KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Gintarė Viščiūtė

DISKREČIOJO SVEIKASKAITINIO LE GALL SPEKTRO APSKAIČIAVIMO DVIMAČIO VAIZDO FRAGMENTAMS ALGORITMŲ ANALIZĖ

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

DISKREČIOJO SVEIKASKAITINIO LE GALL SPEKTRO APSKAIČIAVIMO DVIMAČIO VAIZDO FRAGMENTAMS ALGORITMŲ ANALIZĖ

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
prof. dr. J. Valantinas
2012 06 01

Recenzentas
doc. dr. P. Kanapeckas
2012 06 01

Atliko
FMMM-0 gr. stud.
G. Viščiūtė
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“)

Viščiūtė G. Procedures for finding discrete reversible Le Gall spectrum of two dimensional image fragments and their analysis: 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. – 54 p.

SUMMARY

Ever-increasing flows of graphical information (digital images) lead to the perception that it's necessary to search for new efficient image processing algorithms and ideas. Over the last few decades, the discrete wavelet transform, as well as wavelets themselves, has gained widespread acceptance in digital image and signal processing. Many kernels (mother wavelets) can be used for the discrete wavelet transform, for instance, Haar, Le Gall, Daubechies, etc. The discrete reversible (integer-to-integer) Le Gall wavelet transform is of particular importance. Among new solid applications of the integer Le Gall wavelet transform, there are image pattern recognition, progressive image encoding, defect localization in textural images and so on.

In this work, a new original procedure for finding the discrete reversible (integer-to-integer) Le Gall wavelet transform (spectrum) of the selected image blocks is presented. The presented procedure leans upper the assumption that the discrete Le Gall wavelet spectrum of the original input image is known, and appears to be much faster than direct evaluation of Le Gall spectra for the respective image blocks.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Nuolat intensyvėjantys grafinės informacijos (skaitmeninių vaizdų) srautai byloja apie tai, jog būtina ieškoti naujų efektyvių šios informacijos apdorojimo algoritmų, požiūrių bei idėjų. Šiuo metu populiariausi ir efektyviausi skaitmeninių vaizdų (signalų) apdorojimo algoritmai realizuojami spektrinėje diskrečiųjų bangelių (Haar, Le Gall, Daubechies) srityje.

Naujausi diskrečiųjų bangelių panaudojimo praktikoje atvejai – tai požymių išskyrimas vaizde, progresyvus vaizdų kodavimas, defektų aptikimas tekstūriniuose vaizduose ir pan.

Darbe analizuojama apgręžiamoji (sveikaskaitinė) Le Gall bangelių transformacija (bazinė ir modifikuota versijos), jų savybės bei greiti skaičiavimo algoritmai. Modifikuota versija leido sudaryti ir realizuoti greito perėjimo (nuo diskrečiojo Le Gall spektro visam vaizdui prie Le Gall spektro pasirinktam vaizdo fragmentui) procedūrą.

Šio darbo tikslas – ištirti diskrečiosios sveikaskaitinės Le Gall transformacijos (spektro) apskaičiavimo pasirinktiems dvimačio skaitmeninio vaizdo fragmentams procedūrų, taikant tiesioginį bei naujai sudarytą algoritmus, efektyvumą.

Šia tema jau buvo skaityti du moksliniai pranešimai „Diskrečioji Le Gall transformacija su daline blokų dekoreliacija“ ir „Diskrečiojo Le Gall spektro apskaičiavimo vienmačio vaizdo fragmentams ypatumai“ – atitinkamai IX-ojoje ir X-joje studentų konferencijoje „Taikomoji matematika”, vykusiose Kauno technologijos universitete 2011 ir 2012 metais.

Taip pat buvo skaitytas pranešimas ir pateiktas straipsnis tema „Diskrečiosios apgręžiamosios Le Gall bangelių transformacijos energijos pakavimo savybės tyrimas“ („Testing energy compaction property of the discrete recersible Le Gall wavelet transform“) leidiniui „Matematika ir matematinis modeliavimas 2012“.

1. DISKREČIOSIOS BANGELIŲ TRANSFORMACIJOS (DBT)

1.1 BENDROSIOS BANGELIŲ SAVYBĖS

Tolydžioji bangelių transformacija užrašoma lygybe

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

čia: $\psi^*_{s,\tau}(t)$ – bazinės funkcijos (bangelės); kompleksinė jungtinė išraiška; $f(t)$ – funkcija (signalas); s ir τ – nauji kintamieji (mastelis ir poslinkis), gaunami po bangelių transformacijos.

(1.1) išraiška parodo, kaip funkcija (signalas) $f(t)$ yra išskleidžiama bazinėmis bangelių funkcijomis $\psi_{s,\tau}$.

Atvirkštinė tolydžioji bangelių transformacija nusakoma lygybe

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

Bangelės $\psi_{s,\tau}(t)$ generuojamos iš vienos bazinės bangelės $\psi(t)$, kuri vadinama „motinine“ bangele, t.y.

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

čia s yra mastelio koeficientas, τ – poslinkio koeficientas, o $\sqrt{s}$ – energijos normalizavimo koeficientas.

Svarbu pabrėžti, jog (1.1), (1.2) ir (1.3) išraiškose bazinės bangelės funkcijos nėra apibrėžiamos (įvardijamos). Tai pagrindinis skirtumas tarp bangelių transformacijos ir kitų klasikinių (tarkim, Furjė) transformacijų.

Bangelės ((1.3) išraiška) turi tenkinti tam tikras griežtai suformuluotas sąlygas. Pagrindinės jų – tinkamumo ir reguliarumo sąlygos, suteikiančias bangelėms (bazinėms funkcijos) jų vardą.

Sakoma, kad integruojama kvadratu bangelė (funkcija) $\psi(t)$ tenkina tinkamumo sąlygą, jei

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

čia $\Psi(\omega)$ žymi funkcijos $\psi(t)$ Furjė transformaciją. Tinkamumo sąlyga leidžia analizuoti signalą, bei jį rekonstruoti be informacijos praradimo. Kita vertus, tinkamumo sąlyga reiškia, kad Furjė transformacija prilygsta nuliui taške $\omega = 0$, t.y.

$$\left|\Psi(\omega)\right|^2\Big|_{\omega=0} = 0\,. \tag{1.5}$$

Nulinė reikšmė, kai $\omega = 0$, taipogi byloja apie tai, kad suvidurkintos (laiko skalėje) bangelės reikšmė turi būti lygi nuliui, t.y.

$$\int \psi(t)dt = 0\,. \tag{1.6}$$

Vadinasi, $\psi(t)$ turi būti „bangelė“.

Reguliarumo sąlyga siejama su bangelių skleidinio konvergavimo greičio samprata. Pastaroji sąlyga teigia, kad bangelės funkcija turi būti glodi ir koncentruota tiek laiko, tiek dažnio srityse. Reguliarumo sąvoka yra pakankamai sudėtinga, todėl bandysime ją paaiškinti pasinaudodami nykstančiais pradiniais momentais.

Išskleiskime bangelės funkciją (1.1) Teiloro eilute (taško $t = 0$ aplinkoje). Tarkime, kad $\tau = 0$. Tada gausime:

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

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

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

(1.7) išraišką galime perrašyti taip

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

Iš tinkamumo sąlygos žinome, kad nulinis momentas $M_0 = 0$. Vadinasi, (1.9) išraiškos pirmasis narys lygus nuliui. Jeigu pavyktų ir kitus momentus (iki *n*-tosios eilės) prilyginti nuliui, tai bangelių transformacijos koeficientai $\gamma(s,\tau)$ tolydžiajam signalui $f(t)$ gestų greičiu s^{n+2}.

Praktiniuose taikymuose visiškai pakanka (tą patvirtina eksperimentai), kad pradiniai momentai būtų bent jau artimi nuliui, [1].

Tolydžiają bangelių transformaciją sunku įgyvendinti kompiuterinėje aplinkoje dėl keleto priežasčių. Visų pirma, ši bangelių transformacija pasižymi pertekliškumu, nes transfomacijos skaičiavimui naudojamos tolydžiosios mąstelio ir poslinkio funkcijos. Antra, dauguma bangelių transformacijų neturi analizinių sprendinių ir įgyvendinamos tiktai skaitmeniniu būdu. Taigi, reikalingi greitieji algoritmai bangelėms „perkelti“ į naują lygį.

Minėtoms problemos spręsti įvedama diskrečioji bangelių transformacija, kai mąstelio ir poslinkio funkcijos yra diskretizuojamos (nustatytu dažniu) laiko intervale. Tuo tikslu bangelė ((1.3) išraiška) yra modifikuojama:

$$\psi_{j,k}(t) = \frac{1}{\sqrt{s_0^j}} \psi\left(\frac{t - k\tau_0 s_0^j}{s_0^j} \right); \qquad \textbf{(1.10)}$$

čia j ir k yra sveikieji skaičiai, $s_0 > 1$ – diskretizavimo (mastelio) žingnis (paprastai, $s_0 = 2$), τ_0 – poslinkio koeficientas.

Pastebėsime, kad nors (1.10) išraiška ir vadinama diskrečiąja bangele, iš tikrųjų ji yra (dalimis) tolydžioji funkcija.

Įvedę diskrečiosios bangelės sąvoką, panaikinome pertekliškumą ir galime apibrėžti ortogonalumo sąlygą:

$$\int \psi_{j,k}(t) \psi^*_{m,n}(t) dt = \begin{cases} N, \text{ kai } j = m \text{ ir } k = n, \\ 0, \text{ kitu atveju;} \end{cases} \qquad \textbf{(1.11)}$$

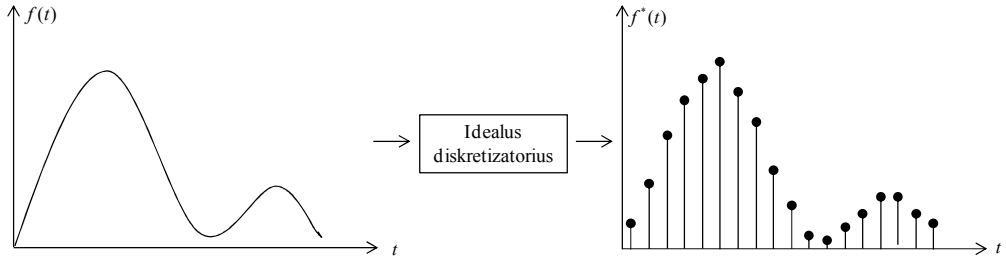
kai $N = 1$, tai funkcijų sistema $\{\psi_{j,k}(t)\}$ vadinama ortonormuota.

Atvirkštinė diskrečioji bangelių transformacija (DBT) nusakoma lygybe:

$$f(t) = \sum_{j,k} \gamma(j,k) \psi_{j,k}(t). \qquad \textbf{(1.12)}$$

1.2 DBT REALIZAVIMO YPATUMAI

Norint analoginį (tolydųjį) signalą $f(t)$ paversti skaitmeniniu, pirmiausia reikia jį diskretizuoti, t.y. signalo $f(t)$ reikšmes reikia fiksuoti (nuskaitinėti) tik tam tikruose diskretizavimo taškuose (1.1 pav.).



1.1 pav. Skaitmeninio signalo samprata

Diskretizavimo rezultatas – duomenų vektorius $X = [X(m)] = (X(0)\ X(1)\ \ldots\ X(N-1))^T$, kurį toliau vadinsime vienmačiu skaitmeniu signalu (vaizdu).

Bendru atveju, diskrečioji transformacija sudarytam vaizdui apibrėžiama taip:

$$Y = (Y(0)\ Y(1)\ \ldots\ Y(N-1))^T = \frac{1}{N} \cdot T \cdot X\ , \qquad \textbf{(1.13)}$$

kur Y – diskretusis vienmačio vaizdo (duomenų vektoriaus) X spektras, T – transformacijos matrica, kurios pagrindinė savybė – ortogonalumas, t.y. $T^T = N \cdot T^{-1}$. Beje, matricos T eilutės dažnai vadinamos baziniais DBT vektoriais.

Atvirkštinę diskrečiają transformaciją galima realizuoti taip

$$X = T^T \cdot Y\ . \qquad \textbf{(1.14)}$$

Pastebėsime, jog egzistuoja daug įvairių matricų T, tenkinančių ortogonalumo sąlygą. Keletas jų (plačiai taikomos praktikoje) pateiktos žemiau ($N = 2^n$, $n \in \square$).

Diskrečiosios Harro transformacijos matrica

$$T_{HT}(n) = \begin{pmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & -1 \end{pmatrix}$$

Diskrečiosios Le Gall transformacijos matrica

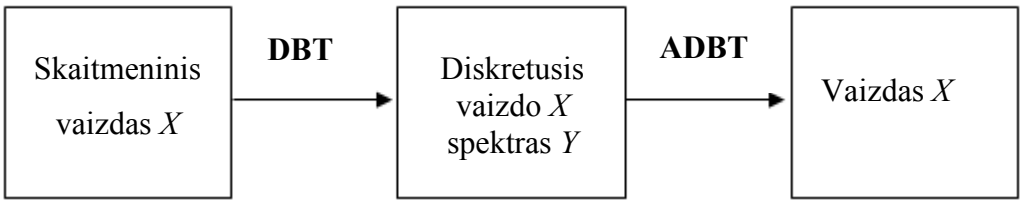
$$T_{DLGT}(n)=\begin{matrix} -\frac{1}{8} & \frac{1}{4} \end{matrix}\begin{pmatrix} \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 \end{pmatrix}\begin{matrix} \\ \\ \\ \\ -\frac{1}{8} \\ \\ \\ \\ \\ -\frac{1}{2} \end{matrix}$$

Diskrečiosios Daubechies D4 transformacijos matrica

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

kur $h_0=\frac{1+\sqrt{3}}{4\sqrt{2}}$, $h_1=\frac{3+\sqrt{3}}{4\sqrt{2}}$, $h_2=\frac{3-\sqrt{3}}{4\sqrt{2}}$, $h_3=\frac{1-\sqrt{3}}{4\sqrt{2}}$, $g_0=h_3$, $g_1=-h_2$, $g_2=h_1$, $g_3=-h_0$.

Diskrečioji bangelių transformacija (DBT) leidžia rasti ne tik diskretųjį vaizdo spektrą, bet ir vienareikšmiškai atkurti patį vaizdą, tam panaudojant atvirkštinę diskrečiąją transformaciją. Bendra schema pateikta 1.2 pav.

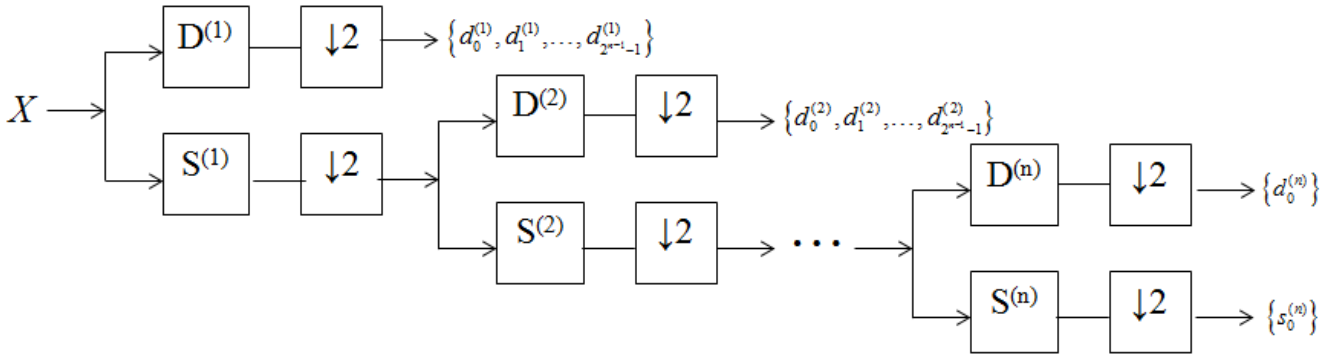


1.2 pav. Vaizdo spektro radimo ir jo atkūrimo schema

Diskrečiąją bangelių transformaciją (DBT) galima realizuoti iteraciniu būdu, panaudojant mastelio funkcijos koeficientus (žemo dažnio filtrą) ir bangelės funkcijos koeficientus (aukšto dažnio filtrą) (1.3 pav.).

Kaip parodyta 1.3 pav., po pirmosios iteracijos, panaudojant aukšto dažnio filtrą, apskaičiuojama pusė DBT spektro koeficientų, kita dalis formuojama žemo dažnio filtro pagalba ir pakartotinai filtruojama (antroji iteracija), taikant tiek žemo dažnio filtrą, tiek aukšto dažnio filtrą. Kiekvienos iteracijos metu (žemo ir aukšto dažnio filtrų pagalba) gautus tarpinius duomenų vektorius žymėsime atitinkamai $S^{(i)}$ ir $D^{(i)}$, o jų elementų kiekį apdorojamo vektoriaus dalyje – $\downarrow 2$.

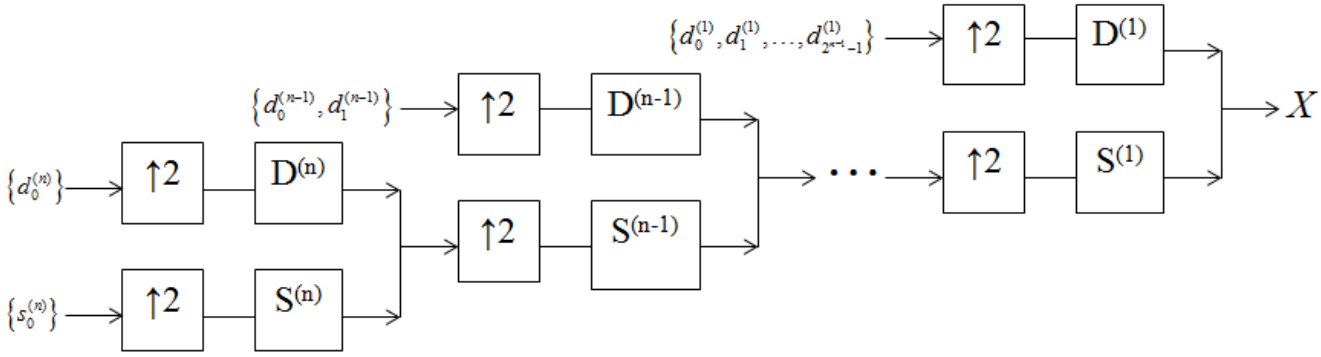
Pastebėsime, kad po antrosios iteracijos (po pakartotinio aukšto dažnio filtro panaudojimo) apskaičiuojamas dar ketvirtadalis DBT spektrinių koeficientų.



1.3 pav. Supaprastinta filtravimo schema (vienmatis atvejis)

Diskretusis DBT spektras Y suformuojamas po n iteracijų, panaudojant paskutinės iteracijos metu žemo dažnio filtro pagalba gautą koeficiento reikšmę $s^{(n)}$ ir visus aukšto dažnio filtro pagalba gautus koeficientus ($d^{(n)}, \ldots, d^{(2)}, d^{(1)}$).

Vaizdo atkūrimo schema (1.4 pav.) vadinama atvirkštine diskrečiąja bangelių transformacija. Atvirkštinė schema veikia analogiškai tiesioginei, filtruojant koeficietus atitinkamais atkūrimo (sintezės) filtrais.



1.4 pav. Supaprastinta vaizdo atkūrimo (filtravimo) schema

Remiantis žemo ir aukšto dažnių filtrų samprata bei jų interpretacija yra konstruojamos itin efektyvios diskrečiųjų bangelių transformacijų (Harro, Morlet, Le Gall, Daubechies) apskaičiavimo schemos. Diskrečiosios Le Gall bangelių transformacijos specifiką aptarsime kituose skyreliuose.

Diskrečiąją bangelių transformaciją galima apibendrinti dvimačiams skaitmeniniams vaizdams.

Imkime duomenų masyvą (dvimatį skaitmeninį vaizdą)

$$[X(m_1, m_2)] = \begin{pmatrix} X(0,0) & X(0,1) & \ldots & X(0, N-1) \\ X(1,0) & X(1,1) & \ldots & X(1, N-1) \\ \ldots & \ldots & \ldots & \ldots \\ X(N-1,0) & X(N-1,0) & \ldots & X(N-1, N-1) \end{pmatrix}.$$

Dvimatė DBT šiam vaizdui apibrėžiama tokiu būdu:

$$\left[Y\left(k_1, k_2\right)\right] = \frac{1}{N^2} \cdot T \cdot \left[X(m_1, m_2)\right] \cdot T^T; \qquad \textbf{(1.15)}$$

čia T – transformacijos matrica $(N \times N)$, $N = 2^n$ ir $n \in \square$.

Praktinė dvimatės DBT realizacija suvedama į $2N$ -kartinį vienmatės DBT panaudojimą, būtent:

(1) Pirmiausia apdorojami vienmačiai duomenų vektoriai $[X(m_1, m_2)]$, esant fiksuotoms indekso m_2 reikšmėms $\left(m_2 \in \{0,1,...,N-1\}\right)$, t.y jiems taikoma vienmatė DBT. Gaunamas tarpinis duomenų masyvas $[Y(k_1, m_2)]$.

(2) Toliau, esant fiksuotoms k_1 reikšmėms $\left(k_1 \in \{0,1,...,N-1\}\right)$ apdorojamos tarpinio masyvo eilutės, t.y. joms taikoma irgi vienmatė DBT. Po pastarojo etapo gaunamas pradinio vaizdo $\left[X(m_1,m_2)\right]$ dvimatis diskretusis DBT spektras $\left[Y(k_1,k_2)\right]$.

Daugeliu atveju (išskyrus sveikaskaitines DBT) eilučių ir stulpelių apdorojimo tvarka galutiniam rezultatui įtakos neturi.

1.3 DISKREČIOJI SVEIKASKAITINĖ LE GALL BANGELIŲ TRANSFORMACIJA (DLGT): SAVYBĖS, SKAIČIAVIMO ALGORITMAI

1.3.1 BAZINĖ DLGT

Diskrečioji Le Gall bangelių transformacija (DLGT) apibūdinama baigtiniais mastelio funkcijos ir bangelės funkcijos koeficientų rinkiniais, būtent: mastelio funkcija (žemo dažnio filtras) – $h_0 = -1/8$, $h_1 = 1/4$, $h_2 = 3/4$, $h_3 = 1/4$, $h_4 = -1/8$, o bangelės funkcija (aukšto dažnio filtras) – $g_0 = -1/2$, $g_1 = 1$, $g_2 = -1/2$.

Atitinkamai, įvairių eilių DLGT transformacijos matricos turi pavidalą:

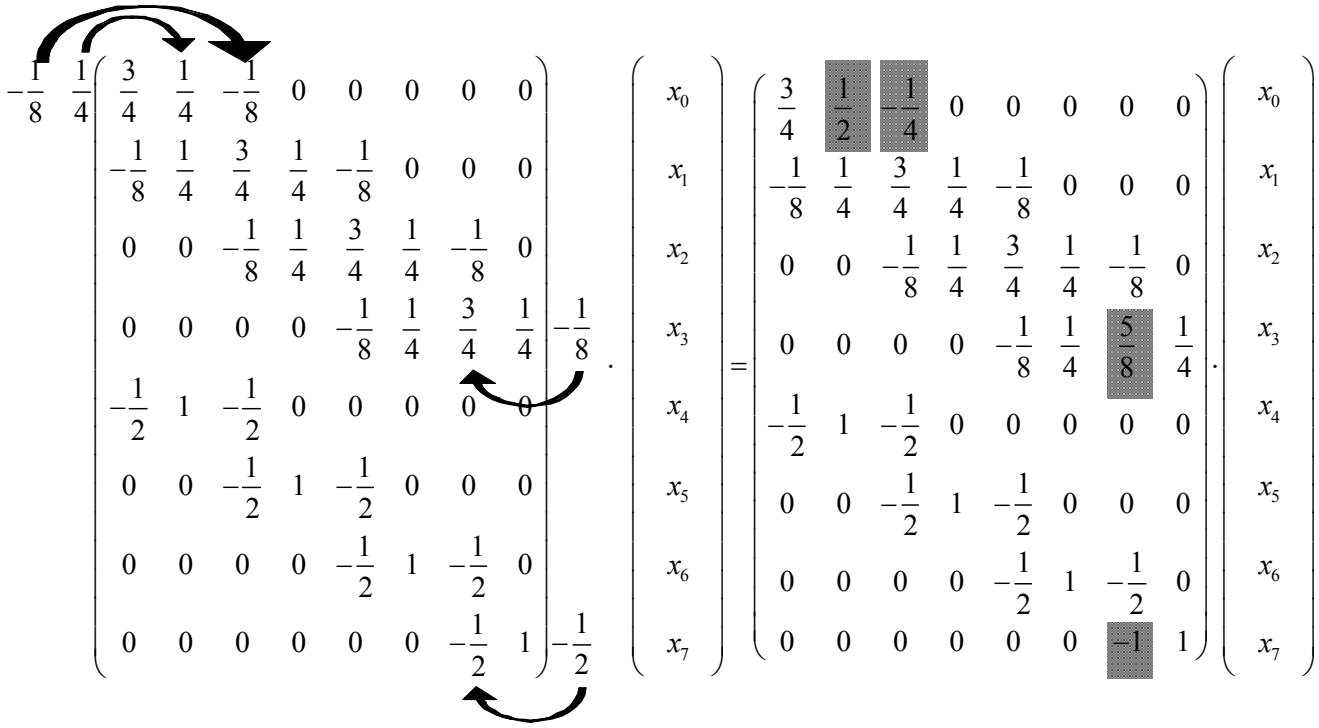
$$T_{DLGT}(1) = \begin{matrix} -\frac{1}{8} & \frac{1}{4} \end{matrix} \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \\ -\frac{1}{2} & 1 \end{pmatrix} \begin{matrix} -\frac{1}{8} \\ -\frac{1}{2} \end{matrix},$$

$$T_{DLGT}(2) = \begin{matrix} -\frac{1}{8} & \frac{1}{4} \end{matrix} \begin{pmatrix} \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 \\ -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} & 1 \end{pmatrix} \begin{matrix} -\frac{1}{8} \\ \\ \\ -\frac{1}{2} \end{matrix},$$

$$T_{DLGT}(3) = \begin{matrix} -\frac{1}{8} \\ \frac{1}{4} \\ \\ \\ \\ \\ \\ \\ -\frac{1}{8} \\ \\ \\ \\ \\ \\ -\frac{1}{2} \end{matrix} \begin{pmatrix} \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & \\ 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ -\frac{1}{8} \\ \\ \\ \\ -\frac{1}{2} \end{matrix}$$

ir taip toliau.

Iš pateiktų transformacijos matricų išraiškų nesunku pastebėti, jog skaičiuojant DLGT susiduriama su taip vadinama signalo (duomenų vektoriaus) „krašto“ problema („pritrūkstama“ duomenų realizuojant matricinius veiksmus), kuri gali būti sprendžiama įvairiais būdais (duomenų vektorių laikant periodiniu, vektoriaus galuose taikant „veidrodinį“ atspindį ir pan.). Dažniausiai „krašto“ problema sprendžiama panaudojant „veidrodinį“ atspindį abiejuose apdorojamo duomenų vektoriaus (vaizdo) galuose (1.5 pav.). Tai reiškia, kad elementus „išeinančius“ už transformacijos matricos ribų pridedame prie elementų esančių matricos viduje, atitinkamai pradedant antruoju ar priešpaskutiniuoju stulpeliu.



1.5 pav. DLGT „krašto“ problemos sprendimas ($N = 8$)

Imkime vienmatį skaitmeninį vaizdą (duomenų vektorių)

$$X = (x_0 x_1 x_2 \ldots x_{N-2} x_{N-1})^T = (o_0^{(0)} e_0^{(0)} o_1^{(0)} e_1^{(0)} \ldots o_{N/2-1}^{(0)} e_{N/2-1}^{(0)})^T ,$$

kai $N = 2^n$ ir $n \in$ □ .

DLGT apskaičiavimui taikysime iteracinę procedūrą. Tegu

$$S^{(i)} = (s_0^{(i)} s_1^{(i)} s_2^{(i)} \ldots s_{2^{n-i}-1}^{(i)})^T = (o_0^{(i)} e_0^{(i)} o_1^{(i)} e_1^{(i)} \ldots o_{2^{n-i-1}-1}^{(i)} e_{2^{n-i-1}-1}^{(i)})^T$$

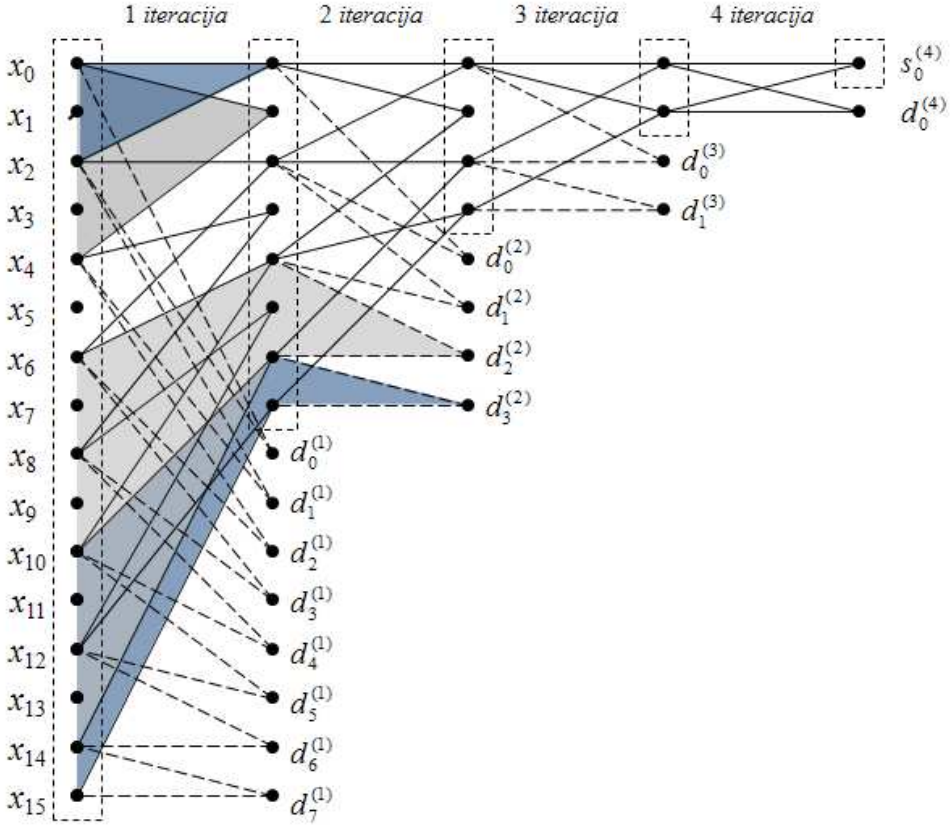
ir

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

žymi po *i*-osios iteracijos (žemo ir aukšto dažnio filtrų pagalba) gautus tarpinius duomenų vektorius; $i \in \{1,2,\ldots,n\}$ ir $S^{(0)} = X$.

Diskretusis sveikaskaitinis Le Gall spektras Y duomenų vektoriui $X = (x_0 x_1 x_2 \ldots x_{N-1})^T$ gaunamas po *n* iteracijų ir suformuojamas tokiu būdu (1.6 pav.):

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



1.6 pav. DLGT spektro apskaičiavimo signalinis grafas $(N=16)$

DLGT skaičiavimui naudojame greito skaičiavimo algoritmą (Lifting Scheme, [2]), skirtą darbui su sveikaisiais skaičiais. Gautasis spektras bus sudarytas irgi iš sveikųjų skaičių.

Algoritmo darbą nusako šios išraiškos:

$$d_k^{(i)} = e_k^{(i-1)} - \left\lfloor \frac{1}{2}\left(o_k^{(i-1)} + o_{k+1}^{(i-1)}\right) \right\rfloor, \tag{1.16}$$

$$s_k^{(i)} = o_k^{(i-1)} + \left\lfloor \frac{1}{4}\left(d_{k-1}^{(i)} + d_k^{(i)}\right) + \frac{1}{2} \right\rfloor, \tag{1.17}$$

su visais $k \in 0,1,\ldots 2^{n-i}-1$; beje, $o_{2^{n-i}}^{(i-1)} := o_{2^{n-i}-1}^{(i-1)}$, $d_{-1}^{(i)} := d_0^{(i)}$ ir $i \in \{1,2,\ldots,n\}$. Pastebėsime, jog norint apskaičiuoti s_k, reikia visų pirma apskaičiuoti d_k; (1.16) ir (1.17) išraiškose $\lfloor x \rfloor$ žymi skaičiaus x sveikąją dalį.

Atvirkštinei sveikaskaitinei DLGT apskaičiuoti skirtos šios išraiškos:

$$o_k^{(i-1)} = s_k^{(i)} - \left\lfloor \frac{1}{4}\left(d_{k-1}^{(i)} + d_k^{(i)}\right) + \frac{1}{2} \right\rfloor, \tag{1.18}$$

$$e_k^{(i-1)} = d_k^{(i)} + \left\lfloor \frac{1}{2}\left(o_k^{(i-1)} + o_{k+1}^{(i-1)}\right) \right\rfloor, \qquad \textbf{(1.19)}$$

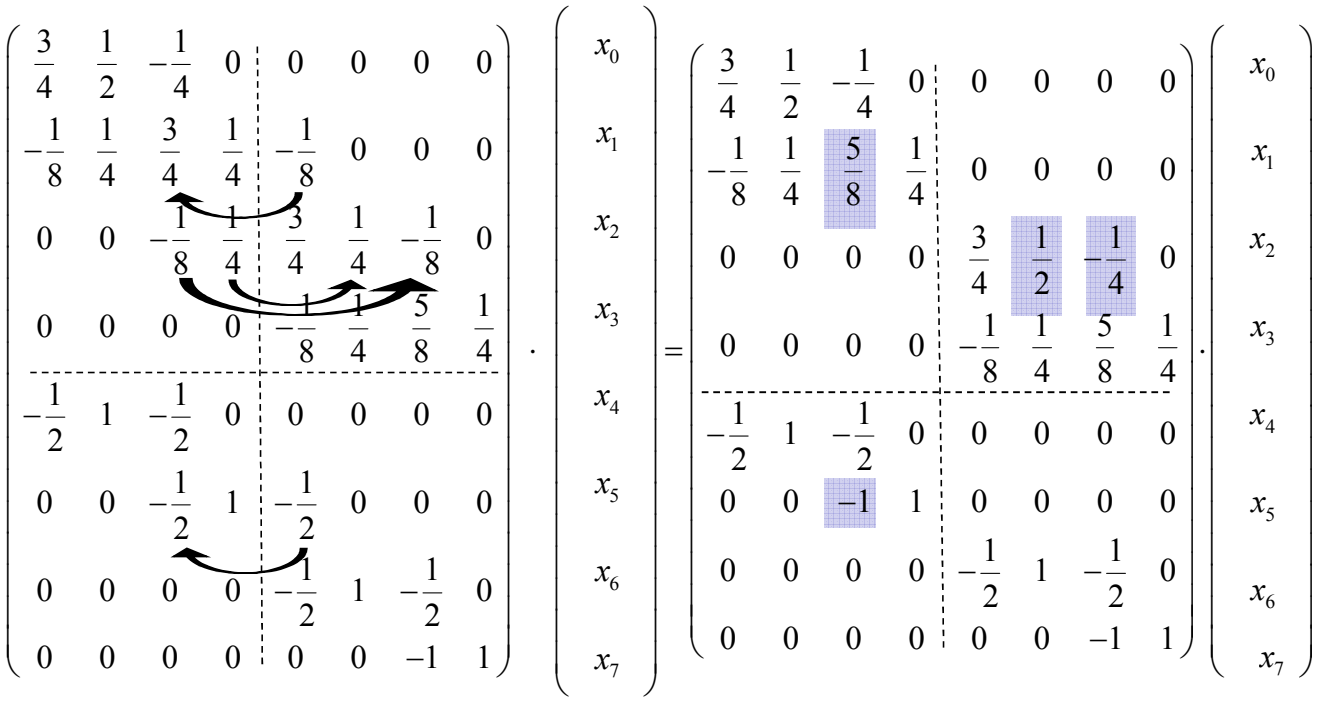
su visais $k \in 0,1,\ldots 2^{n-i} - 1$ ir $i \in \{1,2,\ldots,n\}$. Taigi, norint apskaičiuoti e_k, reikia visų pirma apskaičiuoti o_k.

Būtina pabrėžti, jog kiekvieno DLGT spektrinio koeficiento skaitinė reikšmė nusakoma tam tikru (griežtai fiksuotu) duomenų vektoriaus X elementų poaibiu (1.6 pav.), t.y. po kiekvienos iteracijos aukšto dažnio filtro (bangelės funkcija) pagalba gautus spektrinius koeficientus (jų skaitines reikšmes) įtakoja vaizdo (duomenų vektoriaus) poaibiai, kurių sąjunga sutampa su duomenų vektoriumi X ir kurie tarpusavyje (dalinai!) persidengia. Ši situacija įvairiuose taikymuose įvardijama kaip DLGT dalinio lokalizavimo erdvėje savybė.

Palyginimui pastebėsime, jog diskrečioji Harro bangelių transformacija (1.2 skyrelis) yra pilnai lokalizuota erdvėje, t.y. po atskiros iteracijos gautų spektrinių Harro koeficientų skaitinės reikšmės nusakomos nepersidengiančiais duomenų vektoriaus (vaizdo) fragmentais, kurių sąjunga sutampa su apdorojamu vektoriumi.

1.3.2 SPEKTRINIŲ DLGT KOEFICIENTŲ DEKORELIACIJA AUKŠTUOSE DAŽNIUOSE – MODIFIKUOTA SVEIKASKAITINĖ DLGT

Ankstesniajame skyrelyje paminėta DLGT (skirtos darbui tiek su realiaisiais skaičiais, tiek su sveikaisiais skaičiais) dalinio lokalizavimo erdvėje savybė siaurina šios transformacijos praktinio panaudojimo sritį – iškyla problemų, realizuojant progresyvųjį vaizdų kodavimą, vaizdų klasifikavimą, požymių išskyrimą, defektų aptikimą tekstūriniuose paviršiuose ir pan. Siekiant praplėsti DLGT taikymo ribas, buvo pabandyta ją modifikuoti (pagerinti spektrinių DLGT koeficientų dekoreliaciją aukštuose dažniuose). Esmė – pradinis vaizdas X padalijamas į baigtinį nepersidengiančių dydžio 2^p $(1 \le p \le n-1)$ blokų. „Krašto“ problema sprendžiama jau ne viso apdorojamo duomenų vektoriaus (vaizdo) rėmuose, o kiekvieno bloko „galuose“ (1.7 pav.).



1.7 pav. Modifikuota DLGT – „krašto“ problemos sprendimas ($N=8,\ p=2$)

Buvo sudaryta tokia modifikuotos sveikaskaitinės DLGT greito apskaičiavimo vienmačiam vaizdui

$$X=(x_0x_1x_2\ldots x_{N-2}x_{N-1})^T=(o_0^{(0)}e_0^{(0)}o_1^{(0)}e_1^{(0)}\ldots o_{N/2-1}^{(0)}e_{N/2-1}^{(0)})^T$$

algoritmo schema ($N=2^n$ ir $n\in$):

$$d_k^{(i)}=\begin{cases}e_k^{(i-1)}-o_k^{(i-1)}, & k\in\left\{\alpha_i\cdot t-1\middle|t=1,2,\ldots,\beta_i\right\},\\ e_k^{(i-1)}-\left\lfloor\dfrac{1}{2}\left(o_k^{(i-1)}+o_{k+1}^{(i-1)}\right)\right\rfloor, & \text{kitu atveju,}\end{cases} \tag{1.20}$$

$$s_k^{(i)}=\begin{cases}o_k^{(i-1)}+\left\lfloor\dfrac{1}{2}d_k^{(i)}+\dfrac{1}{2}\right\rfloor, & k\in\left\{\alpha_i\cdot(t-1)\middle|t=1,2,\ldots,\beta_i\right\}\\ o_k^{(i-1)}+\left\lfloor\dfrac{1}{4}\left(d_{k-1}^{(i)}+d_k^{(i)}\right)+\dfrac{1}{2}\right\rfloor, & \text{kitu atveju,}\end{cases} \tag{1.21}$$

su visais $k\in 0,1,\ldots 2^{n-i}-1$ ir $i\in\{1,2,\ldots,n\}$; čia $\alpha_i=2^{p-i}$, $\beta_i=2^{n-p}$, kai $i\in 1,2,\ldots,p$, ir $\alpha_i=1$, $\beta_i=2^{n-i}$, kai $i\in p+1,p+2,\ldots,n$. Beje, norint apskaičiuoti s_k, reikia visų pirma apskaičiuoti d_k; (1.20) ir (1.21) išraiškose $\lfloor x\rfloor$ žymi sveikąją skaičiaus x dalį.

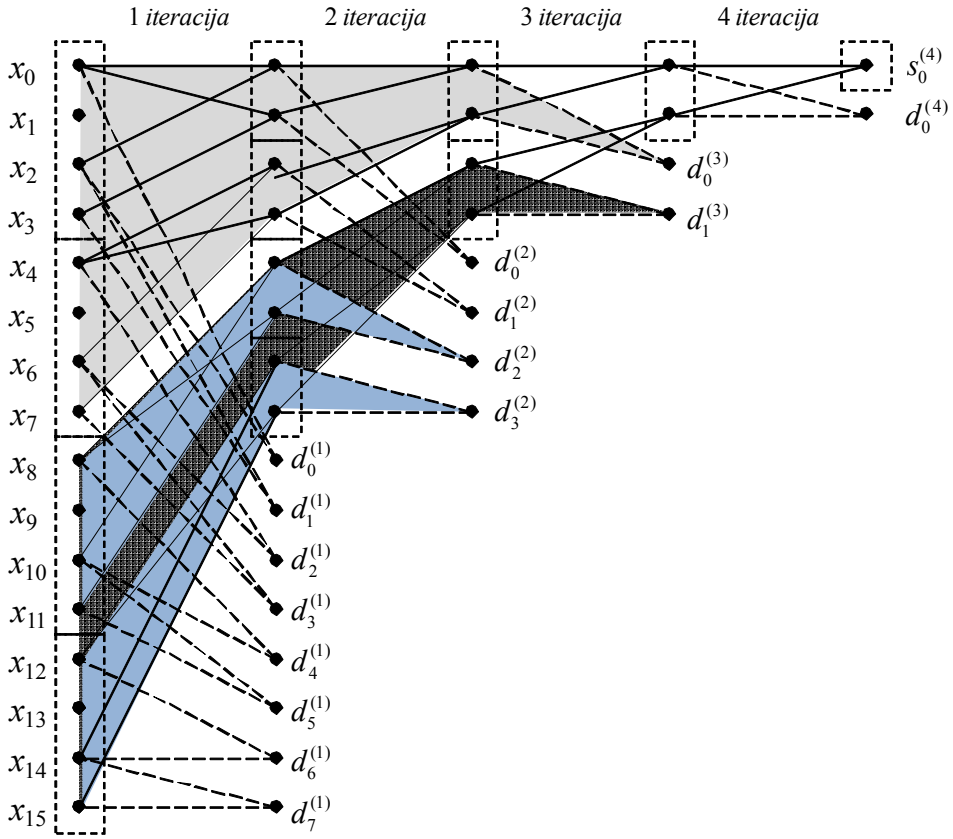
Modifikuotas greito atvirkštinės DLGT apskaičiavimo algoritmas nusakomas išraiškomis:

$$o_k^{(i-1)} = \begin{cases} s_k^{(i)} - \left\lfloor \frac{1}{2} d_k^{(i)} + \frac{1}{2} \right\rfloor, & k \in \left\{ \alpha_i \cdot (t-1) \middle| t = 1,2,\ldots,\beta_i \right\}, \\ s_k^{(i)} - \left\lfloor \frac{1}{4} \left(d_{k-1}^{(i)} + d_k^{(i)} \right) + \frac{1}{2} \right\rfloor, & \text{kitu atveju,} \end{cases} \tag{1.22}$$

$$e_k^{(i-1)} = \begin{cases} d_k^{(i)} + o_k^{(i-1)}, & k \in \left\{ \alpha_i \cdot t - 1 \middle| t = 1,2,\ldots,\beta_i \right\}, \\ d_k^{(i)} + \left\lfloor \frac{1}{2} \left(o_k^{(i-1)} + o_{k+1}^{(i-1)} \right) \right\rfloor, & \text{kitu atveju,} \end{cases} \tag{1.23}$$

su visais $k \in 0,1,\ldots 2^{n-i} - 1$ ir $i \in \{1,2,\ldots,n\}$. Vėlgi, norint apskaičiuoti e_k, reikia visų pirma apskaičiuoti o_k; (1.22) ir (1.23) išraiškose $\lfloor x \rfloor$ žymi sveikąją skaičiaus x dalį.

Akivaizdu, jog pradedant *p*-ąją iteracija, užtikrinamas pilnas lokalizavimas erdvėje, t.y. bet kuris DLGT spektrinis koeficientas $d_j^{(i)}$ $(i \in \{p, p+1,\ldots,n\},\ j \in \{0,1,\ldots,2^{n-i}-1\})$ tampa susijęs tik su vieninteliu vaizdo (duomenų vektoriaus) X bloku $X_j^{(i)} = (x_{2^i \cdot j}\ x_{2^i \cdot j+1}\ \ldots\ x_{2^i \cdot (j+1)-1})^T$.



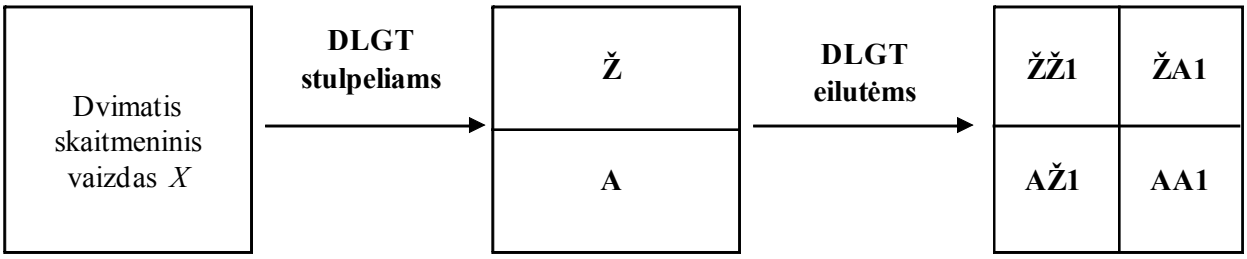
1.8 pav. Modifikuotos DLGT spektrinių koeficientų apskaičiavimo schema ($N = 16$, $p = 2$)

Kita vertus, esant fiksuotam i $(p \le i \le n)$, su spektriniais koeficientais $\{d_j^{(i)} \mid j = 0,1,\ldots,2^{n-i}-1\}$ susiję vaizdo (duomenų vektoriaus) X blokai nepersidengia, o jų sąjunga sutampa su vaizdu X (1.8 pav.).

1.3.3 DVIMATĖ DLGT

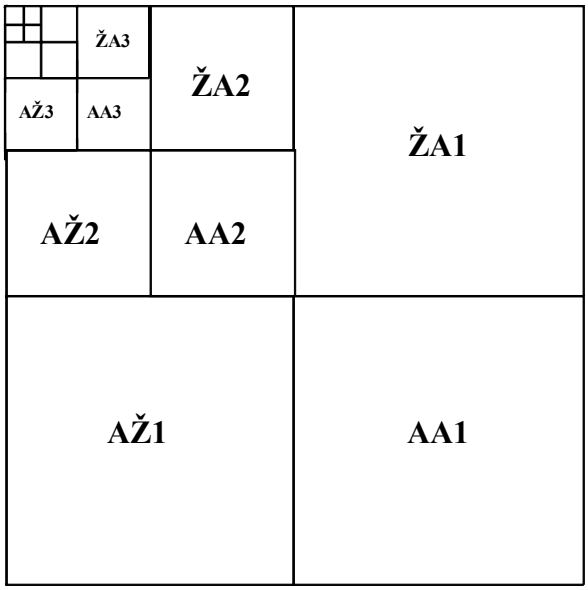
Dvimatę diskrečiąją (sveikaskaitinę) Le Gall transformaciją nesunkiai įgyvendinsime $2N$ kartų taikydami vienmatę DLGT (šį atvejį jau aprašėme 1.2 skyrelyje).

Taigi, atliekant vienmatę DLGT stulpeliui yra suformuojami žemo (Ž) ir aukšto dažnio (A) komponentės. Tuomet atliekama vienmatė DLGT eilutėms ir suformuojamas tarpinis duomenų masyvas (ŽŽ, AŽ, ŽA, AA; (1.9 pav.)).



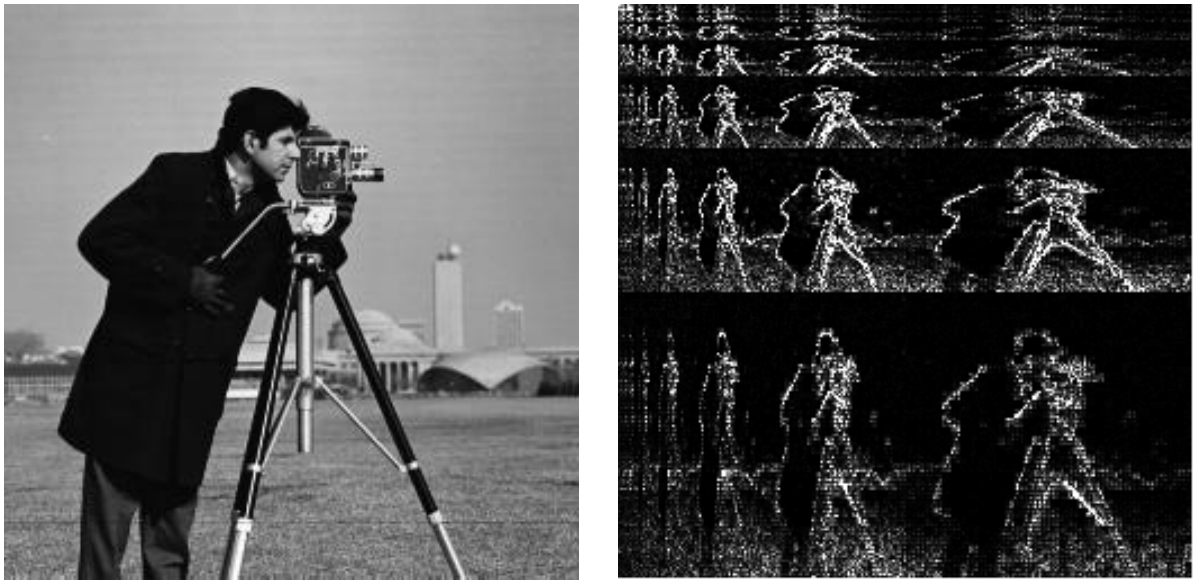
1.9 pav. Aukšto ir žemo dažnio komponenčių išsidėstymas spektre

Toliau procedūra ŽŽ komponentėms kartojama reikiama skaičių kartų (1.10 pav.).



1.10 pav. Dvimačio DLGT spektro hierarchija

Vizuali šios procedūros interpretacija konkrečiam vaizdui parodyta 1.11 pav.



(a) (b)

1.11 pav. Dvimačio spektro hierarchija: (a) vaizdas *Cameraman.bmp*; (b) vaizdo DLGT spektras

Pastebėsime dar kartą, jog skaičiuojant sveikaskaitinį Le Gall spektrą dvimačiam vaizdui, svarbu įvertinti tą faktą, jog asociatyvumo savybė ((1.15) išraiška, 1.2 skyrelis) jau negalioja. Tai reiškia, jog vienmatės DLGT taikymo tvarka (apdorojamo masyvo eilutėms ir stulpeliams) turi būti suderinta, atliekant tiesioginę ir atvirkštinę dvimates sveikaskaitines DLGT.

2. MODIFIKUOTOS DLGT APSKAIČIAVIMO VAIZDO FRAGMENTAMS ALGORITMŲ SUDARYMAS IR JŲ EFEKTYVUMO TYRIMAS

1.3.2 skyrelyje įvesta sveikaskaitinės DLGT modifikacija leido sudaryti ir realizuoti efektyvų pasirinkto vaizdo fragmento spektro apskaičiavimo procedūrą. Siūloma procedūra remiasi prielaida, kad pradinio vaizdo modifikuotas DLGT spektras yra žinomas.

2.1 DLGT SPEKTRO PASIRINKTIEMS VIENMAČIO VAIZDO FRAGMENTAMS RADIMAS

Imkime vienmatį skaitmeninį vaizdą (signalą) $X = (x_0 x_1 x_2 \ldots x_{N-1})^T$. Tarkime, kad modifikuotas sveikaskaitinis DLGT spektras

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

yra žinomas.

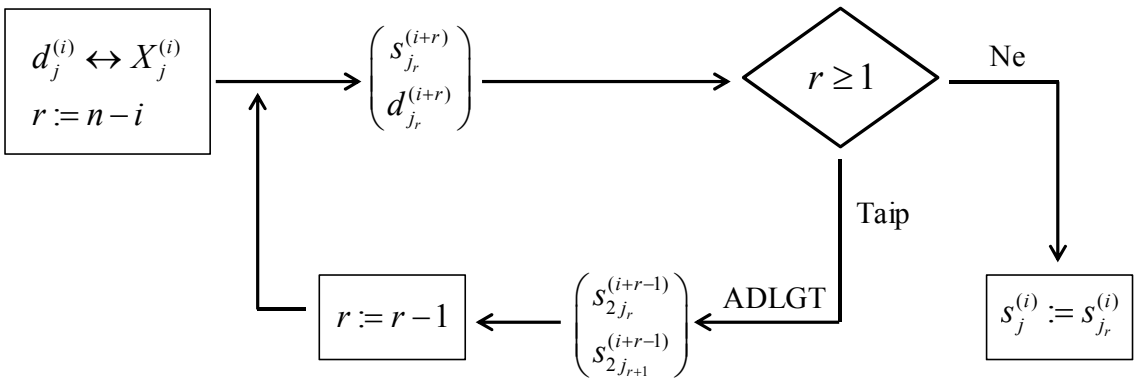
1.3.2 skyrelyje užsiminėme, kad bet kuris DLGT spektrinis koeficientas $d_j^{(i)}$ $(i \in \{p, p+1, \ldots, n\}$, $j \in \{0,1,\ldots,2^{n-i}-1\})$ yra susijęs tik su vieninteliu vaizdo X fragmentu $X_j^{(i)} = (x_{2^i \cdot j} \; x_{2^i \cdot j+1} \ldots \; x_{2^i \cdot (j+1)-1})^T$. Fragmento $X_j^{(i)}$ (modifikuotą) spektrą pažymėkime $Y_j^{(i)}$.

2.1.1 ir 2.1.2 skyreliuose pateiksime vienmačio vaizdo X fragmento $X_j^{(i)}$ spektro $Y_j^{(i)}$ apskaičiavimo bendrą schemą ir algoritmus.

2.1.1 BENDRA SCHEMA

Norint rasti vienmačio vaizdo X fragmento $X_j^{(i)}$ spektrą $Y_j^{(i)}$, reikia realizuoti tokius etapus:

1. Pirmasis spektro $Y_j^{(i)}$ DLGT koeficientas $s_j^{(i)}$ (poaibiui $X_j^{(i)}$) apskaičiuojamas pagal 2.1 pav. pateiktą schemą, kur $j_0 = j$, $j_r = \lfloor j_{r-1} / 2 \rfloor$, su visais $r = 1,2,\ldots n-i$; čia $\lfloor x \rfloor$ žymi sveikąją skaičiaus x dalį.



2.1 pav. Bendra spektro $Y_j^{(i)}$ elemento $s_j^{(i)}$ radimo schema

2. Likusieji spektro $Y_j^{(i)}$ elementai atrenkami iš vaizdo X modifikuoto DLGT spektro Y, t.y.

$$\left\{d_j^{(i)}, d_{2j}^{(i-1)}, d_{2j+1}^{(i-1)}, d_{4j}^{(i-2)}, \ldots, d_{2^{i-1}\cdot j}^{(1)}, d_{2^{i-1}\cdot j+1}^{(1)}, \ldots, d_{2^{i-1}(j+1)-1}^{(1)}\right\}.$$

(Modifikuotas) diskretusis Le Gall spektras $Y_j^{(i)}$ vaizdo fragmentui $X_j^{(i)}$ suformuojamas taip:

$$Y_j^{(i)} = (s_j^{(i)} d_j^{(i)} d_{2j}^{(i-1)} d_{2j+1}^{(i-1)} d_{4j}^{(i-2)} \ldots d_{2^{i-1}\cdot j}^{(1)} d_{2^{i-1}\cdot j+1}^{(1)} \ldots d_{2^{i-1}(j+1)-1}^{(1)})^T.$$

Taigi, esminis (tiek sudėtingumo, tiek laikinių sąnaudų prasme) yra pirmasis algoritmo etapas.

2.1.2 GREITAS ALGORITMAS

Išanalizavus bendrą schemą (2.1 pav.), t.y. atitinkamai apjungus iteraciniu būdu atliekamus veiksmus galima pateikti DLGT spektro $Y_j^{(i)}$ apskaičiavimo pasirinktam vienmačio vaizdo X poaibiui (fragmentui) $X_j^{(i)}$ algoritmą tokiu būdu:

1. Pirmasis spektro $Y_j^{(i)}$ koeficientas $s_j^{(i)}$ apskaičiuojamas pagal formulę

$$s_j^{(i)} = s_0^{(n)} + \sum_{r=1}^{n-i}\left(\frac{1-(-1)^{j_{r-1}}}{2}\cdot d_{j_r}^{(i+r)} - \left\lfloor \frac{1}{2} d_{j_r}^{(i+r)} + \frac{1}{2} \right\rfloor\right), \qquad \textbf{(2.1)}$$

kur $i \in \{p, p+1, \ldots, n\}$ ir $j \in \{0,1,\ldots,2^{n-i}-1\}$, o $j_0 = j$, $j_r = \lfloor j_{r-1}/2 \rfloor$, su visais $r = 1,2,\ldots n-i$.

2. Likusieji spektriniai DLGT koeficientai išrenkami iš pradinio spektro Y (žiūrėti 2.1.1 skyrelį).

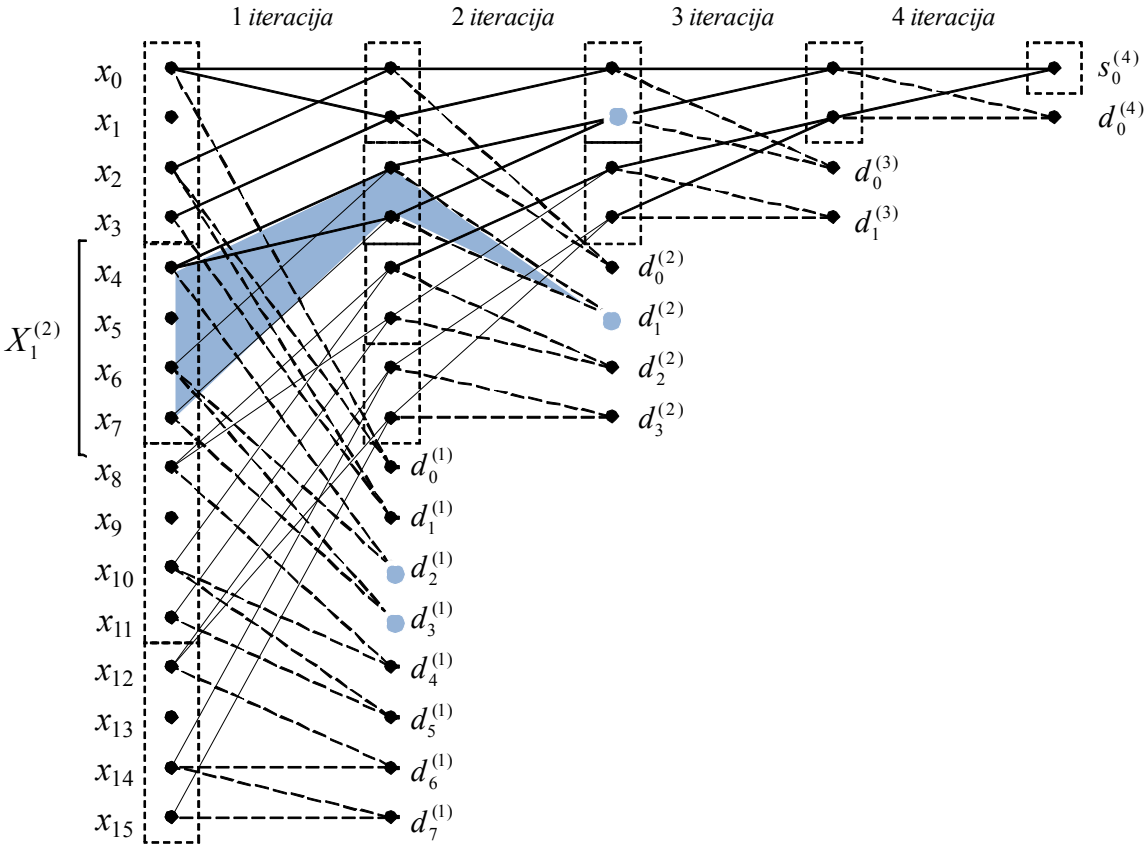
Vizualus greito modifikuoto DLGT spektro radimas pavaizduotas 2.2 pav. Tarkime, kad vaizdo (duomenų vektoriaus) ilgis yra $N = 16 = 2^4$; vadinasi $n = 4$, dekoreliacijos lygį parenkame $p = 2$. Jau po antrosios iteracijos DLGT spektriniai koeficientai tenkina pilno lokalizavimo erdvėje savybę ir juos galime susieti su pradinio vektoriaus fragmentais.

Pavyzdžiui, imkime spektrinį koeficientą $d_1^{(2)}$. Nesunkiai randame su juo susijusį pradinio vektoriaus fragmentą $X_1^{(2)} = \{x_4 x_5 x_6 x_7\}$. Šiam fragmentui realizuosime greitą jo DLGT spektro apskaičiavimo algoritmą.

Randame pirmąjį spektrinį koeficientą $s_1^{(2)}$ (pagal (2.1) formulę). Kadangi $i=2$, $j=1$, $j_0=1$, $j_1, j_2=0$; gauname

$$s_1^{(2)} = s_0^{(4)} + d_0^{(3)} - \left\lfloor \frac{1}{2} d_0^{(3)} + \frac{1}{2} \right\rfloor - \left\lfloor \frac{1}{2} d_0^{(4)} + \frac{1}{2} \right\rfloor.$$

Kaip matome, šiam koeficientui apskaičiuoti prireiks tik trijų mums žinomų modifikuoto DLGT spektro koeficientų $s_0^{(4)}$, $d_0^{(3)}$, $d_0^{(4)}$, su jau nustatytais ženklais. Likusieji elementai „paimami“ tiesiogiai iš DLGT spektro Y, t.y. $d_1^{(2)}$, $d_2^{(1)}$, $d_3^{(1)}$.



2.2 pav. Modifikuoto DLGT spektro vaizdo fragmentui radimas ($N=16$, $p=2$)

Taigi, fragmento $X_1^{(2)}$ sveikaskaitinis DLGT spektras bus $Y_1^{(2)} = \left(s_1^{(2)} d_1^{(2)} d_2^{(1)} d_3^{(1)}\right)^T$.

2.2 DLGT SPEKTRO PASIRINKTIEMS DVIMAČIO VAIZDO FRAGMENTAS (BLOKAMS) RADIMAS

Imkime dvimatį skaitmeninį (su pilka šviesos intensyvumo skale) vaizdą $[X(m_1,m_2)]$; čia $m_1,m_2 \in \{0,1,...,N-1\}$; $N=2^n$, $n \in \square$. Tarkime, kad $[Y(k_1,k_2)]$ $(k_1,k_2 \in \{0,1,...,N-1\})$ yra modifikuotas vaizdo $[X(m_1,m_2)]$ DLGT spektras.

2.2.1 DLGT KOEFICIENTŲ SĄSAJA SU VAIZDO FRAGMENTAIS

Kiekvienas DLGT spektrinis koeficientas $Y(k_1,k_2)$, $k_1,k_2 \in \{0,1,...,2^{n-p+1}-1\}$ pasižymi gana įdomiomis ir praktiniu požiūriu labai vertingomis savybėmis, būtent:

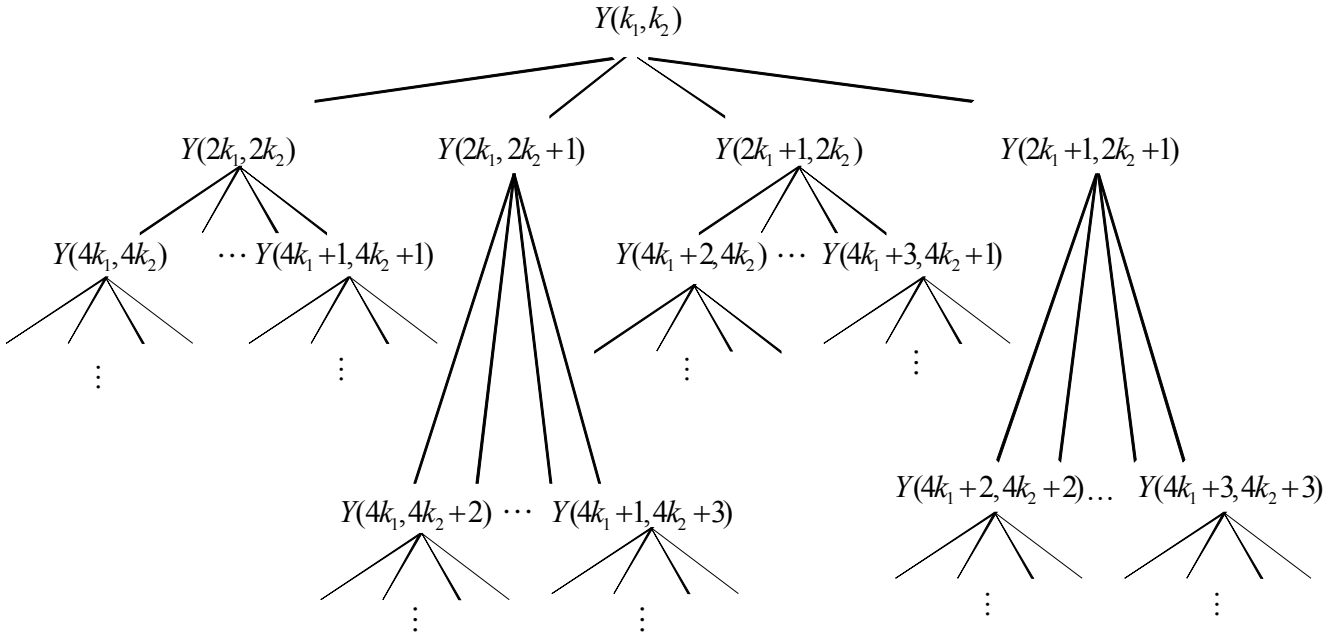
1. Koeficientas $Y(k_1,k_2)$ $(k_1=2^{n-i_1}+j_1,\quad k_2=2^{n-i_2}+j_2,\quad$ kur $i_1,i_2 \in \{1,2,\ldots,n\}$, $j_1 \in \{0,1,...,2^{n-i_1}-1\}$, $j_2 \in \{0,1,...,2^{n-i_2}-1\})$ yra susiję su vaizdo fragmentu $X^{(k_1,k_2)}=[X(\tilde{m}_1,\tilde{m}_2)]$, kur $(\tilde{m}_1,\tilde{m}_2) \in V_{k_1} \times V_{k_2}$ ir $V_{k_r}=\{j_r 2^{i_r}, j_r 2^{i_r}+1,\ldots,(j_r+1)2^{i_r}-1\}$, $r=1,2$. Kitaip tarus, koeficiento $Y(k_1,k_2)$ reikšmė yra (išskirtinai) nusakoma vaizdo fragmentu $X^{(k_1,k_2)}$.

2. Jeigu $i_1>1$ ir $i_2>1$, koeficientui $Y(k_1,k_2)$ galima priskirti kvad-medį (2.3 pav.), kurio viršūnės (spektriniai koeficientai) aprašomos aibe:

$$\left\{Y(k_1^*,k_2^*) \mid (k_1^*,k_2^*) \in \bigcup_{q=1}^{\min\{i_1-1,i_2-1\}} \left(\Im_{k_1}(q) \times \Im_{k_1}(q)\right)\right\}; \qquad \textbf{(2.2)}$$

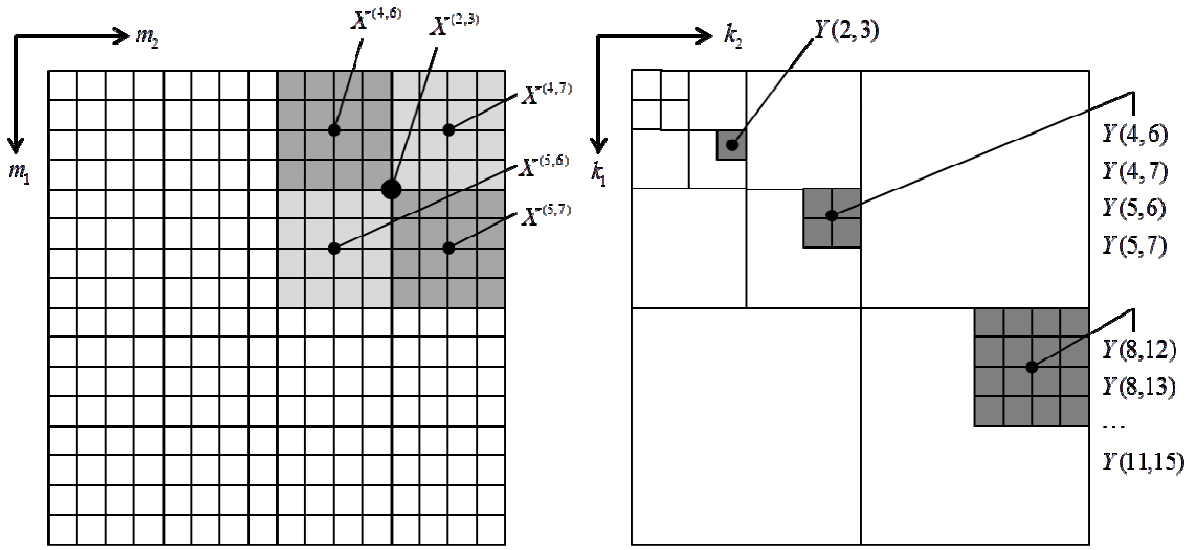
čia $\Im_{k_r}(q)=\left\{2^q k_r, 2^q k_r+1,\ldots,2^q(k_r+1)-1\right\}$, $r=1,2$.

Koeficientas $Y(k_1,k_2)$ dažnai vadinamas kvad-medžio šaknimi.



2.3 pav. Su koeficientu $Y(k_1,k_2)$ susijusio medžio struktūra

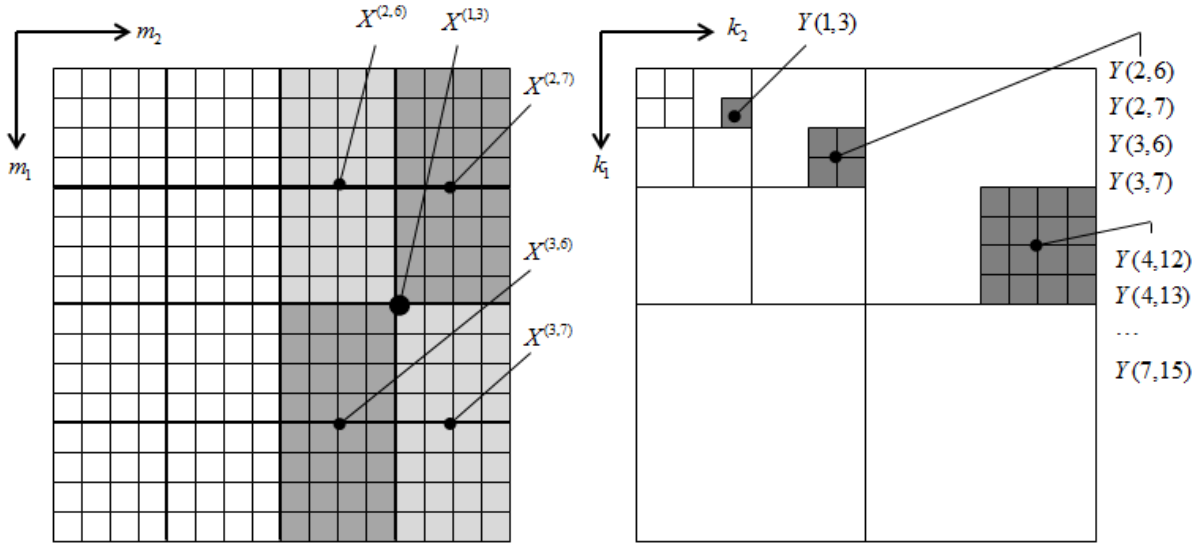
Grafinė išvardytų savybių interpretacija, kai $N=16$ ir $p=2$, pateikta 2.4 pav.



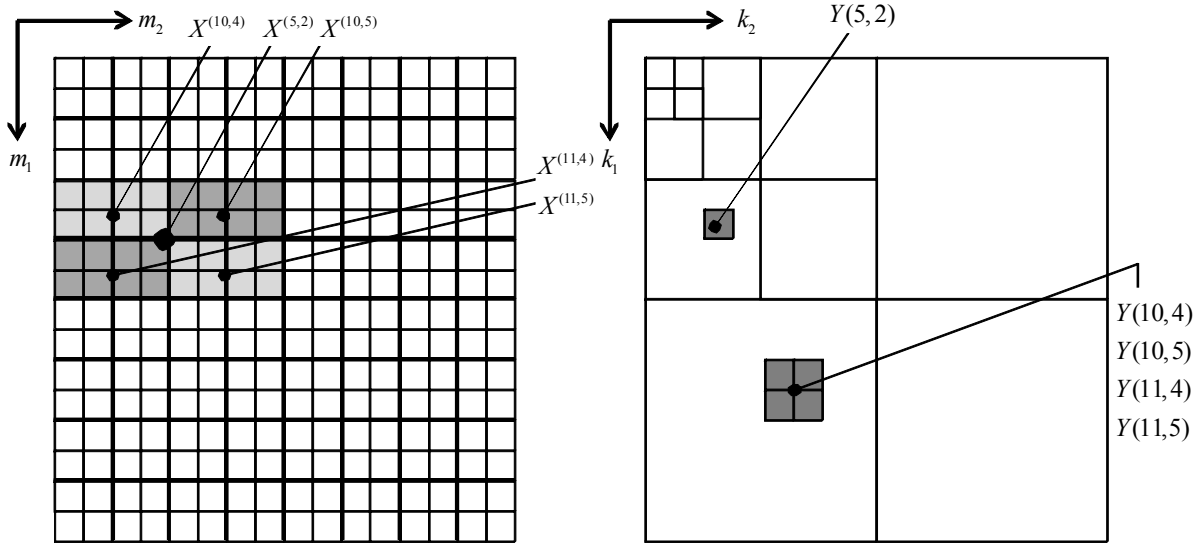
(a) (b)

2.4 pav. Grafinė spektrinių koeficientų interpretacija: (a) vaizdas $[X(m_1,m_2)]$; koeficientas (kvad-medžio šaknis) $Y(2,3)$ yra susijęs su vaizdo fragmentu $X^{(2,3)}$; (b) modifikuotas spektras $[Y(k_1,k_2)]$; koeficientui $Y(2,3)$ priskirtas kvad-medis

Idomu tai, jog fragmento orientacija priklauso nuo spektrinio koeficiento $Y(k_1,k_2)$ parinkimo vietos spektre, t.y. nuo indeksų i_1 ir i_2 (2.4, 2.5, 2.6 pav.).



2.5 pav. Koeficiento $Y(1,3)$ sąsaja su vaizdo fragmentu $X^{(1,3)}$ $(N=16,\ p=2)$



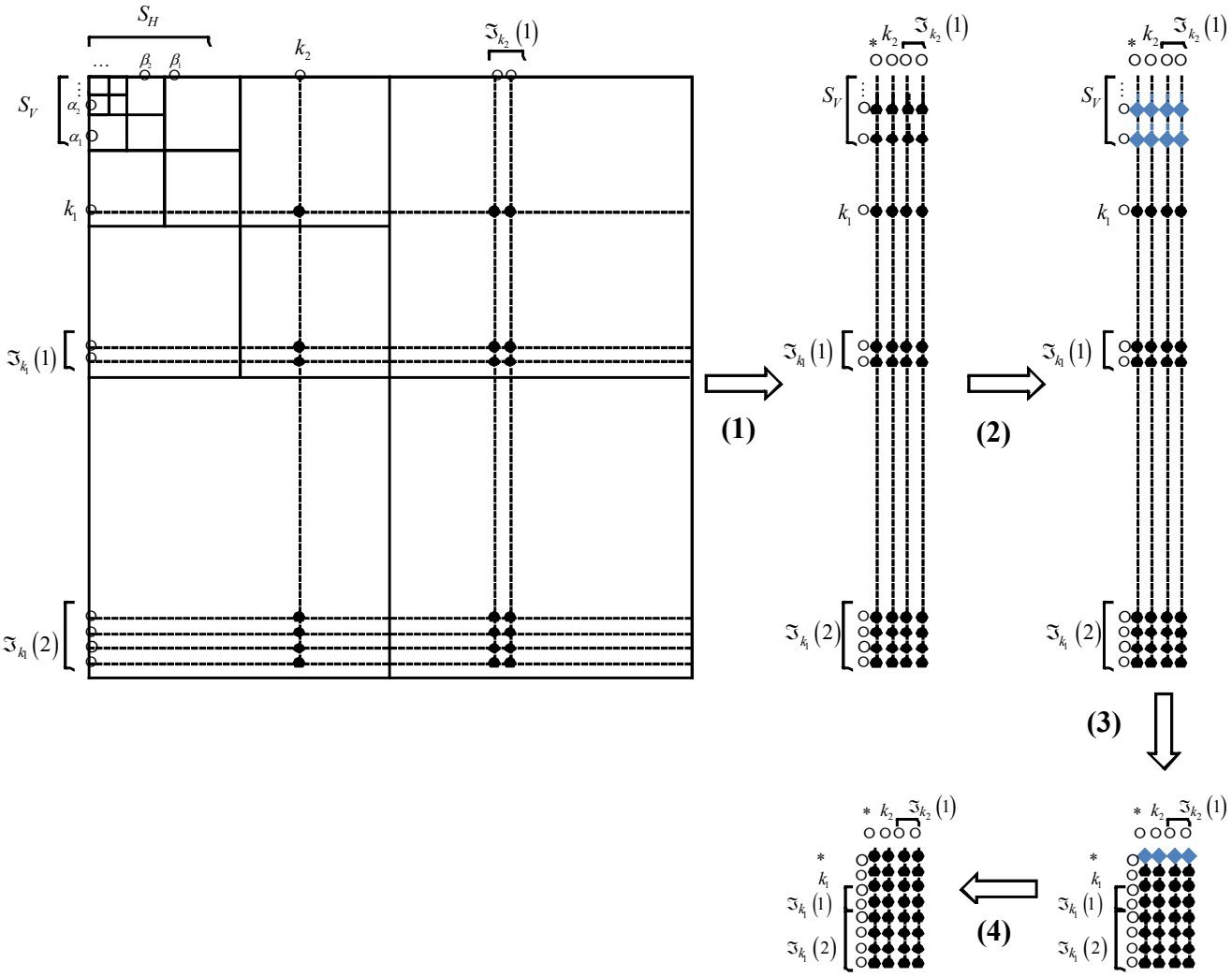
2.6 pav. Koeficiento $Y(5,2)$ sąsaja su vaizdo fragmentu $X^{(5,2)}$ $(N=16,\ p=1)$

Taigi, jei koeficientas $Y(k_1,k_2)$ yra toks, kad $i_1=i_2$, tai gauti vaizdo fragmentai bus kvadratai (2.4 pav.), jei $i_1\neq i_2$, tai vaizdo fragmentai bus stačiakampiai (su skirtingo ilgio kraštinėmis) (2.5, 2.6 pav.).

2.2.2 BENDRA SCHEMA

2.7 pav. pateikiama nauja sveikaskaitinio DLGT spektro apskaičiavimo pasirinktiems dvimačio vaizdo fragmentams (blokams) schema. Schemą galima taikyti tais atvejais, kai žinomas sveikaskaitinis modifikuotas pradinio vaizdo DLGT spektras.

Tarkime, jog pasirenkamas su sveikaskaitiniu DLGT koeficientu $Y(k_1,k_2)$ susijęs vaizdo X blokas $X^{(k_1,k_2)}$.



2.7 pav. Greito DLGT spektro vaizdo fragmentui apskaičiavimo schema

Pirmajame etape (pažymėta **(1)**, 2.7 pav.) formuojamas koeficientą $Y(1,k_2)$ atitinkančio vaizdo X fragmento (bloko) $X^{(1,k_2)}$ DLGT spektras. Tam panaudojamas 2.1.2 skyrelyje pateiktas greitas algoritmas. Antrajame etape (pažymėta **(2)**, 2.7 pav.) tam tikroms (su indeksu k_1 susijusioms) eilutėms

taikoma vienmatė atvirkštinė DLGT (detalesni komentarai pateikiami 2.2.3 skyrelyje). Trečiasis ir ketvitasis (pažymėta **(3)** ir **(4)**, 2.7 pav.) etapai skirti vaizdo bloko $X^{(k_1,k_2)}$ DLGT spektrui formuoti. Tam panaudojamas vėlgi 2.1.2 skyrelyje aprašytasis greitas algoritmas bei tiesioginė vienmatė DLGT.

Schemos (2.7 pav.) algoritmizavimas ir realizacija aprašyta 2.2.3 skyrelyje.

Atkreiptinas dėmesys į tai, jog didžioji dalis spektro $Y(k_1,k_2)$ koeficientų tiesiog atrenkami iš modifikuoto sveikaskaitinio DLGT spektro (vaizdui X).

2.2.3 GREITAS ALGORITMAS

Imkime spektrinį DLGT koeficientą $Y(k_1,k_2)$, kur $k_r = 2^{n-i_r} + j_r$ ir $j_r \in \{0,1,...,2^{n-i_r}-1\}$, $r = 1,2$. Koeficientas $Y(k_1,k_2)$ yra susijęs su vaizdo $[X(m_1,m_2)]$ fragmentu $[X^{(k_1,k_2)}(m_1,m_2)]$. Šio fragmento DLGT spektrą pažymėkime $[Y^{(k_1,k_2)}(u,v)]$, kur $u \in \{0,1,...,2^{i_1}-1\}$ ir $v \in \{0,1,...,2^{i_2}-1\}$.

Efektyvus perėjimas nuo sveikaskaitinio DLGT spektro visam vaizdui $[X(m_1,m_2)]$ prie DLGT spektro vaizdo fragmentui $[X^{(k_1,k_2)}(m_1,m_2)]$ realizuojamas tokiu būdu:

1. Formuojamos aibės:

   $S_V = \{\alpha_1,\alpha_2,\ldots,\alpha_{n-i_1}\}$; čia $\alpha_s = \lfloor \alpha_{s-1}/2 \rfloor$, $s = 1,2,\ldots,n-i_1$; $\alpha_0 = k_1$;

   $S_H = \{\beta_1,\beta_2,\ldots,\beta_{n-i_2}\}$; čia $\beta_t = \lfloor \beta_{t-1}/2 \rfloor$, $t = 1,2,\ldots,n-i_2$; $\beta_0 = k_2$;

   $\gamma_1 = \{\gamma_1(0),\gamma_1(1),\ldots,\gamma_1(n-i_1-1)\}$; čia $\gamma_1(s) = \lfloor \gamma_1(s-1)/2 \rfloor$, $s = 1,2,\ldots,n-i_1-1$; $\gamma_1(0) = j_1$;

   $\gamma_2 = \{\gamma_2(0),\gamma_2(1),\ldots,\gamma_2(n-i_2-1)\}$; čia $\gamma_2(t) = \lfloor \gamma_2(t-1)/2 \rfloor$, $t = 1,2,\ldots,n-i_2-1$; $\gamma_2(0) = j_2$;

   $\Im_{k_1} = \{k_1\} \cup \left\{ \bigcup_{q=1}^{i_1-1} \Im_{k_1}(q) \right\}$, $\Im_{k_1}(q) = \left\{ 2^q k_1, 2^q k_1 + 1, \ldots, 2^q (k_1+1) - 1 \right\}$;

   $\Im_{k_2} = \{k_2\} \cup \left\{ \bigcup_{q=1}^{i_2-1} \Im_{k_2}(q) \right\}$, $\Im_{k_2}(q) = \left\{ 2^q k_2, 2^q k_2 + 1, \ldots, 2^q (k_2+1) - 1 \right\}$.

2. Formuojamas masyvas $[\hat{A}^{(k_1,k_2)}(i,j)]$; čia $i \in \{0\} \cup S_V \cup \Im_{k_1}$, $j \in \{0\} \cup \Im_{k_2}$ (**(1)** etapas; 2.7 pav.):

$$\hat{A}^{(k_1,k_2)}(i,0) = Y(i,0) + \sum_{t=1}^{n-i_2} \left(\frac{1-(-1)^{\gamma_2(t-1)}}{2} Y(i,\beta_t) - \left\lfloor \frac{1}{2} Y(i,\beta_t) + \frac{1}{2} \right\rfloor \right), \; i \in \{0\} \cup S_V \cup \Im_{k_1}; \qquad \textbf{(2.3)}$$

$\hat{A}^{(k_1,k_2)}(i,j) = Y(i,j)$, $i \in \{0\} \cup S_V \cup \Im_{k_1}$, $j \in \Im_{k_2}$. **(2.4)**

3. Masyvo $[\hat{A}^{(k_1,k_2)}(i,j)]$ eilutėms $(\hat{A}^{(k_1,k_2)}(i,0) \cdots \hat{A}^{(k_1,k_2)}(i,j) \cdots)$, $i \in \{0\} \cup S_V$, taikoma ADLGT su įvesta modifikacija (**(2)** etapas; 2.7 pav.). Gauname:

$$\left(A^{(k_1,k_2)}(i,0) \cdots A^{(k_1,k_2)}(i,j) \cdots\right),\ i \in \{0\} \cup S_V.$$

4. Formuojamas vektorius $(B^{(k_1,k_2)}(0,0) \cdots B^{(k_1,k_2)}(0,j) \cdots)$, būtent (**(3)** etapas; 2.7.pav):

$$B^{(k_1,k_2)}(0,j) = A^{(k_1,k_2)}(0,j) + \sum_{s=1}^{n-i_1} \left(\frac{1-(-1)^{\gamma_1(s-1)}}{2} A^{(k_1,k_2)}(\alpha_s, j) - \left\lfloor \frac{1}{2} A^{(k_1,k_2)}(\alpha_s, j)) + \frac{1}{2} \right\rfloor \right), \quad \textbf{(2.5)}$$

$j \in \{0\} \cup \Im_{k_2}$.

5. Vektoriui $(B^{(k_1,k_2)}(0,0) \cdots B^{(k_1,k_2)}(0,j) \cdots)$ taikoma DLGT su modifikacija (**(4)** etapas; 2.7 pav.). Gauname:

$$\left(\hat{B}^{(k_1,k_2)}(0,0) \cdots \hat{B}^{(k_1,k_2)}(0,j) \cdots\right).$$

6. Užrašomas pasirinkto vaizdo $[X(m_1,m_2)]$ fragmento $[X^{(k_1,k_2)}(m_1,m_2)]$ sveikaskaitinis DLGT spektras $[Y^{(k_1,k_2)}(u,v)]$ $(u \in \{0,1,...,2^{i_1}-1\}, v \in \{0,1,...,2^{i_2}-1\})$, būtent:

$Y^{(k_1,k_2)}(0,0) = \hat{B}^{(k_1,k_2)}(0,0)$,

$Y^{(k_1,k_2)}(u,0) = \hat{A}^{(k_1,k_2)}(k_1^*,0)$, su visais $u \in \{1,2,...,2^{i_1}-1\}$; k_1^* yra u-tasis aibės $\Im_{k_1}$ elementas (elementų numeracija aibėje $\Im_{k_1}$ pradedama nuo vieneto).

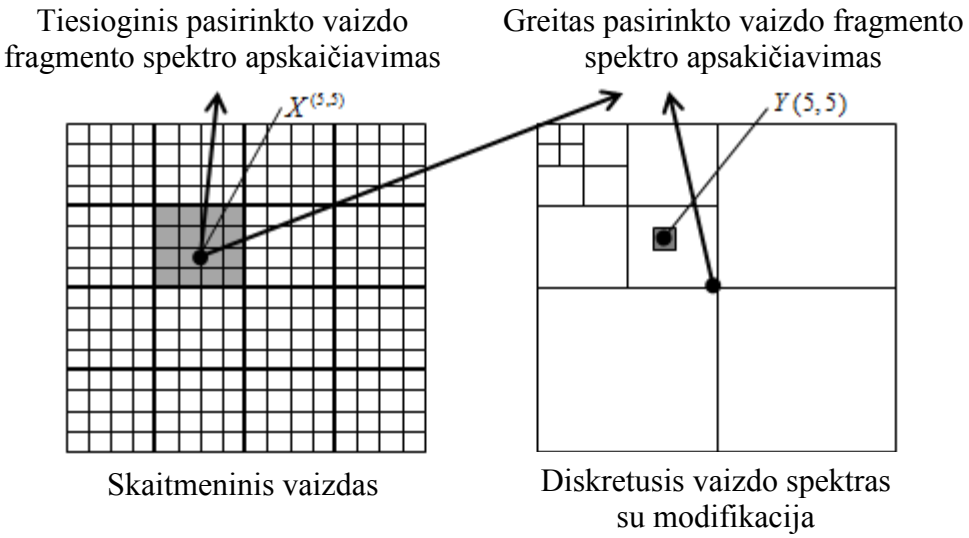
$Y^{(k_1,k_2)}(u,v) = \hat{B}^{(k_1,k_2)}(0,k_2^*)$, su visais $v \in \{1,2,...,2^{i_2}-1\}$; k_2^* yra v-tasis aibės $\Im_{k_2}$ elementas (elementų numeracija aibėje $\Im_{k_2}$ pradedama nuo vieneto).

$Y^{(k_1,k_2)}(u,v) = \hat{A}^{(k_1,k_2)}(k_1^*,k_2^*)$, su visais $u \in \{1,2,...,2^{i_1}-1\}$ ir $v \in \{1,2,...,2^{i_2}-1\}$; k_1^* ir k_2^* yra atitinkamai u-tasis ir v-tasis aibės $\Im_{k_1}$ ir $\Im_{k_2}$ elementai (elementų numeracija aibėse $\Im_{k_1}$ ir $\Im_{k_2}$ pradedama nuo vieneto).

2.3 PALYGINAMOJI EKSPERIMENTINĖ ANALIZĖ

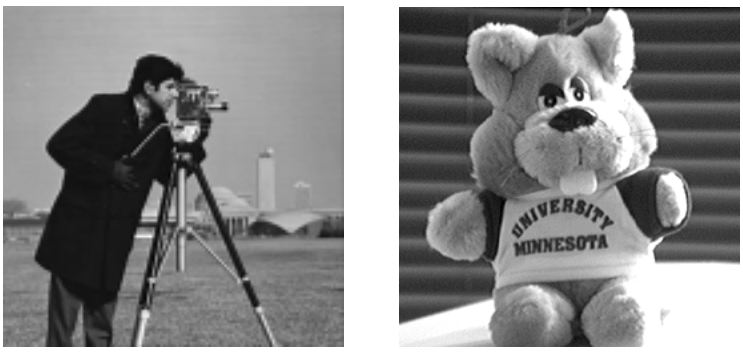
2.3.1 BENDRA SCHEMA

Darbe siekiama eksperimentiškai ištirti sudarytos sveikaskaitinio DLGT spektro apskaičiavimo pasirinktiems vaizdo fragmentams efektyvumą (2.8 pav.). Tyrimas buvo atliekamas dviem etapais. Pirmiausia, pasirinkto vaizdo fragmento DLGT spektras buvo apskaičiuojamas tiesiogiai, panaudojant standartinę modifikuotą DLGT apskaičiavimo procedūrą (1.3.2 skyrelis), po to, tam pačiam vaizdo fragmetui DLGT spektras buvo apskaičiuojamas taikant sudarytąjį algoritmą (2.2.3 skyrelis), t.y. pasinaudojant viso vaizdo *X* DLGT spektru.



2.8 pav. Analizės schema

Eksperimentui buvo panaudoti skirtingo dydžio su pilka šviesos intensyvumo skale *.bmp* formato skaitmeniniai vaizdai (2.9 pav.).



(a) (b)

(c)

(d)

2.9 pav. Skaitmeniniai vaizdai: (a) *Cameraman.bmp*, 128×128**; (b) *Toy.bmp*,** 256×256**; (c) *Boat.bmp*,** 512×512**; (c) *Man.bmp*,** 1024×1024

2.3.2 ANALIZĖS REZULTATAI

Testiniams skaitmeniniams vaizdams (2.9 pav.) atliktas eksperimentas pagal 2.8 pav. pateiktą schemą. Panaudota visa vaizdo fragmento dydžių skalė $2^p\times2^p$ (p = 2, 3, 4, 5, 6, 7, 8, 9).

Įveskime pažymėjimus: τ_t – laiko intervalas (sekundėmis), reikalingas tiesioginiam DLGT spektro apskaičiavimui; τ_g – laiko intervalas (sekundėmis), reikalingas sudarytojo algoritmo (2.2.3 skyrelis) realizavimui. Gauti pirminiai ekperimentinio tyrimo rezultatai pateikti 2.1 ir 2.2 lentelėse.

2.1 lentelė

DLGT spektro apskaičiavimo (pasirinktam vaizdo fragmentui $2^p\times2^p$, p = 2, 3, 4, 5**) greičio testo rezultatai (sekundėmis)**

p / $N\times N$	2		3		4		5	
	τ_t	τ_g	τ_t	τ_g	τ_t	τ_g	τ_t	τ_g
128×128	7,80E-06	9,40E-06	2,65E-05	1,25E-05	1,03E-04	2,03E-05	4,27E-04	3,12E-05
256×256	6,30E-06	1,09E-05	2,65E-05	1,56E-05	1,05E-04	2,49E-05	4,26E-04	3,74E-05
512×512	6,20E-06	1,24E-05	2,66E-05	1,87E-05	1,03E-04	2,96E-05	4,29E-04	5,15E-05
1024×1024	6,20E-06	1,41E-05	2,65E-05	2,18E-05	1,05E-04	3,43E-05	4,05E-04	6,30E-05

2.2 lentelė

DLGT spektro apskaičiavimo (pasirinktam vaizdo fragmentui $2^p \times 2^p$, p = 6, 7, 8, 9) greičio testo rezultatai (sekundėmis)

p / $N \times N$	6		7		8		9	
	τ_t	τ_g	τ_t	τ_g	τ_t	τ_g	τ_t	τ_g
128×128	1,91E-03	5,93E-05	-	-	-	-	-	-
256×256	1,92E-03	7,80E-05	9,77E-03	1,72E-04	-	-	-	-
512×512	1,90E-03	1,10E-04	9,78E-03	2,02E-04	5,31E-02	4,99E-04	-	-
1024×1024	1,90E-03	1,09E-04	9,75E-03	2,34E-04	5,28E-02	5,93E-04	3,27E-01	1,69E-03

Akivaizdu (2.1, 2.2. lentelės), jog pasiūlytas sveikaskaitinio DLGT spektro pasirinktiems vaizdo fragmentams algoritmas dirba greičiau už standartinį (tiesioginis apskaičiavimas) algoritmą, kai $p \geq 3$. Kai p = 2, greitas algoritmas neatlieka savo funkcijos – spektrinių DLGT koeficientų išrinkimas užtrunka ilgiau nei tiesioginis apskaičiavimas. Pastebėsime, jog praktikoje (paprastai) imame $p \in \{5, 6, ..., n-1\}$, t.y. apdorojimui skirtas vaizdas dalijamas į vizualiai „apčiuopiamus“ fragmentus (blokus).

Toliau įvertinsime greito algoritmo duodamą greičio išlošį, kurį pažymėsime $\rho = \tau_t / \tau_g$ (2.3 ir 2.4 lentelės).

2.3 lentelė

DLGT spektro apskaičiavimo (pasirinktam vaizdo fragmentui $2^p \times 2^p$, p = 2, 3, 4, 5) naujuoju algoritmu efektyvumas

p / $N \times N$	2	3	4	5
	ρ			
128×128	0,83	2,12	5,07	13,70
256×256	0,58	1,70	4,20	11,39
512×512	0,50	1,42	3,48	8,33
1024×1024	0,44	1,22	3,05	6,43

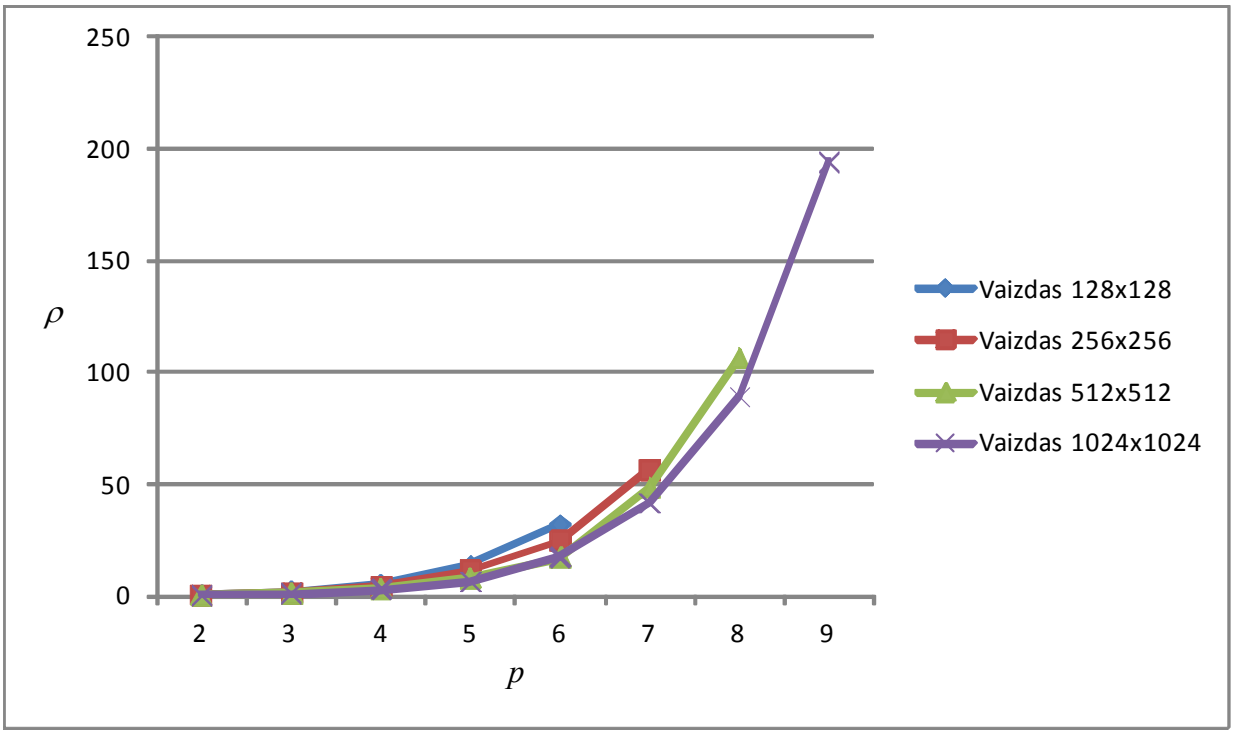
2.4 lentelė

DLGT spektro apskaičiavimo (pasirinktam vaizdo fragmentui $2^p \times 2^p$, $p = 6, 7, 8, 9$) naujuoju algoritmu efektyvumas

$N \times N$ \ p	6	7	8	9
	ρ			
128×128	32,28	-	-	-
256×256	24,60	56,78	-	-
512×512	17,30	48,42	106,32	-
1024×1024	17,47	41,67	88,97	193,95

Iš 2.3 ir 2.4 lentelių matome, jog didžiausios išlošis gaunamas, kai pasirinkto vaizdo fragmentas yra pakankamai didelis, t.y. kai $p \in \{n-1, n-2\}$. Tai galima paaiškinti tuo, kad greito algoritmo vykdymo metu, didelė dalis spektrinių koeficientų tiesiog išrenkama iš modifikuoto sveikaskaitinio pradinio vaizdo DLGT spektro, kai tuo tarpu tiesioginė DLGT turi atlikti masyvius ir ilgai trunkančius skaičiavimus.

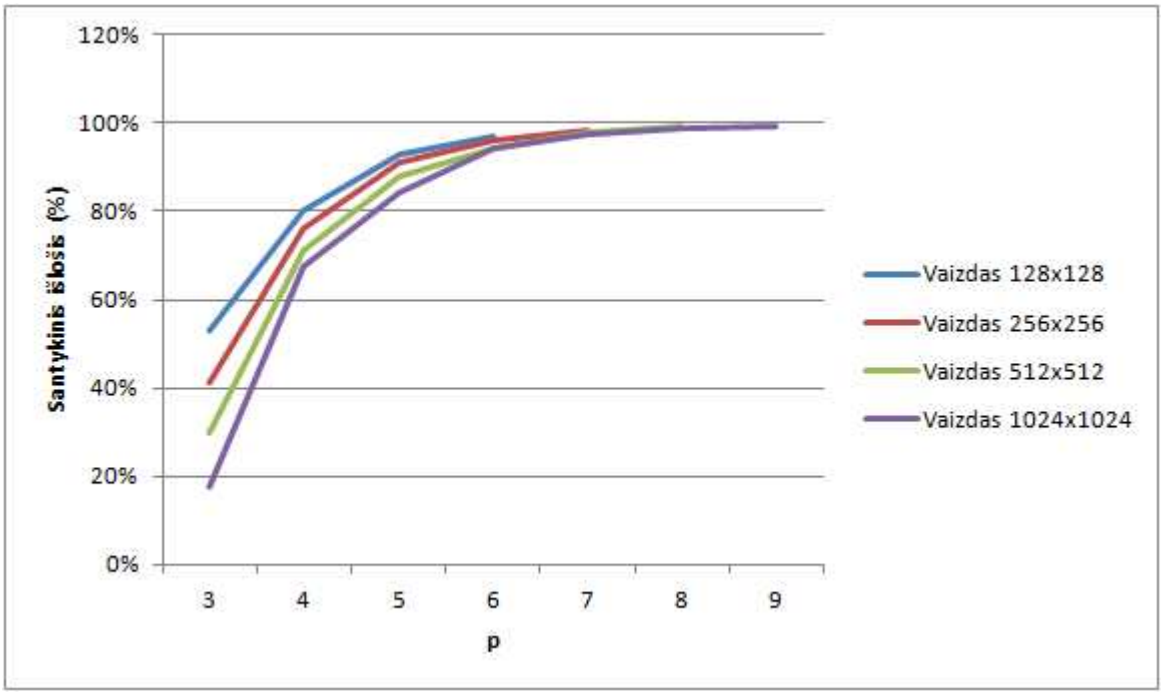
DLGT spektro pasirinktam vaizdo fragmentui $2^p \times 2^p$ apskaičiavimo (greitas algoritmas) greičio išlošio ρ priklausomybė nuo bloko dydžio p pavaizduota 2.10 pav.



2.10 pav. DLGT spektro apskaičiavimo pasirinktam vaizdo fragmentui efektyvumas

Dar kartą galima patvirtinti, kad didėjant pasirenkamo vaizdo fragmento matmenims, greičio išlošis didėja (2.10 pav.).

Grafiškai pavaizduosime santykinį greičio išlošį (procentais), sietina su greito DLGT spektro apskaičiavimo vaizdo fragmentams algoritmo taikymu. Greičio išlošis nusakomas išraiška $\left|\tau_t - \tau_g\right| / \tau_t \cdot 100(\%)$.



2.11 pav. DLGT spektro apskaičiavimo pasirinktam vaizdo fragmentui efektyvumas (%)

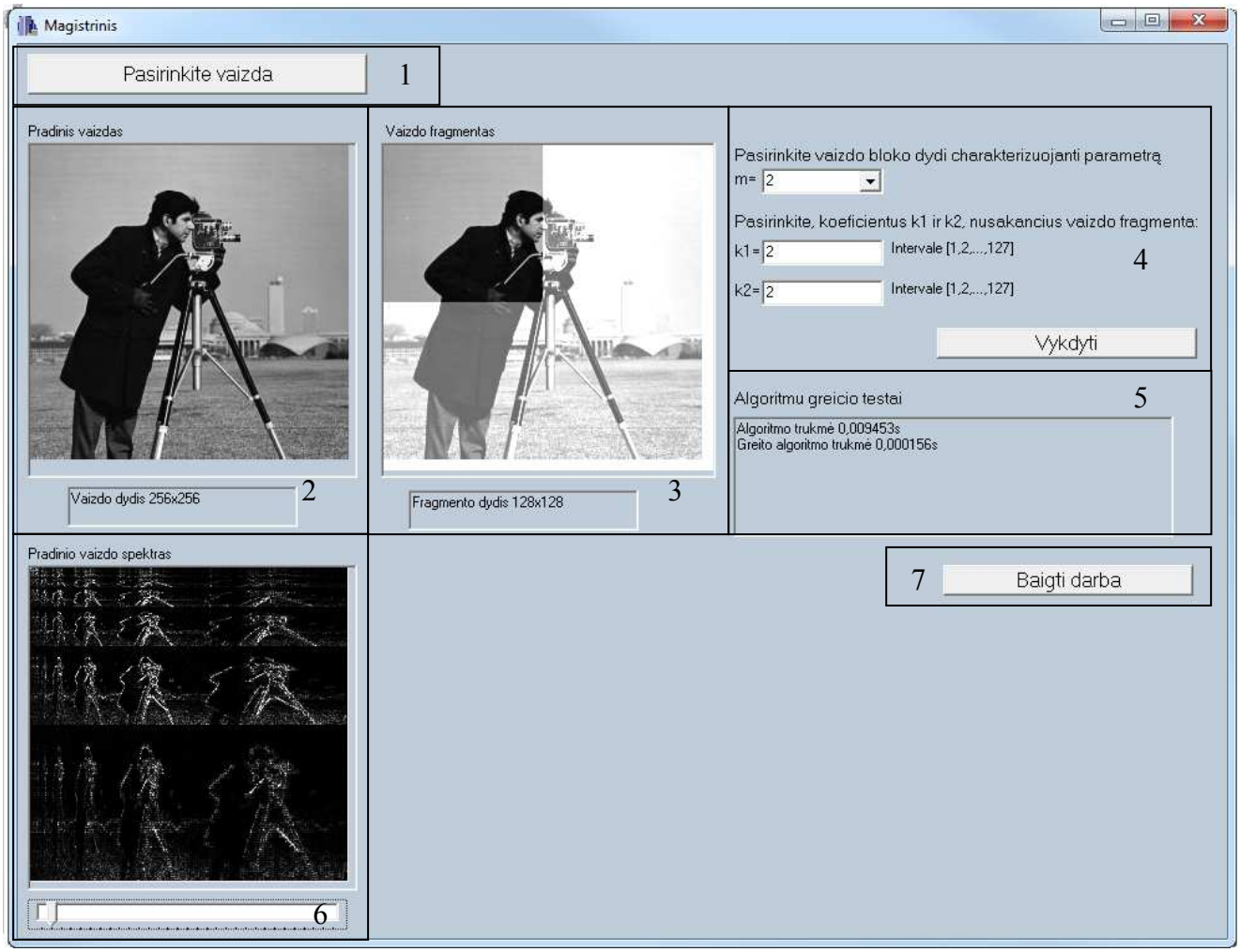
Kaip matome (2.11 pav.), santykinis greičio išlošis didėja, didėjant pasirenkamam vaizdo fragmentų (blokų) matmenims. Kita vertus, išlošį įtakoja ir pradinio vaizdo dydis – esant fiksuotam fragmento dydžiui, išlošis auga mažėjant apdorojamo vaizdo matmenims.

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Darbe realizuoti bazinės sveikaskaitinės DLGT ir modifikuotos jos versijos (su daline vaizdo blokų dekoreliacija aukštuose dažniuose) apskaičiavimo algoritmai. Taip pat sudarytas vaizdo fragmento išskyrimo ir jo sveikaskaitinio DLGT spektro apskaičiavimo procedūros: tiesiogiai ir panaudojus greitąjį algoritmą. Naudotas Borland C++ Builder 5 programinis paketas. Sukurta grafinė

vartotojo sąsaja, kurios pagalba, pasirinkus pradinius parametrus, galima išskirti vaizdo fragmentą bei gauti to fragmento DLGT spektro apskaičiavimo greičių įverčius.

Programa skirta dirbti su skaitmeniniais (pilkos šviesos intensyvumo skalės) dydžio $N = 2^n$ ($n \in \square$) vaizdais. Vaizdo analizei tinka *.bmp* skaitmeninių vaizdų formatas.



3.1 pav. Programos vykdymo langas

Programos vykdymo lango sudedamosios dalys (3.1 pav):

1. Dvimačių vaizdų bibliotekos atvėrimo mygtukas.
2. Pagrindinio vaizdo laukas.
3. Pažymėtas vaizdo fragmentas.
4. Fragmento valdymo laukas. Galima įrašyti dekoreliacijos lygį ir spektro koeficientus, pagal kuriuos išrenkamas vaizdo fragmentas.

5. Greičio testo rezultatų laukas.

6. Diskrečiojo sveikaskaitinio DLGT spektro laukas. Šiame lauke vaizduojamas spektras, gautas pritaikius DLGT pradiniam vaizdui, kurio ryškumą keičiame slinkties juostos pagalba.

7. Darbo pabaigos mygtukas.

Naudojimosi eiga:

1. Paleiskite programą paspaudus ***project1.exe***. Atsivers tuščias programos vykdymo langas (3.1 pav.).

2. Paspauskite mygtuką ***Pasirinkite vaizdą*** ir iš esančių duomenų failų išsirinkite norimą vaizdą. Vaizdas iškart pasirodys lange „***Pradinis vaizdas“***.

3. Pasirinkite norimą vaizdo dekoreliacijos lygį p, nuo kurio priklausys analizei tinkami vaizdo matmenys.

4. Pasirinkite vaizdo fragmentą įvedę pageidaujamus spektro koeficientus *k1* ir *k2*. Programa automatiškai (pagal dekoreliacijos lygį) nurodo galimus koeficientų intervalus.

5. Paspauskite mygtuką ***„Vykdyti“***. Programa iš karto apskaičiuoja ir atvaizduoja vaizdo fragmentą lange „***Vaizdo fragmentas“***, įgyvendina greičio testą, bei gautus rezultatus išveda lange „***Greičio testai***“. Taip pat galima atvaizduoti DLGT spektrą lange ***„Vaizdo spektras“*** (ryškumą keičiame slinkties juostos pagalba).

6. Paspaudus mygtuką ***Baigti darbą***, uždaromas programos vykdymo langas.

Tuo atveju, jei būtų pasirinktas vaizdas su neteisingais parametrais, išvedamas pranešimas apie klaidą.

IŠVADOS

1. Pastebėta, jog realizuojant tiesioginę sveikaskaitinę Le Gall bangelių transformaciją (DLGT) daugiamačiams vaizdams, būtina vienareikšmiai fiksuoti, kokia tvarka (pagal kokias erdvines vaizdo koordinates) bus taikoma vienmatė DLGT. Sukeitus šią tvarką, bendru atveju, gaunamas skirtingas to paties vaizdo sveikaskaitinis DLGT spektras. Tai ypač svarbu realizuojant greitas DLGT spektro apskaičiavimo pasirinktiems apdorojamo vaizdo fragmentams procedūras.

2. Išskirtinė diskrečiųjų bangelių transformacijų savybė (lyginant su klasikinėmis diskrečiosiomis transformacijomis, tokiomis kaip Furjė, Volšo, Kosinusinė ir kt.) – lokalizavimas erdvėje. Pilnas lokalizavimas erdvėje būdingas tiktai (paprasčiausiai) Harro bangelių transformacijai. Aukštesnių eilių diskrečiosios bangelių (Le Gall, Daubechies ir pan.) transformacijoms priskirtinas dalinis lokalizavimas erdvėje. Darbe realizuota ir ištirta modifikuota sveikaskaitinė Le Gall bangelių transformacija su pagerintomis lokalizavimo erdvėje savybėmis aukštesniuose dažniuose.

3. Algoritmizuota, realizuota ir ištirta sveikaskaitinio DLGT spektro apskaičiavimo atskiriems dvimačio vaizdo blokams procedūra (algoritmas). Esperimentiniai tyrimai su testiniais vaizdais parodė, jog sudarytasis algoritmas žymiai efektyvesnis už tiesiogines DLGT spektro apskaičiavimo tiems patiems vaizdo blokams procedūras. Laiko išlošis svyruoja nuo 6,43 kartų (vaizdo 512×512 fragmentas 64×64) iki 193,95 kartų (vaizdo 1024×1024 fragmentas 512×512).

4. Įvairiapusė pažintis su diskrečiosiomis bangelių transformacijomis, jų savybėmis, atlikti ekperimentiniai tyrimai leidžia teigti, jog modifikuota sveikaskaitinė DLGT kartu su fragmentiniu (blokiniu) DLGT apskaičiavimo algoritmu atveria naujas šios bangelių transformacijos praktinio taikymo sritis – požymių išskyrimas vaizde, lokaliai progresyvus vaizdų kodavimas, defektų aptikimas tekstūriniuose vaizduose ir pan.

LITERATŪRA

1. Daubechies I. Ten Lectures on Wavelets. SIAM, Philadelphia, 1992.
2. Sweldens W. The Lifting Scheme: A Construction of Second Generation Wavelets, Siam J. Math. Anal, 29(2), 511-546, 1997.
3. Valantinas J. On the use of Le Gall wavelets in implementing locally progressive digital signal coding idea, 6th International Symposium on Image and Signal Processing and Analysis; Austria, Salzburg, 2009.
4. Valantinas J., Kančelkis D. A new wavelets – based approach to progressive encoding of regions of interest in a digital signal, Informacinės technologijos ir valdymas. - ISSN 1392-124X.- 2009, nr. 3(38).
5. Valantinas J. Diskrečiosios transformacijos: mokomoji knyga, Kaunas: Technologija, 2008. – 85 p.
6. Valantinas J. Diskrečiosios transformacijos: mokymo priemonė, Kaunas: Technologija, 1993. – 76 p.
7. Valens C. A really friendly guide to wavelets, 1999. – 23 p.
8. Rao R. M., Bopardikar A. S. Wavelet Transforms: Introduction to Theory & Applications, Pearson Education, Asia, 2002.

1 PRIEDAS. PROGRAMOS KODAS

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>
#include <time.h>
#include <Windows.h>
#include <mmsystem.h>
#include <iostream>
using namespace std;
#include <iomanip>
const int Cn = 1024;                // Maksimalus eiluciu kiekis
const int Cm = 1024;                // Maksimalus stulpeliu kiekis
const int Cz = 1024;                // Maksimalus vektoriaus ilgis
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:                                                              //
IDE-managed Components
        TButton *Button4;
        TButton *Button2;
        TOpenPictureDialog *OpenPictureDialog1;
        TLabel *Label2;
        TButton *Button1;
        TMemo *Memo1;
        TMemo *Memo2;
        TLabel *Label4;
        TComboBox *ComboBox3;
        TLabel *Label1;
        TLabel *Label3;
        TEdit *Edit1;
        TEdit *Edit2;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TMemo *Memo3;
        TScrollBox *ScrollBox1;
        TImage *Image1;
        TScrollBox *ScrollBox2;
```

```
        TImage *Image2;
        TLabel *Label9;
        TLabel *Label10;
        TTrackBar *TrackBar1;
        TScrollBox *ScrollBox3;
        TImage *Image3;
        TLabel *Label11;
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall ComboBox3Change(TObject *Sender);
        void __fastcall TrackBar1Change(TObject *Sender);


private:                                                                    //
User declarations
   int
       E[Cn][Cm],              // Lyginiu (even) skaiciu masyvas
       O[Cn][Cm],              // Nelyginiu (odd) skaiciu masyvas
       D[Cn][Cm],              // Dk
       S[Cn][Cm],              // Sk
       P[Cn][Cm],              // Pradiniu duomenu masyvas
       T[Cn][Cm],              // Pilnas naujas masyvas
       F[Cn][Cm],              // Fragmentas
       X[Cn][Cm],
       A[Cn][Cm],
       Z[Cn][Cm],              // greito spektro traukimo algoritmo matrica
       W[Cn][Cm],              // tarpinė matrica Z[0] eilutei suformuoti
       Sv[Cz], Sh[Cz],G1[Cz], G2[Cz], T1[Cz],T2[Cz],Q[Cn][Cm],
       i1,i2,it,
       k1,k2,
       m,n,
       blocks,                 // Bloko dydi charakterizuojantis parametras
       alpha,
       gama,
       itstulp,iteil;
   float
       delta,
       deltadek;
   double
       j1,j2;
   void StulpelisDLGT(int stulp,int z, int X[Cm][Cn]);
   void EiluteDLGT (int eil,int z, int X[Cm][Cn]);
   void PerskaiciuotiADLGT(int X[Cm][Cn]);
   void Pikseliai();
   void StulpelisADLGT(int stulp,int z, int X[Cm][Cn]);
   void EiluteADLGT (int eil,int z, int X[Cm][Cn]);
   void PiestiPaveiksleli();
   void PiestiPaveiksleli2();
   void IrasytiStulpeliADLGT(int stulp,int z, int A[Cm][Cn]);
   void SkaiciuotiDLGTde(bool stulp,int it1, int it2, int X[Cm][Cn]);
   void IrasytiEiluteADLGT(int eil,int z, int A[Cm][Cn]);
   void SkaiciuotiADLGTde(bool stulp,bool kvadr, int it1, int it2, int X[Cm][Cn],
int A[Cm][Cn]);
   int nMax(int size);
   void UzpildytiMreiksmes();
   double Klaipsnis (int size);
   void Fragmentas();
   bool IsPixelIn(int x1,int y1, int x2, int y2, int i, int j);
   void GreitasSpektroRadimas();
```

```
   void ChangeLuminance(int A[Cn][Cm], int RowCount, int ColCount,
                              int multiplier, TImage *IM);
   public:
                                                                                //
User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

TForm1 klasės failas Unit.cpp

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

#include <vcl.h>
#pragma hdrstop
#include <math.h>
#include <iomanip>
#include <iostream>
#include <cmath>
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
using namespace std;

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

//----Mygtuko Vykdyti veiksmai-----------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
//;
  Image2->Picture->Assign(NULL);
  Image3->Picture->Assign(NULL);
  Memo2->Clear();

  Pikseliai();
  k1=StrToInt(Edit1->Text);
  k2=StrToInt(Edit2->Text);
  blocks=StrToInt(ComboBox3->Text);
  Fragmentas();
  PiestiPaveiksleli();
  Memo1->Lines->Clear();
  double TotalTime=0;
  clock_t start, end;

  int test=1;
  start = clock();
  for(int j = 0; j < test ; j++ ){
    SkaiciuotiDLGTde(true,itstulp,iteil,F);
    PerskaiciuotiADLGT(F);
    SkaiciuotiDLGTde(false,iteil,itstulp,F);
```

```
    PerskaiciuotiADLGT(F);
  }
  end = clock();
  TotalTime=0;
  TotalTime = ((end-start)/(CLK_TCK))/test;
  Memo1->Lines->Add("Algoritmo trukmė "+FloatToStr(TotalTime) + " s");

  itstulp=it;
  iteil=it;
  SkaiciuotiDLGTde(true,itstulp,iteil,P);
  PerskaiciuotiADLGT(P);
  SkaiciuotiDLGTde(false,iteil,itstulp,P);
  PerskaiciuotiADLGT(P);

  start = clock();
  for(int m = 0; m < test; m++ ){
     GreitasSpektroRadimas();
  }
  end = clock();
  TotalTime = ((end-start)/(CLK_TCK))/test;
  Memo1->Lines->Add("Greito algoritmo trukmė "+FloatToStr(TotalTime) + " s");
}
//----Mygtuko Pasirinkite vaizda veiksmai-----------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Image1->Picture->Assign(NULL);
  Image2->Picture->Assign(NULL);
  Image3->Picture->Assign(NULL);
  Memo1->Clear();
  Memo2->Clear();
  Memo3->Clear();
  OpenPictureDialog1->Filter = "Bitmap paveiksleliai(*.bmp)|*.bmp";
  if(OpenPictureDialog1->Execute()){
    Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
    if (Image1->Picture->Bitmap->PixelFormat != pf8bit)
      ShowMessage("Programa apdoroja tik 8 bitu vaizdus!");
    if (((Image1->Picture->Bitmap->Height) !=(Image1->Picture->Bitmap->Width)))
      ShowMessage("Vaizdo aukstis turi buti lygus plociui!");
    m = Image1->Picture->Bitmap->Height;
    n = Image1->Picture->Bitmap->Width;

    Memo3->Lines->Add("Vaizdo dydis "+FloatToStr(m)+"x"+FloatToStr(n));
    it=nMax(n);
    ComboBox3->Clear();
    UzpildytiMreiksmes();
  }
}
//----Mygtuko Baigti veiksmai-----------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
  Close();
}
//------------------------------- METODAI ----------------------------------
//----Uzpildo ComboBox galimomis M reiksmemis-------------------------------
void TForm1::UzpildytiMreiksmes()
{
  for (int i = 1; i < it; i=i+1) {
    ComboBox3->Items->Add(IntToStr(i));
  }
}
```

```
//---------------------------------------------------------------------------
//----Is duomenu matricos paima stulpeli DLGT skaiciavimui-----------------
void TForm1::StulpelisDLGT(int stulp,int z, int X[Cm][Cn])
{
  int no = 0;
  for(int j = 0; j < pow(2,z); j=j+2 ){
    O[0][no]=X[j][stulp];
    E[0][no]=X[j+1][stulp];
    no++;
  }
}
//----Bloko dydzio reiksmiu radimas-----------------------------------------
int TForm1::nMax(int size)
{
  int max;
  max = Log2(size);
  return max;
}
//----Is duomenu matricos pasiima eilute DLGT skaiciavimui------------------
void TForm1::EiluteDLGT(int eil,int z, int X[Cm][Cn])
{
  int no = 0;
  for(int j = 0; j < pow(2,z);j=j+2 ){
    O[0][no]=X[eil][j];
    E[0][no]=X[eil][j+1];
    no++;
  }
}
//---Skaitmenini vaizda pavercia i pikseliu masyva--------------------------
void TForm1::Pikseliai()
{
  assert(Image1->Picture->Bitmap->PixelFormat == pf8bit);
  for (int i = 0; i < m; i++) {
    unsigned char const* const p_row =
      static_cast<unsigned char*>(Image1->Picture->Bitmap->ScanLine[i]);
    for (int j = 0; j < n; j++) {
      P[i][j] = p_row[j];
      X[i][j] = P[i][j];
    }
  }
}
//----Masyvo perskaiciavimas------------------------------------------------
void TForm1::PerskaiciuotiADLGT(int X[Cm][Cn])
{
  for(int i = 0 ; i < pow(2,iteil); i++ )  {
    for(int j = 0; j < pow(2,itstulp); j++ )
      X[i][j]=A[i][j] ;
  }
}
//----Pikseliu masyvas paverciamas i paveiksleli----------------------------
void TForm1::PiestiPaveiksleli()
{
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++){
      if (T[i][j] < 0)
        T[i][j] = 0;
      if (T[i][j] > 255)
        T[i][j] = 255;
      Image2->Canvas->Pixels[j][i] = (TColor)RGB(T[i][j],T[i][j],T[i][j]);
    }
```

```
}
//----Is duomenu matricos pasiima stulpeli ADLGT skaiciavimui---------------
void TForm1::StulpelisADLGT(int stulp, int z, int X[Cm][Cn])
{
  int k = 1;
  int no = 1;
  S[z][0]=X[0][stulp];
  for(int i = z;  i>0; i-- ){
    for(int j = 0; j < k; j++ ){
     D[i][j]=X[no][stulp];
     no++;
    }
    k=k*2;
  }
}
//----Is duomenu matricos pasiima eulute ADLGT skaiciavimui-----------------
void TForm1::EiluteADLGT(int eil,int z, int X[Cm][Cn])
{
  int k = 1;
  int no = 1;
  S[z][0]=X[eil][0];
  for(int i = z;  i > 0; i-- ){
    for(int j = 0; j < k; j++ ){
     D[i][j]=X[eil][no];
    no++;
    }
    k=k*2;
  }
}
//----ADLGT rezultata iraso i stulpeli----------------------------------------
void TForm1::IrasytiStulpeliADLGT(int stulp,int z, int A[Cm][Cn])
{
   int no = 0;
   for(int j = 0; j < pow(2,iteil)/2; j++ ){
     A[no][stulp]=O[0][j];
     no++;
     A[no][stulp]=E[0][j];
     no++;
   }
}
//----ADLGT rezultata iraso i eilute------------------------------------------
void TForm1::IrasytiEiluteADLGT(int eil,int z, int A[Cm][Cn])
{
   int no = 0;
   for(int j = 0; j < pow(2,z)/2; j++ ){
     A[eil][no]=O[0][j];
     no++;
     A[eil][no]=E[0][j];
     no++;
   }
}
//----DLGT skaiciavimas su dekoreliacija--------------------------------------
void TForm1::SkaiciuotiDLGTde(bool stulp,int it1, int it2, int X[Cm][Cn])
{
  for (int z = 0; z < pow(2,it1)  ; z++ ) {
   if (stulp) {
     StulpelisDLGT(z,it2,X);
   }
   else  {
     EiluteDLGT(z,it2,X);
```

```
   }
   int p;
   alpha=0;
   gama=0;
   bool foundk = false;
   for (int i = 1; i <= it2 ; i++ ) {
     if (i > blocks){
       alpha = 1;
       gama = pow(2,it2-i);
     }
     else {
       alpha = pow(2,blocks-i);
       gama = pow(2,it2-blocks);
     }
     for (int k = 0; k <= pow(2,(it2-i))-1 ;k++){
       for (int t = 1; t <= gama ;t++)
         if (alpha*t-1 == k)
           foundk=true;
       if (foundk == true)
         D[i][k]=E[i-1][k]-O[i-1][k];
       else
         D[i][k]= E[i-1][k]-(O[i-1][k]+O[i-1][k+1])/2;
       p = pow(2,(it2-i))+k;
       if (stulp)
         A[p][z]=D[i][k];
       else
         A[z][p]=D[i][k];
       foundk=false;
       for (int t = 1; t <= gama ;t++)
         if (alpha*(t-1)==k)
           foundk=true;
       if (foundk == true)
         S[i][k]=O[i-1][k]+(D[i][k]+1)/2;
       else
         S[i][k]=O[i-1][k]+((D[i][k-1]+D[i][k]+2)/4);
       foundk=false;
     }
     float temp=0;
     int q;
     for (int z = 0; z<=(pow(2,(it2-i))-1);z++) {
     q=temp/1;
     temp+=0.5;
     if (z % 2 ==0)
       O[i][q]=S[i][z];
     else
       E[i][q]=S[i][z];
     }
   }
   if (stulp)
     A[0][z]=S[it2][0];
   else
     A[z][0]=S[it2][0];
  }
}
//----ADLGT skaiciavimas su dekoreliacija-------------------------------------
void TForm1::SkaiciuotiADLGTde(bool stulp,bool kvadr,int it1, int it2, int
X[Cm][Cn], int A[Cm][Cn])
{
  int c = 0;
  if (kvadr)
```

```
    c = pow(2,it1);
  else
    c = it1;

  for (int z  = 0; z < c   ; z++ ) {
   if (stulp) {
     StulpelisADLGT(z,it2,X);
   }
   else  {
     EiluteADLGT(z,it2,X);
   }
   int p;
   alpha,gama=0;
   bool foundk = false;
   for (int i = it2; i > 0; i-- ) {
     int index=0;
      if (i > blocks){
        alpha = 1;
        gama = pow(2,it2-i);
      }
      else {
        alpha = pow(2,blocks-i);
        gama = pow(2,it2-blocks);
      }
      for (int k = 0; k <= pow(2,(it2-i))-1 ;k++){
        for (int t = 1; t <= gama ;t++)
         if (alpha*(t-1)==k)
           foundk=true;
        if (foundk == true) {
          O[i-1][k]= S[i][k]-(D[i][k]+1)/2;
          S[i-1][index]=O[i-1][k];  index=index+2;
        }
        else {
          O[i-1][k]= S[i][k]-(D[i][k-1]+D[i][k]+2)/4;
          S[i-1][index]=O[i-1][k];  index=index+2;
        }
        foundk=false;
      }
      index=1;
      for (int k = 0; k <= pow(2,(it2-i))-1 ;k++){
        for (int t = 1; t <= gama ;t++)
          if (alpha*t-1==k)
             foundk=true;
        if (foundk == true) {
          E[i-1][k]=D[i][k]+O[i-1][k];
          S[i-1][index]=E[i-1][k];  index=index+2;
        }
        else    {
          E[i-1][k]= D[i][k]+(O[i-1][k]+O[i-1][k+1])/2;
          S[i-1][index]=E[i-1][k];  index=index+2;
        }
        foundk=false;
      }
   }
   if (stulp)
     IrasytiStulpeliADLGT(z,it2,A);
   else
     IrasytiEiluteADLGT(z,it2,A);
  }
}
```

```
//----Pikseliu masyvas paverciamas i paveiksleli----------------------------
void TForm1::PiestiPaveiksleli2()
{
  for (int i = 0; i < pow(2,iteil); i++)
    for (int j = 0; j < pow(2,itstulp); 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]);
  }
}
//----Suranda k1 ir k2 kitimo intervalus-------------------------------------
void __fastcall TForm1::ComboBox3Change(TObject *Sender)
{
  double z = Klaipsnis(StrToInt(ComboBox3->Text));
  Label5->Caption = "Intervale [1,2,...,"+FloatToStr(z)+"]";
  Label6->Caption = "Intervale [1,2,...,"+FloatToStr(z)+"]";
  Edit1->Clear();
  Edit2->Clear();
}
//----Randa k1 ir k2 paskutini kitimo intervalo nari-------------------------
double TForm1::Klaipsnis (int size)
{
  double z= pow(2,it-size+1)-1;
  return z;
}
//----Randa su spektro koeficientu susijusi vaizdo fragmenta----------------
void TForm1::Fragmentas()
{
  i1=it-(int)Log2(k1);
  j1=k1-pow(2,it-i1);
  i2=it-(int)Log2(k2);
  j2=k2-pow(2,it-i2);

  int firstPixelx = (j1*pow(2,i1));
  int firstPixely = (j2*pow(2,i2));
  int lastPixelx = ((j1+1)*pow(2,i1)- 1);
  int lastPixely = ((j2+1)*pow(2,i2)- 1);
  int k = 0;
  int l=0;
  bool frag=false;
  Memo2->Lines->Add("Fragmento dydis "+FloatToStr(lastPixelx-
firstPixelx+1)+"x"+FloatToStr(lastPixely-firstPixely+1));

  iteil = nMax(lastPixelx-firstPixelx+1);
  itstulp = nMax(lastPixely-firstPixely+1);

  for (int i = 0; i < m; i++){
    for (int j = 0; j < n; j++){
      if (IsPixelIn(firstPixelx,firstPixely,lastPixelx,lastPixely,i,j)){
        T[i][j] = P[i][j];
        F[k][l] = P[i][j];
        frag=true;
        l++;
      }
      else
        T[i][j] = P[i][j]+100;
    }
    if (frag) {
```

```
      k++;
      l=0;
    }
  }
}
//----Tikrina ar vaizdo pikselis priklauso fragmetui-------------------------
 bool TForm1::IsPixelIn(int x1,int y1, int x2, int y2, int i, int j)
{
  bool IsIn = false;
  if ((x1<=i) && (i<=x2))
    if ((y1<=j) && (j<=y2))
      IsIn=true;
  return IsIn;
}
//----Skaiciuoja fragmeto spektra greituoju algoritmu------------------------
void TForm1::GreitasSpektroRadimas()
{ // 1 etapas
  int t1ilgis=0;
  for( int i =0; i < i1; i++)
    t1ilgis+= pow(2,i);

  int t2ilgis=0;
  for( int i =0; i < i2; i++)
    t2ilgis+= pow(2,i);

  for (int i = 1; i <= it-i1; i++) {
    if (i==1)
      Sv[i] = (int) k1/2;
    else
      Sv[i] = (int) Sv[i-1]/2;
  }

  G1[0]=j1;
  for (int i = 1; i < it-i1; i++)
    G1[i] = (int) G1[i-1]/2;

  G2[0]=j2;
  for (int i = 1; i < it-i2; i++)
    G2[i] = (int) G2[i-1]/2;

  T2[1]=k2;
  int t = 1;
  for (int i = 1; i<=i2-1; i++)
    for (int j=pow(2,i)*k2; j<= pow(2,i)*(k2+1)-1;j++) {
      t++;
      T2[t]= j;
    }

  T1[1]=k1;
  t =1;
  for (int i = 1; i<=i1-1; i++)
    for (int j=pow(2,i)*k1; j<= pow(2,i)*(k1+1)-1;j++) {
      t++;
      T1[t]= j;
    }

  for (int i = 1; i <= it-i2 ; i++) {
    if (i==1)
      Sh[i] = (int) k2/2;
    else
```

```
    Sh[i] = (int) Sh[i-1]/2;
}
// 2 etapas
int q = 0;
int r = 0;
int y = 0;
int sum =0;
double z=0;
int e=0;
int s=0;
for ( t = 1; t <= it-i2; t++) {                                    // {0}
  r=Sh[t];
  z= (double) ((P[q][r]/2.) + (1./2.));
  sum+=((1-pow(-1,G2[t-1]))/2)*P[q][r]-(int)z;
}

W[e][s]= P[q][s]+ sum;
for ( t = 1; t <= t2ilgis; t++) {
  y=T2[t];
  s++;
  W[e][s]= P[q][y];
}

sum=0;
for (int i = it-i1; i >=1 ; i--) {                                 // Sv
  q=Sv[i];
  for (int t = 1; t <= it-i2; t++) {
    r=Sh[t];
    z= (double) ((P[q][r]/2.) + (1./2.));
    sum+=(1-pow(-1,G2[t-1]))/2*P[q][r]-(int)z;
  }
  e++;
  s=0;
  W[e][s]= P[q][0]+ sum;

 sum=0;
 for (int x = 1; x <= t2ilgis; x++) {
   s++;
   y=T2[x];
   W[e][s]= P[q][y];
 }
}

e=0;
s=0;
sum=0;
for (int i = 1; i <= t1ilgis; i++) {                               // T1
  q=T1[i];
  for ( t = 1; t <= it-i2; t++) {
    r=Sh[t];
    z= (double) ((P[q][r]/2.) + (1./2.));
    sum+=(1-pow(-1,G2[t-1]))/2*P[q][r]-(int)z;
  }
  e++;
  s=0;
  Z[e][s]= P[q][0]+ sum;
  sum=0;
  for (int x = 1; x <= t2ilgis; x++) {
    y=T2[x];
    s++;
```

```
      Z[e][s]= P[q][y];
    }
  }
  // 3 etapas
  SkaiciuotiADLGTde(false,false,it-i1+1,Log2(t2ilgis+1),W,W);
  // 4 etapas
  for (int i = 0; i <= t2ilgis; i++) {
    sum=0;
    for ( t = 1; t <= it-i1; t++) {
      z= (double) ((W[it-i1-t+1][i]/2.) + (1./2.));
      sum+=(1-pow(-1,G1[t-1]))/2*W[it-i1-t+1][i]-(int)z;
    }
  Q[0][i]= W[0][i]+ sum;
  }
  // 5 etapas
  SkaiciuotiDLGTde(false,0,Log2(1+t2ilgis),Q);
  for ( t = 0; t <= t2ilgis; t++) {
    Z[0][t]=A[0][t];
  }
}
//----Keicia vaizdo spektro ryskuma-----------------------------------------
void TForm1::ChangeLuminance(int A[Cn][Cm], int RowCount, int ColCount,
                             int multiplier, TImage *IM)
{
  for (int i = 0; i < RowCount; i++)
    for (int j = 0; j < ColCount; j++)
    {
      IM->Canvas->Pixels[j][i] = (TColor)RGB(255,255,255);
      if (multiplier != 1)
      {
        double absolute = fabs(A[i][j]) * multiplier;
        if (absolute > 255)
          absolute = 255;

        if ((i+1 < RowCount) && (j+1 < ColCount))
        IM->Canvas->Pixels[j][i] = (TColor)RGB(absolute,absolute,absolute);
      }
      else
      {
        IM->Canvas->Pixels[j][i] = (TColor)RGB(A[i][j],A[i][j],A[i][j]);
      }
    }
   }
//----Vaizdo spektro ryskumo keitimas---------------------------------------
void __fastcall TForm1::TrackBar1Change(TObject *Sender)
{
  int multiplier = pow(2,TrackBar1->Position);
  ChangeLuminance(P, m, n, multiplier, Image3);
  Image3->Picture->SaveToFile("Spektras.bmp");
}
//---------------------------------------------------------------------------
```