Lietuvos matematikos rinkinys
Lietuvos matematikų draugijos darbai, ser. B
54 t., 2013, 178–183

ISSN 0132-2818
www.mii.lt/LMR/

Daugiamačių skaitinių masyvų vizualizavimas kietakūniais objektais

Olegas Ramašauskas

Klaipėdos universitetas, Lietuvos verslo kolegija
H. Manto g. 84, LT-92291 Klaipėda
E. paštas: olegas@ik.ku.lt

Santrauka. Straipsnis yra skirtas matematikos ir informatikos studijų dalykų supratimo pagilinimui trimatės kompiuterinės grafikos aplinkoje. Darbe aprašytas daugiamačių skaitinių masyvų vizualizavimas naudojant objektinio programavimo priemones. Metodas leidžia iš daugiamačiuose skaitiniuose masyvuose saugomų duomenų simuliuoti virtualiosios realybės produktus. Parodyta kita galimybė modeliuoti erdvinius vizualinius objektus taikant naujus skaičiuojamosios matematikos programų paketus.

Raktiniai žodžiai: skaitmeninis vaizdas, skaitinis masyvas, programavimas.

Įvadas

Erdvės suvokimo gebėjimai yra svarbūs daugelyje disciplinų, įskaitant, bet neapsiribojant, inžinerijos, architektūros, biomedicinos moksluose, robotikoje ir geografinėse informacijos sistemose [1, 8]. Erdvinė informacija apibrėžia fizinę objektų sritį ir ryšį tarp objektų. Erdvinės informacijos pramonė yra platesnė informacinių technologijų sektoriaus specializuota dalis, kuri turi ryšį su planavimu, natūralių išteklių valdymu, inžinerija ir sveikatos paslaugomis [7]. Gal būt, šią pramonę galima būtų vadinti erdvinės informacijos gavyba pagal analogiją su duomenų gavyba (laikomą atpažinimo teorijos dalimi), kurioje reikšmingų objektų paieška atliekama apdorojant taškus daugiamatėje erdvėje ir atskiriant, klasifikuojant jų klasterius.

Antra vertus, populiarių matematinių programų (Maple, Mathematica, Scientific Notebook, Octave, R ir kitų) pagalba galima atlikti daugelį tradicinių užduočių, pavyzdžiui, išspręsti lygčių sistemas, nagrinėti diferencialines lygtis, pavaizduoti funkcijas, sukurti įvairius daugiamačius modelius ir ištirti jų savybes. Tokias programas besimokantiems įsisavinti paprasčiau negu tradicines programavimo kalbas [2], tačiau kai kuriuos mokslinius ir taikomuosius erdvinius uždavinius lengviau išspręsti specializuotų programų paketų priemonėmis, pavyzdžiui, National Instruments LabVIEW Tools Network, System Design Software, Signal Processing, Analysis and Connectivity, kuriuose naujausi 3D regos algoritmai įtraukti į NI Vision plėtros modulį [5], taip pat SolidWorks, Inventor ir įvairiomis CAGD (angl. *Computer Aided Geometric Design*) [3] programomis, paplitusiomis industrinėse sferose ir informatikos inžinerijos moksluose. Nagrinėdami elementarių vienetinių objektų programavimo aspektus matricų metodais darbe [2] mažiau dėmesio skyrėme sudėtingesnių objektų kūrimui.

Šiame darbe vizualizuosime daugiamačius, vadinamus CAGD kietakūnius objektus (angl. *solids*), turinčius vidinę struktūrą: tūrį, masę, erdvės pikselius. Tikslui pasiekti

naudosime daugiamačių skaitinių masyvų išraiškas, įvairias jų transformacijas bei Matlab notifikaciją.

1 Daugiamačių išraiškos priemonių apžvalga

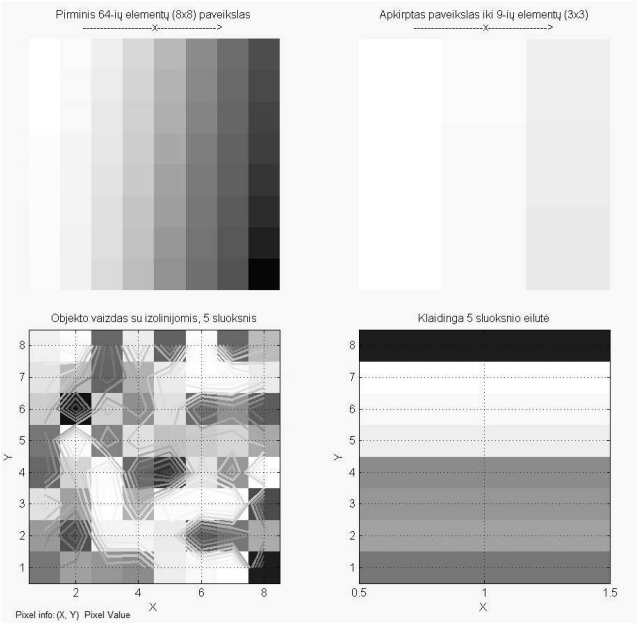
Daugiamačiai objektai dažnai aprašomi daugiamačiais skaitiniais masyvais, kurie Matlab sistemoje interpretuojami matriciniu pavidalu [2, 4]. Nepriklausomai nuo aprašomų objektų matiškumo, jų suvokimo erdvę IP^2 įprasta laikyti plokščia [3], t. y. dvimate (foto juostelė, CMOS jutiklis, kompiuterio ekranas). Tokia laikytina ir akies tinklainė, nors ji yra gerokai išgaubta, tačiau to išgaubtumo sukeliamas projekcines paklaidas kompensuoja smegenų neuronų tinklas, o binokulinė rega sukuria teisingą trimačio pasaulio scenos vaizdą. Kompiuteriniame vaizdų rekonstravime labiausiai paplitęs klasikinis baigtiniu atstumu nutolusios kameros modelis, kuria trimačių objektų taškai projektuojami į plokštumą. Visos tiesės, nubrėžtos per trimačių objektų taškus ir jų projekcijas plokštumoje, susikerta viename taške. Gautasis taškas vadinamas kameros centru, o plokštuma, į kurią projektuojami trimačių objektų taškai, vadinama vaizdo plokštuma arba židinio plokštuma. Kitaip tariant, IP^3 erdvinis taškas su koordinatėmis $P(x, y, z) = (X, Y, Z)^T$, yra projektuojamas į vaizdo plokštumą ten, kur jungianti tašką $P(x, y, z)$ ir kameros centrą tiesė kertasi su vaizdo plokštuma [3, 8]. Vaizdo atpažinimo algoritmus realizuojanti programa lygina kiekvieno pirminio paveikslo pikselio reikšmę su kiekviena tikslo paveikslo pikselio reikšme ir suskaičiuoja sutapimus. Kadangi algoritmų pikselių palyginimui yra ne vienas, sutapimų ir nesutapimų santykiai taip pat gali būti skirtingi šalia viso to, pritaikyta fuzifikacija, glodinimas ar grubinimas, todėl vieningo metodo skaitmeninio vaizdo atpažinimui ir įvertinimui kol kas nėra sukurta. Todėl aiškiai apibrėžus uždavinį įprasta pritaikyti vizualizavimo algoritmus, naudojant atitinkamus programų paketus, kurių pasirinkimas taip pat teikia nemažai laisvės. Tarkime, naudoti Sobelio (Sobel), Valiso (Wallis), Freičeno (Frei-Chen), Kiršo (Kirsch), Hukelio (Hueckel), Noblio (Noble), Šitomasio (Shi-Tomasi), Kenio (Kenny) ir kitų mokslininkų vardais pavadintus vaizdo elementų, kraštų ar būdingųjų taškų aptikimo algoritmus [4, 6, 7] arba kurti savus.

2 Daugiamačių skaitinių masyvų interpretavimas

Vizualinius objektus programuosime Matlab 2013a x64 aplinkoje. Joje įdiegtos vaizdų apdorojimo priemonės (Computer Vision System Toolbox, Simulink 3D Animation, Image Acquisition Toolbox ir kitos), sąsajų kūrimo ir autonominių programų kompiliavimo priemonės. Tai leidžia atsisakyti gremėzdiškų C++ ar C# įrankių su OpenCV ir kitomis grafinėmis bibliotekomis. Sukuriame išvedimo aplinką, automatiškai išvalomą kiekvieno kartojimo metu.

```
clc; clear; close all force; % išsivalome darbo aplinką,
mon = get(0,'ScreenSize');   % ruošiame pilkšvą monitoringo langą;
f1 = figure('Name','Skaitmeniniai vaizdai',...
    'ToolBar','none','Position',[30 50 mon(3)/1.78 mon(4)/1.12]);
set(0,'CurrentFigure',f1); % paruošiame išvedimo langų ašis,
set(gcf,'Renderer','painters','Color',[.9 .9 .9]);
colormap('gray'); % pateiktims tiktų ryškesnė paletė, pvz. 'hot'.
```

Patikriname $f1$ nuostatas sukarpę dvimačius elementus polangiuose $h1$ ir $h2$.



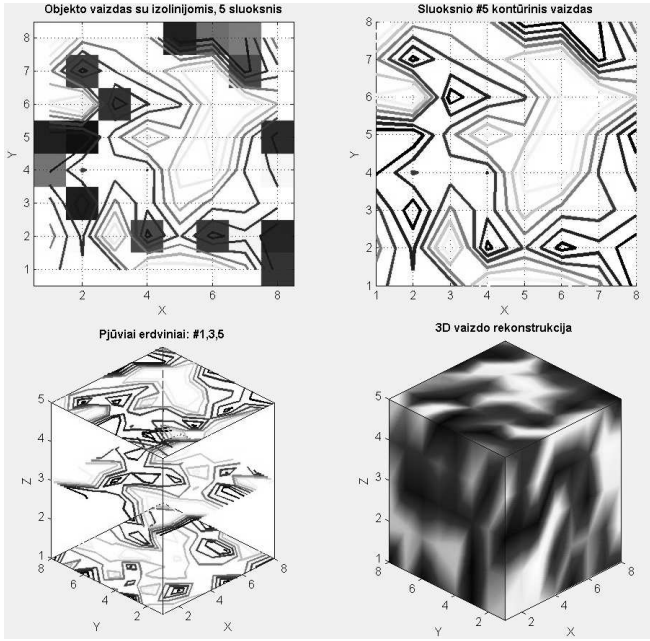
1 pav. Viršuje 2D karpiniai, apačioje – 3D vaizdų horizontalūs pjūviai.

```
y = 1 - linspace(0,1,64);   % 64-ių realiųjų skaičių [0 1] eilutė;
I = reshape(y,[8 8])        % eilutė sukarpoma į 8x8 matricą (64 el.);
r = 1; c = 1; m = 3; n = 3; % (r,c) = 1, reikės atmesti iš (m,n) = 3;
J = imcrop(I,[r c m-1 n-1]) % apkarpoma [Xmin Ymin Plotis Aukštis];
h1 = subplot(1,2,1,'Parent',f1), imshow(I), title({'...';'--x-->'}); % titulai sutrumpinti;
h2 = subplot(1,2,2,'Parent',f1), imshow(J), title({'...';'--x-->'}); % titulai sutrumpinti.
```

Sekančiuose polangiuose ($h3, 4$) generuojamas nedidelis erdvinis objektas, kurio vaizdo failo formatą sudaro elementų spalvų bei $P(x, y, z)$ koordinačių masyvas. Kadangi tokio masyvo matiškumas lygus 4, o pavaizdavimui geriau naudoti trimatį pavidalą, tai teks atlikti erdvinę transformaciją ir sumažinti matiškumą iki 3-jų. Matlab priemonėmis tai galima atlikti keliais būdais, pavyzdžiui, funkcijos *squeeze*(D) arba *shiftdim*(D, n) pagalba, čia D – daugiamatis skaitmeninis vaizdo masyvas (vaizdo matrica), dažnai – perteklinis; n – postūmio (ofseto) žingsnis per stulpelius ar eilutes (aiškiai matyti analogija su dvejetainių skaičių bitų postūmiu kompiuterio registrų ląstelėse, kai programuojama, pavyzdžiui, asembleriu). Pirmiausia užsiduodame, kad mūsų objektas $D(V)$ bus stačiakampis gretasienis kūnas, kurio pagrindą sudarys $8 \times 8 = 64$ erdviniai pikseliai, atsitiktinai sudėlioti penkiais sluoksniais, taigi, iš viso jis turės 320 elementų:

```
Mx = 8; My = 8; % objekto ilgio ir pločio koordinatės (X,Y);
Mo = 1; Mz = 5; % objekto aukščio pradinė ir galinė koordinatės (Z1,2);
V = [My Mx Mo Mz]; % vaizdo matricos parametrus aprašantis vektorius;
vidpp = round(V(4)/2); % objekto vidurinio pjūvio numerio radimas;
kiekis = 10;    % pjūvio izolinijų skaičiaus nustatymo parametras;
rng('shuffle'); % atsitiktinių skaičių generatoriaus aktyvatorius;
D = round(255*rand(V)) % išvedame objekto matricinį pavidalą.
```

Keisdami parametrus polangiuose galime stebėti programos elgesį (1 pav.)



2 pav. Kietakūnio pjūvių vaizdai, kontūrinė ir erdvinė rekonstrukcija.

```
A = squeeze(D) % transformacija iš 4D į 3D formatą, pašalinant vienetinius masyvus;
h3 = subplot(2,2,3,'Parent',f1), % (1-asis būdas);
image(A(:,:,Mz)) % parodome paskutinįjį sudėtinio vaizdo sluoksnį;
axis xy tight, xlabel('X'), ylabel('Y'), grid on, daspect('auto');
x = xlim; y = ylim;
pvz3 = contourslice(A,[],[],round(Mz),kiekis); % papildomai uždedamas kontūrinis vaizdas;
title(['Objekto vaizdas su izolinijomis, ',int2str(Mz),' sluoksnis']);
set(pvz3,'LineWidth',2), box on, xlim(x), ylim(y), daspect('auto');
```

Pirmame paveiksle matyti, kaip kuriamas atitinkamo pjūvio pikselinis paveikslas ir ant jo uždedami vaizdo intensyvumo skiriamųjų ribų kontūrai. Toks sprendimas gali būti naudingas atliekant vaizduojamų objektų atpažinimo procedūras, projektuojant GIS aplikacijas ar atliekant skenuotų vaizdų tyrimus.

```
B = shiftdim(D,1) % Vaizdavimas su ofsetavimu (pirmasis metodas);
h4 = subplot(2,2,4,'Parent',f1);
image(B(:,:,Mz))  % parodome paskutinįjį sudėtinio vaizdo sluoksnį
title(['Klaidinga ' int2str(Mz),' sluoksnio eilutė']);
axis xy tight, xlabel('X'), ylabel('Y'), grid on, daspect('auto'); impixelinfo;
```

Reikia pastebėti, kad atlikus paprastą ofsetavimą $h4$ polangyje matoma tik viena vaizdo eilutė ir ta pati su klaidomis, todėl po čia pateikto algoritmo žingsnio $f1(B \to h4)$ programą modifikuojame, pašaliname nereikalingas funkcijas, spalvinę paletę pasikeičiame į gražesnę ir pertvarkome polangius. Naujajame lange (2 pav.) į polangius $h1, 2$ nukreipiame A objekto masyvo plokščiąjį srautą, o polangiuose $h3, 4$ generuojame $A(V)$ 3D pjūvius ir rekonstruojame visą objektą:

```
h3 = subplot(2,2,3,'Parent',f1); % Trimačiai pjūviai (antrasis metodas);
pav3 = contourslice(A,[],[],[V(3),vidpp,V(4)],detal);
```

```
box on, view(-45,30), set(pav3,'LineWidth',1),
axis(h3,'vis3d','tight'), xlabel('X'), ylabel('Y'), zlabel('Z'),
daspect([1,1,0.4]) % aplikatės ištempimo koeficientas 1/0,4=2,5;
```

čia 3D kietakūnio rekonstrukcija iš anksčiau suformuoto A daugiamačio masyvo:

```
h4 = subplot(2,2,4,'Parent',f1);
As = smooth3(A); hiso = patch(isosurface(As, 2),'EdgeColor','none');
hcap = patch(isocaps(A, 2),'FaceColor','interp','EdgeColor','none');
box on, view(-45,30), daspect([1,1,0.4]),
axis(h4,'vis3d','tight'), xlabel('X'), ylabel('Y'), zlabel('Z');
lightangle(-20,40), lighting phong, isonormals(As, hiso);
set(hcap,'AmbientStrength',0.6);
set(hiso,'SpecularColorReflectance',0,'SpecularExponent',50); % scenarijaus pabaiga.
```

Atliktas darbas patvirtina, kad naudojantis aukšto lygio objektinėmis Matlab programavimo priemonėmis nesunku kurti ir įrašyti spalvotus, vienspalvius ar nespalvotus skaitmeninius vaizdus, neaprašinėjant atskirų vaizdo elementų. Programų paketas turi būtiniausias sąsajos kūrimo galimybes, grafinio programavimo konstrukcijas bei matematinį aparatą, todėl nesunku išmokti jį naudoti. Rezultatai gali būti panaudoti ne tik skaitinio eksperimento duomenų apdorojimo ir/arba vizualizavimo pratyboms, bet ir kitų skaičiuojamųjų dalykų studijoms tiek viduriniojo, tiek aukštojo mokslo įstaigose.

3 Išvados

Matlab programų paketo naudojimas trimatės kompiuterinės grafikos aplinkoje leidžia: 1) pagilinti matematikos ir informatikos studijų dalykų supratimą, 2) geriau suvokti skaitmeninio vaizdo esmę, 3) įsisavinti naujas virtualiosios realybės kūrimo priemones.

Trimačių objektų rekonstrukcijos iš daugiamačių duomenų masyvų gali pasitarnauti projektuojant programines priemones biomedicininių vaizdų analizei, jūros dugno mozaikų tyrimams povandeninių foto robotų pagalba ir kitose mokslo srityse, susijusiose su skaitmeninio vaizdo panaudojimu.

Literatūra

[1] G.R. Bertoline, N. Hartman and N. Adamo-Villani. *Computer-aided Design, Computer-aided Engineering and Visualization*. Springer-Verlag, Berlin, 2009, Ch. 37, pp. 639-651.

[2] R. Biršky tė, ir O. Ramašauskas. Skaitmeninio vaizdo elementų programavimo aspektai. *Liet. mat. rink. LMD darbai, ser. B*, **52**:145–150, 2012. ISSN 0132-2818, adresas internete: http://www.mii.lt/LMR.

[3] G. Farin. *NURBS from Projective Geometry to Practical Use*. AK Peters Ltd., MA, Natick, 1999.

[4] *Inspired Learning. Learn with MATLAB and Simulink. Teach with MATLAB and Simulink*. MathWorks, 2013. Available from Internet: http://www.mathworks.se/academia.

[5] *National Instruments. LabVIEW System Design Software. Products and Services*. LabVIEW, 2013. Available from Internet: http://www.ni.com/labview.

[6] A. Serackis. *Vaizdo rekonstravimo technologijos baltymų pėdsakams parametrizuoti*. Daktarinė disertacija. VGTU, Vilnius, 2008.

[7] L. Stabingienė. *Vaizdų analizė naudojant Bajeso diskriminantines funkcijas*. Daktarinė disertacija. VU MII, Vilnius, 2012.

[8] J. Yue. Spatial visualization by realistic 3D views. *Eng. Design Graph. J.*, **72**(1):28–38, 2008. Available from Internet: http://www.edgj.org/index.php/EDGJ/issue/archive. ISSN 1949-9167

SUMMARY

Visualization of multidimensional numerical arrays of solid-state objects
O. Ramašauskas

This article is intended for math and computer science subjects study deepening and understanding of multidimensional computer graphics environment. The paper describes multidimensional numeric arrays visualization using object-oriented programming tools. The method allows the multidimensional numeric arrays stored data to simulate virtual reality products which, in turn, can become the reference now intensively developed new embedded system components. Shown an another option to simulate the multidimensional visual objects that may be transformed from capturing or graphically present on the computer screen using a new computational mathematics packages for solving advanced computer graphics tasks. It seems comfortable to use new estimated mathematics programs and packages, which already are enriched with digital image processing tools, allowing create a various objects of the real world.

Keywords: Digital image, digital array, programming.