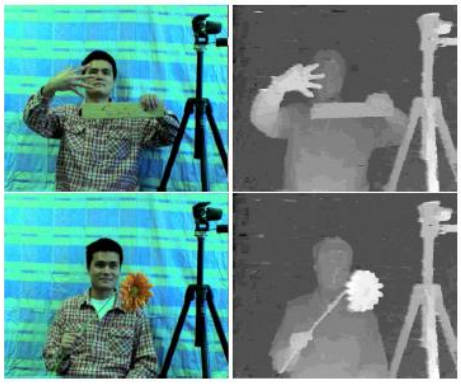
کاربرد بینایی استریو در واحد پردازش گرافیکی (GPU)

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.

2. 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 $P_L(x,y)$ و $P_R(x',y')$ مربوط به تصویر سمت چپ و راست باشند، سپس اختلاف $P_L(x,y)$ و $P_R(x',y')$ 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:

$$|P_L(x,y) - P_R(x+d,y)| . \qquad (1)$$

d مقدار تقارن فرض شده است. برای هر پیکسل $P_L(x,y)$ 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 P^0 فیلتر زیر شروع می‌کند و به صورت بازگشتی اعمال می‌شود:

$$P_{u,v}^{i+1} = \frac{1}{4} \sum_{q=2v}^{2v+1} \sum_{p=2u}^{2u+1} P_{p,q}^{i},$$

که (u,v) و (p,q) مختصات پیکسل هستند. بنابراین، $2^n * 2^n$ 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 l در منطقه پشتیبانی، هزینه تطبیق l با اختلاف رنگ (Δc_{pl}) بین p و l، و فاصله اقلیدسی (Δg_{pl}) بین p و l در سطح تصویر می‌باشد. فرمول وزن w(p,l) عبارتند از:

$$w(p,l) = \exp\left(-\left(\frac{\Delta c_{pl}}{\gamma_c} + \frac{\Delta g_{pl}}{\gamma_g}\right)\right),$$

که γ_c و γ_g وزن ثابت هستند.

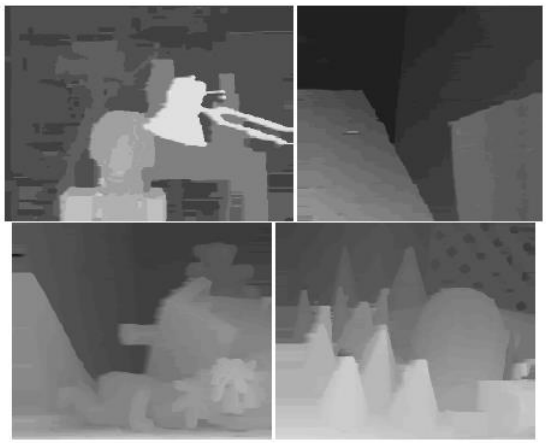
هزینه‌ای که جمع‌آوری شده به عنوان یک مجموع وزنی هر هزینه پیکسل محاسبه می‌شود. این رویکرد ساده به طور موثر و قابل ملاحظه‌ای نشان داده شده است. با این حال، این طرح تجمیعی از لحاظ محاسبات بسیار گران می‌باشد. برخلاف بسیاری از طرح‌هایی که از یک پنجره مستطیلی استفاده می‌کنند، می‌توانند به صورت موثر محاسبه 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

	MIPMAP+WTA	AW+WTA	CW+WTA	CW+DP
MDE/s	980	560	47	53

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.

4. REFERENCES

[1] D. Scharstein and R. Szeliski. www.middlebury.edu/stereo.

[2] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality Real-time Stereo using Adaptive Cost Aggregation and Dynamic Programming. In *Proceedings of 3DPVT*, 2006.

[3] L. Wang, M. Liao, and R. Yang. `http://www.vis.uky.edu/liaomiao/GPUstereo.htm`.

[4] R. Yang and M. Pollefeys. Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 211–218, 2003.

[5] R. Yang, M. Pollefeys, and S. Li. Improved Real-Time Stereo on Commodity Graphics Hardware. In *IEEE Workshop on Real-time 3D Sensors and Their Use (in conjunction with CVPR'04)*, 2004.

[6] K.-J. Yoon and I.-S. Kweon. Locally Adaptive Support-Weight Approach for Visual Correspondence Search. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 924–931, 2005.