Liet. matem. rink, **45***, spec. nr.*, 2005, 411–416

Lygiagretieji algoritmai vaizdų filtravime

Raimondas ČIEGIS, Aleksandr JAKUŠEV (VGTU)
el. paštas: alexj@fm.vtu.lt, rc@fm.vtu.lt

1. Vaizdų glodinimas

Vaizdų glodinimas gali turėti daug taikymų, pavyzdžiui, triukšmų šalinimas arba kraštų detekcija. Tai, svarbu tokiose srityse, kaip kompiuterinė tomografija arba kompiuterinis regėjimas (*machine vision*).

1.1. Matematinis modelis

Labai dažnai vaizdų filtravimui yra naudojamas Gauso filtras. Šio filtro taikymas yra analogiškas tiesinio parabolinio uždavinio sprendimui [7]. Tačiau norint, kad glodinimas veiktų tik triukšmą, o ne vaizdo kraštus, reikia apibrėžti netiesinį difuzijos modelį. Vieną tokį variantą pasiūlė Perona ir Malik, o modifikavo Catte, Lions, Morel ir Coll [2], [4]:

$$\begin{cases} \dfrac{\partial u(X,t)}{\partial t} = \sum\limits_{i=1}^{2} \dfrac{\partial}{\partial x_i}\Big(g\big(\big|\nabla G_\sigma * u(X,t)\big|\big)\dfrac{\partial u(X,t)}{\partial x_i}\Big) + f(u_0-u), \\ \partial_\nu u = 0, \quad (X,t) \in \partial Q \times (0,T], \\ u(X,0) = u_0(X), \quad X \in Q, \end{cases} \tag{1}$$

čia $G_\sigma > 0$ yra atitinkamos tiesinės nestacionarios difuzijos lygties Gryno funkcija. Funkcija f skirta neleisti suglodintam vaizdui labai nutolti nuo pradinio vaizdo (t.y. turime baudos funkciją, pvz. $f(v) = \beta v$).

Difuziniai filtrai išsamiai aprašyti Weickert'o knygoje [6]. Mikulos darbe [3] aptartas bendresnių diferencialinių lygčių dalinėmis išvestinėmis naudojimas vaizdų filtravimui ir analizei.

1.2. Skaitinis modelis

Skaitmeninio vaizdo struktūra leidžia pasirinkti stačiakampį tinklelį, taip pat imsime vienodą diskretųjį laiko žingsnį. Tada, naudodami baigtinių tūrių metodą erdvinių išvestinių aproksimavimui ir Eulerio metodą laiko išvestinės aproksimavimui, gauname tokią išreikštinę diskrčiąją schemą

$$\partial_t U_{ij}^{n+1} = \sum_{\alpha=1}^{2} \partial_{x_\alpha}^{+}\big(a_\alpha(U_{ij}^n)\,\partial_{x_\alpha}^{-} U_{ij}^n\big) + f\big(u_{0,ij} - U_{ij}^n\big). \tag{2}$$

Čia naudoti tokie baigtinių skirtumų operatorių žymėjimai:

$$U_{ij}^n = U(x_{1i}, x_{2j}, t^n), \quad (x_{1i}, x_{2j}, t^n) \in \omega_h \times \omega_\tau ,$$

$$\partial_t U_{ij}^n = \frac{U_{ij}^n - U_{i,j}^{n-1}}{\tau}, \quad \partial_{x_1}^- U_{ij}^n = \frac{U_{ij}^n - U_{i-1,j}^n}{h}, \quad \partial_{x_1}^+ U_{ij}^n = \frac{U_{i+1,j}^n - U_{ij}^n}{h},$$

$$\partial_{x_2}^- U_{ij}^n = \frac{U_{ij}^n - U_{i,j-1}^n}{h}, \quad \partial_{x_2}^+ U_{ij}^n = \frac{U_{i,j+1}^n - U_{ij}^n}{h}.$$

Difuzijos koeficientą aproksimuojame funkcionalu a_α imdami standartinę trapecijų vidurkinimo formulę (žr. [2]).

Nesunku patikrinti, kad išreikštinė schema (2) stabili tik kai $\tau \leqslant ch^2$. Norėdami turėti nesąlygiškai stabilią schemą, galime naudoti tokią tiesinę neišreikštinę aproksimaciją:

$$\partial_t U_{ij}^{n+1} = \sum_{\alpha=1}^{2} \partial_{x_\alpha}^+ \left(a_\alpha(U_{ij}^n)\, \partial_{x_\alpha}^- U_{ij}^{n+1} \right) + f\left(u_{0,ij} - U_{ij}^n \right). \tag{3}$$

2. Algoritmų lygiagretinimas

Pasaulyje egzistuoja nemažai lygiagrečiųjų algoritmų realizavimo priemonių ir standartų. Čia apžvelgsim tuos pavyzdžius, kurie gali būti naudojami taikant duomenų lygiagretumo algoritmus:

MPI (*Message Passing Interface*) yra C/C++ ir Fortran'o bibliotekų standartas [5]. Jis yra plačiai paplitęs, tačiau gana sudėtingas, todėl lygiagretinti programas jo pagalba yra pakankamai sunku, reikia įgyti specifinių žinių.

HPF (*High Performance Fortran*) yra programavimo kalbos FORTRAN standarto išplėtimas. HPF geriausiai tinka uždaviniams, kuriems būdingas duomenų lygiagretumas. Iš pradžių sukuriamas nuoseklus algoritmas, jis programuojamas įprastiniu Fortranu, laikantis tam tikrų taisyklių (viena iš jų – naudotis Fortrano masyvų operacijomis). Programos lygiagretinimas atliekamas pateikiant kelias nesudėtingas nuorodas, kaip paskirstyti užduotis/duomenis tarp procesų. Po to gautasis kodas kompiliuojamas HPF kompiliatoriumi. Palyginus su MPI, tai yra žymiai paprastesnis procesas. Deja, Fortran kalbos populiarumas mažėja, be to, HPF – tai atskiras kompiliatorius, kurio vystymas reikalauja papildomų sąnaudų.

OpenMP (*Open Multi Programming*) yra C kalbos išplėtimas, skirtas lygiagretiesiams kompiuteriams su bendrąja atmintimi. Programos lygiagretinimas gali būti pasiektas kelių direktyvų pagalba.

UPC (*Unified Parallel C*) yra C kalbos išplėtimas [1], skirtas SPMD tipo programų lygiagretinimui. Jis yra sudėtingesnis, nei HPF.

3. ParSol – lygiagrečiųjų masyvų paketas

ParSol lygiagrečiųjų masyvų paketas buvo sukurtas, norint realizuoti C++ kalboje HPF funkcionalumą. Viena iš priežasčių, kodėl C++ kalboje tokio analogo nėra, yra ribotas

gimtųjų C/C++ masyvų funkcionalumas. Tačiau C++ yra objektiškai orientuota kalba, taigi galime kurti naujas norimo funkcionalumo klases.

ParSol realizuotas C++ kalba, esminiai remtasi OOP technologijomis. Buvo naudojamos tik standartinės C++ kalbos galimybės. Duomenų apsikeitimas realizuotas naudojantis MPI 1.1 standartu [5]. Sukurta labai portabili masyvų–objektų biblioteka, ji jau išbandytas šiose operacinėse terpėse (ir kompiuteriuose): a) MS-Windows terpė, MS Visual C++ 6.0 kompiliatorius, MPI realizacija MPICH; b) Linux terpė, gcc kompiliatorius, MPI realizacija LAM-MPI (VGTU kompiuterių klasteris Vilkas); c) IBM SP4 superkompiuteris, VisualAge C++ kompiliatorius, IBM MPI realizacija (`http://www.cineca.it`).

ParSol klasių diagrama pavaizduota 1 pav.

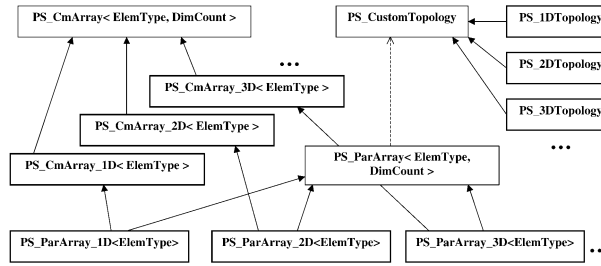
Pagrindiniai bibliotekos elementai yra tokie:

Nuosekliųjų masyvų klasės. Tai klasės, kurios turi būti naudojamos vietoj standartinių C/C++ masyvų. Skaitiniame modeliavime naujieji ParSol masyvai yra patogesni, nes jiems realizuota daug specialių metodų (maksimalaus elemento radimas, skaliarinės sandaugos bei įvairių normų skaičiavimas, veiksmai su matricų operatoriais). Naudojant tik nuosekliųjų masyvų klases, MPI biblioteka nėra reikalinga.

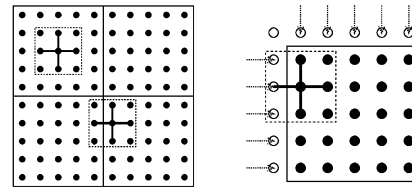
Lygiagrečiųjų masyvų klasės. Jos naudojamos vietoj nuosekliųjų masyvų, lygiagretinant programą. Siekiant supaprastinti šį algoritmo generavimo etapą, visos lygiagrečiosios klasės yra atitinkamų nuosekliųjų klasių palikuonys, o duomenų pasikeitimo algoritmo pagrindiniai metodai, nustatantys kievieno proceso kaimynus ir atliekantys duomenų apsikeitimą, realizuoti kaip atskira klasė. 2 pav. parodytas lygiagretinimo principas: *a* dalyje pavaizduotas nuoseklusis masyvas, o *b* dalyje matome duomenų sritį, priklausančią vienam iš lygiagrečiųjų procesų.

Topologijos klasės. Šių klasių paskirtis yra tokia pat, kaip ir HPF direktyvų, nurodančių virtualiojo kompiuterio tinklo topologiją. Kadangi mes kuriame ne naują C++ kompiliatorių, o biblioteką, šiam tikslui irgi naudojamos specialios klasės.

Šablonų klasės. Jos skirtos nurodyti kokia lokalia informacija keičiasi kaimynai procesai (priminsime, kad HPF tokia analizė atliekama automatiškai). Pavyzdžiui, baigtinių skirtumų metodu sprendžiant antrosios eilės diferencialines lygtis, dažnai



1 pav. ParSol bibliotekos klasių diagrama.



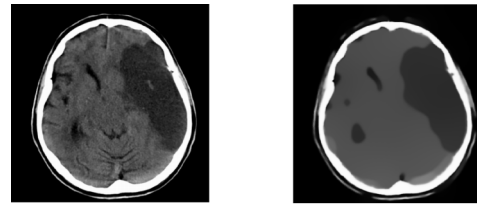
2 pav. ParSol klasių veikimo principas.

naudojamas penkių tinklo taškų šablonas (2 pav., stora linija). Skaičiuojant naujos iteracijos masyvo elemento reikšmę, reikia žinoti jo reikšmę prieš tai buvusioje iteracijoje, bei jo kaimynų iš viršaus, apačios, kairės ir dešinės reikšmes.

4. ParSol taikymai vaizdų filtravimui

Naudojant ParSol buvo išlygiagretinti išreikštinis ir neišreikštinis netiesinės difuzijos algoritmai. Skaičiavimai atlikti VGTU klasteryje Vilkas bei CINECA centro Italijoje IBM SP4 kompiuteryje. Žemiau pateikti skaičiavimo rezultatai.

3 pav. pavaizduotas difuzinio filtro taikymo pavyzdys: *a* dalyje pateiktas pradinis vaizdas, o *b* dalyje pavaizduotas vaizdas, gautas po 100 iteracijų. Matome, kad filtras naikina triukšmus, nepažeisdamas vaizdo kontūrų.



3 pav. Tomografinis žmogaus insulto vaizdas prieš ir po skaitmeninio filtravimo.

1 lentelė. Išreikštinio algoritmo (2) pagreitėjimas ir efektyvumas PC klasteryje

p	$S_p(160)$	$E_p(160)$	$S_p(240)$	$E_p(240)$	$S_p(320)$	$E_p(320)$
2	1.56	0.780	1.76	0.880	1.87	0.934
4	2.36	0.590	3.00	0.750	3.45	0.862
6	2.78	0.463	3.93	0.655	4.77	0.795
8	2.95	0.369	4.69	0.585	5.88	0.735
9	3.16	0.351	5.04	0.560	6.28	0.698
11	3.33	0.303	5.50	0.500	7.09	0.644
12	3.35	0.279	5.64	0.470	7.47	0.623
15	3.39	0.226	6.38	0.425	8.56	0.571

1 lentelėje pateikti lygiagrečiojo išreikštinio algoritmo (2) pagreitėjimo ir efektyvumo koeficientų reikšmės, kai skaičiavimai atlikti asmeninių kompiuterių klasteryje, 2 lentelėje pateikti analogiški duomenys, gauti naudojant SP4 superkompiuterį.

Realizuojant lygiagretųjį neišreikštinį algoritmą svarbiausiu efektyviai spręsti tiesinių lygčių sistemas. Naudojome jungtinių gradientų algoritmą. Jo lygiagrečiosios

2 lentelė. Išreikštinio algoritmo (2) pagreitėjimas ir efektyvumas SP4 kompiuteryje

p	$S_p(80)$	$E_p(80)$	$S_p(160)$	$E_p(160)$	$S_p(320)$	$E_p(320)$
2	1.975	0.988	1.984	0.992	2.004	1.002
3	2.794	0.931	2.950	0.985	2.970	0.990
4	3.741	0.935	3.928	0.982	3.986	0.996
6	5.168	0.861	5.463	0.910	5.916	0.986
8	6.766	0.846	7.293	0.911	7.831	0.979
9	6.784	0.754	7.604	0.845	8.467	0.941
12	8.701	0.725	10.19	0.849	11.216	0.934
16	10.84	0.677	12.75	0.797	15.041	0.940
24	14.18	0.591	18.24	0.760	21.961	0.915

3 lentelė
CPU laikai $T_p(N)$ sprendžiant uždavinį tiesiniu neišreikštiniu algoritmu SP4 kompiuteriu (3), SP4

p	$T_p(160)$	$T_p(320)$	$T_p(480)$
1	64.97	241.4	281.9
2	26.82	86.71	118.5
4	12.89	40.37	63.94
6	9.24	26.91	42.84
8	7.59	21.37	32.44
16	4.83	10.30	16.44

4 lentelė. 3D Puasono lygtis, naudojant JG metodą ir 7 taškų šabloną (PC klasteris)

Procesai	Iteracijos	Dydis	Laikas	S_p	E_p
1	188	100	24.099		
2	188	100	13.224	1.822	0.911
4	188	100	6.648	3.625	0.906
8	188	100	4.034	5.974	0.747
1	350	200	366.542		
2	350	200	185.512	1.976	0.98
4	350	200	94.994	3.859	0.965
8	350	200	51.579	7.106	0.888
4	453	300	407.570		
8	453	300	215.597		

versijos efektyvumas esminiai priklauso nuo kolektyvinių operacijų (dviejų vektorių skaliarinės sandaugos ir vektoriaus normų) skaičiavimo sąnaudų. 3 lentelėje pateikti šio algoritmo vykdymo SP4 kompiuteryje laikai.

Lygiagrečiojo jungtinių gradientų metodo efektyvumą nagrinėjome ir asmeninių kompiuterių klateryje. Sprendėme modelinį trimatį Puasono uždavinį, kurį aproksimavome baigtinių skirtumų schema naudodami centrinių baigtinių skirtumų operatorius ir septynių tinklo taškų šabloną. Skaičiavimo rezultatai pateikti 4 lentelėje.

Iš pateiktų rezultatų aiškiai matome didelę duomenų perdavimo sąnaudų įtaką PC klasteryje, tai mažina lygiagrečiojo algoritmo efektyvumą nedideliam uždaviniui, didėjant procesų skaičiui. Tačiau, kai uždavinio apimtis didėja, arba esant mažiems komunikacijos kaštams, mes matome labai aukštą lygiagrečiųjų algoritmų, gautų naudojant ParSol, efektyvumą.

Literatūra

1. W. Carlson, J. Draper, D. Culler, K. Yelick, E. Brooks, K. Warren, *Introduction to UPC and Language Specification*, CCS-TR-99-157, IDA Center for Computing Sciences (1999).
2. F. Catte, P.L. Lions, J.M. Morel, T. Coll, Image selective smoothing and edge detection by nonlinear diffusion, *SIAM J. Numer. Anal.*, **29**(1), 182–193 (1992).
3. K. Mikula, N. Ramorosy, Semi-implicit finite volume scheme for solving nonlinear diffusion equations in image processing, *Numer. Math.*, **89**, 561–590 (2001).
4. P. Perona, J. Malik, Scale space and edge detection using anisotropic diffusion, in: *Proc. IEEE Computer Society Workshop on Computer Vision* (1987).
5. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, *MPI*: *the Complete Reference*, vol. 1, The MIT Press (1998).
6. J. Weickert, *Anisotropic Diffusion in Computer Vision*, Teubner, Stuttgart (1998).
7. A.P. Witkin, Scale-space filtering, in: *Proc. Eight Internat. Conference on Artificial Inteligence*, vol. 2 (1983), pp. 1019–1022.

SUMMARY

R. Čiegis, A. Jakušev. Parallel algorithms in image filtering

In this paper, image-filtering parallelization is described. A description of mathematical problem is given, and general parallelization tools suitable in this case are overviewed. Then the ParSol parallel array package used for parallelization is described, together with computational results of its application.

Keywords: parallel algorithms, parallelization tools, modelling, image filtering.