KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Inga Barušauskaitė

GEOMETRIŠKAI DEFORMUOTŲ VAIZDŲ ATKŪRIMO PROCEDŪROS SUDARYMAS IR TYRIMAS

Magistro darbas

Vadovas
prof. J. Valantinas

KAUNAS, 2014

KAUNO TECHNOLOGIJOS UNIVERSITETAS

MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2014 06 02

GEOMETRIŠKAI DEFORMUOTŲ VAIZDŲ ATKŪRIMO PROCEDŪROS SUDARYMAS IR TYRIMAS

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
prof. J. Valantinas
2014 06 01

Recenzentas
prof. dr. V. Jusas
2014 06 01

Atliko
FMMM-2 gr. stud.
I. Barušauskaitė
2014 05 30

KAUNAS, 2014

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Juozas Augutis, profesorius (VDU)

Sekretorius: Eimutis Valakevičius, profesorius (KTU)

Nariai:

Jonas Valantinas, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Kristina Šutienė, docentė (KTU)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., direktoriaus pavaduotojas (UAB „Danet Baltic“)

Barušauskaitė I. Geometriškai deformuotų vaizdų atstatymo procedūros sudarymas ir tyrimas: Taikomosios matematikos magistro baigiamasis darbas / mokslinis vadovas prof. J. Valantinas; Kauno technologijos universitetas, Matematikos ir gamtos mokslų fakultetas, Taikomosios matematikos katedra. – Kaunas, 2014. – 59 p.

SANTRAUKA

Darbo tema aktuali foto nuotraukų analizei, siekiant atkurti pirmines vaizdo formas. Šiame darbe nagrinėjamas geometriškai deformuotas fotografuotas vaizdas, pasižymintis gardeline struktūra. Dažniausiai fotografuojamas vaizdas minimaliai iškraipomas (deformuojamas). Ši savybė labiausiai pajuntama taisyklingas formas turintiems geometriniams objektams – kvadratams, apskritimams, tiesėms. Šio darbo tikslas sudaryti gardeline struktūra pasižyminčių, geometriškai deformuotų, vaizdų atkūrimo procedūrą; patikrinti geometrinių transformacijų veikimą; pasiūlyti optimalų tiriamo Sudoku vaizdo atkūrimo metodą ir vizualiai įvertinti atkurto vaizdo kokybę.

Darbe pateikiama vaizdo atkūrimo schema. Darbe išsamiai pateikti kiekvieno atkūrimo procedūros etapo metodai, jų matematinės išraiškos. Taip pat, pateikiami tarpiniai vaizdo apdorojimo rezultatai. Tam tikslui sudaryta eksperimentiniams tyrimams skirta geometriškai deformuotų specializuotų vaizdų bazė.

Remiantis gautais rezultatais, formuluojamos išvados apie geometriškai deformuoto vaizdo atkūrimo procedūrą, atskirus jos etapuose naudojamus metodus. Bet to, pateikiama su darbu susijusi literatūra.

Darbe gautais rezultatais paremtas pranešimas „Apie geometriškai deformuotų gardelinių vaizdų atkūrimą“. Pranešimas perskaitytas XII-oje taikomosios matematikos konferencijoje (KTU, 2014). Pranešimo tekstas pateikiamas priede Nr. 1.

Barušauskaitė I. Development and Analysis of Restoring Procedures for Geometrically Deformed Images: Master's Thesis in applied mathematics / Supervisor Prof. J. Valantinas; Department of Applied mathematics, Faculty of Mathematics and Natural Sciences, Kaunas University of Technology. – Kaunas, 2014. – 59 p.

SUMMARY

The topic of the paper is important for the analysis of photo images when aiming to restore the initial form of an image. This paper examines a geometrically deformed photographed image, characterized by the lattice structure. Generally, the photographed image is minimally distorted (deformed). This feature is more common for regular geometric objects, such as squares, circles and straight lines. This paper aims to develop restoration procedure of geometrically deformed images, characterized by the lattice structure; to test the functioning of geometric transformations; to suggest the optimal method of image restoration, and to visually evaluate the quality of restored images.

The paper presents an image restoration scheme. Detailed methods of each stage of the restoration procedure and their mathematical expressions are comprehensively outlined in the present work. Intermediate results of image processing are also provided. For this purpose, a specialized base of geometrically distorted images has been created for experimental studies

Conclusions about restoration procedure of geometrically deformed images and separate stages of the employed methods are drawn based on the obtained results. Moreover, work-related literature is presented in this paper.

A report on "Restoration of Geometrically Deformed Lattice Images" is based on the results of the research. The report was presented at the Twelfth Conference of Applied Mathematics (KTU, 2014). The report is available in Appendix 1.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Mobiliosioms technologijoms sparčiai žengiant į priekį, žmogaus gyvenimas tampa vis dinamiškesnis. Pasiekimai vaizdų analizėje ir apdorojime atveria naujas galimybes įvairioms mokslo, meno, verslo, pramogų ir kitoms sritims. Pasitelkus mobiliąsias technologijas ir vaizdų apdorojimą galima sukurti naujus įrankius, programas, leidžiančias žmogaus gyvenimą paversti lengvesniu. Šiame darbe pagrindinis dėmesys skiriamas geometriškai deformuotų vaizdų, sietinų su gardeline struktūra pasižyminčiais žaidimais (Sudoku, šaškėmis, šachmatais ir pan.), atkūrimui. Su geometriškai deformuotais vaizdais susiduriama įvairiose situacijose. Viena jų, kai žmogus užsimanęs sužaisti laikraštyje, stende ar brošiūroje pamatytą žaidimo variantą, pastarąjį išmaniojo telefono pagalba paprasčiausiai nufotografuoja, tuo sutaupydamas nemažai brangaus laiko. Darbo idėja – atkurti norimo žaisti žaidimo geometriją, po to atpažinti vaizde esančius simbolius (skaitmenis, figūras ir pan.) ir sudaryti žmogui sąlygas jau po keleto sekundžių užbaigti pasirinktą žaidimo variantą.

Darbe analizuojamas tradicinis Sudoku galvosūkis. Išanalizavus įvairius vaizdo apdorojimo momentus pateikiama geometriškai deformuoto žaidimo vaizdo atkūrimo procedūra ir palyginama atkurto vaizdo kokybė. Didesnis dėmesys skiriamas tiriamojo objekto radimui vaizde, objekto kampų išskyrimui ir vaizdo atkūrimui skirtų transformacijų pasirinkimui bei panaudojimui.

1 TEORINĖ DALIS

Teorinėje dalyje apžvelgiami skaitmeniniai vaizdai, jų specifika, vaizdų apdorojimo sritys. Taip pat, pateikiami geometriškai deformuotų vaizdų atkūrimui naudojami matematiniai metodai ir įrankiai.

1.1. SKAITMENINIAI VAIZDAI IR JŲ APDOROJIMO PROBLEMATIKA

Rega – tai labiausiai išvystytas ir daugiausiai naudojamas žmogaus pojūtis. Teigiama, kad būtent regėjimas suteikia apie 80% visos gaunamos informacijos apie mus supantį pasaulį [9]. Todėl vaizdai, kaip informacijos šaltinis, yra neatsiejama žmogaus gyvenimo dalis. Būtent dėl šios priežasties vaizdus imta naudoti vis įvairesnėse gyvenimo srityse – nuo paprasčiausio vaizdų įamžinimo ir saugojimo iki vaizdų analizės ir jų taikymo. Šiomis dienomis vaizdų analizė plačiai taikoma medicinoje, inžinerijoje, pramonėje, meteorologijoje, kriminalistikoje, karyboje, astronautikoje, moksliniuose tyrimuose ir kitur.

1.1.1. SKAITMENINIAI VAIZDAI – KOMPIUTERINIAI REALAUS PASAULIO VAIZDŲ ANALOGAI

Skaitmeniniai vaizdai – tai diskretizuoti realaus pasaulio vaizdai (objektai), pateikti dvimačiu (trimačiu) baigtiniu skaičių masyvu X, (1.1 formulė), sudarytu iš mažų sudedamųjų elementų – pikselių (vokselių, trimačiu atveju). Kiekvienas pikselis (angl. *pixel, „pic(ture) el(ement)“* - vaizdo elementas) reprezentuoja tam tikrą šviesos dažnį ir intensyvumą. Kitaip sakant, vaizdas turi spalvines, chromatines ir achromatines, savybes, kurios aprašomos panaudojant realių skaičių intervalą $L = [0{,}255]$, fizinius matmenis („atramą“), žymimus: $\square = \{(x, y) | a \leq x \leq b, c \leq y \leq d\}$, ir realiją dvimatę funkciją $f\colon \square \to L$. [5] Funkcija $f(x, y)$ nusako vaizdo pikselio *(x, y)* intensyvumą.

$$[X(m_1, m_2)] = \begin{bmatrix} X(1,1) & \cdots & X(1, M_2) \\ \vdots & \ddots & \vdots \\ X(M_1, 1) & \cdots & X(M_1, M_2) \end{bmatrix};\ m_1 = \{1, \dots, M_1\}; m_2 = \{1, \dots, M_2\}. \qquad (1.1)$$

Kompiuteriniai vaizdai pagal jų kodavimo būdą skirstomi į dvi kategorijas: rastrinius ir vektorinius vaizdus. [4] Vektoriniai vaizdai – tai vaizdai, kuriuose sukaupta informacija aprašoma matematinėmis formulėmis. Deformuojant vektorinius vaizdus keičiasi tik juose esančios informacijos matematinių formulių parametrai, todėl jie pasižymi gera vaizdo kokybe. Tačiau vektoriniai vaizdai geriausiai tinka mažai spalvų turintiems vaizdams, o vaizdo objektai turi būti sudaryti iš lengvai matematinėmis formulėmis aprašomų formų, tokių kaip tiesės, kreivės, apskritimai, elipsės. Šiame darbe nagrinėsime tik rastrinius (taškinius) vaizdus, kurie pagal savo savybes dar skirstomi į [14]:

a) rastrinius intensyvumo vaizdus. Šio tipo vaizdo pikselio reikšmė atitinka signalo intensyvumą;

b) grafinius rastrinius vaizdus kitaip dar vadinamus binarinius (dvejetainius) vaizdus. Binarinio vaizdo pikseliai sudaryti iš loginių reikšmių 0, 1 masyvo;

c) RGB kodavimo vaizdus. Vaizdo pikseliai įgyja tris bazinių spalvų (raudonos, žalios ir mėlynos) svorius – spalvos kodą ir

d) indeksuotus, kai pikselio reikšmė yra nuoroda į spalvos kodą.

Matematiniu požiūriu patogu apibrėžti metrinę skaitmeninių vaizdų erdvę $(S^2(n), \delta)$, kur $S^2(n) = \{[X(m)] | m = (m_1, m_2)\}$ – skaitmeninių vaizdų aibė, o $\delta = \delta(X_1, X_2) = \left(\frac{1}{N^2}\sum_m X_2(m) - X_1(m)^2\right)^{1/2}$ - vidutinė kvadratinė paklaida (atstumas, metrika) tarp bet kurių dviejų aibės $S^2(n)$ elementų (vaizdų). [5]

Darbe naudosime kelių tipų rastrinius vaizdus.

1.1.2. GEOMETRIŠKAI DEFORMUOTI VAIZDAI IR JŲ SPECIFIKA

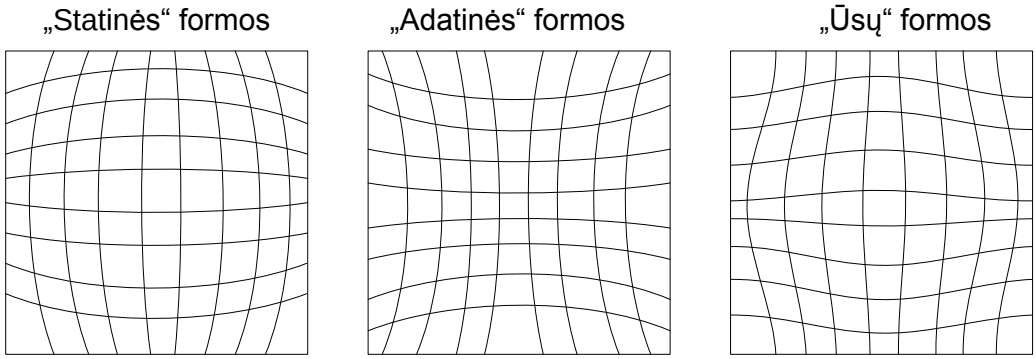
Deformuoti vaizdai sudaro skaitmeninių vaizdų aibės poaibį. Vaizdą vadiname geometriškai deformuotu, jeigu vaizdas arba jo objektai yra pakeitę savo natūralią formą, dydį ar poziciją. Tačiau iš kur atsiranda deformuotas vaizdas? Vaizdo įamžinimo momentu susiduriama su dviejų tipų iškraipymais (angl. *distortion*): optiniais ir perspektyviniais. [10] Dėl to vaizdas įgyja nežymių arba net labai pastebimų deformacijų. Optiniai iškraipymai atsiranda dėl kameros optinės objektyvo konstrukcijos. Yra žinomi trys optiniai iškraipymų tipai

a) išlenkimas į vidų arba „statinės“ (angl. *barrel*) formos iškraipymas;

b) išlenkimas į išorę nuo centro arba „adatinės“ (angl. *pincushion*) formos iškraipymas;

c) banguojantis arba „ūsų“ (angl. *moustache*) formos iškraipymas.

Optinių iškraipymų tipų vizuali interpretacija pateikta 1.1 paveikslėlyje.



1.1 pav. Optinių iškraipymų tipai

Perspektyvinis iškraipymas, priešingai nei optinis, visai nesusijęs su kameros vidinėmis savybėmis. Šio tipo iškraipymą lemia kameros pozicija, fotografuojamo objekto atžvilgiu, arba objekto, esančio įrėmintame vaizde, pozicija. [10] Siekiant suprojektuoti trimatį vaizdą į dvimatę erdvę susiduriama su kameros ir objekto pozicijos pasirinkimu, pvz. iš arti fotografuojamas vienas objekto elementas perspektyviai iškreipia likusio vaizdo matmenis, todėl tolumoje esantys objektai atrodo mažesni nei yra iš tikrųjų. Šio tipo vaizdo iškraipymo pavyzdys pateiktas 1.2 paveikslėlyje.

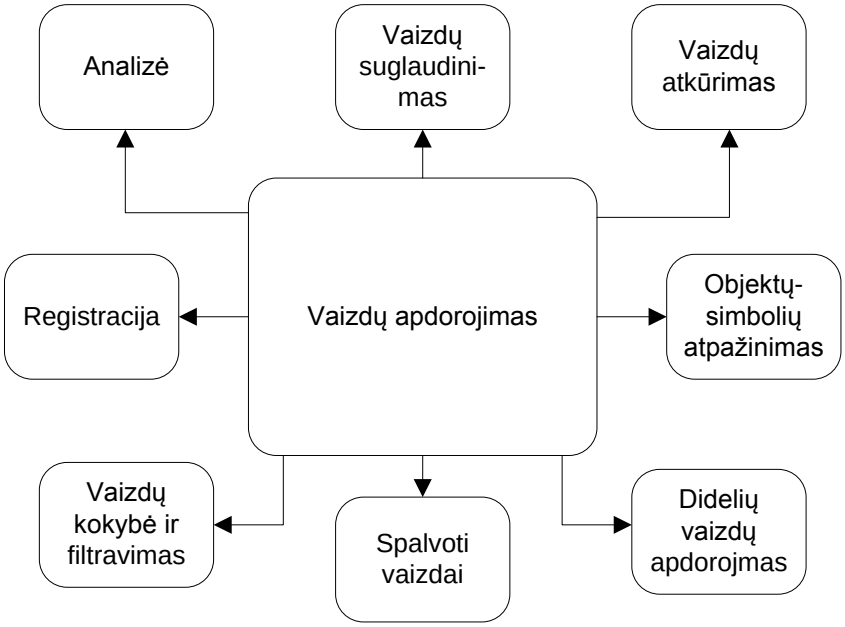
1.2 pav. Perspektyvinis vaizdo iškraipymas

Perspektyvinius iškraipymus galime suskirstyti į: ištempimo (angl. *extension*), suspaudimo (angl. *compression*), pasvyrimo (angl. *leaning*) ir pasukimo iškraipymus. Darbe nagrinėsime būtent dėl pastarojo tipo iškraipymų atsiradusias geometrines deformacijas.

1.1.3. SKAITMENINIŲ VAIZDŲ APDOROJIMO SRIČIŲ APŽVALGA

Skaitmeniniai vaizdai neša labai daug įvairios informacijos. Norint ją išgauti naudojami įvairūs vaizdų apdorojimo metodai. Skaitmeninis vaizdo apdorojimas (angl. *digital image processing*), tai triukšmo pašalinimas, vaizdo atkūrimas, segmentacija, požymių išskyrimas ir kiti apdorojimo būdai bei metodai. Didžiausią įtaką skaitmeninių vaizdų apdorojimui turėjo trys veiksniai: a) spar-

tus kompiuterinės technologijos vystymasis, b) matematinių mokslo žinių taikymo plėtimas ir c) vaizdų apdorojimo paklausos padidėjimas įvairiose mokslo, meno, pramonės šakose. Kadangi vaizdų apdorojimas apima labai daug sričių, tai ir patys apdorojimo metodai yra labai plačios paskirties. Vaizdų apdorojimo problematika pavaizduota 1.3 paveikslėlyje.



1.3 pav. Vaizdų apdorojimo problematika [7]

Vaizdų apdorojimas – manipuliavimas vaizdu siekiant išgauti kitą vaizdą, kuo nors besiskiriantį nuo pradinio. Apdorojimui dar priskiriamos ir kai kurios analizės operacijos, kuriomis skaičiuojamos vaizdo charakteristikos. Vaizdo analizės tikslas yra išgauti informaciją iš vaizdo. Analizė, taip pat, padeda identifikuoti ar klasifikuoti tam tikrus objektus. Plačiau aptarkime tam tikras vaizdų apdorojimo sritis.

Vaizdo kokybės gerinimas (angl. *enhancement*) susietas su žmogaus regėjimu. Būtent žmogus yra subjektyvus vaizdo kokybės vertintojas, t.y. vizualiai įvertinamas tam tikro vaizdo kokybės gerinimo metodo efektyvumas. Dažniausiai naudojamos vaizdo ryškinimo, filtravimo, suliejimo naikinimo (angl. *deblurring*) ir paaštrinimo (angl. *sharpen*), triukšmo panaikinimo, histogramos išlyginimo bei kontrasto reguliavimo priemonės. [15]

Vaizdo atkūrimas (angl. *restoration*) irgi susietas su subjektyviu žmogaus vaizdo kokybės vertinimo kriterijumi. Tačiau priešingai nei vaizdų kokybės gerinimo atveju, vaizdo atkūrimo procedūra pagrįsta matematiniais arba tikimybiniais modeliais. Geometriškai deformuoti vaizdai yra viena iš vaizdų atkūrimo rūšių. Deformuotus vaizdus galima atkurti įvairiomis transformacijomis (žr.1.2.1 skyrelio 1.1 lentelę).

Vaizdo analizė apima labai įvairius vaizdo apdorojimo aspektus, tokius kaip segmentavimas ir morfologiniai operatoriai, vaizdo transformacijos, statistika ir vaizdo matmenys. *Segmentavimas*

taikomas kraštų, kontūrų vaizde radimui, taip pat, objektų aptikimui, atpažinimo užduotims ir medicininiams vaizdams. Po segmentavimo naudojami *morfologiniai operatoriai,* kurie priskiria kiekvieną objektą atskirai sričiai ir gali pateikti objektų pastorintus (angl. *thickening*), paplonintus (angl. *thinning*) kontūrus ar karkasinius vaizdus. Vaizdų analizėje, taip pat, naudojamos statistinės charakteristikos, tokias kaip: vidurkis, standartinis nuokrypis, intensyvumas, ir histogramos metodas.

1.2. GEOMETRIŠKAI DEFORMUOTŲ SKAITMENINIŲ VAIZDŲ ATKŪRIMO PROBLEMA

Skaitmeninio geometriškai deformuoto vaizdo atkūrimo problematika apima įvairias vaizdo apdorojimo sritis, tokias kaip: vaizdo atpažinimas, analizė, morfologija, segmentavimas, vaizdo atkūrimas ir t.t.

1.2.1. PROBLEMOS AKTUALUMAS IR SPRENDIMO BŪDAI

Atliekant įvairius tyrimus ir analizes, susijusias su vaizdais ir jų apdorojimu, dažnai susiduriama su deformuoto vaizdo atkūrimo problema. Kiekviename tyrime yra pritaikomi specifiniai deformacijų šalinimo metodai. Šiame darbe nagrinėjami keturkampio formą turinčio tradicinio Sudoku žaidimo vaizdai (nuotraukos) ir galimi perspektyviniai jų iškraipymai.

Geometriškai deformuotų vaizdų atkūrimui naudojamos įvairios transformacijos. Vaizdas gali būti transformuojamas dvejopai: a) pirminio vaizdo elemento reikšmės keičiamos kitomis arba b) keičiama tik elemento pozicija vaizde. Pastarajai priklauso geometrinės transformacijos. Atliekant geometrines operacijas, vaizdas X_1 transformuojamas į naują vaizdą X_2, modifikuojant vaizdo X_1 pikselių koordinates. Šių vaizdų X_1 ir X_2 pikseliai yra sveikieji skaičiai. Todėl apibrėžiant geometrines transformacijas skaitmeninio vaizdo $X_1(x, y)$ plokštumoje bendrą transformacijų matematinę išraišką galima užrašyti taip:

$$[X_2(\tilde{x}, \tilde{y})] = \left[X_1 \left(T_x\ (x, y), T_y\ (x, y) \right) \right], \tag{1.2}$$

kur T_x, T_y – atitinkamos koordinatės transformacijos funkcijos. Jeigu po transformacijos gaunamos trupmeninės reikšmės taikomas vienas iš interpoliavimo metodų (žr. 2.2.5 skyrelį), siekiant gauti sveikaskaitines transformuoto vaizdo koordinačių reikšmes.

Geometrinės transformacijos skirstomos į globaliąsias ir lokaliąsias. [1] Globaliosioms priklauso šios transformacijų grupės: atitikimo (euklidinės), afiniosios, dvitiesės (angl. *biliniear*), projekcinės, bikvadratinės (angl. *biquadratic*) ir polinominės arba paprastųjų daugianarių. Šių transformacijų matematinė išraiška taikoma viso vaizdo transformavimui. Lokaliųjų transformacijų skirtingos matematinės išraiškos taikomos skirtingoms vaizdo vietoms. Darbe aptariamos tik globaliosios transformacijos, nes lokaliai būtų galima taikyti ir globaliai taikomas matematines išraiškas.

Plačiau apžvelkime afiniąsias transformacijas. Bendra afiniosios transformacijos išraiška yra tokia:

$$\begin{pmatrix}\tilde{x}\\ \tilde{y}\end{pmatrix} = \begin{pmatrix}a_1 & a_2\\ b_1 & b_2\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}a_0\\ b_0\end{pmatrix}, \tag{1.3}$$

kur $\tilde{x}, \tilde{y}$ – naujo vaizdo koordinatės, x, y – pirminio vaizdo koordinatės, $\{a_i, b_i | i = 0,1,2\}$ afiniosios transformacijos matricos koeficientai. Koeficientai a_1, a_2, b_1 ir b_2 apibrėžia posūkio, šlyties ir atspindžio transformacijas, o a_0, b_0 – poslinkio transformaciją.

Atliekant vaizdo afiniąją transformaciją tiesės pereina į tieses, iš kurių lygiagrečios išlieka lygiagrečiomis ir po transformacijos. Afiniosioms transformacijoms priskiriamos šios geometrinės transformacijos: a) mastelio keitimas – sumažinimas (suspaudimas), padidinimas; b) perkėlimas, poslinkis; c) atspindys d) posūkis; e) šlytis. Atitikimo, arba panašumo, transformacijos (angl. *linear conformal or similarities*) yra afiniųjų transformacijų pogrupis. [1]

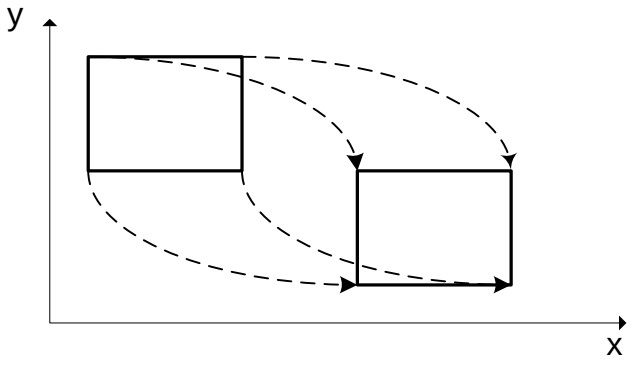
Matematinės geometrinių transformacijų išraiškos pateikiamos žemiau.

1. Poslinkio transformacija

Ši transformacija yra globali, nes transformacijos metu vaizdo pikselių tarpusavio padėtis nesikeičia, keičias tik jų padėtis plokštumoje, t.y. visi pikseliai paslenkami vienodu atstumu. Poslinkio transformacija apibrėžiama taip:

$$\begin{pmatrix}\tilde{x}\\ \tilde{y}\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \cdot \begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}a_0\\ b_0\end{pmatrix}, \tag{1.4}$$

kur a_0, b_0 – poslinkio transformacijos parametrai. Būtent šie parametrai nurodo kokia kryptimi ir per kiek paslenkamas visas vaizdas. Kai vaizdas slenkamas už pirminio vaizdo ribų, tai naujas vaizdas padidinamas, o neužpildytos vaizdo vietos, pagal nutylėjimą, būna juodos.



1.4 pav. Poslinkio transformacija

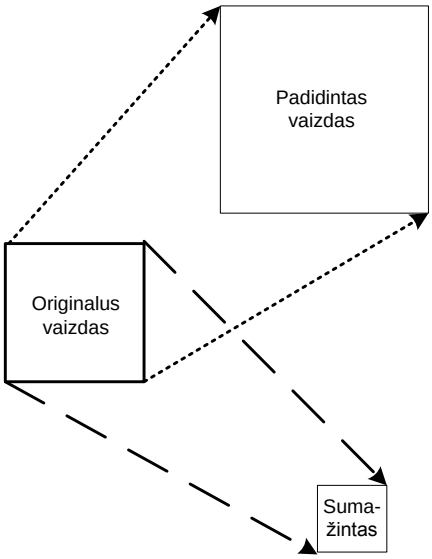
Jei poslinkio reikšmės sveikieji skaičiai, po transformacijos neatsiranda naujų pikselių, kurių reikšmės yra nežinomos, todėl interpoliavimas neatliekamas. Jei poslinkio reikšmės trupmeninės, po transformacijos vaizdo reikšmėms taikomas interpoliacijos metodas (žr. 2.2.5 skyrelį), siekiant gauti sveikas transformuoto vaizdo koordinačių reikšmes.

2. *Mastelio keitimo transformacijos*

Mastelio keitimo transformacija yra globali, ji padidina arba sumažina vaizdą pakeisdama vaizdo pikselių kiekį. [1] Mastelio keitimo transformacija apibrėžiama taip:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}, \tag{1.5}$$

kur *x, y* – vaizdo taškų koordinatės, S_x, S_y – mastelio keitimo transformacijos parametrai. Jei $S_x = S_y$, tai po mastelio keitimo vaizdo dydis atitinkamai pakinta, o objektų forma išlieka tokia pati kaip ir originaliame vaizde.



1.5 pav. Mastelio keitimo transformacija

Dalis informacijos prarandama, kai vaizdas mažinamas, nes originalūs pikseliai ištrinami. Po tokios transformacijos gautame vaizde lieka mažiau pikselių negu originaliame. Kai vaizdas didinamas, gauname daugiau pikselių turintį vaizdą. Didinant mastelio koeficientus, vaizdo kokybė prastėja ir atsiranda briaunų „laipuotumo“ efektas arba raibuliavimas. Po didinimo transformacijos naujame vaizde atsiranda tuščių reikšmių, jiems užpildyti naudojami interpoliacijos metodai.

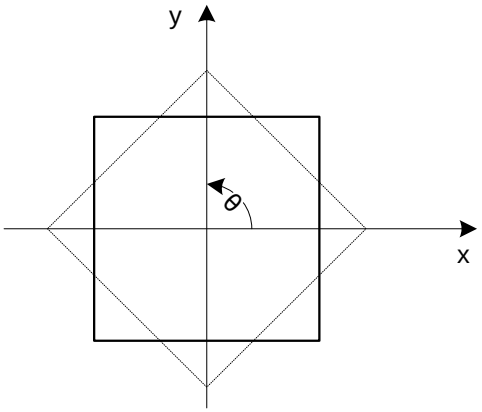
3. Posūkio transformacijos

Posūkio transformacija pasuką visą vaizdą erdvėje nurodytu kampu. Posūkio transformacija apibrėžiama taip:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}, \tag{1.6}$$

kur *x, y* – vaizdo taškų koordinatės, θ – posūkio kampas. Atlikus posūkio transformaciją vaizdo orientacija pakinta, o objektų forma išlieka tokia pati. Vaizdo dydis gali kisti, kad tilptų visas pasuktas vaizdas. Jei $\theta > 0$, tai vaizdas sukamas prieš laikrodžio rodyklę, kitu atveju pagal. Pasukto vaizdo pikselių intensyvumų reikšmėms paskaičiuoti naudojami įvairūs interpoliacijos metodai.

Sukamo vaizdo kokybė prastėja ir kuo daugiau kartų sukamas vaizdas tuo blogesnė gaunama vaizdo kokybė.



1.6 pav. Posūkio transformacija

Tik kai vaizdas sukamas $\theta = \pm 90°$ kampu vaizdo kokybė nenukenčia. Kitokio kampo sukimo operacija yra sudėtingesnė. Tuščių tarpų užpildymui taikomi interpoliavimo metodai.

4. Atspindžio transformacijos

Atspindžio transformacijos atveju apsukami viso vaizdo matricos eilutės ir stulpeliai. Apsukus eilutes gaunamas horizontalus atspindys, stulpelius – vertikalus.

Horizontalus atspindys apibrėžiamas taip:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{1.7}$$

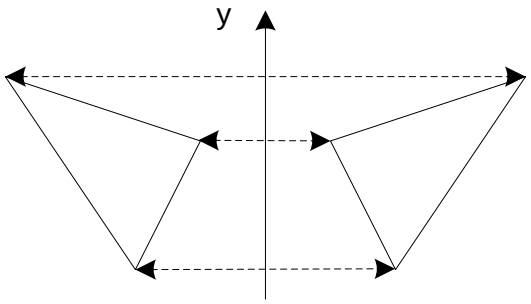
Vertikalaus atspindžio formulė:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{1.8}$$

Atspindžio ašies, statmenos vaizdo plokštumai, išraiška yra tokia:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \tag{1.9}$$

kur *x, y* – vaizdo taškų koordinatės. Po atspindžio transformacijos vaizdo orientacija nepakinta, o vaizdo dydis ir objektų forma išlieka tokie patys.



1.7 pav. Atspindžio transformacija pagal y ašį

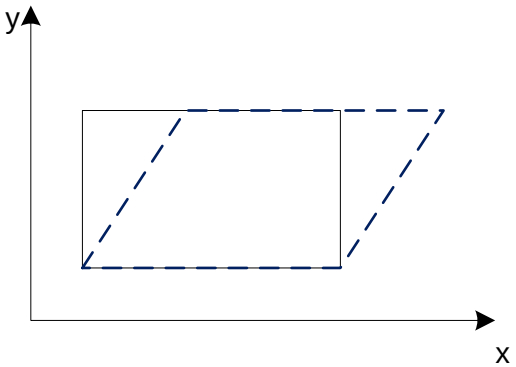
Taip pat, po šios transformacijos neatsiranda naujų pikselių, todėl nereikia atlikti interpoliavimo veiksmų, nebent atspindžio ašies *x* arba *y* reikšmė yra realus skaičius. Tokiu atveju pikselių reikšmės po transformacijos nustatomos taikant interpoliavimo metodus.

5. *Šlyties transformacija*

Šlyties transformacijos metu visi vaizdo pikseliai paslenkami atstumu, lygiagrečiu su šlyties ašimi, proporcingu jų atstumui iki šlyties ašies. [1] Esant *x* ašies šlyčiai sh_y, pikseliai paslenkami horizontaliai, tolygiai didėjančiu atstumu tolstant nuo šlyties pradžios ašies. Esant *y* ašies šlyčiai sh_x, pikseliai paslenkami vertikaliai tolydžiai didėjančiu atstumu tolstant nuo vertikalios šlyties pradžios ašies. Šlyties transformacijos išraiška yra tokia:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} 1 & sh_x \\ sh_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}, \tag{1.10}$$

kur *x*, *y* – vaizdo taškų koordinatės, sh_x, sh_y šlyties parametrai. Po šlyties transformacijos, vaizdo objektų forma pasikeičia šlyties kryptimi.



1.8 pav. Šlyties transformacijos pagal x ašį iliustracija

Vaizdo dydis gali kisti siekiant sutalpinti po transformacijos gautą vaizdą. Įrašius abu šlyties parametrus galima gauti dvigubą šlyties efektą, t.y. stačiakampis pereina į lygiagretainį. Šlyties transformacijos taikomos projekcinėms transformacijoms realistiniam vaizdui kurti.

Šių transformacijų mišiniai gali atkurti įvairius iškraipymus arba priešingai – gali iškraipyti turimą objektą į norimą formą. Visos lokaliųjų transformacijų grupės pateikiamos 1.1 lentelėje. Paprastumo dėlei jos aprašytos matematinių lygčių forma.

1.1 lentelė.

Geometrinių transformacijų funkcijos [16]

Transformacijų funkcijos	**Matematinės koordinačių perėjimo išraiškos**
Euklidinė	$\tilde{x} = ax + by + t_x;$ $\tilde{y} = -bx + ay + t_y;$
Afinioji	$\tilde{x} = a_0 + a_1 x + a_2 y;$ $\tilde{y} = b_0 + b_1 x + b_2 y;$
Dvitiesis (angl. *bilinear*)	$\tilde{x} = a_0 + a_1 xy + a_2 x + a_3 y;$ $\tilde{y} = b_0 + b_1 xy + b_2 x + b_3 y;$
Perspektyvinė	$\tilde{x} = \frac{(a_0 + a_1 x + a_2 y)}{c_0 x + c_1 y + 1}; \tilde{y} = \frac{(b_0 + b_1 x + b_2 y)}{c_0 x + c_1 y + 1};$
Bikvadratinė (angl. *biquadratic*)	$\tilde{x} = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 y^2 + a_5 xy;$ $\tilde{y} = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 y^2 + b_5 xy;$
Paprastųjų daugianarių (polinomų) (angl. *general polynomial*)	$\tilde{x} = \sum_i \sum_j a_{ij} x^i y^j; \tilde{y} = \sum_i \sum_j b_{ij} x^i y^j;$

Geometrinių transformacijų matematinių išraiškų koeficientams rasti reikalingos vaizde esančio objekto ir transformuoto objekto atitinkančių taškų koordinačių reikšmės. Kuo daugiau reikšmių (taškų) reikia transformacijos koeficientams rasti, tuo didesnį laisvės laipsnį turi transformacija.

Kadangi darbe sukurta deformuoto Sudoku galvosūkio vaizdo atkūrimo procedūra skirta mobiliosioms technologijoms, tai atkūrimui parenkamos paprastesnės, mažiau skaičiavimų reikalaujančios, transformacijos. Todėl darbe remiamasi afiniosiomis transformacijomis. Pasiūlytos geometrinės transformacijos plačiau apžvelgiamos 2.2. skyrelyje.

2 TIRIAMOJI DALIS

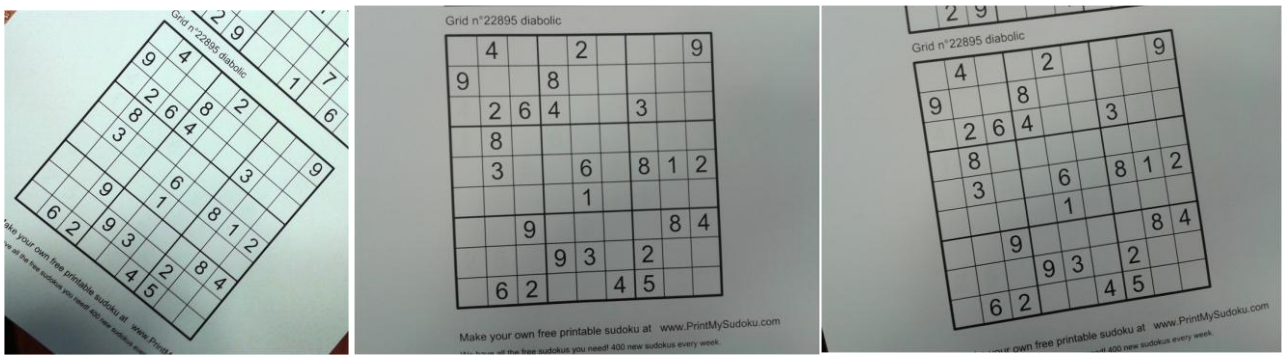
Šioje dalyje pateikiama geometriškai deformuoto vaizdo atkūrimo procedūra ir galimos jos variacijos su rezultatų pavyzdžiais. Taip pat, pateikiami gautų vaizdų kokybės įvertinimas taikant skirtingus atkūrimo metodus.

2.1. GEOMETRIŠKAI DEFORMUOTŲ SKAITMENINIŲ VAIZDŲ IDENTIFIKAVIMAS

Tarkime, turime geometriškai deformuotą vaizdą $X = \{X(i,j) | i = 1, \dots, M; j = 1, \dots, N\}$, ($M \times N$ – vaizdo matmenys), kuriame yra pavaizduotas tiriamasis objektas – tradicinis Sudoku žaidimas. Nustatykime eksperimentiniams tyrimams naudojamam vaizdui X taikomas sąlygas:

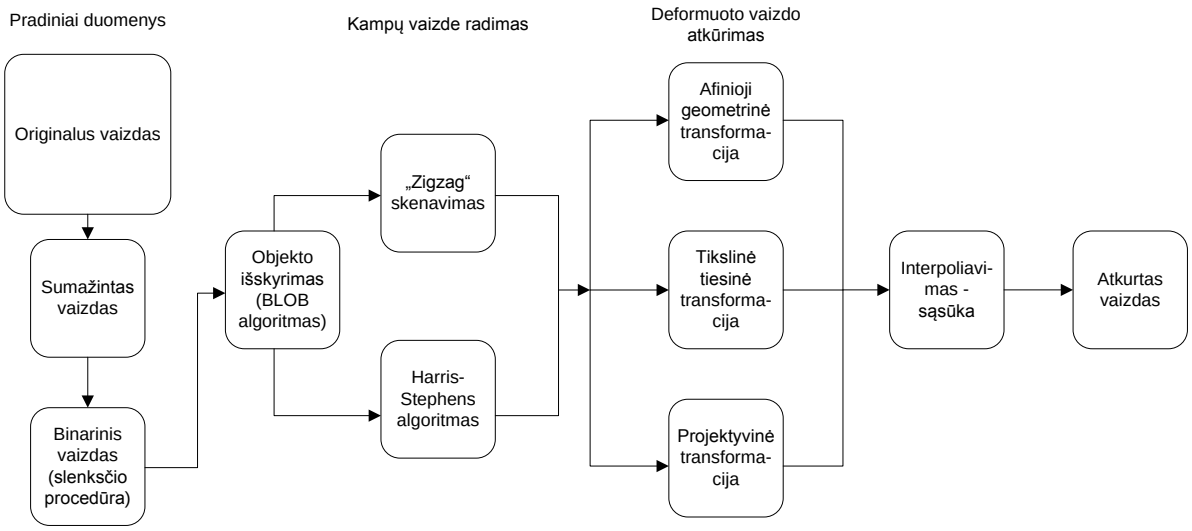
- Vaizde esantis galvosūkis yra svarbiausias ir pagrindinis šio vaizdo objektas, t.y. nuotraukoje žaidimas užima didžiąja vaizdo dalį.
- Sudoku galvosūkį stengiamasi nufotografuoti kuo artimesnį jo natūraliai formai – kvadratui.
- Vaizde matomos visos Sudoku žaidimo kraštinės ir kampai.
- Vaizdas nepasižymi optiniais iškraipymais.
- Pateikiamas vaizdas nėra apverstas.

Keletas pavyzdžių, iš sudarytos eksperimentiniams tyrimams naudojamų vaizdų bazės, pateikti 2.1 paveikslėlyje.



2.1 pav. Eksperimentiniams tyrimams skirti vaizdai

Šie vaizdai atitinka aukščiau paminėtas tiriamam vaizdui taikomas sąlygas. Toliau pateikiama pasiūlyta geometriškai deformuoto vaizdo atkūrimo schema.



2.2 pav. Geometriškai deformuoto vaizdo atkūrimo procedūros schema

Išskiriami keturi pagrindiniai Sudoku galvosūkio vaizdo atkūrimo etapai:

1) Pradinių duomenų paruošimas – nuskaitymas, sumažinimas, konvertavimas į pilko atspalvio ir dvejetainį vaizdus;
2) Kampų vaizde radimas;
3) Deformuoto vaizdo atkūrimas;
4) Atkurto vaizdo kokybės gerinimas ir vizualizavimas.

Sekančiuose skyreliuose detaliau aptariami esminiai vaizdo atkūrimo etapai.

2.1.1. PRADINIŲ VAIZDŲ PARUOŠIMAS

Labai svarbi yra geometriškai deformuoto vaizdo paruošimo procedūra. Pradinių duomenų paruošimas – tai preliminarus deformuoto vaizdo apdorojimas. Dažniausiai kamera fiksuoja trimatį RGB kodavimo spalvotą vaizdą. Skaitmeninis RGB (angl. *Red, Green, Blue*) trimatis vaizdo vokselis saugo tris reikšmes – kiekvienos spalvos dedamąsias. Toks vaizdo formatas, deformuoto Sudoku galvosūkio vaizdo atkūrimo skaičiavimams, netinka, todėl spalvotas vaizdas konvertuojamas į vaizdą su pilka šviesos intensyvumo skale. Naudojamos įvairios RGB konvertavimo, į pilkos spalvos intensyvumo vaizdą, lygtys. Populiariausios yra šios [20]:

$$Pilka = \frac{Raudona + Žalia + Mėlyna}{3};$$

$$Pilka = \left(0.2126 \cdot Raudona + 0.7152 \cdot Žalia + 0.0722 \cdot Mėlyna\right);$$

$$Pilka = \left(0.299 \cdot Raudona + 0.587 \cdot Žalia + 0.114 \cdot Mėlyna\right);$$

$$Pilka = \frac{MAX\left(Raudona, Žalia, Mėlyna\right) + MIN\left(Raudona, Žalia, Mėlyna\right)}{2};$$

kur spalvų pavadinimai $(Raudona, Žalia, Mėlyna)$ – atitinkamos spalvos vokselio reikšmė, o $Pilka$ – nauja konvertuoto vaizdo pikselio reikšmė.

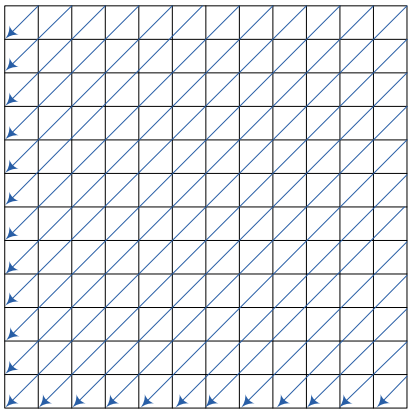
Tolimesniems skaičiavimams, taip pat, reikalingas ir loginių reikšmių masyvo vaizdas, tad pilkos spalvos atspalvio vaizdas, parinkus tinkamą slenkstinę (angl. *threshold*) procedūrą su slenksčiu, konvertuojamas į binarinį vaizdą. Slenkstinės procedūros metodų grupės [17]:

- Histogramos formos paremti metodai analizuoja jos formą – aukštumas, žemumas, nuokrypį;
- Grupavimu paremti metodai grupuoja vaizdą į dvi dalis: foną ir priekį;
- Entropija paremti metodai;
- Objekto bruožais paremti metodai;
- Erdviniai metodai paremti tikimybiniu skirstiniu ir / arba koreliacija tarp pikselių;
- Lokalūs metodai paremti aplinkinių pikselių charakteristikomis.

Darbe taikomi vidurkiu, grupavimu ir histogramos forma paremti metodai. Toliau galima taikyti vaizdo radimo (segmentavimo), išskyrimo (morfologinius) ir deformuoto vaizdo atkūrimo metodus.

2.1.2. SPECIALIZUOTO VAIZDO KONTŪRŲ DETEKTORIAUS TAIKYMAS

Nagrinėjant tam tikrą objektą vaizde, pirmiausia reikia jį rasti. Atsižvelgiant į individualias ieškomo objekto savybes, t.y. į gardelinę struktūrą, randame optimalų vaizdo išskyrimo būdą. Šiam tikslui taikysime segmentavimą ir morfologijos operacijas. Kontūrų vaizde išskyrimui pritaikytas patobulintas susietų sričių (ang. *connected-regions*) aptikimas skenavimo būdu ir didžiausio binominio objekto – BLOB (ang. *Binomial Ladge OBject*) aptikimo algoritmas. Susietų sričių aptikimo algoritmo esmė – surasti atskirus dvejetainio vaizdo objekto elementus ir juos pažymėti skirtingais lygiais. Mūsų atveju, aktualus tik daugiausiai erdvės vaizde užimantis objektas, todėl BLOB algoritmas būtent jį išskiria iš rezultatų, gautų po susietų sričių aptikimo.



2.3 pav. Pasiūlytas skenavimo būdas

Susietų sričių išskyrimo algoritmo pseudo kodas [18]*:*

Algoritmas susideda iš dviejų skenavimų. Tarkime, turime vaizdo matricą X, jos elementai $X(i,j)$.

Pirmas skenavimas:

Vaizdas X skenuojamas pasiūlytu skenavimo būdu (2.3 pav.).

- Jei *X(i, j)* – fonas, tikrinamas kitas elementas.
- Jei matricos elementas *X(i, j)* – ne fonas,

Pirma iteracija:

- *X(i, j)* priskiriame žymeklį v_s;
- Sudarome kaimynų matricą *K*:

$$K = \begin{pmatrix} X(i-1,j-1) & X(i,j-1) & X(i+1,j-1) \\ X(i-1,j) & X(i,j) & X(i+1,j) \\ X(i-1,j+1) & X(i,j+1) & X(i+1,j+1) \end{pmatrix} \quad (2.1)$$

- $\forall K(i,j)$ pikseliui, nelygiam fonui, priskiriame žymeklį v_s;
- Iteracija+1;

Kitos iteracijos

- Jei *X(i, j)* – fonas, tikrinamas kitas elementas.
- Jei matricos elementas *X(i, j)* ne fonas,
  - Sudarome 3×3 kaimynų matricą K (2.1);
  - Matricoje *K* randame mažiausią žymeklį $min(v)$;
  - Jei tokio nėra, sukuriame naują v_{s+1} ir priskiriame jį elementui *X(i, j)*;
  - Kitu atveju $\forall K(i,j)$ elementui, nelygiam fonui, priskiriame mažiausią žymeklį;
  - Iteracija+1;

Antras skenavimas: Rastrinis skenavimas: pirmiausiai stulpelio elementus skenuoja, o po to eilučių.

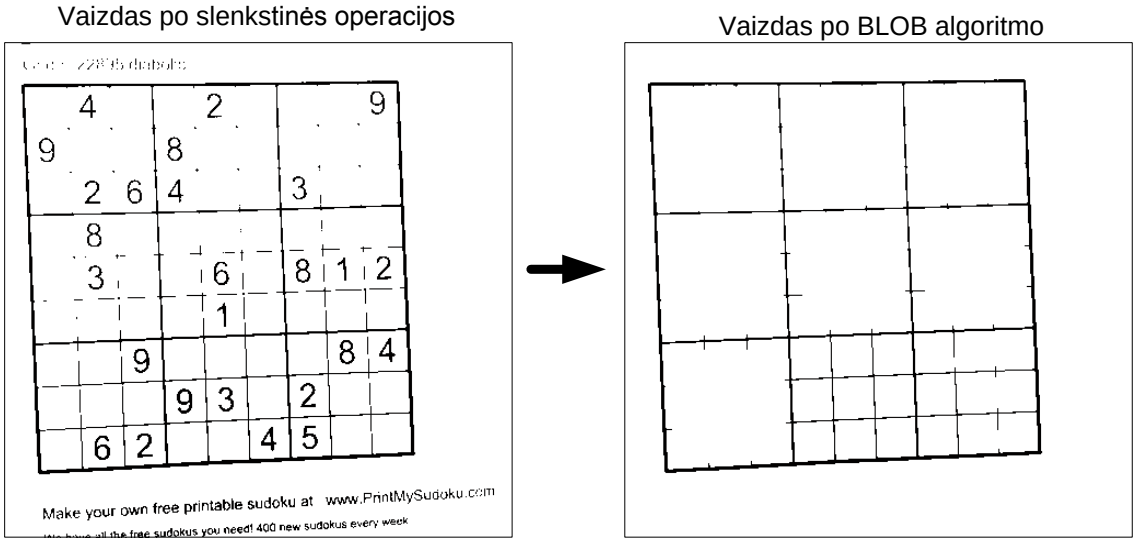
- Jei *X(i, j)* – fonas, tikrinamas kitas elementas.
- Jei matricos elementas *X(i, j)* – bet kuris žymeklis
  - Sudarome 3×3 kaimynų matricą K;
  - Matricoje *K* randame mažiausią žymeklį $min(v)$;
  - $\forall K(i,j)$ elementui, nelygiam fonui, priskiriame mažiausią žymeklį;

BLOB algoritmas: Išrenkamas didžiausią sritį vaizde užimantis objektas ir jis pavaizduojamas.

Algoritmo pabaiga.

Susietų sričių aptikimo ir BLOB algoritmo programos tekstas pateiktas priede Nr. 2. MatLab programos failas *blob.m*.

Atlikus susietų sričių aptikimo algoritmą kiekvienam objektui priskirti skirtingi lygiai leidžia rasti didžiausiai erdvės užimantį vientisą objektą. Pavyzdys pateiktas 2.4 paveikslėlyje. Kad būtų išskirtas visas objektas reikia parinkti tinkamą slenkstinės operacijos lygį, kitu atveju galimas objekto dalių praradimas arba išorinių pikseliu priskyrimas objektui.



2.4 pav. Sudoku žaidimo vaizdas po BLOB didžiausio objekto išskyrimo algoritmo

Kitas žingsnis – rasto didžiausio objekto kampų išskyrimas.

2.1.3. KAMPŲ VAIZDE APTIKIMO ALGORITMAI

Kampas – tai dviejų dominuojančių linijų, kraštų susikirtimo vieta. Literatūroje aprašomi įvairūs kampų aptikimo algoritmai, tokie kaip Harris-Stephens, Shi-Tomasi, Forstner, Vang ir Brandy, SUSAN, Traikovic ir Hedley, bei Moravec kampų atpažinimo algoritmai. Pirmieji du integruoti „MatLab“ programoje. Darbe naudojamas Harris-Stephens kampų aptikimo algoritmas ir pasiūlytas konkrečių kampų aptikimo lokalusis skenavimas. Plačiau aptarkime juos atskirai.

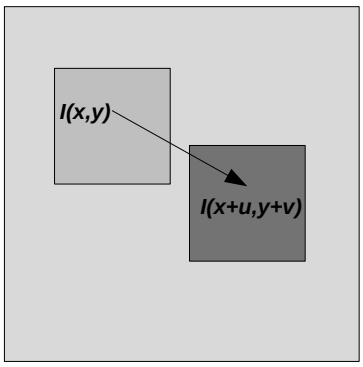
1) Harris-Stephens algoritmas

Harris-Stephens algoritmas, nepriklausomai nuo vaizdo posūkio, padidinimo ar vaizde esančių ryškumo pokyčių, puikiai identifikuoja kampus. [11] Kampo aptikimo algoritmo idėja – rasti mažą vaizdo „langelį“, pasižymintį dideliu intensyvumo pokyčiu, apskaičiuojamu judinat „langelį“ aplinkui. Tarp esamo ir pajudinto per koordinates [*u, v*] „langelio“ intensyvumo pokytis *E* aprašomas taip:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \qquad (2.2)$$

– $w(x,y)$ – „langelio“ funkcija;
– $I(x,y)$ – vaizdo pikselio, esančio $[x,y]$ pozicijoje, intensyvumo reikšmė;

– $I(x+u, y+v)$ – paslinkto „langelio“ pikselio, esančio $[x+u, y+v]$ pozicijoje, intensyvumo reikšmė;



2.5 pav. Harris-Stephens algoritmo „langelių“ judėjimas

„Langelio“ funkcija veikia kaip kaukė, užtikrina, kad tik norimas „langelis“ būtų naudojamas. Paprasčiausiu atveju $w(x, y)$ įgyja dvi reikšmes: „1“, kai pikselis priklauso “langeliui” ir „0“ – jei nepriklauso. Kitu atveju „langelio“ funkcija gali būti ir tam tikra pasiskirstymo funkcija, pavyzdžiui Gauso:

$$w(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Jeigu *u* ir *v* maži, tai galima (2.2) formulei taikyti Teiloro eilutę ir ji perrašoma taip:

$$E(u, v) = \sum_{x,y} w(x, y)\left[I(x, y) + I_x u + I_y v - I(x, y)\right]^2, \quad (2.2a)$$

kur I_x, I_y – gradientai atitinkamai koordinačių ašiai. (2.2a) formulė suprastinama ir pakeliama laipsniu. Gaunama nauja formulė:

$$E(u, v) = \sum_{x,y} w(x, y)\,[{I_x}^2 u^2 + 2uvI_xI_y + {I_y}^2 v^2] \quad (2.2b)$$

(2.2b) formulė perrašoma matricniu pavidalu:

$$E(u, v) = [u, v]\, B \begin{bmatrix} u \\ v \end{bmatrix} \quad (2.3)$$

Matrica *B* yra apskaičiuota naudojantis vaizdo gradiento sąvoka. Matrica yra geometrinės figūros elipsės formulės matricinė forma:

$$B = \sum_{x,y} w(x, y) \begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix}, \quad (2.3)$$

Matrica *B* charakterizuoja kaip intensyvumas kinta tam tikra kryptimi, o I_x, I_y pikselio dalinės išvestinės, apskaičiuojamos šitaip:

Baigtinis diferencijavimas į priekį:

$$\frac{I(x+h)-I(x)}{h} = I_x + O(h) \quad (2.5)$$

Baigtinis diferencijavimas atgal:

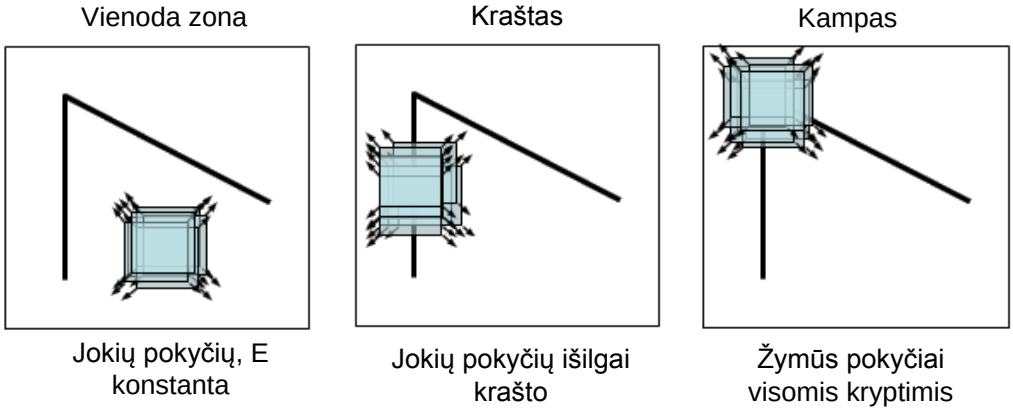
$$\frac{I(x)-I(x-h)}{h} = I_x + O(h) \quad (2.6)$$

Baigtinis centrinis diferencijavimas:

$$\frac{I(x+h)-I(x-h)}{2h} = I_x + O(h^2) \quad (2.7)$$

Dažniausiai naudojamas būtent centrinis diferencijavimas.

Algoritme apskaičiuojamos matricos *B* tikrinės reikšmės λ_1, λ_2, kurios nustato „langelyje“ esančio vaizdo tipą (2.6 pav.).



2.6 pav. Harris-Stephens kampų aptikimo algoritmo galimi „langelio“ tipai [11]

Kiekvienas tipas nustatomas pagal tikrinių reikšmių reikšmes. [6]

1) Jeigu λ_1, λ_2 tikrinės reikšmės labai mažos, o intensyvumas E beveik konstanta, tai „langelyje“ yra lygi (angl. *flat*) zona.
2) Jeigu viena tikrinė reikšmė maža, o kita didėle, tai lokaliai aptiktas kraštas. Poslinkis pagal kraštą iššaukia mažus intensyvumo E pokyčius.
3) Kai λ_1 ir λ_2 yra didėlės ir artimos viena kitai, o E įgyja dideles reikšmes visomis krytimis, tai „langelyje“ aptiktas kampas.

Kampų identifikavimui dar įvedamas kampo reagavimo dydis *R*, priklausomas tik nuo tikrinių reikšmių:

$$R = detB - k(traceB)^2, \quad (2.8)$$

kur $detB = \lambda_1\lambda_2$ B matricos determinantas, o $traceB = \lambda_1 + \lambda_2$ matricos pėdsakas; k – empirinė konstanta, kurios reikšmės gali būti *k=0,04 – 0,06*. „Langelio“ tipo ir R reikšmės sąryšis pateiktas 2.1 lentelėje.

2.1 lentelė.

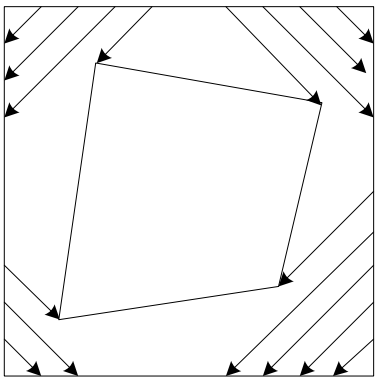
Harris-Stephens algoritmo „langelio“ tipų ir kampo reagavimo dydžio *R* sąryšis

„Langelio“ tipas	***R* reikšmė**
Kampas	*R* didelis (*R* > slenkstis)
Kraštas	*R* > 0
Lygi zona	*R* mažas (*R* < slenkstis)

Harris-Stephens algoritmas naudojamas programoje. Šio algoritmo taikymas ir lokalių kampų radimo programos tekstas pateiktas priede Nr. 2. MatLab programos failas *kampai.m*.

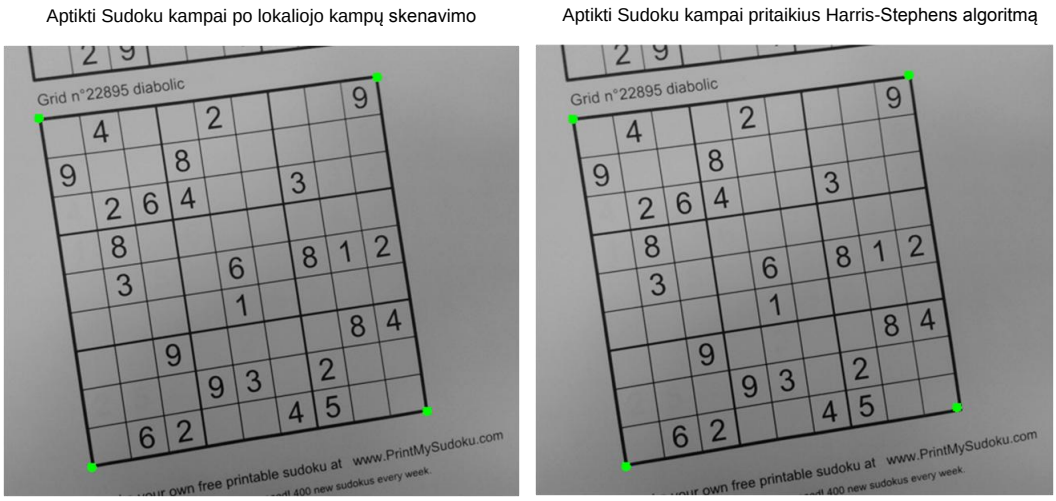
2) Lokalusis pasiūlytas skenavimas

Pasiūlytas skenavimas lokaliai randa konkretų binarinio vaizdo objekto kampo poziciją. Rastą kampą fiksuoja ir pereina prie kito kampo pozicijos paieškos. Taip randami visos keturių kampų pozicijos. Skenavimas iliustruojamas 2.7 paveikslėlyje.



2.7 pav. Objekto kampų aptikimas lokaliuoju skenavimu

Pasiūlyto kampų aptikimo skenavimo programos tekstas pateiktas priede Nr. 2. MatLab programos failas *ZigZag.m*. Kampų aptikimo lokaliojo skenavimo ir Harris-Stephens algoritmo rezultatas, naudojant eksperimentiniams tyrimams skirtą vaizdą, pavaizduotas 2.8 paveikslėlyje. Taškai žymi rastus kampus vaizde.



2.8 pav. Aptikti kampai pritaikius lokalųjį skenavimą ir Harris-Stephens algoritmą

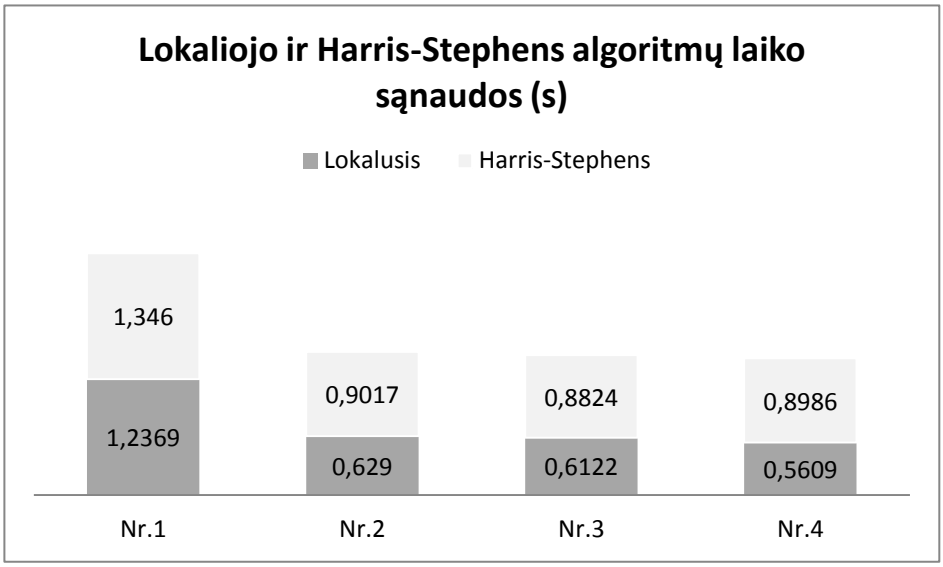
Visos eksperimentinio vaizdo (2.8 pav.) kampų koordinatės pateikiamos 2.2 lentelėje.

2.2 lentelė.

Skirtingais metodais aptiktos kampų koordinatės

Kampų paieškos metodas	**Kampų koordinatės**			
Harris-Stephens algoritmas	(121, 75)	(456, 31)	(174, 422)	(504, 363)
Lokalusis skenavimas	(120, 74)	(458, 33)	(173, 422)	(508, 366)

Atlikto tyrimo metu nustatytas abiem metodais palygintas kampų aptikimo vaizde tikslumas. Nustatytas nedidelis vidutinis nuokrypis (2,99 pozicijos). Pastebėta, kad Harris-Stephens algoritmas randa visus globalius taškus. Todėl dominančių kampų radimui tenka įvertinti galimas kampų pozicijas, o tai ne visuomet pavyksta. Tuo tarpu lokalusis ieško konkrečios reikšmės. Atliktas laiko sąnaudų eksperimentas su keturiais bandomaisiais vaizdas. Gauti suvidurkinti rezultatai pateikiami 2.9 paveikslėlyje.



2.9 pav. Lokaliojo ir Harris-Stephens algoritmų laiko sąnaudų palyginimo grafikas

Tyrimo meto nustatyta, kad lokalusis skenavimas deformuoto Sudoku galvosūkio kampus aptinka 1,39 kartų (t.y. 26,63%) greičiau nei Harris-Stephens kampų detektorius.

2.1.4. KITI METODAI

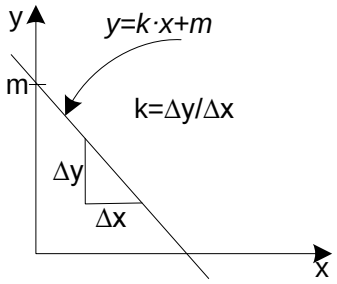
Kampų identifikavimui, taip pat, galima taikyti ir Hough transformaciją, kuri aptinka objekte esančias linijas ir apskritimus. Šiuo atveju dėmesys skiriamas linijoms, atitinkančioms objekto kraštus, ir jų susijungimo taškams – objekto kampams.

Hough transformacijos pagrindinė idėja [19] pateikiama žemiau.

Tiesės aprašomos tokia lygtimi:

$$y = kx + m,$$

kur k –gradientas, m tiesės susikirtimo taškas su y ašimi.

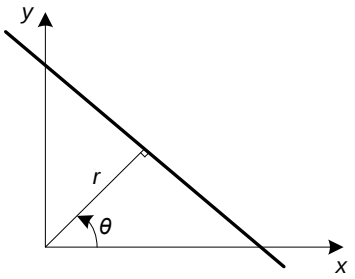


2.10 pav. Tiesės lygtis

Jeigu *x* ir *y* laikysime kaip parametrus, tai viena tiesės lygtis parametrų *k, m* ašyje atitiks dvi lygtis. Šių lygčių susikirtimas reprezentuoja lygties *x, y* ašyje parametrus. Šitaip aprašomos lygčių formulės, kai tiesė yra lygiagreti vienai iš ašių, netinkamos, todėl naudojamos (r, θ) polinės koordinatės. Perėjimas prie polinių koordinačių:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right)$$

Parametras *r* reprezentuoja trumpiausią atstumą tarp linijos ir koordinačių ašių pradžios, o θ yra kampas tarp koordinačių pradžios ir *r*.

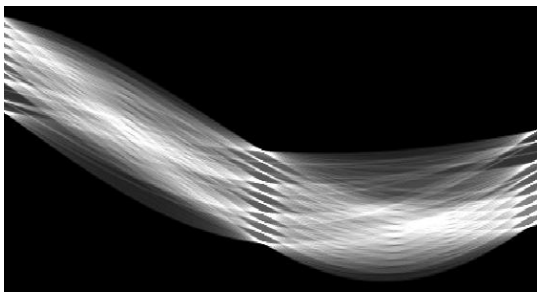


2.11 pav. Perėjimas prie polinių koordinačių

Perrašoma lygtis į (r, θ) parametrų erdvę:

$$r(\theta) = xcos(\theta) + ysin(\theta)$$

Lygtis atitinka sinusoidinę kreivę, unikalią (*x, y*) taškui. Jei kreivės priklausančios dviem skirtingiems taškams susikerta (r, θ) erdvėje, tai jie priklauso vienai tiesei vaizdo erdvėje.



2.12 pav. Tiesių susikirtimai polinėse koordinatėse

Kitaip sakant, jei taškai priklauso vienai tiesei, tai jų sinusoidės susikirs parametrų erdvėje ir susikirtimo parametrai aprašys tą tiesę.

2.2. GEOMETRIŠKAI DEFORMUOTŲ SKAITMENINIŲ VAIZDŲ ATKŪRIMAS

Darbe deformuoto vaizdo atkūrimui naudojamos trys transformacijos: afinioji ir tiksliniai parinkta transformacijų superpozicija, dvi nuosekliai taikomos afiniosios transformacijos ir vaizdo taškų perėjimo lygtys. Kiekvienas metodas atskirai apžvelgiamas plačiau sekančiuose skyreliuose. Pirmiau įvertinkime vienos afiniosios transformacijos veikimą keturkampiams vaizdams.

2.2.1. AFINIŲJŲ TRANSFORMACIJŲ PANAUDOJIMAS

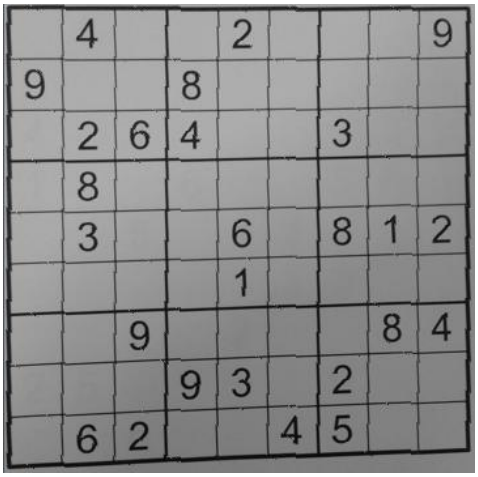
1) Tiesioginė afinioji transformacija

Afiniosios transformacijos lygtys pateiktos 1.1 lentelėje (žr. 1.2.1 skyrelį). Transformacijos matricinė forma aprašoma taip [2]:

$$\tilde{x} = Ax + a$$

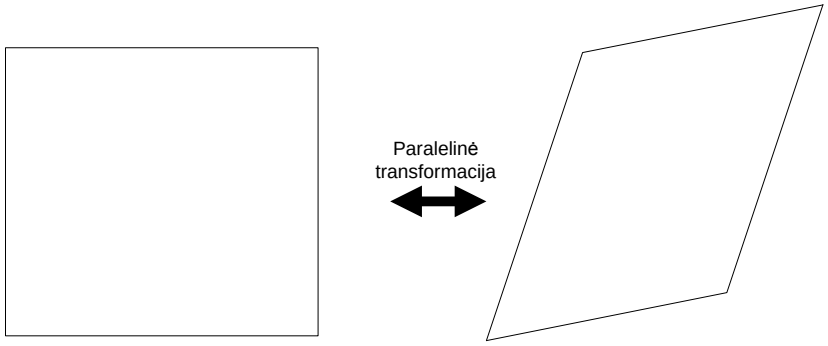
$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_0 \\ b_0 \end{pmatrix}, \quad (2.9)$$

čia $\{a_i, b_i | \; i = 0,1,2\}$ – afiniosios transformacijos koeficientai. Afinioji transformacija turi 6 laisvės laipsnius, todėl koeficientams apskaičiuoti reikalingos šešios lygtys, t.y. trys vaizdo taškai ir jų perėjimo koordinatės. Vienas taškas atitinka dvi lygtis. Patikrinamas afiniosios transformacijos veikimas specifiniam vaizdui (2.13 pav.).



2.13 pav. Afiniosios transformacijos taikymas deformuotam Sukodu galvosūkio vaizdui

Kaip matome atkurtas deformuotas Sudoku galvosūkio vaizdas (2.13 pav.) nėra kvadrato formos. Įsitikiname, kad viena tiesioginė afinioji transformacija tinkama tik paraleliniams iškraipymams atkurti (2.14 pav.).

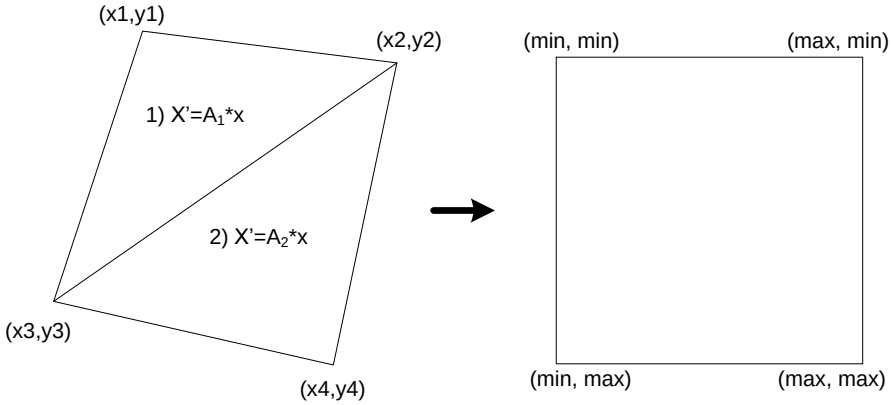


2.14 pav. Afiniosios transformacijos veikimo pavyzdys

Kadangi darbe naudojamų eksperimentinių vaizdų kraštinės dažniausiai yra neparalelinės, tai šiuo atveju tiesioginis vienos afiniosios taikymas nepriimtinas.

2) Dviejų afiniųjų transformacijų panaudojimas

Remiantis logika, kad kvadratas – tai du trikampiai, o afinioji transformacija skirta trijų taškų objektui, būtent trikampiu, pritaikome dvi afiniąsias transformacijas skirtingoms keturkampio pusėms.



2.15 Pav. Dviejų afiniųjų transformacijų taikymas.

Šiai transformacijai išvedami skirtingų afiniųjų transformacijų koeficientai.

Pirmosios afiniosios transformacijos A_1 koeficientai:

$$a_0 = \min + \frac{(\max - \min)(y_3 x_1 - x_3 y_1)}{(y_2 - y_1)(x_3 - x_1) - (y_3 - y_1)(x_2 - x_1)}$$

$$a_1 = -\frac{(\max - \min)(y_3 - y_1)}{(y_2 - y_1)(x_3 - x_1) - (y_3 - y_1)(x_2 - x_1)}$$

$$a_2 = \frac{(\max - \min)(x_3 - x_1)}{(y_2 - y_1)(x_3 - x_1) - (y_3 - y_1)(x_2 - x_1)}$$

$$b_0 = \min + \frac{(\min - \max)(y_2 x_1 - x_2 y_1)}{(y_2 - y_1)(x_3 - x_1) - (y_3 - y_1)(x_2 - x_1)}$$

$$b_1 = -\frac{(\min - \max)(y_2 - y_1)}{(y_2 - y_1)(x_3 - x_1) - (y_3 - y_1)(x_2 - x_1)}$$

$$b_2 = \frac{(\min - \max)(x_2 - x_1)}{(y_2 - y_1)(x_3 - x_1) - (y_3 - y_1)(x_2 - x_1)}$$

Antrosios afiniosios transformacijos A_2 koeficientai:

$$a_0 = \max + \frac{(\min - \max)(y_4 x_2 - x_4 y_2)}{(y_3 - y_2)(x_4 - x_2) - (y_4 - y_2)(x_3 - x_2)}$$

$$a_1 = \frac{-(\min - \max)(y_4 - y_2)}{(y_3 - y_2)(x_4 - x_2) - (y_4 - y_2)(x_3 - x_2)}$$

$$a_2 = \frac{(\min - \max)(x_4 - x_2)}{(y_3 - y_2)(x_4 - x_2) - (y_4 - y_2)(x_3 - x_2)}$$

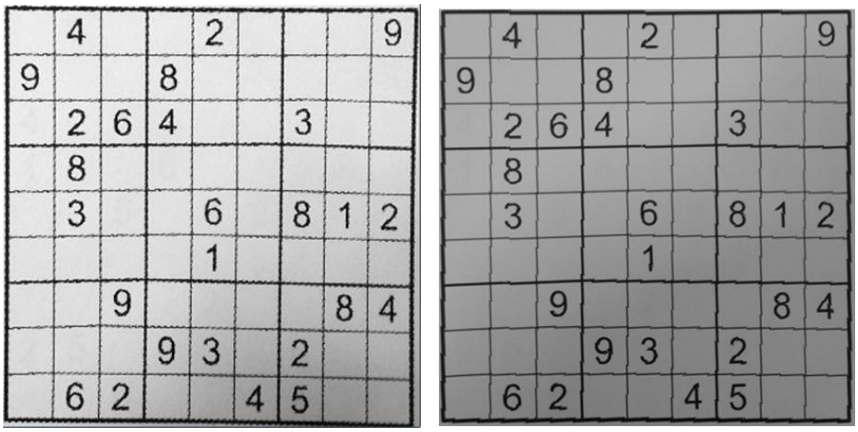
$$b_0 = \max + \frac{(\max - \min)(y_4 x_3 - x_4 y_3)}{(y_4 - y_3)(x_2 - x_3) - (y_2 - y_3)(x_4 - x_3)}$$

$$b_1 = \frac{-(\max - \min)(y_4 - y_3)}{(y_4 - y_3)(x_2 - x_3) - (y_2 - y_3)(x_4 - x_3)}$$

$$b_2 = \frac{(\max - \min)(x_4 - x_3)}{(y_4 - y_3)(x_2 - x_3) - (y_2 - y_3)(x_4 - x_3)}$$

Čia $\{x_i, y_i | \ i = 1, \dots 4\}$ atitinkamų kampų koordinatės; *max* ir *min* naujos koordinatės (2.15 pav.).

Dviejų afiniųjų transformacijų pritaikymo programos tekstas pateiktas priede Nr. 2. MatLab programos failas *affine_2.m*. Pritaikius šią transformaciją eksperimentiniams vaizdams gauti rezultatai pateikti 2.16 paveikslėlyje.

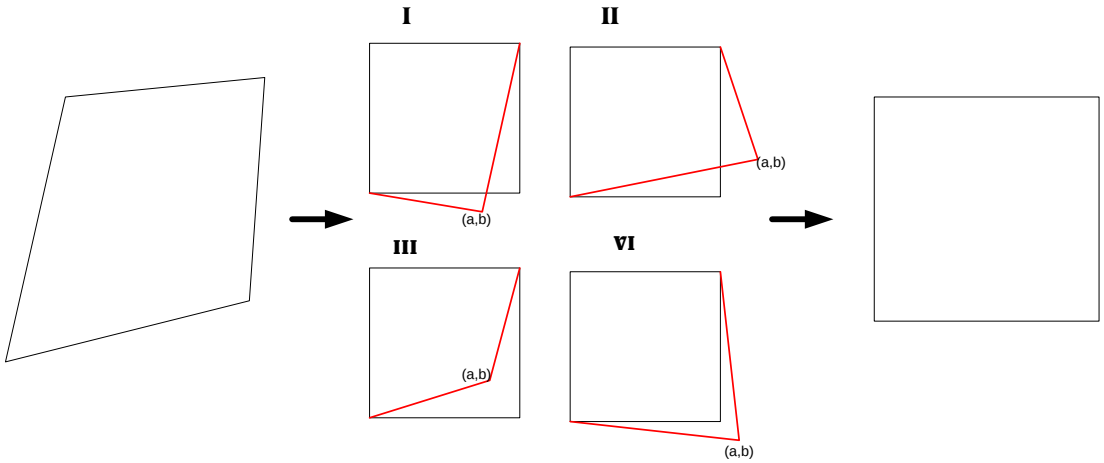


2.16 pav. Atkurti eksperimentiniai deformuoti vaizdai taikant dvi afiniąsias transformacijas

Kai kuriuose atkurtuose vaizduose matomas nežymus, kitiems žymus vaizdo tolygumo praradimas. Vaizdas atrodo lyg būtų sujungtas iš dviejų atskirų „gabalų“. Rezultatas netenkina, todėl šis metodas yra atmetamas.

3) Afiniosios ir tiksliniai parinktos transformacijos panaudojimas

Taip pat, deformuotą vaizdą siūloma atkurti naudojant afiniosios ir tiksliniai parinktos transformacijos superpoziciją. Visiems taškams *(x, y)* siūloma pritaikyti afiniąją transformaciją ir po to tiksliniai parinktas išraiškas, pervedančias tuos taškus į naujas pozicijas $(\tilde{x}, \tilde{y})$ (2.17 pav.).



2.17 pav. Vaizdo atkūrimo, taikant afiniąsias ir tiksliniai parinktas transformacijas, eiga

Išveskime šias lygtis. Pirmiausiai raskime *x* koordinatės transformacijos formulę. Tarkime turime tiesę tarp (x_{max}, y_{min}) ir (a, b) taškų. Bendroji tiesės lygtis (2.9 pav. žr. 2.13 skyrelį) aprašoma taip:

$$y = kx + m \tag{2.10}$$

Sudaroma lygčių sistema:

$$\begin{cases} b = ak + m \\ y_{min} = x_{max}k + m \end{cases} \tag{2.11}$$

Išsprendus (2.11) lygčių sistemą gaunami parametrai *k* ir *m*:

$$k = \frac{b - y_{min}}{a - x_{max}}$$

$$m = y_{min} - x_{max}\left(\frac{b - y_{min}}{a - x_{max}}\right)$$

Įstatome parametrus į 2.10 lygtį ir gaunamos ieškomos lygties formulės:

$$y = \frac{(b-y_{min})(x-x_{max})}{a-x_{max}} + y_{min} \tag{2.12a}$$

$$x = \frac{(a-x_{max})(y-y_{min})}{b-y_{min}} + x_{max} \tag{2.12b}$$

Tuomet sudaroma perėjimo formulė iš šios tiesės į mus dominančią tiesę, esančią tarp (x_{max}, y_{min}) ir (x_{max}, y_{max}) taškų. Sudaroma lygčių sistemą:

$$\begin{cases} x_{min} = kx_{min} + m \\ x_{max} = k\left(\frac{(a-x_{max})(y-y_{min})}{b-y_{min}} + x_{max}\right) + m \end{cases} \quad (2.13)$$

Išsprendus lygčių sistemą randami parametrai *k* ir *m* ir įstatomi į bendrąją tiesės lygtį (2.10). randama *x* koordinatės perėjimo lygtis:

$$\tilde{x} = \left(\frac{(x_{max}-x_{min})(b-y_{min})}{(a-x_{max})(y-y_{min})+(x_{max}-x_{min})(b-y_{min})}\right)(x - x_{min}) + x_{min} \quad (2.14)$$

Ieškant y koordinatės perėjimo formulės naudojamos šios tiesės, tarp (x_{min}, y_{max}) ir (a, b) taškų, lygtys:

$$y = \frac{(b-y_{max})(x-x_{min})}{a-x_{min}} + y_{max} \quad (2.15a)$$

arba

$$x = \frac{(a-x_{min})(y-y_{max})}{b-y_{max}} + x_{min} \quad (2.15b)$$

Sudaroma lygčių sistema:

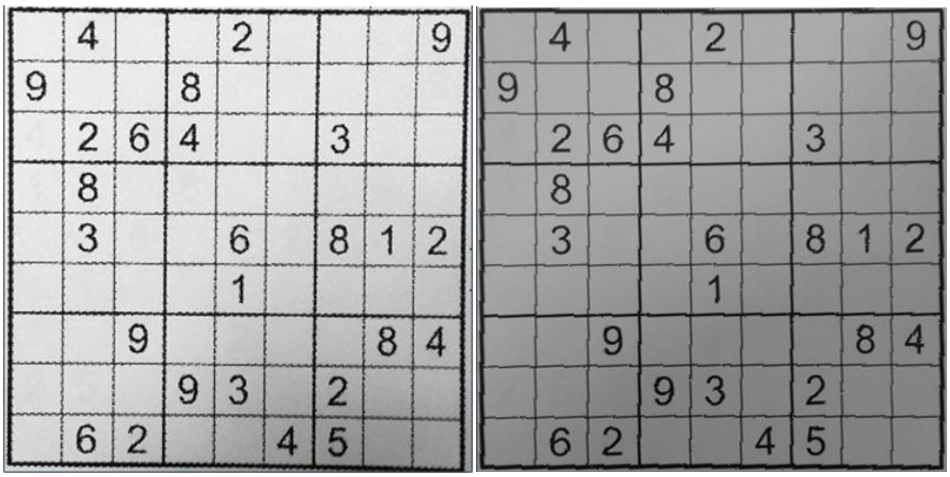
$$\begin{cases} y_{min} = ky_{min} + m \\ y_{max} = k\left(\frac{(b-y_{max})(x-x_{min})}{a-x_{min}} + y_{max}\right) + m \end{cases} \quad (2.16)$$

Analogiškai randama ir *y* koordinatės perėjimo lygtis:

$$\tilde{y} = \left(\frac{(y_{max}-y_{min})(a-x_{min})}{(b-y_{max})(x-x_{mix})+(y_{max}-y_{min})(a-x_{min})}\right)(y - y_{min}) + y_{min} \quad (2.17)$$

Čia (*a*, *b)* vis dar iškreipto vaizdo, po afiniosios transformacijos, dešiniojo apatinio kampo koordinatės, o (x_{max}, y_{max}) yra atkurto vaizdo dešiniojo apatinio kampo koordinatės.

Afiniosios ir tiksliniai parinktos transformacijų superpozicijos pritaikymo programos tekstas pateiktas priede Nr. 2. MatLab programos failas *affine_transf.m*. Pritaikius šią transformaciją eksperimentiniams vaizdams gauti rezultatai pateikti 2.18 paveikslėlyje.

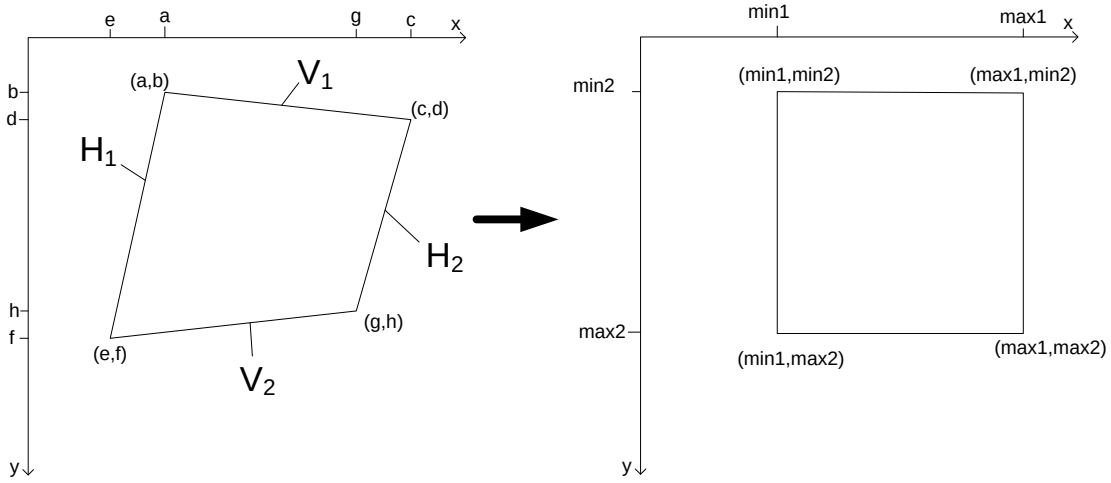


2.18 pav. Atkurti eksperimentiniai deformuoti vaizdai taikant afiniąją ir tiksliniai atrinktas transformacijas

Atkurti geometriškai deformuoti vaizdai su šia transformacijų superpoziciją pasižymi vientiesumu, priešingai nei po dviejų afiniųjų trnasformacijų taikymo.

2.2.2. VAIZDO TAŠKŲ PERĖJIMO LYGČIŲ TAIKYMAS

Remiantis tuo pačiu principu, jog tiesės lygtį galima iš vienos koordinačių ašies pervesti į kitą tiesę, išvedamos viso deformuoto vaizdo pervedimo, į taisyklingą keturkampį, formulės. Šis procesas pateiktas 2.19 paveikslėlyje.



2.19 pav. Geometriškai deformuoto vaizdo (keturšonio) perėjimas į kvadratą

Šios transformacijos *x* arba *y* koordinačių perėjimo formulių radimui reikės dviejų tiesės lygčių. Aiškumo dėlei naudojami 2.19 paveikslėlyje pateikti žymėjimai. Atlikus skaičiavimus pateikiamos visos keturios tiesių lygtys:

$$H_1 = \frac{(e-a)}{(f-b)}(y-b) + a \qquad (2.18a)$$

$$H_2 = \frac{(g-c)}{(h-d)}(y-d) + c \qquad (2.18b)$$

$$V_1 = \frac{(d-b)}{(c-a)}(x-a) + b \tag{2.19a}$$

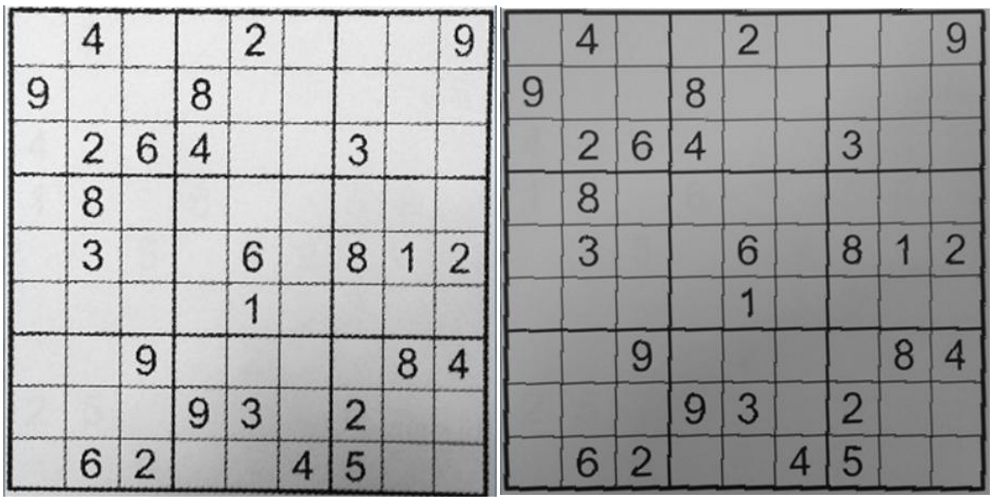
$$V_2 = \frac{(h-f)}{(g-e)}(x-e) + c \tag{2.19b}$$

Perėjimo lygtys išvedamos tokiu pat principu kaip ir tiksliniai parinktų transformacijų lygtys (žr. 2.2.1 skyrelį). Perėjimo lygtys aprašomos šitaip:

$$\tilde{x} = \frac{(max-min)(x-H_1)}{(H_2-H_1)} + min \tag{2.20}$$

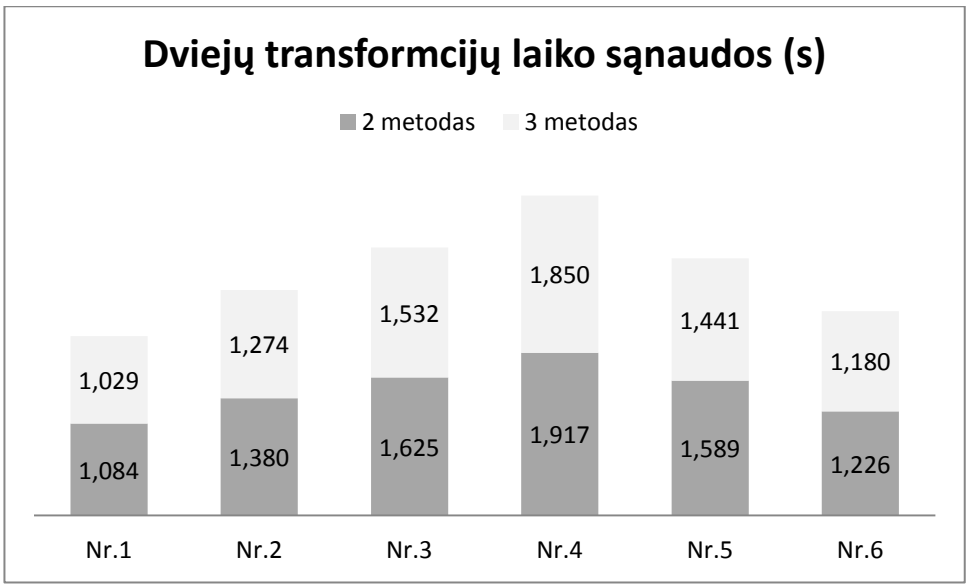
$$\tilde{y} = \frac{(max-min)(y-V_1)}{(V_2-V_1)} + min \tag{2.21}$$

Čia a, b, c, d, e, f, g, h – kampų koordinatės (2.19 pav.). Šios lygtys remiasi esamo ir būsimo objekto kampų koordinatėmis. Pritaikius šią transformaciją eksperimentiniams vaizdams gauti rezultatai pateikti 2.20 paveikslėlyje.



2.20 pav. Atkurti eksperimentiniai deformuoti vaizdai taikant vaizdo taškų perėjimo lygtis

Vaizdo kokybė po vaizdo taškų perėjimo lygčių taikymo prilygsta po afiniosios ir tiksliniai parinktos transformacijos superpozicijos taikymo gautiems vaizdams. Atliktas afiniosios ir tiksliniai parinktos transformacijų superpozicijos (2 metodas) ir vaizdo taškų perėjimo lygčių taikomo (3 metodas) laiko sąnaudų eksperimentas su šešiais bandomaisiais vaizdas. Gauti suvidurkinti rezultatai pateikiami 2.21 paveikslėlyje.



2.21 pav. Dviejų transformacijų laiko sąnaudų palyginimo grafikas

Iš tyrimo metu surinkų duomenų pastebėta, kad vaizdo taškų perėjimo lygčių metodas nežymiai (6%) greičiau atlieka deformuoto vaizdo atkūrimo procedūrą.

2.2.3. KITOS TRANSFORMACIJOS

Yra ir daugiau metodų atkurti deformuotą keturkampį į kvadratą, tokie kaip perspektyvinė, dvitiesio perėjimo ir polinominė transformacija. Transformacijų lygtys aprašytos 1.1 lentelėje (žr. 1.2.1 skyrelį). Plačiau apžvelkime kiekvieną atskirai.

1. *Perspektyvinė transformacija*

Perspektyvinė transformacija skirta ne tik paralelinius deformavimus atstatyti. Priešingai nei afinioji ir tikslinės transformacijų taikymo atveju perspektyvinė transformacija nėra tiesinė. Matricinė perspektyvinės transformacijos išraiška aprašoma taip [3]:

$$\begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2.22a}$$

$$\tilde{x} = Hx \tag{2.22b}$$

Čia H – homogeninė matrica. Perspektyvinė transformacija turi 8 laisvės laipsnių, todėl matricos H koeficientams apskaičiuoti reikalingos 8 lygtys. Vienas taškas ir jo perėjimo koordinatės atitinka dvi lygtis.

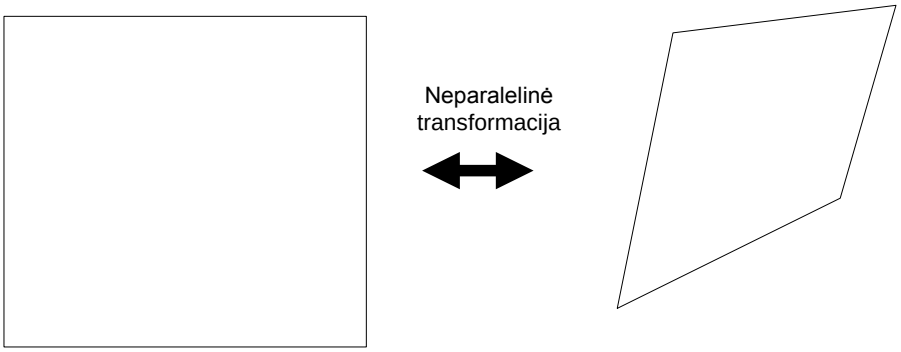
2. *Dvitiesio perėjimo transformacija*

Šio tipo perėjimas skirtas iš keturkampio pereiti į kvadratą ir atvirkščiai (2.20 pav.). Perėjimo lygtys aprašomos taip:

$$\tilde{x} = a_0 + a_1 x + a_2 y + a_3 xy; \quad (2.23)$$

$$\tilde{y} = b_0 + b_1 x + b_2 y + b_3 xy; \quad (2.24)$$

Dvitiesis perėjimas turi 8 laisvės laipsnius, todėl reikalingos 8 lygtys – keturių taškų koordinatės pervedamos į kitas keturių taškų koordinates.



2.22 pav. Neparalelinių transformacijų veikimas

3. Polinominė transformacija

Polinominė transformacija – tai bendra afiniosiojo ir tiesinio perėjimo forma aprašoma tokiomis lygtimis:

$$\tilde{x} = \sum_i \sum_j a_{ij} x^i y^j \quad (2.25a)$$

$$\tilde{y} = \sum_i \sum_j b_{ij} x^i y^j; \quad (2.25b)$$

Polinominė transformacija apima visas transformacijas, kurias gali sumodeliuoti daugianaris transformavimas.

2.2.4. INTERPOLIACIJA

Rastrinio vaizdo geometrinis transformavimas į kitą skaitmeninį vaizdą, siekiant kuo mažesnių kokybės nuostolių, nėra paprastas uždavinys. [1] Transformacijos metu vaizdo pikseliai arba jų koordinatės įgyja realųjį skaičių. Dėl šios priežasties naujame vaizde atsiranda neužpildytų pikselių, o tai blogina vaizdo kokybę. Tuščių taškų užpildymui galima naudoti įvairius interpoliacijos metodus. Vaizdo interpoliacija (lot. *inter* reiškia tarp, o *polire* – suderinti) – tai procesas, kurio metu konkrečiam atkurto vaizdo pikseliui randama originalaus vaizdo pikselio reikšmė ir / arba po geometrinės transformacijos nustatomos nežinomos pikselių reikšmės neužpildytuose naujo vaizdo vietose pagal žinomas aplinkines reikšmes.

Interpoliacijos algoritmai skirstomi į dvi kategorijas: adaptyvūs ir neadaptyvūs. [1] Adaptyvūs metodai keičia tai ką interpoliuoja. Neadaptyvūs metodai traktuoja visus pikselius vienodai. Pastarojo metodai yra šie: artimiausio kaimyno, dvitiesis, bikubinis, splainų, Lanczos ir kiti. Priklausomai nuo algoritmo sudėtingumo, interpoliuojant naudojama nuo 0 iki 256 (ar net daugiau) aplinki-

nių pikselių. [1] Kuo daugiau vertinama pikselių, tuo daugiau laiko sąnaudų reikia interpoliavimo procesui. Dažniausiai taikomi artimiausio kaimyno ir dvitiesis interpoliavimo metodai. Plačiau apžvelgiami būtent šie metodai.

1) Artimiausio kaimyno interpoliacijos metodas

Artimiausio kaimyno metodas (angl. *nearest-neighbor*), arba nulinio laipsnio interpoliacija, yra vienas iš paprasčiausių ir greičiausių interpoliacijos metodų. Jo užduotis rasti atitinkamo transformuoto vaizdo taškui artimiausią pikselį ir priskirti originalaus vaizdo reikšmę (2.23 pav.). Pikselis randamas suapvalinus artimiausias duoto taško koordinates iki sveikojo skaičiaus.

$$X_2(x_2, y_2) = X_2(x_2', y_2')$$

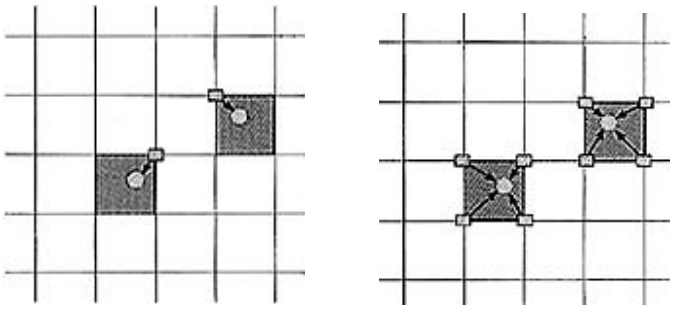
$$x_2' = round(x_2) = [x_2 + 0{,}5] \tag{2.26a}$$

$$y_2' = round(y_2) = [y_2 + 0{,}5] \tag{2.26b}$$

Artimiausio kaimyno interpoliacija gali būti atlikta ir kaip tiesinė sąsūka, kuri nekeičia pikselių pozicijos, bet keičia pikselio reikšmę. Artimiausio kaimyno atveju pikselio reikšmė nesikeičia, nes imama tokia pati originalaus vaizdo reikšmė esanti arčiausiai rasto taško. Artimiausio kaimyno interpoliacija pasižymi „dantytumo“ (angl. *blocking*) efektu. [1]

2) Dvitiesis interpoliacijos metodas

Dvitiesis (angl. *bilinear*) interpoliacijos metodas, dar vadinamas pirmojo laipsnio interpoliacija. Kaip ir artimiausio kaimyno atveju pasižymi skaičiavimų paprastumu.



2.23 pav. Artimiausio kaimyno ir dvitiesė interpoliacija

Dvitiesės interpoliacijos atveju surandami keturi originalaus vaizdo pikseliai, kurie yra artimiausi nagrinėjamo transformuoto vaizdo taškui. Nauja transformuoto vaizdo pikselio reikšmė apibrėžiama kaip dvitiesė keturių artimiausių originalaus vaizdo pikselių funkcija. (2.23 pav.) Dvitiesę interpoliaciją galima pakeist dviejų pirmos dimensijos branduolių *w* sandauga [1]:

$$W(x,y) = w(x)w(y) = \begin{cases} 1 - x - y - xy, kai\ 0 \le |x|, |y| < 1 \\ 0 \qquad\qquad kitu\ atveju \end{cases}$$

3) Sąsūka

Sąsūka taikoma neužpildytų atkurto vaizdo reikšmėms užpildyti. Matematinė sąsūkos išraiška vaizdams aprašoma taip [1]:

$$p^*_{xy} = \frac{\sum_{i=-m}^{m} \sum_{j=-m}^{m} p_{x+i,y+j} w_{i,j}}{\sum_{i=-m}^{m} \sum_{j=-m}^{m} w_{i,j}}$$

p^*_{xy} - nauja pikselio reikšmė, p_{xy} – vaizdo pikselio reikšmė, $w_{i,j}$ – filtro matricos elemento reikšmė. Filtro matrica *W* yra simetrinė. Darbe naudojamas filtras:

$$W = 1/4 \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Sąsūkos operacija slenka per visą atkurtą vaizdą, priskirdama neužpildytiems pikseliams naujas reikšmes.

MatLab aplinkoje esančios geometrinės transformacijos funkcijos jau turi integruotą interpoliavimą. Darbe naudojamoms transformacijoms taikomas artimiausio kaimyno interpoliavimas ir sąsūka, kaip interpoliacijos įrankis.

IŠVADOS

1. Deformuoto vaizdo atkūrimo procedūros sudarymas apima labai įvairias vaizdų apdorojimo sritis, būtent: vaizdo segmentavimo, analizės, atkūrimo ir pan.. Kiekviename atkūrimo procedūros etape naudojami atitinkami metodai. Universalaus sprendimo, visai geometriškai deformuotų vaizdų aibei, nėra.
2. Tyrimo metu nustatyta, jog susietų sričių išskyrimo vaizde procedūra, grindžiama tinkamai parinktais konvertavimo slenksčiais, leidžia efektyviai ir kokybiškai identifikuoti vaizde esančius atskirus objektus.
3. Harris-Stephens algoritmas tiksliau randa tik globalius deformuoto vaizdo kampus, o darbe siūloma skenavimo procedūra tiksliau juos lokalizuoja. Taip pat, pastaroji procedūra kampų lokalizavimą atlieka su mažesnėmis laiko sąnaudomis. (26,64% greičiau)
4. Deformuoto vaizdo geometrijos atkūrimas nuosekliai taikant dvi afiniąsias transformacijas (1 metodas) nėra priimtinas, kadangi atkurtas vaizdas praranda tolygumą, t.y. pasireiškia vaizdo „suliejimo“ iš dviejų atskirų gabalų efektas.
5. Deformuoto vaizdo geometrijai atkurti tikslinga taikyti afiniosios ir tiksliniai parinktos transformacijų superpoziciją (2 metodas). Po atkūrimo gauti vaizdai pasižymi aukšta kokybe.
6. Vaizdo taškų perkėlimo lygčių panaudojimas (3 metodas) vaizdo geometrijai atkurti sietinas su šiek tiek mažesnėmis (6%) laiko sąnaudomis, lyginant su antruoju metodu, bei nebloga atkurtų vaizdų kokybe.

LITERATŪRA

1. Girūta Kazakevičiūtė-Januškevičienė. „Rastrinių vaizdų geometrinės transformacijos“. Vilnius „Technika“ 2012.
2. Bakr Albatran. „Projective Geometry and Transformations in Space“ [interaktyvus]. 2005 liepa 1, [žiūrėta 2014-05-08]. Prieiga per internetą http://campar.in.tum.de/twiki/pub/Chair/TeachingSS05CVSeminar/02AlbatranHandout.pdf
3. Edmond Boyer. Lecture Notes: „Projective Geometry: A Short Introduction“. [interaktyvus], [žiūrėta 2014- 05-08]. Prieiga per internetą http://morpheo.inrialpes.fr/people/Boyer/Teaching/M2R/geoProj.pdf
4. Dr. P. Kasparaitis. Metodinė modulio „Skaitmeninis vaizdų apdorojimas“ medžiaga. [interaktyvus]. 2010, [žiūrėta 2014-05-08]. Prieiga per internetą http://www.mif.vu.lt/~pijus/SVA/sva.htm
5. Jonas Valantinas. Metodinė modulio „Matematinis skaitmeninių vaizdų apdorojimas“ medžiaga. Kaunas, 2014.
6. Chris Harris & Mike Stephens. „A Combined Corner and Edge Detector“ [interaktyvus]. 1988, [žiūrėta 2014-05-08]. Prieiga per internetą http://www.bmva.org/bmvc/1988/avc-88-023.pdf
7. Anil K. Jain. „Fundamentals of Digital Image Processing“. [interaktyvus]. 1988, [žiūrėta 2014-05-08]. Prieiga per internetą http://alok007.files.wordpress.com/2010/11/fundamentals_of_digital_image_processing_-_anil_k_jain.pdf
8. David Jacobs. „Image Gradients“. [interaktyvus] 2005, [žiūrėta 2014-05-08]. Prieiga per internetą http://www.cs.umd.edu/~djacobs/CMSC426/ImageGradients.pdf
9. http://www.visionaware.org/section.aspx?FolderID=8&SectionID=115&TopicID=519 [žiūrėta 2014-05-09]
10. http://photographylife.com/what-is-distortion [žiūrėta 2014-05-09]
11. http://www.aishack.in/2010/04/harris-corner-detector/ [žiūrėta 2014-05-09]
12. http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/ [žiūrėta 2014-05-09]
13. http://www.matdat.life.ku.dk/ia/sessions/session4-4up.pdf [žiūrėta 2014-05-10]

14. http://projektai.vu.lt/s-lasercenter/ppt/Andrius%20Melninkaitis%20LTC%20Seminaras%202006%2002%2023.pdf [žiūrėta 2014-05-10]

15. http://www.mathworks.se/products/image/description3.html [žiūrėta 2014-05-10]

16. Mikkel B.Stegmann. „Image Warping“. Denmark, 2001.

17. http://en.wikipedia.org/wiki/Thresholding_%28image_processing%29 [žiūrėta 2014-05-11]

18. http://en.wikipedia.org/wiki/Connected-component_labeling [žiūrėta 2014-05-12]

19. http://www.uio.no/studier/emner/matnat/ifi/INF4300/h09/undervisningsmateriale/hough09.pdf [žiūrėta 2014-05-12]

20. http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/ [žiūrėta 2014-05-20]

21. Rafael C. Gonzalez, Richard E. Woods. „Digital Image Processing“. [interaktyvus] 2008, [žiūrėta 2014-05-13]. Prieiga per internetą http://lit.fe.uni-lj.si/showpdf.php?lang=slo&type=doc&doc=dip&format=0

Priedas Nr. 1 Taikomosios matematikos konferencijos straipsnis „Apie geometriškai deformuotų gardelinių vaizdų atkūrimą“.

APIE GEOMETRIŠKAI DEFORMUOTŲ GARDELINIŲ VAIZDŲ ATKŪRIMĄ

I. Barušauskaitė, prof. J. Valantinas

Kauno technologijos universitetas

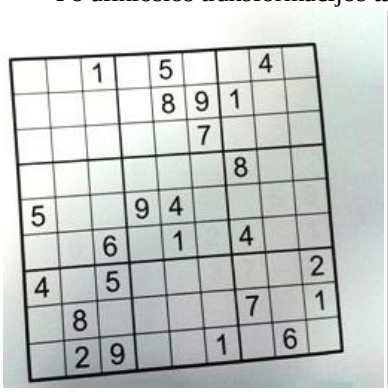
Mobiliosioms technologijos sparčiai žengiant į priekį, žmogaus gyvenimas tampa vis dinamiškesnis. Šiandien sunku įsivaizduoti darbovietę, mokslo įstaigą, pagaliau, modernų pramogų centrą, kur nebūtų panaudojami kompiuteriai, naujausios technologijos. Šiame straipsnyje pagrindinis dėmesys skiriamas geometriškai deformuotų vaizdų, sietinų su gardelinę struktūra pasižyminčiais žaidimais (sudoku, šaškėmis, šachmatais ir pan.), atkūrimui. Su geometriškai deformuotais vaizdais susiduriama įvairiose situacijose. Viena jų, kai žmogus užsimanęs sužaisti laikraštyje, stende ar brošiūroje pamatytą žaidimo variantą, pastarąjį išmaniojo telefono pagalba paprasčiausiai nufotografuoja, tuo sutaupydamas nemažai brangaus laiko. Straipsnyje siūlomos idėjos esmė – mobiliosios programėlės pagalba atkurti po fotografavimo gautą geometriškai deformuotą vaizdą, atpažinti vaizde esančius simbolius (skaitmenis, figūras ir pan.) ir sudaryti žmogui sąlygas jau po keleto sekundžių užbaigti pasirinktą žaidimo variantą.

Pirmoji problema su kuria susiduriama – tai netaisyklingo keturkampio vaizdo (nuotraukos) keitimas taisyklingu stačiakampiu vaizdu.

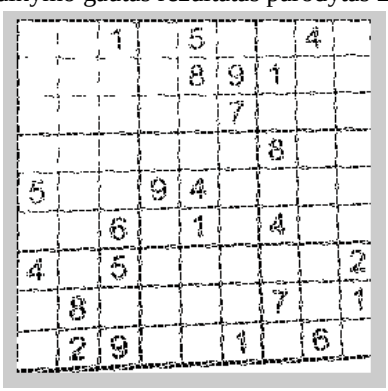
Straipsnyje siūloma deformuoto vaizdo (1 pav.) atkūrimą realizuoti dviem etapais. Pirmiausia, siūloma išlyginti kairiąją ir viršutinę netaisyklingo keturkampio briaunas, tam panaudojant dvimatę afiniąją transformaciją, būtent:

$$\begin{pmatrix}\tilde{x}\\ \tilde{y}\end{pmatrix} = \begin{pmatrix}a & b\\ c & d\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} + \begin{pmatrix}e\\ f\end{pmatrix} \qquad (1)$$

Po afiniosios transformacijos taikymo gautas rezultatas parodytas 2 pav.



1 pav. Originalus vaizdas.



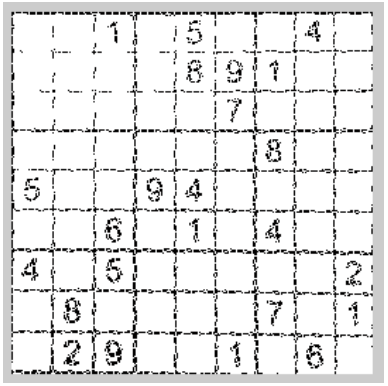
2 pav. Vaizdas po afiniosios transformacijos

Antra, siekiant išlyginti likusias dešiniąją ir apatinę vis dar deformuoto vaizdo (2 pav.) briaunas, visiems taškams (x, y) siūloma pritaikyti tiksliniai sudarytas išraiškas, pervedančias tuos taškus į naujas pozicijas $(\tilde{x}, \tilde{y})$, būtent:

$$\tilde{x} = \left(\frac{(c-1)(b-1)}{(a-c)(y-1)+(c-1)(b-1)}\right)(x-1)+1 \qquad (2)$$

$$\tilde{y} = \left(\frac{(c-1)(a-1)}{(b-c)(x-1)+(c-1)(a-1)}\right)(y-1)+1 \qquad (3)$$

čia a yra vaizdo (2 pav.) dešiniojo apatinio kampo abscisė, b yra to paties kampo ordinatė ir c yra atkurto vaizdo matmuo. Galutinai atkurtas vaizdas parodytas 3 pav.



3 pav. Atkurtas vaizdas

Po netaisyklingo keturkampio atstatymo galima bandyti atpažinti atkurtame paveiksle (3 pav.) išsibarsčiusius skaitmenis (figūras, jei tai kitas žaidimas). Ir tuomet atpažintus simbolius sustatyti į naują gardelinę struktūrą pasižymintį šabloninį paveikslą.

Priedas Nr. 2 Programos tekstas

Darbe naudojama MatLab programavimo kalba. Pateikiamas programos tekstas.

Transformacijos.m programos tekstas:

```
clc;
clear all;
close all;
% r nusako kokia transformacija bus naudojama

r=-1;
% r=0, afinioji
%r=-1, afinioji su tiksliniai parinkta transformacija
%r=-2 -dvi afiniosios;
%r=1 - perspective;
%r=2 vaizdo tasku pervedimo lygciu taikymas
% p nusako kaip aptinkami kaimpai

p=0;
%galimi variantai: p=0 - Zigzag arba lokalusis skenavimas; p=1 - Harris corned detector
dim=600; %i koki matmeni orientuojames
thresh_level=1; % galimi variantai [0.4 - 1]; esant tamsesniu kampu sumazinti si skaiciu

%nuskaitome paveiksliuka
X = imread('sudoku2.jpg');
imwrite(X,'sudoku256.pgm');
A = imread('sudoku256.pgm');
figure;imshow(A);
oldClass = class(A);
[M,N]=size(A);

%sumazina jei reikia
if M>dim || N>dim
    Res=dim./max(M,N);
    A=imresize(A,Res);
    [M,N]=size(A);
End

%gaunam dvejetaini 0,1 paveiksliuka:
B = im2bw(A, thresh_level*graythresh(A));
C=B;

% figure;imshow(B);
%-----BLOB dideli objektu aptikimo algoritmas------
C=blob(C); %Funkcija

%-----"Lokalusis (Zigzag) skenavimas ieskant kampu------
if p==0
    Kampai_i_j=ZigZag(C); %pirmoje eiluteje i, antorje j

%------"Harris" corner detector--------------------
elseif p==1
    Kampai_i_j=kampai(C); %pirmoje eiluteje i, antorje j
End

figure;imshow(A);
hold on;
plot(Kampai_i_j(2,:), Kampai_i_j(1,:), 'O', 'Color', 'g','LineWidth',2);
hold off;

%iskiriame mums aktualia vaizdo dali
Min_ordinate=min(Kampai_i_j(1,1),Kampai_i_j(1,2)); %12
Max_ordinate=max(Kampai_i_j(1,3),Kampai_i_j(1,4)); %34
```

```
Min_abscise=min(Kampai_i_j(2,1),Kampai_i_j(2,3)); %13
Max_abscise=max(Kampai_i_j(2,2),Kampai_i_j(2,4)); %24
%----iskerpame mus dominanti vaizda---
AA=imcrop(A,[Min_abscise-30 Min_ordinate-30 -Min_abscise+Max_abscise+50 -
Min_ordinate+Max_ordinate+50]);
[I,J]=size(AA);
DD = im2bw(AA, thresh_level*graythresh(A));
DD=blob(DD); % naudojam BLOB antra karta
oClass = class(DD);

%----surandame naujo vaizdo kampus-----------------
%-----"Zigzag skenavimas ieskant kampu-------------
if p==0
    Kampai_i_j=ZigZag(DD); %pirmoje eiluteje i, antorje j
%------"Harris" corner detector--------------------
elseif p==1
    Kampai_i_j=kampai(DD); %pirmoje eiluteje i, antorje j
end

%priskiriami kampu koordinates kintamiesiems
a=Kampai_i_j(2,1);c=Kampai_i_j(2,2);e=Kampai_i_j(2,3);g=Kampai_i_j(2,4);
b=Kampai_i_j(1,1);d=Kampai_i_j(1,2);f=Kampai_i_j(1,3);h=Kampai_i_j(1,4);
%--- randame naujo kvadrato kooridnates (didinam pagal ilgiausia krastine (kaires ir
visaus))----
if f-b<=c-a % (1) - a,b; (2) - c,d; (3) - e,f; (4) - g,h;
    min_=a;     max_=c;
else
    min_=b;    max_=f;
end

%-----Afinioji transformacija---------------
if r==0
    [M,n1,d1,n2,d2]=AFFINE_(Kampai_i_j,min_,max_);
    tform=maketform('affine',double(M));
    N_Image=imtransform(AA,tform);
    imshow(N_Image);
    N_Image=imcrop(N_Image,[ min_-2 min_-2 max_-min_+4 max_-min_+4]);
    N_Image = cast(N_Image,oldClass); %grazinama paveiksliuko klase
    figure;imshow(N_Image);

%----Afinioji ir tiksliniai parinkta transformacija---------
elseif r==-1
    N_Image=affine_transf(double(AA),Kampai_i_j,min_, max_);
    N_Image=imcrop(N_Image,[ min_-2 min_-2 max_-min_+6 max_-min_+6]);
    N_Image = cast(N_Image,oldClass); %grazinama paveiksliuko klase
    figure;imshow(N_Image);
    pic_new=Sasuka(N_Image);
    pic_new = cast(pic_new,oldClass);
    figure;imshow(pic_new);

%-----------------Afiniojosios dvi-------------------------
elseif r==-2
    N_Image=affine_2x(double(AA),Kampai_i_j,min_, max_);
    N_Image=imcrop(N_Image,[ min_-2 min_-2 max_-min_+6 max_-min_+6]);
    N_Image = cast(N_Image,oldClass); %grazinama paveiksliuko klase
    figure;imshow(N_Image);
    pic_new=Sasuka(N_Image);
    pic_new = cast(pic_new,oldClass);
    figure;imshow(pic_new);

%-------------Perspektyvine transformacija--------------------
elseif r==1 %kaip puikiai tinka projective transformacija
    tform=maketform('projective',[a b;c d;e f;g h],...
                    [min_ min_;max_ min_;min_ max_;max_ max_]);
    N_Image=imtransform(AA,tform); %pritaikom vaizdui AA
    imshow(N_Image);
    %randame atstatyto vaizdo koordinates iskirpimui
    if min_<=e
        min1_=e;
        max1_=max_+e-min_;
    else
```

```
        min1_=min_;
        max1_=max_;
    end
    if min_<=d
        min2_=c;
        max2_=max_+c-min_;
    else
        min2_=min_;
        max2_=max_;
    end
    %iskerpam vaizda
    N_Image=imcrop(N_Image,[ min1_-2 min2_-2 max1_-min1_+6 max2_-min2_+6]);
    N_Image = cast(N_Image,oldClass); %grazinama paveiksliuko klase
    figure;imshow(N_Image);

 %----Vaizdo tasku perejimo lygciu taikymas------------------
elseif r==2
   for y=1:I
      for x=1:J
        x1=(y-b)*(e-a)./(f-b)+a;
        x2=(y-d)*(g-c)./(h-d)+c;
        y1=(x-a)*(d-b)./(c-a)+b;
        y2=(x-e)*(h-f)./(g-e)+f;
        N_x=(x-x1)*(max_-min_)./(x2-x1)+min_; %naujas x
        N_y=(y-y1)*(max_-min_)./(y2-y1)+min_; %naujas y
        if round(N_x)>=1 &&round(N_y)>=1 %kad neisliptu uz ribos
            N_Image(round(N_y),round(N_x))=double(AA(y,x));
            N_Image(round(N_y+0.1),round(N_x+0.1))=double(AA(y,x));
        end
      end
   end
   N_Image=imcrop(N_Image,[ min_-2 min_-2 max_-min_+6 max_-min_+6]);
   N_Image = cast(N_Image,oldClass); %grazinama paveiksliuko klase
   figure;imshow(N_Image);
   %viena iš interpoliavimo metodu  sasuka
   pic_new=Sasuka(N_Image);
   pic_new = cast(pic_new,oldClass);
   figure;imshow(pic_new);
end
```

Toliau pateikiamos programoje naudotos funkcijos:

blob.m – susietų sričių paieška skenavimo būdų ir BLOB.

```
function [ A ] = blob( Image ) % Image ir A dvejetainis [0,1] vaizdas

Image=im2bw(Image);
BR=double(Image);
CC=BR;
A=BR;
imshow(CC);
count=5; %label skaiciukas
label(1)=count; %nurodom pirmaji label
q=0;
iter=0;
[N,M]=size(BR);

%-------------kai turim kvadrata----------------
if N==M
   for k=2:M+N-4
     if k<=M-1 %iki istrizaines kai dideja
         for s=k-1:-1:1
           i=(k+1)-s;
           j=s+1;
           if BR(i,j)~=1 %nelygu baltai
             iter=iter+1;
             if iter==1
                 BR(i,j)=label(1);%label priskyrimas
             end
             %surasykime BR(i,j) kaimynus
```

```
            q=0;
            turim_label=0;
            for g=0:2
               for f=0:2
                  Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                  if Kaimynai(g+1,f+1)>1
                      q=q+1; %label rodiklis
                      turim_label(q)=Kaimynai(g+1,f+1);
                      naujas_label=min(turim_label(1:q));
                  end
               end
            end
            for g=0:2
               for f=0:2
                  if Kaimynai(g+1,f+1)~=1
                      if q>0 % turim tarp kaimynu ir pacio branduolio labeliu
                          BR(i-1+g,j-1+f) =naujas_label;
                      else
                          count=count+1; %sukuriamas naujas label
                          q=1;
                          naujas_label=count;
                          BR(i-1+g,j-1+f)=naujas_label;
                                %ateiciai saugome label kad zinoti kokiu
                                %turime ir jais remiantis rasime
                                %didziausia plota uzimancius label
                          label(count-label(1)+1)=count;
                      end
                  end
               end
            end
          end
        end

%-------------------------------------------------------------------------
     elseif k>M-1 %nuo istrizaines kai mazeja
        for s=1:(N-1)+(M-1)-k-1
          i=(k-M)+2+s;
          j=M-s;
          if BR(i,j)~=1 %nelygu baltai
             iter=iter+1;
             %surasykime BR(i,j) kaimynus
             turim_label=0;
             q=0;
             for g=0:2
                for f=0:2
                   Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                   if Kaimynai(g+1,f+1)>1
                     q=q+1; %label rodiklis
                     turim_label(q)=Kaimynai(g+1,f+1);
                     naujas_label=min(turim_label(1:q));
                   end
                end
             end
             for g=0:2
                for f=0:2
                   if Kaimynai(g+1,f+1)~=1
                       if q>0 % turim tarp kaimynu ir pacio branduolio labeliu
                           BR(i-1+g,j-1+f) =naujas_label;
                       else
                           count=count+1; %sukuriamas naujas label
                           q=1;
                           naujas_label=count;
                           BR(i-1+g,j-1+f)=naujas_label;
                                %ateiciai saugome label kad zinoti kokiu
                                %turime ir jais remiantis rasime
                                %didziausia plota uzimancius label
                           label(count-label(1)+1)=count;
                       end
                   end
                end
             end
```

```
          end
        end
      end
    end
    A=BR;
end

% %-------------kai turim staciakampi-------------
if M<N
    for k=2:M+N-4
 %---------Pirma dalis kai dideja------------
         if k<=M-1
             for s=k-1:-1:1
               i=(k+1)-s;
               j=s+1;
               if BR(i,j)~=1 %nelygu baltai
                   iter=iter+1;
                   if iter==1
                       BR(i,j)=label(1);%label priskyrimas
                   end
                   %surasykime BR(i,j) kaimynus
                   q=0;
                   turim_label=0;
                   for g=0:2
                      for f=0:2
                         Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                         if Kaimynai(g+1,f+1)>1
                            q=q+1; %label rodiklis
                            turim_label(q)=Kaimynai(g+1,f+1);
                            naujas_label=min(turim_label(1:q));
                         end
                      end
                   end
                   for g=0:2
                      for f=0:2
                         if Kaimynai(g+1,f+1)~=1
                            if q>0 %turim tarp kaimynu ir pacio branduolio labeliu
                                 BR(i-1+g,j-1+f) =naujas_label;
                            else
                                 count=count+1; %sukuriamas naujas label
                                 q=1;
                                 naujas_label=count;
                                 BR(i-1+g,j-1+f)=naujas_label;
                                     %ateiciai saugome label kad zinoti kokiu
                                     %turime ir jais remiantis rasime
                                     %didziausia plota uzimancius label
                                 label(count-label(1)+1)=count;
                            end
                         end
                      end
                   end
                 end
              end

%---------pastovi dalis nekinta ilgis----------------------
         elseif k>M-1 && k<=N-1
           for s=M-1:-1:2
             i=(k+2)-s;
             j=s;
             if BR(i,j)~=1 %nelygu baltai
                iter=iter+1;
                %surasykime BR(i,j) kaimynus
                turim_label=0;
                q=0;
                for g=0:2
                  for f=0:2
                     Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                     if Kaimynai(g+1,f+1)>1
                         q=q+1; %label rodiklis
                         turim_label(q)=Kaimynai(g+1,f+1);
                         naujas_label=min(turim_label(1:q));
```

```
                    end
                  end
                end
                for g=0:2
                  for f=0:2
                    if Kaimynai(g+1,f+1)~=1
                      if q>0 %turim tarp kaimynu ir pacio branduolio labeliu
                          BR(i-1+g,j-1+f) =naujas_label;
                      else
                          count=count+1; %sukuriamas naujas label
                          q=1;
                          naujas_label=count;
                          BR(i-1+g,j-1+f)=naujas_label;
                              %ateiciai saugome label kad zinoti kokiu
                              %turime ir jais remiantis rasime
                              %didziausia plota uzimancius label
                          label(count-label(1)+1)=count;
                      end
                    end
                  end
                end
              end
            end

%---------Trumpeja----------------------------
          elseif k>N-1
            for s=1:(M-1)+(N-1)-k-1
               i=k-M+2+s;
               j=M-s;
               if BR(i,j)~=1 %nelygu baltai
                   iter=iter+1;
                   %surasykime BR(i,j) kaimynus
                   turim_label=0;
                   q=0;
                   for g=0:2
                      for f=0:2
                         Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                         if Kaimynai(g+1,f+1)>1
                             q=q+1; %label rodiklis
                             turim_label(q)=Kaimynai(g+1,f+1);
                             naujas_label=min(turim_label(1:q));
                          end
                      end
                   end
                   for g=0:2
                      for f=0:2
                         if Kaimynai(g+1,f+1)~=1
                            if q>0 %turim tarp kaimynu ir pacio branduolio labeliu
                                BR(i-1+g,j-1+f) =naujas_label;
                            else
                                count=count+1; %sukuriamas naujas label
                                q=1;
                                naujas_label=count;
                                BR(i-1+g,j-1+f)=naujas_label;
                                    %ateiciai saugome label kad zinoti kokiu
                                    %turime ir jais remiantis rasime
                                    %didziausia plota uzimancius label
                                    label(count-label(1)+1)=count;
                            end
                         end
                      end
                   end
               end

            end
          end  %cia  pabaiga if k<=M-1
      end % cia pabaiga for k=2:M+N-4

% %-------------kai turim staciakampi-------------
elseif M>N
    for k=2:M+N-4
```

```
if k<N-1
    for s=k-1:-1:1
        i=(k+1)-s;
        j=s+1;
        if BR(i,j)~=1 %nelygu baltai
            iter=iter+1;
            if iter==1
               BR(i,j)=label(1);%label priskyrimas
            end
            %surasykime BR(i,j) kaimynus
            q=0;
            turim_label=0;
            for g=0:2
                for f=0:2
                     Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                     if Kaimynai(g+1,f+1)>1
                         q=q+1; %label rodiklis
                         turim_label(q)=Kaimynai(g+1,f+1);
                         naujas_label=min(turim_label(1:q));
                     end
                end
            end
            for g=0:2
                for f=0:2
                     if Kaimynai(g+1,f+1)~=1
                         if q>0 % turim tarp kaimynu ir pacio branduolio labeliu
                             BR(i-1+g,j-1+f) =naujas_label;
                         else
                             count=count+1; %sukuriamas naujas label
                             q=1;
                             naujas_label=count;
                             BR(i-1+g,j-1+f)=naujas_label;
                                 %ateiciai saugome label kad zinoti kokiu
                                 %turime ir jais remiantis rasime
                                 %didziausia plota uzimancius label
                             label(count-label(1)+1)=count;
                         end
                     end
                end
            end
        end
    end
elseif k>=N-1 && k<=M-1
    for s=N-1:-1:2
         i=N+1-s;
         j=s+(k+1-N);
         if BR(i,j)~=1 %nelygu baltai
             iter=iter+1;
             %surasykime BR(i,j) kaimynus
             turim_label=0;
             q=0;
             for g=0:2
                for f=0:2
                     Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                     if Kaimynai(g+1,f+1)>1
                         q=q+1; %label rodiklis
                         turim_label(q)=Kaimynai(g+1,f+1);
                         naujas_label=min(turim_label(1:q));
                     end
                end
             end
             for g=0:2
                for f=0:2
                   if Kaimynai(g+1,f+1)~=1
                      if q>0 % turim tarp kaimynu ir pacio branduolio labeliu
                          BR(i-1+g,j-1+f) =naujas_label;
                      else
                          count=count+1; %sukuriamas naujas label
                          q=1;
                          naujas_label=count;
                          BR(i-1+g,j-1+f)=naujas_label;
```

```
                                    %ateiciai saugome label kad zinoti kokiu
                                    %turime ir jais remiantis rasime
                                    %didziausia plota uzimancius label
                                label(count-label(1)+1)=count;
                            end
                        end
                    end
                end
            end
        elseif k>M-1
            for s=1:(M-1)+(N-1)-k-1
                i=k-M+2+s;
                j=M-s;
                if BR(i,j)~=1 %nelygu baltai
                    iter=iter+1;
                    %surasykime BR(i,j) kaimynus
                    turim_label=0;
                    q=0;
                    for g=0:2
                       for f=0:2
                          Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                          if Kaimynai(g+1,f+1)>1
                              q=q+1; %label rodiklis
                              turim_label(q)=Kaimynai(g+1,f+1);
                              naujas_label=min(turim_label(1:q));
                          end
                       end
                    end
                    for g=0:2
                       for f=0:2
                          if Kaimynai(g+1,f+1)~=1
                              if q>0 % turim tarp kaimynu ir pacio branduolio labeliu
                                  BR(i-1+g,j-1+f) =naujas_label;
                              else
                                  count=count+1; %sukuriamas naujas label
                                  q=1;
                                  naujas_label=count;
                                  BR(i-1+g,j-1+f)=naujas_label;
                                      %ateiciai saugome label kad zinoti kokiu
                                      %turime ir jais remiantis rasime
                                      %didziausia plota uzimancius label
                                  label(count-label(1)+1)=count;
                              end
                          end
                       end
                    end
                end
            end
        end %cia pabaiga k<N-1
    end %cia pabaiga k=2:M+N-4
end% cia pabaiga if M<N

%-------antras skenavimas-----------------
for i=2:N-1
   for j=2:M-1
      if BR(i,j)~=1 %nelygu baltai
          turim_label=0;
          q=0;
          for g=0:2
             for f=0:2
                Kaimynai(g+1,f+1)=BR(i-1+g,j-1+f);
                if Kaimynai(g+1,f+1)>1
                    q=q+1; %label rodiklis
                    turim_label(q)=Kaimynai(g+1,f+1);
                    naujas_label=min(turim_label(1:q));
                end
             end
          end
          for g=0:2
             for f=0:2
```

```
                if Kaimynai(g+1,f+1)~=1
                    if q>0 % turim tarp kaimynu ir pacio branduolio labeliu
                        BR(i-1+g,j-1+f) =naujas_label;
                    end
                end
            end
         end
      end
   end
end

 A=BR;

for k=1:numel(label(:))
   Label=0;
   for i=1:N
      for j=1:M
         if BR(i,j)==double(label(k))
             Label=Label+1;
         end
         Labell(k,1)=label(k);
         Labell(k,2)=Label;
      end
   end
end

MAX1=max(Labell(2,:));


DuomSort=sortrows(Labell,-2);
Skaicius1=DuomSort(1,1);
for i=1:N
   for j=1:M
      if BR(i,j)==Skaicius1
          BR(i,j)=double(0);
      else
          BR(i,j)=double(1);
      end
   end
end
A=BR;
imshow(A);
```

Kita funkcija skurta kampų aptikimui:

kampai.m

```
function [ Kampai ] = kampai( Im ) %naudojamas po BLOB

Corner = corner(Im,190); % output x,y stulpeliai
c=size(Corner); %output i size, j size
Corner(:,3)=Corner(:,1)+Corner(:,2); % maziausia suma kampas (1), didziausia (4).
% imshow(Im);
% hold on
% plot(Corner(:,1), Corner(:,2), '.', 'Color', 'g')

DuomSort=sortrows(Corner,3);
%---virsutinis-kaire (1) kampas-----
Kampai(2,1)=DuomSort(1,1); %Kampai output i (y) eilute ir j (x) eilute
Kampai(1,1)=DuomSort(1,2);

%-----apatinis-desine (4) kampas ----
Kampai(2,4)=DuomSort(c(1,1),1);
Kampai(1,4)=DuomSort(c(1,1),2);

DuomSort=DuomSort(2:c-1,:);

%nagrinejamas virsutinis kampas
%jei x-y asi yra kazkas kairiau-auksciau tinka, jei nera paliekam kaip yra
```

```
if Kampai(2,1)>min(DuomSort(:,1))%kampai(2,1)=x
        X_k=1;
else    X_k=0;
end
if Kampai(1,1)>min(DuomSort(:,2))%kampai(1,1)=y
        X_a=1;
else    X_a=0;
end

%nagrinejamas apatinis kampas
%jei x-y asi yra kazkas desiniau-zemiau tinka, jei nera paliekam kaip yra
if Kampai(2,4)<max(DuomSort(:,1))%kampai(2,1)=x
        X_d=1;
else    X_d=0;
end
if Kampai(1,4)<max(DuomSort(:,2))%kampai(1,1)=y
        X_z=1;
else    X_z=0;
end

if X_k==1
    Duom=sortrows(Corner,1);
    for i=1:c(1,1)
        if Duom(i,1)<Kampai(2,1)
            Z_K(i,1)=Duom(i,1);
            Z_K(i,2)=Duom(i,2);
        end
    end
    if X_z==1
        Duom=sortrows(Z_K,-2);
        clear Z_K;
    for i=1:size(Duom(:,1))
        if Duom(i,2)>Kampai(1,4)
            Z_K(i,1)=Duom(i,1);
            Z_K(i,2)=Duom(i,2);
        end
    end
end

elseif X_z==1
        Duom=sortrows(Corner,-2);
        clear Z_K;
    for i=1:c(1,1)
        if Duom(i,2)>Kampai(1,4)
            Z_K(i,1)=Duom(i,1);
            Z_K(i,2)=Duom(i,2);
        end
    end

end
if X_a==1
    Duom=sortrows(Corner,2);
    for i=1:c(1,1)
        if Duom(i,2)<Kampai(1,1)
            A_D(i,1)=Duom(i,1);
            A_D(i,2)=Duom(i,2);
        end
    end
    if X_d==1
        Duom=sortrows(A_D,-1);
        clear A_D;
    for i=1:size(Duom(:,1))
        if Duom(i,1)>Kampai(2,4)
            A_D(i,1)=Duom(i,1);
            A_D(i,2)=Duom(i,2);
        end
    end
end

elseif X_d==1
        Duom=sortrows(Corner,-1);
```

```
        clear A_D;
    for i=1:c(1,1)
        if Duom(i,1)>Kampai(2,4)
            A_D(i,1)=Duom(i,1);
            A_D(i,2)=Duom(i,2);
        end
    end
end
if X_d==0 && X_a==0
    A_D=DuomSort;
    a_d=1;
elseif X_d==1 || X_a==1
    a_d=0;
    Duom=sortrows(sortrows(A_D,2),-1);
    A_D_kampas_x=Duom(1,1);
    A_D_kampas_y=Duom(1,2);
    for i=2:size(Duom(:,1))
       if X_d==1 && X_a==1
           if A_D_kampas_y>Duom(i,2)
              if A_D_kampas_x<Duom(i,1)
                  A_D_kampas_x=Duom(i,1);
                  A_D_kampas_y=Duom(i,2);
              end
           end
       elseif X_a==1
           if A_D_kampas_y>Duom(i,2)
               A_D_kampas_x=Duom(i,1);
               A_D_kampas_y=Duom(i,2);
           end
       elseif X_d==1
           if A_D_kampas_x<Duom(i,1)
               A_D_kampas_x=Duom(i,1);
               A_D_kampas_y=Duom(i,2);
           end
       end
  end
Kampai(2,2)=A_D_kampas_x;
Kampai(1,2)=A_D_kampas_y;
End

if X_k==0 && X_z==0
    Z_K=DuomSort;
    z_k=1;
elseif X_k==1 || X_z==1
    z_k=0;
    Duom=sortrows(sortrows(Z_K,1),-2);
    Z_K_kampas_x=Duom(1,1);
    Z_K_kampas_y=Duom(1,2);
    for i=2:size(Duom(:,1))
        if X_k==1 && X_z==1
            if Z_K_kampas_y<Duom(i,2)
                if Z_K_kampas_x>Duom(i,1)
                    Z_K_kampas_x=Duom(i,1);
                    Z_K_kampas_y=Duom(i,2);
                end
            end
        elseif X_k==1
            if Z_K_kampas_x>Duom(i,1)
                Z_K_kampas_x=Duom(i,1);
                Z_K_kampas_y=Duom(i,2);
            end
        elseif X_z==1
            if Z_K_kampas_y<Duom(i,2)
                Z_K_kampas_x=Duom(i,1);
                Z_K_kampas_y=Duom(i,2);
            end
        end
    end
Kampai(2,3)=Z_K_kampas_x;
Kampai(1,3)=Z_K_kampas_y;
```

```
end
% figure;imshow(Im);
% hold on;
% plot(Kampai(2,:), Kampai(1,:), 'O', 'Color', 'g','LineWidth',2);
% hold off;
End
```

Lokalusis skenavimas:

ZigZag.m

```
function [ Kampai ] = ZigZag( A ) %A pradine vaizdo matrica

B=double(A);
[I, J]=size(B);
sk=0;k=1;
KAS=0; %ieskom juodo ar balto

%-----virsus desine-----
while sk==0
   for j=k:-1:1
      L=[(k+1)-j j];
      if B((k+1)-j,j)==KAS
          sk=sk+1;
          Kampas1(1,sk)=(k+1)-j;
         Kampas1(2,sk)=j;
      end
   end
   if sk==0
       k=k+1;
   end
end

    if sk>=2
        KAMPAI1(:,1)=Kampas1(1,:);
        KAMPAI1(:,2)=Kampas1(2,:);
        SORT=sortrows(KAMPAI1,-1);
        Kampas1(1,1)=SORT(1,1);
        Kampas1(2,1)=SORT(1,2);
    end
 Kampas1; %i j  virsus desine

 %----virsus kaire----
 sk=0; k=1;
 while sk==0
    for j=1:k
       L=[j  (J-k)+j];
       H=B(j,(J-k)+j);
       if B(j,(J-k)+j)==KAS
           sk=sk+1;
           Kampas2(2,sk)=(J-k)+j;
           Kampas2(1,sk)=j;
       end
    end
    if sk==0
        k=k+1;
    end

end

 if sk>=2
     KAMPAI2(:,1)=Kampas2(1,:);
     KAMPAI2(:,2)=Kampas2(2,:);
     SORT=sortrows(KAMPAI2,-1);
     Kampas2(1,1)=SORT(1,1);
     Kampas2(2,1)=SORT(1,2);
end
 Kampas2;%i j virsus kaire

  %----apacia desine----
```

```
  sk=0; k=1;
 while sk==0
    for j=1:k
       L=[(I-k)+j j ];
       if B((I-k)+j,j)==KAS
           sk=sk+1;
           Kampas3(1,sk)=(I-k)+j;
           Kampas3(2,sk)=j;
       end
    end
    if sk==0
        k=k+1;
    end

 end
    if sk>=2
        KAMPAI(:,1)=Kampas3(1,:);
        KAMPAI(:,2)=Kampas3(2,:);
        SORT=sortrows(KAMPAI,-2);
        Kampas3(1,1)=SORT(1,1);
        Kampas3(2,1)=SORT(1,2);
    end
 Kampas3;%i j apacia desine


 %-----apacia kaire----
  sk=0; k=1;
 while sk==0
    for j=1:k
       L=[(I-k)+j J+k-(j+k-1) ];
       if B((I-k)+j,J+k-(j+k-1))==KAS
           sk=sk+1;
           Kampas4(1,sk)=(I-k)+j;
           Kampas4(2,sk)=J+k-(j+k-1);
       end
    end
    if sk==0
        k=k+1;
    end

end
   if sk>=2
       KAMPAI(:,1)=Kampas4(1,:);
       KAMPAI(:,2)=Kampas4(2,:);
       SORT=sortrows(KAMPAI,1);
       Kampas4(1,1)=SORT(1,1);
       Kampas4(2,1)=SORT(1,2);
   end
 Kampas4;%i j apacia kaire

Kampai(:,1)=Kampas1(:,1); %virsus desine
Kampai(:,2)=Kampas2(:,1); %virsus kaire
Kampai(:,3)=Kampas3(:,1); %apacia desine
Kampai(:,4)=Kampas4(:,1); %apacia kaire

% figure;imshow(A);
% hold on;
% plot(Kampai(2,:), Kampai(1,:), 'O', 'Color', 'g','LineWidth',2);
% hold off;
end
```

Afiniosios transformacijos matricos radimas:

AFFINE_.m

```
function [ Matrica, n1, d1, n2, d2 ] = AFFINE_( Kampas, min_,max_ )
syms a b c d e f;
n_y=[min_ min_ max_ max_];
n_x=[min_ max_ min_ max_];
SS=solve(a*Kampas(2,1:3)+b*Kampas(1,1:3)+e==n_x(1,1:3),a,b,e);
```

```
SW11 = [SS.a, SS.b, SS.e];
SS=solve(c*Kampas(2,1:3)+d*Kampas(1,1:3)+f==n_y(1,1:3),c,d,f);
SW12 = [SS.c, SS.d, SS.f];

double(SW11);
double(SW12);
[n1, d1] = numden(SW11);
[n2, d2] = numden(SW12);
Matrica=[n1(1,1)/d1(1,1) n1(1,2)/d1(1,2) 0; n2(1,1)/d2(1,1) n2(1,2)/d2(1,2) 0;
n1(1,3)/d1(1,3) n2(1,3)/d2(1,3) 1];

end
```

Afiniosios ir tiksliniai parinktos transformacijų superpozicijų taikymas:

affine_transf.m

```
function [ Vaizdas ] = affine_transf( A,Kampai_i_j, min_, max_ )

[I,J]=size(A);

x1=Kampai_i_j(2,1);x2=Kampai_i_j(2,2);x3=Kampai_i_j(2,3);x4=Kampai_i_j(2,4);
y1=Kampai_i_j(1,1);y2=Kampai_i_j(1,2);y3=Kampai_i_j(1,3);y4=Kampai_i_j(1,4);
        %randa afiniosios transformacijos koeficientus
        a=-(max_-min_)*(y2-y1)/((y3-y1)*(x2-x1)-(y2-y1)*(x3-x1));
        b=(max_-min_)*(x2-x1)/((y3-y1)*(x2-x1)-(y2-y1)*(x3-x1));
        e=min_+(max_-min_)*(y2*x1-x2*y1)/((y3-y1)*(x2-x1)-(y2-y1)*(x3-x1));

        c=-(max_-min_)*(y3-y1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));
        d=(max_-min_)*(x3-x1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));
        f=min_+(max_-min_)*(y3*x1-x3*y1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));

        a1=a*x4+b*y4+e;
        b1=c*x4+d*y4+f;

for y=1:I
   for x=1:J
      x_=a*x+b*y+e;
      y_=c*x+d*y+f;

      if round(x_)>=1 &&round(y_)>=1
%           M(round(x_),round(y_))=double(A(y,x));  %po afniosios
%           M(round(x_+0.5),round(y_+0.5))=double(A(y,x)); %po afiniosios

            N_x=((max_-min_)*(b1-min_)/((a1-max_)*(y_-min_)+(max_-min_)*(b1-min_)))*(x_-
min_)+min_;
            N_y=((max_-min_)*(a1-min_)/((b1-max_)*(x_-min_)+(max_-min_)*(a1-min_)))*(y_-
min_)+min_;
            if round(N_x)>=1 &&round(N_y)>=1
                N_Image(round(N_x),round(N_y))=double(A(y,x));
                N_Image(round(N_x+0.5),round(N_y+0.5))=double(A(y,x));
            end
      end
   end
end
Vaizdas=N_Image;
end
```

Dviejų afiniųjų taikymas:

affine_2.m

```
function [ Vaizdas ] = affine_2x( A, Kampai_i_j, min_, max_ )
[I,J]=size(A);

x1=Kampai_i_j(2,1);x2=Kampai_i_j(2,2);x3=Kampai_i_j(2,3);x4=Kampai_i_j(2,4);
y1=Kampai_i_j(1,1);y2=Kampai_i_j(1,2);y3=Kampai_i_j(1,3);y4=Kampai_i_j(1,4);

a=-(max_-min_)*(y3-y1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));
```

```
b=(max_-min_)*(x3-x1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));
e=min_+(max_-min_)*(y3*x1-x3*y1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));

c=-(min_-max_)*(y2-y1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));
d=(min_-max_)*(x2-x1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));
f=min_+(min_-max_)*(y2*x1-x2*y1)/((y2-y1)*(x3-x1)-(y3-y1)*(x2-x1));

%randa antros afiniosios transformacijos koeficientus
a2=-(min_-max_)*(y4-y2)/((y3-y2)*(x4-x2)-(y4-y2)*(x3-x2));
b2=(min_-max_)*(x4-x2)/((y3-y2)*(x4-x2)-(y4-y2)*(x3-x2));
e2=max_+(min_-max_)*(y4*x2-x4*y2)/((y3-y2)*(x4-x2)-(y4-y2)*(x3-x2));

c2=-(max_-min_)*(y4-y3)/((y4-y3)*(x2-x3)-(y2-y3)*(x4-x3));
d2=(max_-min_)*(x4-x3)/((y4-y3)*(x2-x3)-(y2-y3)*(x4-x3));
f2=max_+(max_-min_)*(y4*x3-x4*y3)/((y4-y3)*(x2-x3)-(y2-y3)*(x4-x3));

for y=1:I
   x_max=(y-y2)*(x3-x2)/(y3-y2)+x2;
   if x_max>1
       for x=1:J
          if x<=round(x_max)
              x_=a*x+b*y+e;
              y_=c*x+d*y+f;
              if round(x_)>=1 &&round(y_)>=1
                  N_Image(round(y_),round(x_))=double(A(y,x));
                  N_Image(round(y_+0.1),round(x_+0.1))=double(A(y,x));
              end
          else
              x_=a2*x+b2*y+e2;
              y_=c2*x+d2*y+f2;
              if round(x_)>=1 &&round(y_)>=1
                  N_Image(round(y_),round(x_))=double(A(y,x));
                  N_Image(round(y_+0.1),round(x_+0.1))=double(A(y,x));
              end
          end
       end
   else
       for xxx=1:J
            x_=a2*xxx+b2*y+e2;
            y_=c2*xxx+d2*y+f2;
            if round(x_)>=1 &&round(y_)>=1
                N_Image(round(y_),round(x_))=double(A(y,xxx));
                N_Image(round(y_+0.1),round(x_+0.1))=double(A(y,xxx));
            end
       end
   end
end

Vaizdas=N_Image;
end
```