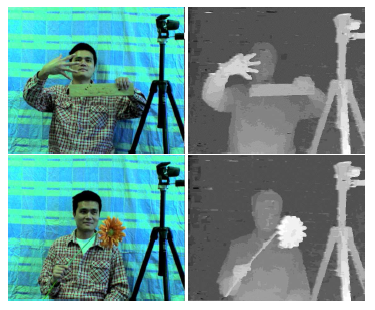
****

**کاربرد بینایی استریو در واحد پردازش گرافیکی (GPU)**

1. **مقدمه**

مبحث عمق استریو یکی از مرسوم­ترین و مهمترین موضوع در بینایی کامپیوتر است. اخیرا در این منطقه توسعه به طور قابل ملاحظه­ای پیشرفت چشمگیری داشته است. با این حال، از لحاظ سرعت، الگوریتم­های استریو که معمولا چند ثانیه تا چند دقیقه به منظور تولید یک نقشه و محدود کردن برنامه­های کاربردی پردازش می­شوند بهترین هستند. برنامه­های کاربردی بسیار جالبی مانند ناوبری ربات و واقعیت افزوده، که در آن عمق اطلاعات با کیفیت و سرعت ویدئو بالا بسیار مهم است وجود دارد.

در این مقاله ما تعدادی از تکنیک­هایی را که طی چند سال گذشته توسعه یافته­اند، به منظور بهبود سرعت و کیفیت الگوریتم­های استریو در زمان واقعی خلاصه می­کنیم. یکی از رایج­ترین تکنیک­های محاسبات سخت­افزاری گرافیکی کالاها (یعنی پردازنده­های گرافیکی) بهره­برداری از پهنای باند موازی است. ارزیابی با استفاده از پایگاه داده میدلبری نشان می­دهد که رویکردهایی که ما تاکنون ارائه داده­ایم یکی از بهترین و سریع­ترین الگوریتم­های استریو هستند.



شکل 1: دو نمونه تصویر و عمق نقشه آنها در سیستم زندگی ما در رایانه با پردازش 3.0 گیگاهرتز و کارت گرافیک Radeon XL 1800 نشان می­دهد. با این کیفیت، ما می­توانیم به 43 فریم در ثانیه با تصویر ورودی 240\*320 و 16 سطح پراکنده دسترسی پیدا کنیم.

1. **روش**

با توجه به یک جفت تصویر، هدف الگوریتم استریو این است که پیوندهای پیکسلی بین دو تصویر ایجاد کند. به طور کلی مکاتبات می­توانند به عنوان یک بردار ناسازگار بیان شوند، و با توجه به یک جفت تصاویر، هدف الگوریتم استریو ایجاد پیوندهای پیکسلی بین دو تصویر است. به طور کلی مکاتبات می­توانند به عنوان یک بردار ناسازگار بیان شوند، اگر پیکسل­های P­L(x,y) و PR(x’,y’) مربوط به تصویر سمت چپ و راست باشند، سپس اختلاف PL(x,y) و PR(x’,y’) در تصاویر آنها به عنوان مختصات-[x-x’,y-y’] تعریف می­شود. بنابراین، خروجی یک الگوریتم استریو در نقشه نامتوازن است، به عنوان مثال، نقشه­ای که اختلاف هر بردار را برای هر پیکسل در یک تصویر ثبت می­کند (تصویر مرجع) – تقشه نامتوازنی را برای تصویر دیگر به صورت خودکار با توجه به تقارن بردارهای تناوبی تعریف می­کند.

چارچوب استریو شامل چهار مرحله عمده است: اصلاح، هزینه تطبیق محاسبات، هزینه جمع­آوری، و در نهایت گزینش اختلاف. اصلاح شامل تغییر شکل 2 بعدی در هر تصویر می­باشد تا خطوط اپی­پولار با خطوط اسکن هم­تراز شوند. در مرحله دوم، هزینه تطبیق محاسبات برای هر مقدار اختلاف به صورت اختیاری برای هر پیکسل محاسبه می­شود. برای کاهش ابهام در تطبیق، هزینه یک پنجره کوچک همسایه (منطقه پشتیبانی) در مرحله سوم جمع­آوری محاسبه می­شود. آخرین گام گزینش اختلاف در هر مقدار مطلوب پیکسل برآورد می­شود. در چند بخش بعدی، ما چند روش ارائه می­دهیم تا سخت افزار گرافیکی حداکثر شتاب را دریافت کنند.

**2.1 اصلاح**

رویکرد استاندارد برای عمل اصلاح جفت تصاویر شامل اعمال هموگرافی 3\*3 تصاویر استریو می­باشد که خطوط اپی­پولار را با خطوط مربوطه مرتبط می­کند. همچنین یک روش معمول برای اصلاح لنز غیرخطی در یک زمان می­باشد. بهینه­سازی رایج این است که یک جدول جستجو را ایجاد کنید که افست پیکسل را اعوجاج لنز و متوازنی را تصحیح کند. آخرین نسل از پشتیبانیِ سخت افزار گرافیکی به بافت وابسته می­باشد و باعث می­شود اصلاح دقیق در هر پیکسل ممکن باشد.

**2.2 هزینه تطبیق محاسبات**

هزینه تطبیق به طور وسیع مورد استفاده قرار می­گیرد و اختلافِ قدر مطلق بین شدت پیکسل چپ و راست به صورت زیر می­باشد:

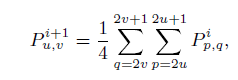
|PL(x,y) – PR(x+d,y)| . (1)

d مقدار تقارن فرض شده است. برای هر پیکسل PL(x,y) در تصویر مرجع، ما تمام فرضیه­های عدم انطباق را برای هزینه­های مربوطه با استفاده از معادله 2.2 محاسبه می­کنیم. در نهایت، ما مقدار هزینه تطبیق C را به دست می­آوریم- و آرایه سه بعدی x, y, d مشخص می­شود. محاسبات را می­توان در GPU با استفاده از برنامه­های قطعه­بندی و نقشه­برداری بافت انجام داد. علاوه بر این، پیکسل­ها می­توانند برای استفاده از قابلیت پردازش بردار GPU طبقه­بندی شوند. جزئیات را می­توان در [5] یافت.

**2.3 هزینه جمع­آوری**

وظیفه هزینه جمع­آوری بیش از هزینه تطبیق در پیکسل در یک پنجره کوچک است بدین منظور که ابهام در هزینه تطبیق کاهش یابد. در آنسوی هدف، ما سه روش مختلف را توسعه داده­ایم که عبارتند از: MIPMAP، پنجره تطبیق، و وزن رنگ.

GPUهای MIPMAP مدرن دارای جعبه­های ساخته شده در فیلتر هستند تا به طور موثر تمام سطوح MIPMAP مورد نیاز برای ساخت بافت را تولید کنند. از یک تصویر پایه P0 فیلتر زیر شروع می­کند و به صورت بازگشتی اعمال می­شود:



که (u,v) و (p,q) مختصات پیکسل هستند. بنابراین، 2n \* 2n به طور موثر جمع مقادیر کارآمد در پنجره است. هزینه تطبیق نهایی برابر اختلاف پنجره­ها می­باشد. این رویکرد ویژگی­های سراسری پنجره­های بزرگ را با حداقل ضلع محلی پنجره­های کوچک ترکیب می­کند. جزئیات را می­توان در [4] یافت.

پنجره تطبیقی (AW) یکی­دیگر از طرح­های جمع­آوری هزینه است که از یک پنجره سازگار که می­تواند در هر مکان پیکسل دقیقی را ارزیابی کند استفاده می­کند. طرح ما دارای شش پنجره پشتبانی متفاوت می­باشد که هر کدام به پیکربندی شکل گوشه، لبه و غیره مربوط می­شوند. یکی با حداقل امتیاز به عنوان هزینه تطبیق جمع­آوری مورد استفاده قرار می­گیرد. برای کارآمد بودن این هزینه در GPU، ما از پشتیبانی سخت­افزاری برای درون­یابی بافت دارای دو خط استفاده می­کنیم. با نمونه­برداری در وسط 4 پیکسل، به صورت احتمالی میانگین پیکسل­ها محاسبه می­شوند. برای جمع کردن مقادیر پنجره بزرگ، ما الگوریتم 2 مرحله­ای را پیاده­سازی کردیم. اطلاعات بیشتر را می­توان در [5] یافت.

وزن رنگ (CW) ما رویکرد وزن سازگار را از [6] اتخاذ کردیم. ایده اصلی این است که هزینه تطبیق را براساس هر دو رنگ و از لحاظ نزدیکی به هندسی تجمیع کنید. با توجه به یک پیکسل p (ما شاخص­های مختصات را برای سادگی در نماد کاهش دادیم) و یک پیکسل I در منطقه پشتیبانی، هزینه تطبیق I با اختلاف رنگ (∆cpl) بین p و I، و فاصله اقلیدسی (∆gpl) بین p و I در سطح تصویر می­باشد. فرمول وزن w(p,l) عبارتند از:



که و وزن ثابت هستند.

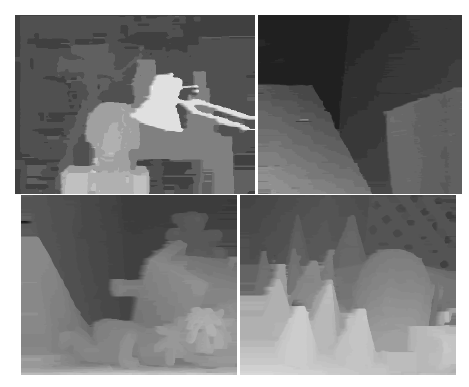
هزینه­ای که جمع­آوری شده به عنوان یک مجموع وزنی هر هزینه پیکسل محاسبه می­شود. این رویکرد ساده به طور موثر و قابل ملاحظه­ای نشان داده شده است. با این حال، این طرح تجمیعی از لحاظ محاسبات بسیار گران می­باشد. برخلاف بسیاری از طرح­هایی که از یک پنجره مستطیلی استفاده می­کنند، می­توانند به صورت موثر محاسبه یا به صورت جداگانه انجام شوند، و در این صورت وزن ماسک از پیکسل به پیکسل متفاوت می­باشد، زیرا پیکسل­ مرکز دارای رنگ­های مختلف است. همانطور که در [6] گزارش شد، برای ساخت یک نقشه با عمقی کوچک (به عنوان مثال، حدود 0.03 Mde/second) حدود یک دقیقه طول می­کشد که برنامه کاربردی در زمان واقعی به صورت غیرممکنی ساخته شود. ما دو طرح تقریبی را روی GPU طراحی کردیم تا سرعت این فرآیند را دو مرتبه افزایش دهیم. جزئیات بیشتر را می­توان در [2] یافت.

**2.4 گزینش نابرابری**

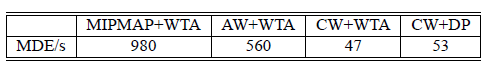
با توجه به محدودیت زمانی، فقط دو گزینه برای گزینش اختلاف وجود دارد: “برنده-همه” (WTA) که هر پیکسل را به مقدار تقاربی با حداقل هزینه اختصاص می­دهد یا برای استفاده از برنامه پویا (DP) نتایج را براساس اسکن خطی بهینه­سازی می­کند. WTA بسیار ساده­تر از محاسبات است اما نتیجه حاصل از آن به خطای تصویر و کالیبراسیون حساس می­باشد. DP نتایج را به صورت بهینه­سازی شده ارائه می­دهد، اما سازگاری بین اسکن خطی را اجرا نمی­کند. هر دو WTA و DP می­توانند در GPU اجرا شوند. ما دریافتیم که DP در GPUها به طور قابل توجهی احتمالا با توجه به ساختار پیچیده و ساختار حلقه دریافت نمی­شود.

**3. نتایج و نتیجه­گیری**

ما الگوریتم را با استفاده از مجموعه داده میدلبوری ارزیابی کردیم [1]. وزن رنگ براساس تجمیع ترکیبی با برنامه­نویسی پویا بهترین نتایج را به دست می­آورد، و در شکل 2 نیز نشان داده شده است. همانطور که در جدول 1 نشان داده شده است، از لحاظ سرعت MIPMAP+WTA سریع­ترین است و نزدیک به 1000 میلیون دیفرانسیل در ثانیه (MDE/s)1 را ارزیابی می­کند. در مقابل، بسته­های استریو تجاری فقط می­توانند حدود 150 MDE/s را دریافت کنند. برخی از نتایج ویدئویی را می­توان در [3] یافت.



شکل 2: نتیجه نگاشت عمقی با استفاده از color-weight+DP



جدول 1: سرعت الگوریتم­ها متفاوت می­باشد. سیستم تست یک کارت گرافیک 3.0 Ghz با یک کارت گرافیک Radeon از ATI، XL1800 است.

در حال حاضر ما بهبود کیفیت و سرعت بیشتر، از جمله روش­های مبتنی بر بهینه­سازی سراسری و بررسی هماهنگی زمانی را شروع کردیم. ما معتقدیم که انعطاف­پذیری بیشترِ عملیات نوشتن در سخت افزار کالا می­تواند به الگوریتم­های پیچیده­تر اجازه دهد تا از توانایی­های کامل خود بهره ببرند.

