STANDAARD PARALLELLE HARDWARE ALS FLEXIBEL EN ECONOMISCH ALTERNATIEF

Sneller vision-algoritmes verwerken

HOE KUNNEN WE SEQUENTIËLE VISION-ALGORITMES SNELLER VERWERKEN? EN HOE KUNNEN WE DIT OP EEN KOSTEN EFFICIËNTE MANIER DOEN EN DIT VERVOLGENS UITROLLEN NAAR EEN VEELVOUD AAN PLATFORMEN? JAAP VAN DE LOOSDRECHT VAN HET NHL KENNISCENTRUM COMPUTER VISION SCHREEF ER IN 2013 EEN THESIS OVER. HIJ ONDERZOCHT DE VOORS EN TEGENS VAN EEN VOOR DE HAND LIGGEND ANTWOORD: PARALLELLISATIE MET STANDAARD VERKRIJGBARE HARDWARE.

Tekst: Nick de With

Het laatste decennium is de vraag naar inspectie met behulp van computervision in de industrie enorm toegenomen. Applicaties worden in rap tempo almaar complexer, wat steeds vaker beperkingen oplevert in het real-time domein. Tegelijkertijd is de klokfrequentie van CPU's er sinds 2004 niet echt op vooruitgegaan. Computervisionapplicaties vragen dus om meer rekenkracht, maar lopen tegen de grenzen aan van de sequentiële architecturen van processoren. De manier om meer rekenkracht met standaard hardware te realiseren ligt voor de hand: parallel programmeren.

Academische aanpak

Zoals verwacht mag worden van een academisch onderzoek is Van de Loosdrecht zeer grondig en systematisch te werk gegaan om de winst van een parallelle benadering te kunnen kwantificeren. "Ik ben begonnen met een survey van 22 standaarden voor parallel programmeren", legt Van de Loosdrecht het begin van zijn zoektocht uit. "OpenMP bleek destijds de meest geschikte standaard voor het programmeren van multicore CPU's en OpenCL de beste standaard voor het programmeren van de GPU (red. Graphical Processing Unit). De factoren die voor ons hierbij de doorslag gaven waren de leverancieronafhankelijkheid en de efficiëntie van het parallelliseren van de code." Om de economische aspecten van parallellisatie in relatie tot de ontwikkelingsinspanningen en de mogelijke winst in verwerkingssnelheid te bepalen, maakte de NHL-onderzoeker gebruik van een bestaande bibliotheek. Van de Loosdrecht: "De gebruikte bibliotheek, VisionLab, een eigen ontwikkeling waar vele jaren aan ervaring is ingestopt, bevat meer dan 100.000 regels aan ANSI C++ sequentiële broncode. De bibliotheek bood welliswaar een mooi platformonafhankelijk uitgangspunt, maar het volledige parallelliseren ervan zou veel te veel tijd kosten. Gelukkig kunnen de standaard low-level operatoren voor beeldverwerking worden geclassificeerd in puntoperatoren, lokale neighbour-operatoren, global-operatoren en operatoren die te maken hebben met connectiviteit. Door vervolgens de aanname te maken dat vergelijkbare algoritmes een vergelijkbare snelheidswinst opleveren, kon gekozen worden voor een representatieve operator voor elke klasse. Er is gekozen voor de volgende operators threshold, convolution, histogram en connected component labeling. Vervolgens zijn deze operatoren in OpenMP 3.0 en OpenCL 1.1 geparallelliseerd en gebenchmarkt." Voor laatstgenoemde benchmark heeft Van de Loosdrecht een eigen benchmarkomgeving ontwikkeld om vast te kunnen stellen wat de voordelen zijn van geparallelliseerde algoritmen ten opzichte van hun sequentiële tegenhangers. De benchmarkopstelling is volledig opgebouwd uit standaard verkrijgbare parallelle multicore processoren en grafische kaarten. "Voor het onderzoek werd een quad-core Intel I7 processor gebruikt die draait op Windows 7, samen met een betaalbare low-end NVIDIA en een AMD grafische kaart. Veder werd een quad-core ARM gebruikt die op Linux draait. Voor het programmeren van de algoritmes zijn Visual Studio en de GNU tool chains gebruikt", aldus Van de Loosdrecht.

Parallelliseren van de CPU met OpenMP

Wie zich in de wereld van parallellisering wil storten, zou zich kunnen laten afschrikken door de extra leercurve die wordt verondersteld. Hier komt Van de Loosdrecht echter met een geruststellende bevinding voor wat betreft het gebruiken van OpenMP. "OpenMP is in feite een uitbreiding op C++ . Het kent een aantal specifieke pragma's en functies om parallellistie te realiseren. Het onder de knie krijgen van van OpenMP bleek relatief eenvoudig. Er is namelijk maar een beperkt aantal concepten aanwezig met een hoge mate van abstractie. Vaak betekende dit in de praktijk dat het parallelliseren van algoritmes als Threshold en Convolution neerkwam op het toevoegen van slechts een extra regel code met het OpenMP pragma. De meer ingewikkelde algoritmes zoals Histogram en Connectivity Component Labeling vergde respectievelijk wat meer en veel meer moeite." Uit de testresultaten die volgden bleek dat de 'overhead' van het extra werk niet opwoog tegen de tijdswinst die werd geboekt wanneer er met kleine beelden werd gewerkt. Om deze reden geniet het volgens Van de Loosdrecht de aanbeveling om op voorhand te kunnen voorspellen of parallellisatie voordelig is. Voor de portabiliteit hoeft de broncode volgens Van de Loosdrecht alleen maar opnieuw voor een ander platform te worden gecompileerd. "We hebben het getest voor Linux AM en Windows i7. Het poorten van de broncode verliep zonder problemen."

De resultaten

De maximale toenames in snelheid die voor elk van de vier operators werden bereikt, zijn weergegeven in de bovenstaande tabel. De grootste toename in snelheid werd geboekt bij grote beeldbestanden (Van de Loosdrecht, 2013).

Algoritme	OpenMP	OpenCL
Threshold	2.9	18.4
Convolution	4.9	60.9
Histogram	5.4	14.1
Connected component labeling	3.6	4.0

Parallelliseren van de gpu met OpenCL

Het parallelliseren van beeldverwerkingsalgoritmes voor de grafische kaart met OpenCL is volgens Van de Loosdrecht een stuk ingewikkelder. Wel zijn de winsten navenant. "De OpenCL standaard beschikt over een taal (gebaseerd op C99) voor het programmeren van kernels, functies die op de grafische kaart worden uitgevoerd en een API die op een CPU draait voor het bouwen, starten en besturen van de kernels op de CPU. Het leren en doorgronden van OpenCL was dan ook een lastige en tijdrovende klus. OpenCL introduceert namelijk een groot aantal nieuwe concepten met een laag abstractieniveau. Daarentegen is de OpenCL kerneltaal uiteindelijk wel goed te begrijpen en kunnen er grote winsten in verwerkingssnelheid worden geboekt als alle details van de gebruikte hardware goed worden begrepen. Bovendien kan OpenCL worden gebruikt voor het implementeren van algoritmes op de CPU, waardoor er met vectoren kan worden gewerkt. Conclusie? Voor de eenvoudige algoritmes kunnen snel grote tijdswinsten worden geboekt. Maar wil men de snelheid nog verder opvoeren dan zal men complexere en hardware afhankelijke optimalisaties moeten maken. En het parallelliseren hiervan middels OpenCL kan daarom een aardige kluif zijn. Voor het Connected Component Labeling algoritme was zelfs een volledig nieuwe aanpak nodig."

In plaats van alle host API code in C++ te schrijven, maakte Van de Loosdrecht gebruik van zijn eigen VisionLab scripts. Deze scripttaal werd uitgebreid met OpenCL host API commando's. Van de Loosdrecht: "Door commando's te scrip-

ten konden de ontwikkelingstijd en de testtijd van het coderen van de hosting aanzienlijk verkort worden." Volgens Van de Loosdrecht laten de testen zien dat de GPU implementaties voor verschillende GPU's elk hun eigen aanpak verdienen voor een optimale speedup. Hij verwacht dat de OpenCL operatoren wel te poorten zijn, maar dat de prestaties niet noodzakelijk worden overgedragen aan andere grafische apparaten.

Mooie spin-off

Hoewel het doel een gedegen onderzoek naar parallellisatie was, hebben de werkzaamheden van Van de Loosdrecht een mooie spin-off opgeleverd voor zijn VisionLab bibliotheek, die ook commercieel als product verkrijgbaar is. De vier basisalgoritmes konden als template gebruikt worden om ook andere operatoren in OpenMP te parallelliseren. Het kostte twee manmaanden werk, maar nu zijn er maar liefst 170 geparallelliseerde operators in VisionLab voorhanden, inclusief een runtime voorspellingsmechanisme ten aanzien van de te behalen snelheidswinst. VisionLab gebruikers hoeven hun scripts, C++ of C# code niet aan te passen om van deze parallellisatie gebruik te maken. Voor wat betreft OpenCL is een toolbox aan VisionLab toegevoegd. Gebruikers kunnen hiermee OpenCL host code schrijven op basis van scripttaal en hun kernels aanpassen. De OpenCL host interface is getest voor NVDIA, AMD en Intel OpenCL platformen."

Conclusie

Aplicaties met computervision worden snel complexer, met hogere real-time eisen. Meer rekenkracht is dus geen overbodige luxe, helemaal wanneer het einde van de resolutierace voorlopig nog niet in zicht is. Door parallelle VisionLab scripts toe te passen kunnen visiongebruikers zonder enige aanpassing direct een impuls geven aan de rekenkracht van hun systeem. Nog meer rekenkracht is mogelijk – helemaal op basis van tailor made OpenCL parallellisatie, maar dit brengt wel het nodige programmeerwerk met zich mee. Van de Loosdrecht: "Het blijft uiteindelijk de vraag of meer snelheid de moeite waard is. Het antwoord is grotendeels afhankelijk van de applicatie. Om een complete sequentiële bibliotheek te paralleliseren op basis van OpenCL lijkt me economisch geen haalbare kaart. Maar OpenCL heeft de potentie om een enorme boost aan de rekenkracht van een grafische kaart te geven, wat voor specifieke toepassingen erg aantrekkelijk kan zijn."

Toekomstblik

Tot slot vragen we Van de Loosdrecht een kijkje te nemen in de toekomst. Zijn er bijvoorbeeld nieuwe architecturen of versies die het speelveld zullen veranderen? "Ten dele wel. Voor de hedendaagse GPU's zien we bijvoorbeeld dat de overhead van dataoverdracht tussen de PC en de grafische kaart behoorlijk is wanneer deze wordt vergeleken met de executietijd van de kernels. Maar wanneer de nieuwe aangekondigde hetrogene CPU/GPU architecuren op de markt komen, dan zal deze overhead aanzienlijk minder zijn. Voor wat betreft nieuwe standaarden hebben we sinds kort OpenMP 4.0. Deze bevat zogenoemde 'directives for attached accelerators'. Hierdoor wordt het mogelijk zowel CPU's als GPU's poortbaar in OpenMP te programmeren, en wordt het gebruiken van pragma's daarmee ook mogelijk voor GPU's. Hetzelfde geldt voor het gebruik kunnen maken van vectoren. Ik verwacht alleen wel dat, omdat deze nieuwe standaard een hoger abstractieniveau kent, de code nog steeds niet zo efficiënt zal zijn als wanneer er geprogrammeerd word met OpenGL."
Cursusmateriaal over dit onderwerp is te vinden www.nhlcomputervision.nl/course.

Referenties

Khronos Group, 2011. Open Standards for Media Authoring and Acceleration. [online] : Khronos Group. Available at: http://www.khronos.org.
OpenMP, 2011. The OpenMP® API specification for parallel programming. [online] : OpenMP.org. http://openmp.org.
Van de Loosdrecht, J., 2013. Accelerating sequential Computer Vision algorithms using commodity parallel hardware. www.vdlmv.nl/thesis.
VdLMV, 2014. VisionLab. Demo version VisionLab , www.vdlmv.nl.

Contact informatie: Jaap van de Loosdrecht, NHL Kenniscentrum Computer Vision, j.van.de.loosdrecht@nhl.nl