Situace se ale zásadně změní v případě filtrů s nutností složitějších výpočtů v plovoucí řádové čárce, jako je tomu např. u transformací barevných prostorů, řešení saturačních matic, šumových filtrů (*obr. 5*) atd. Potom může využití GPU zrychlit výpočty až o dva řády. Jednoznačnou úlohou pro GPU jsou i takové filtry, kdy pro výpočet každého pixelu je nutné číst mnoho pixelů z jeho okolí – příkladem mohou být algoritmy lokálního prahování obrazu (*obr. 6*). Pak je schopnost GPU načítat desítky gigapixelů za sekundu z textur nenahraditelná. Řešení s GPU může být i několiksetkrát rychlejší než snaha o totéž v CPU.

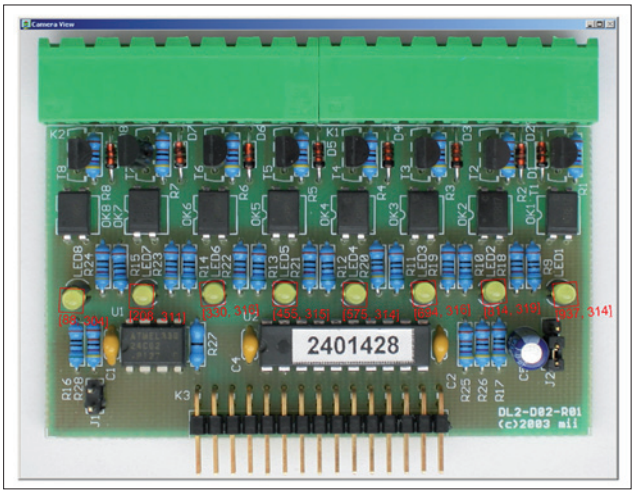
Ve prospěch GPU se situace dále přiklání ve všech případech, kdy je nutné určitý filtr vícekrát opakovat. Tento požadavek je charakteristický např. u morfologických filtrů, které nejsou samy o sobě výpočetně příliš náročné, ale často potřebují desítky i stovky opakovaných běhů – příkladem může být např. hledání koster objektů nebo segmentace ploch okolo objektů. Obrazová data během opakovaných běhů filtru neopouštějí grafickou paměť a jsou stále rychle přístupná pro GPU. Pro tyto algoritmy je masivní paralelizace velkým přínosem.

Transformační kódování

Příkladem transformačního kódování může být dvourozměrná Fourierova transformace – převod obrazu z prostorové (časové) do frekvenční domény (krok *gpu_fast_fourier_transform_2D*). Byť je zde použit algoritmus rychlé Fourierovy transformace (FFT), transformace obrazu je natolik výpočetně intenzivní, že v reálném čase není jinak než s využitím GPU realizovatelná. Zde není možné použít starší GPU – potřebné jsou vektory vstupních textur i výstupních bufferů s plovoucí řádovou čárkou. Takže není na výběr – bez moderního GPU to v tomto případě nejde.

Další možnosti

Veškeré dosud uváděné příklady byly celkem jednoznačné. Existují však i problematické úlohy, kde přínosy GPU nemusí být takto zřejmé. Příkladem takové úlohy může být např. rozpoznávání vzorů realizované prostřednictvím GPU (krok *gpu_match_monochrome* v systému VisionLab). Při hledání obrazového vzoru musí systém mnohokrát provádět normalizovanou křížovou korelaci s velkým počtem pixelů. To vyžaduje mnohonásobně opakované výpočty středních jasů, směrodatných odchylek, odmocnin atd., a to vše s plovoucí řádovou čárkou. Navíc krok hledá i pootočené nebo zvětšené



Obr. 8. Hledání vzorů realizované prostřednictvím GPU – tento příklad trvá na procesoru G80 asi 100 ms

či zmenšené obrazy předloh. Veškeré výpočty tedy navíc podléhají lineárním transformacím. Toto všechno vypadá na první pohled jako výborná úloha pro GPU. Avšak je zde závažná potíž – výpočetní intenzita algoritmu je natolik velká, že není „hrubou silou“ rozumně proveditelný ani nejvýkonnějšími GPU. Podobně jako v algoritmech pro CPU, musí i zde nastoupit řada optimalizací. A právě v těchto optimalizacích, bez kterých ale problém řešit nelze, se skrývá kámen úrazu. Algoritmus je nutné rozdělit do několika fází, v jejichž přechodech musí probíhat výměna dat mezi GPU a CPU. Časové ztráty způsobuje jednak kopírování bloků dat, ale také to, že CPU i GPU musí na sebe vzájemně mezi jednotlivými fázemi čekat a nemohou běžet paralelně. Tento krok je tedy výhodné realizovat jen s moderními velmi výkonnými GPU – čas hledání může být až desetinásobně kratší než na vícejádrových CPU. S méně výkonnými grafickými procesory se ale může situace obrátit a krok může být pomalejší než obdobný krok realizovaný čistě v CPU. Autor aplikace tedy musí vše zvážit a v praxi vyzkoušet.

Obdobně jako existují úlohy, pro které nelze masivně použít paralelní algoritmy v GPU, je spousta úloh, které nelze efektivně řešit dokonce ani paralelními vlákny v CPU. Neexistuje jediný lék na všechno. Pro každou aplikaci musí její autor zvolit nejlepší způsob řešení. A k tomuto rozhodování patří i to, kde využije obrovské schopnosti grafických procesorů.

Závěr

Systém strojového vidění VisionLab poskytuje krokům, které využívají GPU, značný prostor. S vykreslovacím jádrem systému Control Web i virtuálním přístrojem *gl_camera* komunikuje jen prostřednictvím velmi jednoduchého povelového rozhraní. Kroky dostanou k dispozici paměťový pixel buffer dané velikosti a formátu pixelů a mohou si zcela samostatně alokovat libovolné prostředky GPU – mohou si vytvářet např. vlastní texturové objekty, další objekty *frame buffer objects*, mohou načítat a spouštět vlastní shadery atd. Kroky strojového vidění tak mohou využívat veškerá nejnovější rozšíření, která přinášejí aktuální grafické ovladače. Tyto možnosti dovolují systému strojového vidění VisionLab stále sledovat poslední vývoj metod zpracování obrazu, a přinášet tak svým uživatelům vysoký užitek a konkurenční výhodu.

Roman Cagaš, Moravské přístroje (rc@mii.cz)

krátké zprávy

▶ NE 142 Funkční bezpečnost elektrických komponent

Nejdůležitější úlohou elektrických komponent ve vztahu k funkční bezpečnosti je bezpečnostní odpojení elektrických obvodů. Doporučení NE 142 NAMUR, německého sdružení uživatelů automatizační techniky v procesním průmyslu, je příručka s praktickými radami, jak pomocí elektrických komponent zajistit funkční bezpečnost výrobních zařízení. Nenahrazuje ovšem předběžné úvahy, během nichž se musí konstruktér rozhodnout, zda funkční bezpečnost zajistí elektrickými komponentami nebo jiným způsobem.

Dokument popisuje pro každý stupeň SIL tipy pro výběr a konstrukci elektrických komponent, které mají podstatnou funkci pro zajištění bezpečnosti, jako jsou bezpečnostní spínače a ochrany. Příklady popsané v doporučení NE 124 vedou především k použití normalizovaných komponent a možnostem jejich použití, popř. s nutnými úpravami, v systémech s úrovní funkční bezpečnosti SIL 1 až SIL 3.

Doporučení je dostupné (v němčině a angličtině) na www.namur.de. *(Bk)*