KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Vytautas Dusevičius

EFEKTYVAUS VAIZDŲ SUSPAUDIMO ALGORITMO SUDARYMAS IR TYRIMAS

Magistro darbas

Darbo vadovas

doc. dr. E. Kazanavičius

KAUNAS, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

TVIRTINU

Katedros vedėjas

doc. dr. E. Kazanavičius

2004 05

EFEKTYVAUS VAIZDŲ SUSPAUDIMO ALGORITMO SUDARYMAS IR TYRIMAS

Informatikos mokslų magistro baigiamasis darbas

Lietuvių kalbos konsultantė

dr. Jurgita Mikelionienė

2004 05

Recenzentas

doc. dr. J. Valantinas

2004 05

Vadovas

doc. dr. E. Kazanavičius

2004 05

Atliko

IFM – 8/1 gr. stud.

Vytautas Dusevičius

2004 05 19

KAUNAS, 2004

Dusevičius V. Analysis and Development of Efficient Image Compression Algorithm: Master's Work in Informatics/ supervisor assoc. prof. dr. E. Kazanavičius; Computer Department, Faculty of Informatics, Kaunas University of Technology. – Kaunas, 2004. -47 p.

SUMMARY

Uncompressed multimedia data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications even more sustained the need for more efficient ways to encode such data.

There are two types of image compression schemes – lossless and lossy algorithms. In lossless compression schemes, the reconstructed image, after compression, is numerically identical to the original image. However lossless compression can only achieve a modest amount of compression. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because the compression scheme completely discards redundant information. However, lossy schemes are capable of achieving much higher compression.

The aim of this research is to create an efficient lossy image compression algorithm, using heuristic data clusterization methods; perform experiments of the new algorithm, measure its performance, analyze advantages and disadvantages of the proposed method, propose possible improvements and compare it with other popular algorithms.

In this paper is presented new algorithm for image compression, which uses data base of popular image fragments. Proposed algorithm is best suited for natural greyscale and color images. Performed experiments show that high compression ratios can be achieved using proposed method.

TURINYS

1. ĮVADAS

Nesuspausti skaitmeniniai vaizdo, garso ir foto duomenys užima didelę dalį informacijos saugojimo įrenginių vietos bei sunaudoja didžiąją informacijos srautų dalį internete. Nors nuolatos sparčiai didėja pastoviosios atminties įrenginių apimtys, procesorių skaičiavimo greičiai ir duomenų perdavimo sparta internetu, reikalavimai saugojamų duomenų dydžiui ir perdavimo greičiui viršija esamų technologijų galimybes. Šiuo metu vis populiarėjančios internetinės programos bei svetainės, naudojančios didelius kiekius skaitmeninės informacijos, dar labiau sustiprina efektyvių algoritmų ir metodų poreikį, skirtų tokio pobūdžio duomenims suspausti ir tuo pačiu išspręsti didelių duomenų kiekio saugojimo, apdorojimo ir platinimo internetu problemas.

Galima išskirti dvi pagrindines duomenų suspaudimo algoritmų klases – suspaudimo algoritmai be informacijos praradimo (angl. *lossless*) ir algoritmai su duomenų praradimu (angl. *lossy*) [Saha, 2000]. Naudojant algoritmus be informacijos praradimo, rekonstruotas vaizdas yra skaitiškai identiškas originaliam vaizdui. Tačiau, naudojant šio tipo algoritmus pasiekiamas tik nežymus duomenų suspaudimo lygis. Todėl suspaudimo algoritmai be duomenų praradimo naudojami tose srityse, kur duomenų dydžiai yra aktuali problema, tačiau bet koks duomenų praradimas yra nepageidaujamas – kaip pavyzdžiai gali būti medicinos ir astronomijos mokslai, kur atkurtų vaizdų kokybė yra ypač svarbi. Kai naudojami algoritmai su duomenų praradimu, atkurtas vaizdas netiksliai atkartoja originalų vaizdą. Informacijos praradimas įvyksta todėl, kad suspaudimo metu yra visiškai atmetama perteklinė (mažiausiai svarbi) informacija. Šis būdas leidžia daug didesnius duomenų suspaudimo lygius. Net ir esant skaitiniams duomenų praradimams, dažnai rekonstruotas ir pradinis vaizdas yra vizualiai identiški (angl. *visually lossless*).

Šio darbo tikslas yra **sukurti efektyvų dvimačių vaizdų suspaudimo algoritmą su informacijos praradimu, naudojant euristinius duomenų klasterizavimo metodus, eksperimentiškai patikrinti sudarytąjį algoritmą ir išmatuoti gautus rezultatus, ištirti sudarytojo algoritmo privalumus bei trūkumus, įvardinti galimus patobulinimus bei palyginti algoritmą su kitais šiuo metu populiariais algoritmais**. Algoritmas turėtų suspausti vaizdą bent 10 kartų, nesant ryškiems kokybės pokyčiams. Taip pat vaizdo apdorojimo laikas neturėtų labai skirtis nuo kitų populiariausių vaizdų suspaudimo algoritmų, naudojančių duomenų suspaudimo schemas su informacijos praradimu.

Šiame darbe pristatomas naujas vaizdų suspaudimo algoritmas, naudojantis populiariausių vaizdo fragmentų duomenų bazę, kuriai sudaryti naudojami euristiniai duomenų klasterizavimo (klasifikavimo, grupavimo) metodai. Aprašytasis algoritmas tinka

pilkiems (angl. *greyscale*) bei spalvotiems vaizdams. Esant didelei bazinių vaizdų duomenų bazei, pasiekiami ypač dideli vaizdų suspaudimo lygiai. Vaizdo suspaudimo koeficientas yra nepriklausomas nuo vaizdo pobūdžio, o tik nuo duomenų suspaudimui naudojamų parametrų.

Darbe yra apžvelgiami keli populiariausi duomenų suspaudimo su informacijos praradimu algoritmai, atlikta algoritmų privalumų bei trūkumų analizė. Taip pat išanalizuoti algoritmai, naudojami naujojo suspaudimo algoritmo tarpiniuose etapuose – tai duomenų klasterizavimo algoritmai, entropijos kodavimo algoritmai, duomenų apdorojimo bei transformacijos algoritmai.

Antrajame magistratūros semestre, studijuojant Danijos Aalborgo universitete, daugiamačių signalų apdorojimo (angl. *Multidimensional Signal Processing*) modulyje buvo pristatyta bazinė siūlomo algoritmo idėja, parengtas pranešimas bei suprogramuotas programinės įrangos karkasas. Taip pat buvo atliktas darbas „Java ir C kalbų tinkamumas signalų apdorojimo uždaviniams spręsti“. Šio darbo rezultatai buvo panaudoti renkantis programavimo priemones, taip pat ta tema parašytas straipsnis, įtrauktas į tarptautinės konferencijos, skirtos įterptinėms sistemoms, pranešimų medžiagą (žr. 1 priedą).

Naudojant darbe aprašytąjį algoritmą, buvo sukurta programinė įranga dvimačių vaizdų suspaudimui ir rekonstravimui atlikti, įvykdyti matavimų eksperimentai, gauti rezultatai palyginti su kitais populiariais algoritmais. Išanalizavus visus gautus rezultatus, darbe pateikiami naujojo algoritmo privalumai bei trūkumai, vystymo perspektyvos ir galimi tolesni moksliniai tyrimai bei eksperimentai.

2. BAZINĖS VAIZDŲ SUSPAUDIMO SĄVOKOS

2.1. Dvimačio vaizdo sąvoka

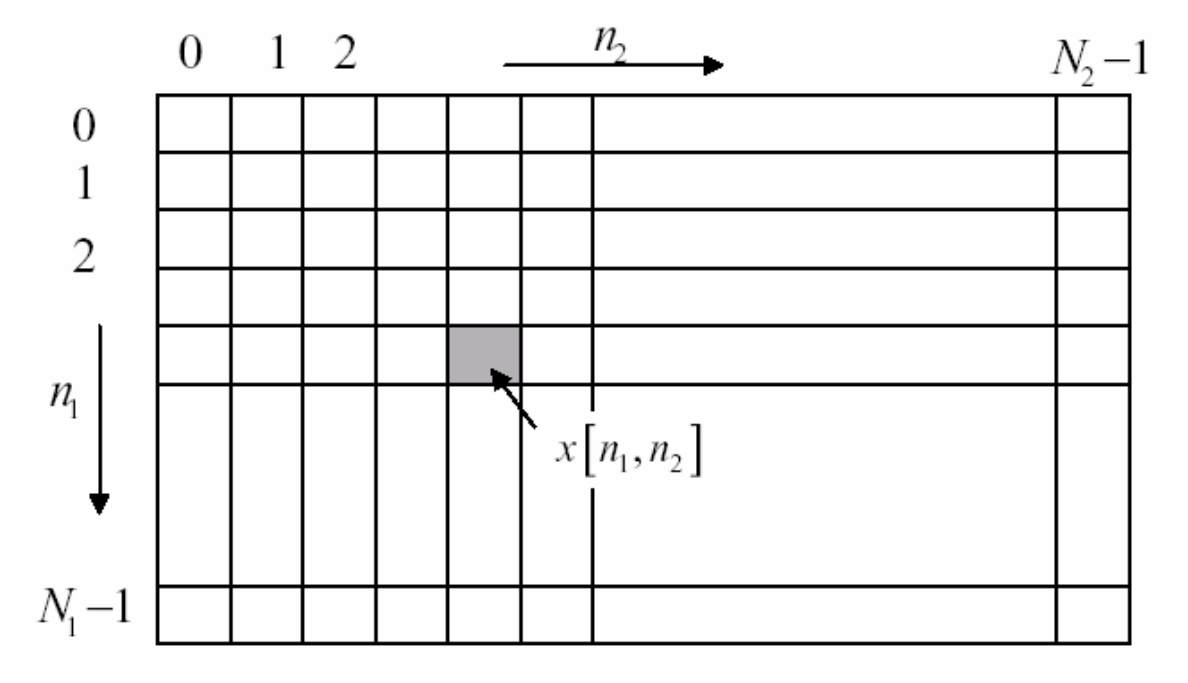
Šiame darbe dažnai vartojama dvimačio skaitmeninio vaizdo sąvoka. Dvimatis skaitmeninis vaizdas – tai dvimatė seka reikšmių

$$x[n_1, n_2], \quad 0 \le n_1 < N_1,\ 0 \le n_2 < N_2; \tag{1}$$

čia $x[n_1, n_2]$ - konkretaus vaizdo pikselio (taško) reikšmė;

N_1 - vaizdo aukštis pikseliais;

N_2 - vaizdo plotis pikseliais.



1 pav. Dvimatis skaitmeninis vaizdas [Girod, 2004: 2]

Vienam vaizdo taškui saugoti yra skiriama B bitų ir jis gali įgauti tokias reikšmes:

$$x[n_1, n_2] \in \{0,1,..,2^B - 1\}; \tag{2}$$

čia $x[n_1, n_2]$ - konkretaus vaizdo pikselio reikšmė;

B - bitų kiekis, skiriamas vienam vaizdo pikseliui saugoti.

Dažnai yra patogu naudoti ir kitą to paties užrašo interpretaciją:

$$x[n_1, n_2] = \langle 2^B x'[n_1, n_2] \rangle; \tag{3}$$

čia $x[n_1, n_2]$ - konkretaus vaizdo pikselio reikšmė;

B - bitų kiekis, skiriamas vienam vaizdo pikseliui saugoti;

$x'[n_1, n_2]$ - pikselio realios reikšmės intervale [0..1);

<> - skliaustai reiškia apvalinimą iki artimiausio sveikojo skaičiaus.

Spalvotiems vaizdams saugoti kiekvienam taškui naudojamos trys vaizdo komponentės:

$$x_R[n_1, n_2] \quad x_G[n_1, n_2] \quad x_B[n_1, n_2]; \tag{4}$$

čia $x_R[n_1, n_2]$ - konkretaus vaizdo pikselio raudonos spalvos intensyvumo reikšmė;

$x_G[n_1, n_2]$ - konkretaus vaizdo pikselio žalios spalvos intensyvumo reikšmė;

$x_B[n_1, n_2]$ - konkretaus vaizdo pikselio mėlynos spalvos intensyvumo reikšmė.

Tokia spalvoto vaizdo saugojimo schema vadinama RGB (nuo angliškų santrumpų ***R**ed **G**reen **B**lue*). Ši schema yra populiariausia, tačiau toli gražu ne vienintelė [Gonzalez, 1993].

Galima išskirti kelias pagrindines dvimačių vaizdų klases:

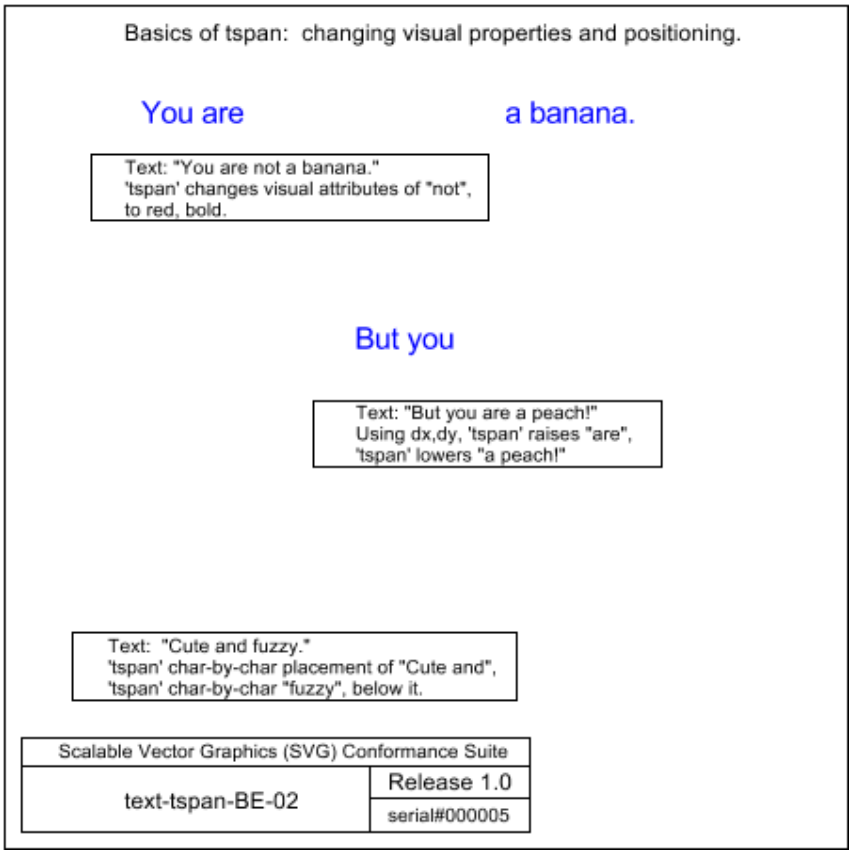
- realaus pasaulio vaizdai;
- vaizdai su tekstine informacija;
- pieštinė grafika.

Realaus pasaulio vaizdai – tai gamtos, žmonių ar kitų mus supančių objektų nuotraukos ar filmuoto vaizdo kadrai. Tai didžiausia ir populiariausia vaizdų klasė, tad ir didžioji dalis vaizdų suspaudimo algoritmų yra skirti būtent šio tipo vaizdams spausti.

2 pav. Realaus pasaulio vaizdo pavyzdys

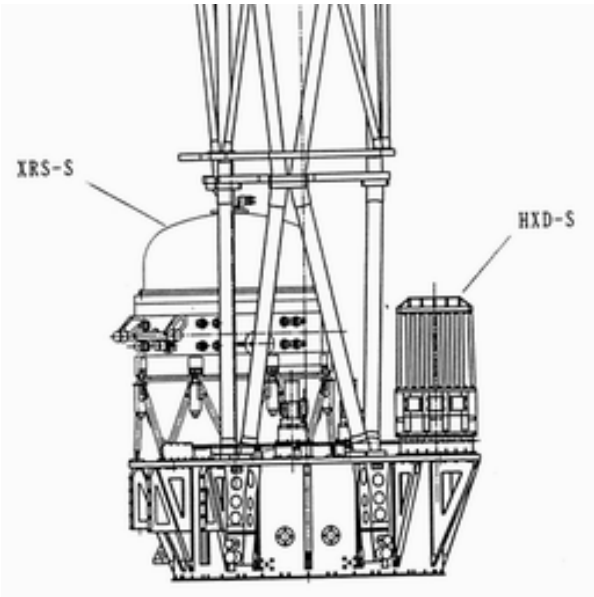
Vaizdai su tekstine informacija saugo spausdintinį, rašytą ranka ar kitokio pobūdžio tekstą, tačiau informacija saugoma ne raidžių ar žodžių pavidale, o aprašant vaizdo taškus.

Spaudžiant tokius vaizdus svarbu išlaikyti gerą atkurto vaizdo kokybę, tokiems vaizdams taip pat dažnai taikomos teksto atpažinimo vaizde programos [Matteson, 1995].



3 pav. Vaizdo su tekstine informacija pavyzdys

Pieštinė grafika – tai piešiniai, brėžiniai, kompiuterinė grafika ir pan. Dažniausiai tai juodai balti piešiniai arba nedaug spalvų naudojantys vaizdai.



4 pav. Pieštinės grafikos vaizdo pavyzdys

2.2. Duomenų suspaudimo metodų poreikis

Vis labiau didėjanti kompiuterinės technikos ir internetinių technologijų sparta leidžia vis labiau naudoti vaizdo, garso ir foto informaciją. Nepaisant dabartinių kompiuterių pastoviosios atminties dydžių bei interneto ryšio spartos, daugeliu atveju neįmanoma saugoti ar tuo labiau internetu platinti nesuspaustą foto ar vaizdo medžiagą.

Filmuota medžiaga saugoma kaip vienas po kito einantys kadrai (nuotraukos). Panagrinėkime pavyzdį, kuris gerai parodo duomenų suspaudimo metodų poreikį. Tarkime, kad vartotojas naudojasi interneto ryšiu, kurio maksimali sparta yra 56 Kbit/s (kilobitai per sekundę). Klausimas – kiek laiko užtruktų filmo, išsaugoto nesuspaustu formatu ir trunkančio 90 minučių, siuntimas? Tarkime, kad vaizdo dydis yra 240×320 taškų. Tada vienas vaizdo kadras užimtų apie 1.8 Mbit (megabitų). Kadangi per sekundę parodoma 30 kadrų, tai 1 sekundės filmuota medžiaga užima apie 52.7 Mbit. Tada 90 minučių filmas užimtų 278.1 Gbit (gigabitų). Esant pastoviam siuntimo greičiui, lygiam 56 Kbit/s, vartotojas filmą siųstų maždaug 60 dienų! Reiktų atkreipti dėmesį į tai, kad garsinė informacija nebuvo įskaičiuota, be to, retai siuntimas vyksta pastoviu maksimaliu greičiu. Tad realiai siuntimo ilgis padidėtų dar bent du kartus.

1 lentelė. Skirtingų duomenų dydžiai ir siuntimo laikai [Saha, 2000]

Duomenys	**Dydis / trukmė**	**Bitai / pikseliui**	**Nesuspaustų duomenų dydis**	**Siuntimo laikas, naudojant 28.8 Kbit/s modemą**
Puslapis teksto	A4 formatas	–	apie 50 Kbit	1-2 sek.
Telefoninės kokybės pokalbis	10 sekundžių	8	640 Kbit	22 sek.
Pilkos skalės nuotrauka	512×512 taškų	8	2 Mbit	1 min 13 sek.
Spalvota nuotrauka	512×512 taškų	24	6 Mbit	3 min. 39 sek.
Medicininė nuotrauka	2048×1680 taškų	12	41 Mbit	23 min 54 sek.
Filmuota medžiaga	2048×1680 taškų, 1 minutės trukmės	24	221 Mbit	5 dienos 8 val.

1 lentelė aiškiai parodo, koks didelis yra saugojimo vietos, interneto spartos ir laiko poreikis garso bei vaizdo duomenims. Esant šiuolaikiniams techniniams pasiekimams, duomenų suspaudimas prieš saugojimą ar siuntimą yra vienintelė įmanoma išeitis.

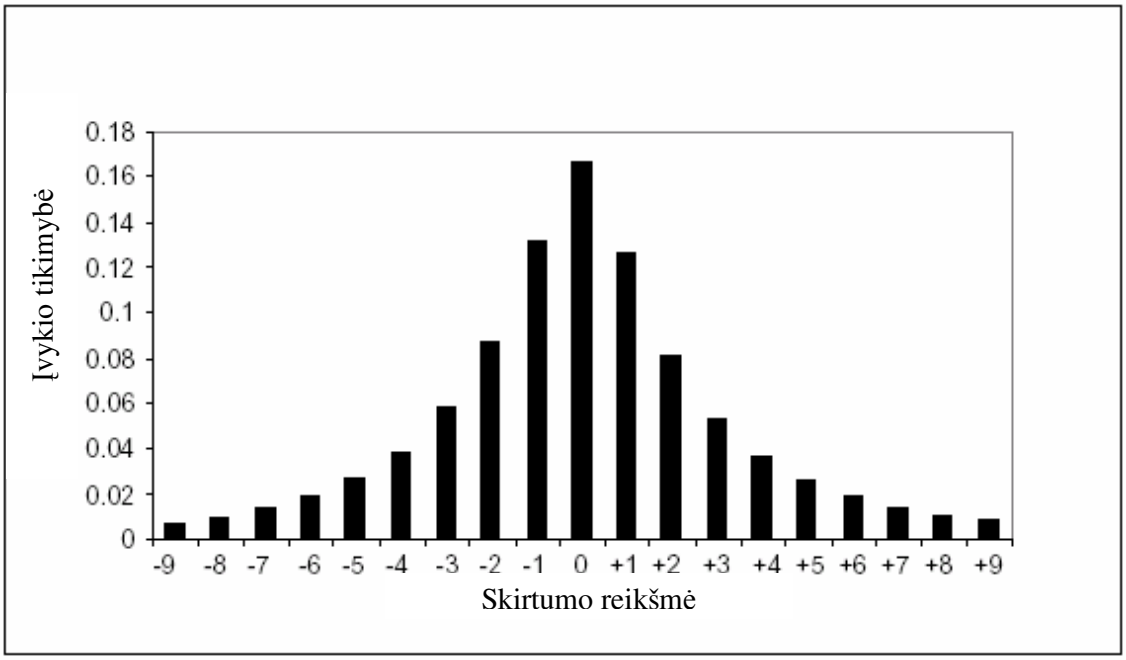
2.3. Duomenų suspaudimo baziniai principai

Bendra didžiosios dalies vaizdų savybė yra ta, kad gretimi vaizdo taškai koreliuoja tarpusavyje, vadinasi yra perteklinės informacijos. Tokiu atveju, pagrindinė užduotis yra surasti mažiausiai besikoreliuojantį pradinio vaizdo atvaizdą. Dveji fundamentalūs suspaudimo principai yra pertekliškumo (angl. *redundancy*) ir nereikšmingumo (angl. *irrelevancy*) mažinimas [Saha, 2000].

Mažinant pertekliškumą, siekiama pašalinti besidubliuojančias duomenų signalo (vaizdo) reikšmes. Nereikšmingumo mažinimo metu iš duomenų signalo yra pašalinamos tos reikšmės, kurios nebus pastebėtos signalo gavėjo. Kalbant apie vaizdus, tą įmanoma padaryti įvertinus žmogaus regos sistemos (angl. *human visual system*) savybes. Galima išskirti tokius pertekliškumo tipus:

- statistinis pertekliškumas
    - erdvinis pertekliškumas (angl. *spatial redundancy*)
    - kodavimo pertekliškumas
- vizualinis pertekliškumas (angl. *psychovisual redundancy*)
    - spalvinis pertekliškumas
    - dažninis pertekliškumas

Erdvinis pertekliškumas atsiranda todėl, kad pikselių reikšmės nėra tarpusavyje nepriklausomos. Paprastai yra labai didelė gretimų taškų koreliacija. Panagrinėjus vaizdų gretimų taškų spalvos skirtumus gaunami tokie rezultatai:



5 pav. Gretimų vaizdo taškų reikšmių skirtumų histograma [Gonzalez, 1993]

Kodavimo pertekliškumas atsiranda koduojant pikselių reikšmes į galutinį suspaustą dvejetainį kodą.

2 lentelė. Alfabeto dvejetainio kodavimo pavyzdys

Simbolis	Pasitaikymo tikimybė	I kodas	II kodas
a_1	0.1	000	0000
a_2	0.2	001	01
a_3	0.5	010	1
a_4	0.05	011	0001
a_5	0.15	100	001

2 lentelėje pavaizduotas alfabetas iš 5 simbolių su kiekvieno simbolio pasirodymo tikimybe. Taip pat pateikti du unikalūs kodai, kurių pagalba galima koduoti duotojo alfabeto simbolius. Naudojant I-ąjį kodą gaunamas toks vidutinis kodo dydis:

$$L_{avg,1} = 3 \text{ bitai / simboliui};\tag{5}$$

Naudojant II-ąjį kodą vidutinį kodo dydį galima rasti taip:

$$L_{avg,2} = 4 \cdot 0.1 + 2 \cdot 0.2 + 0.5 + 4 \cdot 0.05 + 3 \cdot 0.15 = 1.95 \text{ bitų / simboliui};\tag{6}$$

Matome, jog antrasis kodas yra taip pat unikalus, tačiau trumpesnis už pirmąjį.

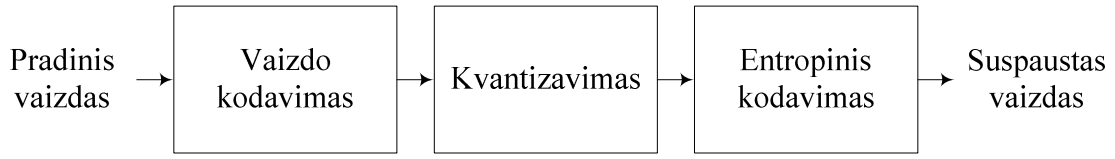
Spalvinį (intensyvumo) pertekliškumą nusako Vėberio dėsnis (angl. *Weber's law*): dirgiklis turi didėti tam tikra proporcija, kad sukeltų tiesinį atsako padidėjimą [Mullen, 1985]. Šiuo atveju, koduojant yra išnaudojama žmogaus regos savybė, kad žmogus į šviesos pokytį reaguoja ne tiesiškai, bet logaritmiškai.

Dažninis pertekliškumas atsiranda taip pat dėl žmogaus regos savybių. Žmogaus akis veikia kaip žemo dažnio filtras, todėl spaudžiant vaizdą galima panaikinti aukšto dažnio harmonikas ir tai liks nepastebima žmogaus akimi. Tai bene labiausiai išnaudojama savybė vaizdų suspaudimo algoritmuose. Kaip pavyzdys gali būti hiperbolinio filtravimo algoritmai, kurie kodavimo metu visiškai pašalina aukščiausius dažnius ir taip pasiekiamas nemažas suspaudimo lygis, nesant dideliems kokybės pokyčiams.

2.4. Tipinė vaizdo suspaudimo schema

Tipinė vaizdo suspaudimo su informacijos praradimu schema pavaizduota 6 paveiksle. Schema sudaryta iš trijų tarpusavyje susijusių komponentų – vaizdo kodavimo komponento,

kvantizacijos komponento ir entropinio kodavimo komponento. Suspaudimo efektas pasiekiamas tokiu būdu - pirmiausia yra pradinis vaizdas tiesiškai transformuojamas, kad sumažinti duomenų tarpusavio koreliaciją; gauti transformacijos koeficientai yra kvantizuojami ir, galiausiai, pritaikomas entropinis kodavimas.



6 pav. Tipinė vaizdo suspaudimo schema

Vaizdo kodavimo (arba tiesinio transformavimo) algoritmų sukurta daug ir įvairių, kiekvienas turintis savų privalumų bei trūkumų. Galima paminėti tik kelis labiausiai paplitusius transformacijų algoritmus: DFT (angl. *Discrete Fourier Transform*) [Oppenheim, 1999: 541], DCT (angl. *Discrete Cosine Transform*) [Ifeachor, 1993: 79], DWT (angl. *Discrete Wavelet Transform*) [Vatterli, 1995].

Kvantizavimo metu yra sumažinamas bitų skaičius, reikalingas transformuotiems koeficientams išsaugoti, mažinant saugojamų reikšmių tikslumą. Kadangi tai yra atvaizdavimas daug-su-vienu (angl. *many-to-one mapping*), tai šio proceso metu yra prarandama informacija. Tačiau šio procesu metu yra pasiekiama didžioji vaizdo kompresijos dalis. Kvantizacija gali būti atliekama kiekvienai reikšmei atskirai (skaliarinė kvantizacija) arba atliekama tam tikrai koeficientų grupei (vektorinė kvantizacija).

Entropinio kodavimo metu toliau spaudžiamos kvantizuotos reikšmės, naudojant suspaudimo algoritmus be informacijos praradimo, siekiant dar labiau padidinti vaizdo suspaudimo lygį. Labiausiai paplitę entropinio kodavimo algoritmai yra: Hufmano kodavimas, aritmetinis kodavimas, bitinis kodavimas (angl. *run-length coding*), žodyno kodavimas [Nelson, 1995].

2.5. Pagrindinės vaizdo suspaudimo charakteristikos

Yra trys pagrindinės charakteristikos, kurių pagalba galima įvertinti vaizdų suspaudimo algoritmus:

- suspaudimo lygis (angl. *compression ratio*);
- suspaudimo greitis;
- suspausto vaizdo kokybė.

Šių charakteristikų svarba labiausiai priklauso nuo taikomosios srities ir kitų reikalavimų (pvz. naudojamos kompiuterinės technikos galimybės).

Suspaudimo lygį (koeficientą) galima rasti taip:

$$\alpha = \frac{I}{I'}; \tag{7}$$

čia α - vaizdo suspaudimo koeficientas;

I - pradinio vaizdo dydis (baitais);

I' - suspausto vaizdo dydis (baitais).

Suspaudimo koeficientas parodo, kiek kartų suspaustas vaizdas yra mažesnis už pradinį vaizdą. Paprastai šis dydis labai smarkiai įtakoja vaizdo kokybę – kuo aukštesnis šis koeficientas, tuo prastesnė suspausto vaizdo kokybė. Šį faktą labai svarbu įvertinti, spaudžiant ar lyginant kelis suspaustus vaizdus. Be to, naudojant tam tikrus algoritmus, suspaudimo lygis gali būti labai priklausomas nuo paties vaizdo. Vaizdų suspaudimo srityje šis aspektas vadinamas duomenų priklausomumu (angl. *data dependency*). Naudojant algoritmą, kuris pasižymi dideliu duomenų priklausomumu, spaudžiant didelio detalumo vaizdą (pvz., minios vaizdas šventės metu), gali būti pasiekiamas labai nedidelis suspaudimo lygis. Tačiau vaizdui, kuris nepasižymi detalumu (pvz., jūros ir dangaus nuotrauka), tas pats algoritmas gali pasiekti labai didelį suspaudimo lygį.

Suspaudimo greitis – tai laikas, kuris sunaudojamas vaizdo suspaudimui bei atstatymui atlikti. Suspaudimo greitis labiausiai priklauso:

- nuo suspaudimo algoritmo sudėtingumo;
- nuo efektyvaus algoritmo realizavimo programinėmis priemonėmis;
- nuo naudojamos aparatūrinės įrangos greičio ir architektūros.

Nors tam tikrais atvejais suspaudimo greitis yra kritinė charakteristika (pvz., realaus laiko sistemose), daugeliu atveju labiausiai vertinamos yra suspaudimo lygio ir vaizdo kokybės charakteristikos.

Vaizdo kokybė nusako tikslumą, kuriuo yra atkuriamas vaizdas iš suspaustų duomenų, lyginant jį su pradiniu (nesuspaustu) vaizdu. Suspaudimo algoritmai, naudojantys vaizdo suspaudimo be informacijos praradimo schemas, išlaiko visą originalią informaciją. Tad ši charakteristika taikoma tik algoritmams, kurie naudoja suspaudimo schemas su informacijos praradimu. Vaizdo kokybei įvertinti dažniausiai naudojama vidutinė kvadratinė paklaida, tačiau tai nėra vienintelis įmanomas kokybės (paklaidos) įvertis. Vidutinė kvadratinė paklaida (angl. *mean square error*) randama pagal formulę:

$$MSE = \frac{1}{M \cdot N} \sum_{y=1}^{M} \sum_{x=1}^{N} \left[I(x, y) - I'(x, y) \right]^2 ; \quad (8)$$

čia
- MSE - vidutinė kvadratinė paklaida;
- M,N - vaizdo dimensijos (aukštis ir plotis) pikseliais;
- $I(x,y)$ - originalaus vaizdo konkretaus taško reikšmė;
- $I'(x,y)$ - suspausto vaizdo konkretaus taško reikšmė.

Kuo labiau suspausto vaizdo taškų reikšmės skiriasi nuo originalaus vaizdo taškų reikšmių, tuo didesnė yra vidutinė kvadratinė paklaida. Vadinasi, jei keli algoritmai spaudžia vienodą vaizdą, tai geriausią vaizdo kokybę atkuria algoritmas, kurio MSE įvertis yra mažiausias.

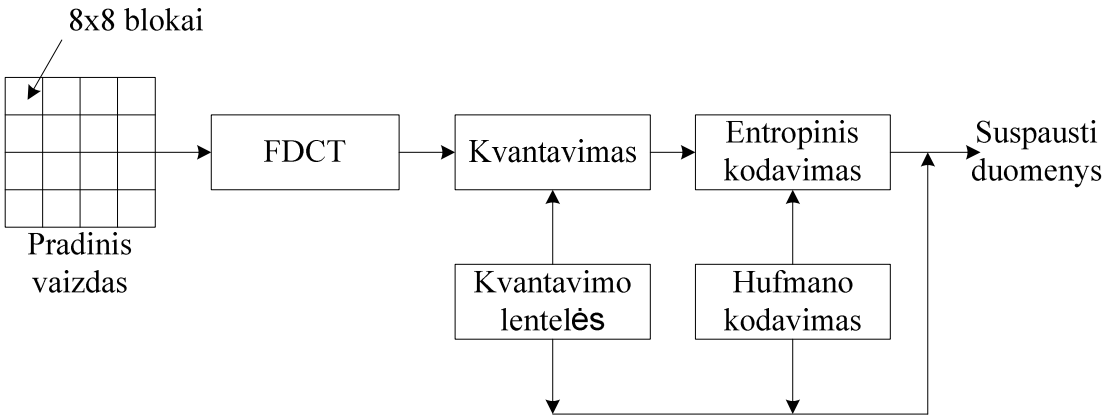
3. POPULIARIŲ VAIZDO SUSPAUDIMO ALGORITMŲ APŽVALGA

3.1. JPEG algoritmas

JPEG (angl. *Joint Photographic Expert Group*) komitetas buvo įkurtas 1986 metais CCITT ir ISO organizacijų iniciatyva, siekiant sukurti pasaulinius vaizdų suspaudimo standartus [Pennebaker, 1993]. Techniškai darbas buvo baigtas 1991 metais. 2001 metais buvo sukurtas naujas JPEG-2000 standartas, tačiau jis dar nėra plačiai paplitęs.

JPEG standartas aprašo keturis pagrindinius suspaudimo tipus: nuoseklus, progresyvinis ir hierarchinis (kodavimui su informacijos praradimu), bei vienas režimas be informacijos praradimo. Labiausiai paplitęs yra nuoseklus suspaudimo algoritmas.

Bazinė JPEG algoritmo veikimo schema parodyta 7 paveiksle.



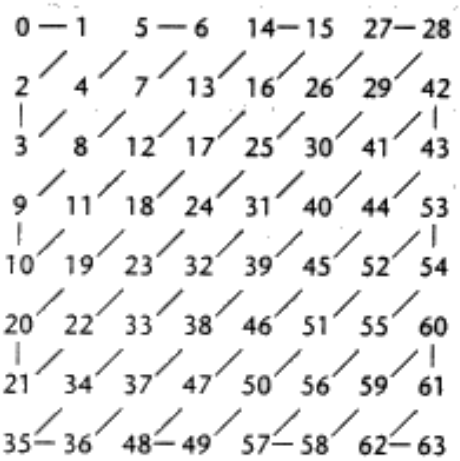
7 pav. Bazinė JPEG algoritmo kodavimo schema [Saha, 2000]

JPEG vaizdo suspaudimas atliekamas 4 žingsniais:

1. Pirmiausia vaizdas yra suskaidomas į 8×8 pikselių vaizdo blokus. Algoritmas veikia ne spalvinėje, o skaisties / chromatiškumo erdvėje (angl. *luminance / chrominance*), todėl pradžioj atliekama spalvų konversija į YCbCr formatą [Bracamonte, 2000]. Tai leidžia taikyti skirtingą suspaudimą abiems šiems faktoriams. Kadangi žmonių rega daug jautresnė skaisčiai nei chromatiškumui, tai suspaustame faile išlaikoma daugiau skaisties informacijos nei chromatiškumo informacijos.
2. Toliau yra pritaikoma FDCT (angl. *Forward Discrete Cosine Transform*) transformacija kiekvienam 8×8 vaizdo blokui [Ifeachor, 1993: 79]. Šio etapo

metu nėra pasiekiamas suspaudimas, o tiesiog duomenų sritis pakeičiama į spektrinę sritį.

3. Atlikus duomenų transformaciją į spektrinę sritį, tikrasis suspaudimo algoritmas pradeda darbą. Priklausomai nuo vartotojo pasirinktos norimos vaizdo kokybės, yra parenkamos dvi kvantizavimo lentelės – po atskirą lentelę skaisties ir chromatiškumo reikšmėms. Šios lentelės naudojamos DCT koeficientų kvantizavimui, kurie yra padalinami iš atitinkamos lentelės reikšmės ir vėliau suapvalinami iki artimiausio sveikojo skaičiaus. To pasekoje dideli DCT koeficientai praranda dalį tikslumo, o maži koeficientai yra prilyginami nuliui, kas vėliau labiausiai ir įtakoja kokybės praradimą.
4. Galiausiai kvantizacijos rezultatai nuskaitomi zigzagine tvarka, kaip parodyta 8 paveiksle, ir gautai reikšmių sekai yra naudojamas Hufmano kodavimas. Tai yra antrasis (be informacijos praradimo) suspaudimo algoritmas, kuris pritaikomas realizuojant JPEG vaizdų suspaudimo standartą.



Pav. 1. Zigzaginė DCT koeficientų nuskaitymo tvarka

Pagrindiniai JPEG algoritmo privalumai:

- tai populiariausias iš šiuo metu naudojamų vaizdų suspaudimo algoritmų;
- pasiekiamas vidutinis 15 kartų suspaudimo lygis be didelių kokybės nuostolių;
- tam tikriems vaizdams bei kai nereikalinga labai aukšta atkurto vaizdo kokybė, pasiekiamas apie 20 kartų suspaudimo lygis.

JPEG algoritmo trūkumai:

- netinka juodai baltiems (bitiniams) vaizdams ;
- prie didelio suspaudimo ryškėja blokelių struktūra;
- pasireiškia Gibso efektas („kontūrų aidas“);

- mažai tinkamas animacinei grafikai, brėžiniams, tekstinei grafikai bei vaizdams su 256 ir mažiau spalvų.

9 paveiksle parodyta originali nuotrauka ir 13 kartų suspausta nuotrauka, panaudojant JPEG suspaudimo algoritmą.

9 pav. Originali ir 13 kartų suspausta nuotrauka (dešinėje)

Kol suspaudimo koeficientas nesiekia maždaug 15, yra išlaikoma labai gera kokybė, žmogaus akimi atkurtas vaizdas sunkiai atskiriamas nuo originalaus vaizdo. Tačiau padidinus suspaudimo koeficientą, aiškiai pasireiškia JPEG algoritmo blogosios savybės:

10 pav. Originali ir 40 kartų suspausta nuotrauka (dešinėje)

3.2. BTC algoritmas

BTC (angl. *Block Truncation Coding*) algoritmas yra vienas iš vaizdų suspaudimo algoritmų, naudojančių vaizdo suspaudimo su informacijos praradimu schemą. Algoritmo idėja – suskaidyti vaizdą į nedidelius blokus ir kiekvienam blokui sumažinti jo pilkumo skalės lygius. Kvantavimas atliekamas atsižvelgiant į vaizdo bloko reikšmių statistiką ir pasiskirstymus. Nauji pilkumo lygiai parenkami taip, kad būtų minimizuotas pasirinktas klaidos kriterijus, o tada visos bloko reikšmės pakeičiamos dviem naujais lygiais [Franti, 1994].

Baziniame algoritmo variante vaizdas skaidomas į 4×4 pikselių vaizdo blokus. Bloko pikselių reikšmės koduojamos dviem lygiais – *a* ir *b*. Kvantavimo lygiai *a* ir *b* parenkami taip, kad būtų išsaugomi pirmieji momentai υ_1 ir υ_2:

$$\upsilon_1 = \frac{1}{m}\sum_{i=1}^{m} x_i \ , \ \upsilon_2 = \frac{1}{m}\sum_{i=1}^{m} x_i^2 \ ; \tag{9}$$

čia m - vaizdo bloko (fragmento) pikselių kiekis;

x_i - vaizdo bloko *i*-tasis pikselis.

Norint suskaičiuoti kvantavimo lygių *a* ir *b* reikšmes, sprendžiama lygčių sistema:

$$\begin{cases} \frac{1}{m}\left((m-q)\cdot a + q\cdot b\right) = \upsilon_1 \\ \frac{1}{m}\left((m-q)\cdot a^2 + q\cdot b^2\right) = \upsilon_2 \end{cases} ; \tag{10}$$

čia m - vaizdo bloko (fragmento) pikselių kiekis;

q - pikselių, kuriems bus priskirta reikšmė *b*, skaičius;

a ir *b* - nauji bloko kvantavimo lygiai;

υ_1 ir υ_2 - bloko pirmos ir antros eilės momentai.

Reikšmė q randama tokiu būdu: *i*-tajam pikseliui priskiriama reikšmė *a*, jeigu $x_i < x_{slenkstinė}$, kitu atveju – priskiriama reikšmė *b*. Baziniame algoritmo variante $x_{slenkstinė} = \upsilon_1$.

Išsprendus lygčių sistemą, randami kvantavimo lygiai:

$$a = \upsilon_1 - \delta\sqrt{\frac{q}{m-q}} \ , \ b = \upsilon_1 + \delta\sqrt{\frac{m-q}{q}} \ ; \tag{11}$$

čia *a* ir *b* - nauji bloko kvantavimo lygiai;

υ_1 - bloko pirmos eilės momentas;

q - pikselių, kuriems bus priskirta reikšmė *b*, skaičius;

m - vaizdo bloko (fragmento) pikselių kiekis;

$$\delta = \sqrt{v_2 - v_1^2} \ .$$

Suspaustam blokui užkoduoti reikia 3 dydžių – *a*, *b* ir *B*, kur *B* – bitinė plokštuma (1 – pikseliui priskirta *a* reikšmė, 2 – pikseliui priskirta *b* reikšmė). Suspaudimo koeficientą galima rasti pagal formulę:

$$\alpha = \frac{m \cdot p}{2 \cdot p + m}; \tag{12}$$

čia α - vaizdo suspaudimo koeficientas;

m - vaizdo bloko pikselių kiekis;

p - bitų kiekis, skiriamas vienam pikseliui koduoti (paprastai $p = 8$).

BTC algoritmo privalumai:

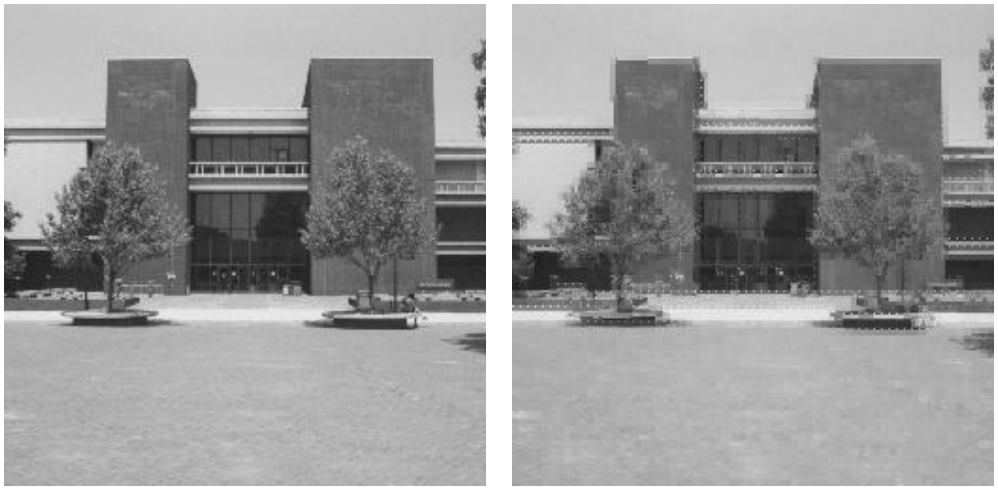
- labai greitai ir mažai reiklus skaičiavimams algoritmas;
- lengvai suprantamas ir realizuojamas;
- esant nedideliam suspaudimo koeficientui išlaikoma labai gera atkurto vaizdo kokybė;

BTC algoritmo trūkumai:

- pasiekiamas nedidelis vaizdo suspaudimo koeficientas, paprastai neviršijantis 4-8 kartų.
- prie didesnių suspaudimo lygių pasireiškia blokinė struktūra ir laiptinis spalvų efektas.

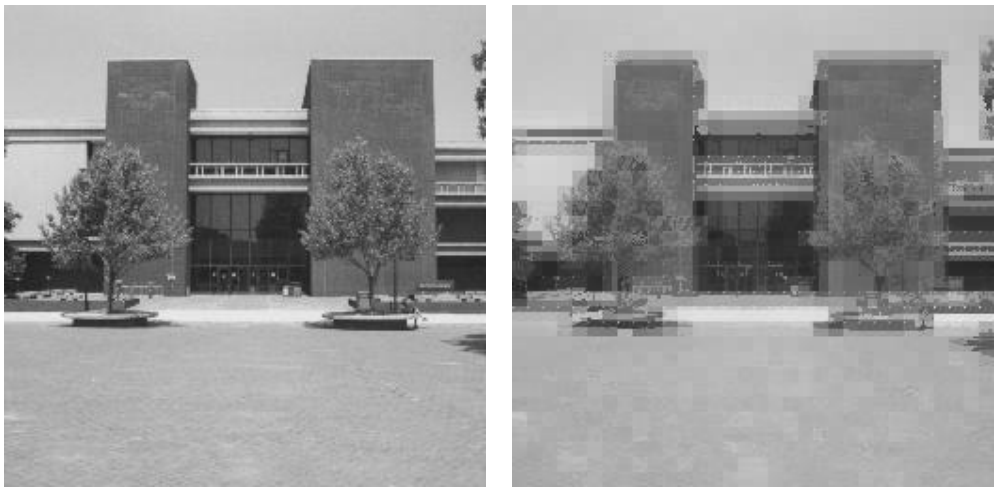
Tobulinant BTC algoritmą, yra sukurta daug jo modifikacijų [Rabiee, 1995], tačiau vis tiek nėra pasiekiami suspaudimo lygiai, artimi DCT pagrindu sukurtiems suspaudimo algoritmams. Taip pat yra šio algoritmo modifikacijos spalvotiems vaizdams spausti [Qiu, 2003].

Pavyzdiniai vaizdai, esant skirtingiems suspaudimo lygiams, pavaizduoti 11 ir 12 paveiksluose.



11 pav. Originali ir 4 kartus suspausta nuotrauka (dešinėje)

Kaip minėta, prie nedidelių suspaudimo koeficientų, išlaikoma gera vaizdo kokybė, tačiau praktinis BTC algoritmo pritaikymas esant didesniems suspaudimo koeficientams yra labai ribotas.



12 pav. Originali ir 8 kartus suspausta nuotrauka (dešinėje)

3.3. Kiti algoritmai

Šiame darbe nėra iškeltas tikslas aprašyti daugelį vaizdų suspaudimo algoritmų, tai kiti populiariausi algoritmai šiame poskyryje bus tik trumpai paminėti.

Vaizdų suspaudimas, naudojant dispersinius kriterijus. Dažniausiai šio pobūdžio algoritmai taikomi tam tikrai vaizdų klasei. Efektyvaus kodavimo uždavinys formuluojamas taip: panaudojant tam tikrus kriterijus atrinkti transformacijos (spektrinių) koeficientų poaibį, kuris yra mažesnis už visų vaizdo reikšmių aibę. Likusieji transformacijos koeficientai yra atmetami. Ši procedūra, atkuriant pradinį vaizdą, neturi iššaukti kiek žymesnės paklaidos. Kitaip tariant, ieškoma diskrečioji transformacija, kuri, esant fiksuotam vaizdo suspaudimo

koeficientui, leidžia minimizuoti vidutinę kvadratinę paklaidą. Įrodyta, kad optimali diskrečioji transformacija yra tokia, kurios matricos baziniai vektoriai sutampa su tikriniais vaizdų klasės kovariacinės matricos vektoriais. Pagrindinė problema yra ta, kad surasti optimalią transformacijų matricą yra labai sunku arba tam tikrais atvejais net neįmanoma. Todėl dažniausiai naudojamos kitos transformacijos – DCT (angl. *Discrete Cosine Transform*), Volšo ir Adamaro DT (diskrečioji transformacija), diskrečioji Haaro transformacija ir kitos.

Baigtinių automatų teorijos taikymas, apdorojant skaitmeninius vaizdus [Valentinas, 1997]. Šio algoritmo bazinis principas – skaidyti vaizdą hierarchiniais skirtingos detalizacijos lygiais. Kiekvienas skirtingas vaizdo fragmentas atitinka būseną determinuotame baigtiniame automate. Jei tam tikra būsena neturi atitikmens jau esamų būsenų aibėje, ją atitinkantis blokas yra skaidomas toliau, kol išskaidytoms dalims randamos atitinkamos (ar panašios) būsenos arba yra pasiekiamas mažiausias detalizacijos lygis. Vėliau gautasis automatas aprašomas atitinkama reguliariąja išraiška, kurios kodavimui reikia mažiau vietos nei pradiniam vaizdui koduoti.

Bangelių (angl. *wavelet*) metodas [Graps, 1995]. Bangelinės funkcijos – tai baigtiniame intervale apibrėžtos funkcijos, kurių vidurkis yra lygus nuliui. Bangelinės transformacijos bazinė idėja – bet kokią funkciją *f(t)* galima atvaizduoti kitų bazinių funkcijų sumine išraiška. Tos bazinės funkcijos sudaromos iš vieningo prototipo, naudojant masteliavimą ir postūmius. Bangelių metodas paremtas signalo (vaizdo) dažnių juostos išskaidymu ir toliau kodavimu kiekvienos išskaidytos dažninės juostos atskirai, naudojant parametrus, geriausiai atitinkančius konkrečios dažninės juostos statistinius duomenis.

Tai buvo tik keli patys populiariausi metodai vaizdų suspaudimui atlikti. Savaime suprantama, kad metodų yra žymiai daugiau, tame tarpe ir algoritmai be informacijos praradimo, kurie šiame darbe visai nėra aptarti. Tuo labiau, didžiajai daliai „klasikinių“ algoritmų yra sukurta įvairiausių modifikacijų bei patobulinimų, tačiau visa tai jau nebetilptų į šio darbo rėmus.

4. DVIMAČIŲ VAIZDŲ SUSPAUDIMO ALGORITMAS, PANAUDOJANT VAIZDO FRAGMENTŲ KLASTERIZACIJĄ

4.1. Prielaidos algoritmui atsirasti

Jei visus realaus pasaulio vaizdus būtų galima surinkti į vieną visumą ir juos visus suskaidyti į nedidelius blokus, tai tų blokų būtų nesuskaičiuojama galybė. Matematiškai, žinoma, galima rasti kiek iš viso yra įmanomų variantų sudaryti pasirinkto dydžio vaizdo blokus: tarkime, jei bloko dydis yra 8×8 pikselių, o vienam pikseliui saugoti skiriama 8 bitai, t. y. 256 spalvos (jei vaizdas saugomas pilkumo skalėje), tai skirtingų tokio dydžio blokų bus $256^{8\cdot8} = 256^{64}$, o tai yra skaičius, turintis daugiau nei 150 skaitmenų! Tačiau tikėtina, jo realiuose vaizduose ne visos šios kombinacijos pasitaiko, be to, vienos iš jų yra labai dažnos, o kitos realiuose vaizduose randamos tik labai retai. Tuo labiau, iš visos aibės blokų, pasitaikančių realiuose vaizduose, žmogui vizualiai besiskiriančių blokų yra gerokai mažiau. Tad algoritmas ir paremtas žmogaus regos netobulumu ir pagrindinis algoritmo darbo tikslas yra kažkokiais būdais atrinkti aibę blokų, kurie gerai atspindėtų likusias kombinacijas, o ypač tas, kurios dažniausiai pasitaiko realaus pasaulio vaizduose. Pagrindiniai algoritmo darbo žingsniai yra aprašomi 4.2 poskyryje.

4.2. Pagrindiniai algoritmo žingsniai

Algoritmą sudaro šie pagrindiniai žingsniai:

- bazinių vaizdų atrinkimas;
- bazinių blokų atrinkimas;
- atrinktų bazinių blokų klasterizavimas;
- vaizdo suspaudimas;
- pradinio vaizdo atstatymas.

Tolimesniuose šio skyriaus poskyriuose visi anksčiau išvardintieji žingsniai yra aprašomi išsamiau.

4.3. Bazinių vaizdų atrinkimas

Tai žingsnis, kuriame atrenkami vaizdai, naudojami baziniams blokams sudaryti. Svarbu atrinkti tokius vaizdus, kurie kuo geriau atspindi visą vaizdų aibę, nes iš jų sudaryti blokai bus klasterizuojami, o klasterizacijos rezultatai labiausiai daro įtaką suspausto vaizdo kokybei. Kuo daugiau bus bazinių vaizdų, tuo labiau tikėtina, kad jie gerai atspindės realaus pasaulio vaizdus. Tačiau tuo pačiu didės ir bazinių blokų kiekis, o tai gali įtakoti laiko

sąnaudų padidėjimą klasterizuojant bazinius blokus. Problema šiame žingsnyje yra ta, jog bazinius vaizdus galima atrinkti tik eksperimentiškai, nes neįmanoma nusakyti, kiek ir kokie vaizdai gerai atspindi visą vaizdų aibę. Tačiau eksperimentų metu buvo pastebėta, kad šiame žingsnyje reiktų nenaudoti didelių tekstūrinių vaizdų, kurie sudaryti iš pasikartojančių nedidelių vaizdo fragmentų, kadangi toks vaizdas „duoda“ daug vienodų blokų, kurie gali smarkiai iškreipti klasterizacijos rezultatus. Eksperimentų metu, buvo naudojama nuo kelių iki 100 bazinių vaizdų, priklausomai nuo tolimesnių algoritmų darbo greičių, bei charakteristikų, kurias norima pamatuoti. Visi vaizdai buvo pilkumo skalėje, dydžiai suvienodinti iki 256×256 pikselių. Visi vaizdai užsaugoti duomenų formate, nenaudojančiame suspaudimo algoritmų – TIFF (angl. *Tagged Image File Format*). Nė vienas iš bazinių vaizdų nebuvo naudojamas matuojant algoritmo suspaudimo lygį bei atkurto vaizdo kokybę, kadangi tokie vaizdai galėtų duoti geresnius rezultatus.

4.4. Bazinių blokų atrinkimas

Baziniai blokai – tai bazinių vaizdų fragmentai, kurie vėliau yra klasterizuojami ir rezultatai naudojami vaizdams spausti. Šiame žingsnyje iš visų bazinių vaizdų yra išskiriami baziniai blokai. Čia yra galimos kelios strategijos:

- galima suskirstyti vaizdą į nepersidengiančius fragmentus, kurių dydis lygus bloko dydžiui, taip, kad visi vaizdo taškai priklausytų vienam ir tik vienam blokui.
- galima vaizdą suskirstyti į persidengiančius fragmentus, naudojant „slenkančio lango“ principą, t. y. kai sekantis vaizdo fragmentas skiriasi tik viena eilute ar stulpeliu (kai fragmentas „paslenkamas“ vaizde vienu pikseliu). Tokiu būdu iš vaizdo išrenkamos visos įmanomos užduoto dydžio blokų kombinacijos. Aiškiai matosi, jog skaidant tokiu būdu bus gauta daug daugiau bazinių vektorių.

Taip pat, nepriklausomai nuo pasirinktos aukščiau aprašytosios strategijos, galima taikyti kiekvienam blokui įvairias transformacijas (pasukimas, veidrodinis atspindys, spalvų inversija ir kt.) bei jų kombinacijas, tokiu būdu dar padidinant bazinių blokų kiekį nuo kelių iki keliolikos ar net keliasdešimties kartų. Vienintelė priežastis didinti bazinių blokų kiekį yra ta, jog tikėtina, kad daugiau blokų geriau atspindės visų įmanomų blokų aibę. Savaime suprantama, kad negalima pamiršti ir klasterizacijos algoritmo laiko sąnaudų, kurios gali smarkiai išaugti, padidėjus blokų kiekiui.

Algoritmo realizavimo ir testavimo metu buvo naudojama nuo 1000 iki beveik pusės milijono bazinių blokų. Tai yra didžiuliai kiekiai informacijos, kadangi programiškai kiekvienam blokui sukuriamas objektas, kuriame saugoma jo duomenų matrica, tyrimams

buvo skaičiuojamas spektras, taip pat saugomi kiti dydžiai, įtakojantys rezultatus ar darbo greitį: minimali bloko spalvos reikšmė, bloko glodumo reikšmė ir kt. Tokiems duomenims apdoroti reikalingas ne tik galingas kompiuteris, tačiau ir optimizuoti algoritmai, kurie ne tik greitai skaičiuoja, tačiau ir grąžina rezultatus, tenkinančius reikalavimus vėlesniems žingsniams.

Darbo metu buvo realizuoti eksperimentai, kurių tikslas – sumažinti pradinių blokų kiekį, tuo būdu mažinant tolimesnio žingsnio (klasterizacijos) darbo laiko sąnaudas.

Mažinant bazinių blokų kiekį buvo atliekami šie veiksmai:

- visoje atrinktų bazinių vaizdų aibėje ieškomi pasikartojantys blokai. Radus tokį bloką galima jį arba tiesiog pašalinti arba paliktajam blokui didinti koeficientą, nurodantį bloko svorį tolimesniuose žingsniuose. T. y. nors toks blokas vėliau būtų tik vienas, tačiau jis darytų didesnę įtaką skaičiavimams, nei blokai, kurių svoris yra mažesnis.
- Toliau ieškoma, kiek yra labai vienas į kitą panašių blokų. Panašūs blokai buvo ieškomi prieš tai atlikus blokų spalvų kvantizaciją, t.y. mažinant bloko spalvų kiekį nuo 256 galimų iki 64 galimų spalvų. Jei prieš ši etapą dviejų blokų pikseliai skyrėsi labai nežymiai, tai po kvantizacijos jų reikšmės suvienodėja. Tada taikant vieną iš ankstesniame žingsnyje aprašytų strategijų, galima pašalinti panašius blokus.

Eksperimentai buvo atliekami pasirinkus 20 bazinių vaizdų, iš kurių sudaromi baziniai blokai ir juose ieškomi pasikartojantys bei panašūs blokai. Skaičiavimai buvo pakartoti prie skirtingų bazinių blokų dydžių. Gauti rezultatai yra apibendrinti 3 lentelėje.

3 lentelė. Eksperimento, baziniams blokams sumažinti, rezultatai

	Bloko dydis				
	4×4	6×6	8×8	10×10	12×12
Iš viso blokų	81920	36980	20480	13520	9680
Pasikartojančių blokų	3,63 %	3,19 %	2,71 %	2,67 %	2,48 %
Panašių blokų, kartu su pasikartojančiais	12,01 %	5,28 %	3,94 %	3,51 %	3,13 %

Išanalizavus 3 lentelėje gautus duomenis, buvo padarytos tokios išvados:

- priešingai negu buvo tikėtasi, pasikartojančių, t.y. identiškų blokų vaizduose yra labai nedaug. Tas procentas dar labiau mažėja, didėjant bazinio bloko išmatavimams;

- panašių blokų yra šiek tiek daugiau, tačiau žymiau šis dydis skiriasi tik esant mažam bloko dydžiui (iki 6×6 pikselių). Kadangi didelis vaizdo suspaudimo lygis yra įmanomas tik esant didesniems baziniams blokams, tai labiau vertinti reiktų 8-10 pikselių dydžio blokus;
- kadangi vienodų bei panašių blokų paieškos algoritmas yra kvadratinio sudėtingumo, tai paieškos algoritmas gali trukti labai ilgai, esant dideliems duomenų kiekiams. Tačiau procentinės priklausomybės nuo to kinta nežymiai;
- pašalinus pasikartojančius bei panašius blokus, atsirastų svoriniai koeficientai, dėl kurių tolimesniuose žingsniuose gali žymiai pasudėtingėti klasterizacijos algoritmai.

Tad įvertinus visas šias išvadas, bazinių blokų mažinimo etapo buvo atsisakyta. Kadangi blokų sumažėtų tik 3-5 %, o laiko sąnaudos išaugtų labai smarkiai. Be to, tai gali turėti neigiamos įtakos rezultatams, o ir klasterizacijos algoritmai būtų sudėtingesni.

4.5. Bazinių blokų klasterizavimas

Šiame poskyryje yra aprašomas pagrindinis algoritmo darbo žingsnis – bazinių blokų klasterizacija. Tai etapas, kurio metu sudaroma blokų duomenų bazė, grupuojant visus bazinius blokus pagal pasirinktus kriterijus į iš anksto nustatytą kiekį pogrupių (klasterių). Tolimesniuose skyreliuose yra smulkiau aprašyta klasterizacijos sąvoka, algoritmai bei eksperimentų rezultatai.

4.5.1. Klasifikacijos ir klasterizacijos sąvoka

Klasifikacija – tai tam tikro objekto priskyrimas vienai iš galimų klasių. Klasifikacija yra klasikinis uždavinys, labiausiai vystomas statistikos ir dirbtinio intelekto specialistų. Klasifikacija skirstoma į dvi pagrindines rūšis: prižiūrima (angl. *supervised*) ir neprižiūrima (angl. *unsupervised*). Neprižiūrima klasifikacija dar yra vadinama **klasterizacija** (angl. *clustering*). Vienas iš klasterizacijos skirtumų nuo klasifikacijos yra tas, kad klasterizacijoje nėra iš anksto nustatytų klasių.

Klasteris – tai rinkinys vienas į kitą panašių duomenų objektų, kurie dėl savo panašumo gali būti traktuojami kaip viena grupė.

Klasterizacija – tai duomenų ar objektų aibės dalijimas į prasmingus poklasius, vadinamus klasteriais.

Gera klasterizacija yra tada kai:

- vidinės klasės (t. y. klasterio) objektų panašumas yra didelis;
- yra mažas panašumas objektų, priklausančių skirtingiems klasteriams.

Penki pagrindiniai metodai, taikomi klasterizacijoje:

- skaidymo algoritmai (sudaryti įvairius duomenų aibės poklasius ir juos įvertinti pagal tam tikrą kriterijų);
- hierarchiniai algoritmai (sudaryti hierarchinę duomenų ar objektų dekompoziciją);
- tankumu pagrįsti metodai (pagrįsti sąryšio ir tankumo funkcijomis);
- tinkleliu pagrįsti metodai (pagrįsti daugelio lygių panašumo struktūra);
- modeliu pagrįsti metodai (kiekvienas klasteris turi savo hipotetinį modelį. Tikslas – sudaryti kuo geriau tinkančius modelius).

4.5.2. Klasterizacija vaizdų suspaudimui

Šio algoritmo idėja – panaudoti klasterizaciją vaizdams suspausti. Tą įmanoma padaryti, jei klasterio individai labai nedaug skiriasi nuo klasterio centroido (vidurkio) ir klasterių centroidai smarkiai skiriasi vienas nuo kito. Klasterizacijos algoritmo rezultatams įvertinti naudojamos dvi charakteristikos:

- **maksimalus klasterio diametras** – tai didžiausias atstumas tarp visų klasterio individų ir klasterio centro. Šis atstumas skaičiuojamas visiems klasteriams ir išrenkamas maksimalus. Renkantis ar modifikuojant klasterizacijos algoritmą, šį dydį stengiamasi minimizuoti;
- **minimalus atstumas tarp klasterių** – tai mažiausias atstumas iš visų atstumų tarp bet kurių dviejų klasterių centroidų. Šį dydį yra siekiama maksimizuoti.

Yra daug įvairių būdų skaičiuoti atstumus tarp vaizdo blokų, tačiau buvo pasirinktas toks variantas:

$$d = \frac{1}{N}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(A_{ij} - B_{ij}\right)^2 ; \qquad (13)$$

čia A ir B - blokai, tarp kurių skaičiuojamas atstumas;

A_{ij} ir B_{ij} - konkretus bloko A ir atitinkamas bloko B pikselis;

n - bloko kraštinės dydis;

N - bloko pikselių kiekis ($N=n^2$).

Algoritmas paremtas tokia bazine idėja: jei klasterių yra pakankamai daug ir jų diametrai yra nedideli, tai klasterio centroidas labai nežymiai skiriasi nuo visų klasterio individų. Vadinasi, sudarius duomenų bazę ir išsaugojus klasterių centroidus, spaudžiant vaizdą, vietoj jo bloko būtų galima saugoti tik klasterio numerį duomenų bazėje. Jei baziniai blokai gerai atspindėjo realaus pasaulio vaizdus ir jų buvo pakankamai daug, taip pat klasterių

buvo pakankamai daug, tai vietoj originalaus vaizdo bloko įrašant klasterio centroidą, kokybė nukentėtų nesmarkiai, o suspaudimo lygį (teoriškai) galima pasiekti labai didelį.

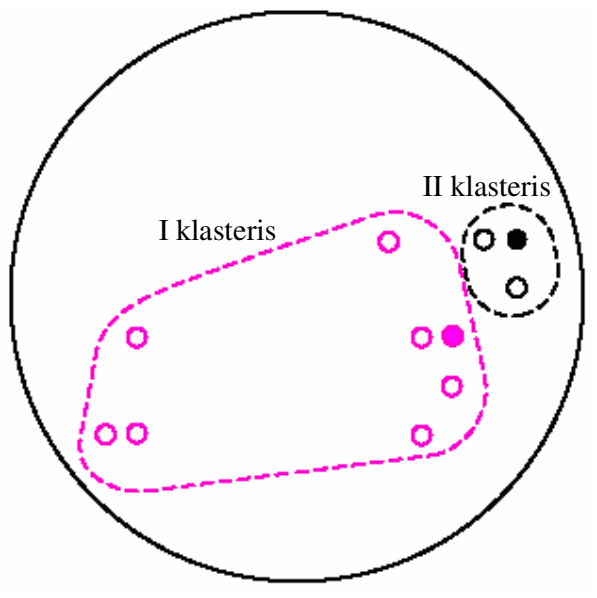
Galima sakyti, kad klasterizacija yra kritinis šio algoritmo žingsnis – nuo jo ne tik smarkiai priklauso kokybė, bet taip pat šis žingsnis yra labai reiklus laikui, ypač dirbant su dideliais bazinių blokų rinkiniais. Todėl buvo ištirti ir realizuoti keli klasterizacijos algoritmai, taip pat pasiūlytas modifikuotas (jungtinis) klasterizacijos algoritmas.

4.5.3. K-vidurkių algoritmas

K-vidurkių (angl. *k-means*) algoritmas yra iteratyvaus pobūdžio. Klasterių kiekis nurodomas iš anksto. Pirmame žingsnyje visi individai yra išskirstomi į klasterius. Tą galima daryti atsitiktinai, arba pagal kažkokį kriterijų išrenkant „specifinius“ blokus. Šiame darbe buvo pasirinktas tarpinis variantas – kiekvienam klasteriui buvo priskirta po vieną atsitiktinį bloką, o tada kiti blokai buvo priskiriami taip, kaip tai daroma sekančiuose algoritmo žingsniuose: i-tajame žingsnyje kiekvienam blokui yra žinomi jo centroidai. Tada visi individai yra perskirstomi į klasterius taip, kad atstumas iki klasterio centro būtų minimalus. Klasteriui priskyrus naują individą yra perskaičiuojamas jo centroidas ir procesas analogiškai tęsiamas toliau. Algoritmas nutraukiamas, kai vienos iteracijos metu nei vienas individas nebuvo priskirtas kitam klasteriui.

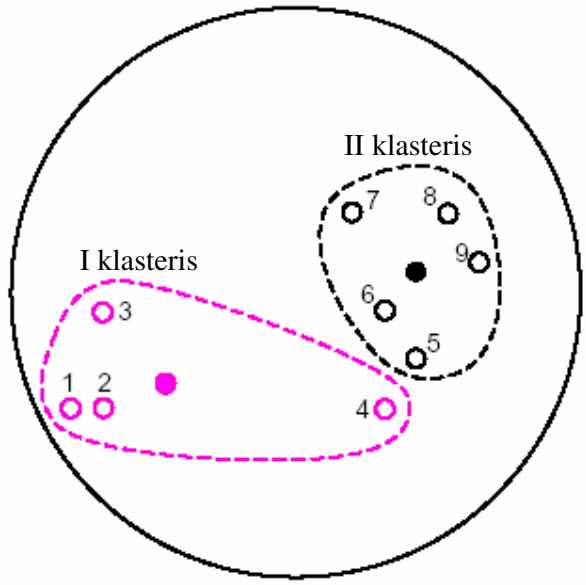
Kaip pavyzdį galima panagrinėti 9 taškų suskirstymą į du klasterius, naudojant standartinį k-vidurkių algoritmą [Faber, 1994: 142]. Šis pavyzdys nėra susijęs su vaizdo blokų klasterizacija, tačiau pavaizduoti vaizdo blokų klasterizaciją būtų labai sudėtinga – tiek vizualiai, tiek r skautiškai, kadangi kiekvienas blokas yra matrica, o klasterio centroidas – tai taip pat matrica, kuri gaunama suvidurkinus visų į klasterį papuolančių blokų matricų reikšmes. Individai (taškai) žymimi tuščiaviduriais apskritimais, klasterių centroidai – pilnaviduriais apskritimais, o klasteris pažymimas punktyrine linija. Reiktų pastebėti, kad iteratyvaus procesas konverguoja labai greitai, net esant labai blogam pradiniam individų paskirstymui į klasterius:

a) **Algoritmo inicializacija.** Pradiniai taškai pasirenkami atsitiktinai, o likę taškai priskiriami į klasterius taip, kad būtų minimalus atstumas iki klasterio centroido. Klasterių centroidai žymimi pilnaviduriais apskritimais.



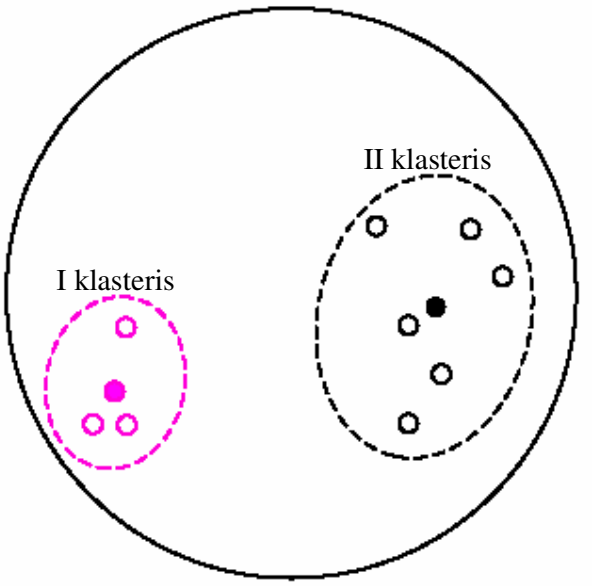
13 pav. K-vidurkių algoritmo inicializacija

b) **Rezultatai po pirmos iteracijos.** Perskaičiuojami klasterių centroidai ir visi taškai, tokia tvarka kaip jie sunumeruoti vėl perskirstomi į klasterius.



14 pav. K-vidurkių algoritmo II žingsnis

c) **Rezultatai po antros iteracijos.** Taip atrodo nusistovėjęs rezultatas, kuris, tęsiant iteracijas, nebesikeistų.



15 pav. K-vidurkių algoritmo klasterizacijos rezultatai

Matuojant algoritmo greitį buvo naudojami tik du baziniai vaizdai, siekiant sumažinti laiką, eksperimentams atlikti. Vaizdai, naudoti klasterizacijos algoritmų matavimams atlikti, pateikti 16 paveiksle.

16 pav. Vaizdai, naudoti klasterizacijos algoritmų testavimui

4 lentelėje pateikti rezultatai, apibendrinantys K-vidurkių algoritmo veikimo greitį bei kokybę, klasterizuojant bazinius blokus į 256 klasterius.

4 lentelė. K-vidurkių algoritmo matavimų rezultatai

	Bloko dydis				
	4×4	6×6	8×8	10×10	12×12
Iš viso blokų	8192	3698	2048	1352	968

Atliktų iteracijų kiekis	31	14	7	7	6
Sugaištas laikas	26,4 s	9,2 s	3,5 s	2,8 s	2,2 s
Vidutinis klasterio diametras	5,95	6,06	5,57	5,58	4,52
Vidutinis atstumas tarp klasterių	7,67	26,26	27,53	32,75	33,87
Vidutinis klasterio dydis	32	14,4	8	5,3	3,8
Maksimalus klasterio dydis	100	50	56	44	45
Minimalus klasterio dydis	3	1	1	1	1

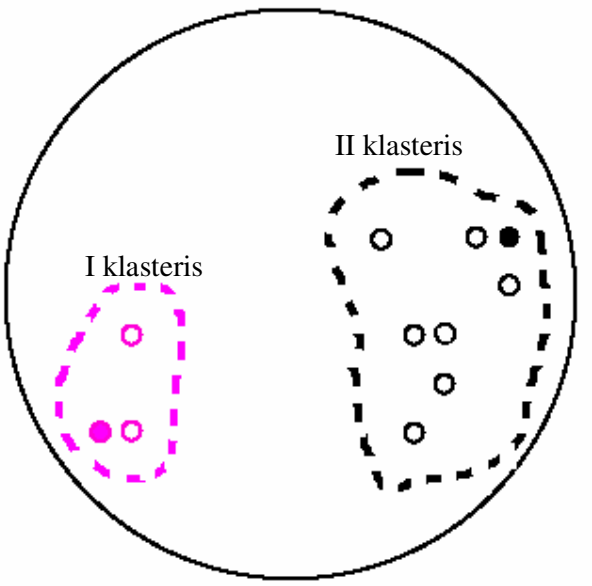
4 lentelėje vidutinis klasterio diametras ir atstumas tarp klasterių suskaičiuoti pasinaudojant 13 formule. Galima pastebėti, kad visuomet atstumas tarp panašiausių klasterių yra didesnis už atstumą nuo tolimiausiojo bloko, esančio klasteryje su didžiausiu diametru, iki to klasterio centro. Taip pat galima pastebėti netiesinį algoritmo skaičiavimų laiko augimą, didėjant blokų kiekiui, kadangi šio algoritmo sudėtingumas yra kvadratinis.

Pagrindiniai šio algoritmo privalumai – jis pakankamai greitas ir paprastas, trūkumas – klasterizacijos metrikų prasme (maksimalus klasterio diametras ir minimalus atstumas tarp klasterių) duoda prastesnius rezultatus nei kiti šiame darbe aprašyti algoritmai.

4.5.4. Tolimiausių kaimynų algoritmas

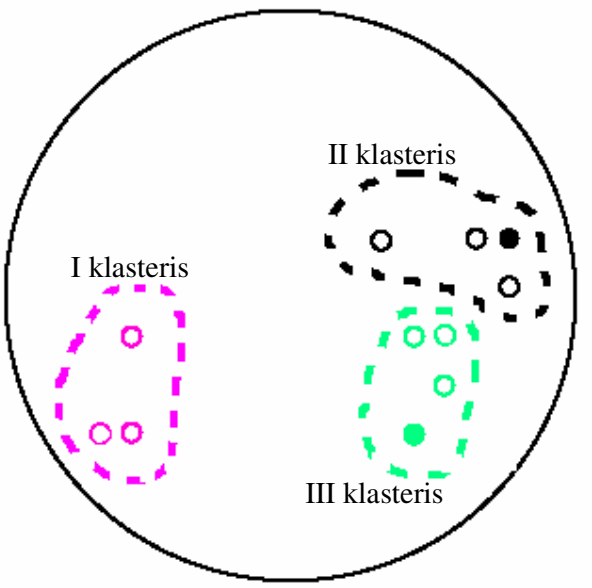
Tai algoritmas, kuris skirsto visus individus į klasterius palaipsniui – sukuria po vieną naują klasterį kiekvienoje iteracijoje. Pirmuoju žingsniu visi individai priskiriami vienam klasteriui. Tada tame klasteryje randami du individai, kurie labiausiai skiriasi vienas nuo kito. Toliau klasteris skaidomas į du klasterius, kurių pradiniai individai yra skaidomojo klasterio labiausiai besiskiriantys individai. Kol nepriskirti kiti individai, tai jie sutampa su išskaidytųjų klasterių centroidais. Toliau visi likę individai priskiriami vienam iš dviejų klasterių, analogišku principu, kaip aprašyta K-vidurkių algoritme. I-tojoje algoritmo iteracijoje pasirenkamas vienas klasteris, pagal tam tikrus kriterijus ir jis vėl skaidomas į du klasterius. Kriterijai gali būti įvairūs – pagal klasterio dydį, pagal maksimalų klasterio diametrą ir pan. Skaidymas tęsiamas tol, kol gaunamas reikiamas kiekis klasterių.

Panaudojus pavyzdį, aprašytąjį 4.5.3 skyrelyje, po pirmosios iteracijos (kai vienas klasteris yra išskaidomas į du), jau gaunami tokie patys rezultatai:



17 pav. Rezultatai po pirmosios tolimiausių kaimynų algoritmo iteracijos

Pradžioje pasirinkti du labiausiai vienas nuo kito nutolę taškai (17 paveiksle jie yra pilnaviduriai), tada visi kiti taškai priskiriami vienam ar kitam naujai sudarytam klasteriui. Tęsiant iteracijas toliau, sekančiame žingsnyje būtų skaidomas antrasis klasteris:



18 pav. Tolimiausių kaimynų algoritmo antrasis žingsnis

5 lentelėje pateikti matavimai, klasterizuojant bazinius blokus į 256 klasterius, naudojant tolimiausių kaimynų algoritmą.

5 lentelė. Tolimiausių kaimynų algoritmo matavimų rezultatai

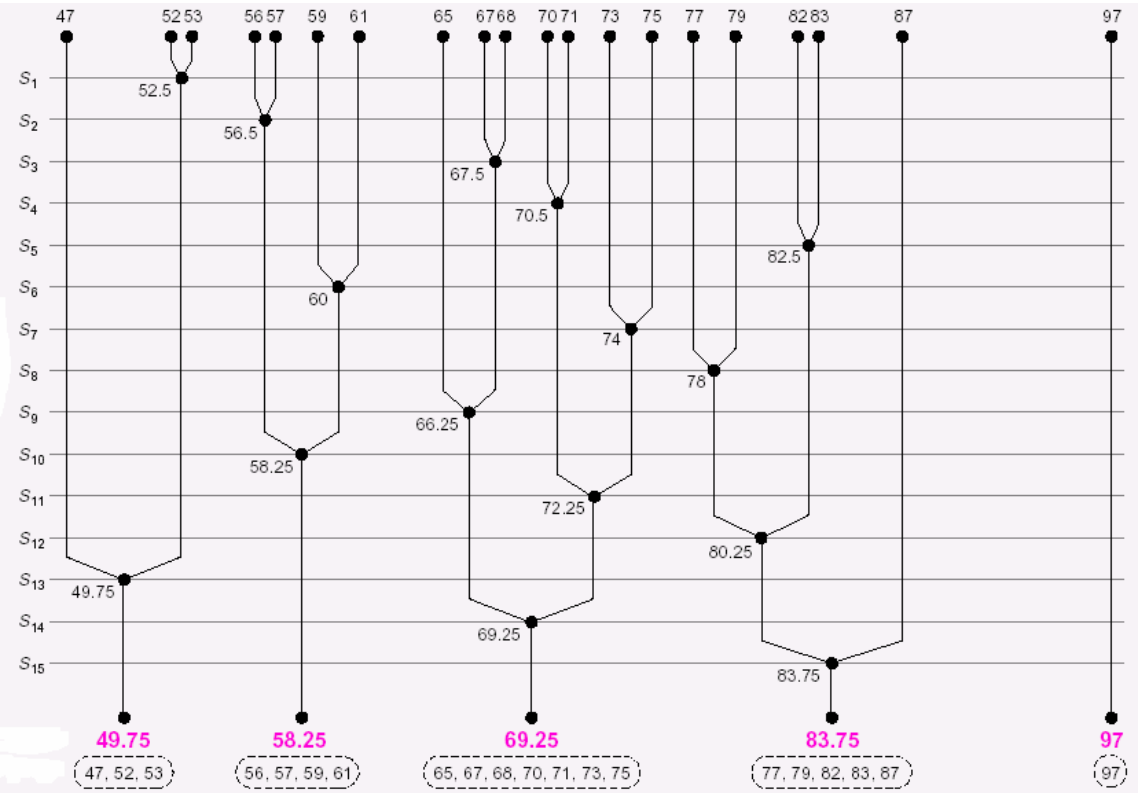
	Bloko dydis				
	4×4	6×6	8×8	10×10	12×12
Iš viso blokų	8192	3698	2048	1352	968
Sugaištas laikas	63,3 s	47,9 s	40,8 s	34,9 s	25,9 s
Vidutinis klasterio diametras	5,4	3,7	2,5	2,4	1,63
Vidutinis atstumas tarp klasterių	49,9	48,2	50,1	47,5	46,8
Vidutinis klasterio dydis	32	14,4	8	5,3	3,8
Maksimalus klasterio dydis	7352	3182	1513	834	513
Minimalus klasterio dydis	1	1	1	1	1

Iš 5 lentelės rezultatų galime pastebėti, kad šio algoritmo darbo laikas yra ilgesnis už K-kaimynų algoritmą, tačiau laiko sąnaudų didėjimas, didėjant blokų kiekiui, yra žymiai lėtesnis. Tad esant didesniems pradinių duomenų kiekiams šis algoritmas dirba greičiau. Klasterizacijos metrikų prasme šis algoritmas taip pat duoda geresnius rezultatus negu K-kaimynų algoritmas. Tolimiausių kaimynų metodo trūkumas yra tas, jog keli pirmieji algoritmo žingsniai užtrunka ilgiau, nei visos likusios iteracijos. Tą problemą galima sumažinti, pradžioj naudojant kitą (pvz., K-vidurkių) algoritmą, kuris greitai suskirsto didelį kiekį blokų į nedidelį kiekį klasterių, o toliau skaidymą jau atlikti šiuo algoritmu. Vienas iš svarbių skirtumų nuo K-vidurkių algoritmo yra tas, kad maksimalaus klasterio dydis šiuo atveju yra žymiai didesnis, o tai reiškia, jog tolimiausių kaimynų metodas atskiria labai besiskiriančius blokus vieną nuo kito, tačiau didžioji dalis blokų patenka į vieną klusterį ir, nors ir klasterizacijos rezultatai yra geresni, vėliau, atkuriant vaizdą, tai gali duoti prastesnius rezultatus. Jei į vieną klasterį patenka labai daug bazinių blokų, tai jo vidurkis negali gerai atspindėti visų į tą klasterį patenkančių blokų ir todėl gali pasireikšti blokų reikšmių vidurkinimo efektas.

4.5.5. Artimiausių kaimynų algoritmas

Tai yra algoritmas, veikiantis priešingu principu nei tolimiausių kaimynų algoritmas. Pradžioje traktuojama, kad visi individai priklauso skirtingiems klasteriams. Tada ieškomi du klasteriai su mažiausiu atstumu tarp jų centroidų ir jie sujungiami. Procesas kartojamas tol, kol gaunamas reikiamas klasterių kiekis.

Kaip pavyzdį, pademonstruokime 20 studentų testo rezultatų suskirstymą į 5 klasterius, atitinkančius 5 raidžių vertinimo sistemą (A – geriausias įvertinimas, F – prasčiausias) [Faber, 1994: 140].



19 pav. Artimiausio kaimyno algoritmo demonstracinis pavyzdys

19 paveiksle taškai reiškia kiekvieno studento testo įvertinimą (procentais ar šimtabalėje sistemoje). Kairėje pusėje S_1 – S_{15} yra algoritmo darbo žingsniai. Apačioje punktyrine linija yra apvesti nauji klasteriai, kur kiekvieno jų reikšmė atitinka įvertinimą F – A. Kiekvieno klasterio viršuje parašytas jo vidurkis. Kiekvienoje iteracijoje du artimiausi klasteriai apjungiami, kol lieka penki klasteriai. Jie atitinka naujos vertinimo sistemos F – A įvertinimus.

6 lentelė. Artimiausių kaimynų algoritmo matavimų rezultatai

	Bloko dydis		
	8×8	10×10	12×12
Iš viso blokų	2048	1352	968
Sugaištas laikas	785,9 s	284,9 s	136,2 s
Vidutinis klasterio diametras	1,15	0,89	0,53

Vidutinis atstumas tarp klasterių	52,79	50,67	49,86
Vidutinis klasterio dydis	8	5,3	3,8
Maksimalus klasterio dydis	1568	561	405
Minimalus klasterio dydis	1	1	1

Nepaisant to, jog šis algoritmas duoda geriausius klasterizacijos rezultatų įverčius, jo praktiškai neįmanoma pritaikyti esant dideliems pradinių duomenų rinkiniams. Problema yra ta, jog kiekviename žingsnyje reikia rasti atstumus tarp visų įmanomų klasterių porų, o sekančioje iteracijoje klasterių sumažėja tik vienetu. Kaip pvz., 1 milijoną taškų suskirstyti į 256 klasterius galima $\frac{256^{1,000,000}}{256!}$ būdais. Šis skaičius yra didesnis už vienetą su dviem milijonais nulių [Faber, 1994: 141].

4.5.6. „Jungtinis“ algoritmas

Įvertinus visų anksčiau aprašytų algoritmų privalumus ir trūkumus, įmanoma juos sujungti taip, kad tuo pačiu būtu gerinamas ir algoritmo darbo laikas ir rezultatų kokybė. Tam reiktų įvertinti, kuris algoritmas kokiomis sąlygomis dirba greičiausiai:

- **K-vidurkių** algoritmas dirba greitai, jei klasterių kiekis nėra didelis. Didelis blokų kiekis mažiau veikia darbo greitį nei didelis klasterių kiekis. Taip yra todėl, kad kiekvienoje iteracijoje visus blokus reikia palyginti su visais klasteriais, tačiau palyginimų kiekis skiriasi nežymiai nuo blokų kiekio, jei blokų yra daug daugiau nei klasterių. Vadinasi, šis algoritmas gerai tiktų pradiniam individų suskirstymui į nedidelį kiekį klasterių;
- **tolimiausių kaimynų** algoritmas lėčiausiai dirba pirmosiose iteracijose, kai reikia suskaidyti didelius klasterius, nes jo sudėtingumas yra kvadratinės priklausomybės nuo individų kiekio klasteryje, kadangi reikia rasti du labiausiai besiskiriančius blokus klasteryje. Tarkime, kad pirmojo klasterio dydis yra *N*. Jame surasti du labiausiai besiskiriančius blokus prireiks P_1 palyginimų:

$$P_1 = \frac{N \cdot (N-1)}{2} = \frac{N^2 - N}{2}; \qquad (14)$$

Tarkime, jog po išskaidymo bus sukurti du nauji klasteriai, kurių dydžiai yra apie $\frac{N}{2}$. Išskaidyti tokiam klasteriui prireiks P_2 palyginimų:

$$P_2 = \frac{N/2 \cdot \left(N/2 - 1\right)}{2} = \frac{N^2 - 2N}{8}; \quad (15)$$

Vadinasi, abiejų naujų klasterių išskaidymui reikės daugiau nei dvigubai trumpesnio laiko nei pradinio klasterio išskaidymui. Todėl šis algoritmas tiktų tada, kai didelis kiekis individų jau yra suskirstytas į tam tikrą kiekį pradinių klasterių;

- **artimiausių kaimynų** algoritmas pagreitinti skaičiavimo rezultatų negali, tačiau pritaikius ir jį, galima nemažai pagerinti klasterizacijos rezultatus. Jei po tolimiausių kaimynų algoritmo darbo bus sudaryta šiek tiek daugiau klasterių nei yra reikalinga, naudojant artimiausių kaimynų metodą, galima sujungti klasterius tol, kol bus gautas reikiamas jų kiekis, o rezultatai, tikėtina, kad bus geresni, nei iš karto kuriant reikiamą kiekį klasterių su tolimiausių kaimynų metodu.

Apibendrinant algoritmą galima apibūdinti taip: tarkime, kad reikia individus suskirstyti į N klasterių. Tada, panaudojus K-vidurkių algoritmą, individai suskirstomi į M klasterių ($M < N$). Po šio žingsnio, panaudojus tolimiausių kaimynų algoritmą sudaromi K klasteriai ($K > N$). Ir galiausiai su artimiausių kaimynų algoritmu yra sudaromi N klasterių. Skaičiai M bei K yra parenkami eksperimentiškai, atsižvelgiant į blokų bei klasterių kiekį, bei tikslą (kokybę ar greitį siekiama optimizuoti).

7 lentelė. Jungtinio algoritmo matavimų rezultatai

	Bloko dydis				
	4×4	6×6	8×8	10×10	12×12
Iš viso blokų	8192	3698	2048	1352	968
Sugaištas laikas	4,8 s	3,7 s	3,0 s	3,2 s	3,1 s
Vidutinis klasterio diametras	5,28	4,06	2,21	1,54	0,86
Vidutinis atstumas tarp klasterių	50,48	49,42	50,91	48,34	48,46
Vidutinis klasterio dydis	32	14,4	8	5,3	3,8
Maksimalus klasterio dydis	5709	1928	928	542	372

Minimalus klasterio dydis	1	1	1	1	1

4.5.7. Klasterizacijos algoritmų apibendrinimas

Renkantis klasterizacijos algoritmą, kurio sudaryta duomenų bazė vėliau bus naudojama vaizdams spausti, reiktų atkreipti dėmesį į keletą dalykų:

- klasterizacijos metu naudojami didžiuliai duomenų kiekiai, tad algoritmas turi rasti rezultatą per priimtiną laiko tarpą;
- nors klasterizuojant laikas yra labai svarbus parametras, tačiau ne svarbiausias. Klasterizacija yra atliekama vieną kartą ir toliau naudojami tik jos rezultatai;
- svarbiausias klasterizacijos parametras – rezultatų kokybė. Problema yra ta, kad bendru atveju taikomos klasterizacijos metrikos, tokios kaip klasterio diametras ir atstumas tarp klasterių, nebūtinai garantuoja geresnį vizualinį vaizdą, atkūrus vaizdą iš suspausto formato;
- klasterizacija yra kritinis žingsnis, labiausiai įtakojantis atkurto vaizdo kokybę. Tolimesni tyrimai turėtų būti vykdomi būtent klasterizacijos algoritmų ir parametrų tyrimui, siekiant išsiaiškinti jų įtaką atstatyto vaizdo kokybei.

Klasterizacijos įtaką atkurto vaizdo kokybei galima aiškiai pamatyti 20 paveiksle. Šiuo atveju, klasterizacijos pradiniai duomenys buvo vaizdai, nurodyti 16 paveiksle. Vaizdai buvo skaidomi į 4×4 dydžio vaizdo blokus ir klasterizuojami į 512 klasterių. Kairėje pusėje matomi originalūs vaizdai, viduryje – atkurti vaizdai, kai naudojamas K-vidurkių algoritmas, dešinėje – atkurti vaizdai, kai naudojamas jungtinis klasterizacijos algoritmas, aprašytas 4.5.6 skyrelyje.

20 pav. Klasterizacijos algoritmų įtaka atkurto vaizdo kokybei

Paanalizavus 20 paveikslą galima padaryti keletą galutinių išvadų apie klasterizacijos algoritmus ir jų įtaką vaizdo kokybei:

- vaizdo suspaudimo algoritmas duoda gerus rezultatus net ir tada, kai bazinių vaizdų yra labai nedaug ir klasterių taip pat nėra daug. Didinat bazinių vaizdų kiekį ir klasterių kiekį kokybė tik gerėtų, o tada galima didinti suspaudimo lygį, didinat bazinio bloko dydį, jei tenkina atkurto vaizdo kokybė;
- esant tiems patiems baziniams vaizdams ir tokios paties dydžio baziniams blokams bei klasterių kiekiui, skirtingi algoritmai ne tik dirba skirtingą laiko tarpą, tačiau duoda ir skirtingus rezultatus, o tai įtakoja ir atkurto vaizdo kokybę. Nors jungtinis algoritmas ir duoda geresnius klasterizacijos metrikų rezultatus, tačiau vizualiai rezultatai gali būti prastesni, tas ypač matosi 20 paveikslo portretiniame vaizde – viduryje yra vaizdas, kuriam buvo naudojamas K-vidurkių algoritmas, o dešinėje – vaizdas, kuriam buvo naudojamas jungtinis algoritmas. Jungtinio algoritmo atkurtame vaizde daug ryškiau pasireiškia blokinis efektas ir blokų reikšmių vidurkinimo efektas. Peizažo vaizde kokybė skiriasi ne taip ryškiai, vadinasi, algoritmo pasirinkimas gali priklausyti ir nuo vaizdų klasės, kuriai pagrinde bus taikomas suspaudimas.

4.6. Vaizdo suspaudimas

Šiame etape atliekamas tikrasis vaizdo suspaudimas, jo metu naudojami klasterizacijos algoritmo sudarytų klasterių duomenų bazė. Pirmiausia vaizdas skaidomas į blokus, kurių dydis sutampa su duomenų bazės blokų dydžiais. Duomenų bazėje yra saugomi kiekvieno klasterio centroidai, t. y. vidurkiai. Jei klasterių diametrai nėra dideli, tai centroidai yra labai panašūs į bet kurį to klasterio individą. Todėl vietoj originalaus bloko galima saugoti tik klasterio, į kurį blokas yra panašiausias, numerį. Bloko panašumas randamas pagal 13 formulę, skaičiuojant atstumą nuo nagrinėjamojo bloko iki kiekvieno klasterio duomenų bazėje centroido. Jei klasterių yra *N*, tai tokiam numeriui užsaugoti reikės $\log_2 N$ bitų (saugant originalų bloką, reiktų saugoti $n \cdot n$ baitų (vaizdams su pilkumo skale (angl. *greyscale*)), kur n – bloko dydis).

Taip pat svarbu paminėti, kad prieš klasterizaciją blokai dar yra apdorojami – iš kiekvieno bloko pikselio spalvos reikšmės yra atimama to bloko minimalios spalvos reikšmė. Tokiu būdu blokai, kurie skiriasi tik pastovia dedamąja (duomenų srityje), kuri vizualiai atrodo kaip šviesos intensyvumas, yra traktuojami kaip vienodi blokai. Tad koduojamam blokui realiai yra saugoma jo minimali spalva bei jo klasterio numeris.

Reiktų dar paminėti, jog be visų anksčiau aprašytųjų veiksmų yra atliekamas ir dar vienas veiksmas – prieš klasterizacija bei kodavimą, iš visų bloko pikselių reikšmių yra atimama to bloko minimali pikselio reikšmė. Tokiu būdu blokai, kurie skiriasi tik pastovia dedamąja (duomenų srityje), kuri vizualiai atrodo kaip šviesos intensyvumas, yra traktuojami kaip vienodi blokai. Tad koduojamam blokui realiai yra saugoma jo minimali spalva bei jo klasterio numeris.

Vaizdo suspaudimo koeficientą galima rasti pagal tokią formulę:

$$\alpha = \frac{n^2 \cdot p}{p + \log_2 N}; \tag{16}$$

čia
- α – vaizdo suspaudimo koeficientas, parodantis kiek kartų suspaustas vaizdas yra mažesnis už pradinį vaizdą;
- n – bloko kraštinės dydis pikseliais;
- p – pikselių kiekis, skiriamas vienai spalvai koduoti (paprastai $p = 8$);
- N – klasterių kiekis ($N = 2^m$);

Skaitiklyje yra randamas originalaus bloko dydis – n^2 taškų ir kiekvienam iš jų skiriama p bitų. Vardiklyje – p bitų skiriama minimaliai spalvai ir $\log_2 N$ bitų skiriama klasterio numeriui išsaugoti. Pavyzdiniai suspaudimo koeficiento dydžiai pavaizduoti 8 lentelėje

(viršuje – klasterių kiekis, kairėje – bloko dydis, susikirtimuose – suspaudimo koeficientas; pilka spalva pažymėtos tos reikšmės, kur realiai galima pasiekti be ypač didelių duomenų bazės sudarymo sąnaudų ir kol nelabai daug nukenčia vaizdo kokybė):

8 lentelė. Pavyzdiniai algoritmo duomenų suspaudimo koeficiento dydžiai

	256	1024	4096	16384	65536	262144	1048576
4	8,0	7,1	6,4	5,8	5,3	4,9	4,6
6	18,0	16,0	14,4	13,1	12,0	11,1	10,3
8	32,0	28,4	25,6	23,3	21,3	19,7	18,3
10	50,0	44,4	40,0	36,4	33,3	30,8	28,6
12	72,0	64,0	57,6	52,4	48,0	44,3	41,1
14	98,0	87,1	78,4	71,3	65,3	60,3	56,0
16	128,0	113,8	102,4	93,1	85,3	78,8	73,1

Taip pat svarbu paminėti, kad vaizdo suspaudimas yra pastovus, esant pasirinktam kiekiui klasterių ir bazinių blokų dydžiui, t. y. vaizdo suspaudimo koeficientas yra nepriklausomas nuo paties vaizdo. Taip pat tai yra tik paties kodavimo išgaunamas suspaudimas. Tokie duomenys dar neturėtų būti tiesiogiai rašomi į failą, bet jiems dar turi būti pritaikomas entropinis kodavimas, t. y. suspaudimo algoritmas, naudojantis suspaudimo schemą be informacijos praradimo, kaip pvz. Hufmano kodavimas ar kodavimas sudarant žodyną. Pritaikius entropinį kodavimą įmanoma pasiekti nuo keliolikos procentų iki kelių kartų suspaudimą. Tad realus suspaudimas būtų dar gerokai didesnis, nei pateikta 8 lentelėje.

4.7. Pradinio vaizdo atstatymas

Vaizdo atkūrimo metu nuskaitoma informacija iš suspausto vaizdo duomenų failo ir vartotojui sugeneruojamas vaizdas, kiek įmanoma panašesnis į pradinį vaizdą. Ši procedūra pakankamai greita ir paprasta – kiekvienam užkoduotam blokui saugoma tokia informacija: minimali bloko spalva bei klasterio numeris duomenų bazėje. Atkuriant vaizdą, originalus blokas pakeičiamas klasterio iš duomenų bazės centroidu ir prie jo pikselių reikšmių yra prisumuojama minimali bloko spalva.

Pavyzdinis vaizdas ir atstatyti po suspaudimo jo vaizdai, esant skirtingiems suspaudimo parametrams, pavaizduoti 21 – 23 paveiksluose.

21 pav. Originalus vaizdas ir 6,4 kartus suspaustas vaizdas (dešinėje)

21 paveiksle pavaizduotas suspaustas vaizdas buvo atkurtas naudojant 32768 klasterius, esant 4×4 baziniam bloko dydžiui.

22 pav. Originalus ir 13 kartų suspaustas vaizdas (dešinėje)

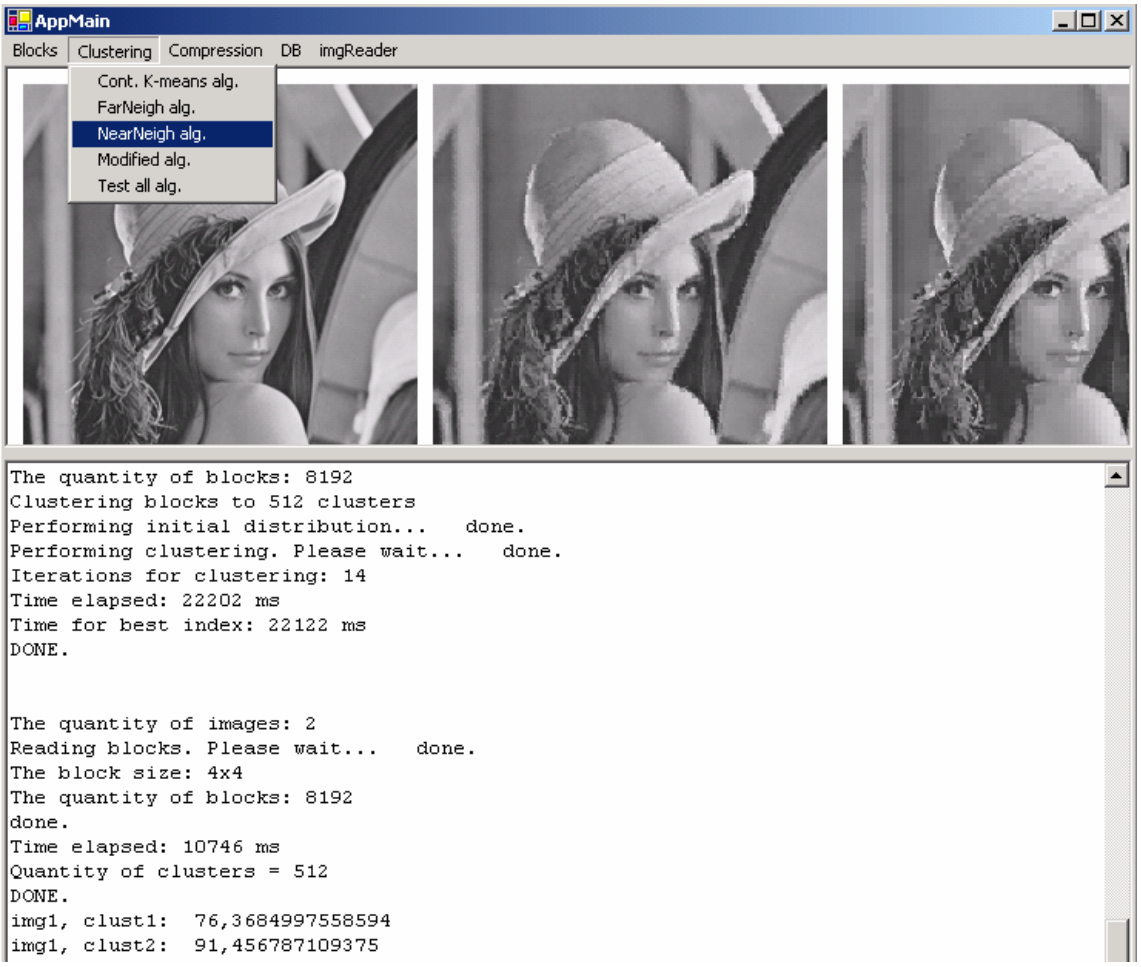
22 paveiksle pavaizduotas suspaustas vaizdas buvo atkurtas naudojant 16384 klasterius, esant 6×6 baziniam bloko dydžiui.

23 pav. Originalus ir 22 kartus suspaustas vaizdas (dešinėje)

23 paveiksle pavaizduotas suspaustas vaizdas buvo atkurtas naudojant 32768 klasterius, esant 8×8 baziniam bloko dydžiui. Šiame paveiksle jau gana aiškiai matosi blokinis efektas. Taip yra todėl, kad bazinio bloko dydis yra didelis (8 pikseliai), o klasterių kiekis yra palyginus mažas. Norint pagerinti vaizdo kokybę, esant tam pačiam bloko dydžiui, reiktų naudoti bent kelis kartus daugiau klasterių. Dėl to nežymiai nukristų suspaudimo lygis, tačiau smarkiai pagerėtų atkurto vaizdo kokybė. Tokiu būdu, didinant bazinio bloko dydį ir tuo pačiu didinant klasterių kiekį, galima išlaikyti pastovią vaizdo kokybę, tačiau didinti vaizdo suspaudimo koeficientą. Teoriškai taip galima pasiekti labai didelį suspaudimo koeficientą. Praktiškai atsiremiama į realizacijos problemas: reikia daug vietos duomenų bazei saugoti, ilgai trunka klasterizacijos procesas, esant dideliai duomenų bazei ilgėja vaizdo suspaudimo laikas ir pan.

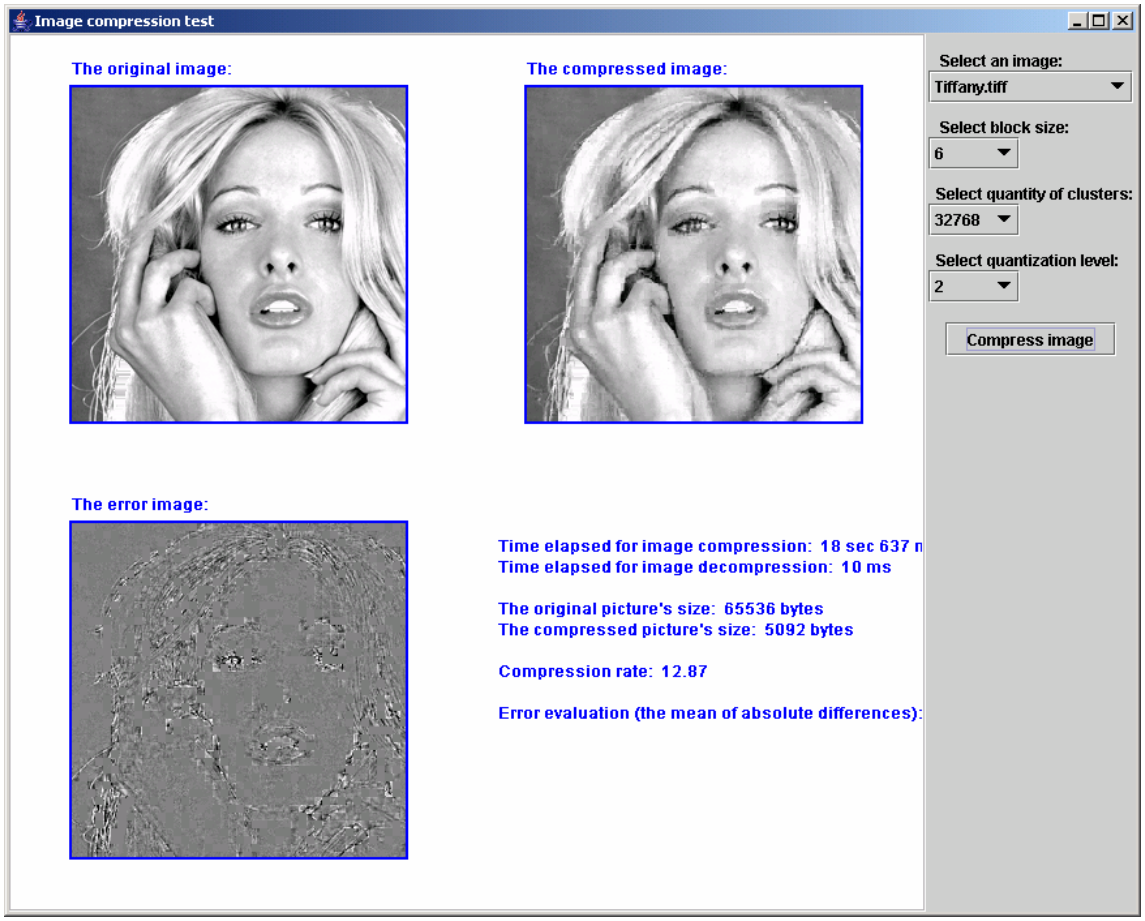
4.8. Realizuota programinė įranga

Šio darbo metu buvo realizuoti keli programiniai paketai, skirti algoritmo testavimui bei greičio matavimui. Viena sistema buvo sukurta naudojant Microsoft .NET platformą ir C# programavimo kalbą. Ši programa buvo daugiausiai naudojama eksperimentams atlikti.



24 pav. Algoritmo testavimo darbo aplinka

Antroji sistema buvo sukurta naudojant Java programavimo kalbą ir jos paskirtis – vaizdo blokų duomenų bazės sudarymas, vaizdo suspaudimas, suspaustų duomenų įrašymas į failą bei duomenų iš failo nuskaitymas ir vaizdo atstatymas.



25 pav. Java programinis paketas vaizdų suspaudimui ir atstatymui atlikti

5. IŠVADOS

5.1. Atlikti darbai

Šiame darbe pasiūlomas naujas vaizdų suspaudimo algoritmas, naudojantis euristinius duomenų klasterizacijos metodus realaus pasaulio vaizdų fragmentų klasterizacijai atlikti bei duomenų bazei sudaryti. Nors klasterizacijos algoritmai žinomi ir naudojami jau seniai, nerasta darbų, kad klasterizacija būtų naudojama vaizdų suspaudimui atlikti. Tad galima teigti, jog darbas yra naujas, nors atskiruose darbo etapuose ir naudojami klasikiniai algoritmai.

Darbo metu buvo apžvelgi šiuo metu populiariausi vaizdų suspaudimo algoritmai, aptarti algoritmų privalumai bei trūkumai. Taip pat padaryta klasterizacijos algoritmų apžvalga bei tyrimai, siekiant parinkti tinkamiausią metodą vaizdų suspaudimui atlikti.

Antrajame magistratūros semestre, studijuojant Danijos Aalborgo universitete, daugiamačių signalų apdorojimo (angl. *Multidimensional Signal Processing*) modulyje buvo pristatyta bazinė siūlomo algoritmo idėja, parengtas pranešimas bei suprogramuotas programinės įrangos karkasas. Taip pat buvo atliktas darbas „Java ir C kalbų tinkamumas signalų apdorojimo uždaviniams spręsti“. Šio darbo rezultatai buvo panaudoti renkantis programavimo priemones, taip pat ta tema parašytas straipsnis, įtrauktas į tarptautinės konferencijos, skirtos įterptinėms sistemoms, pranešimų medžiagą (žr. 1 priedą).

Nepaisant siūlomo algoritmo naujumo, yra pasiekiami dideli vaizdų suspaudimo lygiai, išlaikant gerą vaizdo kokybę. Žinoma, algoritmas turi ir savų trūkumų, tačiau didžiąją jų dalį įmanoma panaikinti, tęsiant toliau darbus, kaip aprašyta 5.2 skyriuje.

5.2. Tolimesni tyrimai

Siekiant tobulinti pasiūlytąjį algoritmą, darbai turėtų būti tęsiami:

- siekiant gerinti algoritmo atkurto vaizdo kokybę, reikia atlikti išsamesnę klasterizacijos algoritmų analizę bei įvertinti algoritmų bei jų parametrų įtaką atkuriamo vaizdo kokybei;
- siekiant pagreitinti klasterizacijos etapą, reiktų modifikuoti klasterizacijos algoritmą. Vienas iš galimų patobulinimų – atlikti „protingesnį“ pradinių duomenų suskirstymą į klasterius. Šiuo metu pradinis suskirstymas yra atliekamas atsitiktinai. Siūloma modifikacija – panaudoti vaizdų glodumo charakteristiką pradiniame klasterizacijos etape, t. y. prieš klasterizaciją kiekvienam blokui būtų suskaičiuojamas jo spektro glodumas ir pirmiausia

blokai būtų klasterizuojami tik pagal glodumą. Tai paspartintų klasterizaciją keliasdešimt ar net kelis šimtus kartų. Tokių būdų būtų sudaroma pradinė aibė klasterių, kurie vėliau būtų skaidomi naudojant K-vidurkių (ar kitą) klasterizacijos algoritmą;

- tyrimai parodė, kad klasterizacijos algoritmas, duodantis geresnes klasterizacijos metrikų reikšmes, nebūtinai duoda geresnius vizualinius rezultatus. Bendradarbiaujant su žmogaus regos specialistais reiktų rasti savybes ar charakteristikas, kurių pagalba būtų galima atlikti kokybiškesnę bazinių blokų klasterizaciją;

Taip pat, keletą šio algoritmo idėjų galima pritaikyti ir kitiems šiuo metu populiariems ar dar tik kuriamiems vaizdų suspaudimo algoritmams. Labiausiai iš jų verta paminėti fraktalinius vaizdų suspaudimo algoritmus, kuriuose būtų galima pritaikyti blokų klasterizacijos idėją siekiant paspartinti panašių blokų paiešką vaizde.

Galima atlikti statistinius tyrimus, siekiant ištirti dažniausiai pasitaikančių vaizdo fragmentų pobūdį bei kitas priklausomybes. Į tai būtų galima atsižvelgi sudarant bazinių blokų duomenų bazę. Taip pat galima atlikti jau sudarytos duomenų bazės papildymus, jei spaudžiamojo vaizdo konkretus blokas nėra pakankamai panašus nė į vieną duomenų bazės klasterį.

Šiuo metu algoritmo realizacijoje nevykdomas entropinis duomenų kodavimas, po duomenų suspaudimo, naudojant sudarytąją duomenų bazę. Šis žingsnis yra neabejotinai reikalingas, kadangi tai gali nemažai padidinti vaizdo suspaudimo koeficientą, tačiau dėl to visiškai nenukenčia atstatomo vaizdo kokybė.

Tolimesniuose etapuose sukurtą metodą galima pritaikyti spalvotų vaizdų bei filmuotos medžiagos suspaudimui atlikti.

6. LITERATŪRA

1. Bracamonte, J. A Multiplierless Implementation Scheme for the JPEG Image Coding Algorithm // J. Bracamonte, P. Stadelmann, M. Ansorge, F. Pellandini. IEEE Nordig Signal Procesing Symposium, 2000.
2. Faber, V. Clustering and the Continuous k-Means Algorithm // Los Alamos Science, 1994, Nr. 22, p. 138-144.
3. Franti, P. Conpression of Digital Images by Block Truncation Coding: A Survey // P. Franti, O. Nevalainen, T. Kaukoranta. The Computer Journal, Vol. 37, No. 4, 1994.
4. Girod, B. Image Compression Overview [online]. 2004 [žiūrėta 2004-05-08]. Prieiga per internetą: <http://www.stanford.edu/class/ee398a/handouts/01-ImageCompressionOverview.pdf>.
5. Gonzalez, C. Digital Image Processing / C. Gonzalez, E. Woods. Addison-Wesley, 1993. 716 p. ISBN 0-201-50803-6.
6. Graps, A.L. An Introduction to Wavelets // IEEE Computational Sciences and Engineering, Vol. 2, No. 2, 1995.
7. Ifeachor, E.C. Digital Signal Processing. A Practical Approach / E.C. Ifeachor, B.W. Jervis. Addison-Wesley, 1993. 760 p.
8. Matteson, R.G. Introduction to Document Image Processing Techniques. Norwood, 1995. 259 p. ISBN 0-89006-492-X.
9. Mullen, K.T. The contrast sensitivity of human color vision to red-green and blue-yellow chromatic gratings. // J. Physiol, 1985, Nr. 359, 381-400 p.
10. Nelson, M. Data Compression Book / M. Nelson, J.L. Gailly. Hungry Minds, 1995. 557 p.
11. Oppenheim, A.V. Discrete-Time Signal Processing 2nd ed. / A.V. Oppenheim, R.W. Shafer. Prentice Hall, 1999. 870 p.
12. Pennebaker, W.B. Jpeg: Still Image Data Compression Standard / W.B. Pennebaker, J.L. Mitchell. Kluwer Academic Publishers, 1993. 638 p. ISBN 0442012721.
13. Qiu, G. Color Image Indexing Using BTC // IEEE Transactions on Image Processing, Vol. 12, No. 1, 2003.
14. Rabiee, H.R. Multiresolution Image Compression with BSP Trees and Multilevel BTC // H.R. Rabiee, R.L. Kashyap, H. Radha. IEEE International Conference on Image Processing (ICIP'95), Washington D.C., 1995.

15. Saha, S. Image Compression – from DCT to Wavelets: A Review // ACM Crossroads Student Magazine, Nr. 4, 2000.
16. Valantinas, J. Apie baigtinių automatų teorijos taikymą apdorojant dvimačius vaizdus // Lietuvos matematikų draugijos XXXVIII konferencijos darbai, Vilnius, Technika, 1997.
17. Vatterli, M. Wavelets and Subband coding / M. Vatterli, J. Kovacevic. Prentice Hall, 1995. 488 p. ISBN 0130970808.

7. PRIEDAI

1 Priedas. Java ir C greitaveikos įvertinimas skaitmeninio signalų apdorojimo srityje (straipsnis)