ISSN 2029-7149 online
ISBN 978-9955-28-835-0

GREITAS IR TIKSLUS OBJEKTO PARAMETRŲ NUSTATYMAS MAŠININĖS REGOS SISTEMOSE

Tadas Kazakevičius
Vilniaus Gedimino technikos universitetas
El. paštas: ktadas@gmail.com

Santrauka. Darbe aprašomas įgyvendintas realaus laiko objekto pozicijos nustatymo metodas, skirtas vykdyti su GPU lygiagretaus skaičiavimo architektūra. Lygiagrečiais skaičiavimais paremtų vaizdo apdorojimo algoritmų valdymui naudojama GLSL programavimo kalba ir OpenGL grafikos biblioteka. Pateiktas metodas yra atsparus vaizdo triukšmui, objekto daliniam uždengimui ir apšvietimo kitimui.

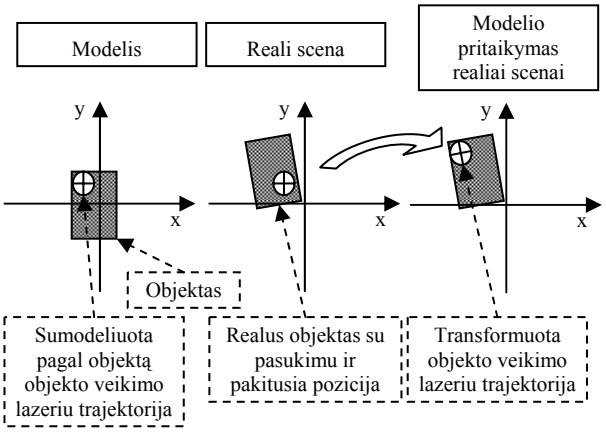
Reikšminiai žodžiai: mašininė rega, objekto atpažinimas, vaizdo apdorojimas su GPU.

Įvadas

Objekto atpažinimas ir pozicijos nustatymas gali būti pritaikomas daugelyje pramonėje egzistuojančių uždavinių, tokių kaip: paimti objektą ir perdėti į kitą vietą, nustatyti ir užtikrinti pagaminto gaminio kokybę, surūšiuoti ar suskaičiuoti objektus pagal spalvą, dydį, formą ar kitą savybę.

Šis darbas buvo skirtas lazerinės pramonės uždaviniams spręsti.

Viena iš aktualiausių lazerinės pramonės užduočių - sumodeliuotą graviravimo ar kitą lazerinio apdirbimo algoritmą (trajektorijas) pagal objekto modelį transformuoti taip, kad modelio objektas atitiktų realioje scenoje esantį objektą (1 pav.).



1 pav. Lazerinio proceso modeliavimas ir vykdymas.

Tokiu būdu koreguojant objekto veikimo lazeriu trajektorijas, galima greitai ir tiksliai atlikti tą pačią lazerinę procedūrą (pvz. užrašo graviravimas, pjovimas, medžiagos pašalinimas) dideliam kiekiui objektų. Objektas į lazerinio apdirbimo zoną galėtų patekti tiek padedant objektą operatoriaus, tiek techninio įrenginio - roboto ar konvejerio pagalba.

Greitis ir tikslumas yra labai svarbūs bruožai šiuolaikinių pramoninių, ypač precizinių lazerinių sistemų. Kompiuterinės regos užduotys reikalauja daug kompiuterio resursų, yra monotoniškos, dažnai išnaudoja visą CPU (centrinį procesorių) siekiant vaizdą apdoroti realiu laiku, tuo pačiu paliekant mažai CPU laiko atlikti sudėtingas užduotis.

Šiandienos vaizdo apdorojimo plokštės (GPU) tapo galingais skaičiavimo įrenginiais, kurių platesnės valdymo (programavimo) galimybės atsivėrė tik visai nesenai. GPU yra sukurtas darbui su grafiniais primityvais, tarp jų ir paveikslėlio taškais, todėl vaizdo apdorojimas su GPU dažniausiai yra greitesnis nei CPU. Esminis dalykas kodėl vaizdas greičiau apdorojamas su GPU, nei su CPU, tai lygiagreti GPU architektūra, su kuria galima vykdyti net iki kelių tūkstančių operacijų vienu metu lyginant su dešimtimis ar vienetais tokių pat operacijų su CPU.

Šiame darbe buvo naudojama GLSL programavimo kalba, skirta vykdyti operacijas su grafinėmis vaizdo plokštėmis, kuria kuriami programų moduliai kompiliuojami ir valdomi(paleidžiami) su OpenGL biblioteka.

Objekto parametrų nustatymo metodas

Pramoniniu būdu gaminamų, apdorojamų ar tikrinamų daiktų yra be galo daug, kurie turi įvairias spalvas, tekstūras, formas ar kitas savybes, kurios

komplikuoja objekto pozicijos nustatymą mašininės regos pagalba ir sunkina trajektorijų korekcijos įvedimą.

Praktikoje dažniausiai naudojama šablono paieška, kuri paremta didžiausio panašumo tarp paveikslėlio ir šablono taškų intensyvumų paieška (Hornberg 2006; Steger 2008). Detalesniam paieškos įsivaizdavimui galima įsivaizduoti slenkamą šabloną (paveikslėlį tik su dominančiu objektu) per visus paveikslėlio taškus ir skaičiuojamą taškų intensyvumų panašumą. Vieta, kur panašumo reikšmė didžiausia, parodo kuri paveikslėlio vieta yra panašiausia į šabloną. Deja šis metodas be pakeitimų ir optimizacijų netoleruoja nei apšvietimo pokyčių nei dalinio objekto uždengimo ir paieškos atlikimas reikalauja daug operacijų.

Kad metodas toleruotų šviesos pokyčius, šablono lyginimo metu lyginami tik objekto kraštai ar kampai, kurie beveik nejautrūs šviesos kitimui (Borgefors 1988; Hornberg 2006; Steger 2002; Strzodka 2003). Objekto kraštų išskyrimui paprastai naudojamas Canny metodas, kuris gali būti pritaikytas vykdyti su GPU (Luo 2008). Kraštų, o ne viso šablono taškų intensyvumų lyginimas taip pat leidžia turėti dalinai uždengtą objektą, kurio dalies kraštų netekimas nepaveiks lyginimo rezultato jei tik bus parinktas ne per daug griežtas šablono ir paveikslėlio dalies sutapimo slenkstis.

Kad lyginimas būtų greitas, mažinama vaizdo rezoliucija (dėl ko prarandamas tikslumas) arba naudojamas piramidinis-hierarchinis lyginimo principas, kurio metu pilnas šablono lyginimas atliekamas tik grubiausiame lygyje, o detalesniuose rezultatai tik patikslinami (Hornberg 2006; Steger 2002). Nagrinėti metodai yra gana greiti, tačiau nėra realaus laiko metodai. Kadangi šiuolaikinės vaizdo apdorojimo plokštės yra pigios lyginant su centriniais procesoriais, jų skaičiavimo galia dažniausiai stipriai lenkia centrinių procesorių galią, nagrinėtoje literatūroje GPU sėkmingai pritaikomi vaizdų apdorojimui(Borovikov 2009; Luo 2008; Strzodka 2003; Ziegler 2006), todėl verta išnaudoti GPU skaičiavimo galią vaizdų apdorojimo uždaviniams spręsti.

Šiame darbe pateiktas objekto parametrų nustatymo metodas, kuris naudoja GPU, susideda iš šių etapų:

- Šablono suformavimas
    1. Kraštų radimas
    2. Piramidės suformavimas
    3. Modelio suformavimas kiekvienam piramidės lygiui
- Šablono paieška
    1. Kraštų radimas
    2. Piramidės suformavimas
    3. Paieškos atlikimas

4. Paieškos rezultato suformavimas

Kad paieškos algoritmas vyktų efektyviai (greitai), reikia minimizuoti duomenų apsikeitimą tarp CPU ir GPU erdvės. Todėl pagrindinis metodo tikslas: įgyvendinti visus paieškos etapus su grafikos procesoriumi, kad būtų minimizuojami duomenų srautai.

Pirmas paveikslėlio apdorojimo etapas - paveikslėlio nusiuntimas į grafikos procesorių. Nusiuntimas vykdomas su standartinėmis OpenGL bibliotekos funkcijomis. Kadangi vaizdų apdorojime daugiausia dirbama su vaizdo taškais, šiame darbe naudojamos tik taškinės (angl. fragment shader) grafikos procesoriaus programos, kurios dažniausiai operuoja su taškais sudarytais iš raudonos, mėlynos, žalios ir permatomumo komponenčių (RGBA). Su keturiomis komponentėmis vietoje vienos verta dirbti dėl kelių priežasčių:

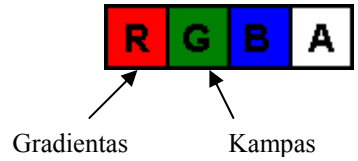
1. Grafikos procesorių architektūra leidžia su keturiomis komponentėmis dirbti taip pat greitai kaip su viena komponente.
2. Algoritmais apdorotus duomenis patogu saugoti komponentėse.
3. Dalį veiksmų galima keturis kartus greičiau atlikti apdorojant duomenis vienu metu (viena instrukcija) visose keturiose komponentėse.

Kraštų išskyrimas

Kraštų radimui naudojamas Canny kraštų radimo metodas, kuris pradedamas spalvoto paveikslėlio vertimu į nespalvotą. Kadangi vaizdo procesoriaus taškų programos rezultatas taškas, taško pilkumo lygio reikšmė įrašoma į visas komponentes. Toliau atliekama glodinimo operacija, kurios pagrindinis tikslas - panaikinti vaizdo paveikslėlyje esantį triukšmą. Triukšmas gali atsirasti dėl įvairių priežasčių. Pagrindinės yra dvi priežastys: kameros kokybė ir apšvietimo kokybė. Jei objektas prastai apšviestas, dažniausiai didinamas kameros sensoriaus jautrumas, dėl ko gali atsirasti kameros vaizdo triukšmų, kurie nepageidaujami kraštų išryškinimo metu (triukšmingi taškai būtų išryškinti). Todėl vaizdo glodinimui naudojamas Gauso filtras. Gauso filtravimas atliekamas dviem etapais. Visų pirma vienmatis Gauso filtras pritaikomas horizontalia kryptimi, po to vertikalia kryptimi. Jei imsime Gauso filtro matricos dydį n, tai apdorojamų taškų kiekis sumažėja 0,5*n kartų, kai filtravimas atliekamas dviem etapais (horizontalia ir vertikalia kryptimis) vienmačiu Gauso filtru, o ne vienu etapu dvimačiu Gauso filtru.

Po glodinimo operacijos, naudojant Sobel kraštų radimo operatorių, atliekama kraštų išryškinimo procedūra. Kraštų išryškinimo procedūra taip pat yra išskaidyta į du etapus. Visų pirma randami kraštų gradientai ir kampai, o toliau vykdomas nemaksimalių verčių atmetimas. Kiekvienam vaizdo taškui yra apskaičiuojamas gradientas ir gradiento kryptis. Apskaičiavimui naudojamas vertikalus ir horizontalus Sobel operatorius.

Kad galėtume atlikti ne maksimalių kraštų atmetimo procedūrą, suskaičiuotos gradiento ir kampo vertės turi būti išsaugotos į vaizdo tašką. Saugojimo procedūros metu į paveikslėlio taško raudoną komponentę yra saugoma gradiento, o į žalią kampo reikšmė (2 pav.).



2 pav. Gradiento ir gradiento krypties (kampo) saugojimas spalvinėse komponentėse.

Ne maksimalių kraštų atmetimo procedūra naudojama, kad būtų išskirtas tik gradiento lokalus maksimumas pagal gradiento kryptį. Kraštas pažymimas tik tuomet, jei krašto gradientas gradiento kryptimi yra lokaliai maksimalus ir viršija pasirinktą slenkstį.

Piramidės sudarymas

Toliau vykdoma piramidės sudarymo iš išskirtų modelio kraštų procedūra. Apatiniame piramidės lygyje (0 lygis) yra originalus paveikslėlis – modelio kraštai. Aukštesniame lygyje paveikslėlis yra du kartus mažesnis nei žemesniame lygyje esantis paveikslėlis. Kiekvieno aukštesnio lygio paveikslėlis sudaromas iš žemesniame lygyje esančio paveikslėlio.

Aukštesnio lygio taško vertė nusakoma kaip maksimali vertė žemesniame lygyje esančių keturių taškų. Tokiu būdu daroma piramidė, kol jos aukščiausio lygio paveikslėlio matmenys didesni už nustatytą minimalių dydį.

Toliau sudaryta piramidė perkeliama iš GPU į CPU erdvę. Tuomet vykdomas modelio duomenų struktūros kūrimo procesas. Modelio koordinatės sudedamos į 2D masyvą - tekstūrą, kuri nusiunčiama į GPU, kad kiekvieną kartą, prieš vykdant paiešką, modelio duomenys būtų perkelti į GPU ir nereikėtų iš naujo siųsti.

Šablono paieška

Šablono paieškos pradžia yra tokia pati kaip ir modelio sudarymo etape. Visų pirma spalvotas paveikslėlis verčiamas į pilkos skalės, pritaikomas Gauso filtras, atliekamas kraštų išskyrimas Canny metodu ir sudaroma paveikslėlio detalumo piramidė.

Šablono paieškos metu vaizdo šablonas slenkamas per ieškomą paveikslėlį ir tikrinama ar šablono kraštai atitinka paveikslėlyje esančio objekto kraštus. Lyginimo operacijos rezultatas - sutampantys vaizdo taškai. Kuo daugiau šablono ir ieškomo paveikslėlio taškų sutampa, tuo labiau tikėtina, kad paslinkto šablono pozicija sutampa su ieškomo paveikslėlyje objekto pozicija. Tokiu būdu slenkant šabloną per visus paveikslėlio taškus, patikrinamos visos galimos pozicijos, kur šablonas galėtų geriausiai atitikti ieškomą objektą.

Kad pasiekti paieškos invariantiškumą objekto pasukimui, modelis yra papildomai pasukamas ir vykdoma ta pati šablono slinkimo ir lyginimo su paveikslėlyje esančiais objekto kraštais schema. Objekto pasukimų skaičius priklauso nuo reikalaujamo tikslumo ir vartotojo nustatyto didžiausio ir mažiausio galimo objekto pasukimo kampo.

Tokį pat principą galima pritaikyti objekto dydžio kitimo kompensavimui ir objekto dydžio radimui.

Kad aprašytas šablono paieškos metodas reikalautų mažiau operacijų, naudojama hierarchinė vaizdo apdorojimo schema. Visų taškų tikrinimas yra vykdomas tik aukščiausiame lygyje, kur sudaromos hipotezės apie galimas modelio buvimo vietas ir pasukimo kampus. Kadangi lyginimas atliekamas aukščiausiame lygyje, kur paveikslėlio ir modelio dydžiai maži, labai sumažėja lyginimo operacijų.

Lyginimo operacijos rezultatas - spalva. Spalvos raudona komponentė gauna reikšmę 1, jei lyginimo rezultatas viršija slenkstį, kitu atveju 0. Žalioje ir mėlynoje komponentėse saugomos šablono pasukimo kampų pradžia ir pabaiga, kuomet šablono lyginimo rezultatas viršija nustatytą slenkstį.

Šiame metode slenkstis parinktas 50 procentų nuo modelio saugomų taškų kiekio. T.y. jei bent 50% taškų atitiks, sistema priims lyginimo rezultatą. Žinoma, šį modelio taškų sutapimo parametrą vartotojas gali valdyti. Tai ypač aktualu, jei yra didelė tikimybė, kad ieškomas objektas gali būti dalinai uždengtas.

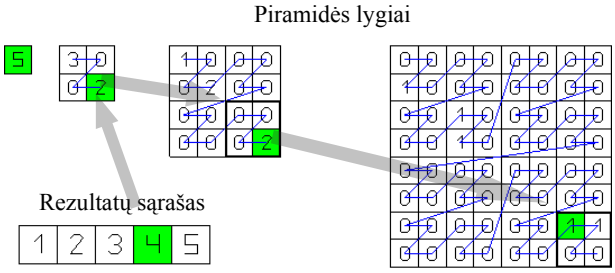
Žemesniame lygyje $I_n(x,y)$ taške yra tikrinama raudonos komponentės reikšmė n-1 lygyje (x/2,y/2) koordinatėje ir šios koordinatės aplinkiniuose taškuose. Jei reikšmė nors viename taške 1, tuomet daroma hipotezė, kad šis (x,y) taškas yra modelio centrinis taškas ir vykdomas šablono lyginimas, kitu atveju taškas

atmetamas. Lyginimas atliekamas tik tose kampų ribose, kurios buvo išsaugotos į žalią ir mėlyną taško komponentes.

Rezultato formavimas

Kadangi šablono lyginimo rezultatai saugomi didelėje tekstūroje, efektyvus ir tvarkingas rezultatų sudėjimas į sąrašą yra būtinas. Su CPU tai būtų daroma įprastu būdu: keliaujama per viso paveikslėlio taškų masyvą, tikrinamas kiekvienas vaizdo taškas ir kiekvienas rastas tinkamas taškas pridedamas į rezultatų sąrašą. Su grafikos procesoriumi tai gali būti padaryta ir paskirstytu (angl. parallel) būdu (Smistad 2010, Ziegler 2006). Šio būdo privalumas ne vien, kad algoritmas gali būti vykdomas paskirstytai, bet ir nereikia atsisiųsti viso paveikslėlio iš GPU į CPU erdvę, kas leidžia sutrumpinti algoritmo vykdymo laiką.

Paieškos rezultato suformavimas (3 pav.) pradedamas piramidės suformavimu nuo šablono lyginimo rezultatų masyvo-paveikslėlio. Pirmas piramidės lygis bus du kartus mažesnis nei šablono paieškos rezultatų masyvas, iš kurio formuojamas pirmas lygis, kurio vaizdo taškuose saugomas rastų šablonų kiekis (aptiktų šablonų paveikslėlyje skaičius). Aukštesniuose lygiuose yra sumuojami keturi kaimyniniai žemesnio lygio taškai ir tęsiama tol, kol piramidės plotis ir aukštis bus lygus vienam. Aukščiausiame piramidės lygyje esanti reikšmė parodo šablono rezultatų kiekį.



3 pav. Rezultatų formavimo proceso pavyzdys.

Rezultato sąrašo užpildymas paremtas Mortono kodu, kuris leidžia patogiai ir efektyviai iš 1D masyvo pereiti į 2D masyvą ir atvirkščiai. Rezultato buferio pildymas yra vykdomas lygiagrečiai, t.y. kiekvienas buferio elementas turi indeksą, pagal kurį vykdoma paieška suformuotoje piramidėje.

Rezultatai

Vaizdo apdorojimo procedūra pagal nagrinėtą literatūrą turėjo būti žymiai greitesnė su GPU nei su CPU. Kadangi objekto parametrų išskyrimo metodą sudėtinga lyginti su kitų autorių atliktais darbais dėl skirtingų sąlygų (kompiuterio, metodo, duomenų), o specialios komercinės bibliotekos sunkiai prieinamos, todėl buvo palygintas populiarus Canny kraštų išskyrimo metodas, kuris sudarytas iš trijų dalių: glodinimas, kraštų išskyrimo operatorius ir ne maksimalių lokalių reikšmių pagal gradiento kryptį atmetimas. Testui atlikti buvo naudojamas nešiojamas kompiuteris su dviejų branduolių Intel 2,26GHz procesoriu ir NVIDIA Quadro NVS 160 vaizdo plokšte. Taip pat buvo testuota su stacionariu kompiuteriu, kuris turi keturių branduolių AMD Phenom 3,2GHz procesorių ir ATI Radeon HD 5550 vaizdo plokštę. Įgyvendintas Canny metodas vykdymui su GPU buvo lyginamas su populiarios atviro kodo OpenCV bibliotekos pateikiamu Canny metodu. Vaizdo apdorojimo testas su GPU buvo vykdomas, kai vienai taško spalvinei komponentei buvo skirta 8 bitai, kitu atveju 32 bitai. 32 bitu formatas yra labai priimtinas, kadangi komponentės reikšmė gali kisti dideliame float formato intervale. Tai leidžia į komponentes saugoti pakankamai dideles reikšmes, kurių saugojimas naudojamas kituose modelio paieškos etapuose. Deja, kaip galima matyti iš 1 lentelės, greitis nukrenta 2-3 kartus lyginant su 8 bitų formatu.

Pagal 1 lentelę galima pastebėti, kad Canny kraštų

1 lentelė. Canny algoritmo atlikimo laikas su CPU naudojant OpenCV bibioteką ir įgyvendintas metodas vykdymui su GPU.

Kompiuteris	Veiksmas	Paveikslėlio dydis 512 x 512			Paveikslėlio dydis 1024 x 1024		
		CPU OpenCV	GPU 8-bit	GPU 32-bit	CPU OpenCV	GPU 8-bit	GPU 32-bit
Intel 2x2,2GHz + Nvidia NVS 160M	Gauso 9x9 glodinimas	10 ms.	1,5 ms.	2,5 ms.	40 ms.	5,5 ms.	9,5 ms.
	Canny (Sobel+nemaksimalių reikšmių atmetimas)	14 ms.	2,1 ms.	3,1 ms.	35 ms.	8,5 ms.	11,5 ms.
	Duomenų perkėlimas	-	2,4 ms.	8,4 ms.	-	11 ms.	32 ms.
	Bendras apdorojimas su duomenų perkėlimu	24 ms.	6 ms.	14 ms.	75 ms.	24 ms.	53 ms.
AMD Phenom 4x3.2GHz + ATI Radeon HD 5550	Bendras apdorojimas su duomenų perkėlimu	**25 ms.**	**0,8 ms.**	2,2 ms.	**94 ms.**	**2,9 ms.**	8,8 ms.

išskyrimas veikia iki 30 kartų greičiau su GPU, nei CPU (OpenCV – 25ms., o pateiktas metodas – 0,8ms., kai paveikslėlio dydis 512x512). Deja, dėl duomenų perkėlimo į GPU juos apdoroti ir apdorojus nuskaityti prarandama daug laiko.

Nešiojamuose kompiuteriuose naudojami grafikos procesoriai dažniausiai neturi daug procesorių, tuo tarpu stacionariuose kompiuteriuose integruojami grafikos procesoriai turi iki kelių tūkstančių procesorių, lyginant su vienetais centrinių procesorių. Dėl šių priežasčių stacionariuose kompiuteriuose vykdomas objekto pozicijos nustatymo algoritmas su GPU veikia daug kartų greičiau lyginant su CPU. Dar pastebimai didesnį greitį galima pasiekti, jei būtų naudojama galingesnė vaizdo plokštė, tuo tarpu geresnis procesorius reikšmingo metodo pagreitėjimo neduos. Taip pat galima pastebėti, kad OpenCV bibliotekos pateikiamo metodo greitis nepriklauso nuo procesorių skaičiaus.

Visas objekto parametrų nustatymo metodas su nešiojamu kompiuteriu trunka apie 50 ms., kai paveikslėlio dydis 512x512 ir 110ms., kai paveikslėlio dydis 512x512. Aprašytas metodas atsparus vaizdo triukšmui, daliniam objekto uždengimui, daliniam kontrasto tarp aplinkos ir objekto nebuvimui. Dažniausiai industriniuose taikymuose objekto dydis nekinta, todėl objekto dydis laikomas pastoviu ir nelabai aktualiu objekto pozicijos radime. Įgyvendintas metodas paveikslėlyje esantį objektą randa vieno taško tikslumu, todėl galima teigti, kad įgyvendintas metodas yra greitas ir tikslus.

Išvados

Šiame darbe buvo aptartas ir įgyvendintas greitas ir tikslus metodas, skirtas vaizdo paveikslėlyje rasti ir išskirti užduotą modelį. Modelio paieškos algoritmo etapai buvo įgyvendinti vykdymui su grafikos procesoriumi. Taip pat buvo palyginti vaizdų apdorojimo greičiai. Kartu buvo nuodugniai aprašytas lygiagretus algoritmas, skirtas paveikslėlyje esančią informaciją tvarkingai sudėti į rezultatų buferį, kuris gali būti vykdomas vaizdo procesoriumi.

Atlikti tyrimai parodė, kad GPU žymiai pagreitina vaizdų apdorojimą ir leidžia greičiau įvykdyti tam tikras užduotis. Palyginus OpenCV bibliotekos Canny objekto kraštų išskyrimo metodą ir šiame darbe pateikiamą metodą, nustatyta, kad įgyvendintas kraštų išskyrimo su GPU metodas veikia iki 30 kartų greičiau.

Literatūra

Borgefors, G. 1988. Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6): 849-865

Borovikov, I. 2009. GPU-acceleration for Surgical Eye Imaging, in *2009 Proceedings of the Fourth SIAM Conference on Mathematics for Industry (MI09), San Francisco, California, USA, 2009*

Hornberg, A. 2006. *Handbook of Machine Vision*. Berlin. 626-654

Luo, Y.; Duraiswami, R. 2008. Canny edge detection on NVIDIA CUDA, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops,* 1-8.

Smistad, E. 2010. *High-speed Marching Cubes using Histogram Pyramids*. [žiūrėta 2011 m. sausio 5d.]. Prieiga per internetą: <http://www.idi.ntnu.no/~janchris/tdt24-f10/slidesets/dyken-smistad17092010.pdf>

Steger, C. 2002. Occlusion, clutter, and illumination invariant object recognition, *International Archives of Photogrammetry and Remote Sensing*, 34(3): 345-350

Strzodka, R.; Ihrke, I.; Magnor, M. 2003. A graphics hardware implementation of the Generalized Hough Transform for fast object recognition, scale, and 3d pose detection, in *Proceedings of IEEE International Conference on Image Analysis and Processing (ICIAP'03)*, 188–193

Treiber, M. 2010. *An Introduction to Object Recognition*. Miunchenas: Springer. 216 p.

Ziegler, G., et al. 2006. *On-the-fly Point Clouds through Histogram Pyramids*. [žiūrėta 2011 m. sausio 5d.]. Prieiga per internetą: <http://www.mpi-inf.mpg.de/~gziegler/gpu_pointlist/paper17_gpu_pointclouds.pdf>

FAST AND ACCURATE OBJECT PARAMETER DETECTION IN MACHINE VISION

Tadas Kazakevičius

Abstract

A real time object position and rotation detection algorithm is proposed. Algorithm is dedicated to be used on the GPU parallel architecture. GLSL language and OpenGL library is used to run and control object position detection algorithm. The proposed method is resistant to varying illumination, noise and partial object occlusion.

Keywords: machine vision, object recognition, image processing on GPU.