KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Vytautas Milbutas

DISKREČIOSIOS DAUBECHIES D4 TRANSFORMACIJOS SKAIČIAVIMO ALGORITMŲ REALIZACIJA IR TYRIMAS

Magistro darbas

Vadovas
prof. dr. J. Valantinas

KAUNAS, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N.Listopadskis
2012 06 02

DISKREČIOSIOS DAUBECHIES D4 TRANSFORMACIJOS SKAIČIAVIMO ALGORITMŲ REALIZACIJA IR TYRIMAS

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
prof. dr. J. Valantinas
2012 06 01

Recenzentas
V. Sekliuckis
2012 06 01

Atliko
FMMM-0 gr. stud.
V. Milbutas
2012 05 30

KAUNAS, 2012

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Rimantas Rudzkis, profesorius (VU MII)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai:

Jonas Valantinas, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Milbutas V. Implementation and analysis of computational algorithms for the discrete Daubechies (D4) transform: Master's work in Applied Mathematics / supervisor prof. dr. J. Valantinas; Department of Applied Mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2012. - 50 p.

SUMMARY

The discrete wavelet (Haar, Le Gall, Daubechies etc.) transforms have been used effectively in many vital applications such as transient signal analysis, numerical analysis, computer vision, image compression and in other audio visual applications.

The discrete Daubechies D4 wavelet transform has been chosen for this work. Two versions of the transform are reviewed in general, representing and analyzing their main properties.

To evaluate the energy compaction property of the modified version of the D4 transform (with respect to its basic version), the hyperbolic image filters and image smoothness estimates have been employed.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Didėjant skaitmeninių vaizdų raiškai, plečiantis vaizdų panaudojimui pramoninėje srityje (HD televizija, fotomontažas ir pan.) būtina spręsti efektyvaus skaitmeninių vaizdų suglaudinimo problemą bei tobulinti tam skirtas matematines priemones. Grafinės informacijos perdavimui riboto pralaidumo kanalais (internetu) gali pasitarnauti lokaliai progresyvus skaitmeninių vaizdų kodavimas (atsiunčiama ir atskleidžiama tik aktuali informacija).

Naujausi, efektyviausi ir populiariausi dabartiniu laikmečiu skaitmeninių vaizdų (signalų) apdorojimo (suglaudinimo) algoritmai realizuojami spektrinėje bangelių (Haar, LeGall, Daubechies) srityje. Būtent diskrečiosios bangelės leidžia pasiekti gana aukštą vaizdų suglaudinimo efektą, išsaugant vizualiai priimtiną po kodavimo atkurtų vaizdų kokybę.

Šiame darbe susipažįstama su diskrečiąja Daubechies D4 bangelių transformacija bei įvertinama jos specifinės savybės.

Darbo tikslas – sudaryti bazinės diskrečiosios Daubechies (D4) transformacijos ir modifikuotos D4 transformacijos (dalinė vaizdo blokų dekoraliacija aukštuose dažniuose) apskaičiavimo dvimačiams skaitmeniniams vaizdams procedūras bei ištirti šių transformacijų savybes panaudojant hiperbolinių vaizdo filtrus bei pasitelkiant vaizdo glodumo įverčius.

Šia tema jau buvo skaitytas pranešimas „Diskrečioji Daubechies D4 transformacija“ X-joje studentų konferencijoje „Taikomoji matematika” (2012 m.).

1. DISKREČIOSIOS BANGELIŲ TRANSFORMACIJOS (DBT)

1.1. BENDROSIOS BANGELIŲ SAVYBĖS

Tolydžiosios bangelių transformacijos išraiška yra tokia

$$\gamma(s,\tau) = \int f(t)\psi^{*}_{s,\tau}(t)dt \qquad \textbf{(1.1)}$$

čia $\psi_{s,\tau}$ - bazinės bangelių funkcijos, s ir τ - nauji argumentai (po bangelių transformacijos).

Atvirkštinė bangelių transformacija nusakoma išraiška:

$$f(t) = \int\int \gamma(s,\tau)\psi_{s,\tau}(t)d\tau ds \qquad \textbf{(1.2)}$$

Bangelės generuojamos iš vienos bazinės bangelės $\psi(t)$, kuri vadinama motinine bangele, t.y.

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \qquad \textbf{(1.3)}$$

kur s yra mastelio koeficientas, τ – poslinkio koeficientas, o $s^{-1/2}$ energijos normalizavimo koeficientas.

Pagrindinės bangelių savybės yra tinkamumas ir reguliarumas. Trumpai jas aptarti.

Funkcija $\psi(t)$ tenkina tinkamumo savybę, jei

$$\int \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < +\infty, \qquad \textbf{(1.4)}$$

kur $\Psi(\omega)$ žymi funkcijos $\psi(t)$ Furjė transformaciją. Iš (1.4) nelygybės tiesiogiai išplaukia, kad Furjė transformacija $\Psi(\omega)$ prilygsta nuliui, kai dažnis ω yra lygus 0, t.y.

$$|\Psi(\omega)|^2\Big|_{\omega=0} = 0. \qquad \textbf{(1.5)}$$

Šis faktas gana svarbus, kadangi iš jo tiesiogiai išplaukia, jog bangelės turi apribotą dažnių juostą.

Kita vertus,

$$\int \psi(t)dt = 0, \qquad \textbf{(1.6)}$$

t.y. funkcija $\psi(t)$ yra „bangelė“.

Reguliarumo savybė siejama su bangelių funkcijų konvergavimo greičio pagerinimu. Bangelės funkcija turi būti glodi ir koncentruota tiek laiko, tiek dažnio srityse. Reguliarumas - gana sudėtinga sąvoka. Jos paaiškinimui ir interpretacijai pasinaudosime nykstančiais pradiniais momentais.

Bangelės funkciją (1.1) išskleidžiame Teiloro eilute taško $t = 0$ aplinkoje (imkime $\tau = 0$). Tada

$$\gamma(s,0) = \frac{1}{\sqrt{s}}\left[\sum_{p=0}^{n} f^{(p)}(0)\int \frac{t^p}{p!}\psi\left(\frac{t}{s}\right)dt + O(n+1)\right], \quad \textbf{(1.7)}$$

kur $f^{(p)}$ - funkcijos *p*-osios eilės išvestinė, o $O(n+1)$ - skleidinio liekamasis narys. Pažymėję bangelės *p*-osios eilės pradinį momentą M_p, t.y.

$$M_p = \int t^p \psi(t)dt, \quad \textbf{(1.8)}$$

(1.7) išraišką galime perrašyti taip

$$\gamma(s,0) = \frac{1}{\sqrt{s}}\left[f(0)M_o s + \frac{f^{(1)}(0)}{1!}M_1 s^2 + \frac{f^{(2)}(0)}{2!}M_1 s^3 + \ldots + \frac{f^{(n)}(0)}{n!}M_n s^{n+1} + O(s^{n+2})\right] \quad \textbf{(1.9)}$$

Remiantis „tinkamumo“ savybe, galime teigti, kad $M_0 = 0$. Vadinasi, (1.9) išraiškos pirmasis narys lygus nuliui. Jeigu pavyktų ir kitus momentus (iki *n*-tosios eilės) prilyginti nuliui, tada bangelių transformacijos koeficientai $\gamma(s,\tau)$ tolydžiam signalui $f(t)$ gestų greičiu s^{n+2}. Beje, praktiniuose taikymuose pakanka, kad pradiniai momentai būtų artimi nuliui.

Norint realizuoti tolydžiasias bangeles ir jų bazėje sukonstruotas diskrečiąsias transformacijas praktikoje, visų pirma bangeles reikia diskretizuoti.

1.2. DISKREČIOJI DAUBECHIES D4 TRANSFORMACIJA: SAVYBĖS, SKAIČIAVIMO ALGORITMAI

1.2.1 D4 REALIZAVIMO YPATUMAI

Imkime duomenų vektorių $X = (x_0 x_1 x_2 \ldots x_{N-1})^T$, $N = 2^n$, $n \in \square$. Disktrečioji D4 transformacija šiam vektoriui, paprastai gaunama taikant griežtai apibrėžtas interacines procedūras. Atskirai iteracijai (tarkime, *i*-tajai ($i \in \{1,2,\ldots,n\}$)) realizuoti panaudojama $(n-i+1)$-tos eilės transformacijų matrica $T_{D4}(n-i+1)$, kurią sudaro žemo ir aukšto dažnio filtrai. Pastebėsime, jog žemo dažnio filtras gaunamas, įvertinus su šia transformacija susijusios mastelio funkcijos koeficientus – rinkinuką

$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$, $h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$, $h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$, $h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$, o aukšto dažnio filtas gaunamas, įvertinus bangelės funkcijos koeficientus – rinkinuką $g_0 = h_3$, $g_1 = -h_2$, $g_2 = h_1$, $g_3 = -h_0$. Atskiru atveju D4 transformacijos matricos turi pavidalą:

$$T_{D4}(1) = \begin{pmatrix} h_0 & h_1 \\ g_0 & g_1 \end{pmatrix} \begin{matrix} h_2 & h_3 \\ g_2 & g_3 \end{matrix}$$

$$T_{D4}(2) = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \\ 0 & 0 & h_0 & h_1 \\ 0 & 0 & g_0 & g_1 \end{pmatrix} \begin{matrix} h_2 & h_3 \\ g_2 & g_3 \end{matrix}$$

$$T_{D4}(3) = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_0 & h_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 \end{pmatrix} \begin{matrix} h_2 & h_3 \\ g_2 & g_3 \end{matrix}$$

...

$$T_{D4}(n) = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & \cdots & 0 & 0 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & h_0 & h_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & g_0 & g_1 \end{pmatrix} \begin{matrix} h_2 & h_3 \\ g_2 & g_3 \end{matrix}$$

Po *n* iteracijų gaunamas diskretusis duomenų vektoriaus *X* spektras

$$Y_X = \left(Y(0)Y(1)\ldots Y(N-1)\right)^T; \qquad \textbf{(1.10)}$$

Atvirkštinė diskrečioji D4 transformacija irgi realizuojama taikant iteracines procedūras. Skirtumas tas, jog atskirų iteracijų metu panaudojamos matricos, užtikrinančios grįžimą nuo diskrečiojo D4 spektro *Y* prie pradinio duomenų vektoriaus *X* .

Įvairių eilių matricų išraiškos tokios:

$$T'_{D4}(1) = \begin{matrix} h_2 & g_2 \\ h_3 & g_3 \end{matrix}\begin{pmatrix} h_0 & g_0 \\ h_1 & g_1 \end{pmatrix};$$

$$T'_{D4}(2) = \begin{matrix} h_2 & g_2 \\ h_3 & g_3 \end{matrix}\begin{pmatrix} h_0 & g_0 & 0 & 0 \\ h_1 & g_1 & 0 & 0 \\ h_2 & g_2 & h_0 & g_0 \\ h_3 & g_3 & h_1 & g_1 \end{pmatrix}$$

$$T'_{D4}(3) = \begin{matrix} h_2 & g_2 \\ h_3 & g_3 \end{matrix}\begin{pmatrix} h_0 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_2 & g_2 & h_0 & g_0 & 0 & 0 & 0 & 0 \\ h_3 & g_3 & h_1 & g_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2 & g_2 & h_0 & g_0 & 0 & 0 \\ 0 & 0 & h_3 & g_3 & h_1 & g_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_2 & g_2 & h_0 & g_0 \\ 0 & 0 & 0 & 0 & h_3 & g_3 & h_1 & g_1 \end{pmatrix}$$

...

$$T'_{D4}(n) = \begin{matrix} h_2 & g_2 \\ h_3 & g_3 \end{matrix}\begin{pmatrix} h_0 & g_0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ h_1 & g_1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ h_2 & g_2 & h_0 & g_0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ h_3 & g_3 & h_1 & g_1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & h_2 & g_2 & h_0 & g_0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & h_3 & g_3 & h_1 & g_1 \end{pmatrix}$$

Atkreipiame dėmesį į tai, jog skaičiuojant tiek tiesioginę, tiek atvirkštinę D4 transformacijas, susiduriama su taip vadinama „krašto“ problema. Ši problema atsiranda dėl to, kad tiek žemo dažnio dažnio filtro, tiek aukšto dažnio filtro koeficientai $\{h_0, h_1, \ldots, g_3, g_4\}$ „išeina“ iš apdorojamo duomenų vektoriaus ribų. Tai akivaizdžiai matyti pažvelgus į aukščiau aprašytas diskrečiają D4 transformaciją realizuojančias matricas.

Žinomi keli krašto problemos sprendimo būdai – apdorojamas duomenų vektorius laikomas pariodiniu, duomenų vektoriaus galuose taikomas „veidrodinis“ atspindys ir panašiai.

Darbe pagrindinis dėmesys skiriamas modifikuotai D4 transformacijos versijai. Detalūs bazinės ir modifikuotos versijų realizavimo algoritmai pateikti atitinkamai 1.2.2 ir 1.2.3 skyreliuose.

1.2.2 BAZINĖ DAUBECHIES D4 TRANSFORMACIJA

Nagrinėkime vaizdą (duomenų vektorių)

$$X = (x_0 x_1 x_2 \ldots x_{N-1})^T = (s_0^{(0)} s_1^{(0)} s_2^{(0)} \ldots s_{N/2-1}^{(0)})^T \quad \textbf{(1.11)}$$

Kai $N = 2^n$, $n \in \square$.

Po pirmosios iteracijos žemo ir aukšto dažnio filtrų pagalba gautus tarpinius duomenų vektorius pažymėkime:

$$S^{(1)} = (s_0^{(1)} s_1^{(1)} s_2^{(1)} \ldots s_{2^{n-1}-1}^{(1)})^T$$

ir

$$D^{(1)} = (d_0^{(1)} d_1^{(1)} d_2^{(1)} \ldots d_{2^{n-1}-1}^{(1)})^T.$$

Reikšmės s_k $\left(k = 0,1,\ldots,2^{n-1}-1\right)$ naudojamos kaip tarpinis duomenų vektorius, pratęsiant iteracinę procedūrą.

Po n iteracijų diskretusis D4 spektras Y duomenų vektoriui $X = (x_0 x_1 x_2 \ldots x_{N-1})^T$ suformuojamas tokiu būdu:

$$Y = (s_0^{(n)} d_0^{(n)} d_0^{(n-1)} d_1^{(n-1)} d_0^{(n-2)} d_1^{(n-2)} d_2^{(n-2)} d_3^{(n-2)} \ldots d_0^{(1)} d_1^{(1)} \ldots d_{2^{n-1}-1}^{(1)})^T .$$

Pradinio vaizdo $X = S^{(0)}$ ir tarpinių vektorių $S^{(i)}$ $\left(i = 1,2,\ldots,n-1\right)$ „krašto" problema sprendžiama laikant apdorojamus vektorius periodiniais.

Diskrečiosios D4 transformacijos (bazinė versija) skaičiavimo algoritmas, kai $N = 2^n \geq 4$, pateikiamas žemiau.

Algoritmas.

1. $i := 0$.
2. Įvesti duomenų vektorių $S^{(i)}$, kurio ilgis yra $N_i = 2^{n-i}$ (beje, $N_0 = N$).
3. $k := 0$.
4. Apskaičiuoti: $s_k^{(i+1)} := A_k^{(i)}$, $d_k^{(i+1)} := B_k^{(i)}$.
5. $k := k+1$. Jei $k < N_i/2 - 1$, tai pereiti prie 4 etapo.
6. Apskaičiuoti: $s_{N_i/2-1}^{(i+1)} := U_{N_i/2-1}^{(i)}$, $d_{N_i/2-1}^{(i+1)} := V_{N_i/2-1}^{(i)}$
7. $i := i+1$. Jei $N_i \geq 4$, tai pereiti prie 2 etapo.
8. Įvesti $S^{(n-1)}$. Apskaičiuoti $s_0^{(n)} := \frac{1}{\sqrt{2}}\left(s_0^{(n-1)} + s_1^{(n-1)}\right)$, $d_0^{(n)} := \frac{1}{\sqrt{2}}\left(s_0^{(n-1)} - s_1^{(n-1)}\right)$.

9. Pabaiga.

Pastaba.

$$\begin{pmatrix} A_k^{(i)} \\ B_k^{(i)} \end{pmatrix} = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix} \cdot \begin{pmatrix} s_{2k}^{(i)} \\ s_{2k+1}^{(i)} \\ s_{2k+2}^{(i)} \\ s_{2k+3}^{(i)} \end{pmatrix}; \quad \begin{pmatrix} U_{N_i/2-1}^{(i)} \\ V_{N_i/2-1}^{(i)} \end{pmatrix} = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix} \cdot \begin{pmatrix} s_{N_i-2}^{(i)} \\ s_{N_i-1}^{(i)} \\ s_0^{(i)} \\ s_1^{(i)} \end{pmatrix}.$$

Atvirkštinės diskrečiosios D4 transformacijos (bazinė versija) apskaičiavimo algoritmas:

1. $i := n$.
2. Apskaičiuoti: $s_0^{(n-1)} := \frac{1}{\sqrt{2}}\left(s_0^{(n)} + d_0^{(n)}\right),\ s_1^{(n-1)} := \frac{1}{\sqrt{2}}\left(s_0^{(n)} - d_0^{(n)}\right).\ i := i-1.$
3. $i := i-1$
4. Apskaičiuoti: $s_k^{(i-1)} := \hat{U}_0^{(i)},\ s_1^{(i-1)} := \hat{V}_0^{(i)}$.
5. Su visais $k := 1,2,\ldots,2^{n-i}-1$, apskaičiuoti $s_k^{(i-1)} := \hat{A}_k^{(i)},\ s_{2k+1}^{(i-1)} := \hat{B}_k^{(i)}$.
6. Jei $i > 1$, tai pereiti prie 3 etapo.
7. Pabaiga.

Pastaba.

$$\begin{pmatrix} \hat{U}_0^{(i)} \\ \hat{V}_0^{(i)} \end{pmatrix} = \begin{pmatrix} h_2 & g_2 & h_0 & g_0 \\ h_3 & g_3 & h_1 & g_1 \end{pmatrix} \cdot \begin{pmatrix} s_{N_i/2-1}^{(i)} \\ d_{N_i/2-1}^{(i)} \\ s_0^{(i)} \\ d_0^{(i)} \end{pmatrix}; \quad \begin{pmatrix} \hat{A}_k^{(i)} \\ \hat{B}_k^{(i)} \end{pmatrix} = \begin{pmatrix} h_2 & g_2 & h_0 & g_0 \\ h_3 & g_3 & h_1 & g_1 \end{pmatrix} \cdot \begin{pmatrix} s_{k-1}^{(i)} \\ d_{k-1}^{(i)} \\ s_k^{(i)} \\ d_k^{(i)} \end{pmatrix}.$$

Būtina pastebėti, kad atskiri diskrečiojo D4 spektro koeficientai $S^{(n)}$ koeficientai $d_j^{(i)}$ $\left(i \in \{1,2,\ldots,n-1\},\ j \in \{0,1,\ldots,2^{n-i}-1\}\right)$ nusakomi griežtai apibrėžto (duomenų vektoriaus X) fragmento elementų reikšmėmis. Fiksuotai *i* reikšmei, pastarųjų fragmentų (blokų) sąjunga sutampa su X, o du gretus koeficientus nusakančių fragmentų sankirta nėra tuščia aibė. Ši situacija įvardijama kaip dalinio lokalizavimo erdvėje savybė.

Kai kurie praktiniai diskrečiųjų bangelių transformacijų (DBT) taikymai (progresyvus vaizdų kodavimas, paviršių (vaizdų) defektų lokalizavimas, klasifikavimo uždaviniai) reikalauja, jog DBT būtų pilnai lokalizuota erdvėje. Tuo tikslu įvedama modifikuota D4.

1.2.3 MODIFIKUOTA DAUBECHIES D4 TRANSFORMACIJA

Pradinis duomenų vektorius ((1.11) išraiška, 1.2.2 skyrelis) skaidomas į baigtinį nepersidengiančių dydžio 2^m $(1 \le m \le n-1)$ blokų skaičių. Apdorojimo metu blokai apeinami nuosekliai, kiekvienam jų taikant diskrečiosios D4 transformacijos apskaičiavimo algoritmą, būtent:

1. $i := 0$.
2. Įvesti duomenų vektorių $S^{(i)}$.
3. Su visais $k = 0,1,\ldots 2^{n-i-1}-1$, apskaičiuoti $s_k^{(i+1)}$ ir $d_k^{(i+1)} := B_k^{(i)}$:
   (a) $s_k^{(i+1)} := A_k^{(i)}$, $d_k^{(i+1)} := B_k^{(i)}$, kai $k \notin \Re_i = \left\{1 \cdot 2^{m-i-1}-1, 2 \cdot 2^{m-i-1}-1, \ldots, 2^{n-m} \cdot 2^{m-i-1}-1\right\}$, ir $i < m-1$;
   (b) $s_k^{(i+1)} := U_k^{(i)}$, $d_k^{(i+1)} := V_k^{(i)}$, kai $k \in \Re_i$, ir $i < m-1$;
   (c) $s_k^{(i+1)} := \frac{1}{\sqrt{2}}\left(s_{2k}^{(i)} + s_{2k+1}^{(i)}\right)$, $d_k^{(i+1)} := \frac{1}{\sqrt{2}}\left(s_{2k}^{(i)} - s_{2k+1}^{(i)}\right)$, kai $i \ge m-1$.
4. Jei $i < n-1$, tai $i := i+1$ ir pereiti prie 2 etapo.
5. Pabaiga.

Pastaba.

$$\begin{pmatrix} A_k^{(i)} \\ B_k^{(i)} \end{pmatrix} = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix} \cdot \begin{pmatrix} s_{2k}^{(i)} \\ s_{2k+1}^{(i)} \\ s_{2k+2}^{(i)} \\ s_{2k+3}^{(i)} \end{pmatrix}; \quad \begin{pmatrix} U_k^{(i)} \\ V_k^{(i)} \end{pmatrix} = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix} \cdot \begin{pmatrix} s_{2k}^{(i)} \\ s_{2k+1}^{(i)} \\ a^{(i)} \\ b^{(i)} \end{pmatrix}.$$

čia $a^{(i)} = s_{2k-2^{m-i}+2}^{(i)}$, $b^{(i)} = s_{2k-2^{m-i}+3}^{(i)}$, jei krašto problema sprendžiama periodiškumu.

Atvirkštinės diskrečiosios D4 transformacijos su daline blokų dekoreliacija apskaičiavimo algoritmas:

1. $i := n$.
2. Su visais $k = 0,1,\ldots 2^{n-i}-1$, apskaičiuoti $s_{2k}^{(i-1)}$ ir $s_{2k+1}^{(i-1)}$:
   (a) $s_{2k}^{(i-1)} := \hat{A}_k^{(i)}$, $s_{2k+1}^{(i-1)} := \hat{B}_k^{(i)}$, kai $k \notin \hat{\Re}_i = \left\{0,1 \cdot 2^{m-i}, 2 \cdot 2^{m-i}, \ldots, (2^{n-m}-1) \cdot 2^{m-i}\right\}$, ir $i < m$;
   (b) $s_{2k}^{(i-1)} := \hat{U}_k^{(i)}$, $s_{2k+1}^{(i-1)} := \hat{V}_k^{(i)}$, kai $k \in \hat{\Re}_i$, ir $i < m$;
   (c) $s_{2k}^{(i-1)} := \frac{1}{\sqrt{2}}\left(s_2^{(i)} + d_k^{(i)}\right)$, $s_{2k+1}^{(i-1)} := \frac{1}{\sqrt{2}}\left(s_2^{(i)} - d_k^{(i)}\right)$, kai $i \ge m$;
3. Jei $i > 1$, tai $i := i-1$ ir pereiti prie 2 etapo.

4. Pabaiga.

Pastaba:

$$\begin{pmatrix} \hat{A}_k^{(i)} \\ \hat{B}_k^{(i)} \end{pmatrix} = \begin{pmatrix} h_2 & g_2 & h_0 & g_0 \\ h_3 & g_3 & h_1 & g_1 \end{pmatrix} \cdot \begin{pmatrix} s_{k-1}^{(i)} \\ d_{k-1}^{(i)} \\ s_k^{(i)} \\ d_k^{(i)} \end{pmatrix}; \quad \begin{pmatrix} \hat{U}_k^{(i)} \\ \hat{V}_k^{(i)} \end{pmatrix} = \begin{pmatrix} h_2 & g_2 & h_0 & g_0 \\ h_3 & g_3 & h_1 & g_1 \end{pmatrix} \cdot \begin{pmatrix} s_{k+2^{m-i}-1}^{(i)} \\ d_{k+2^{m-i}-1}^{(i)} \\ s_k^{(i)} \\ d_k^{(i)} \end{pmatrix}.$$

Dvimatė diskrečioji D4 transformacija vaizdui (duomenų masyvui) $\left[X(m_1, m_2)\right]$ $(m_1, m_2 = 0,1,...,N-1)$ apskaičiuojama $2N$ kartų taikant vienmatę D4 transformaciją (pirmiausia, duomenų masyvo stulpeliams, po to, gauto tarpinio duomenų masyvo eilutėms). Gaunamas dvimatis diskretusis D4 spektras $\left[Y\left(k_1, k_2\right)\right]$, $k_1, k_2 \in \{0,1,...,N-1\}$.

1.2.4 KRAŠTO PROBLEMOS SPRENDIMO ALTERNATYVOS

Realizavus Daubechies D4 bazinę bei modifikuotą versijas ir atlikus išsamų tyrimą paaiškėjo, jog preliminariai naudotas „krašto“ problemos sprendimo būdas (duomenis laikant periodiniais) nėra efektyvus šios transformacijos atveju, kadangi smarkiai nukenčia energijos, sukauptos apdorojamuose duomenyse, „pakavimo“ savybė. Šio darbo metu buvo bandoma ieškoti alternatyvių „krašto“ problemos sprendimo būdų. Pirmiausia buvo imtasi realizuoti kito tipo bangelių kontekste plačiai naudojamą „veidrodinio“ atspindžio būdą. Tačiau greitai buvo įsitikinta, jog naudojant šį būdą nepavyksta vėliau teisingai atkurti pradinių duomenų. To priežastis, kurios kol kas nepavyko pašalinti – pažeidžiamas ortogonalumas. Taigi, imta ieškoti kitų „krašto“ problemos sprendimo būdų.

Tuo tikslu buvo atlikti teoriniai tyrimai ir :krašto“ problemai spręsti, buvo nustatyti sąryšiai tarp įvedamų papildomų elementų „išeinančių“ už apdorojamo duomenų vektoriaus *X* ribų. Tyrimo metu buvo žiūrima, jog nenukentėtų ortogonalumo savybė ir duomenis pavyktų sėkmingai atkurti.

Apsiribosime *l*-tąją D4 transformacijos (bazinė versija) realizacijai skirta iteracija, būtent $T_{D4}(n) \cdot X$, t.y.

$$\begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & \cdots & 0 & 0 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & h_0 & h_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & g_0 & g_1 \end{pmatrix} \begin{matrix} h_2 & h_3 \\ g_2 & g_3 \end{matrix} \square \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{pmatrix} \begin{matrix} \\ \\ \\ \\ \\ a \\ b \end{matrix} = \begin{pmatrix} h_0x_0 + h_1x_1 + h_2x_2 + h_3x_3 \\ g_0x_0 + g_1x_1 + g_2x_2 + g_3x_3 \\ h_0x_2 + h_1x_3 + h_2x_4 + h_3x_5 \\ g_0x_2 + g_1x_3 + g_2x_4 + g_3x_5 \\ \vdots \\ h_0x_{N-2} + h_1x_{N-1} + h_2a + h_3b \\ g_0x_{N-2} + g_1x_{N-1} + g_2a + g_3b \end{pmatrix}$$

Atlikus nesudėtingus skaičiavimus, pavyko nustatyti sąryšį tarp parenkamų reikšmių a ir b. Tam ką tik gautam tarpiniam duomenų vektoriui buvo panaudota matrica $T'_{D4}(n)$ (tikslu grįžti prie pradinio duomenų vektoriaus *X*.

Žemiau akcentuojami tie atkuriamo vektoriaus *X* elementai, kuriems įtaką daryti galėjo reikšmės a ir b. Taigi,

$$\begin{cases} h_2\left(h_0x_{N-2} + h_1x_{N-1} + h_2a + h_3b\right) + g_2\left(g_0x_{N-2} + g_1x_{N-1} + g_2a + g_3b\right) + \\ \quad + h_0\left(h_0x_0 + h_1x_1 + h_2x_2 + h_3x_3\right) + g_0\left(g_0x_0 + g_1x_1 + g_2x_2 + g_3x_3\right) = x_0 \\ h_3\left(h_0x_{N-2} + h_1x_{N-1} + h_2a + h_3b\right) + g_3\left(g_0x_{N-2} + g_1x_{N-1} + g_2a + g_3b\right) + \\ \quad + h_1\left(h_0x_0 + h_1x_1 + h_2x_2 + h_3x_3\right) + g_1\left(g_0x_0 + g_1x_1 + g_2x_2 + g_3x_3\right) = x_1 \end{cases} \quad \textbf{(1.12)}$$

Išsprendus šią sistemą, gauta jog

$$b = \sqrt{3}\left(a - x_0\right) + x_1. \quad \textbf{(1.13)}$$

Vadinasi, parenkant reikšmes a ir b, turi būti išpildoma (1.13) sąlyga. Keli tolimesniems tyrimams atrinkti atvejai:

1. kai $a = x_0$, gauname $b = x$ (t.y. gauname „krašto" problemos sprendimo būdą, kai vektorius interpretuojamas periodiniu);
2. kai $a = \frac{1}{2}\left(x_0 + x_{N-1}\right)$ (gaunamas vektoriaus galo reikšmių vidurkis), tai $b = \frac{\sqrt{3}}{2}\left(x_{N-2} - x_0\right) + x_1$;
3. kai $a = x_{N-2}$ („einama" link veidrodinio atspindžio), tai $b = \sqrt{3}\left(x_{N-2} - x_0\right) + x_1$.

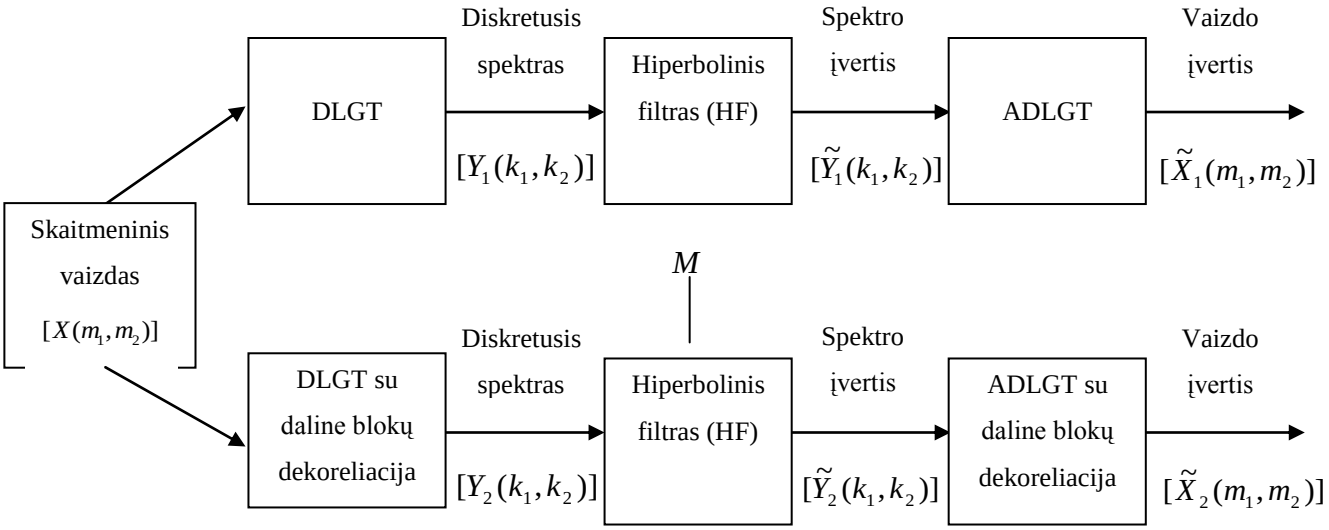
Tolimesniuose tyrimuose buvo numatyta *a* ir *b* reikšmes parinkti taip, kad maksimaliai pagerėtų energijos pakavimos savybė (D4 atveju).

2. BAZINĖS IR MODIFIKUOTOS DAUBECHIES D4 TRANSFORMACIJŲ PALYGINAMOJI ANALIZĖ

2.1 BENDRA SCHEMA

Palyginamosios analizės tikslas – išsiaiškinti, kaip įvesta modifikacija spektrinėje D4 srityje, kai „krašto“ problema sprendžiama remiantis apdorojamo duomenų vektoriaus (vaizdo) periodiškumu, įtakoja D4 transformacijos savybes (konkrečiai – informacijos sutankinimo („pakavimo“) savybę).

Diskrečiosios D4 transformacijos palyginamoji analizė buvo vykdoma pagal 2.1 pav. pateiktą schemą.



2.1 pav. Eksperimento atlikimo schema

Kaip matome (2.1 pav.), pradinis vaizdas $[X(m_1,m_2)]$ apdorojamas dviem būdais: pirmuoju atveju naudojama bazinė D4 versija, hiperbolinio filtro pagalba aukšto dažnio harmonikos D4 spektre paverčiamos nuliais, ir gautas rezultatas atkuriamas atvirkštinės diskrečiosios D4 transformacijos pagalba. Gaunamas to paties vaizdo įvertis.

Antruoju atveju atliekami tie patys veiksmai, naudojant modifikuotą diskrečiąją D4. Svarbu pabrėžti, jog hiperbolinio filtro lygio M reikšmė abiem atvejais imama ta pati. Turint du nepriklausomus gautus vaizdo įverčius atliekama tikslinė palyginamoji jų analizė.

2.2 FILTRAVIMO PAKLAIDŲ ĮVERTINIMAS

Po hiperbolinio filtravimo atkurtuose vaizduose atsiradusioms paklaidoms vertinti yra keletas būdų:

- vizualinis vertinimas;
- vidutinės kvadratinės paklaidos (VKP) kriterijus:

$$\delta\left(X,\widetilde{X}\right)=\sqrt{\frac{1}{N^2}\sum_{m_1,m_2=0}^{N-1}\left(X\left(m_1,m_2\right)-\widetilde{X}\left(m_1,m_2\right)\right)^2}\ ; \qquad \textbf{(2.1)}$$

čia: $\left[X\left(m_1,m_2\right)\right]$ - pradinis vaizdas, - $\left[\widetilde{X}\left(m_1,m_2\right)\right]$ po hiperbolinio filtro atkurtas vaizdas (įvertis).

- signalo ir triukšmo santykio maksimalios reikšmės (PSNR) kriterijus:

$$PSNR=20\cdot\lg\left(\frac{255}{VKP}\right). \qquad \textbf{(2.2)}$$

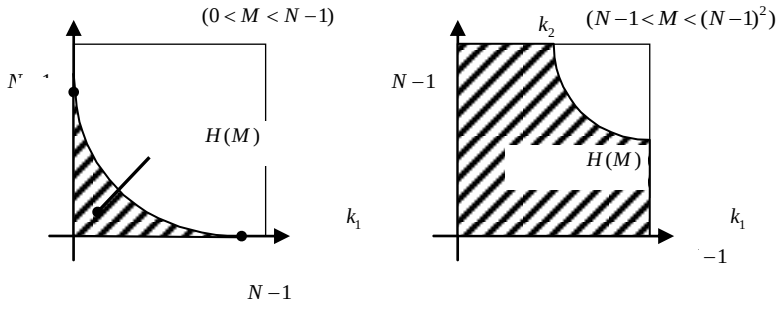
Šiame darbe naudosime vidutinę kvadratinę paklaidą (metriką) δ.

2.3 DVIMAČIŲ HIPERBOLINIŲ FILTRŲ ORGANIZAVIMAS

Hiperbolinio vaizdų filtravimo metodas gali būti taikomas nespalvotiems skaitmeniniais vaizdais, aprašomais dvimačiais duomenų (pikselių) masyvais $[X(m_1,m_2)]$. Metodas grindžiamas šiomis prielaidomis:

- atmetami spektriniai koeficientai yra nykstamai maži, todėl juose kaupiama informacija apie vaizdą $[X(m_1,m_2)]$ yra neesminė;
- atmetamo spektriniai koeficientai atspindi vaizdą $[X(m_1,m_2)]$ sudarančias aukšto dažnio dedamąsias, kurioms žmogaus akis yra mažiau jautri.

Formalizuokime dvimačių skaitmeninių vaizdų hiperbolinio filtravimo idėją. Tegu M yra tam tikras teigiamas skaičius $(1\le M\le(N-1)^2)$. Tarkime, kad $H(M)=\{(k_1,k_2)\,|\,\bar{k}_1\cdot\bar{k}_2\le M\}$, t.y. $H(M)$ yra dvimačio masyvo $N\times N$ eilučių ir stulpelių sutvarkytų eilės numerių porų (k_1,k_2) aibė tokia, kad $\bar{k}_1\cdot\bar{k}_2\le M$ $\left(\bar{k}_i=\max\{1,k_i\};i=1,2\right)$. Aišku, jog aibės $H(M)$ „kraštas“ plokštumoje turi hiperbolės pavidalą (2.3 pav.).



2.2 pav. Hiperbolinis dvimačių skaitmeninių vaizdų filtravimas

Galima parodyti, jog taškų, patenkančių į aibę $H(M)$ skaičius tenkina sąlygą $|H(M)| \le M(1+\ln M)$.

Atliekant hiperbolinį dvimačio skaitmeninio vaizdo filtravimą, spektriniai koeficientai $Y(k_1,k_2)$, kurių indeksai $(k_1,k_2) \notin H(M)$, yra atmetami. Atkuriant pradinį vaizdą $[X(m_1,m_2)]$, atmestieji spektriniai koeficientai keičiami nuliais.

Spektrinių koeficientų keitimo nuliais procedūrą patogu realizuoti, įvedant M lygio hiperbolinius filtrus $H(M,m_1,m_2)$. Pastarojo filtro dvimatis DT spektras užrašomas taip

$$\tilde{Y}\left[X(m_1,m_2)\right](k_1,k_2) = \begin{cases} Y(k_1,k_2), kai \quad \bar{k}_1 \cdot \bar{k}_2 \le M, \\ 0, kai \quad \bar{k}_1 \cdot \bar{k}_2 > M. \end{cases} \qquad \textbf{(2.3)}$$

Hiperbolinio filtro lygis M paprastai parenkamas atsižvelgiant į norimą išgauti vaizdo suglaudinimo (suspaudimo) efektą. Sudarytos išraiškos, nusakančios sąryšį tarp dvimačio skaitmeninio vaizdo suspaudimo koeficiento β ir filtravimo lygio M (2.1 lentelė). Dvimačių hiperbolinių filtrų atveju, $\beta = N^2 / |H(M)|$, kur

$$|H(M)| = 2\min\{N, M+1\} + \sum_{m=2}^{\min\{N-1,M\}} \min\{N, [M/m]+1\}. \qquad \textbf{(2.4)}$$

Nuosekliai didinant M reikšmes ir remiantis pastaraja išraiška, nesunkiai randamas aukščiausias hiperbolinio filtro lygis, užtikrinantis norimą vaizdo suspaudimo koeficientą β.

2.1 lentelė

Vaizdo 256×256 suspaudimo koeficiento β priklausomybė nuo hiperbilinio filtravimo lygio M

Suspaudimo koeficientas β	**Hiperbolinio filtro lygis M, kai vaizdo dydis 256x256**
2	12102
3	6544
4	4349
5	3199
6	2500
7	2039
8	1709
9	1463
10	1273
11	1127
12	1006
13	909
14	824
15	749
16	692
17	641
18	593
19	551
20	512

2.4 VAIZDO GLODUMO NUSTATYMAS

Pirmiausia pakomentuosime vaizdo glodumo sampratą ir vaizdo įverčių gavimo procedūras.

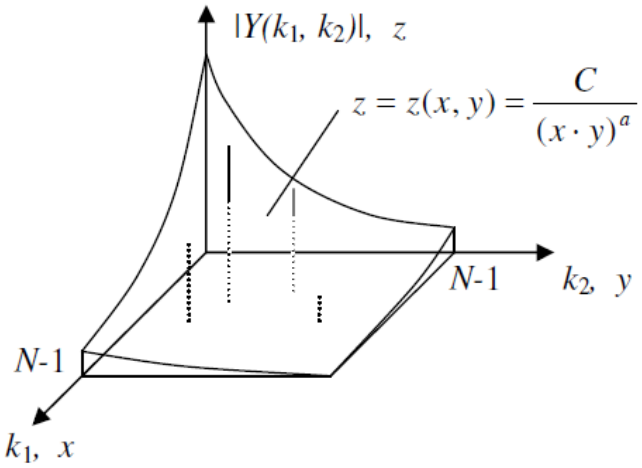
Energijos pakavimo savybės tyrimui (bazinės ir modifikuotos D4 transformacijų versijų kontekste) buvo bandoma pasinaudoti ir testuojamų vaizdų glodumo įverčiais, nustatomais (gaunamais) spektrinėje D4 srityje.

Vadovautasi nuostata – jei to paties vaizdo glodumas spektrinėje tam tikros transformacijos srityje yra maksimalus (kitų transformacijų atžvilgiu), tai ta transformacija geriausiai pakuoja duomenyse (vektoriuje, vaizde) sukauptą informaciją (energiją).

Imkime dvimatį skaitmeninį vaizdą $[X(m_1, m_2)]$, $m_1, m_2 \in \{0,1,\ldots,N-1\}$, $N = 2^n$ ir jo D4 spektrą $[Y(k_1, k_2)]$. Yra žinoma, jog didėjant indeksams k_1, k_2 spektriniai koeficientai mažėja ir gali būti aproksimuojami hiperboliniu paviršiumi (2.3 pav.)

$$z = z(x, y) = \frac{C}{(x, y)^{\alpha}}, \qquad \textbf{(2.5)}$$

kur $C \geq 0$, $\alpha \geq 0$. Reikšmė α vadinama vaizdo $[X(m_1, m_2)]$ glodumo parametru (klase, įverčiu).



2.3 pav. Vaizdo spektrinius koeficientus aproksimuojantis hiperbolinis paviršius.

Norint įvertinti konkretaus vaizdo glodumo parametrą α, remiantis mažiausiųjų kvadratų metodu, buvo sudaryta procedūra, būtent:

1. $\alpha := 0;\ \delta := \delta_{\max}$.
2. Skaičiuojame

$$Y_{\Sigma} = \sum_{k \in H} |Y(k)|^2 .$$

3. Randame:

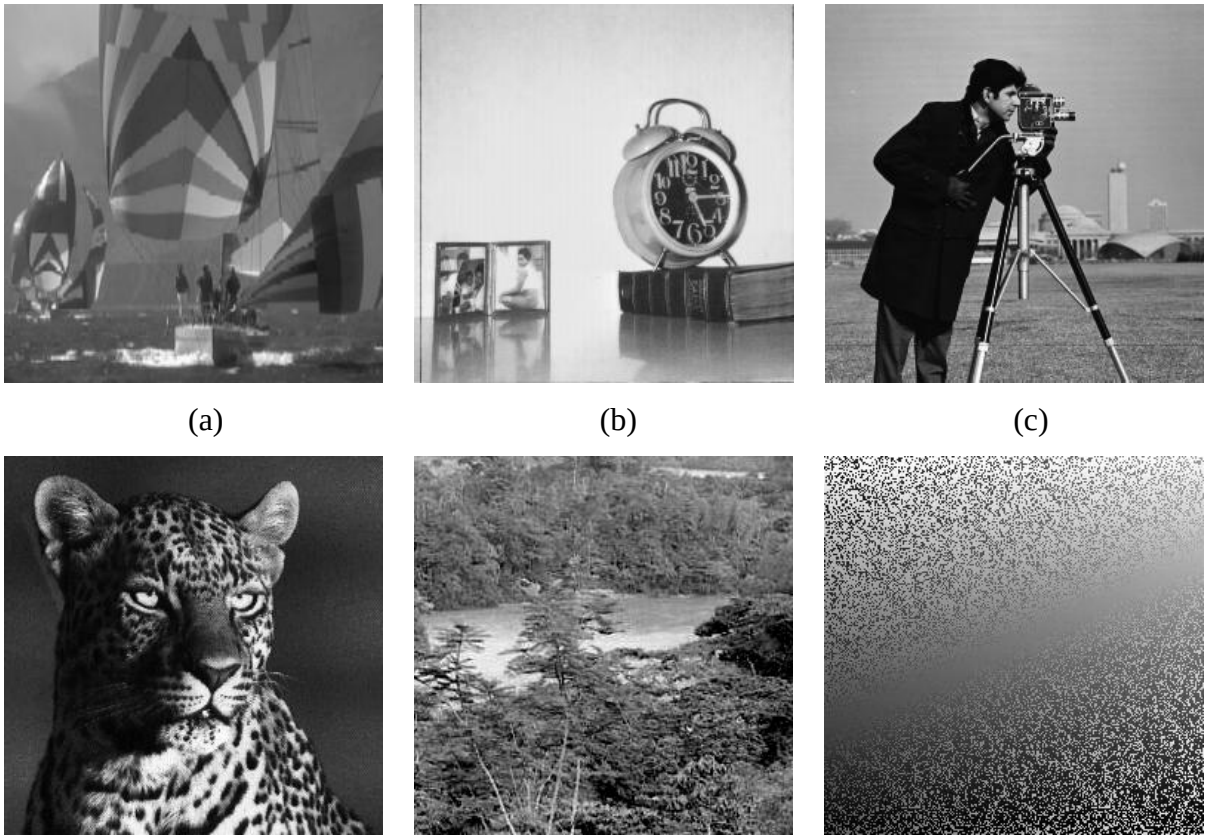
$$Z_Y(\alpha) = \sum_{k \in H} |Y(k)| \cdot A(\alpha, k),$$

$$W_Y(\alpha) = \frac{Z_Y(\alpha)}{A(\alpha)},$$

$$\tau = \tau(\alpha) = Y_{\Sigma} - W_Y(\alpha) \cdot Z_Y(\alpha);$$

kur $A(\alpha, k) = \frac{1}{(\bar{k}_1 \cdot \bar{k}_2)^{\alpha}}$; $A(\alpha) = \sum_{k \in H} A^2(\alpha, k)$

4. Jei $\tau < \delta$, tai $\delta := \tau$, $\alpha := \alpha + h$ ir kartojame 3 etapą, priešingu atveju pereiname prie 5 etapo.
5. Jei $|h| > 0.001$, tuomet $h := -0.1 \cdot h$, $\alpha := \alpha + h$ ir kartojame 3 etapą, priešingu atveju pereiname prie 6 etapo.
6. Pabaiga. Glodumo parametras α rastas.



(a) (b) (c)

(d) (e) (f)

2.4 pav. Skaitmeninių vaizdų 256×256 glodumo analizė: (a) Bures*.bmp*, α = 0.791; (b) *Clock.bmp*, α = 1.225; (c) *Cameraman.bmp*, α = 0.693; (d) *Leopard.bmp*, α = 0.562; (e) *Andes2.bmp*, α = 0.378; (f) *Dissolve.bmp*, α = 0.045

Eksperimentai rodo, jog skaitmeninių vaizdų glodumo įverčiai priklauso intervalui $(0,2)$. Kuo vaizdas glodesnis, tuo jo glodumo įvertis yra didesnis.

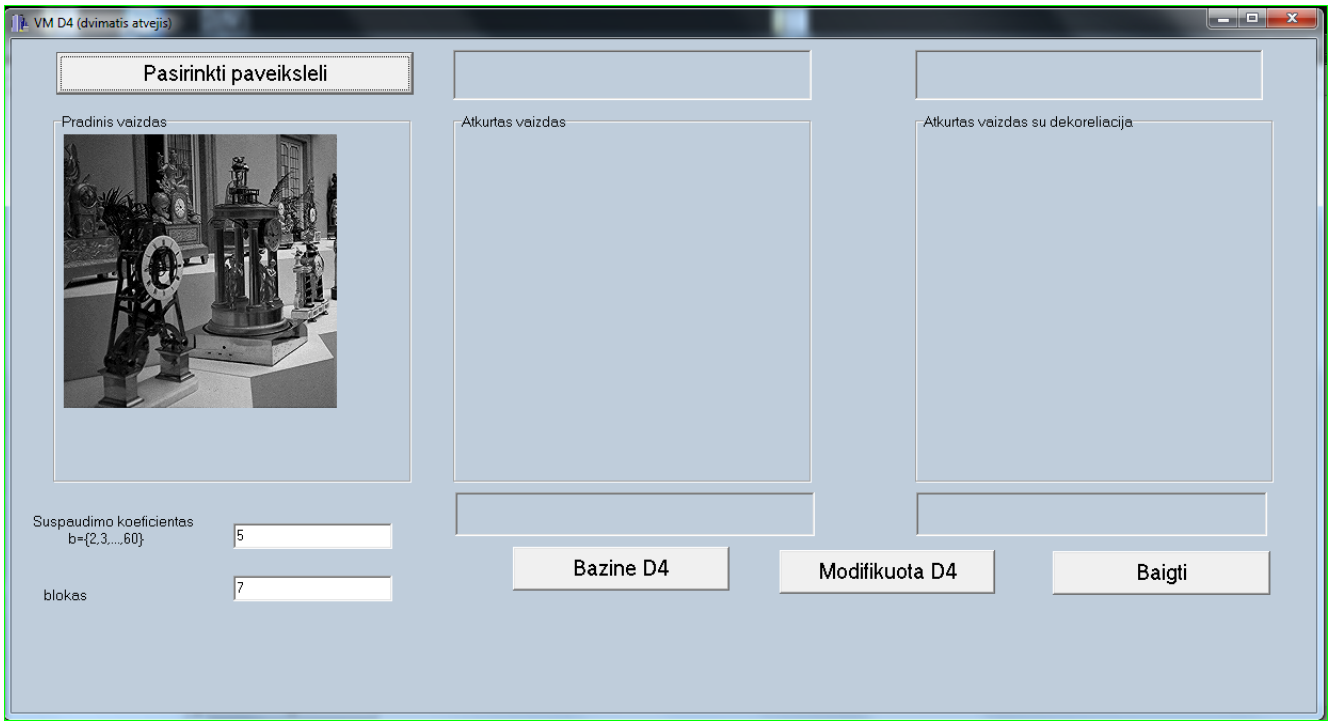
2.5 PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Darbe realizuoti bazinės diskrečiosios D4 ir modifikuotos diskrečiosios D4 (su daline vaizdo blokų dekoreliacija) apskaičiavimo algoritmai. Realizacijai įvykdyti buvo pasirinktas objektinio programavimo paketas „Borland C++". Sukurta paprasta grafinė vartotojo sąsaja. Jos tikslas – kiekvienam vartotojui prieinamos galimos pradinių parametrų modifikacijos, vaizdų atkūrimas atlikus skaičiavimus, matomos skaičiavimo metu atsiradusios paklaidos, vaizdo glodumo parametro α reikšmė.

Programa skirta dirbti su skaitmeniniais pilkos šviesos intensyvumo skalės (grayscale) vaizdais, kurių fiksuotas dydis 256×256.

Naudojimosi eiga:

1. Paleiskite programą paspaudus ***D4reseach.exe***. Atsivers programos vykdymo langas (2.6 pav.).



2.5 pav. Programos vykdymo langas

2. Paspauskite mygtuką ***Įkelti vaizdą,*** iš esančių duomenų failų išsirinkite norimą vaizdą (verta pažymėti, jog galite pasirinkti norimą direktoriją, Jums bus filtuojami *.bmp tipo vaizdai, su kuriais dirba programa, tokiu būdu išvengiama galimų naudojimosi programa klaidų). Vaizdas iškart pasirodys lange „***Pradinis vaizdas“***.
3. Pasirinkite norimą vaizdo suspaudimo lygį ir paspauskite mygtuką ***Bazinė D4***. Bus vykdomas bazinės D4 apskaičiavimo algoritmas, o atkurtas vaizdas pasirodys lange ***„Atkurtas vaizdas“***.
4. Į langelį įrašykite norimą vaizdo blokų skaičių ir paspauskite mygtuką ***Modifikuota D4***. Bus vykdomas modifikuotos D4 apskaičiavimo algoritmas, o atkurtas vaizdas pasirodys lange ***„Atkurtas vaizdas su dekoreliacija“***.

Tuo atveju, jei būtų pasirinktas vaizdas su neteisingais parametrais, išvedamas pranešimas apie klaidą ir ekrane pasirodo įspėjimas - *„Programa gali dirbti tik su 8 bitų vaizdais“*. Taip pat vartotojo patogumui yra sukurta visa apie galimas vartotojo klaidas pranešanti įspėjimų sistema (tokia kaip neteisingai įvestas vaizdo suspaudimo koeficientas β ar vaizdo bloko dydį charakterizuojantis parametras).

3. EKSPERIMENTO REZULTATAI IR JŲ ANALIZĖ

Eksperimento analizei buvo atrinkta keletas (su pilka šviesos intensyvumo skale) vaizdų 256×256, būtent (3.1 pav.): *Acura.bmp*, *Cameraman.bmp*, *Andes2.bmp*.

(a) (b) (c)

3.1 pav. Skaitmeniniai vaizdai, 256×256: (a) *Acura.bmp;* (b) *Cameraman.bmp;* (c) *Andes3.bmp*

Aukščiau matomi vaizdai pasirinkti neatsitiktinai, pagrindinė juos skirianti charakteristika yra jų glodumo klasė. Šią savybę nesunku pamatyti ir plika akimi – „glodus“ vaizdas yra tas, kuriame nėra staigių kontrastingumo pokyčių.

Skaitmeniniams vaizdams (3.1 pav.) atliktas eksperimentas pagal 2.1 pav. (2.1 skyrelis) pateiktą schemą. Panaudota visa vaizdo suspaudimo koeficientų skalė (β = 2, 5, 10). Buvo keičiamas ir kitas parametras – minimalus dekoreliuojamo bloko dydis $2^m \times 2^m$. Naudotos *m* reikšmės yra $\{2, 4, 7\}$.

Svarbu paminėti tai, jog atlikti preliminarūs eksperimentiniai tyrimai atskleidė įdomų faktą – gaunamų rezultatų (atkurtų po hiperbolinio filtaravimo vaizdų kokybės) visiškai neturi įtakos pasirinktas „krašto“ problemos sprendimo būdas (1.2.3 skyrelis).

Tą netrukus patvirtino ir teoriniai samprotavimai, kuriuos pateikiame žemiau.

Naudojant tuos pačius žymėjimus kaip ir 1.2.3 skyrelyje bei skaičiuojant (1-oji iteracija) tarpinį duomenų vektoriaus $T_{D4}(n) \cdot X$ paskutiniuosius du elementus, kuriems įtaką daryti gali reikšmės a ir b, gauname:

$$h_0 x_{N-2} + h_1 x_{N-1} + h_2 a + h_3 \left(\sqrt{3}(a - x_0) + x_1 \right) = h_0 x_{N-2} + h_1 x_{N-1} + h_2 x_0 + h_3 x_1, \quad \textbf{(3.1)}$$

$$g_0 x_{N-2} + g_1 x_{N-1} + g_2 a + g_3 \left(\sqrt{3}(a - x_0) + x_1 \right) = g_0 x_{N-2} + g_1 x_{N-1} + g_2 x_0 + g_3 x_1 \quad \textbf{(3.2)}$$

t.y. kokias bepaimtume reikšmes a ir b, jos niekaip negali daryti įtakos gaunamų tarpinių duomenų vektoriu. Taigi, tyrimams buvo nutarta imti pirmąjį „krašto“ problemos sprendimo būdą (1.2.2-1.2.3 skyreliai).

Atlikto eksperimento tyrimo rezultatai pateikti 3.1 – 3.3 lentelėse Rezultatai vizualiam vertinimui pateikiami 3.1 skyrelyje, o grafinis paklaidų įvertinimas - 3.2 skyrelyje.

3.1 lentelė

Vidutinė kvadratinė paklaida – vaizdas „*Bures.bmp*“

β	Bazinė D4	Modifikuota D4					
		Bloko dydis, *m*					
		2	3	4	5	6	7
2	2,10	2,94	5,07	7,52	10,66	15,54	2,19
3	3,49	4,27	5,71	7,94	10,96	15,79	3,68
4	4,51	5,36	6,38	8,42	11,26	16,04	4,69
5	5,37	6,29	6,87	8,73	11,54	16,31	5,61
6	6,12	7,09	7,58	9,23	11,88	16,56	6,36
7	6,63	7,60	7,97	9,49	12,05	16,72	6,91
8	7,29	8,17	8,42	9,78	12,33	16,99	7,63
9	7,78	8,73	8,94	10,22	12,56	17,21	8,14
10	8,14	9,20	9,37	10,54	12,80	17,31	8,51
11	8,43	9,48	9,60	10,78	12,94	17,41	8,83
12	8,74	9,85	9,92	11,04	13,08	17,55	9,17
13	9,03	10,30	10,31	11,17	13,27	17,68	9,51
14	9,34	10,53	10,48	11,34	13,42	17,82	9,86
15	9,64	11,05	10,91	11,80	13,58	17,95	10,16
16	9,90	11,43	11,29	11,98	13,76	18,10	10,40
17	10,14	11,70	11,58	12,22	13,97	18,21	10,67
18	10,44	11,99	11,81	12,47	14,16	18,33	10,91
19	10,61	12,18	12,01	12,63	14,32	18,40	11,13
20	10,85	12,49	12,31	12,75	14,39	18,50	11,39

3.2 lentelė

Vidutinė kvadratinė paklaida – vaizdas „*Cameraman.bmp*“

β	Bazinė D4	Modifikuota D4					
		Bloko dydis, *m*					
		2	3	4	5	6	7
2	7,47	8,43	11,13	15,24	22,07	26,58	7,63
3	10,36	11,18	13,06	16,62	23,06	27,47	10,53
4	12,27	13,15	14,50	17,74	23,82	28,14	12,62
5	13,45	14,39	15,41	18,43	24,30	28,60	13,78
6	14,28	15,49	16,39	19,10	24,75	28,99	14,87
7	15,38	16,53	17,27	19,78	25,20	29,28	15,87
8	16,13	17,31	17,93	20,22	25,34	29,54	16,50
9	16,76	17,88	18,34	20,55	25,65	29,91	17,21
10	17,23	18,51	18,86	20,97	25,94	30,20	18,04
11	17,92	19,28	19,56	21,56	26,48	30,47	18,78
12	18,59	19,90	20,10	21,93	26,73	30,60	19,43
13	18,98	20,15	20,33	22,14	26,89	30,70	19,70
14	19,36	20,59	20,68	22,29	26,96	30,79	19,97
15	19,66	21,07	21,20	22,53	27,08	30,95	20,43
16	19,94	21,23	21,33	22,69	27,22	31,09	20,82
17	20,07	21,53	21,63	22,90	27,32	31,14	21,28
18	20,36	22,12	22,15	23,19	27,50	31,32	21,62
19	20,60	22,49	22,50	23,45	27,74	31,45	21,93
20	21,04	22,87	22,90	23,78	27,99	31,62	22,24

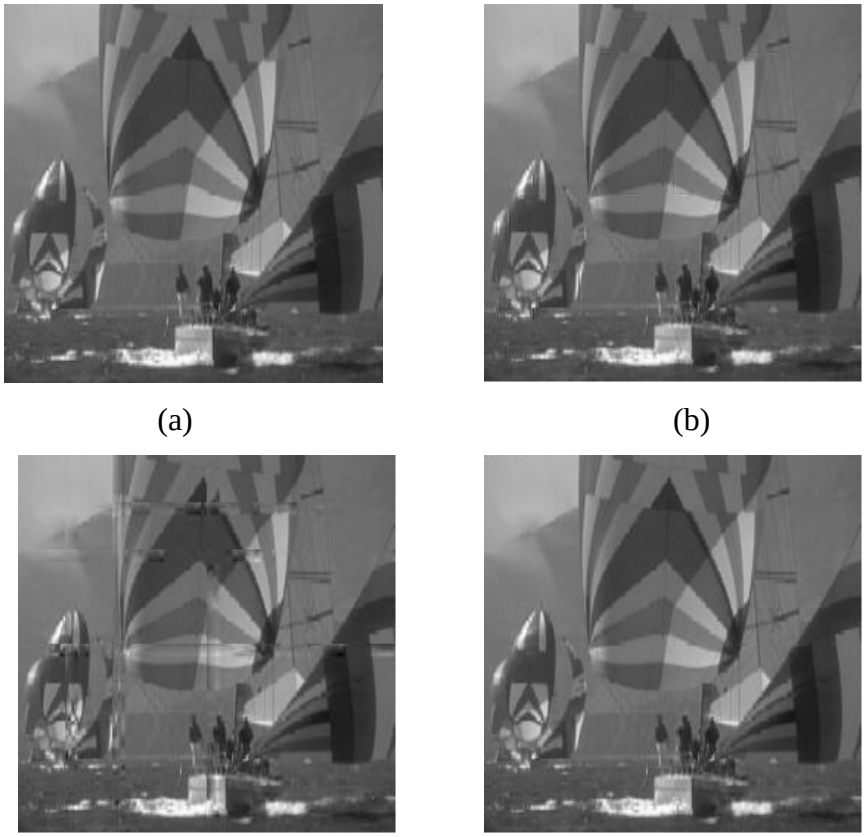
3.3 lentelė

Vidutinės kvadratinės paklaidos – vaizdas „*Andes3.bmp*“

β	Bazinė D4	Modifikuota D4					
		Bloko dydis, m					
		2	3	4	5	6	7
2	14,16	15,88	18,37	21,13	22,82	22,30	14,23
3	19,11	20,53	22,01	24,13	25,78	25,47	19,12
4	21,94	23,01	24,17	26,02	27,51	27,33	22,03
5	24,08	25,03	25,95	27,49	28,93	28,94	24,10
6	25,36	26,30	27,08	28,45	29,82	29,90	25,53
7	26,45	27,45	28,06	29,31	30,60	30,62	26,59
8	27,49	28,41	28,91	30,07	31,27	31,38	27,55
9	28,30	29,06	29,48	30,58	31,77	31,89	28,31
10	28,83	29,56	29,98	30,96	32,16	32,25	28,92
11	29,32	30,06	30,45	31,41	32,55	32,68	29,51
12	29,86	30,53	30,85	31,73	32,87	32,98	30,02
13	30,29	31,06	31,35	32,20	33,21	33,34	30,40
14	30,78	31,47	31,75	32,60	33,51	33,68	30,86
15	31,17	31,90	32,18	32,93	33,75	33,91	31,27
16	31,50	32,14	32,39	33,13	33,90	34,13	31,61
17	31,80	32,36	32,59	33,34	34,08	34,37	31,90
18	31,99	32,67	32,91	33,58	34,32	34,59	32,14
19	32,23	32,93	33,15	33,79	34,52	34,74	32,35
20	32,48	33,14	33,34	33,91	34,66	34,87	32,57

3.1. VAIZDŲ VIZUALINIS VERTINIMAS

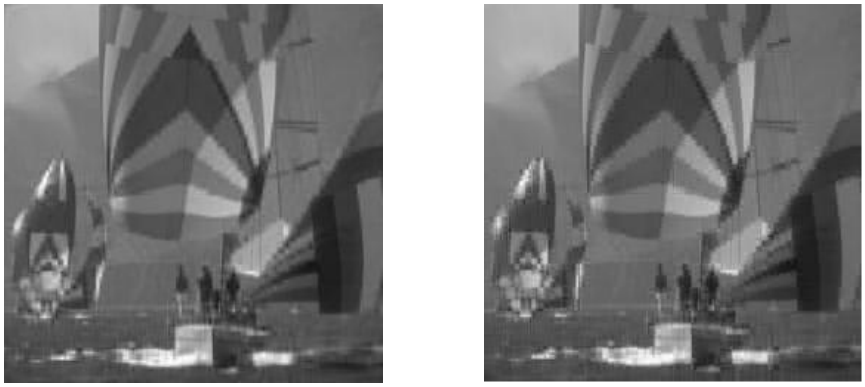
Vaizdo *Bures.bmp* apdorojimo rezultatai pateikti 3.2 – 3.4 pav.



(a) (b)

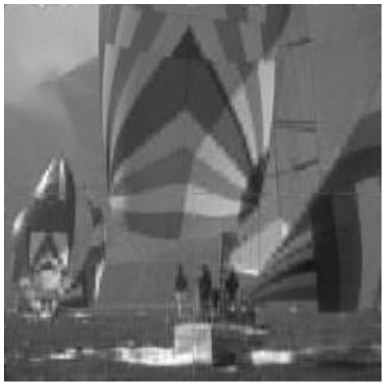
(c) (d)

3.2 pav. Vaizdas *Bures.bmp*, β = 2: (a) bazinė D4, δ = 2,10; (b) modifikuota D4, kai *m* = 2, δ = 2,94; (c), *m* = 4, δ = 7,52; (d), *m* = 7, δ = 2,19



(a) (b)

(c)



(d)

3.3 pav. Vaizdas *Bures.bmp*, β = 5: (a) bazinė D4, δ = 5,37; (b) modifikuota D4, kai *m* = 2, δ = 6,28; (c), *m* = 4, δ = 8,72; (d), *m* = 7, δ = 5,61

(a)

(b)

(c)

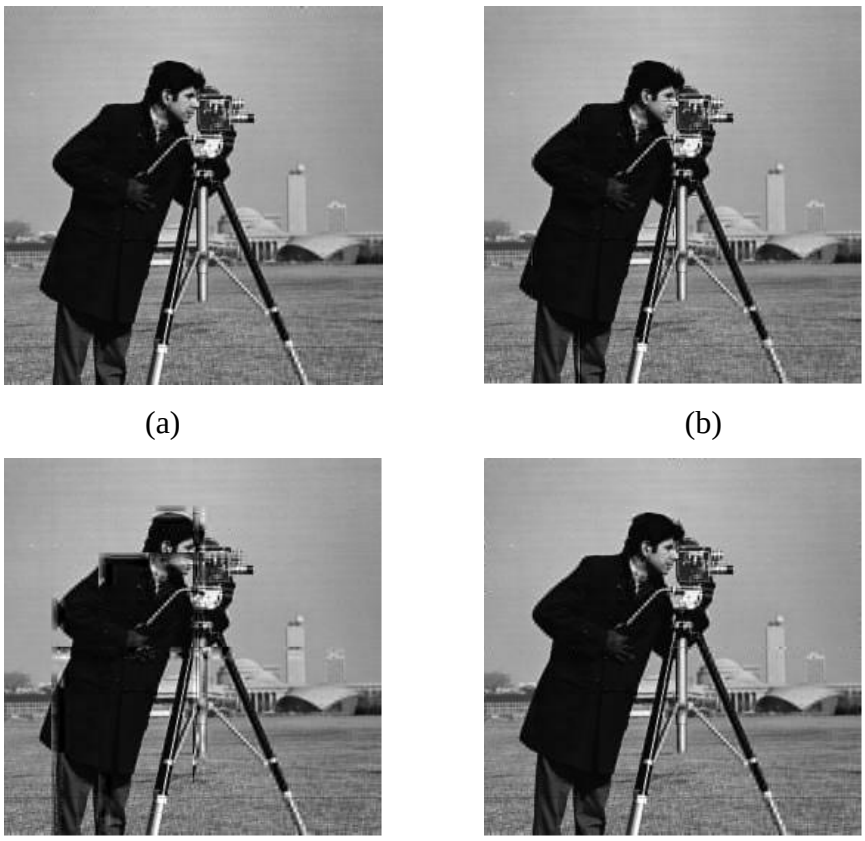
(d)

3.4 pav. Vaizdas *Bures.bmp*, β = 10: (a) bazinė D4, δ = 8,14; (b) modifikuota D4, kai *m* = 2, δ = 9,20; (c), *m* = 4, δ = 10,54; (d), *m* = 7, δ = 8,51

Imkime vizualiai glodų vaizdą *Bures.bmp*. Analizuojant gautus rezultatus (3.2 – 3.4 pav.) pastebime, kad tiktai suspaudimo koeficientui β įgijus reikšmę 10, atkurto vaizdo kokybė ryškiai suprastėja. Galime teigti, kad paveikslėlis yra tinkamas harmonikų sutankinimui ir eksperimentui. Įvesta D4 modifikacija atkurtų (po hiperbolinio filtravimo) vaizdų kokybei neturi didelės įtakos.

Vizualiai skirtumai geriau matomi tik suspaudimo koeficientui β esant 2, tuo tarpu didinant šį koeficientą ir tuo pačiu krentant bendrai vaizdo kokybei modifikuotos diskrečiosios D4 transformacijos įtaka ženkliai sumažėja.

Vaizdo *Cameraman.bmp* apdorojimo rezultatai pateikti 3.5 – 3.7 pav.



(a) (b)

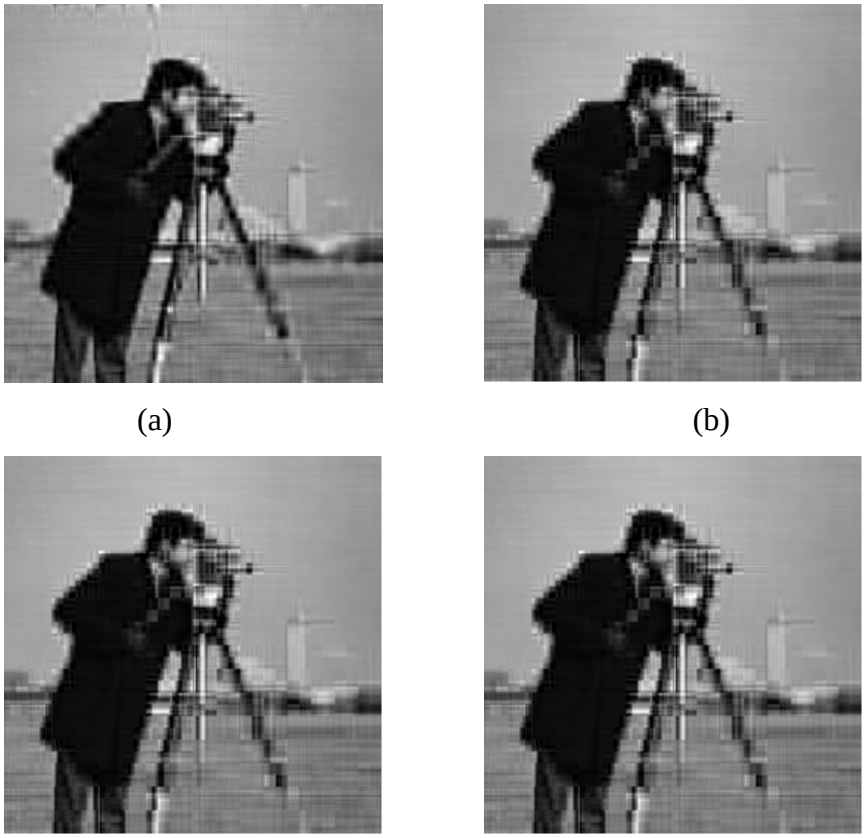
(c) (d)

3.5 pav. Vaizdas *Cameramen.bmp*, β = 2: (a) bazinė D4, δ = 14,71; (b) modifikuota D4, kai *m* = 2, δ = 15,88; (c), *m* = 4, δ = 21,13; (d), *m* = 7, δ = 14,23

(a) (b)

(c) (d)

3.6 pav. Vaizdas *Cameramen.bmp*, β = 5: (a) bazinė D4, δ = 24,08; (b) modifikuota D4, kai m = 2, δ = 25,03; (c), m = 4, δ = 27,49; (d), m = 7, δ = 24,10



(a) (b)

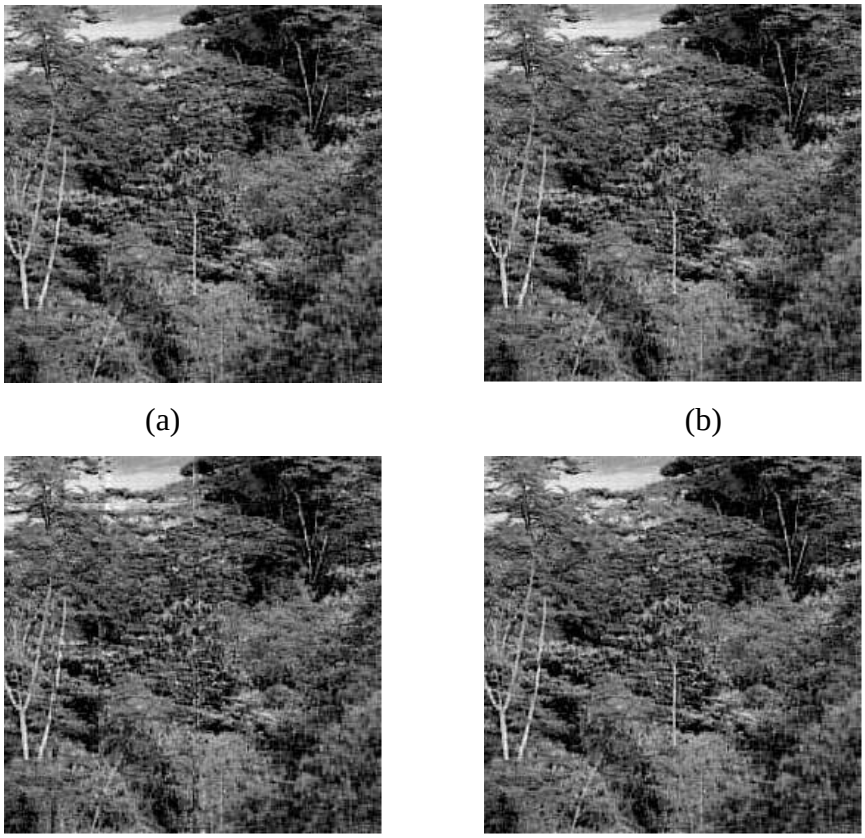
(c) (d)

3.7 pav. Vaizdas *Cameramen.bmp*, β = 10: (a) bazinė D4, δ = 28,83; (b) modifikuota D4, kai m = 2, δ = 29,56; (c), m = 4, δ = 30,96; (d), m = 7, δ = 28,92

Kaip matome (3.5 – 3.7 pav.), vaizdas *Cameraman.bmp* stipriai „deformuojasi“, jį suspaudus 10 kartų. Be to, modifikuotos diskrečiosios D4 transformacijos daroma įtaka kur kas didesnė nei vaizdo *Bures.bmp* atveju. Su bloko dydžiu m = 4 tiek paklaidos, tiek paveikslėlyje esantis vaizdas gerokai

skiriasi nuo šalia esančiųjų. Taigi, vaizdo *Cameraman.bmp* atveju mūsų pasirinktas „krašto“ problemos sprendimas yra visiškai netinkamas.

Vaizdo *Andes3.bmp* apdorojimo rezultatai pateikti 3.8 – 3.10 pav.



(a) (b)

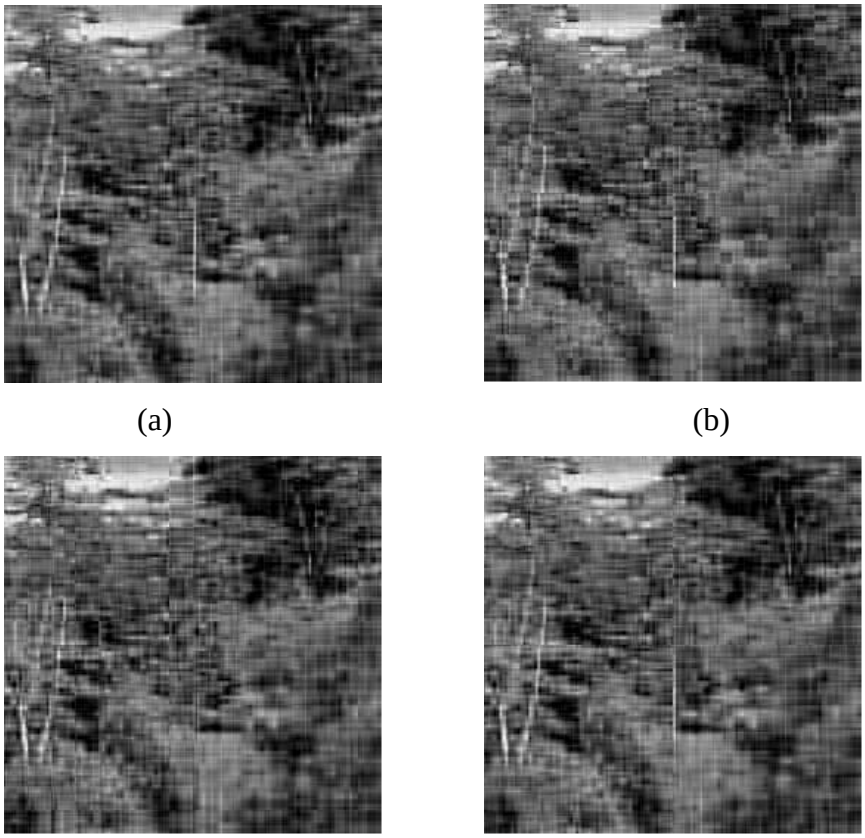
(c) (d)

3.8 pav. Vaizdas *Andes3.bmp*, β = 2: (a) bazinė D4, δ = 14,16; (b) modifikuota D4, kai m = 2, δ = 15,88; (c), m = 4, δ = 21,13; (d), m = 7, δ = 14,23

(a) (b)

(c) (d)

3.9 pav. Vaizdas *Andes3.bmp*, β = 5: (a) bazinė D4, δ = 24,08; (b) modifikuota D4, kai *m* = 2, δ = 25,03; (c), *m* = 4, δ = 27,49; (d), *m* = 7, δ = 24,10



(a) (b)

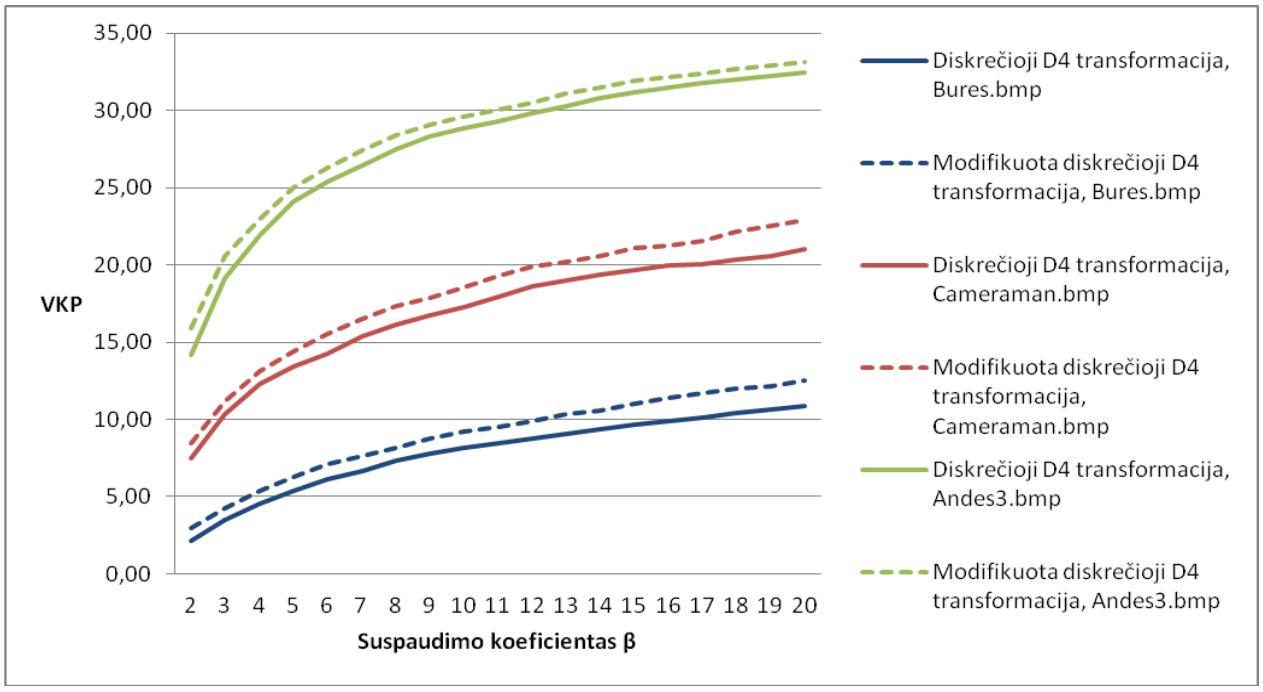
(c) (d)

3.10 pav. Vaizdas *Andes3.bmp*, β = 10: (a) bazinė D4, δ = 28,83; (b) modifikuota D4, kai *m* = 2, δ = 29,56; (c), *m* = 4, δ = 30,96; (d), *m* = 7, δ = 28,92

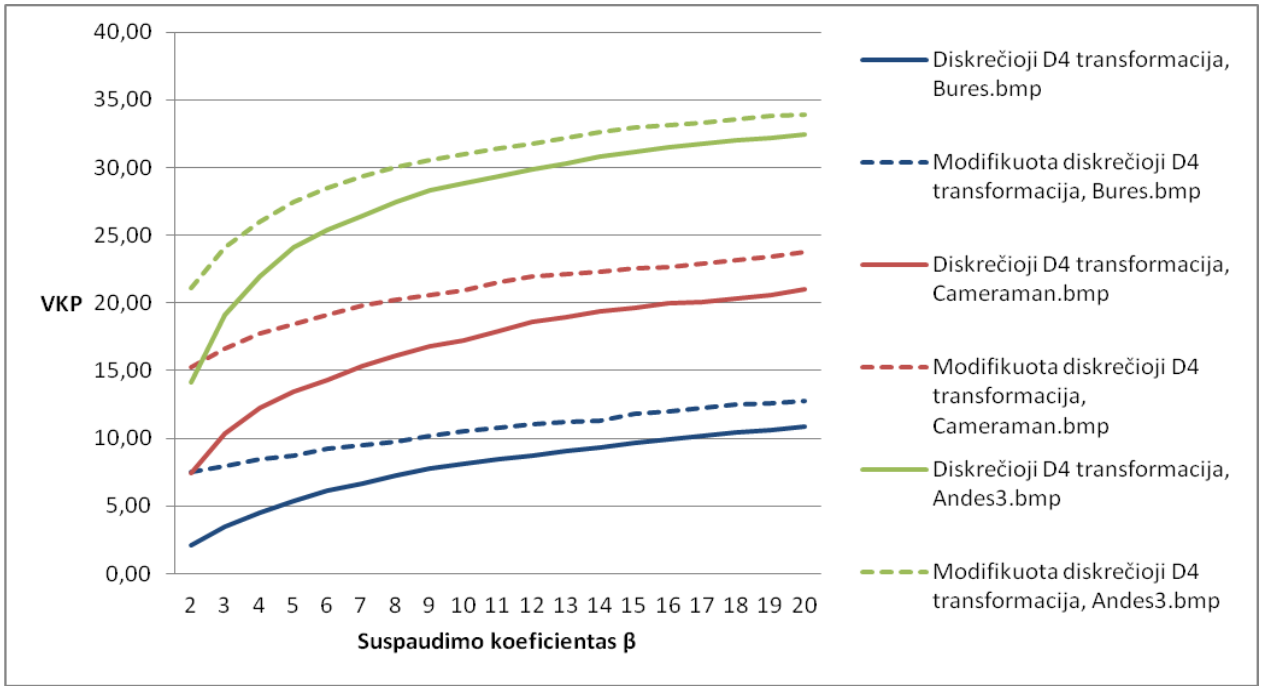
Vaizdas *Andes3.bmp* priklauso neglodžių vaizdų klasei. Kaip matome iš tyrimo rezultatų (3.8 – 3.10 pav.) modifikuotos diskrečiosios D4 transformacijos daroma įtaka atkurto vaizdo kokybei praktiškai nepastebima (su visomis *m* reikšmėmis). Kadangi šis vaizdas kontrastingas, tai natūralu, jog didinant suspaudimo lygį atkurto vaizdo kokybė staigiai mažėja.

Apibendrinant, didžiausią neigiamą įtaką „krašto“ problemos sprendimui pasirinktas „periodiškumas“ kelia „glodiems“ vaizdams – stipriai iškreipiami po hiperbolinio filtravimo atkurti vaizdai. Vizualiai tai ypatingai išryškėja bloko dydžiui esant $m = 4$.

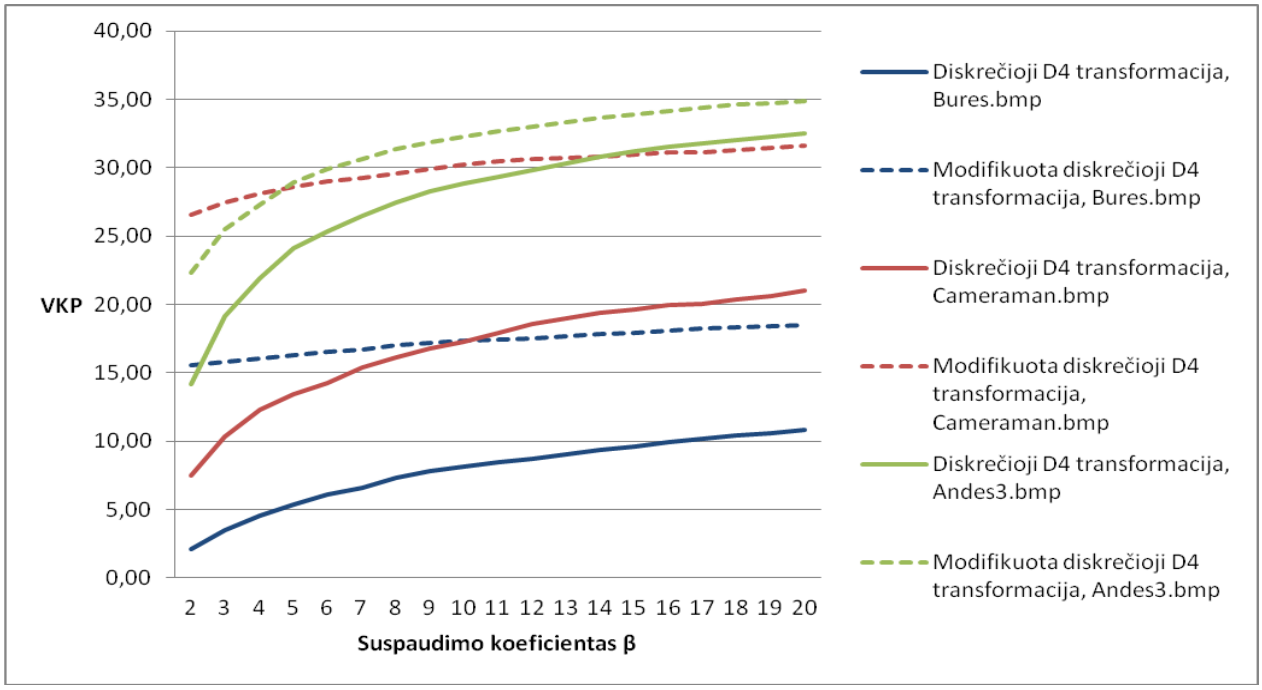
3.2. GRAFINIS PAKLAIDŲ VERTINIMAS



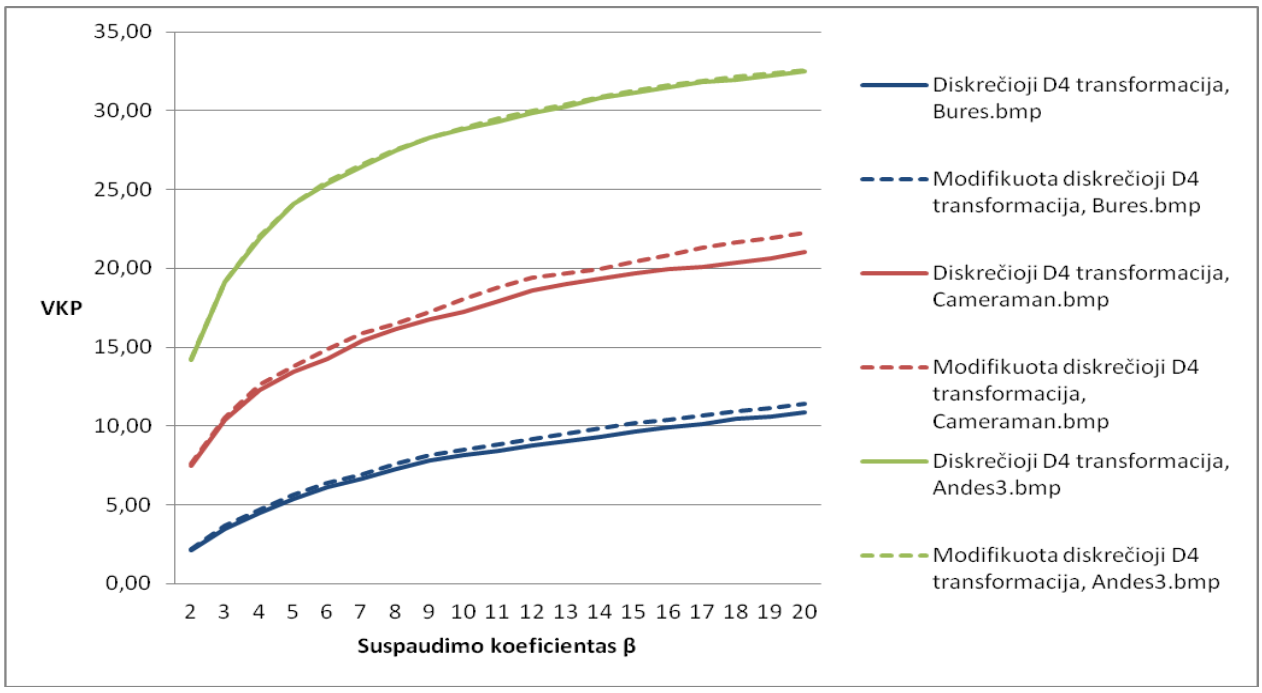
3.11 pav. Vidutinės kvadratinės paklaidos reikšmės, kai $m = 2$



3.12 pav. Vidutinės kvadratinės paklaidos reikšmės, kai $m = 4$



3.13 pav. Vidutinės kvadratinės paklaidos reikšmės, kai *m* = 6



3.14 pav. Vidutinės kvadratinės paklaidos reikšmės, kai *m* = 7

Analizuojant vidutinės kvadratinės paklaidos priklausomybės nuo dekoreliuojamo bloko dydžio $2^m \times 2^m$ grafikus akivaizdu, jog mažiausios paklaidos gaunamos, kai $m \in \{2, 7\}$. Tuo tarpu, kai bloko dydis $m \in \{4, 6\}$ vidutinės kvadratinės paklaidos vertės, lyginant diskrečiąją D4 transformaciją ir modifikuotą diskrečiąją D4 transformaciją, smarkiai skiriasi. Šis faktas ypatingai pastebimas vaizdo

suspaudimo koefientui β esant mažam $\beta \in$(2-5). Tai dar vienas įrodymas, jog „periodiškumas“ nėra pats geriausias būdas spręsti „krašto“ problemą modifikuotos diskrečiosios D4 transformacijos atveju.

IŠVADOS

1. Diskrečiųjų bangelių transformacija (DBT) praktiniu požiūriu turi labai svarbią savybę – apdorojamame skaitmeniniame vaizde sukaupta informacija išdėstoma ne tik pagal dažnį, bet ir erdvėje. Deja, pilnas lokalizavimas erdvėje būdingas tiktai paprasčiausiai DBT šeimos narei – Harro transformacijai. Tuo tarpu aukštesnės eilės DBT (tarkime, Le Gall, Daubechies D4, CDF 9/7 ir kt.) būdingas dalinis lokalizavimas erdvėje.
2. Tiek bazinės, tiek modifikuotos D4 transformacijos taikymo efektyvumas priklauso nuo apdorojamo vaizdo glodumo. Pastebėta, kad kuo glodesnis vaizdas, tuo efektyviau diskrečiosios bangelių transformacijos gali būti panaudotos vaizdams glaudinti (hiperbolinių filtrų kontekste). Be to, vaizdo glodumo įverčius galima panaudoti konkrečios DBT energijos „pakavimo“ savybei tirti.
3. Modifikuotos diskrečiosios D4 transformacijos taikymas kur kas labiau pablogina atkurtų (po hiperbolinio filtravimo) vaizdų kokybę, lyginant su bazine D4 transformacija. Gaunamos paklaidos priklauso tiek nuo minimalaus dekoreliuojamo bloko dydžio $2^m \times 2^m$, tiek nuo hiperbolinio filtro lygio *M*, beveik visais atvejais $\left(m \notin \{2, n-1\}\right)$ paklaidos yra netoleruotinos.
4. „Krašto“ problemos sprendimas, kai daroma prielaida, jog apdorojamas duomenų vektorius (vaizdas) laikomas periodiniu, nėra pats tinkamiausias būdas diskrečiajai D4 transformacijai su modifikacija realizuoti bei tirti. Teorinių ir eksperimentinių tyrimų metu gautas krašto problemai spręsti parenkamas papildomų duomenų vektoriu $X = (x_0 x_1 x_2 \ldots x_{N-1})^T$ rinkinukas *a* ir *b*, kai $b = \sqrt{3}\left(a - x_0\right) + x_1$, nedavė prognozuojamų rezultatų (t.y. energijos „pakavimo“ savybė išliko nepakitusi).

LITERATŪRA

1. Valantinas J. Diskrečiosios transformacijos (mokomoji knyga), Kaunas: Technologija, 2008. – 85 p.
2. Valantinas J. Diskrečiųjų transformacijų taikymas (mokymo priemonė), Kaunas: Technologija, 1993. – 80 p.
3. Valantinas J. On the application of Haar wavelets to locally progressive encoding of grey-level images, Information Technology and Control, No. 2(36), Kaunas: Technologija, 2007, p. 177–186.
4. Valens C. A really friendly guide to wavelets, 1999. – 23 p.
5. Žumbakis T., Valantinas J. Definition, evaluation and task-oriented application of image smoothness estimates, Information Technology and Control, No. 2(31), Kaunas: Technologija, 2004, p. 16–23.
6. Valantinas J. Hyperbolic image filtering under the influence of image dimensionality. Information Technology and Control, No. 3(24), Kaunas: Technologija, 2002, p. 7-13.

1 PRIEDAS. PROGRAMOS KODAS

TForm1 klasės failas Unit1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include <math.h>
#include <iomanip>
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------


//----Mygtuko Rezultatai veiksmai-----------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
/*   P[0][0]=2;
   P[1][0]=4;
   P[2][0]=5;
   P[3][0]=1;
   P[4][0]=7;
   P[5][0]=9;
   P[6][0]=8;
   P[7][0]=2;
   SkaiciuotiD4(true);
   Perskaiciuoti();
   SkaiciuotiAD4(true) ;
*/

   Memo1->Clear();
   Bmp2pix();
   SkaiciuotiD4(true);
   Spausdinti(TarpRezfailas);
   Perskaiciuoti();
   SkaiciuotiD4(false);
   Spausdinti(Rezfailas);
   Perskaiciuoti();
   float glod = ApskaiciuotiGloduma();
   Memo3->Lines->Add("Glodumas baziniam D4: "+FloatToStr(glod));
   if ((StrToInt(Edit1->Text)<1) || (StrToInt(Edit1->Text)>60))
    ShowMessage("Suspaudimo koef. turi buti sveikas naturalusis skaicius, esantis
intervale [2..60]");
   else
   SkaiciuotiHF(IvestiBeta(DuomBeta,StrToInt(Edit1->Text)));
   Spausdinti(HFRezfailas);
   SkaiciuotiAD4(false);
   Spausdinti(ATarpRezfailas);
   Perskaiciuoti();
   SkaiciuotiAD4(true);
   Spausdinti(ARezfailas);
   PiestiPaveiksleli();
   SkaiciuotiPaklaidas();
   SavePictureDialog1->Filter = "Bitmap paveiksleliai (*.bmp)|*.bmp";
```

```
 if (SavePictureDialog1->Execute())
  Image2->Picture->SaveToFile(SavePictureDialog1->FileName);

}
//----Mygtuko Baigti veiksmai-----------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
    Close();
}
//-------------------------------- METODAI -------------------------------------

//----Spausdinimas----------------------------------------------------------------
void TForm1::Spausdinti(char FailoVardas[])
{
  ofstream fr(FailoVardas);
  for(int i = 0 ; i < n; i++ )  {
    for(int j = 0; j < m; j++ )
      fr << setw(16) << A[i][j] ;
    fr<<endl;
  }
  fr.close();

}
/*
//----Spausdinimas----------------------------------------------------------------
void TForm1::SpausdintiD4(char FailoVardas[])
{
  ofstream fr(FailoVardas);
  for(int i = 0; i < n; i++ )  {
    for(int j = 1; j <= m; j++ )
      fr << setw(6) <<A[i][j] ;
    fr<<endl;
  }
  fr.close();
}
//----Spausdinimas----------------------------------------------------------------
void TForm1::SpausdintiAD4(char FailoVardas[])
{
  ofstream fr(FailoVardas);
  for(int i = 0; i < n; i++ ) {
    for(int j = 0; j < m; j++ )
      fr << setw(6) << A[i][j];
    fr<<endl;
  }
  fr.close();
}
*/
//----Spausdinimas---HF-----------------------------------------------------------
void TForm1::SpausdintiHF(char FailoVardas[])
{
  ofstream fr(FailoVardas);
  for(int i = 0; i < n; i++ ) {
    for(int j = 0; j < m; j++ )
      fr << setw(6) << A[i][j];
    fr<<endl;
  }
  fr.close();
}
//----Pasiima stulpeli is P-------------------------------------------------------
void TForm1::StulpelisD4(int stulp)
{
 for(int j = 0; j < m; j++ ) {
```

```
       S[0][j]=P[j][stulp];         //viduj 1 stulp 2 indek.eilute
    }
}
//----Pasiima eilute is P-------------------------------------------------------
-----
void TForm1::EiluteD4(int eil)
{
 for(int j = 0; j < n; j++ ) {
       S[0][j]=A[eil][j];         //viduj 1 stulp 2 indek.eilute
    }
}
//----Tiesiogines diskreciosios D4tranformacijos skaiciavimas-------------
void TForm1::SkaiciuotiD4(bool stulp)
{
for (int z = 0; z < m ; z++ ) {     //pradzia
 if (stulp)
   StulpelisD4(z);
 else
   EiluteD4(z);
 int p;   //iteraciju skaicius
 int newn = m;
  for (int i = 0; newn >= 4 ; i++ ) {                    //iteracijos
    for (int k = 0; k < newn/2-1 ;k++){                  // kinta k
      S[i+1][k]= S[i][2*k]*h0+S[i][2*k+1]*h1+S[i][2*k+2]*h2+S[i][2*k+3]*h3;
      D[i+1][k]= S[i][2*k]*g0+S[i][2*k+1]*g1+S[i][2*k+2]*g2+S[i][2*k+3]*g3;
      if (stulp)
        A[newn/2+k][z]=D[i+1][k];
      else
        A[z][newn/2+k]=D[i+1][k];
    }  // perskaiciuojam pradini masyva
    S[i+1][newn/2-1]= S[i][newn-2]*h0+S[i][newn-1]*h1+S[i][0]*h2+S[i][1]*h3;
    D[i+1][newn/2-1]= S[i][newn-2]*g0+S[i][newn-1]*g1+S[i][0]*g2+S[i][1]*g3;
    if (stulp)
      A[newn-1][z]=D[i+1][newn/2-1];
    else
      A[z][newn-1]=D[i+1][newn/2-1];
    newn= newn/2;
    p=i;
  }
  p=p+2;
  S[p][0]= 1/sqrt(2)*(S[p-1][0]+S[p-1][1]);
  if (stulp)
    A[0][z]=S[p][0];
  else
    A[z][0]=S[p][0];
  D[p][0]=1/sqrt(2)*(S[p-1][0]-S[p-1][1]);
  if (stulp)
    A[1][z]=D[p][0];
  else
    A[z][1]=D[p][0];
}
}
//----Atvirkstines diskreciosios D4transformacijos skaiciavimas-----------
void TForm1::SkaiciuotiAD4(bool stulp)
{
for (int z = 0; z < m ; z++ ) {     //pradzia
 if (stulp)
   StulpelisAD4(z);
 else
   EiluteAD4(z);
 int newn=1;
  for (int i = it; i >= 1 ; i-- ) {
```

```
    for (int k = 0; k < newn;k++){                    // kinta k
      if (it==i) {
        S[i-1][0]= 1/sqrt(2)*(S[i][0]+D[i][0]);
        S[i-1][1]= 1/sqrt(2)*(S[i][0]-D[i][0]);
      }
      else {
        if ((k>0) && (k<=pow(2,(it-i))-1)){
          S[i-1][2*k]  = S[i][k-1]*h2+D[i][k-1]*g2+S[i][k]*h0+D[i][k]*g0;
          S[i-1][2*k+1]= S[i][k-1]*h3+D[i][k-1]*g3+S[i][k]*h1+D[i][k]*g1;

        }
        else {
          S[i-1][0]= S[i][newn-1]*h2+D[i][newn-1]*g2+S[i][0]*h0+D[i][0]*g0;
          S[i-1][1]= S[i][newn-1]*h3+D[i][newn-1]*g3+S[i][0]*h1+D[i][0]*g1;
        }
      }
    }      newn = 2*newn;
  }
  if (stulp)
     IrasytiStulpeliAD4(z);
 else
    IrasytiEiluteAD4(z);
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  OpenPictureDialog1->Filter = "Bitmap paveiksleliai(*.bmp)|*.bmp";
  if(OpenPictureDialog1->Execute())
    Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
}
//---------------------------------------------------------------------------
void TForm1::Bmp2pix()
{

// Check if image is 8 bit
  assert(Image1->Picture->Bitmap->PixelFormat == pf8bit);

// Get height and width of the image
//  m = Image1->Picture->Bitmap->Height;
//  n = Image1->Picture->Bitmap->Width;

  // Get pixels color from image and store to matrix
  for (int i = 0; i < m; i++)
  {
    // Grab a pointer to the y-th row
    unsigned char const* const p_row =
      static_cast<unsigned char*>(Image1->Picture->Bitmap->ScanLine[i]);

    // For each column (pixel of the current row)
    for (int j = 0; j < n; j++)
    {
      P[i][j] = p_row[j];
      X[i][j] = P[i][j];
    }
  }
}
//---------------------------------------------------------------------------
void TForm1::Perskaiciuoti()
{
  for(int i = 0 ; i < n; i++ )  {
    for(int j = 0; j < m; j++ )
```

```
      P[i][j]=A[i][j] ;
  }
}
//---------------------------------------------------------------------------
/*void TForm1::Perskaiciuoti2()
{
  for(int i = 0 ; i < n; i++ )  {
    for(int j = 0; j < m; j++ )
      P[i][j]=A[i][j+1] ;
  }
}
//---------------------------------------------------------------------------
void TForm1::PerskaiciuotiAD4()
{
  for(int i = 0 ; i < n; i++ )  {
    for(int j = 0; j < m; j++ )
      P[i][j]=A[i][j] ;
  }
} */
//---------------------------------------------------------------------------
int TForm1::IvestiBeta(char FailoVardas[],int beta)
{  int filter=0,
       newbeta=0,
       newfilter=0;
  ifstream fd(FailoVardas);
  for(int i = 1 ; i <= 60; i++ )  {
      fd >>  newfilter >> newbeta ;
      if (newbeta==beta)
        filter=newfilter;
  }
  fd.close();
  return filter;
}
//---------------------------------------------------------------------------
void TForm1::PiestiPaveiksleli()
{
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++){
      if (A[i][j] < 0)
        A[i][j] = 0;
      if (A[i][j] > 255)
        A[i][j] = 255;
      Image2->Canvas->Pixels[j][i] = (TColor)RGB(A[i][j],A[i][j],A[i][j]);
  }
}
//------------------------------HF-------------------------------------------
void TForm1::SkaiciuotiHF(int HF)
{
  for(int i = 0; i < n; i++ )
    for(int j = 0; j < m; j++ )  {
     if (i*j > HF)
       P[i][j]=0;
    }
}
//----Pasiima stulpeli is P-------------------------------------------------
void TForm1::StulpelisAD4(int stulp)
{
  int k = 1;
  int no = 1;
  S[it][0]=P[0][stulp];
  for(int i = it;  i>0; i-- ){
    for(int j = 0; j < k; j++ ){
```

```
     D[i][j]=P[no][stulp];
     no++;
    }
    k=k*2;
 }
}
//----Pasiima eilute is P-----------------------------------------------------
void TForm1::EiluteAD4(int eil)
{
  int k = 1;
  int no = 1;
  S[it][0]=P[eil][0];
  for(int i = it;  i>0; i-- ){
    for(int j = 0; j < k; j++ ) {
     D[i][j]=P[eil][no];
    no++;
    }
    k=k*2;
  }
}
//----Iraso i stulpeli rezultata----------------------------------------------
void TForm1::IrasytiStulpeliAD4(int stulp)
{
   int no = 0;
   for(int j = 0; j <  n; j++ ){

     A[no][stulp]=S[0][j];
     no++;
   }
}
//----Iraso i eilute rezultata------------------------------------------------
void TForm1::IrasytiEiluteAD4(int eil)
{
   int no = 0;
   for(int j = 0; j < m; j++ ){
     A[eil][no]=S[0][j];
     no++;
   }
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  //Memo2->Clear();
  blocks=0;
  if ((StrToInt(Edit2->Text)<1) || (StrToInt(Edit2->Text)>it-1))
    ShowMessage("Bloku skaicius turi buti sveikas naturalusis skaicius, nedidesnis
uz duomenu ilgi");
  else
  blocks=StrToInt(Edit2->Text);
  Bmp2pix();
  SkaiciuotiD4de(true);
  Spausdinti(TarpRezfailas);
  Perskaiciuoti();
  SkaiciuotiD4de(false);
  Spausdinti(Rezfailasde);
  Perskaiciuoti();
  float glod2=ApskaiciuotiGloduma();
  Memo4->Lines->Add("Glodumas D4 su dekoreliacija: "+FloatToStr(glod2));
  if ((StrToInt(Edit1->Text)<1) || (StrToInt(Edit1->Text)>60))
    ShowMessage("Suspaudimo koef. turi buti sveikas naturalusis skaicius, esantis
intervale [2..60]");
  else
```

```
  SkaiciuotiHF(IvestiBeta(DuomBeta,StrToInt(Edit1->Text)));
  SpausdintiHF(HFRezfailasde);
  SkaiciuotiAD4de(false);
  Spausdinti(ATarpRezfailasde);
  Perskaiciuoti();
  SkaiciuotiAD4de(true);
  Spausdinti(ARezfailasde);
  SkaiciuotiPaklaidasdek();
  PiestiPaveiksleli2();
    SavePictureDialog2->Filter = "Bitmap paveiksleliai (*.bmp)|*.bmp";
   if (SavePictureDialog2->Execute())
   Image3->Picture->SaveToFile(SavePictureDialog2->FileName);
}
//---------------------------------------------------------------------------
//Tiesiogines diskreciosios D4tranformacijos skaiciavimas su dekoreliacija
void TForm1::SkaiciuotiD4de(bool stulp)
{
for (int z = 0; z < m ; z++ ) {     //pradzia
 if (stulp)
   StulpelisD4(z);
 else
   EiluteD4(z);
 int newn = n;
  for (int i = 0; i < it ; i++ ) {                    //iteracijos
    for (int k = 0; k <= (pow(2,it-i-1)-1) ;k++){       // kinta k
      if (i<blocks-1) {
        if (ArPriklausoR2(i,k)){
          int index1= 2*k+2-pow(2,blocks-i);
          int index2= 2*k+3-pow(2,blocks-i);
          S[i+1][k]= S[i][2*k]*h0+S[i][2*k+1]*h1+S[i][index1]*h2+S[i][index2]*h3;
          D[i+1][k]= S[i][2*k]*g0+S[i][2*k+1]*g1+S[i][index1]*g2+S[i][index2]*g3;
          if (stulp)
            A[newn/2+k][z]=D[i+1][k];
          else
            A[z][newn/2+k]=D[i+1][k];
        }
        else {
          S[i+1][k]= S[i][2*k]*h0+S[i][2*k+1]*h1+S[i][2*k+2]*h2+S[i][2*k+3]*h3;
          D[i+1][k]= S[i][2*k]*g0+S[i][2*k+1]*g1+S[i][2*k+2]*g2+S[i][2*k+3]*g3;
          if (stulp)
            A[newn/2+k][z]=D[i+1][k];
          else
            A[z][newn/2+k]=D[i+1][k];
        }
      }
      else {
        S[i+1][k]= 1/sqrt(2)*(S[i][2*k]+S[i][2*k+1]);
        if (i+1==it) {
          if (stulp)
            A[0][z]=S[i+1][k];
          else
            A[z][0]=S[i+1][k];
        }
        D[i+1][k]=1/sqrt(2)*(S[i][2*k]-S[i][2*k+1]);
        if (stulp)
          A[newn/2+k][z]=D[i+1][k];
        else
          A[z][newn/2+k]=D[i+1][k];
      }
    } newn=newn/2;
  }
}
```

```
}
//---------------------------------------------------------------------------
//Atvirkstines diskreciosios D4transformacijos skaiciavimas su dekoreliacija

void TForm1::SkaiciuotiAD4de(bool stulp)
{
for (int z = 0; z < m ; z++ ) {     //pradzia
 if (stulp)
   StulpelisAD4(z);
 else
   EiluteAD4(z);
 int newn = it;
  for (int i = it; i > 0 ; i-- ) {                    //iteracijos
    for (int k = 0; k <= (pow(2,newn-i)-1) ;k++){       // kinta k
      if (i<blocks) {
        if (ArPriklausoR(i,k)){
           int index1= k+pow(2,blocks-i)-1;
           S[i-1][2*k]  = S[i][index1]*h2+D[i][index1]*g2+S[i][k]*h0+D[i][k]*g0;
           S[i-1][2*k+1]= S[i][index1]*h3+D[i][index1]*g3+S[i][k]*h1+D[i][k]*g1;
        }
        else {
          S[i-1][2*k]  = S[i][k-1]*h2+D[i][k-1]*g2+S[i][k]*h0+D[i][k]*g0;
          S[i-1][2*k+1]= S[i][k-1]*h3+D[i][k-1]*g3+S[i][k]*h1+D[i][k]*g1;
        }
      }
      else {
        S[i-1][2*k]= 1/sqrt(2)*(S[i][k]+D[i][k]);
        S[i-1][2*k+1]= 1/sqrt(2)*(S[i][k]-D[i][k]);
      }
    }
  }
 if (stulp)
     IrasytiStulpeliAD4(z);
 else
    IrasytiEiluteAD4(z);
}

}
//---------------------------------------------------------------------------
void TForm1::PiestiPaveiksleli2()
{
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++){
      if (A[i][j] < 0)
        A[i][j] = 0;
      if (A[i][j] > 255)
        A[i][j] = 255;
      Image3->Canvas->Pixels[j][i] = (TColor)RGB(A[i][j],A[i][j],A[i][j]);
  }
}
//---------------------------------------------------------------------------
void TForm1::SkaiciuotiPaklaidas()
{
  float paklsuma=0;
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++){
        paklsuma+= pow((X[i][j] - A[i][j]),2);
  }
  delta=sqrt(paklsuma/pow(256,2));
  Memo1->Lines->Add("Paklaida "+FloatToStrF(delta,ffGeneral,15,3));
}
//---------------------------------------------------------------------------
```

```
void TForm1::SkaiciuotiPaklaidasdek()
{
  float paklsuma=0;
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++){
        paklsuma+= pow((X[i][j] - A[i][j]),2);
  }
  deltadek=sqrt(paklsuma/pow(256,2));
  Memo2->Lines->Add("Paklaida "+FloatToStrF(deltadek,ffGeneral,15,3));
}
//---------------------------------------------------------------------------
bool TForm1::ArPriklausoR(int ii, int kk)
{
 bool priklauso = false;
   int Rdaugiklis=pow(2,it-blocks)-1;
   for (int q = 0; q <= Rdaugiklis; q++){
     if (kk==(pow(2,q)-1)*pow(2,blocks-ii)){
       priklauso = true;
          //         ShowMessage(kk);
      }
   }
 return priklauso;
}
//---------------------------------------------------------------------------
bool TForm1::ArPriklausoR2(int ii, int kk)
{
 int q = 0;
 bool priklauso = false;
 int Rdaugiklis=pow(2,it-blocks);
 for (q; q <= Rdaugiklis; q++){
   if (kk==(pow(2,q)*(pow(2,blocks-ii-1))-1))
     priklauso = true;
 }
 return priklauso;
}


//---------------------------------------------------------------------------
float TForm1::ApskaiciuotiGloduma()
{
double alfa=0;
double delta= 10000000000000;
long double sumY=0;
long double sumZ=0;
double temp0=0;
long double sumA=0;
double tau=0;
double h=0.1;
int ii =0;
int jj=0;
bool cont=true;
 for (int i = 0; i < m; i++)
   for (int j = 0; j < n; j++)
     if (pow(j,2)+pow(i,2) !=0)
       sumY+=pow(A[i][j],2);


 while (cont) {
   for (int i = 0; i < m; i++){
     for (int j = 0; j < n; j++){
         if (j == 0)
           jj=1;
```

```
        else
          jj=j;
        if (i == 0)
          ii=1;
        else
          ii=i;
        if (pow(j,2)+pow(i,2) !=0){
          sumZ+=fabs(A[i][j])/pow(ii*jj,alfa);              //Zsum
          sumA+=(1/pow(ii*jj,alfa))*(1/pow(ii*jj,alfa));
        }
     }
   }
   tau=sumY-sumZ*sumZ/sumA;
   sumA=0;
   sumZ=0;
   if ( tau < delta){
    delta=tau;
    alfa+=h;
   }
   else {
     temp0=fabs(h);
     if (temp0 > 0.001){
       h=h*(-0.1);
       alfa+=h;
     }
     else       cont=false;
   }
 }
 return alfa;
}
```

TForm1 klasės failas Unit1.h

```
/////---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Dialogs.hpp>
#include <ExtDlgs.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include <assert.h>
#include <Grids.hpp>
#include <jpeg.hpp>
#include <math.hpp>
#include <ComCtrls.hpp>
#include <fstream>
using namespace std;
#include <iomanip>
const int Cn = 300;                                 // Maksimalus eiluciu kiekis
const int Cm = 300;                                 // Maksimalus stulpeliu kiekis
const int it = 8;
const int n = 256;
const int m = 256;
const float h0= 0.482962913,
            h1= 0.836516304,
            h2= 0.224143868,
            h3= -0.129409523,
```

```
            g0= -0.129409523,
            g1= -0.224143868,
            g2= 0.836516304,
            g3= -0.482962913;
char TarpRezfailas[] = "TarpRezultatai.txt";      // Tarpiniu rezultatu failas
char Rezfailas[] = "Rezultatai.txt";              // Rezultatu failas
char HFRezfailas[] = "HFRezultatai.txt";      // Tarpiniu rezultatu failas
char ARezfailas[] = "ARezultatai.txt";              // Rezultatu failas
char ATarpRezfailas[] = "ATarpRezultatai.txt";      // Tarpiniu rezultatu failas
char TarpRezfailasde[] = "TarpRezultatai.txt";      // Tarpiniu rezultatu failas
char Rezfailasde[] = "Rezultataide.txt";              // Rezultatu failas
char HFRezfailasde[] = "HFRezultataide.txt";      // Tarpiniu rezultatu failas
char ARezfailasde[] = "ARezultataide.txt";              // Rezultatu failas
char  ATarpRezfailasde[]  =  "ATarpRezultataide.txt";         //  Tarpiniu  rezultatu
failas
char DuomBeta[] = "M_ir_beta.txt";      // Tarpiniu rezultatu failas
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components                              // Mygtukas
Skaiciuoti
        TButton *Button3;                          // Mygtukas Rodyti
        TButton *Button4;
        TButton *Button2;
        TOpenPictureDialog *OpenPictureDialog1;
        TGroupBox *GroupBox1;
        TImage *Image1;
        TLabel *Label1;
        TGroupBox *GroupBox2;
        TGroupBox *GroupBox3;
        TImage *Image2;
        TImage *Image3;
        TEdit *Edit1;
        TLabel *Label2;
        TEdit *Edit2;
        TButton *Button1;
        TMemo *Memo1;
        TMemo *Memo2;
        TSavePictureDialog *SavePictureDialog1;
        TSavePictureDialog *SavePictureDialog2;
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
 private:   // User declarations
   int
       blocks,
       alpha,
       gama;
   float
       delta,
       deltadek,
       E[Cn][Cm],              //even
       O[Cn][Cm],              //odd
       D[Cn][Cm],              //Dk
       S[Cn][Cm],              //Sk
       P[Cn][Cm],
       A[Cn][Cm],
       X[Cn][Cm];              //Prad duom;
   void SpausdintiTarpD4(char FailoVardas[]);
   void Spausdinti(char FailoVardas[]);
```

```
   void StulpelisD4(int stulp);
   void EiluteD4 (int eil);
   void SkaiciuotiD4(bool stulp);
   void Perskaiciuoti();
   void Perskaiciuoti2();
   void PerskaiciuotiAD4();
   void Bmp2pix();
   int  IvestiBeta(char FailoVardas[],int beta);
   void SpausdintiTarpAD4(char FailoVardas[]);
   void SpausdintiAD4(char FailoVardas[]);
   void SpausdintiHF(char FailoVardas[]);
   void StulpelisAD4(int stulp);
   void EiluteAD4 (int eil);
   void SkaiciuotiAD4(bool stulp);
   void PiestiPaveiksleli();
   void PiestiPaveiksleli2();
   void SkaiciuotiHF(int HF);
   void IrasytiStulpeliAD4(int stulp);
   void SkaiciuotiD4de(bool stulp);
   void IrasytiEiluteAD4(int eil);
   void SkaiciuotiAD4de(bool stulp);
   void TForm1::SkaiciuotiPaklaidas();
   void TForm1::SkaiciuotiPaklaidasdek();
   bool TForm1::ArPriklausoR(int ii, int kk);
   bool TForm1::ArPriklausoR2(int ii, int kk);
   float TForm1::ApskaiciuotiGloduma();
public:           // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```