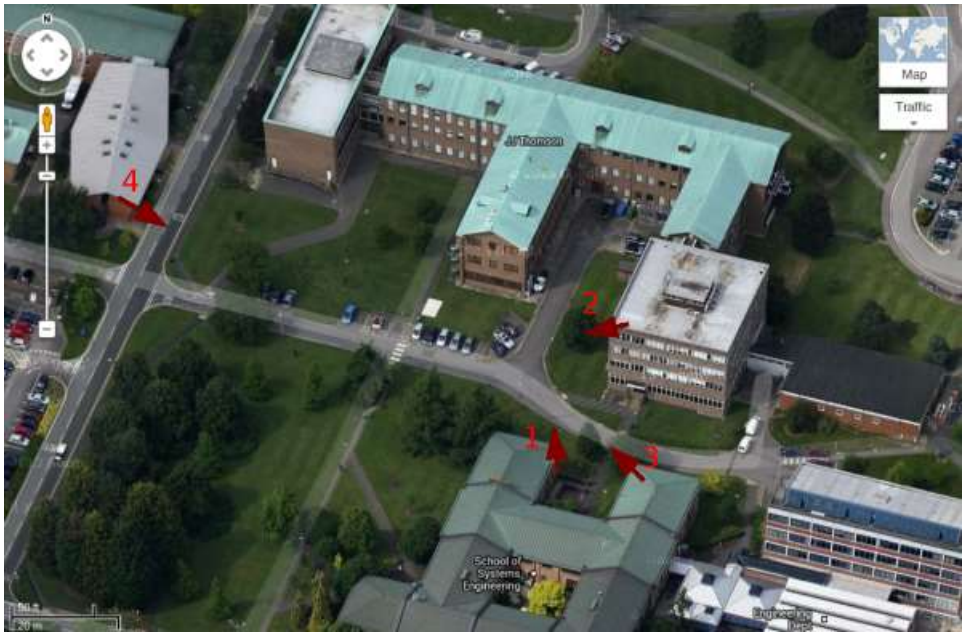
در این فصل نتایج حاصل از ارزیابیِ روش‌های مطرح شده در فصل چهارم را بیان کردیم. همچنین عملکرد زمانیِ روش‌های بیان شده، در حالت‌های پردازشِ موازی، و موازی سازی نشده، با توجه به ابعاد ویدئو و تعداد ذرات (افزایش تعداد ذرات در فیلتر ذره‌ای موجب بهبود تخمین مکان شئ می-شود)، را بررسی کردیم. دیدیم که استفاده از پردازش موازی در حالتی که حجم داده‌های پردازشی زیاد است موجب بهبود سرعت پردازش می‌شود و استفاده از پردازش تک هسته‌ای (موازی سازی نشده) برای پردازش داده‌های با حجم کم، به منظورِ دستیابی به پردازشِ بلادرنگ مناسب است. یکی از علت‌های اصلی افزایش زمان پردازش، در حالتِ موازی سازی شده با استفاده از پردازنده گرافیکی، زمانِ زیادِ انتقال داده‌ها بین پردازنده گرافیکی و پردازنده مرکزی است. مشکلی که در بُردهای پردازشی فشرده با مشترک کردن حافظه در دسترس برای دو پردازنده تا حد زیادی بر طرف شده است.

قابل ذکر است به منظور بررسی روشِ پیشنهاد شده و انجام آزمایشات، بسیاری از ویدئوهای استفاده شده در این پژوهش، از پایگاه داده PETS [۵۳] استخراج شده است. پایگاه داده PETS مجموعه‌ای از تصاویر ویدئویی ضبط شده توسط چندین دوربین است. ویدئوها در محوطه دانشکده علوم کامپیوتر دانشگاه ردینگ انگلستان[۳]، و به منظور بررسی و استفاده در پژوهش‌های مرتبط با ردیابی اشیاء، بررسی رفتار انسانی و تشخیص رفتارهای غیر طبیعی تهیه شده است. ویدئوها برای سال‌های مختلف با درجه تفکیک متفاوت و دوربین‌های مختلف، تهیه شده‌اند. به عنوان نمونه پایگاه داده‌ی ارائه شده در سال ۲۰۱۶ [۵۴] متشکل از بیست و دو نوع نمایش[۴] از رفتارهای انسانی است و بوسیله چهار

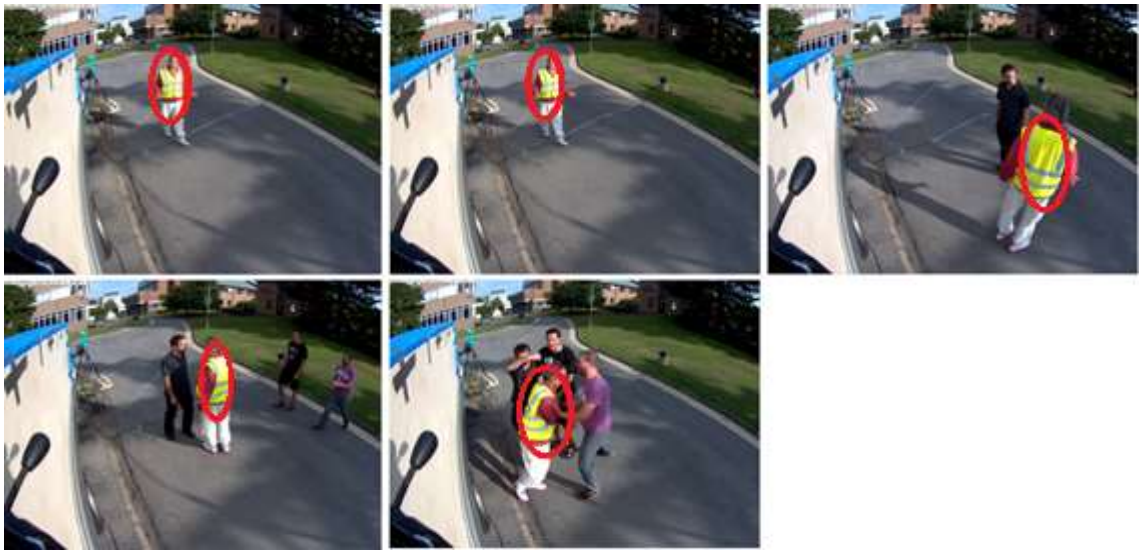
[1] Raspberry pi
[2] Beaglebone
[3] University of Reading
[4] Scenario

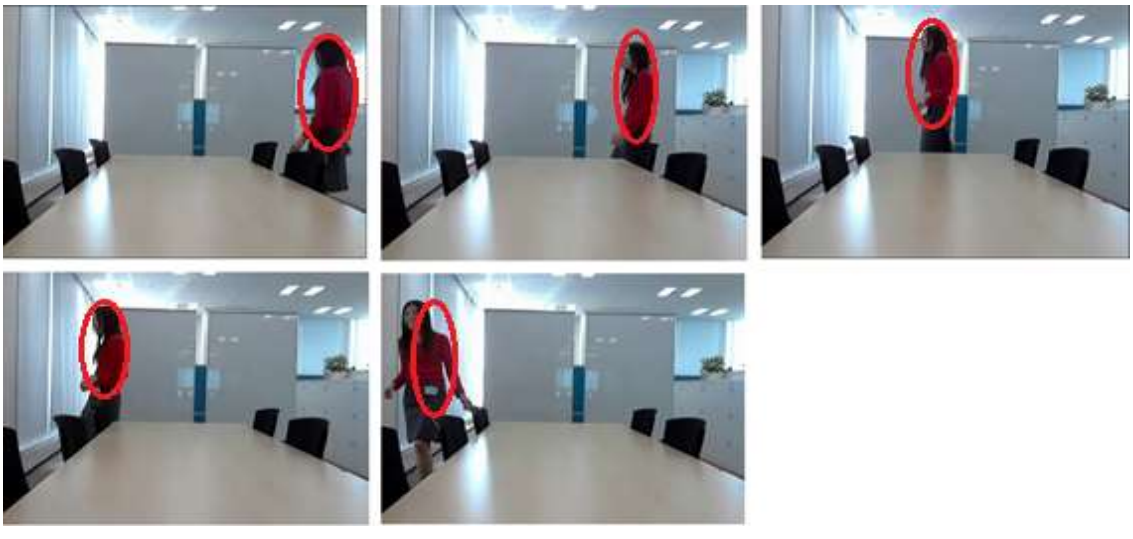
دوربین با درجه تفکیک‌های ۱۲۸۰×۹۶۰ و ۸۰۰×۶۰۰ و با نرخِ قابِ سی تهیه شده‌اند؛ که در مجموع شامل هشتاد و هشت ویدئو می‌شود. در شکل (۵-۶) محل استقرار دوربین‌ها برای تهیه این پایگاه داده برای سال‌های ۲۰۱۴ به بعد نمایش داده شده است.

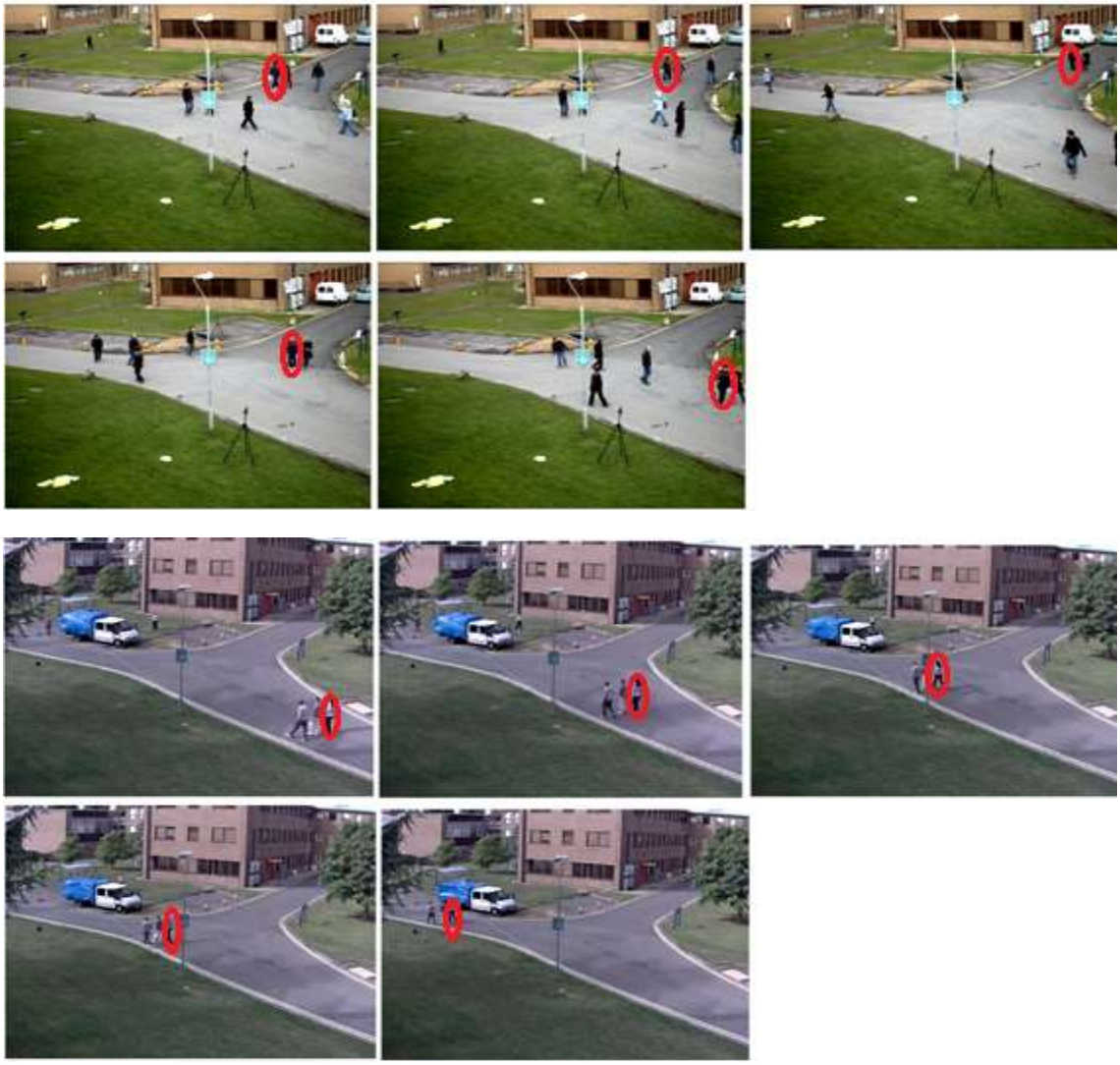
در انتها تعدادی از ویدئوها و عملکرد روش بیان شده در این پژوهش به منظورِ ردیابی، در شکل (۵-۷) نشان داده شده است.



شکل ۵-۶: محل قرارگیری دوربین‌ها برای تهیه پایگاه داده PETS از سال ۲۰۱۴ به بعد [۵۵]







شکل ۵-۷: ردیابی اشیا در نمونه ویدئوهای مورد استفاده

فصل ششم: جمع‌بندی

۶-۱ جمع‌بندی

در این پژوهش، مباحث مرتبط با ردیابی اشیاء مطرح شد. اهمیت موضوع ردیابی بلادرنگ را بررسی کردیم و تعدادی از پژوهش‌های مهم و تاثیر گذار در این حوزه را مرور کردیم. روند کلی فرآیند ردیابی اشیاء بیان شد و هریک از مراحل آن را بطور مجزا، همراه با روش‌های متداول مورد بررسی قرار دادیم.

به منظور افزایش سرعت پردازش و دستیابی به پردازش بلادرنگ پیشنهاد کردیم فرآیند ردیابی را با بهره‌گیری از واحد پردازنده گرافیکی موازی سازی کنیم. به منظور بررسیِ میزانِ تاثیر موازی سازی، روش‌های متداول از روند ردیابی مانند تفاضل قاب و فیلتر ذره‌ای را بر روی واحد پردازنده گرافیکی پیاده سازی کردیم و با توجه به ابعاد ویدئو و انتظار دقتی که از ردیاب داشتیم، سرعت فرآیند موازی سازی شده را با فرآیند در حالت پردازش غیر موازی مورد مقایسه قرار دادیم.

نتایج حاصل از بررسیِ روندِ پیشنهادی نشان می‌دهد که موازی سازی در کنار پردازش تک هسته‌ای می‌تواند به افزایش سرعت پردازش کمک کند و نه به تنهایی. در پردازش داده‌ها با حجم کم استفاده از پردازش غیر موازی از سرعت بیشتری برخوردار است و در پردازش داده‌های با حجم بالا استفاده از پردازش موازی به صورت قابل ملاحظه‌ای سرعت پردازش را افزایش می‌دهد. نتایج بدست آمده مشخص می‌کند، علت این موضوع تا حدود زیادی مربوط به زمان انتقال داده‌ها بین دو پردازنده است؛ و انتظار می‌رود با استفاده از سیستم‌هایی که دارای حافظه مشترک بین پردازنده گرافیکی و پردازنده مرکزی هستند، این مسئله برطرف گردد.

بسیاری از سیستم‌های مورد استفاده در صنعت بینایی ماشین، با پیشرفت تکنولوژی، افزایش حجم داده‌های پردازشی و افزایش انتظارات کاربران، نیازمند بروزرسانی هستند. این بروزرسانی می‌تواند شامل بروزرسانی نرم افزاری و یا سخت افزاری سیستم مورد استفاده، به منظور برآورده کردن انتظارات کاربران و یا دستیابی به سرعت پردازشی بیشتر باشد. با توجه به اینکه قیمت پردازنده‌های

گرافیکی در مقابل پردازنده‌های مرکزی بسیار کمتر است، بجای تعویض پردازنده مرکزی، استفاده از پردازنده‌های گرافیکی در کنار پردازنده‌های مرکزی به منظور افزایش سرعت پردازشِ سیستم می‌تواند به عنوان راهکاری موثر مورد بررسی قرار گیرد. بنابراین از جمله کاربردهای این پژوهش را می‌توان کاهش هزینه‌های بروز رسانی سیستم‌های قدیمی نیز ذکر کرد.

۶-۲ پیشنهادات

در ادامه پیشنهاد می‌شود آزمایش‌های انجام شده بر روی دستگاه‌های مختلفی انجام شود و نتایج استفاده از دستگاه‌های مختلف نیز با یکدیگر مورد مقایسه قرار گیرد. همچنین پیشنهاد می‌شود روش‌های بیشتری موازی سازی شود و سرعت پردازش آن‌ها در حالت موازی سازی شده در مقابل پردازش غیر موازی مورد بررسی قرار گیرد.

در حوزه پژوهش‌های مرتبط با بینایی ماشین و پردازش تصویر بر روی بستر پردازنده گرافیکی، جای خالی یک پایگاه داده مناسب با کیفیت و درجه تفکیک بالا به شدت احساس می‌شود و پیشنهاد می-شود در صورت امکان پایگاه داده استانداردی تهیه شود که نتایج حاصل از پژوهش‌های این حوزه قابلیت مقایسه دقیق و کاربردی را نیز دارا باشند.

مراجع

[1] Y. Wu, J. Lim, and M. H. Yang, "Online Object Tracking: A Benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411-2418.

[2] W. Żorski and P. Skłodowski, "Object tracking and recognition using massively parallel processing with CUDA," in *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2015, pp. 977-982.

[3] M. Rofouei, M. Moazeni, and M. Sarrafzadeh, "Fast GPU-based space-time correlation for activity recognition in video sequences," in *2008 IEEE/ACM/IFIP Workshop on Embedded Systems for Real-Time Multimedia*, 2008, pp. 33-38.

[4] Y. Qian and G. Medioni, "A GPU-based implementation of motion detection from a moving platform," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, pp. 1-6.

[5] J. Macri, "AMD's next generation GPU and high bandwidth memory architecture: FURY," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, 2015, pp. 1-26.

[6] A. Li and S. Yan, "Object Tracking With Only Background Cues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 11, pp. 1911-1919, 2014.

[7] X. Li and C. Xu, "Moving object detection in dynamic scenes based on optical flow and superpixels," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 84-89.

[8] S. V. Kothiya and K. B. Mistree, "A review on real time object tracking in video sequences," in *2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO)*, 2015, pp. 1-4.

[9] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, Accurate Detection of 100,000 Object Classes on a Single Machine," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1814-1821.

[10] C. C. Lee, C. H. Chuang, J. W. Hsieh, M. X. Wu, and K. C. Fan, "Frame difference history image for gait recognition," in *2011 International Conference on Machine Learning and Cybernetics*, 2011, vol. 4, pp. 1785-1788.

[11] X. Han, Y. Gao, Z. Lu, Z. Zhang, and D. Niu, "Research on Moving Object Detection Algorithm Based on Improved Three Frame Difference Method and Optical Flow," in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 2015, pp. 580-584.

[12] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1-4.

[13] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, "Review and evaluation of commonly-implemented background subtraction algorithms," in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1-4.

[14] D. Sun, S. Roth, and M. J. Black, "Secrets of optical flow estimation and their principles," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2432-2439.

[15] M. Gupta, L. Behera, V. K. Subramanian, and M. M. Jamshidi, "A Robust Visual Human Detection Approach With UKF-Based Motion Tracking for a Mobile Robot," *IEEE Systems Journal*, vol. 9, no. 4, pp. 1363-1375, 2015.

[16] N. Y. Khan, B. McCane, and G. Wyvill, "SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset," in *2011 International Conference on Digital Image Computing: Techniques and Applications*, 2011, pp. 501-506.

[17] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, pp. 13-18, 2006.

[18] L. Shengnan, S. Huansheng, C. Hua, and W. Guofeng, "A Point-Based Tracking Algorithm for Vehicle Trajectories in Complex Environment," in *2014 Fifth*

International Conference on Intelligent Systems Design and Engineering Applications, 2014, pp. 69-73.

[19] J. Dunik, O. Straka, M. Simandl, and E. Blasch, "Random-point-based filters: analysis and comparison in target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 2, pp. 1403-1421, 2015.

[20] J. Baskaran and R. Subban, "Compressive object tracking; A review and analysis," in *2014 IEEE International Conference on Computational Intelligence and Computing Research*, 2014, pp. 1-7.

[21] Y. Wu, J. Lim, and M. H. Yang, "Object Tracking Benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834-1848, 2015.

[22] J. Mak, M. Hess-Flores, S. Recker, J. D. Owens, and K. I. Joy, "GPU-accelerated and efficient multi-view triangulation for scene reconstruction," in *IEEE Winter Conference on Applications of Computer Vision*, 2014, pp. 61-68.

[23] A. K. Sahoo, G. Kumar, G. Mishra, and R. Misra, "A new approach for parallel region growing algorithm in image segmentation using MATLAB on GPU architecture," in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*, 2015, pp. 279-283.

[24] N. Shengxiao, Y. Jingjing, W. Sheng, and C. Gengsheng, "Improvement and parallel implementation of canny edge detection algorithm based on GPU," in *2011 9th IEEE International Conference on ASIC*, 2011, pp. 641-644.

[25] K. Pauwels and M. M. V. Hulle, "Realtime phase-based optical flow on the GPU," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008, pp. 1-8.

[26] D. D. Doyle, A. L. Jennings, and J. T. Black, "Optical flow background subtraction for real-time PTZ camera object tracking," in *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2013, pp. 866-871.

[27] H. Cho, S. J. Kang, S. I. Cho, and Y. H. Kim, "Image segmentation using linked mean-shift vectors and its implementation on GPU," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 4, pp. 719-727, 2014.

[28] X. Fan, Y. Cheng, and Q. Fu, "Moving Target Detection Algorithm Based on Susan Edge Detection and Frame Difference," in *2015 2nd International Conference on Information Science and Control Engineering*, 2015, pp. 323-326.

[29] G. M. Wojcik and W. A. Kaminski, "Pattern Separation in the Model of Mammalian Visual System," in *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, 2006, pp. 309-312.

[30] J. Míguez, D. Crisan, and I. P. Mariño, "Particle filtering for Bayesian parameter estimation in a high dimensional state space model," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, 2015, pp. 1241-1245.

[31] https://jp.mathworks.com/matlabcentral/fileexchange/33666-simple-particle-filter-demo, retrieved at 2016-3-07.

[32] X. Zan, F. Gao, J. Han, and Y. Sun, "A Hidden Markov Model Based Framework for Tracking and Predicting of Attack Intention," in *2009 International Conference on Multimedia Information Networking and Security*, 2009, vol. 2, pp. 498-501.

[33] H. Youness, A. Ibraheim, M. Moness, and M. Osama, "An Efficient Implementation of Ant Colony Optimization on GPU for the Satisfiability Problem," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2015, pp. 230-235.

[34] K. Szczepankiewicz, M. Malanowski, and M. Szczepankiewicz, "Effective implementation of passive radar algorithms using general-purpose computing on graphics processing units," in *2015 Signal Processing Symposium (SPSympo)*, 2015, pp. 1-5.

[35] T. True, D. Sandler, and P. Odorico, "GPU-Based Real-Time System for Cinematic Virtual Reality Production," in *SMPTE 2016 Annual Technical Conference and Exhibition*, 2016, pp. 1-13.

[36] https://www.khronos.org/assets/uploads/developers/library/overview/OpenCL-Overview, retrieved at 2015-10-03.

[37] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, "Graphics processing unit (GPU) programming strategies and trends in GPU computing," *Journal of Parallel and Distributed Computing*, vol. 73, no. 1, pp. 4-13, 2013.

[38] A. Majumdar *et al.*, "A Taxonomy of GPGPU Performance Scaling," in *2015 IEEE International Symposium on Workload Characterization*, 2015, pp. 118-119.

[39] G. L. Bernstein and F. Kjolstad, "Why New Programming Languages for Simulation," *ACM Transactions on Graphics (TOG)*, vol .35 ,no. 2, pp. 9-17, 2016.

[40] S. Gupta and M. R. Babu, "Performance Analysis of GPU compared to Single-core and Multi-core CPU for Natural Language Applications," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 5, pp.94-103, 2011.

[41] M. H. Rahmad, M. Soo Saw, E. K. Karuppiah, and O. Hong, "Comparison of CPU and GPU implementation of computing absolute difference," in *2011 IEEE International Conference on Control System, Computing and Engineering*, 2011, pp. 132-137.

[42] M. Scarpino (**2012**), "Opencl in Action: How to Accelerate Graphics and Computation",Vol.1, Manning Publications Co, USA, pp. 86.

[43] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53-82, 2010.

[44] S. D. Gupta, M. Coates, and M. Rabbat, "Error Propagation in Gossip-Based Distributed Particle Filters," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 3, pp. 148-163, 2015.

[45] P. Closas and C. Fernández-Prades, "Particle filtering with adaptive number of particles," in *2011 Aerospace Conference*, 2011, pp. 1-7.

[46] S. A. Dawwd, A. Layla, and M. Noor, "Training Acceleration of Multi-Layer Perceptron using Multicore CPU and GPU under MATLAB Environment," *Al-Rafadain Engineering Journal*, vol. 23, no. 3, pp.43-52, 2015.

[47] P. Li and Y. Luo, "P4GPU: Acceleration of programmable data plane using a CPU-GPU heterogeneous architecture," in *2016 IEEE 17th International Conference on High Performance Switching and Routing (HPSR)*, 2016, pp. 168-175.

[48] Y. Fujii, T. Azumi, N. Nishio, S. Kato, and M. Edahiro, "Data Transfer Matters for GPU Computing," in *2013 International Conference on Parallel and Distributed Systems*, 2013, pp. 275-282.

[49] R. Mokhtari and M. Stumm, "BigKernel -- High Performance CPU-GPU Communication Pipelining for Big Data-Style Applications," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, 2014, pp. 819-828.

[50] Y. Liu, H. W. Tseng, M. Gahagan, J. Li, Y. Jin, and S. Swanson, "Hippogriff: Efficiently moving data in heterogeneous computing systems," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 2016, pp. 376-379.

[51] K. Janard and W. Marurngsith, "Accelerating real-time face detection on a raspberry pi telepresence robot," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 136-141.

[52] S. Solak and E. D. Bolat, "Real time industrial application of single board computer based color detection system," in *2013 8th International Conference on Electrical and Electronics Engineering (ELECO)*, 2013, pp. 353-357.

[53] http://ftp.pets.rdg.ac.uk/pub, retrieved at 2016-08-14.

[54] http://ftp.pets.rdg.ac.uk/pub/PETS2016/, retrieved at 2016-10-20.

[55] http://www.cvg.reading.ac.uk/PETS2014/a.html, retrieved at 2017-01-04.

Abstract

Object tracking can be defined as displaying an object's location changes and tracking it in a series of video images. Most of the practical tracking applications require a real-time algorithm. Also, it's notable that being real-time is more significant in security and military applications and has a remarkable effect on the performance of these systems.

The importance of the object tracking process is due to the fact that while in the recent years we've witnessed the use of cameras with high resolution in tracking systems; in most tracking methods, this issue causes the amount of information in each frame that should be processed to increase. In addition, considering the growing progress of technology and the increasing demands from smart systems, processing flows have become more complex in recent years. Therefore, it's impossible to achieve real-time processing with only software operations, and the use of hardware capabilities is necessary in order to achieve the desired results.

In this thesis, in order to perform real-time object tracking, we use parallel processing by the graphical processing unit besides the CPU, assuming a stationary camera is used. We implement some of the existing methods in real-time object tracking like frame difference, three-frame difference and particle filter on GPU and CPU. Finally, with comparison of the performance of the parallel algorithms on videos with various dimensions and complexities, we show that parallelization of the object tracking process using GPU besides the CPU in order to achieve real-time tracking would be an appropriate approach.

Obtained results show that processing time improves more than thousand times if the problem of too much data transmission delay is solved. For example, in the case of particle filter with 1200 particles, running the algorithm on GPU improves the speed by a factor of 1130 compared to the CPU. Relative to video dimensions using GPU for frame difference and three-frame difference results in speed increase by 1390 and 548 times respectively.

Keywords: Real-time Object Tracking, Parallel Processing, Graphics Processing Unit, OpenCL, Particle Filter, Frame Difference

Shahrood University of Technology

Faculty of Electrical Engineering and Robotic

M.Sc. Thesis in Electronics Engineering

Parallel Processing Toward Real-time Objects Tracking

By: Mehdi Moghimi

Supervisor:
Dr Hossein Khosravi

January 2017