E-kursuse "Pildiinfo töötlus" materjalid

Aine maht 5 EAP

Aare Luts (Tartu Ülikool), 2012

1. teema õppematerjalid

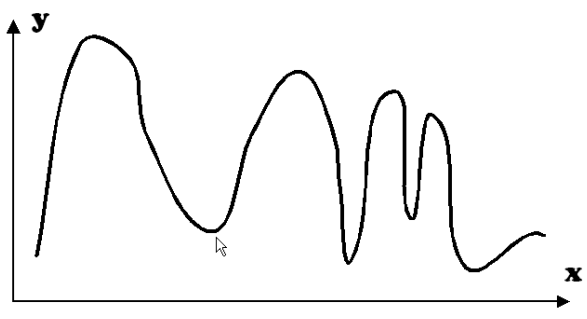
Digitaalpildi põhiomadused. Värvused, valevärvused ja nägemistaju.

Õpikeskkond: TÜ Moodle
Kursus: Pildiinfo töötlus (LOFY.05.055)
Koosta raamat: 1. teema õppematerjalid
Printed by: Aare Luts
Kuupäev: teisipäev, 22 mai 2012, 08:41

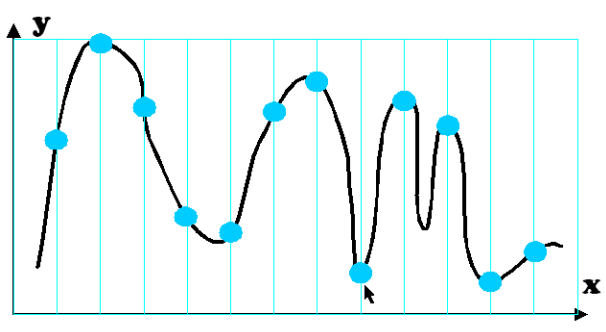
Sisukord

1.1. Signaali digitaliseerimine

Vahetult alljärgnev võib mõnele tunduda triviaalne, aga kogemused näitavad et tihtipeale ei tee paha see üle korrata. Sugugi ei ole nii et kui meil juba on digitaalpilt, siis on see vaat et parem kui analoogpilt, ja siis suudavad kriminalistid sellest alati kõik, kaasaarvatud eluloo välja lugeda. Mõnel juhul võib üht ja teist välja lugeda küll, aga asja sisuliseks tundmiseks peab teadma, mis on võimalik ja millal on see võimalik.

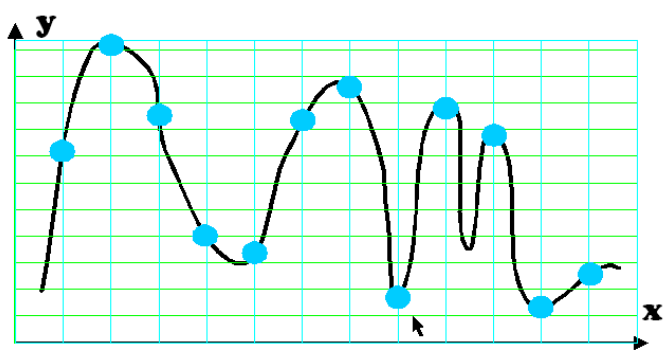


Alustame ümbritsevast maailmast. Olgu meil mingi signaal, ja olgu kas tahtmine või siis vajadus seda digitaliseerida. Olgu signaal näiteks selline nagu näha kõrvaloleval pildil. Reaalmaailmaga tugevama seose loomiseks võib ette kujutada et see on väljavõte helitugevuse muutumisest ajas. X-teljel oleks sellisel juhul aeg ja y-teljel oleks helitugevus mingites füüsikaliselt mõõdetavates ühikutes. Signaali digitaliseerimine tähendab et meil tuleb see ära paigutada mingisse etteantud suurusega arvmassiivi. Arvmassiiv võib olla suur või väike, aga igal juhul on ta lõpliku suurusega. Antud juhul saab hakkama ühemõõtmelise massiiviga, selle massiivi iga element vastaks mingile ajale ja elemendi väärtus vastaks helitugevusele. Oletagem et meile antud (lubatud) massiivi elementide lubatud arv on 14. Nagu juba mainitud, see võib olla ka nii suurem kui väiksem, aga igal juhul on see lõplik. Valitud 14 ühikut pole seetõttu, vähemalt esimeses lähenduses, teistest valikutest ei halvem ega parem. Aga et massiivi elementide mingi arv tuleb valida (ei saa olla nii et me ei tea, kui suur maatriks on, nii ei saa arvutada), valime 14.



Graafiliselt on sellise valiku ekvivalendiks esialgsele signaalile 14-sõlmelise võre asetamine, mille iga sõlm vastab loodava massiivi ühele elemendile (vt joonist). Kui võre on paigas, saame asuda maatriksit täitma. Täitmine tähendab antud juhul seda et maatriksi igasse elementi tuleb kirjutada sellele elemendile vastava sõlme kohal asuv väärtus. Jooniselt on selgesti näha et võre sõlmede vahel signaali väärtus muutub, aga midagi pole teha, neid vahepealseid väärtusi pole meil kuhugi panna, sest meie maatriksi pikkus on ainult 14 väärtust.

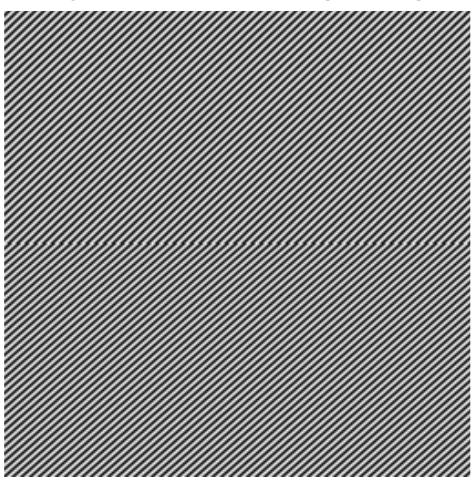
Eelnenud etapis oli meil lubamatu lihtsustus. Nimelt, x-telje sihis võtsime me küll just need väärtused, millised me ka tegelikult saame arvmassiivi panna, aga võetud väärtused ise on lõpmatu täpsusega. Teisisõnu, esialgu oleks justkui nii et meie maatriksisse kirjutatavate arvude täpsus peab olema lõpmatu (võtame selle väärtuse, mis võre sõlme kohal on, ja salvestame selle täpselt). See ei lähe kohe mitte. Ka salvestatavate arvude täpsus peab olema lõplik, sest lõpmatu täpsus nõuaks lõpmatult pikki muutujaid, ehk lõpmatult palju ruumi. Kui me määrame ära, millist täpsust me tahame, võime me öelda et me määrasime ära, mitu üksteisest erinevat arvu on salvestamiseks lubatud. Ütleme et lubatud erinevate arvude hulk on 12. Tundub justkui vähe (näiteks, lubatud on ainult arvud 0 kuni 11), aga põhimõtteliselt pole vahet, millise hulga me valime, on tähtis et see oleks lõplik. 12 on lõplik. Ja pealegi, piltide puhul on tavaliseks väärtuseks 256, nii et ega 12 nüüd nii väike ka ei ole. Jällegi, analoogiliselt eelnenuga 12-sõlmeline võre ja nüüd saab valida ainult ühe arvu neist, mis on ära näidatud kahe võre lõikepunktidega, sest y-teljelise võre iga sõlm tähendab nüüd ühte lubatud arvudest (vt joonist).



X-telje sihis saame me salvestada ainult sinise võre sõlmede kohal, ja y-telje sihis ainult rohelise võre sõlmede kohal. Et mõlemad peavad kehtima, saabki salvestada ainult seal kus võred lõikuvad. Mida see kaasa toob? Nagu jooniselt näha, kõigi siniste punktide parameetreid ei saa sellistena salvestada, sest y-telje sihis pole arvu, mida salvestada. Salvestamiseks tuleb sinised punktid nihutada kohta, kus mõlemad võred lõikuvad, arvatavasti siis tegelikuga võrreldes lähimasse asukohta. Ikkagi, see tekitab täiendava moonutuse, sest nii mõnigi nihutatud punkt ei asu enam joonel, mida me tahtsime digitaliseerida. Seega, salvestatakse mitte originaalile vastav väärtus, vaid sellele lähim võredega sobiv väärtus. Tulemuseks on kogum rohelisi punkte, mis vastavad digitlaiseerimisvõre reeglitele ja neis piirides esitavad esialgset objekti parimal võimalikul viisil (vt joonis). Kui nüüd keegi väidab et see punktikogum on parem kui esialgne joon, julgeksin ma küsida et mis mõttes.

Loogiliselt kerkib küsimus et mida siis teha et digitaliseerumisel säiluksid esialgse signaali kõik olulised omadused. Kõige lühemalt saab öelda et selleks tuleb valida sobiv võre. Milline võre, selle kriteeriumi esitame matamaatiliselt siis kui hakkame tegelema sagedusruumi ja Fourier' teisendusega. Siin tuleks teadmiseks võtta et ebasobiv võre ei sobi digitaliseerimiseks kohe mitte, aga ega ta ise seda ütlema ei tule.

Ebasobiva digitaliseerimisvõre kasutamisel võivad ilmneda ootamatud efektid, näit kõrvaloleva pildi puhul. Jah, see juba on digitaalpilt. Kuid me võime ette kujutada et see on tegelik signaal, mida hakatakse (täiendavalt) digitaliseerima. Antud juhul tähendaks virtuaalne digitaliseerimine seda et originaalpildile asetatakse võre, mille sõlmede vahekaugus on suurem kui üks piksel. Võiks ju arvata et selle tulemusel me lihtsalt kaotame midagi, aga tegelik tulemus on palju "huvitavam". Teatava võre puhul saame tulemuseks sellise pildi, teatava teistsuguse võre puhul aga hoopis sellise pildi.

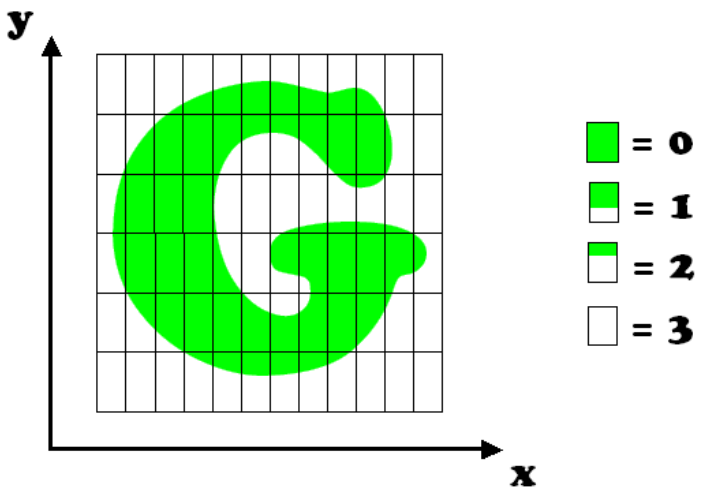


Küsimus: Oletagem et salvestatakse digitaalselt linnulaulu, kasutades mikrofoni, mis on tundlik ka kuulmispiirist kõrgematele sagedustele. Kasutatakse tavalist ja inimkõrva jaoks põhimõtteliselt täiesti piisavat sämplimissagedust 44100 Hz (ajateljelise võre sõlmede vahe on 1/44100 s). Kas ja millisel kujul jääb lindile nahkhiirte kajalokatsioon? **Põhjendada oma vastust joonisega.** Vihje: kajalokatsiooni võiks joonisel kujutada teatava pikalt korduva sinusoidina. Seejärel tuleks saadud sinusoidile asetada (eba)sobiv võre ja vastus on iseenesest käes. Eelnimetatut on üsna hõlbus teha Excel abil. Kõigepealt tekitada sobiv tabel, mille juures tuleks jälgida, et funktsiooni y=sin(ax) kordaja a oleks sobiv vajaliku sageduse tekitamiseks. Seejärel teha tabelist joonis, panna peale sobivad võred ja ongi valmis.

1.2. Eelkirjutatu laiendus digitaalpiltidele

Eelnenud näites oli küll tegemist digitaliseerimisprotsessiga, aga mitte pildi tekitamise protsessiga. Pildi all mõtleme me objekti, kus igal pikslil on x-koordinaat ja y-koordinaat, aga peale selle on sellel pikslil ka heledus või värv. Nagu näha, on pildi puhul tegemist vähemalt kolme koordinaadiga (x, y ja heledus), eelnenud näites oli tegemist ainult kahe koordinaadiga (aeg ja helitugevus). Seega, pildi puhul on digitaliseerimisprotsess selle jagu keerulisem et üks koordinaat tuleb juurde.

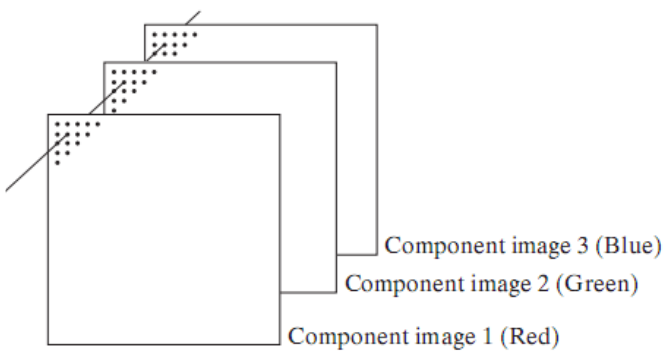
Vaatame teatavat pilti nagu näha all. Lihtsavõitu ta ehk tundub, aga ajab asja ära. Selle pildi



digitaliseerimiseks on tarvis vähemalt kolme võre: x-teljeline, y-teljeline ja heledusteljeline. Paneme nad kõik korraga peale. Digitaliseerimine tähendab tehniliselt nüüd seda et igasse x-ja y-teljeliste võrede poolt tekitatud ristkülikusse (võib ka võrede ristumiskohtade punktidesse, aga ristkülikud näevad rohkem "pikslite" moodi välja) tuleb kirjutada heledusteljelise võre see väärtus, mis kõige paremini iseloomustab selle piksli all olevat kujundit. Antud juhul: kui piksel on täiesti roheline, tuleb kirjutada 0, kui täiesti valge, tuleb kirjutada 3; kui piksli all on nii valget kui ka rohelist, tuleb kirjutada kas 1 või 2, vastavalt sellele kumba (kas rohelist või valget) on rohkem.

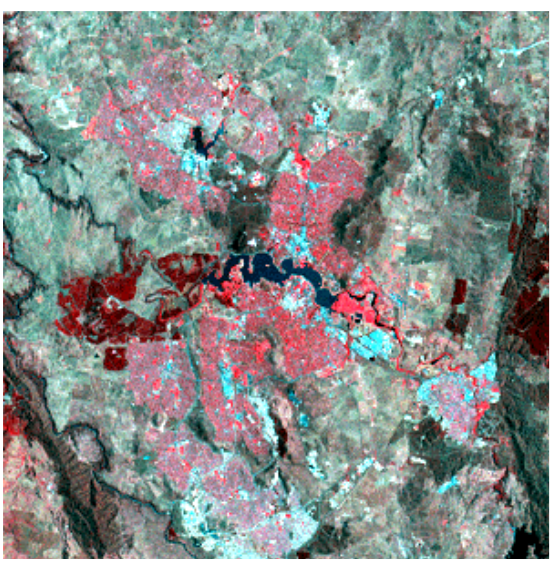
On selge et antud juhul on tulemuseks kõige tavapärasemat sorti arvmaatriks. Ja nii ongi: digipilt on tegelikult maatriks, millest järeldub et mistahes matemaatikat sisaldavad tehted on tegelikult tehted maatriksitega. Ja millest järeldub näiteks ka see et niivõrd kuivõrd Excel sobib maatriksoperatsioonideks, sobib ta ka pilditöötluseks selle matemaatilises mõttes. Tõsi ta on et Excel tulemust pildi kujul ei näita aga pikemate teisenduste puhul ongi tegevuse sisuks kõigepealt arvutused ja alles seejärel pildi näitamine, mistõttu võib neil juhtudel pilditöötluseks sobida just see vahend, mis sobib (maatriks)arvutusteks. Pildi saab pärast ikka kätte, see pole nii suur probleem.

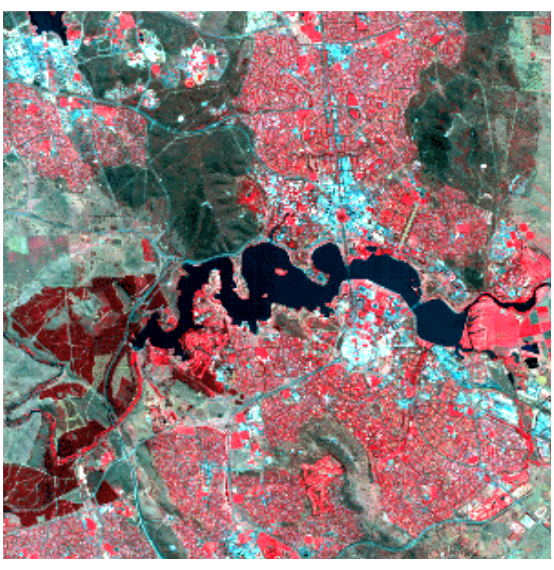
Eespool oli kirjas et tarvis läheb vähemalt kolme võre. See on nii kui tegemist on ühevärvilise pildiga. Iga lisanduv värv lisab veel ühe vajaliku võre. Järelikult, kõige tavalisem kolme primaarvärviga (Red, Green, Blue) pilt vajab aluseks viis võre. Kaks võret määravad pikslite koordinaadid, iga piksliga on seotud kolme värvi kihid. Üldjuhul on kõigi värvide võred samade omadustega (väljendub selles et pildil on kõigis värvikanalites ühepalju piksleid), aga see ei pruugi nii olla. Teatavad pakkimisalgoritmid käsitlevad eri kanaleid erinevalt, ja kaugseirepiltide puhul on üsna sage et eri kanalite loomisel kasutatud võred on erinevad.



Järgnevalt pilditöötluse tekstides sisalduvatest pilti iseloomustavatest mõistetest. Üheks selliseks mõisteks on "lahutused". Eristatakse ruumilist, radiomeetrilist, spektraalset ja ehk veel mõnda lahutust.

Ruumiline lahutus on seotud x-y tasandi maatriksi mõõtmetega, aga mitte sellega identne. Eelnenud "G"-näites võib öelda et ruumiline lahutus on 12 korda 6, see teeb kokku 72 pikslit. Sellest teadmisest võib olla piisavalt juhtudel kui pilt on käsitletav abstraktse objektina, aga mitte juhtudel kui pilti tuleb vaadelda kui tegeliku maailma peegeldust. Viimatinimetatud juhtudel pole primaarne, on pildi suurus 1000 pikslit või mitu megapikslit. Kui oluline on seos tegeliku maailmaga, on primaarne see, mida üks piksel tegelikult peegeldab, ehk milline on pikslile vastav mõõde tegelikus maailmas. Teatavatel juhtudel võib pilt suurusega 100 korda 100 pikslit olla oluliselt kasulikum kui pilt 1000 korda 1000, olenevalt sellest, kuidas on pildistatud ja mida tahetakse pildist tegeliku maailma kohta teada.

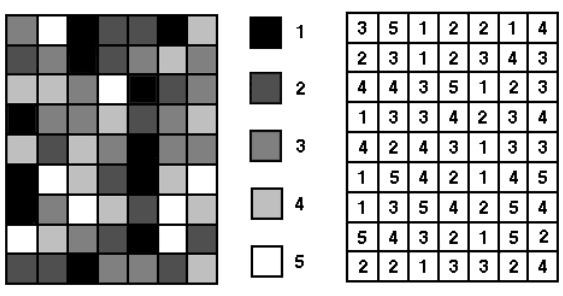




Näide: kaks satelliidipilti. Pikslite arv on mõlemal juhul sama, kuid iga pikslit iseloomustav mõõt ja seega ka ruumiline lahutus on erinevad.

Digitaalpildi **radiomeetriline lahutus** näitab, millist skaalat kasutatakse pildi heledusjaotuse kirjeldamiseks. "G"-näites on radiomeetriliseks lahutuseks 4 (võimalikud on väärtused 0,1,2,3). Kõige tavalisemaks radiomeetriliseks lahutuseks on 256 (sisaldab 256 võimalikku väärtust).

Küsimus: Millised on kõrvaloleva näitepildi ruumilised ja radiomeetrilised lahutused?



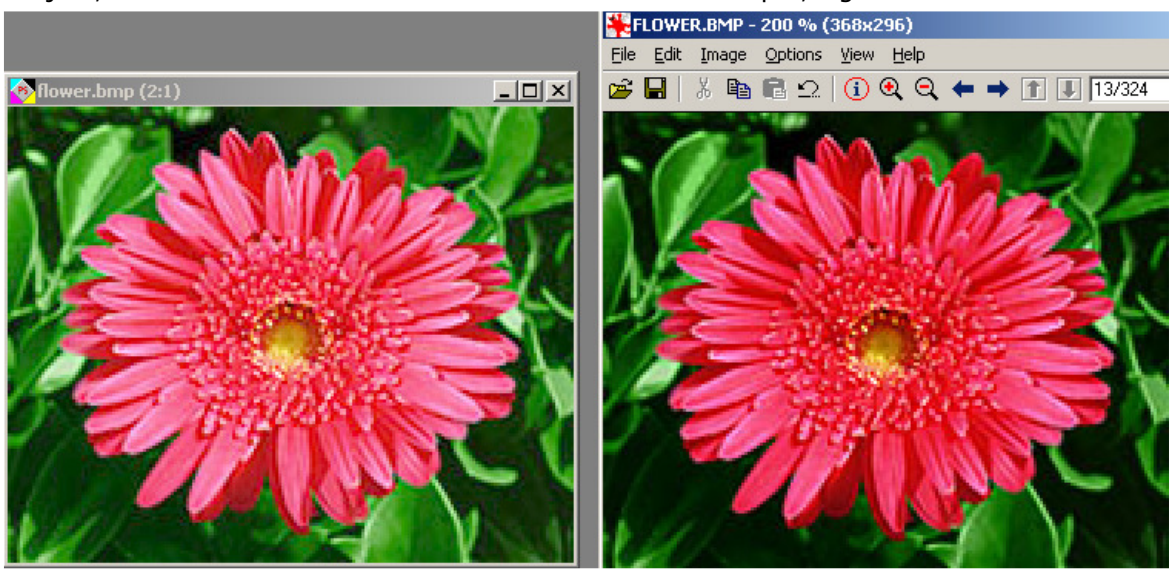
Spektraalne lahutus iseloomustab, millisele spektriosale vastab digitaalpildi mingi kanal ja kuivõrd on pildi erinevad kanalid valgusspektri mõttes eraldatud. Tavafotoaparaat salvestab spektri nähtava osa, siin spektraalsest lahutusest tavaliselt ei räägita. Küll tuleb see mõiste sisse kaugseirepiltide käsitlemisel.

Täiendavaks lugemiseks: õppevahendi 1. peatükk "sissejuhatus pilditöötlusse".

Aare.Luts.1@eesti.ee

2.1. Värvuste ruumid

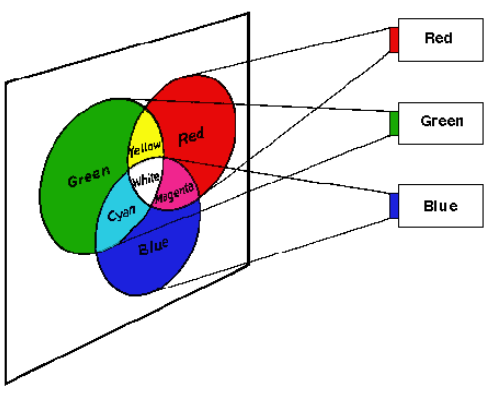
1.1. RGB- ja CMY- ruumid. Värvid ei ole olulised mitte ainult selles mõttes et pilt ilus välja näeks, (õiged) värvid on olulised ka selles mõttes et vajadusel saaks tagada et pildi iga vaataja näeks pilti just neis värvides, mis vaja. Nagu eespool kirjutatud, digitaalpildi aluseks olev(ad) maatriks(id) ei sisalda mitte kusagil infot, kuidas pilt peaks vaatajale tunduma. Maatriksid sisaldavad ainult arve, kuidas aga nende arvude alusel pilt moodustada, see on, vähemalt põhimõtteliselt võttes, iga kasutaja enda asi. Nojah, on olemas teatavad kokkulepped mis peaksid hõlbustama juhtumist sõltumatute sarnaste tulemuste saavutamist. Kui räägitakse RGB-pildist, mõeldakse selle all pilti, kus maatriksi R kihi sisu näib vaatajale teatava punasena ja nii edasi. Ka fotoaparaadid peaksid ära tundma sellesama punase ja kirjutama just sellest punasest tuleneva signaali pildi R kihi sisuks. Tulemuseks peaks olema pilt, mis sarnaneb originaalvaatele. Et asi siiski nii lihtne ei ole ja et nähtav tulemus võib sõltuda veel mitmest (tundmatust) muutujast, selle kohta üks näide. Aluseks on võetud sama pilt, aga seda vaadatakse kursuse



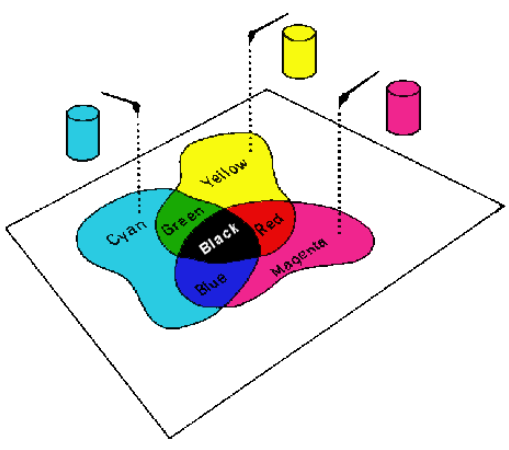
paketti kuuluva kahe erineva programmiga. Mõlema programmi väljundid on monitoril kõrvuti, seetõttu ei saa olla et tulemus sõltub monitorist. Ometigi näivad pildid erinevad. Võib olla küll et mitte väga erinevad, aga erinevus on ometigi märgatav.

Kui juba sama monitori korral on mõningane probleem, kuidas näha pilti just sellisena, "nagu ta on", mis siis veel rääkida erinevatel kandjatel olevatest piltidest. Igapäevaelus võibolla kõige sagedamini esinevaks juhtumiks on ahel "fotokas - arvuti (monitor) - printer". Siin on olukord selles mõttes veel keerulisem et monitor ja printer tekitavad värvused põhimõtteliselt erinevatel viisidel. Kokkuvõttes, tahes-tahtmata on tarvis osata värvusi matemaatiliselt teisendada. Sel juhul võib öelda et värvused teisendatakse ühest ruumist teise. Igal ruumil on omad omadused, nagu ka teatavad piirangud. Ja nagu mistahes matemaatiliselt kirjeldataval ruumil peavad ka siin olema mingid koordinaadid, värvuste ruumi koordinaate sobiks nimetada primaarvärvusteks. Hea värvuste ruum peaks evima vähemalt selliseid omadusi:
1) sisaldama kõiki vajalikke värve;
2) võimaldama matemaatiliselt teisendada vajalikke värve kõigi vajalike erijuhtumite vahel (nt monitor -> printer).

Levinuimad värvuste ruumid on RGB ja CMY. RGB-ruumi primaarvärvusteks on aditiivsed R(ed), G(reen) ja B(lue). Aditiivsus tähendab et kõik teised värvused saadakse primaaride liitumisel. Selle ruumi ajalooliseks realisatsiooniks on CRT-monitorid, kus igale primaarvärvusele vastab oma elektronkiir. Mida suurem on mingi elektronkiire intensiivsus, seda suurem on vastava primaarvärvi heledus. Näiteks, võrdsete koguste R ja G segunemisel saadakse kollane (Y), võrdsete koguste R ja B segunemisel magenta (M). Kui kõiki primaare on võrdselt, saadakse halli erinevad variandid (kui kõigi primaarvärvide intensiivused on minimaalsed, on tulemuseks must, kui primaarvärvuste intensiivsused on maksimaalsed, on tulemuseks valge). Eelnenust järeldub üheselt et pole olemas universaalset "valget" ega ka universaalset "musta", mõlemad sõltuvad monitori omadustest, sellest mis on konkreetse monitori jaoks maksimaalne ja minimaalne heledus.

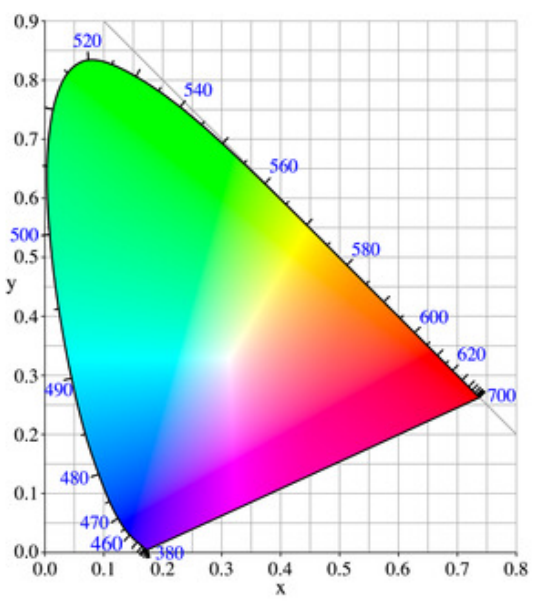


CMY ruumi primaarvärvusteks C(yan), Y(ellow) ja M(agenta). Kuidas need RGB-ruumi primaarvärvuste suhtes paiknevad, seda oli näha juba RGB-ruumi jooniselt. Ka CMY-ruumi jooniselt on näha et näiteks võrdne kogus C ja Y annavad tulemuseks G(reen), mis oli üks RGB-ruumi primaarvärvustest. CMY kasutatakse printerites ja trükitööstuses. Selle ruumi põhimõtteliseks erinevuseks on et ükski primaarvärvustest ei "paista läbi" ja nähtav pilt moodustub mitte värvuste liitumise vaid neeldumiste liitumise tulemusena. C neelab kõiki värvusi peale C, Y neelab kõiki värvusi peale Y. Järelikult, kui nii C, M kui ka Y kokku segada, neelab segu kõiki värvusi ja tulemuseks on must. Teisendusvalem, mille abil saada monitoril nähtav pilt (RGB) printerile sobivasse formaati (CMY), on, vähemalt põhimõtteliselt, vägagi lihtsad:
Cyan = 1 – Red
Magenta = 1 – Green
Y ellow = 1 – Blue
ehk vektorkujul (C,M,Y) = 1 - (R,G,B).



Vastupidine teisendus CMY ruumist RGB ruumi on sama lihtne, vektorvormis esitatuna:
(R,G,B) = 1 – (C,M,Y). RGB ja CMY ruumide eeliseks intuitiivne hoomatavus, nagu ka matemaatiliselt lihtsad teisendusvalemid; puuduseks on asjaolu, et "hea" värvuste ruumi tingimused pole ei RGB ega CMY puhul täielikult rahuldatud.

1.2. CIE- ja HSB- ruumid. Eelnevas käsitletud RGB- ja CMY- ruumide üks otstarve oli selge: olla abiks et piltide väljanägemise mõttes kehtiks olukord WYSIWYG (what you see is what you get). RGB- ruum esitaks osa "what you see" ja CMY-ruum aspekti "what you get". Nendevahelised teisendused on formaalselt lihtsad, aga tegelikult on olukord keerulisem. Kuna tahetakse saavutada WYSIWYG, on aluseks on inimese nägemistaju (omapärad). Järelikult, peaks olema mõttekas võtta aluseks midagi üldisemat, sellist mis kirjeldab inimese võimet tajuda värve. Üldisem ruum peaks olema ka selline et rahuldaks teatavaid matemaatilisi tingimusi, näiteks peaks tema abil olema võimalik teha värvusi puudutavaid arvutusi. Inimese värvustaju mõttes rahuldab tingimusi ruum, mille aluseks on "keskmise" inimese tegeliku värvustajuga seotud "CIE Chromaticity Diagram" (vt joonist). Selle diagrammi kõveral serval asuvad punktid vastavad teatava lainepikkusega monokromaatsetele värvustele; lainepikkused on märgitud serva kõrval. X- ja y-teljed ei esita mitte niivõrd värvusi vaid on matemaatilised objektid mis on ruumis tehtavate arvutuste aluseks. Z-telge pole siin näha. Z-telje eesmärgiks on esitada heledust. Värvusi puudutavaid arvutusreeglid on selles ruumis selged ja lihtsad:



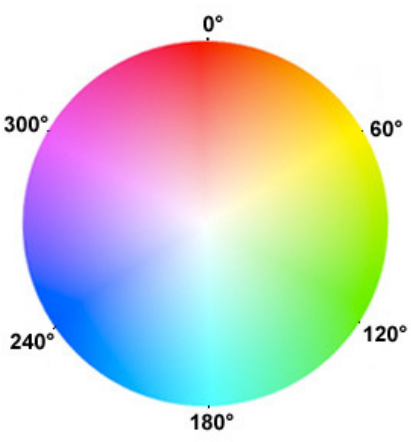
1) Diagrammil võib valida mistahes punktid, kõigil valitud punktidel on teatavad x- ja y- koordinaadid.
2) Valitud punkte võib käsitleda punktidele täpselt vastava värvusega monokromaatsete valgusallikatena. Sel juhul saab nende kahe "valgusallika" abil tekitada kõik need ja ainult need värvused, mis asuvad diagrammil nende punktide vahelisel lõigul.
3) Diagrammil asuva punktide vahele jäävate värvuste koordinaate saab arvutada lineaarse geomeetria reeglite alusel, lähtudes diagrammil valitud punktide x- ja y- koordinaatidest.

CIE XYZ- ruumi eelisteks on asjaolu et ta esitab kõiki "keskmise inimese" poolt tajutavaid värvusi ja esitab neid viisil kus punktide kaudu defineeritavate primaarvärvuste kaudu saab arvutada lineaarsete tehetega kõik värvused, milliseid on võimalik tekitada nende primaarvärvuste abil. Et ruum esitab kõiki nähtavaid värvusi, on kõik sellest ruumist võetud primaarvärvuste abil tekitatavad uued ruumid samuti täielikult nähtavad. Eelpool käsitletud RGB- ja CMY- ruumid ongi XYZ-ruumi mittekattuvad alamhulgad. Kuna nad on mittekattuvad, siis tegelikult eespooltoodud teisendusvalemid alati ei kehti, aga neis valemitest pole seda võimalik välja lugeda. Sellest diagrammist tulenev värvuste ruum on arvutuste aluseks, küll ei ole diagrammil otsest väljundit praktilistesse värviskeemidesse (nt skeemidesse milliseid kasutavad pildigraafika programmid).

Siiski on XYZ-ruumil vähemalt üks puudus: nagu eelnevalt toodud diagrammilt näha, hõlmab siniste toonide piirkond oluliselt väiksema ala kui punaste ja roheliste toonide piirkonnad. Tähendab, ehkki värvuste teisendamise arvutuste mõttes on kõik värvused seotud ühtmoodi lineaarselt, on siniste toonide piirkond geomeetrilises mõttes rohkem kokku pressitud. Rohelises piirkonnas muudab x- või y- koordinaadi muutus värvust vähe, sinises piirkonnas aga muudab koordinaadi arvuliselt samasugune muutus värvust hoopis rohkem. Mõnes mõttes oleks hea kui koordinaadi teatav muutus muudaks värvust alati ligikaudselt ühepalju, sõltumata piirkonnast, kus parajasti asutakse.

Sellele tingimusele vastab paremini **CIE Lab- ruum**, milles on muudetud värvialade suhtelisi laiusi ja on lisatud ka heledus (vt joonist). CIE-Lab ruum omab ka otsest väljundit arvutigraafikasse, mitmed mahukamad programmid võimaldavad valida värvusi just (L, a, b) - koordinaatide kaudu. Paraku, CIE Lab-ruum pole enam lineaarne, seetõttu on temas halvem arvutusi teha.

HSB- ruum. Põhiliselt arvutigraafika vajadusest lähtudes on loodud RGB- ruumiga lineaarselt seotud HSB- ruum. RGB- ruum oli loomulik selles mõttes et ta lähtus värvuste tekitamise tehnilistest seadmetest. Samas, arvutigraafikas võib olla suhteliselt ebamugav ette kujutada, milline osa tuleks võtta mingit primaarvärvust selleks et saada soovitud toon. HSB-ruumis on tooni valik mõnevõrra teistsugune: üks primaarvärvustest (H) tähendab ligikaudu "kohta vikerkaarel" (vt kõrvalolevat joonist). Kui kujutada ette soovitud tooni asukohta vikerkaarel, võib selle asukoha lihtsasti ühe arvuna (mitte R, G, B kombinatsioonina) ette anda. HSB- ruumi S tähendab värvuse küllastatust (100% on puhas värvus), B tähendab heledust. Lisaks arvutigraafikale kasutatakse HSB- ruumi ka mõneks muuks otstarbeks, näiteks piltide pakkimisel.



Loe lisaks:
(värvusruumide kohta) M.David Stone, The Underground Guide to Color Printers, Addison-Wesley, 1996. Üks juba allalaaditud lehekülg: "CIE Color System". Rohkem lisa otsi Internetist, nt CIE ruumi kohta sobiks fraas "CIE Chromaticity Diagram". Üks lehekülgedest, mis selle peale tuleb, on selline.

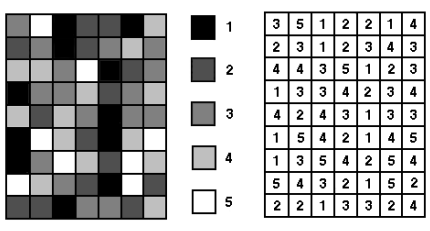
Küsimused:

1) Joonistelt on näha et loomulikul viisil vastavad teineteisele RGB ja CMY ruumid. Tihtipeale räägitakse aga mitte CMY- vaid CMYK- ruumist. Milleks on sel juhul tarvis (ebaloomulikku ?) CMYK ruumi ja kuidas tuleks arvutada K väärtus, teades RGB väärtusi? Kui punkti värvused on antud CMYK ruumis, kuidas leida selle punkti värvusi RGB- ruumis?
2) HSB värvirattal on selgelt näha erinevad värvused. Kus on aga must?
3) Kuidas arvutada punkti HSB koordinaadid, kui on teada RGB koordinaadid? Ja vastupidi?
4) Kui kasutada nt PhotoShop, ilmub liigagi sageli teade "out of color gamut". Mis asi "on väljas" ja millest "ollakse väljas" ja kuidas on see seotud CIE XYZ- ruumiga? Kas see ütleb ehk midagi ka selle kohta, miks RGB ja CMY(K) ruumid ei ole "head" ruumid? Esitada ka vastuseid illustreerivad joonised.

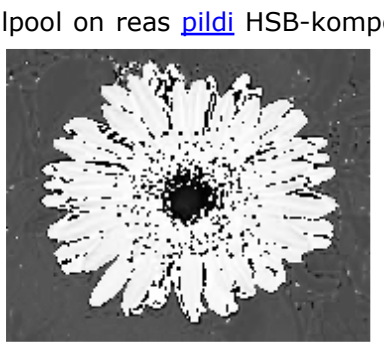
Aare.Luts.1@eesti.ee

2.2. Valevärvused

Eespool oli korduvalt juttu et matemaatilises mõttes on pilt lihtsalt arvmaatriks, selles lähenduses ei ole kusagil kirjas, milline ta peaks pildina välja nägema. Tõsi, oli ka juttu et on olemas teatavad kokkulepped (nt RGB), aga sealsamas kõrval oli ka näide et sama RGB-pildi nähtav kuju võib sõltuda programmist, millega seda pilti vaadatakse. Edasi, käsitledes pilti RGB-ruumis, on meil vähemalt põhimõtteliselt olemas selle ruumi primaarvärvused, mis peaksid ka ise olema ükshaaval pildi kujul vaadeldavad. Eraldades RGB-pildist nt R, G, B-kihid ja vaadeldes neid ükshaaval R, G, B-värvustes, peaks saama üsna tõetruu pildi, kuidas pilt primaarvärvustes välja näeb. Aga mis siis kui tegemist on HSB-ruumiga, või siis CIE Lab-ruumiga? Ka siin on mõlemal juhul kolm primaar"värvust", aga millise nähtava värvi abil peaks neid komponente vaatama?

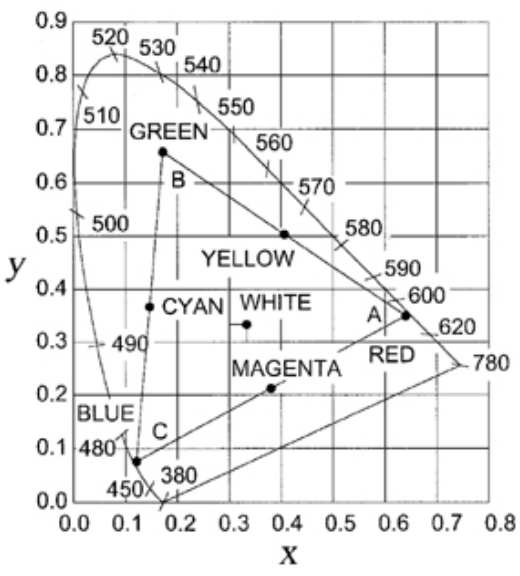


Allpool on reas pildi HSB-komponendid, esitatuna hallskaalas. On ilmne et ei H ega S pole oma



olemuselt hallskaalas, need komponendid on pigemini lihtsalt matemaatilised objektid, millel otsest seost nähtavate värvustega ei ole. Võib ju vaielda et nt H komponent on "koht vikerkaarel", mida peaks ju saama mingi värvusega esitada. Kui aga hakata seda tegelikult esitama, tuleb esitamisel nii ehk teisiti valida vähemalt heledus, aga see ei ole põhimõtteliselt H koosseisus. Niisiis ei ole ka "vikerkaare" esitamisel nähtav päris ja ainult see, mida mõeldakse H sisuna. Seega võib öelda et HSB-komponendid on esitatud mingites kokkuleppelistes värvustes (siin: hallskaalas), mitte neis värvustes mis komponent "ise on".

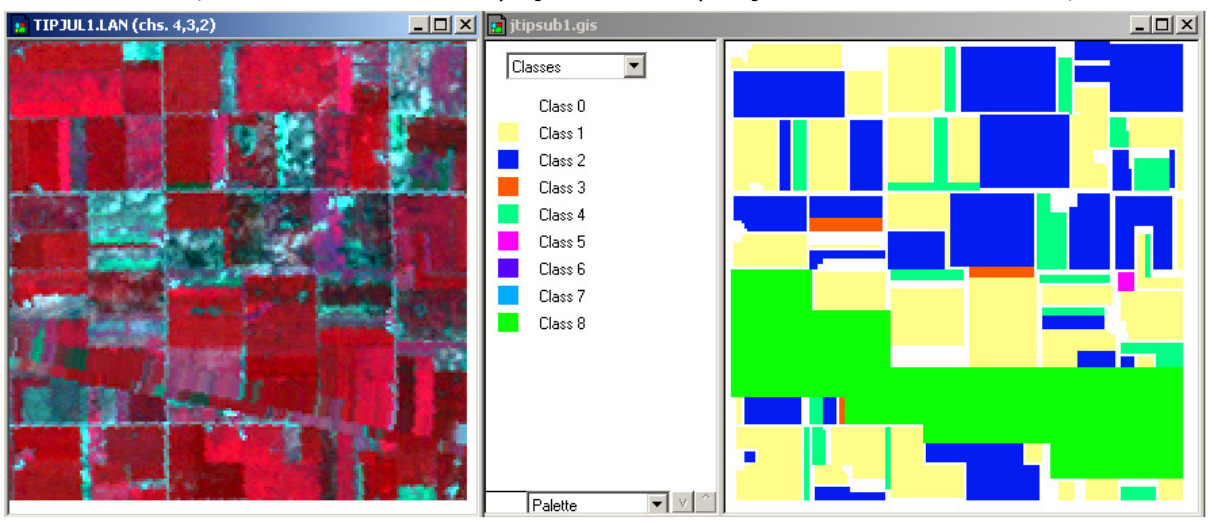
Veel ilmsemaks saab probleem kui vaadata nt CIE XYZ- ruumi x-koordinaati. Diagrammilt on selge et x-koordinaat on teatav, x-telge antud kohas lõikav, vertikaalne sirge. Selle sirge alla jääb hulk värvusi. Kuidas siis tuleks x-koordinaati pildil esitada? Jällegi, seda saab teha kokkuleppelistes värvustes (näiteks: x-koordinaadi nullväärtus esitatakse mustaga, x-koordinaadi maksimaalväärtus esitatakse valgena). Mistahes objekti esitav kokkuleppeline värvus ei ole see mis selle abil esitatav objekt "ise on", esitamisel kasutatud värvust võib nimetada valevärvuseks ja saadud pilti võib nimetada valevärvpildiks. Selles mõttes on ka eespool näitena toodud RGB-pilt mõlemal nähtaval kujul tegelikult valevärvustes, põhjusel et me ei saa olla kindlad, näeme me ikka täpselt neidsamu värvusi mis olid pildistataval objektil.



Valevärvuste kasutamiseks on niisiis vähemalt kaks põhjust: esiteks, me ei tea, on meie nähtavad väärtused need "õiged"; teiseks, teatavad piltide esitamiseks kasutatavad suurused on sellised (nt matemaatilised) et neil ei olegi oma "värvust". Selliseid, "oma värvust" mitte omavaid suurusi on väga palju, levinumatest nt temperatuur või tuule kiirus. On selge et temperatuuril ei ole nähtavat värvust, ometigi tehakse pilte (kaarte), kus tuule erinevad kiirused on märgitud erinevate värvustega.

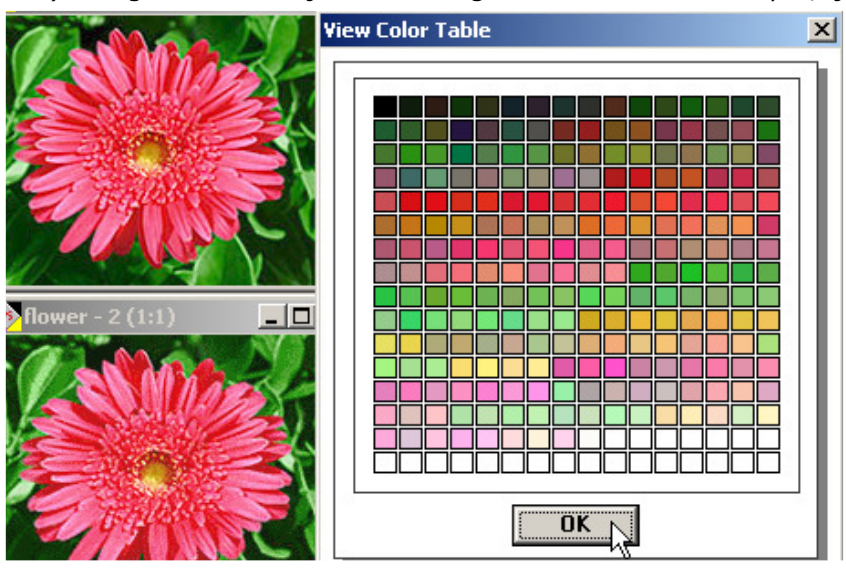
Nende juhtumite kõrval, kus me ei tea, millist värvust kasutada või kus kujutataval objektil tegelikult polegi nähtavat värvust, on väga palju juhtumeid kus kasutatakse teadlikult valesid värvusi. Selliste juhtumite põhjuseks on vajadus eristada või rõhutada mingeid pildil leiduvaid objekte. Inimese nägemistaju suudab eristada suurt arvu erinevaid värvitoone, kuid palju väiksemat arvu erinevaid heledusi (halltoone). Seetõttu, kui on tarvis mahutada pildile palju juba esimese pilguga haaratavat informatsiooni, tuleks halltoonide asemal kasutada värvitoone, ka juhtudel kus pildi aluseks olev info on oma olemuselt pigemini halltoonides (nt röntgenpildid). Värvid on kasulikud ka näiteks juhul, kui on tarvis pildil leiduvat mingit informatsiooni rõhutada. Punane halltoonide taustal mõjub alati rohkem kui veel üks halltoon. Aga ka ainult halltoone sobivalt kasutades saab pildi mingile osale tähelepanu juhtida, nagu kasvõi selles näites, kus on ainult halltoonide sobiva valikuga õnnestunud kaktus "õitsema" panna (kui muidu "õisi" ei näe, suurenda pilti).

Pildi valevärvkuju võib tekitada mitmel viisil. Kõigepealt võib muuta pilti ennast, tehes temaga teisendusi, milliste tulemuseks on uus, soovitud valevärvides pilt. Sellise tegevuse näiteks on kaugseirepiltide klassifitseerimine, kus lähtekohaks on paljukanaliline pilt ja tulemuseks on kaart, kaardil on



huvipakkuvad ja üksteisest eristuvad piirkonnad kujutatud mingite spetsiifiliste värvustega. Või siis ka mõni tavalisem pilt mingitest objektidest, kus näiteks ruudud värvitakse sinisteks, ringid aga kollasteks. Seega võib sedalaadi klassifitseerimisülesandeid formaalselt nimetada valevärvteisendusteks, mis aga lihtsast nimest hoolimata ei tähenda et see teisendus peaks olema triviaalne või muidu kergesti teostatav.

Pildi valevärvkuju võib tekitada ka sel viisil et pilt (pildimaatriksid) jäävad muutumatuteks, aga pildi vaatamisel lisatakse pildile värvifilter, mis muudab pildi nähtavat kuju. Kui originaalpildi primaarvärvideks on R, G, B ja tähistame uute (vale)värvide vektori VV, võib kirjutada
VV=F(R,G,B).
Arvutigraafikas seatakse sellisel juhul pildimaatriksite ja pildi nähtava kuju vahele värvuste tabel, kus igale esialgsele värvusele seatakse vastavusse uus värvus. Pilditöötlusprogrammides on tabeli lisamise tehte nimeks tavaliselt "Convert to paletted" ja lisanduvaks objektiks on "color table" (näide allpool oleval pildil). Kõige lihtsamal juhul on originaaliks halltoonides pilt, ja tulemuseks valevärvides pilt, nt punased toonid halltoonide taustal. Sel juhul on teisendustabeli pikkuseks peaaegu alati 256 elementi, tingitult pildil üldjuhul olevast 256 halltoonist, millistest igaühele tuleb seada vastavusse mingi soovitud värvus. Pildile värvifiltri lisamise eeliseks võib lugeda asjaolu et sel juhul võib tekitada palju erinevaid filtreid, näiteks igaüks mingi konkreetse objekti või konkreetse piirkonna esiletõstmiseks, ja filtrite rakendamine on tavaliselt palju kiirem kui pildi enda muutmine. Jällegi, sobivate filtrite koostamine ei pruugi olla kaugeltki lihtne ülesanne.



Küsimused:

1) Oletagem et pildil, mida tahetakse vaadata kasutades valevärve, saab lugeda kokku rohkem kui 256 värvitooni. Millised oleksid sel juhul variandid teisendustabeli moodustamiseks?

2) Oli juttu et üldjuhul on mistahes pilt valevärvpilt. Kuidas tagada et pildil (nt digifotol) oleksid ikka needsamad värvused mis olid pildistatud objektil?

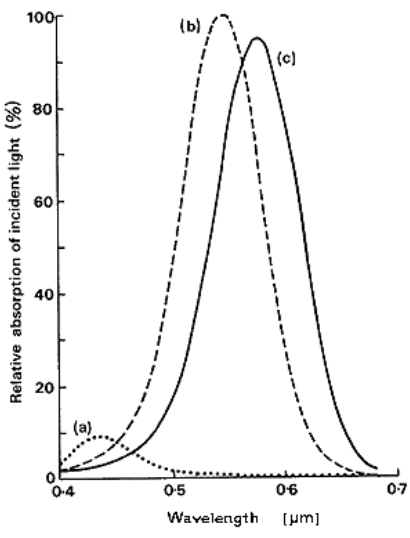
3) Eespool oli kirjas valevärvpildi tekitamise funktsioon koos argumentidega F=F(R,G,B) ehk et funktsiooni väärtus sõltub piksli originaalvärvide kombinatsioonist. Eespool oli ka juttu et valevärvpildi näiteks võiks olla "värvime pildil olevad ruudud sinisteks, ringid aga kollasteks". Kas sellisel kujul funktsiooni saab kasutada eelnimetatud valevärvpildi tekitamiseks? Kui miskit tuleks muuta, siis mida? Või on tarvis mingeid eritingimusi? Pane tähele et vastuseid stiilis "tuleb kirjutada sobiv ja piisavalt keeruline funktsioon" ei saa lugeda õigeks, vähemalt üks tingimus on hoopis olulisem.

Aare.Luts.1@eesti.ee

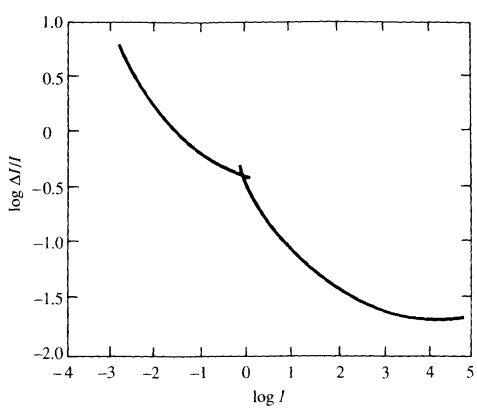
3.1. Silma kui füüsikalise vastuvõtja omadusi

Inimese pilditaju ei ole füüsikalises mõttes kaugeltki täiuslik, kuid taju on ikkagi suuteline nähtu alusel olulisi järeldusi tegema. Enamgi, kui hinnatakse pilditöötlusmeetodite võimsust, võetakse tihtipeale võrdluse aluseks just inimese pilditaju. Nii mitmeski aspektis on pilditaju ikka veel võimsam kui analoogilised arvutusmeetodid. Kui mõnes aspektis tundub pilditaju valetavat, on ka nende aspektide uurimine tähtis sest tihtipeale võib just ekslik taju anda vihjeid, kuidas nägemismeel loogilises mõttes töötab.

Värv moodustub ajus silma 3 liiki retseptorite signaalide koosmõjus. Retseptorite tundlikkuse kõverad on toodud kõrvaloleval joonisel. Kõverad on üksteisest eristuvad ja kõverate tipud (maksimaalsele tundlikkusele vastavad lainepikkused) on erinevas kohas. Esimese kõvera (a) tipp asub sinises piirkonnas (445 nm), teise (b) tipp rohelise kollasepoolses osas (535 nm) ja kolmanda (c) tipp kollase punasepoolses osas (575 nm). Seega tundub RGB- ruum üsna mõistlik, ka inimsilma "primaarvärvid" on mõnes mõttes sarnased. Küll on sinist valgust eelistavad retseptorid kordades nõrgema tundlikkusega kui teised, sellest võib oletada et ehk on tõsi taga arvamusel et veel mõni tuhat aastat tagasi inimesed sinist värvust üldse ei näinud ja retseptorid on alles suhteliselt hiljaaegu arenema hakanud. Igatahes ei ole tundlikkuse kõverad füüsikalises mõttes sugugi ideaalsed: esiteks on eri liiki retseptorite tundlikkused väga erinevad ja teiseks ei ole kõverate tipud jaotunud lainepikkuste järgi ühtlaselt. Värvusele reageerivaid retseptoreid on silmas umbes 8 miljonit, mis jääb ka juba alla digikaamerate pikslite arvule.



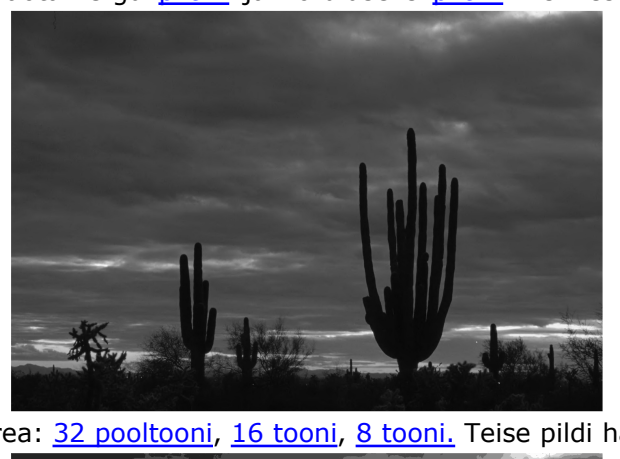
Kui silmas olevate retseptorite arv on tehniliste seadmetega võrreldes veel enamvähem, siis halvemini on lugu halltoonide eristamise võimega. Igaüks teab et nägemisvõime sõltub vaadatava objekti valgustatusest. Objekti heledusest sõltub ka objektil leiduvate halltoonide eristamise võime (vt kõrvalolevat joonist). Selle joonise x-teljel on heledus, seda logaritmilises skaalas, ehk valgusenergia mõttes on "0" ja "5" erinevus sada tuhat korda. Y-teljel on suurus "heleduse muutus jagatud heledusega", mis on sisuliselt pooltoonide arv. Kuna y-telje skaala on logaritmiline, vastab koht "-1.0" kümnele pooltoonile, koht "-2.0" vastab sajale pooltoonile. Kõver ise näitab konkreetsele heledusele adapteerunud silma maksimaalset eristusvõimet. Nagu näha, kui heledus on väga väike (x-telje negatiivsete väärtuste piirkond), silm pooltoone eristada tegelikult ei suuda. Kui objekti heledus suureneb, silma eristamisvõime paraneb, kuid teatava heleduse juures saavutatakse maksimum, sellest suurema heleduse juures hakkab pooltoonide eristamise võime jällegi vähenema, kõveral avaldub see niimoodi et kõver hakkab tagasi ülespoole tulema. Optimaalse heleduse korral suudab silm eristada umbes 40 halltooni (heleduse muutus jagatud heledusega on logaritmilises skaalas ligikaudu võrdne -1.6), mis on oluliselt vähem kui tavaline skänner suudab. Hoolimata "keskpärastest" füüsikalistest karakteristikutest suudab nägemismeel nii üht kui ka teist.



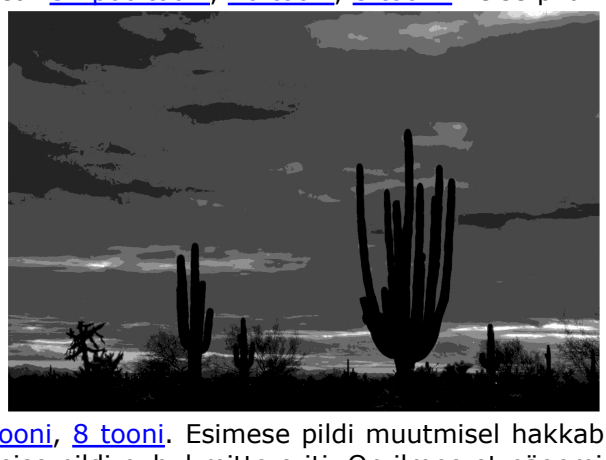
Küsimus eelneva joonise kohta: Oletagem et vaadeldava objekti heledus on 1 (see tähendab et x-telje koordinaat *log I* =0). Mitut pooltooni on selle joonise järgi sellele heledusele adapteerunud "keskmine silm" võimeline vaadeldaval objektil nägema?

3.2. Pooltoonide subjektiivne tähtus

On huvitav et nägemismeele jaoks oluliste pooltoonide arv sõltub pildi iseloomust. Teatavat liiki piltidel märkab nägemismeel kohe, kui pildil pooltoone vähemaks võtta, teatavat liiki piltidel võib aga pooltoonide arvu üsna julgesti vähendada, pilt selle juures oma subjektiivset kvaliteeti ei kaota. Olgu pilt 1 ja võrdluseks pilt 2. Esimese pildi halltoonide arvu vähendamisel tekitame



rea: 32 pooltooni, 16 tooni, 8 tooni. Teise pildi halltoonide arvu vähendamisel kasutame rida: 16

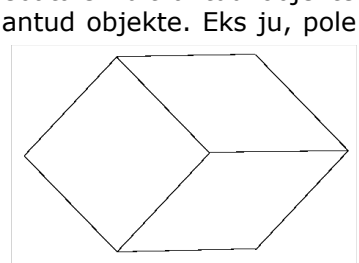


tooni, 8 tooni. Esimese pildi muutmisel hakkab ka pildi subjektiivne kvaliteet kiiresti muutuma, teise pildi puhul mitte eriti. On ilmne et nägemistaju sellist iseärasust saab edukalt ära kasutada näiteks piltide pakkimisel, kui ainult õnnestub mingil viisil automaatselt määrata, mis liigist parajasti pakitav pilt on.

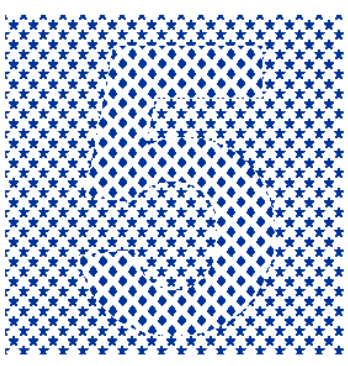
3.3. Pilditaju muud iseärasused

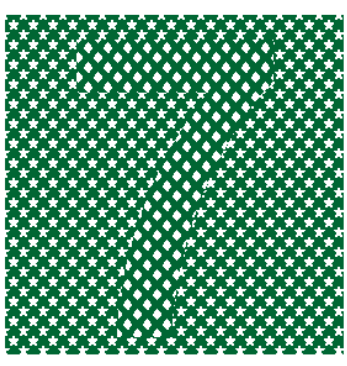
Mõned pildid valesti tajumise juhtumitest on siinsamas (püsti-pikali jooned ja halltoonides

ristkülikud). Piltide uurimise jätaks iseseisvaks tööks koos **küsimusega et mis siin õigupoolest valesti tundub?** Olgu lisatud et teine näide on üks sellitest mis võib öelda midagi nägemismeele toimimise mehhanismi kohta, selleni jõuame edaspidi, kui käsitleme vastavaid tehteid. Valesti tajumise kõrvale ka mõned näited tajuaparaadi võimest tunda ära ebatäielikult antud objekte ja erineva mustriga antud objekte. Eks ju, pole probleemi ära näha,



mis on piltidel. Sama asja matemaatika abil lahendada on hoopistükkis keerulisem. Võtame kasvõi esimese joonise. Kui temal olevat objekti kaua vaadata, hakkab iseendalgi tekkima kahtlus et milline objekt ikkagi pildil on. Veel mõned nägemistaju iseärasused on vaadeldavad siin viidatud failis.





Ülesanne: otsida veel nägemistaju (illusioonide) näiteid. Eespool oli kirjas et üks toodud näidetest võib öelda midagi nägemistaju mehhanismide kohta, ja et täpsemalt selle kohta edaspidi. Aga äkki leiab keegi juba üles, mis mehhanismist on jutt?

TÄIENDAVAT KIRJANDUST

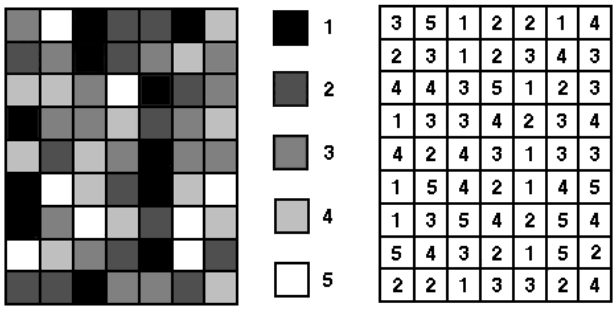
(värviformaatide kohta) M.David Stone, The Underground Guide to Color Printers, Addison-Wesley, 1996.
(pilditaju iseärasuste kohta) Jüri Allik, Aavo Luuk, Nägemispsühholoogia. Tallinn, 1980.

Kirjandus internetist. Soovitan toksida sisse mõni otsingufraas, nt CIE värviruumi kohta sobiks fraas "CIE Chromaticity Diagram", valevärvuste kohta sobiksid fraasid "false color" või "false colour". Mõned silmahakanud leheküegedest:
RGB and CIE XYZ Systems.
CIELAB Color Space.
Color Addition.
Color Subtraction.

Aare.Luts.1@eesti.ee

Teema 1 küsimused on järgmised:

1. Oletagem et salvestatakse digitaalselt linnulaulu, kasutades mikrofoni, mis on tundlik ka kuulmispiirist kõrgematele sagedustele. Kasutatakse tavalist ja inimkõrva jaoks põhimõtteliselt täiesti piisavat sämplimissagedust 44100 Hz (ajateljelise võre sõlmede vahe on 1/44100 s). Kas ja millisel kujul jääb lindile nahkhiirte kajalokatsioon? **Põhjendada oma vastust joonisega.** Vihje: kajalokatsiooni võiks joonisel kujutada teatava pikalt korduva sinusoidina. Seejärel tuleks saadud sinusoidile asetada (eba)sobiv võre ja vastus on iseenesest käes. Eelnimetatut on üsna hõlbus teha Excel abil. Kõigepealt tekitada sobiv tabel, mille juures tuleks jälgida, et funktsiooni y=sin(ax) kordaja a oleks sobiv vajaliku sageduse tekitamiseks. Seejärel teha tabelist joonis, panna peale sobivad võred ja ongi valmis.



2. Millised on kõrvaloleva näitepildi ruumilised ja radiomeetrilised lahutused? Kuidas tuleks joonist muuta (võimalusi on mitu) et radiomeetriline lahutus oleks näiteks 100?

3. Konspekti joonistelt on näha et loomulikul viisil vastavad teineteisele RGB ja CMY ruumid. Tihtipeale räägitakse aga mitte CMY- vaid CMYK- ruumist. Milleks on sel juhul tarvis (ebaloomulikku ?) CMYK ruumi ja kuidas tuleks arvutada K väärtus, teades RGB väärtusi? Kui punkti värvused on antud CMYK ruumis, kuidas leida selle punkti värvusi RGB- ruumis?
4. HSB värvirattal on selgelt näha erinevad värvused. Kus on aga must?
5. Kuidas arvutada punkti HSB koordinaadid, kui on teada RGB koordinaadid? Ja vastupidi?
6. Kui kasutada nt PhotoShop, ilmub liigagi sageli teade "out of color gamut". Mis asi "on väljas" ja millest "ollakse väljas" ja kuidas on see seotud CIE XYZ- ruumiga? Kas see ütleb ehk midagi ka selle kohta, miks RGB ja CMY(K) ruumid ei ole "head" ruumid? Esitada ka vastuseid illustreerivad joonised.

7. Oletagem et pildil, mida tahetakse vaadata kasutades valevärve, saab lugeda kokku rohkem kui 256 värvitooni. Millised oleksid sel juhul variandid teisendustabeli moodustamiseks?

8. Oli juttu et üldjuhul on mistahes pilt valevärvpilt. Kuidas tagada et pildil (nt digifotol) oleksid ikka needsamad värvused mis olid pildistatud objektil?

9. Konspektis oli kirjas valevärvpildi tekitamise funktsioon koos argumentidega F=F(R,G,B) ehk et funktsiooni väärtus sõltub piksli originaalvärvide kombinatsioonist. Eespool oli ka juttu et valevärvpildi näiteks võiks olla "värvime pildil olevad ruudud sinisteks, ringid aga kollasteks". Kas sellisel kujul funktsiooni saab kasutada eelnimetatud valevärvpildi tekitamiseks? Kui miskit tuleks muuta, siis mida? Või on tarvis mingeid eritingimusi?

Pane tähele et vastuseid stiilis "tuleb kirjutada sobiv ja piisavalt keeruline funktsioon" ei saa lugeda õigeks, vähemalt üks tingimus on hoopis olulisem.

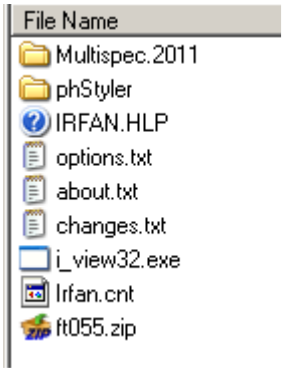
10. Otsida veel nägemistaju (illusioonide) näiteid.

Vastused arutage rühmas (rühmafoorumis) läbi, rühma koostöös vormistage vastused leheküljele "1.teema ühiste vastuste Wiki".

Kursusel kasutatava tarkvara iseloomustus.

Kursuse esimeseks eesmärgiks on õpetada meetodeid. Sellega erineb ta mitmest teisest, kus kõigepealt õpetatakse konkreetset tarkvara, ja alles seejärel seda, mida tarkvaraga teha saab. Keskendumine konkreetsele tarkvarale on ehk halb mitmes aspektis. Esiteks, kui kasutatud vahendid pole käepärast, ei oskagi nagu midagi teha. Teiseks, sedalaadi vahendid maksavad tihtipeale palju, ja pole sugugi hea mõte esitada tööandjale tingimus "kõigepealt tahan ma seda programmi". Selles kursuses seatakse esiplaanile meetodid, ja seejärel otsitakse võimalikult lihtsaid ja laialt levinud vahendeid, millega meetodeid katsetada. Kõik pakutavad vahendid pole kohustuslikud, aga üks võib täiendada teist. Ja veel: kui mõnel osalejal on juba välja kujunenud oma lemmikvahend, ei hakata sugugi sundima et ta peaks sellest loobuma. Sel juhul võib lahendada kõik ülesanded, kasutades oma lemmiktarkvara, eeldusel et see tarkvara võimaldab ülesandeid etteantud viisil lahendada. Sel juhul on tingimuseks et tulemused tuleb esitada kujul, mis ei nõua lugemiseks muud tarkvara kui kursusel üldiselt kasutatakse.



Niisiis, mida sellel kursusel pakutakse kasutada?

1) Kursuse tarbeks koostatud Java-pakett. Pakett asub pakituna failis "ft055.zip". Sellel on mitu head omadust.
Kõigepealt, ta on vabalt täiendatav, kusjuures selleks pole tarvis hakata ära õppima mingit spetsiifilist keelt, täiesti piisab kui osata pea kõigis programeerimiskeskkondades kasutatavat ühiskeelt. Siis, täiendamiseks on juba valmis pakett, mis on võimeline lugema ja kirjutama .bmp-pilte, nagu ka lugema ja kirjutama arvmaatrikseid, see tähendab et kasutajal pole tarvis selle pärast enam aega kulutada. Kolmandaks, paketil on üsna haruldane omadus, nimelt seostab ta arvmaatriksid ja pildid. Kui meenutada klassikalisi pilditöötluspakette, siis ka kõige võimsamad neist (PhotoShop, Adobe Illustrator, Corel Photopaint) ei oska arvmaatriksitega midagi peale hakata. Jah, näiteks Mathcad oskab, aga temas programmeerimiseks on tarvis eraldi harjutamist, ja, kogemused näitavad et MathCad on selles osas ikka väga aeglane.

2) IrfanView. Pakett asub juurkataloogis. Nagu mainitud, Java-pakett loeb/kirjutab ainult .bmp-pilte. See on tingitud vajadusest hoida pakett võimalikult lihtne, ja pole ka suuremat vajadust hakata pildiformaatide konvertereid veel kord sisse kirjutama, seda olukorras, kus on nii palju vabaprogramme, mis käsitlevad paljusid formaate. IrfanView on üks neist. Tema üheks kasutusvaldkonnaks on konverteerida pilte, aga ka pilte kiirkorras vaadata, sest ta teeb seda nii kiiresti ja on nii väike.

3) PhotoStyler. Asub kataloogis "phStyler". Üks näidetest, et kolmkümmend aastat vana programm võistleb mitmes aspektis edukalt kaasaegsetega, ja on selle juures nii väike. Tema põhiliseks kasutusvaldkonnaks on lihtsamad pilditehted. Piiranguks võib olla asjaolu, et tema käsitletavate formaatide nimekiri on suhteliselt lühike. Tegelikult pole see takistus, kui sealsamas on olemas IrvanView.

4) MultiSpec. Asub kataloogis "Multispec.2011". Jälle näide, et mingi ülesande lahendamiseks pole alati tarvis osta seda kõige kallimat (antud juhul: Idrisi). Vaba ja väga kiire programm, mida kasutatakse kaugseirepiltide analüüsil. Et tema vorm (vaba, väike ja kiire) ei vasta sisule, sellest

saab aimu, kui vaadata sisalduvate meetodite nimekirja või siis aluseks olevate teadustööde nimekirja.

5) MathCad. Tema vaieldamatuks eeliseks on sisalduvad matemaatilised meetodid. Pealegi, Tartu Ülikoolis on ta vaba. Vajadusel pöörduda arvutivõrgu mõne administraatori poole. Paraku, nähtud näited kipuvad ikka ja jälle tõestama, et pilditöötluseks ta ei sobi, kasvõi oma aegluse tõttu. Aga on kasutajaid, kes kõigest hoolimata teevad kõik Mathcad-s. Pole mingit plaani hakata neid kasutajaid ümber veenma, kursuse ülesandeid saab teha ka MathCad-s ja seda võibki teha. Ülejäänud kasutajate jaoks on MathCad ülesandeks abistada matemaatilistel arvutustel.

6) Excel. Kõlab imelikult? Aga, tegelikult saab ka Excel-s nii üht kui ka teist arvutada, ja kuna siinsamas kõrval on Java-pakett, mis loeb arvmaatrikseid, saab tulemused seostada piltidega. Kui on Windows, on arvutis tavaliselt ka Excel.

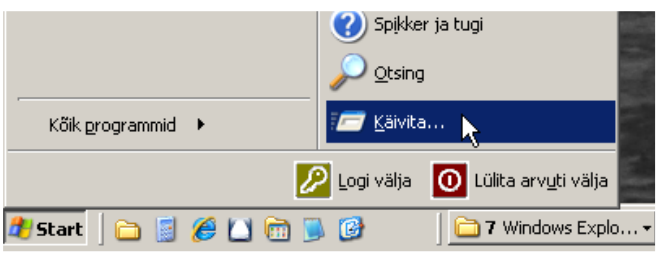
Nimetatud pakettide oskuslikul kombineerimisel saab lahendada üsna mitmeid praktilise pilditöötluse ülesandeid. Ja enamgi, kuna üheka vahendiks on Java-pakett, saab tulemused jäädavalt vormistada ja neid ka edaspidi nii kasutada kui ka edasi arendada.

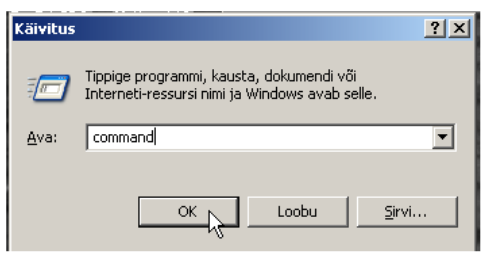
Kursuse Java-paketi juhend

Alljärgnev eeldab et kasutaja arvutis on op-süsteemiks Windows. Arvata on et spetsiaals kursuse jaoks tehtud Java-pakett töötab ka muudes süsteemides, nagu on olemas ka muu süsteemides siin pakutavatele analoogilisi vabavara pilditöötlusvahendeid, aga kumbagi pole ise proovitud ja seetõttu ei lubata muude op-süsteemide korral midagi, ja selle juhu jaoks juhiseid anta.

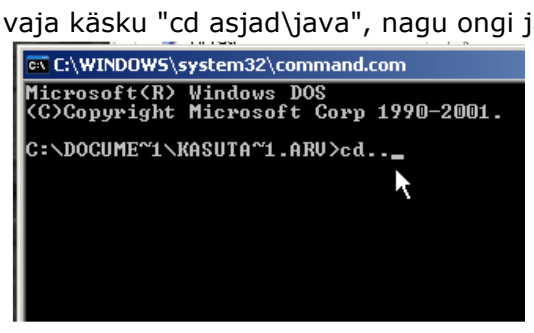
Java keskkonna tekitamiseks tuleb kõigepealt käesolev pakett lahti pakkida, kusjuures selle pal paigutamiseks **on väga soovitav kasutada lihtsate nimedega kataloge**. Kataloogid " Documents" või "Program Files" või nende alamkataloogid (kuhu Windows kipub kõik asjad paner ei sobi mitte kuidagi. Esiteks tuleb vältida tühikutega kataloogi- ja failinimesid, kuna Javal omapära mõnel juhul tühikuid mitte tunnistada. Teiseks on lihtsad nimed head selle poolest vajadusel on neid hea sisse tippida. Paketi autor kasutab nt kataloogi C:\asjad\java. Lihts failinimede nõue üldiselt ei kehti Java süsteemifailide jaoks, see nõue kehtib ainult selle paketi puh

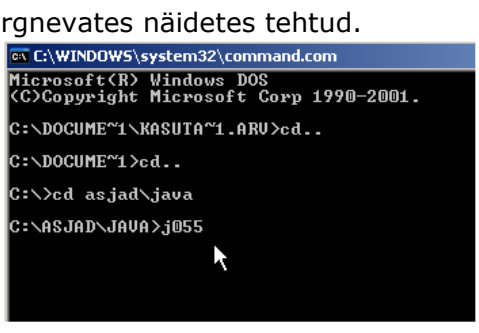
Java-kasutuskeskkonna käivitamiseks tuleb esmalt veenduda et masinas oleks sobiv JDK (Ja Development Kit), paketi tegemiseks on kasutatud versiooni 1.3. On kogemusi et pakett töötab Java uuemate versioonidega, aga kindluse mõttes võiks võtta versiooni 1.3.
Kui on, võib seda lõiku edasi mitte lugeda.
Kas on, saab kontrollida alljärgneval viisil:
1) Avada Dos-aken (nt viisil Run "Command").





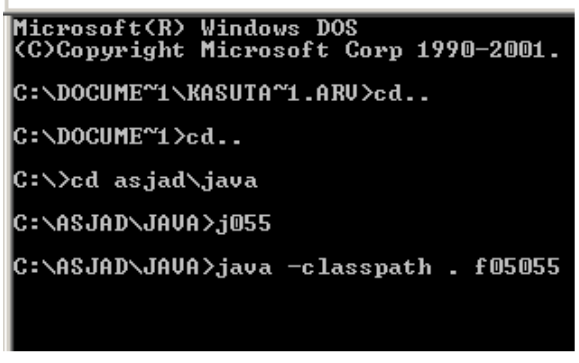
Seejärel liikuda kataloogi kus kursuse Java-keskkond asub (käsk "cd <kataloogi nimi>"). Neile, pole DOS-ga kokku puutunud, veel nii palju et kataloogis ülespoole saab käsuga "cd ..", kataloc allapoole, nagu öeldud, saab käsuga cd <kataloogi nimi>, nt kui hetkel asutakse kataloogis c:\, vaja käsku "cd asjad\java", nagu ongi järgnevates näidetes tehtud.



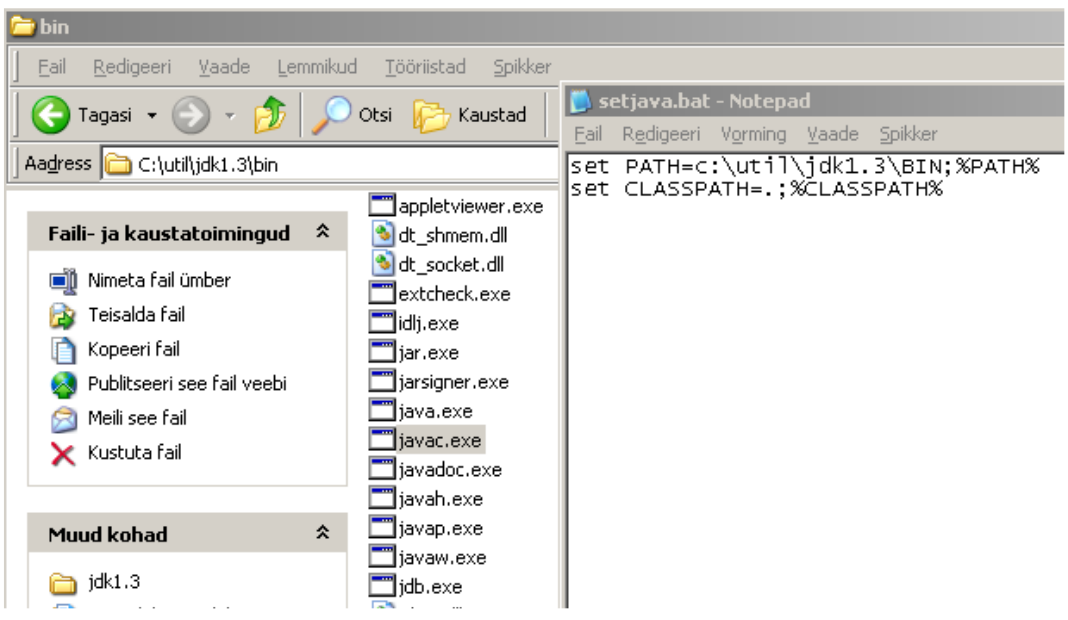


2) Kui DOS-aken avatud, **anda Dos-aknas käsk "j055".** Peaks avanema kursuse "05055 aken" (nagu pildil). Kui avaneb, on lootust. Kui ei avane, ja kataloog on õige (tõepoolest, kursuse java-pakett sai lahti pakitud just sellesse kataloogi), lugeda kohe allpool toodud lõiku.

3) Panna kursuse aken kinni. Anda käsk "c055". Kui veateadet ei tule, on Java konfiguratsiooniga tõenäoselt kõik korras, ja võib järgmise lõigu (lõik 4) vahele jätta. Edukus tähendab et paketti saab mitte ainult käivitada, vaid ka muuta. Proovida uuesti käsku "j055". Kui programmi aken nüüd ei avane, on tegemist juhtumiga kus kompileeris, aga ei käivitu. Põhjus võib olla selles et oled paketi sisu omapäi muutnud (näiteks midagi kustutanud). Mõtle järele, kui muu ei aita, otsi abi. Kui aga "c055" peale tuli veateade, on JDK konfiguratsioon ebasobiv. Sel juhul lugeda järgnevat lõiku.



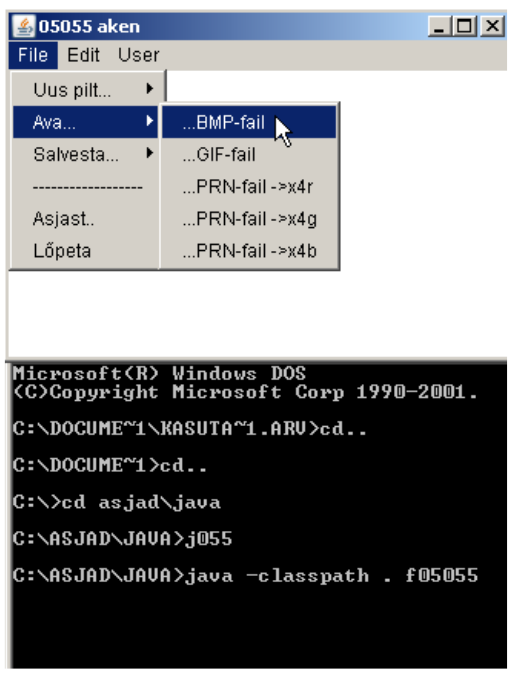
4) Kui JDK masinas ei ole, või ei ole ta sobival kujul (järeldub tõsiasjast et "c055" ja/või "j055" ei töötanud), tuleb JDK kas masinasse panna, sobivaks konfigureerida. Java Development Kit paigalduspaketi saab kasvõi kursuse koduleheküljpaketi saab paigaldada nagu seda tehakse mistahes muu programmiga. **Siiski, enne pak**

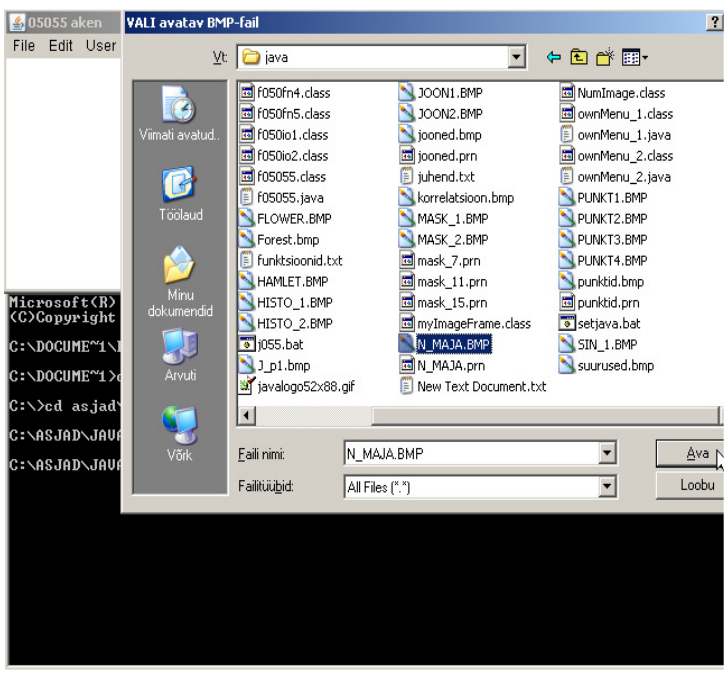


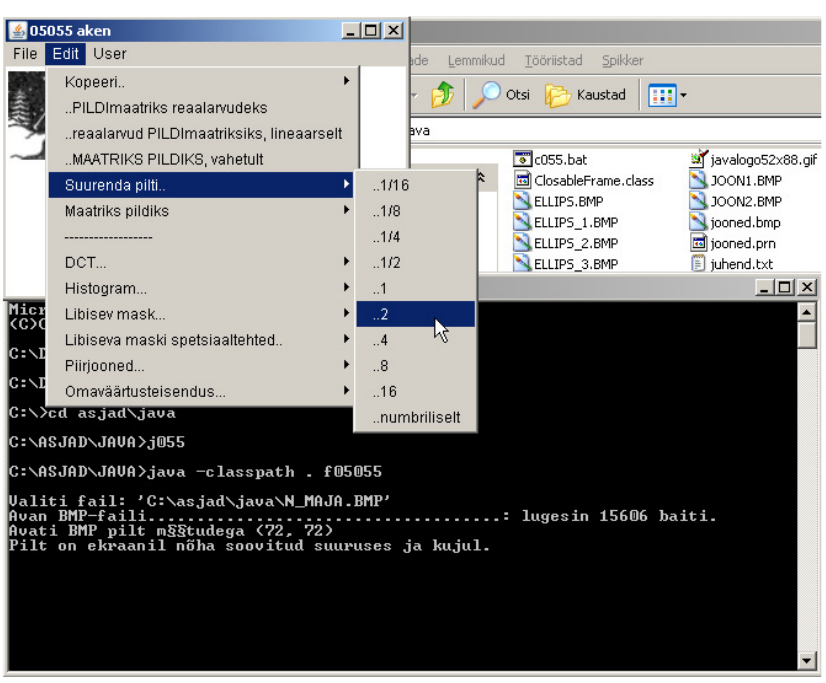
laadimist paigaldamist tuleks veendu et JDK masi tõepoolest ei võib juhtuda pakett on olem aga seda ei l üles. Kui n uuesti paigalda võib tulla jama. veenduda, tul otsida arvutist faili "javac.ex milline peab oler olema, ja asu Java Development Kit alamkataloogis "bin". Kui sellist faili ei ole, võib julgesti asuda JDK paigaldar kui fail aga on, tuleks (esmalt) proovida konfigureerida. Konfigureerimiseks tuleb muuta kurs kataloogis asuvat faili 'setjava.bat', seda vastavalt sellele kus kataloogis JDK tegelikult asub, pak pakutud 'setjava.bat' arvab et JDK kataloog on 'C:\util\jdk1.3\BIN'. Asendada see nimi tegel kataloogi nimega. Seejärel anda Dos-aknas käsk "setjava" (sellega seatakse Java katal aktiivseks), misjärel proovida uuesti, kas käsud "j055" ja "c055" töötavad. Kui töötavad, on korra Kui ikka ei tööta, tuleb JDK arvatavasti uuesti paigaldada (ja vaadata, kuhu ta sai paigaldatud), võibolla tuleb ka abi otsida (nt kirjutada õppejõule).

Nüüd spetsiaalselt kursuse jaoks loodud paketist.

Pakutav pakett võimaldab lugeda ja salvestada .BMP TrueColor pilte. Lisaks failide lugemisele kirjutamisele saab paketis teha süsteemseid tehteid, missuguseid nimelt, seda on näha menüüde üksikasjalikumalt on pakutavaid funktsioone on kirjeldatud failis 'funktsioonid.txt'. Jägnevad pil illustreerivad, kuidas käib .BMP-faili avamine ja kuidas tulemus välja näeb.

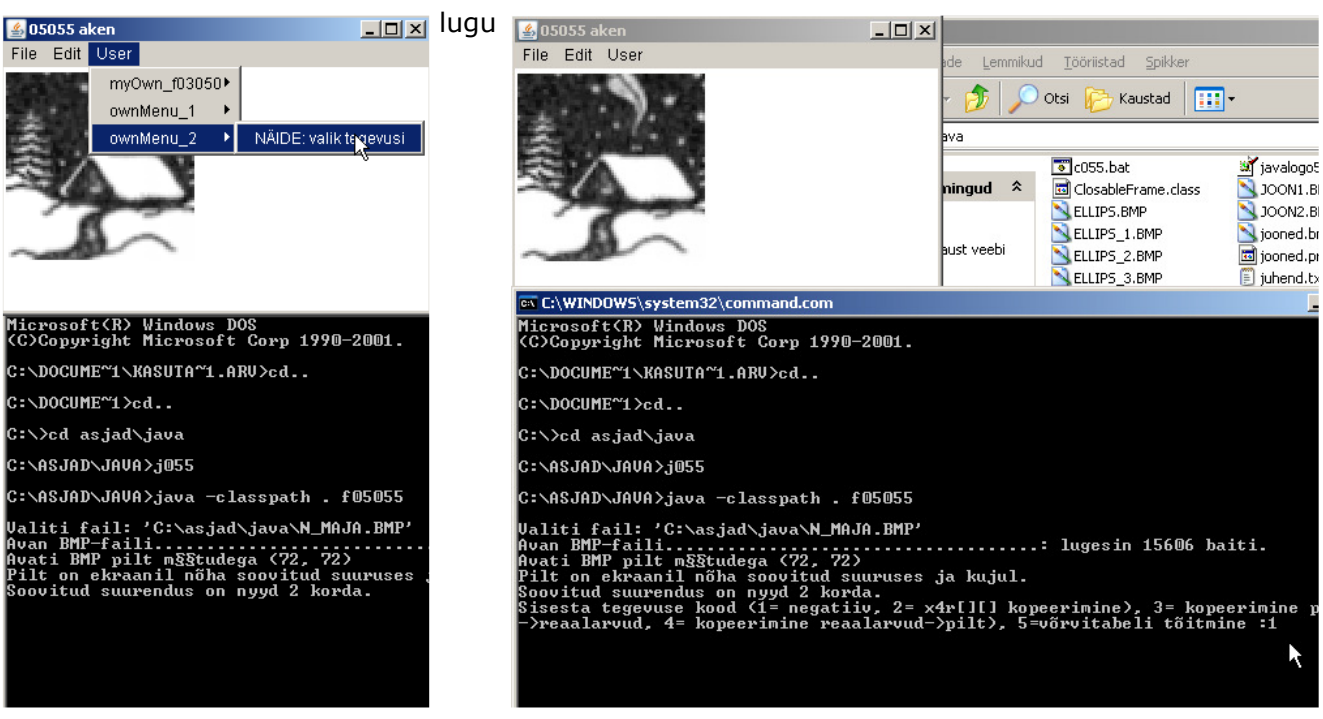




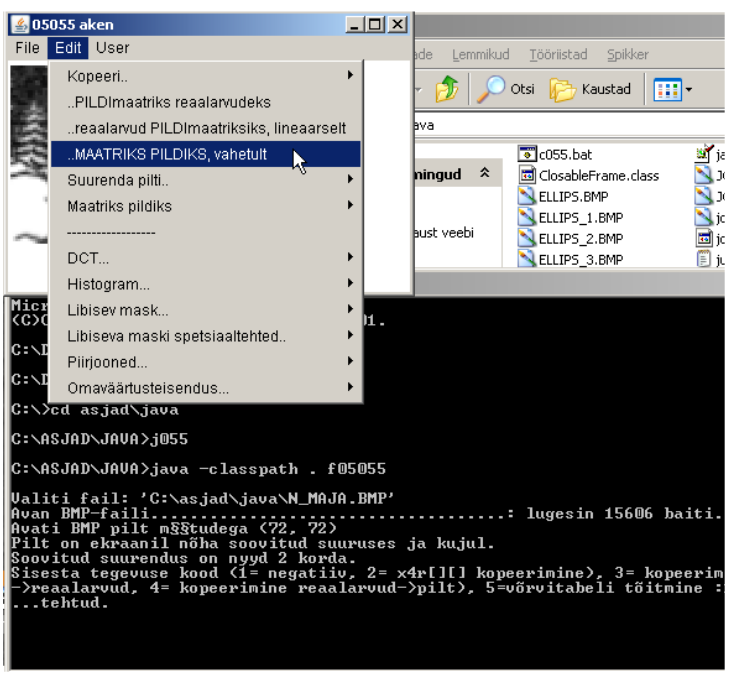


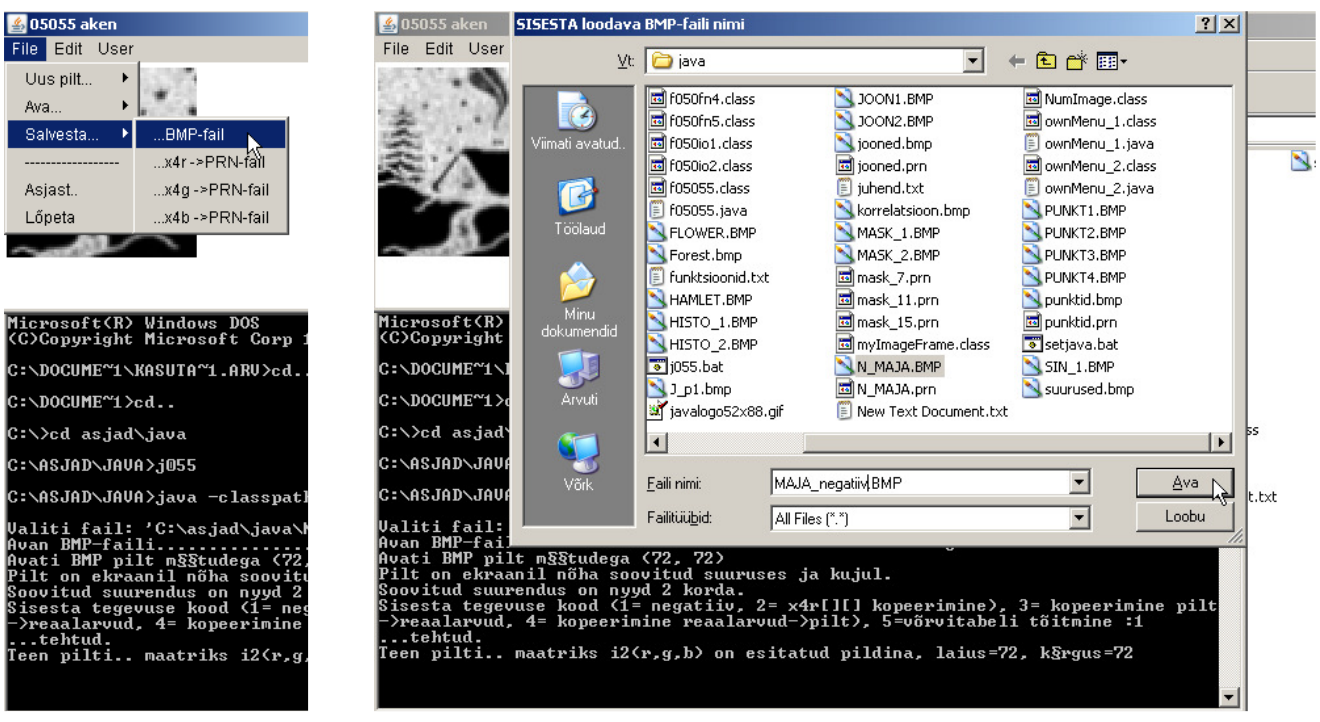
Näidetes valitakse tehe, seejä valitakse fail, siis suurendatakse loe pildi nähtavat kuju kaks kor misjärel on fail soovitud suuru ekraanil. Aknas "05055 ak toimuva kõrval tuleks tähele pa teist, DOS-akent, kus iga kord, menüüdes antakse mingi käsk, iln info, mida tehti (DOS-aknasse ilmu ka veateated). Järelikult, prograr kasutab paralleelselt kahte ake kahte kõrvuti võimalust kä sisestada ja kaht kõrvuti võimal tulemusi näidata. Selline kahe al võimalus peaks sobima nii neile, tahavad tegelda menüüdega, kui neile, kes eelistavad Linux-suhtlemist ehk käskude sisestan käsurealt. Osa käske antaksegi käsurealt, nagu näha siin olevates näidetes.

Näites valitakse selle pildiga tehtavat tegevust. DOS-ekraanilt on näha et tegevuseks "negat tuleb sisestada arv "1". Nagu näha, seda tehaksegi. Tulemuseks on loota pildi negatiivi. Paraku

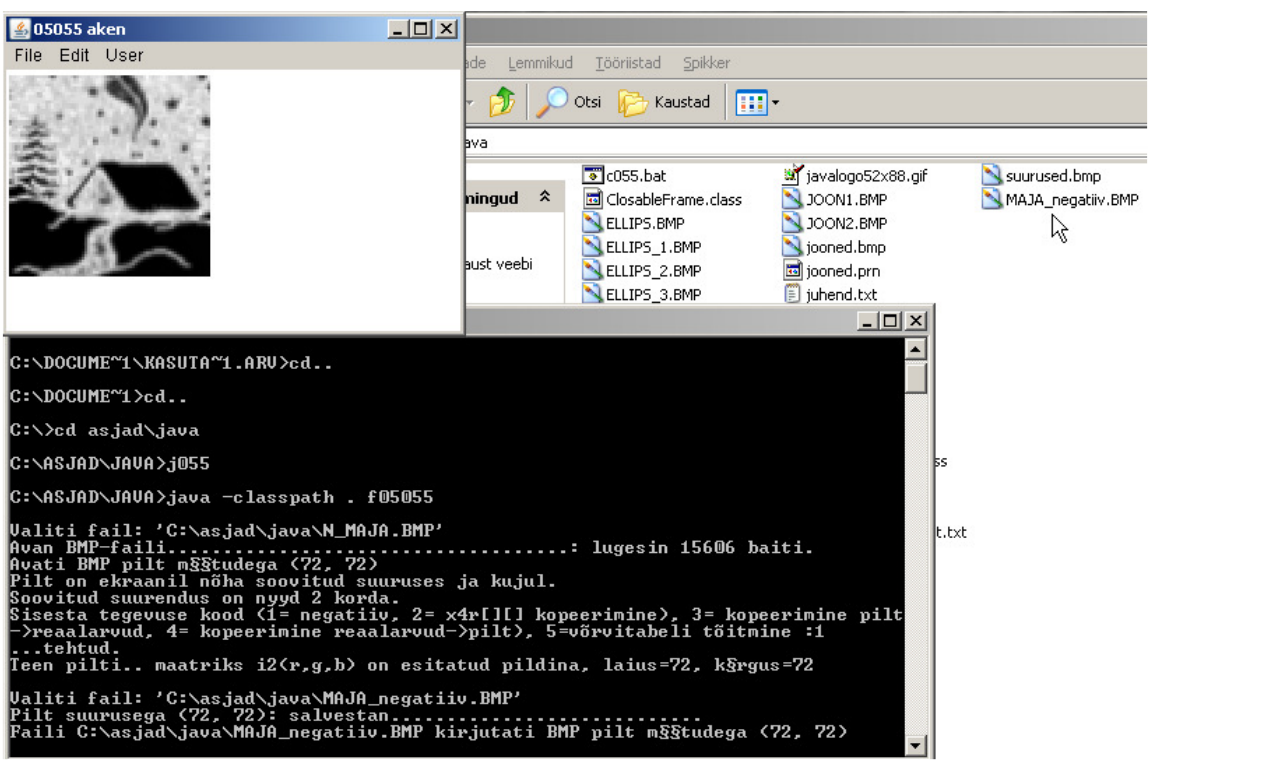


lugu selline et negatiiv küll arvutatakse, aga seda automaatselt ei näidata. Et tulemust näha, tuleb menüüst valida "maatriks pildiks", mida siin toodud näites tehaksegi. Igale sisestatud käsule järgneb mingi tegevus (siin: negatiivi arvutamine), aga ei järgne automaatset pildi uuendamist. Kust peakski programm teadma, tahetakse lihtsalt arvutada või ka kohe tulemust näidata? Kui on pikemad arvutused, oleks iga lõigu järel pildi uuendamise tulemuseks virvendav ekraan, ei midagi meeldivat. Seetõttu on tehtud nii et pildi nägemiseks tuleb seda küsida. Näites valitud käsus olev "vahetult" tähendab seda et näitamisel ei kasutata vahepealseid teisendusi (saab lisada värvitabeli).

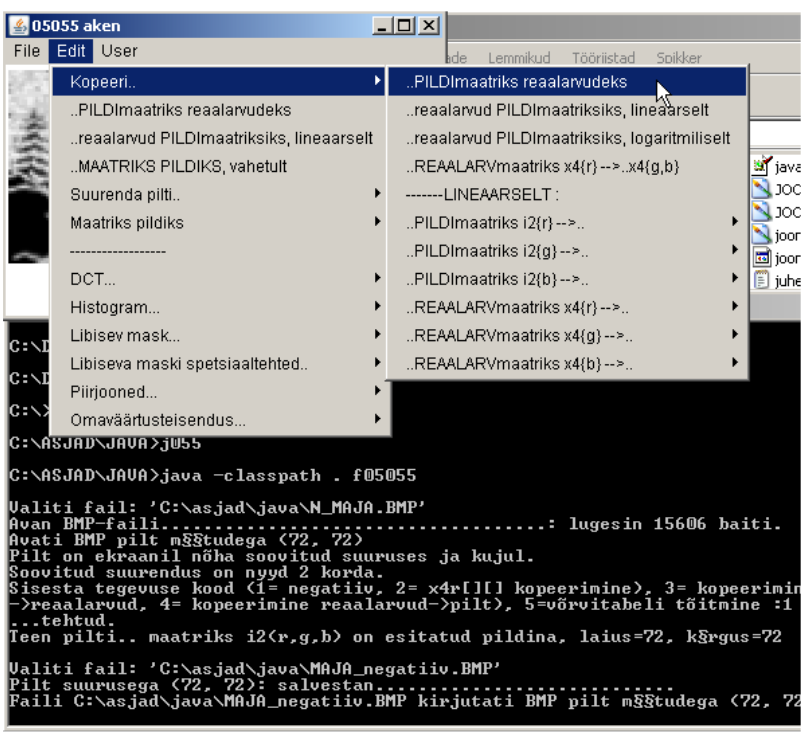




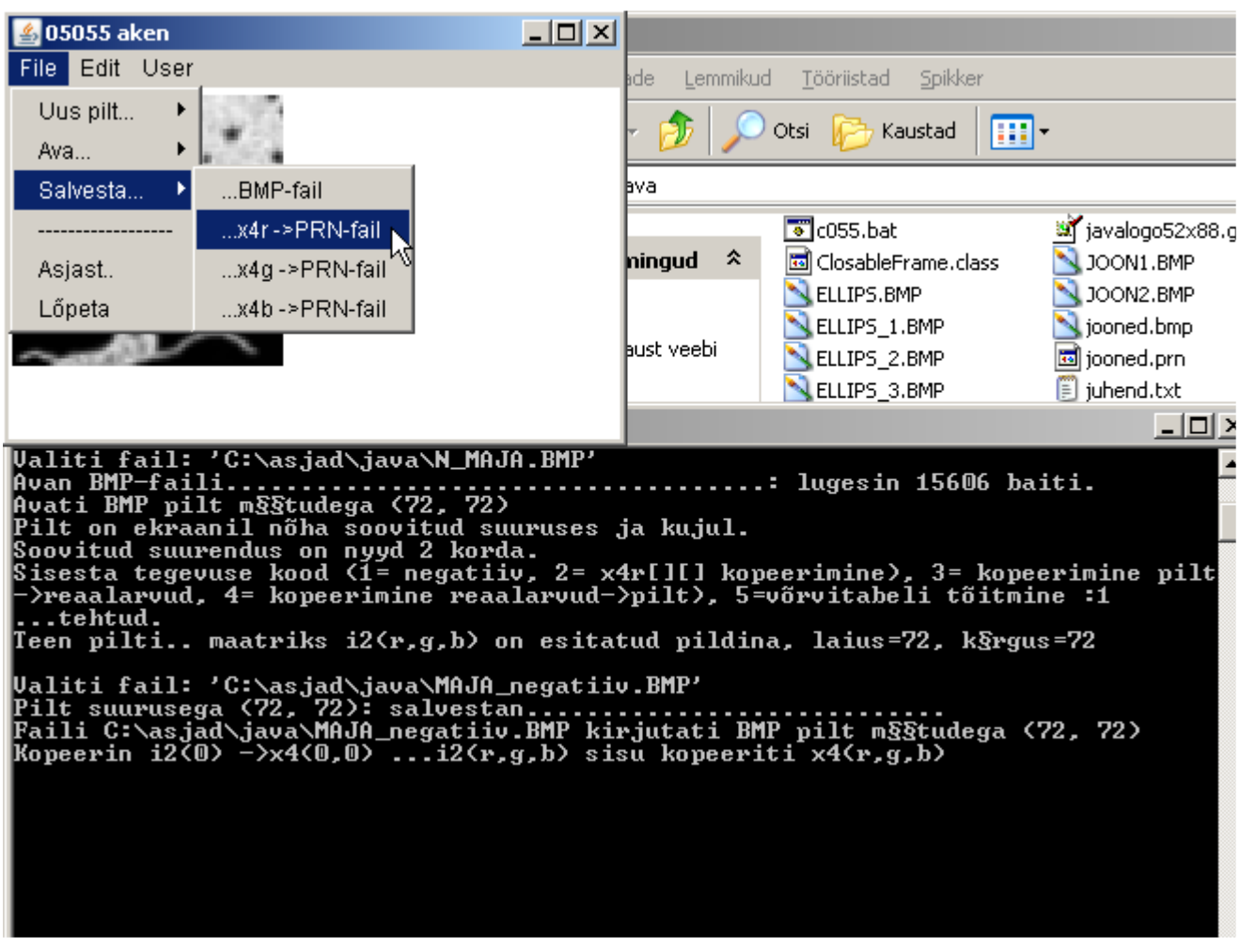
Kui pilt teisendatud (siin: negatiiv arvutatud), võib tekkida tahtmine seda salvestada. F salvestamisega tegeldakse siin toodud näidetes. Nagu näha, saab pilt edukalt salvestatud (katalo ilmub uus, valitud nimega fail).



Lisaks .BMP-failidele saab lugeda ja salvestada .PRN-faile, mis on oma olemuselt komakohaga arvude trükikuju sisaldavad tekstifailid. Et näha, milline üks .PRN-fail välja näeb, salvestame äsja saadud pildi .PRN-failina ja vaatame tulemust Notepad-ga.

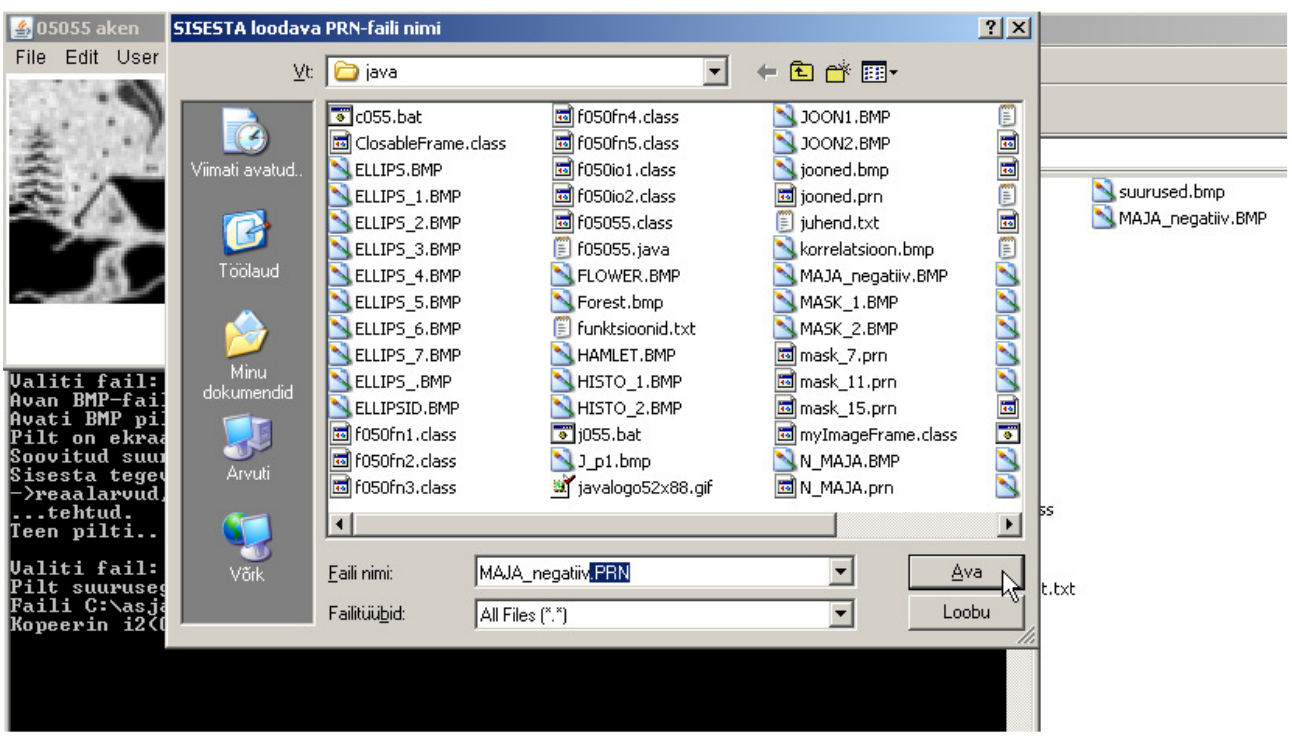


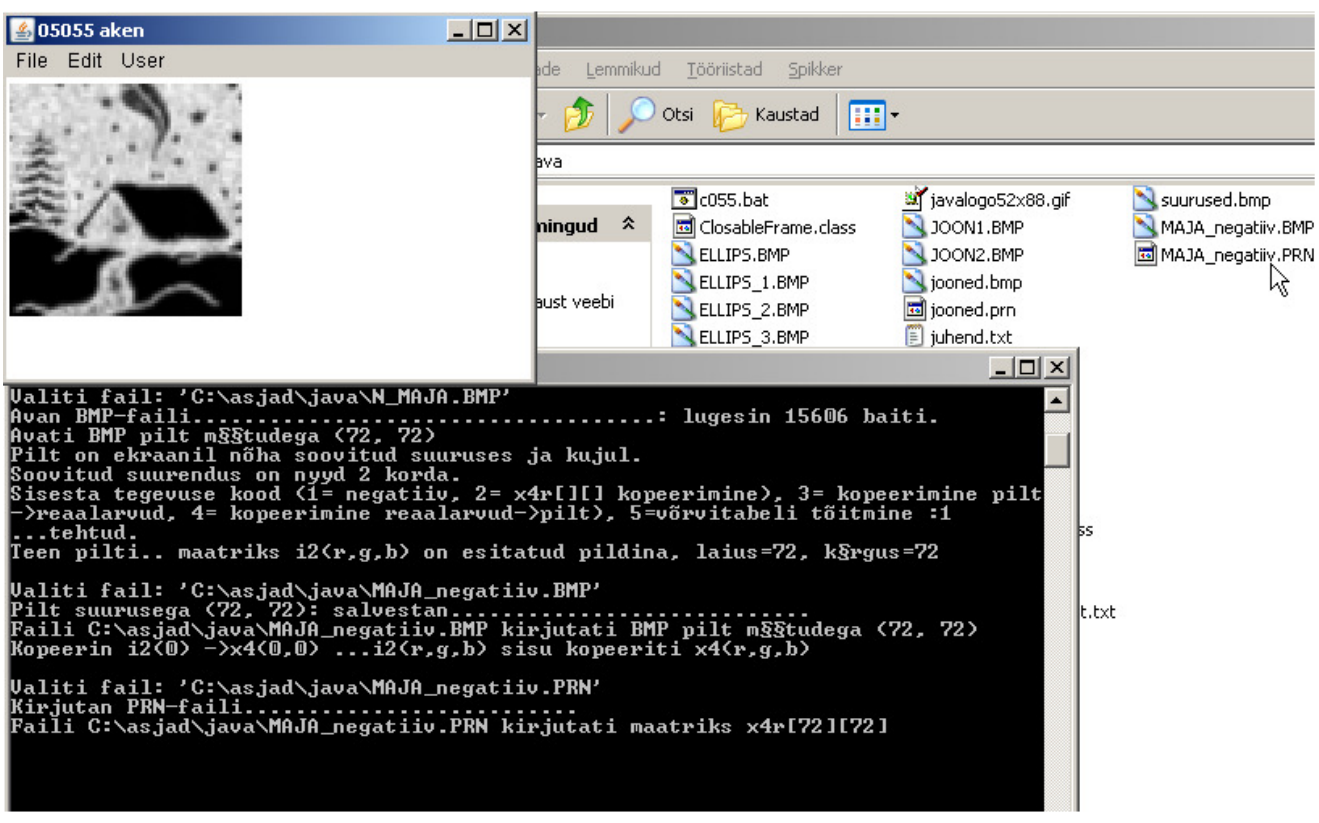
Kõigepealt tuleb rääkida Java-paketi mälujaotusest. Loetud ja salvestatavaid .BMP-faile hoitakse süsteemses täisarv-massiivis, mis on kasutatav kogu süsteemis. Selle kõrval on kogu süsteemis reaalarvmassivid, kus hoitakse (arvutustes kasutatavaid) reaalarve. Need massiivid ei ole omavahel automaatselt seotud. Seega, kui tahta et pildimassivide sisu satuks reaalarvmassiividesse, tuleb see kõigepealt sinna kopeerida (ja vastupidi, kui tahetakse reaala arvutustulemusi esitada pildina, tuleb reaalarvmassiivid esmalt pildimassiividesse kopeerida). Sell on olemas menüüst, aga ka funktsioonidest käivitatavad tehted, mis seda teevad. Näi alustataksegi sellest et kopeeritakse pildi sisu reaalarvudeks.



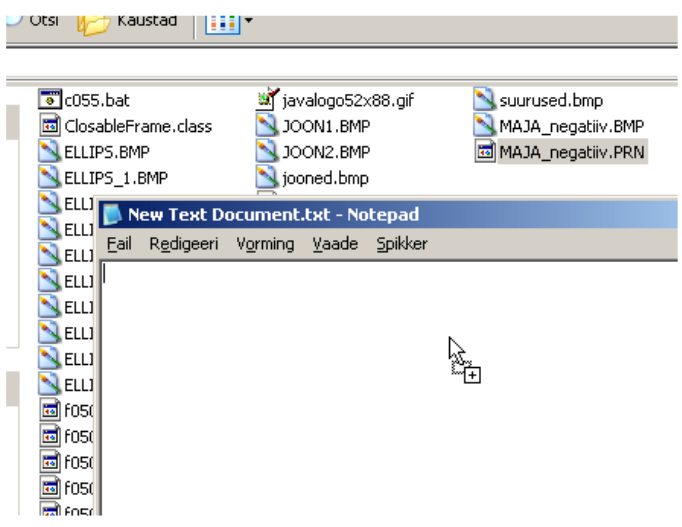
Seejärel on näide, kus salvestatakse reaalarvud .PRN-faili. Esmalt valitakse menüüst tehe, kirjutatakse sobilik nimi. **Selle juures jälgida, et faili nime laiendiks oleks tingimata PR Samuti jälgida, et selle nimega faili varem ei oleks.** Seejärel tuleb vajutada "Ava", eh tegelikult on tegemist salvestamisega. Nagu järgnevalt näha, saab fail edukalt salvestatud.

PRN-failid on mõeldud andmevahetuseks programmidega mis selliseid loevad/kirjutavad, nt MathC või Excel. Aga neid saab ka (nt Notepad-ga) niisama vaadata, mõnel puhul piisab ka niisa vaatamisest et miskit olulist tähele panna. Järgnevalt vaatame loodud faili Notepad-ga.

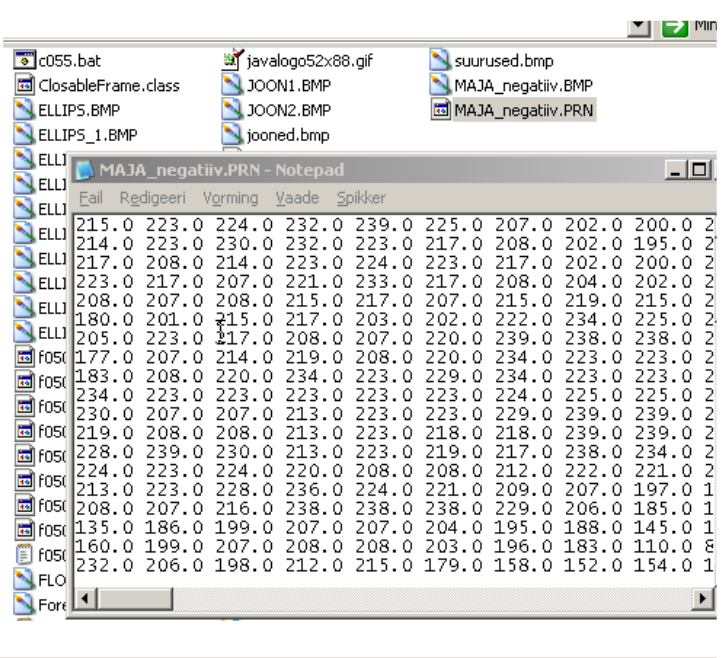




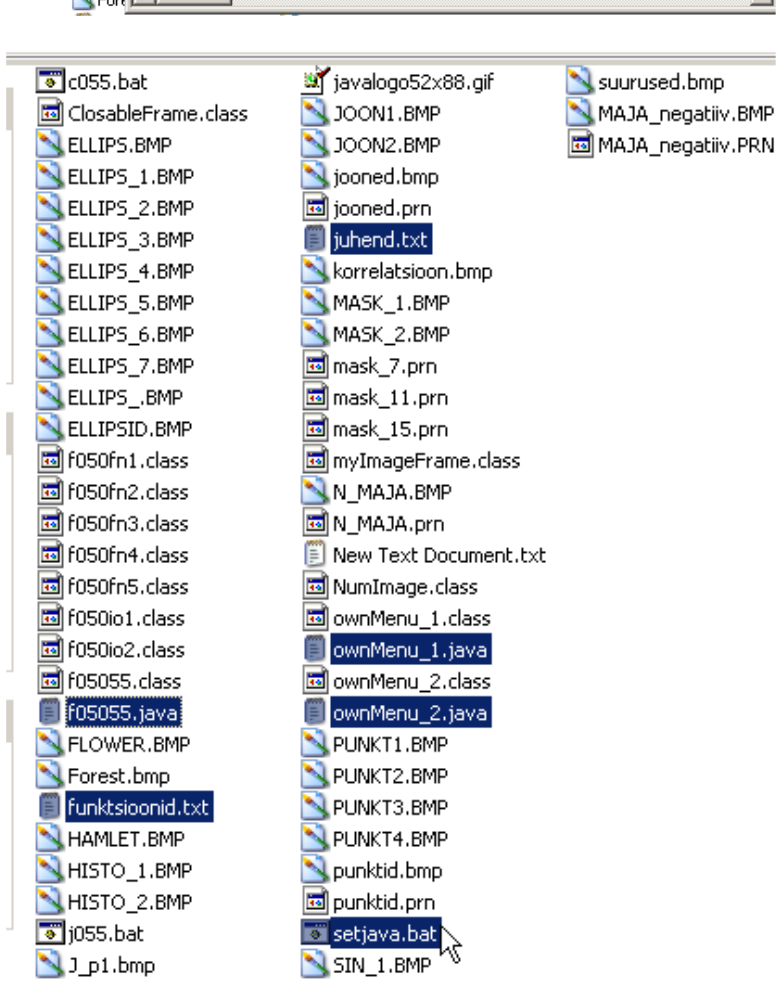
Nagu näha, sisaldab salvestatud fail ühe kümnendkohaga reaalarve, kus kümnendkoht on eraldatud punktiga. Kui faili imporditakse näiteks MathCad-i või siis Excel-sse, tuleb sellise formaadiga arvestada. MathCad-l pole seni probleeme olnud, Excel puhul võib olla vaja parameetreid veidi sättida.



Olles nüüd faili sisu näinud, tuleks veel kord küsida et mida siis õigupoolest salvestati? Kas pildi kogu sisu? Tegelikult mitte. Kui meenutada, siis originaalis on meil tegemist TrueColor pildiga, mis tähendab et see sisaldab kolme sõltumatut kihti (R, G, B). Siin näeme me ainult ühe kihi sisu. Seega salvestati kolmandik pilti, ehk pildi üks kihtidest. Selline lähenemine on üsna loomulik, kuna oleks üpris tülikas salvestada tekstifaili korraga kõik kolm kihti. Kuidas peaks neid kihte pärastpoole eristama? Tulemuse importimisel muudesse programmidesse (nt MathCad) tekiksid kohe tõsised probleemid. Seetõttu, parem üks kiht korraga. Siinkohas siiski üks küsimus. **Nimelt, milline kihtidest (R,G,B) siin salvestati?** Vastus on ühel eelmistest näidetest.

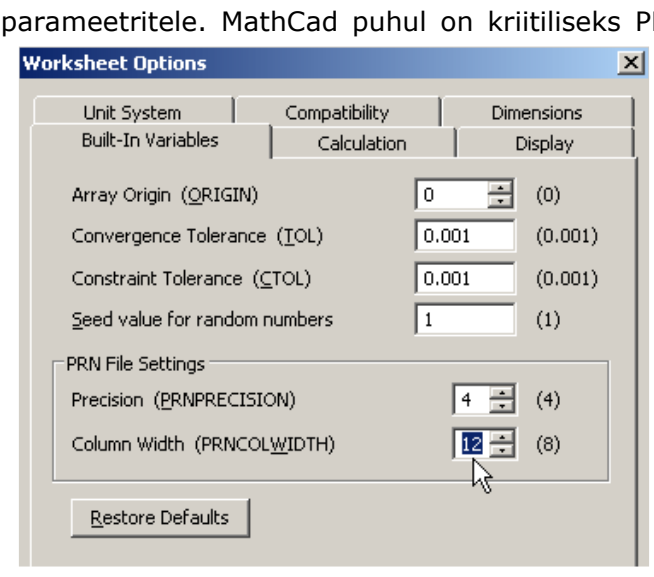


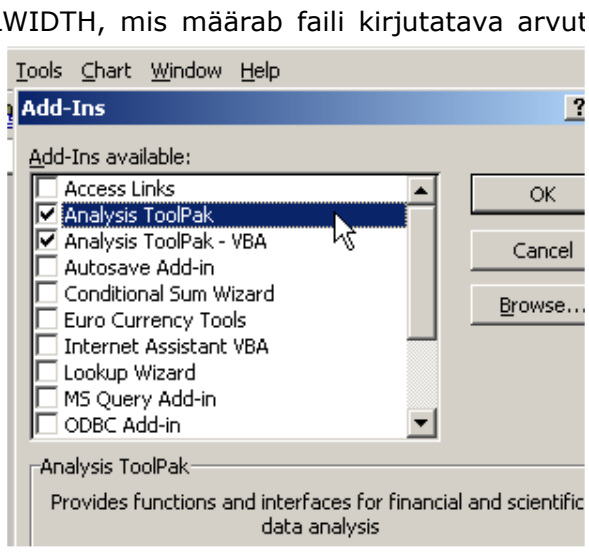
Java-paketi tutvustuse lõpetuseks nimekiri paketis olevatest failidest, koos märgetega, millised failidest on mõeldud muutmiseks. Ülejäänud oleks parem rahule jätta, seda vähemalt esialgsete teadmiste juures. Mitmed nimed räägivad ise enda eest. "Juhend.txt" tähendab juhendit. "Funktsioonid.txt" sisaldab juba valmis funktsioonide kirjeldusi. "Setjava.bat" sisaldab Java keskkonna ülesleidmiseks vajalikku infot, sellest oli juttu eespool. Kui tekib kahtlus et süsteem ei leia Javat üles, on mõistlik anda käsk "setjava", näiteks iga kord pärast arvuti uut käivitamist töö esimese asjana (mõistagi, fail peab sisaldama seda, mida vaja). Failid nime laiendiga "java" on programmi kood, ühtlasi näide kuidas tuleks oma programmid vormistada. Kui katsetamisega usinalt peale hakata, peaks keskkond varsti selgeks saama.



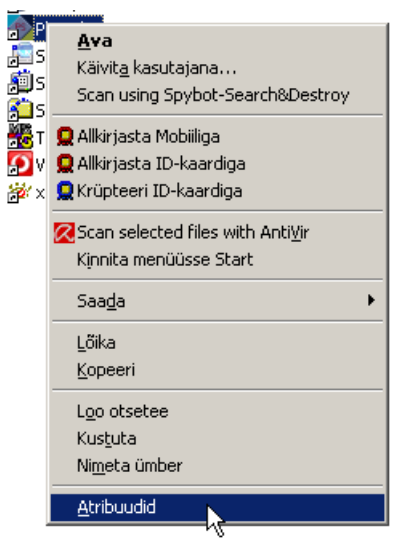
Kursusel soovitatava muu tarkvara tutvustus

Kursusel pakutakse kasutada PhotoStyler, MultiSpec, IrfanView, MathCad ja Excel. viimatinimetatut on üsnagi laialt tuntud, mistõttu nemad pikemat tutvustust ehk ei vaja. Nende p juhiks vaid tähelepanu mõnedele vähem kasutatavatele, aga selle teema raames kriitili parameetritele. MathCad puhul on kriitiliseks PRNCOLWIDTH, mis määrab faili kirjutatava arvut

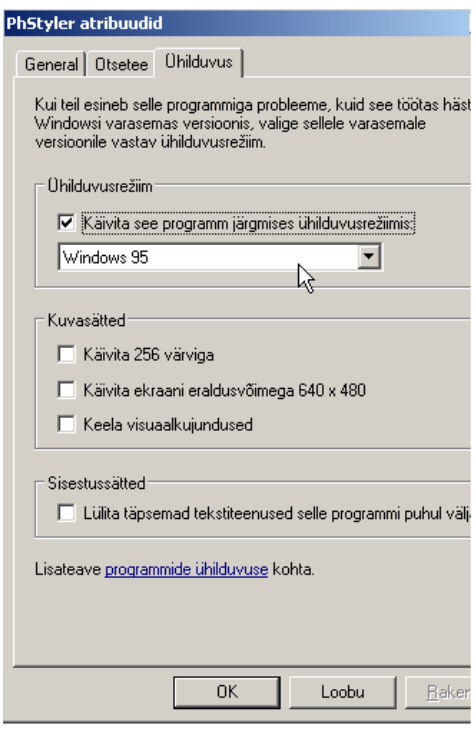




veeru laiuse. Tuleb tagada et veeru laius oleks p (veerud oleksid eraldatud vähemalt ühe tühikuga), muidu ei suuda Java-pakett Mat kirjutatud .prn-faili sisse lugeda. Kõrvalolev pilt näitab, kuidas veeru laiust muuta. Excel puhul kasutamist pakett DataAnalysis. On võimalik, et vaikimisi pole see pakett aktiveeritud. Kui pole (T all pole rida Data Analysis), tuleb pakett aktiveerida viisil, nagu näidatud pildidl. Ka IrfanVie intuitiivselt hästi hoomatav, pealegi omab ta oma Help, mistõttu ka tema üldist tutvustust ehk ei Nimekirja kaks esimest on ehk kõige vähem tuntud, seetõttu keskendume neile.

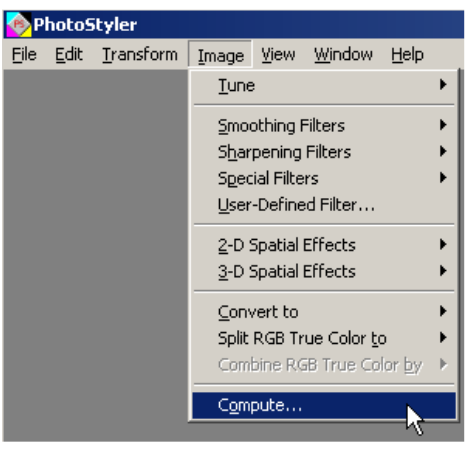


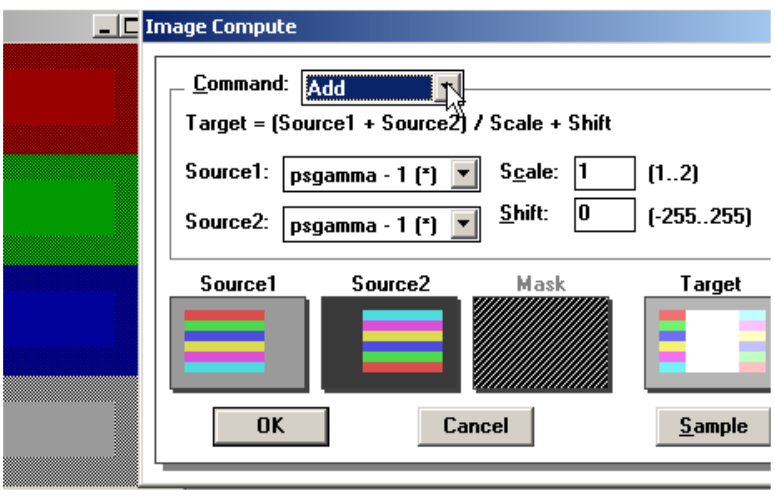
Photostyler on hulk aastaid vana programm, mistõttu võib ta vajada tööreziimi sättimist. Kui ta ikoonil tavapärase klõpsamisega käima ei lähe, võiks võtta ikooni alt "atribuudid" ja seejärel "käivita Windows 95 reziimis". Käivitumisel ilmub esmalt programmi enda logo. Et programm avaneks, tuleb sellel logol klõpsida.



Järgnevalt on rida näiteid illustreerimaks, kus asuvad Photostyler mõned olulisemad tehted. Ühtlasi saab teatava ülevaate Photostyler menüüdest. Kõiki näidetes toodud menüüsid selle teema all ei kasutata, s võib olla põhjust edaspidiste teemade juures sellesse juhendisse tagasi pöörduda.

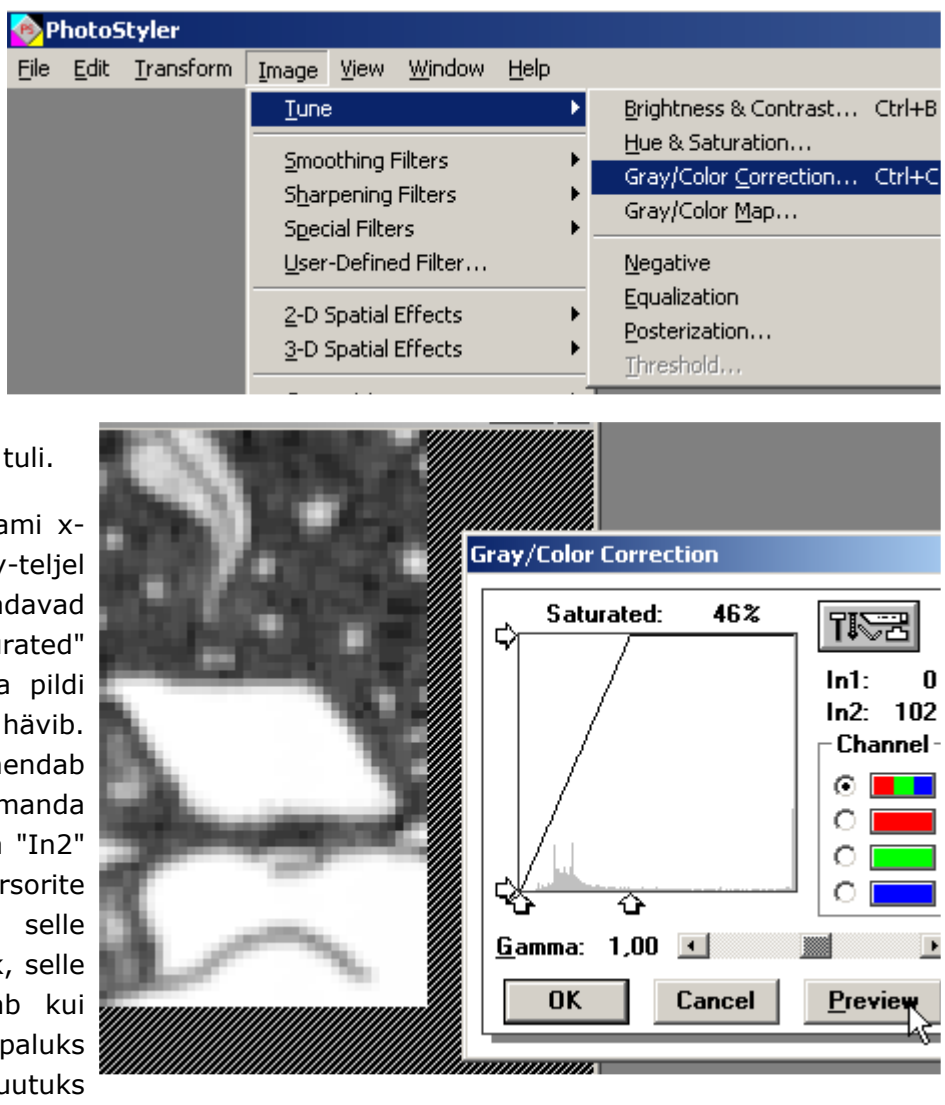
Esimeses näites viidatakse





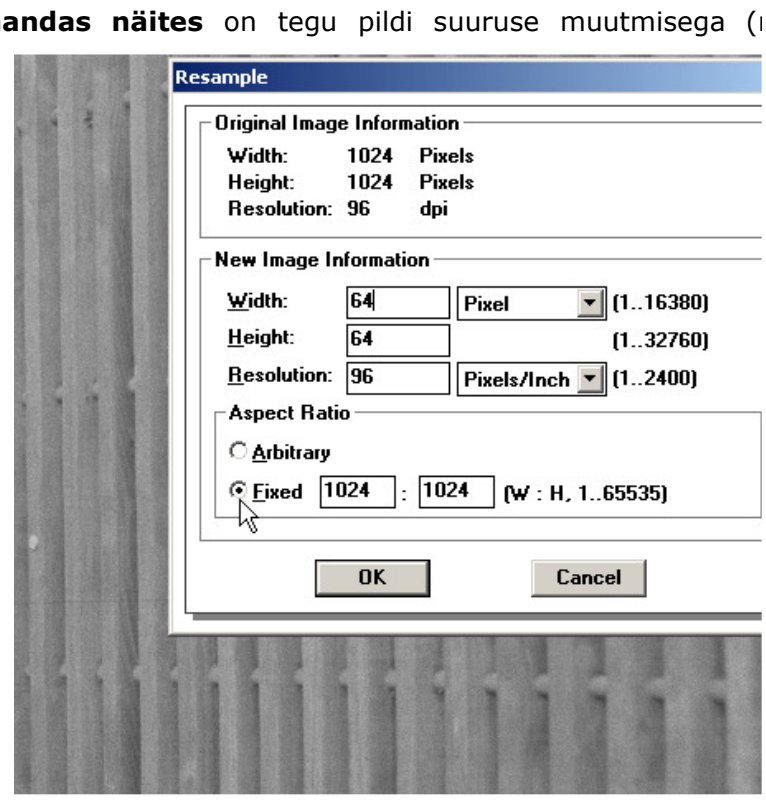
kasulikule, kuid pilditöötlusprogrammides harva esinevale tehtele "Compute". Kursuse esimene teema on läb seega peaks ka olema hoomatav, mida "Compute" teeb. Milline on ühe pildi arvkuju, oli eelr teemas juttu. Siinses tehtes võetakse kaks pilti "Source 1" ja "Source 2" ning tehakse midagi se mida arvudega ikka teha saab. Konkreetset tehet saab valida menüüst "Command", selle p antakse tehte selgitus valemina "Target = ...". Milleks on head valitavad parameetrid "Scale" ja "S on loetav tehte valemist. Kui programmi kasutamise vilumus areneb, saab ka selgeks, kuidas v kolmandat võimalikku operandi "Mask". Paljudele tehnilistele küsimustele saab vastuse Photos Help alt.

Teises näites viidatakse tehtele "Gray/Color correction". See tehe võimaldab vaadata ja muuta pildi histogrammi. Mis on histogramm, sellest tuleb täpsemalt juttu kolmandas teemas. Siinkohas pildiline näide, kuidas saab histogrammi muuta, nagu ka näide, mis sellise muutmise peale välja tuli.



Histogrammi ümbritseva raami x-teljel on vanad heledused, y-teljel on muutmise tulemusel saadavad uued heledused. "Saturated" näitab, kui suur osa pildi sisust muutmise tagajärjel hävib. Küsimus: mida tähendab "hävib"? Vastuse saab kolmanda teema alt. Näidikud "In1" ja "In2" näitavad x-telje kursorite asukohti. Küsimus: mida selle asukohaga peale hakata ehk, selle näite juhul, mida tähendab kui "In2=102"? Ülesanne: paluks kujutada ette, kuidas muutuks pilt selle teisenduse tagajärjel? **Pilt on vasakul näha, milline oleks muudetud pilt?** Vastust näha allpool oleva lingi kaudu, aga paluks seda kohe mitte vaadata. Plokk "Channel" määrab, m kanali histogrammi vaadatakse. Antud juhul vaadatakse pildi keskmist histogrammi. "Gamma" mu pildi visuaalset kuju, sellest täpsemalt kolmandas teemas. Ja kui vajutada "Preview", tulebki tulem

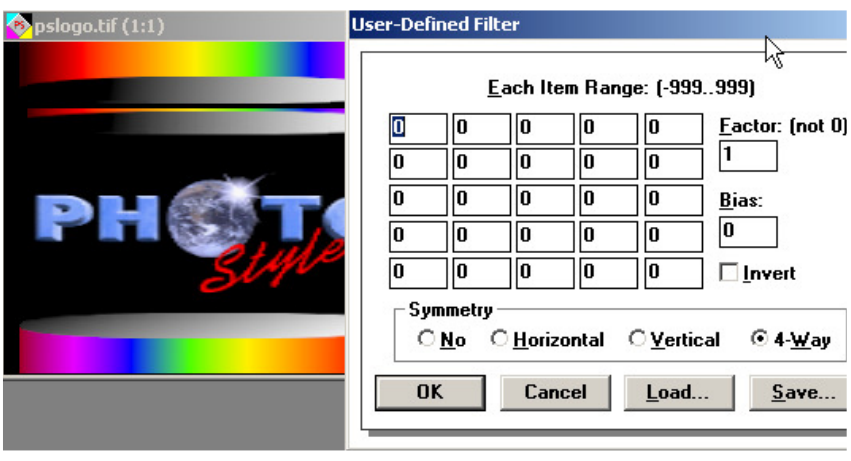
Kolmandas näites on tegu pildi suuruse muutmisega (



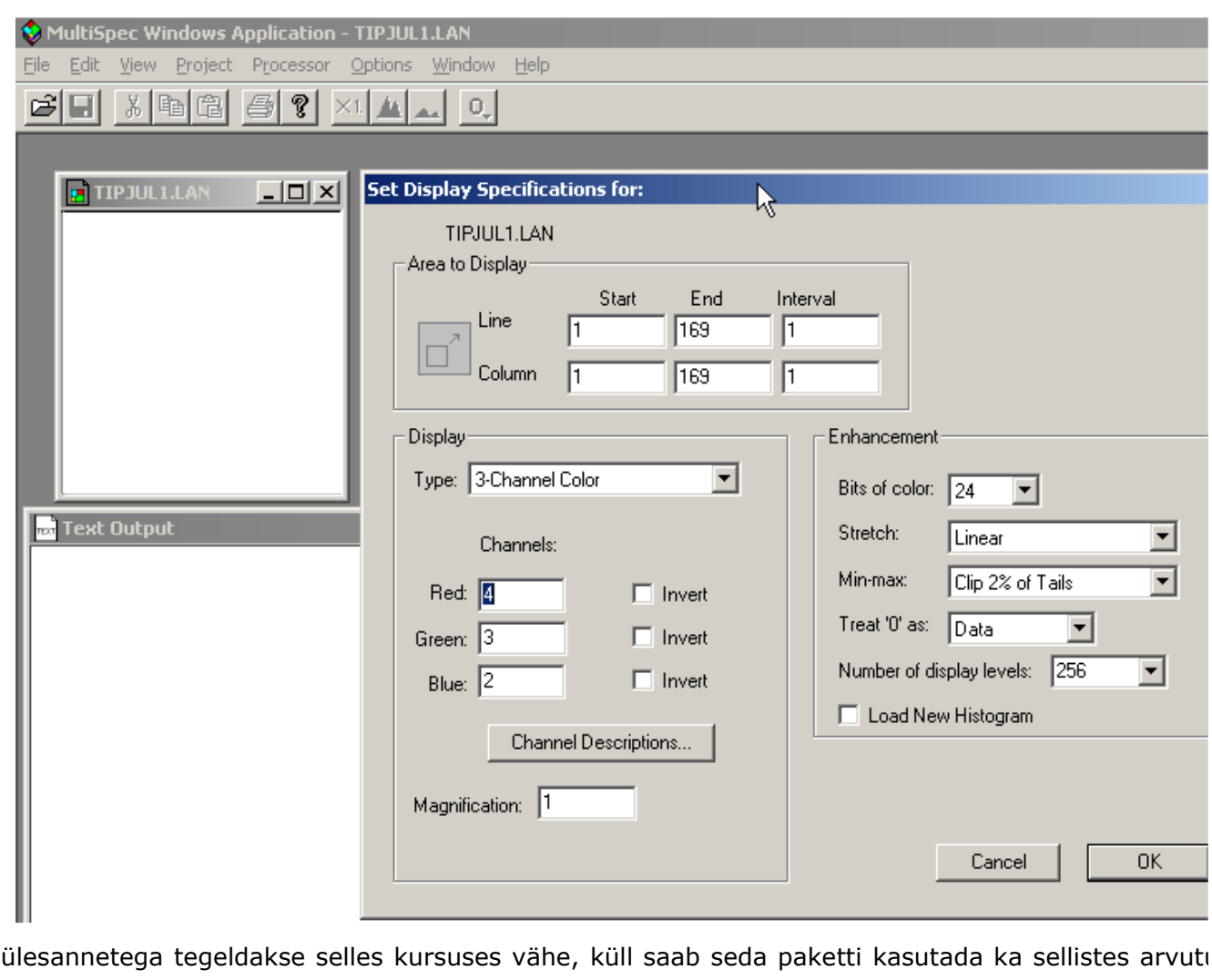
nähtava suuruse muutmisega, vaid

suuruse tegeliku muutmisega). Näide on antud ühe sellise pildi taustal, millist pilti hakat praktilistes töödes ka tegelikult muutma. Nagu näha, valitakse kõigepealt tehe ja seejärel pildi suurus.

Neljandas näites on tegu tehtega "User defined filter". Sisuliselt tegeldakse selle tehtega viiendas teemas, siinkohas olgu selgitatud mõnede väljade sisu. On selge et arve sisaldavatesse väljadesse tuleb kirjutada arvud. Põhimõtteliselt võivad need olla mistahes täisarvud piirkonnas -999 kuni 999. Plokk "Symmetry" määrab 5x5 maatriksi sümmeetria. Paluks ise katsetada. "Factor" tähendab arvu, millega tehte tulemus läbi jagatakse (mis tehte mis tulemus, sellest viie teemas). "Bias" tähendab arvu, mis liidetakse tehte tulemusele. "Load" ja "Save" on edasijõudn (sisestatut saab salvestada ja hiljem kasutada).



MultiSpec on oma olemuselt mõeldud paljukanaliliste kaugseirepiltide töötlemiseks. Se



ülesannetega tegeldakse selles kursuses vähe, küll saab seda paketti kasutada ka sellistes arvutu ja demonstratsioonides, mis on eriti silmatorkavad just juhul kui argumendiks on paljukanal pildid.

Nagu näha, on MultiSpec ühes mõttes sarnane Java-paketile: mõlemad kasutavad paralleelselt pil tekstiaknaid. Seega tuleb ka siin jälgida et kui miski ei ilmunud pildiaknasse, ehk ilmus see (tule tekstiaknasse. Antud juhul hakatakse avama pilti "Tipjul1.lan". Nagu näha, pakutakse kohe h parameetreid, mis kõik ootavad väärtusi. Osa neist parameetritest on omased just paljukanalili piltidele. **Millised?** Ülejäänud parameetreid võiks samahästi küsida mistahes pildi avamise juures kõik nad on seotud digitaalpildi ja selle nähtava kuju vahekorraga ja sellistena väärivad nad tähelepanu. Esmajoones pööraks tähelepanu valikule "Enhancement:min-max". **Mida v tähendada** "clip 2% of tails"? Vastuse peaks saama teisest teemast ja praktilisest katsetami Tegelikult oleks vajalik teada ka mitme teise välja sisu. Niisiis, **mida muudab** Display: Type? Dis Channels? Enhancement: Stretch? Enhancement: number of display levels? Need vastused p saama esimesest teemast ja, jällegi, katsetamisest. Vastuste väljamõtlemisse tuleks suhtuda tõs Esiteks, need on küsimused ja, teiseks, vastused aitavad mõista mistahes pildi siseehitust.

2.teema ülesanded koos lisaküsimustega

_**Vahetekst,** lugemiseks ja juhisteks (valgel taustal).
Näideteks toodud väljavõtted programmitekstidest (helerohelisel taustal).
Ülesanded ja küsimused, mis on mõeldud lahendamiseks/vastamiseks (kollasel taustal).

Need ülesanded, mis on allpool mõeldud lahendamiseks Java-paketi abil, võib lahendada ka mingi muu vahendiga, **eeldustel et** retsenseeriv rühm saab neid kontrollida ja et lahendamine saab tähendama uute tehete ja kasutajaliidese tekitamist, mitte pelgalt olemasolevate tehete rakendamist.

1. Tutvumine tarkvaraga ja pildi valevärvesitus.

1.1. Pildi põhilisimad tehted ja omadused.

Selleks et Java-paketti (koos teiste programmidega) üldse sihipäraselt kasutada, tuleb alustada kahest asjast: 1) Pildi tekitamine või lugemine ja pärastine salvestamine, 2) pildi elementaarseim muutmine. Ühtlasi peab olema kindlalt teada, kus on pildiaknas nähtavate piltide nullpunktid. Esimene ülesanne puudutabki neid tehteid.

1) Tekitada Java-paketis uus pilt suurusega 64 korda 64 pikslit, sõltuvalt soovist kas must või valge. Uue pildi tekitamiseks on olemas menüü File->Uus pilt ja selle alammenüüd. Uue pildi muud parameetrid tuleb sisestada tekstiaknas. 2) Värvida pilt alljärgnevalt: piksel punktis (0,0) olgu maksimaalselt punane, piksel punktis (63,63) olgu maksimaalselt sinine; punktis (x=0,y=63) olgu puhas kollane ja punktis (x=63,y=0) puhas roheline. Kui sobivaid tehted ei leia, tuleb need programmeerida, aga selleks on paketi failides "05055.java" ," ownMenu_1.java" ja "ownMenu_2.java" hulgaliselt näiteid. Pilt salvestada. Salvestamiseks on juhendis näide olemas. Kontrollimiseks esitada saadud pilt.

Seejärel selgitada, kus asuvad x-telg ning nullpunktid PhotoStyler ja IrvanView ekraanidel. Vastused anda sõnaliselt (ega muid variante pole kui horisontaalne/vertikaalne ja vasak/parem alumine/ülemine nurk).

1.2. Pildi teisendus uude värviruumi, kasutades PhotoStyler võimalusi.

Aluseks võtta see pilt. Arvutada pildi alternatiivse värviruumi (H,S,B) primaarvärvid (Hue, Saturation, Brightness). Salvestada tulemused piltidena ja lisaks tekitada ja salvestada pilt (H,B) kanalite kombinatsioonist.

Kontrollimiseks saata saadud pildid (koos üheselt mõistetavate failinimega või, veel parem, tekstilisi kommentaare sisaldava indeksfailiga).

1.3. Pildi nähtava kuju muutmine värvitabeli abil.

Muuta pildi 1 ja pildi 2 nähtavaid kujusid, kasutades selleks Java-paketi selleks loodud/muudetud funktsiooni. **Originaalpilt peab jääma muutumatuks.**

Paketi fail ownMenu_2.java sisaldab protseduuri public static void valik_4 (int tegevuse_liik). Algversioonis on määratud "tegevuse liigid" 1 kuni 5. Lisada uus liik (6), kuhu programmeerida värvitabeli sobiv muutus. Ei tohiks keeruline olla, liigi 5 all on värvitabeli teatava täitmise näide olemas (**mida see näide teeb** ehk, teisisõnu, kuidas muudab liigi 5 alt tulev värvitabel pildi nähtavat kuju?).

Nüüd on mõistlik see "liik 5" all olev tehe kopeerida, lisada uus sisend (6) ja sisendi (6) all muuta värvitabeli täitmise algoritmi (sisendi 5 algoritmi mitte muuta). Lisaks uue sisendi sisu tekitamisele lisada ka kasutajaliides,

ehk kasutaja peab ekraanilt nägema et on tekkinud uus võimalus, ja mida selle võimaluse kasutamiseks teha. Kasutajaliidese muutmiseks tuleb muuta faili "f05055.java", otsides sealt üles sobiva koha ja selle ära muutes. **Vihje:** võiks otsida kohta

```
// mida teeb menüüelement own_2_4.
```

Kui programmide tekstid muudetud, tuleb uued failid uuesti kompileerida (käsk "c05055"). Proovida, kas tulemus on see mida oodati.

Pildi nähtavat kuju muuta järgmiselt: 1) Pildi nähtav kuju peab olema pildi negatiiv 2) Näidata ainult pildi rohelise ja sinise kanali kombinatsiooni (punane kanal peab näima pildil mustana).

Lahenduse tõestamiseks esitada kõik kontrollimiseks vajalikud materjalid. Kui retsensent neid materjale kasutab, peab ta saama ise veenduda et lahendus töötab.

2. Lokaalsed (pikslikaupa tehtavad) pilditehted.
Eesmärgiks on omandada teadmisi pildiga teostatavate matemaatiliste elementaartehete mõjust pildi kujule. Lisaks aitab tehtav omandada tehete iseseisva programmeerimise kogemust, sest ühelt poolt on olemas tehteid realiseerivad tavavahendid (PhotoStyler), ja teiselt poolt on olemas Java-keskkond kus saab samad tehted programmiliselt läbi teha. Ja kui tekib kahtlus tulemuse õigsuses, saab (vähemalt esialgu) Java-ga saadud tulemuse tavavahendiga üle kontrollida.

2.1. Pilditehted tavavahenditega.

Avada PhotoStyleri abil mõni oma lemmikpiltidest, parem kui mitte väga suur (kui ühtki ei leia, siis palun: pilt 1, pilt 2). NB! võib juhtuda et PhotoStyler Open-menüüs mõnda failitüüpi justkui ei olegi. Peavad olema .bmp, .tif, .tga, .pcx. Kui neid ei ole, on mõni läinud menüüs nurga taha, liikuda kursoriga failitüüpide menüüs ülespoole. Vihje: kui pilt avatud, võib olla kaval teha temast koopia(d), ja muuta mitte originaali, vaid koopiat: Duplicate ehk Ctrl+D.

Ülesanded puudutavad tehet "Compute". Valida avatud pilt tehete esimeseks operandiks, teiseks operandiks valida mõni muu pilt. NB! Mõlemad operandid peavad olema sama suurusega, aga see pole meie jaoks ju probleem, või mis? Analüüsida tehete mõju, siis muuta tehete parameetreid ja uurida tehete parameetrite mõju. Tehete mõistmiseks võib olla kaval valida teiseks operandiks mingi ühevärviline "pilt" (must, valge, R, G, B või mõni muu värv). Kui PhotoStyler-s ei õnnestu ühevärvilist pilti tekitada, on kindlasti abiks Java-pakett, milles on tehe File-> Uus pilt-> pildi parameetrid ->pildi suurus (suurus sisestatakse tekstiekraanilt). Kindlasti uurida tehteid Add, Subtract, Difference, Lighter, Darker.

Mille poolest erinevad Subtract ja Difference, **mille poolest erinevad** Lighter ja Darker (selgitada kirjalikult). Soovitavalt uurida ka tehet Composite. Mida see tehe teeb (kirjalik selgitus).

Lisaküsimused ja ülesanded: Mis on pildi difference mustast/valgest pildist? Mis on pilt subtract must/valge pilt? Mis on pildi difference punasest pildist? Lisaks neile kolmele tekitada veel viis huvitavamat tulemust, koos selgitustega.

Retsenseerimiseks esitada vastused küsimustele, kommentaarid ja kasutatud ning saadud pildid. Retsensendil peab olema võimalik kommentaaride abil üheselt aru saada, mida on tehtud, ja tehtut korrata.

2.2. Pilditehete programmeerimine.

Programmeerida Java-keskkonnas vähemalt kolm eelmises ülesandes saadud juhtumit (saadud programm(id) peavad olema võimelised tegema needsamad teisendused, mida tehti PhotoStyler-ga). Lisada kasutajaliides uue menüüelemendi või uute menüüelementide kujul. Arvestada, et Java loeb ainult .bmp-pilte, seetõttu tuleks operandidena kasutada seda tüüpi pilte. Vajadusel saab pildiformaate IrvanView abil pea alati vajalikuks konverteerida. Kuhu ja kuidas programmeerida, selle kohta leiab näiteid failidest ownMenu_1.java, ownMenu_2.java ja f05055.java. Abiks on ka punktis 1.2 antud juhised. Veel vihjeid:
1) Kindlasti on abiks juba tehtud, failis ownMenu_2.java sisalduvad tehted

```
public static void valik_4 (int tegevuse_liik)
```

if (tegevuse_liik == 1) ...
if (tegevuse_liik == 2) ...

2) Ära unusta et Java-paketis on peale pildimassiivide olemas ka reaalarvmassiivid, ja kuna süsteem neid ei muuda, saad neid kasutada teise pildi hoidmiseks.

3) Kui valida Composite tehte programmeerimine, võib minna vaja ka kolmandat pilti. Sel juhul ära unusta et juba on kirjeldatud veel ühed massiivid, vt failis f05055.java sisalduvat tehet

// mida teeb menüüelement own_0_2.

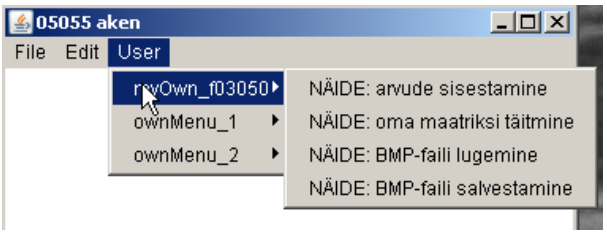
Tõsi, eelnevas näites toodu on kasutatav ainult selles moodulis. Aga kogu süsteemis on kasutatavad

// public static int i4r[][]; (maatriksid mille sisu süsteemselt...
// public static int i4g[][]; ...ei kasutata, eesmärgiks on pakkuda
// public static int i4b[][]; ...igal pool kasutatavat lisamälu)

(vt faili f05055.java algusosa). Nende massiivide kasutamiseks loe lisainfot paketis olevast failist "juhend.txt".

Menüüelementide tekitamise ja aktiveerimise kohta on paketis olevates programmides näited olemas, vt failis 05055.java reale

// Nüüd näide kuidas tuleb defineerida/lisada menüüelemente; järgnevat. Menüüelemente saab lisada menüüsse "User" (vt näidet olemasolevatest menüüdest).



Retsenseerimiseks esitada kõik vajalikud materjalid, et retsensent saaks tehtut oma arvutil kontrollida.

Jõudu ja edu!

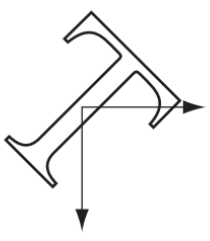
3.teema õppematerjalid

1. Koordinaateisendused. Pildi geomeetrilised teisendused. Pildi koordinaatteisendustega kaasneda võivad probleemid. Geomeetriliste moonutuste kõrvaldamine. 2. Värvusskaala teisendused. Histogrammi olemus. Histogrammiteisenduste olemus. Histogrammi tasandamine (equalization). Histogrammide sarnastamine. Ebahomogeensete moonutustegurite avastamine ja kõrvaldamine. Histogrammi matemaatiline sarnastamine malliga.

Õpikeskkond: TÜ Moodle
Kursus: Pildiinfo töötlus (LOFY.05.055)
Koosta raamat: 3.teema õppematerjalid
Printed by: Aare Luts
Kuupäev: teisipäev, 22 mai 2012, 08:35

Sisukord

1. Pildi geomeetria muutmine.



Siin peatükis vaatame me pildi pikslite koordinaate, koordinaatide teisendusi ja pildi geomeetriliste moonutuste kõrvaldamist. Loomulikult, piksel ei ole mitte lihtsalt piksel kui niisugune, piksliga käib alati kaasas piksli omaduste vektor. Just need omaduste vektorid moodustavad pildi. Omaduste vektor võib sisaldada mistahes omadusi, milliseid on peetud vajalikuks konkreetse piksliga siduda. Kui me vaatame ainult pildi nähtavat kuju ja mitte pildiga kaasneda võivaid muid omadusi, piisab iga piksli omaduste jaoks tavaliselt kolmeelemendilisest piksli värvuse vektorist [R, G, B]. Pildi geomeetria teisenduste juures käivad pikslite omadused alati pikslitega kaasas, kuid teisenduste valemitesse neid tavaliselt ei kirjutata. Seega tuleb geomeetriliste teisenduste juures alati meeles pidada et kui me teisendame konkreetse piksli koordinaate, tuleb uude asukohta liikuva piksliga võtta alati kaasa ka piksli omaduste vektor. Seega, esimeses lähenduses tuleb uue asukohaga pikslile omistada vana asukoha piksli omaduste vektor. See on nii ainult esimeses lähenduses, sest nagu näeme allpool, ei pruugi pikslite omaduste ülekandmine alati sugugi triviaalne olla.

Aare.Luts.1@eesti.ee

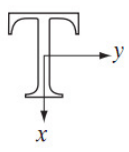
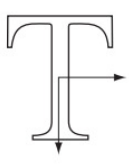
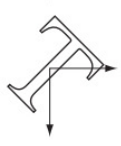
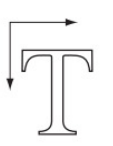
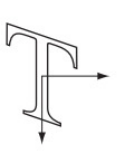
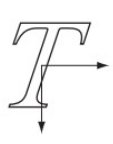
1.1. Geomeetriliste teisenduste põhivalemid.

Vaatame pilti ühe piksli kaupa, ehk vaatame korraga mingit konkreetset pikslit (nagu mainitud, käib selle piksliga alati kaasas omaduste vektor, aga seda pole alljärgnevas kusagil spetsiaalselt ära märgitud). Moodustame piksli koordinaatidest vektori [x, y, 1]. Kaks esimest on piksli x- ja y- koordinaat pildil. Kõige sagedamini mõõdetakse koordinaati pikslite arvuna, arvatuna pildi nullpunktist, aga koordinaati võib mõõta ka mingites muudes pildi mõõtmeid iseloomustavates ühikutes. Koordinaatide vektori kolmas element "1" on toodud sisse valemite üldisuse mõttes. Olgu piksli koordinaadid enne teisendust [v, w, 1] ja pärast teisendust [x, y, 1]. Sellisel juhul saame kirjutada selle konkreetse piksli koordinaatide teisenduse valemi ülaloleval maatrikskujul. [x, y, 1] ja [v, w, 1] on piksli koordinaatide vektorid, **T** on teisendusmaatriks. Piksli teisendusjärgsete koordinaatide vektor arvutatakse sellest valemist, kasutades maatrikskorrutise reegleid (vt siintoodud näidet). Allpool on toodud tabel, mis sisaldab teisendusmaatriksi **T** sisu mõnede

$$[x \;\; y \;\; 1] = [v \;\; w \;\; 1]\,\mathbf{T} = [v \;\; w \;\; 1]\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

koordinaatteisenduste jaoks. Et leida kogu pildi kuju pärast koordinaateisendust, tuleb eespool toodud maatriksvalemisse panna konkreetse juhul vajalik maatriks **T** ja rakendada valemit pildi kõigile pikslitele.

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

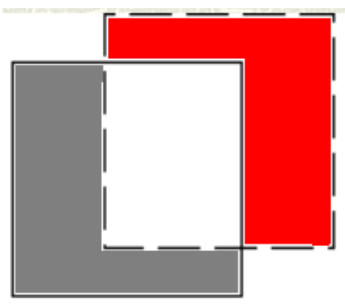
Küsimus: Kasutades ühe konkreetse piksli asukoha teisenduse maatrikskujul esitatud valemit ja teisendusmaatriksi **T** vajalikke kujusid, kirjuta ilmutatud kujul välja koordinaatide teisenduste valemid x=f(u,v) ja y=g(u,v) nihke ja pöörde jaoks.

1.2. Geomeetriliste teisendustega kaasnevad probleemid.

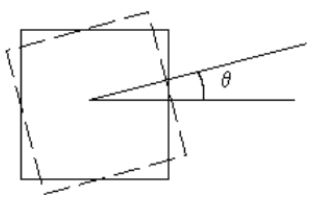
Pildi koordinaatteisendustega võivad kaasneda mitmed probleemid.

1) Kui me **pilti suurendame**, on ilmne et pilt vajab rohkem ruumi, näiteks kui skaleerime nii x- kui ka y-telje koordinaate kahekordseks, saab 64*64 pildist 128*128 pilt. Seetõttu tuleb kogu uue pildi äramahutamiseks varuda piisavalt ruumi.

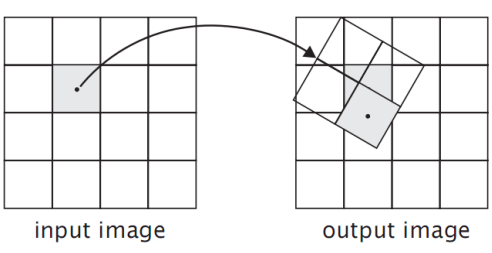
2) Kui **pilti nihutada**, on võimalikud mitmed valikud, mõned pilditöötlusprogrammid ka küsivad, millist varianti rakendada. Pildi suuruse võib jätta samaks, aga see tähendab et osa pildi sisust "läheb pildilt välja" (siin toodud näites: "pildilt läheb välja" pildi punane osa). Kui me seda ei taha, tuleb ka sellisel juhul pilti vajalikul määral suurendada. Seejärel tuleb vastata küsimusele, millega täita endise pildi see osa, kust "pilt minema nihkub" (siin toodud näites: pildi hall osa). Kui me vaatame siin toodud näidet ja pilti ei suurenda ja ei vasta küsimusele, millega täita vabanev osa, saame me tulemuse kus originaalpildilt on alles ainult originaalpildi vasakpoolne alumine nurk, mis on nüüd nihkunud paremasse ülemisse nurka (pildil valge ala), kogu uue pildi vasakpoolne alumine nurk aga omandab juhusliku sisu (siin näites: pildi hall ala). Küsimust ei saa vaadata niimoodi et uus pilt ongi see piirkond, kuhu vana pilt nihkus (näiteks punane pluss valge osa). Kui asja nii vaadata, polegi mingit nihet justkui toimunud sest sel juhul näeks uus pilt välja täpselt samamoodi kui vana. Järelikult, kui me tahame kogu vana sisu ära mahutada, peab uus pilt olema vanast suuremate mõõtudega. Kui me aga ei taha suuremate mõõtudega pilti, on tulemuseks olukord kus osa vana pildi sisust nihkub lubatud piirkonnast välja ja kaob.



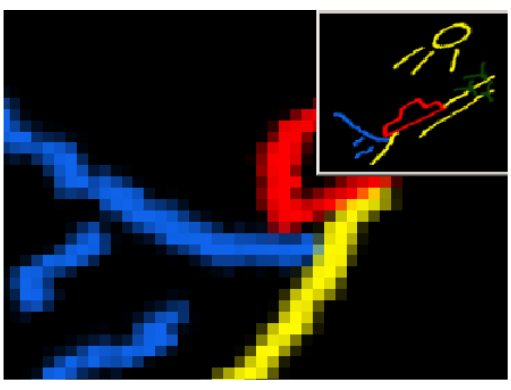
3) Kui **pilti pöörata**, on kogu sisu äramahutamiseks samuti tarvis rohkem ruumi. Nagu näha näitest, peab uus pilt olema nii laiem kui ka kõrgem (uus pilt peab olema nii suur et pööratud ruudu kõik nurgad ära mahuksid). Jällegi ei saa me vaadata olukorda sedaviisi et pööratud pilt ongi see ja ainult see, mida me vajame. Kui me nii teeksime, ei saaks me uuel pildil täheldada mingit muutust. Lisaks tuleb ka nüüd vastata küsimusele, millega täita uue pildi see osa, mis vabaneb pildi vana sisu alt (näites uue pildi kõigi nurkade ümbrused).



4) Kui **pildi geomeetriat teisendada**, nihkuvad pildi pikslid uutele aadressidele. Nagu eespool mainitud, käib iga konkreetse piksliga kaasas selle piksli omaduste vektor, mis tuleb omistada uuele aadressile nihkunud pikslile. Üldjuhul on aga olukord selline et järjekindlalt sel viisil toimides võib juhtuda üks kahest järgnevast. **Esiteks** võivad uue pildi osa piksleid jääda ilma omaduste vektorita, sest nt pildi suurendamisel liigutatakse iga üksik piksel küll uuele aadressile, aga kokku toimub ainult nii mitu nihutamist, kui mitu pikslit vanas pildis oli. Kuna uues, suuremas, pildis on piksleid rohkem, jäävad osa uue pildi piksleid teisendusest puutumata ehk neile ei omistata mingit omaduste vektorit. Kui uus pilt algväärtustati musta taustaga, on sellise olukorra tulemiks uude kohta teisendatud vana pildi fragmendid, milliste vahel on uue pildi algväärtusena kasutatud taust ehk mustad täpid. Sel juhul tuleb muutmatuks jäänud taustapikslitele omistada mingi naabruses leiduv väärtus. **Teiseks** võib juhtuda et uue pildi mõnel pikslil, millele hakatakse vana pildi mõne piksli omaduste vektorit omistama, on omaduste vektor juba olemas. Selline olukord juhtub kindlasti pildi mõõtmete vähendamisel, aga võib juhtuda ka pildi pööramisel. Sel juhul tuleb otsustada, kas kirjutada juba olemasolev omaduste vektor üle või kasutada juba omistatud omaduste vektori ja omistatava omaduste vektori mingit kombinatsiooni.



Valik, millega täita esialgsel teisendusel tühjaks jäänud pikslid või siis, kuidas kombineerida mitmekordselt ülekirjutatavate pikslite omaduste vektoreid, pole sugugi triviaalne. Mõned pilditöötlusprogrammid pakuvad selle koha peal erinevaid valikuid, mõned ei küsi midagi ja teevad asja iseenda parema äranägemise kohaselt. Igal valikul on oma head ja oma vead. Esimene valik on "võtta ühe (lähima) naaberpiksli omadused". Ülekirjutamise puhul tähendaks selline valik et üle ei kirjutata, jääb esimesena salvestatu. Pildi mõõtmete vähendamise puhul tähendaks selline valik et esialgse pildi mõnede pikslite omadused võivad kaduma minna, põhjusel et uues pildis on piksleid vähem ja kõigi vanas pildis esinenud "omaduste" mahutamiseks pole uues pildis enam ruumi. Teine valik on "võtta mingit liiki keskmine (naaberpikslitest)". Pildi mõõtmete vähendamise puhul tähendaks see et uue piksli omaduste vektoriks saab vana pildi mitme piksli omaduste vektorite keskmine. Selle valiku heaks omaduseks on pildi nähtava kuju silumine ja asjaolu et kaduma ei lähe justkui miski, mingi osa vana pildi omadustest säilib (omaduste keskväärtuses) ikka. Paraku tulenevad just sellestsamast ka selle valiku puudused. Kui me võtame hulgaliselt keskväärtusi, määrime me esialgse pildi omadused (nt puhtad värvused) ühel või teisel viisil laiali. Selle tulemuseks on esialgses pildil näha olevate teravate joonte ja servade hägustumine. Tulemust võib käsitleda ka pikslite omadustes leiduva informatsiooni seiskukohast. Oletagem et vana pildi pikslite mingid omadused (nt erkkollane värv) tähendas mingit spetsiifilist nähtust. Kui me uues pildis selle naaberpikslite keskmisi arvutades laiali määrime, siis uues pildis meil erkkollast värvi enam ei ole. Kas sellest tuleb siis järeldada et ka erkkollasega seonduvat nähtust enam ei ole? Nähtus on arvatavasti endiselt olemas, aga see on nüüd avaldunud hoopis mingi(te) muu(de) värvi(de)ga, ja uues pildis tuleb meil alles hakata tuvastama, millise värvi all varem erkkollasega seonduv nähtus nüüd on. Kui me ei arvuta keskmisi vaid võtame ühe naaberpikslite väärtustest, ei tekita me olukorda kus jooned määrduvad ja värvid moonduvad. Tõsi, sel juhul võime me mõnedest pikslitest hoopis ilma jääda.



Olgu siin toodud veel üks näide, kuidas kirkad värvid ja teravad piirjooned võivad geomeetrilistel teisendustel teisenduda ja laiali määrduda. Selguse mõttes soovitan võtta näites toodud originaalpildi ja pärast teisendust saadud pildi ning võrrelda mõne pilditöötlusprogrammi abil pikselhaaval neil piltidel olevaid värvusi. Seda saab teha nii PhotoStyleris kui ka IrfanViews, esimeses on vahend "kraadikaas" ja teises tuleb hiirega pildi soovitud piksli peal klõpsida.

Aare.Luts.1@eesti.ee

1.3. Pildi geomeetriliste moonutuste kõrvaldamine.

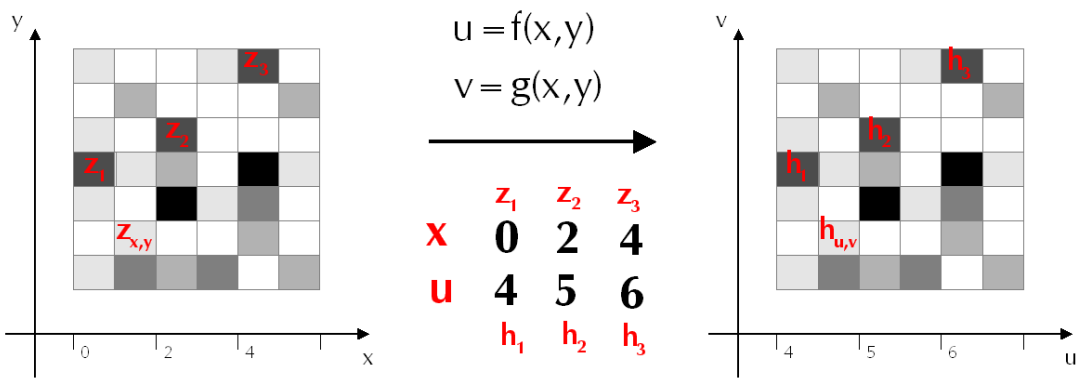
On palju põhjusi, miks samast stseenist või objektist tehtud pildid võivad geomeetrilises mõttes erinevateks osutuda. Siin käsitleme moondeid, mis on põhjustatud nihkest, skaleerimisest ja pöördest. Üks oluline eeldus on veel: geomeetriline muutus peab olema ühesugune pildi mistahes koordinaadi jaoks. Sellisel juhul on geomeetriline teisendus arvutatav lineaarsete teisendustega, järelikult on ka moonutuse kõrvaldamine võimalik lineaarsete võrrandite abil.

Kui pildiga toimunud geomeetrilised teisendused on täpselt teada, sel juhul on mõistagi mõistlik teha täpsed pöödteisendused. Tavaliselt on aga olukord, kus pildiga toimunud teisendused ei ole täpselt teada. Sellisel juhul on pildi geomeetriliste moonutuste kõrvaldamiseks tarvis referentspilti (malli), mille sarnaseks tahetakse moonutatud pilt teha. Mõnikord kutsutakse referentspilti ka kaardiks. Alustuseks defineeritakse kaks koordinaatsüsteemi, näiteks (u, v) ja (x, y). Olgu (u, v) referentspildi koordinaadid ehk need, mille sarnasteks me tahame teist pilti teha. (x, y) oleksid sellisel juhul moonutatud pildi koordinaadid. Selleks et moonutatud pildi koordinaadid teisendada sarnasteks soovitud koordinaatidele, tuleb moodustada teisendusfunktsioonid, funktsioonid võivad olla mistahes matemaatilised avaldised, nt kõrgemat järku polünoomid, nagu kõrval toodud näites. Antud juhul, nagu juba mainitud, eeldame me et pildi geomeetrias toimunud moonutused on lähendatavad lineaarsete võrranditega, seetõttu saame me piirduda kahe muutuja lineaarsete funktsioonidega. Tegemist on just kahe muutuja funktsioonidega, ühest muutujast ei piisa. Põhjuseks on asjaolu et, nagu näha ka eespool toodud teisendusmaatriksitest, x-telje koordinaadi uus väärtus võib sõltuda mitte ainult x-koordinaadi vanast väärtusest, vaid ka y-koordinaadi vanast väärtusest. Seetõttu on ka moonutuse taastamiseks tarvis kahte muutujat: nii x-koordinaadi kui ka y-koordinaadi analooge.

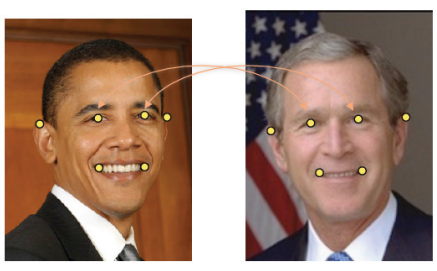
$$u = f(x,y)$$
$$v = g(x,y)$$

$$u = a_o + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2$$
$$v = b_o + b_1x + b_2y + b_3xy + b_4x^2 + b_5y^2$$

Enne kui seda juhtumit konkretiseerma asuda, vaatame veel probleemi, millised funktsioonid võiksid põhimõtteliselt paremad olla (nagu öeldud, antud juhul valime me lineaarfunktsioonid, aga kas põhimõtteliselt võiksid mõned muud paremad olla?) Kõigepealt tuleb vastata küsimusele, kuidas me üldse saame luua seost malli (referentspildi) ja moonutatud pildi vahel. Vaatame allolevat näidet.

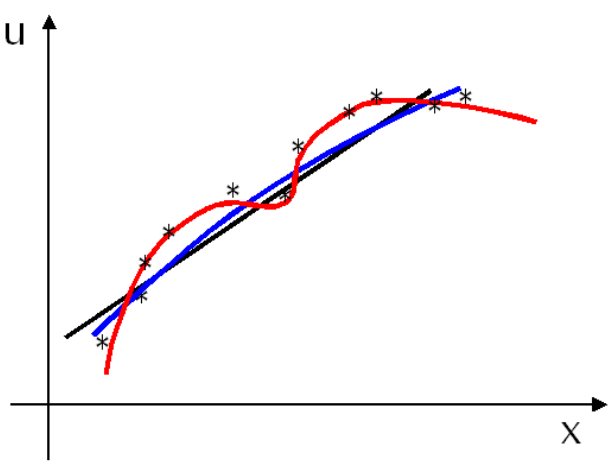


Tegemist on pikslite kujul esitatud väljavõttega kahest pildist, pikslite esitatud kogumid on mõnes mõttes sarnased (saab leida mustreid, mis on sarnased mõlemal pildil nt z_1, z_2, z_3 *versus* h_1, h_2, h_3), aga nendes leiduvate pikslite koordinaadid on erinevad. Teisendusfunktsioon peab looma seose uute ja vanade koordinaatide vahel. Et sellist funktsiooni koostada, on meil tarvis punkte, mis on mõlemal pildil sarnased. Kõrvaloleval pildil on sellisteks punktideks valitud kolm tumedamat halli tooni punkti, valitud punkte nimetatakse referentspunktideks ehk kinnispunktideks, ka "landmarks" or "fiducials". Referentspunktid peavad olema sarnaste mustrite sarnased osad ja nendeks ei sobi näiteks mingi laia lähedast värvi ala keskosa, põhjusel et sellise ala keskosa täpseid koordinaate on halb määrata. Kirjutame välja valitud punktide täpsed koordinaadid nii referentspildil kui ka moonutatud pildil. Antud pildil on on vaadeldud ainult punktide x- ja u-koordinaate ja need koordinaadid kantud tabelisse. Analoogiline tabel tuleb teha ka y- ja v- koordinaatide jaoks.



Loodav funktsioon peab olema suuteline teisendama (x, y) koordinaadid (u, v) koordinaatideks. Nagu mainitud, põhimõtteliselt me võime valida mistahes matemaatilise avaldise. Millist avaldist ma ka ei valiks, suure tõenäosusega saab see avaldis parimal juhul olema selline et teisendab punktide koordinaadid keskeltläbi enam-vähem täpselt, kuid eksib mõne punkti puhul. Eksimuse põhjuseks ei pruugi olla matemaatiline avaldis ise, selle põhjuseks võib olla asjaolu et me eksisime punktide koordinaatide määramisel (või olid punktid nt nii laiali määritud, et täpset koordinaati ei saanudki määrata). Et avaldis ikkagi teisendaks koordinaadid parimal võimalikul viisil, kasutatakse tavaliselt vähimruutude meetodit, meetodi abil leitakse avaldis, mille poolt tehtav viga on vähim võimalikest. Antud juhul, nagu mainitud, otsime me lineaarset avaldist. Enne kui selle juurde minna, vaatame veel korraks ka teistsuguseid avaldisi. Üks võimalik näide on kõrvaoleval joonisel. Tärnidega on kujutatud etteantud punktid, erinevat värvi kõveratega erinevat järku lähenduspolünoomid. Nagu näha, lineaarne kõver (must joon) lähendab punkte "keskpäraselt", aga samas ei erine ühestki punktist liiga palju ja kui punktide asetuse sama tendents kehtib ka väljaspool sellel graafikul kujutatud (x, u)-piirkonda, võib oletada et ta lähendab punkte rahuldavalt ka väljaspool siin kujutatud piirkonda. Kõrgemat järku kõver (punane joon) lähendab olemasolevaid punkte täpsemini, aga kujutatud piirkonna piiridel muudab oluliselt oma suunda. Järelikult võib karta et sellest piirkonnast väljaspool käitub kõrgemat järku kõver täiesti kõlbmatult, mida ei oska karta lineaarse lähenduse puhul. Kokkuvõttes, lineaarne lähendus võib olla keskeltäbi keskpärasem, aga tõenäoselt ei kaldu ta üheski piirkonnas õigest liiga palju kõrvale. Kõrgemat järku lähenduspolünoom võib olla mõnes piirkonnas täpsem, aga mõnes piirkonnas võib ta õigest väga palju kõrvale kalduda, seda eriti juhul kui kõrgemat järku polünoomi määramiseks pole piisavalt punkte.

$$u = a + bx + cy + dx^2 + ey^2 + \ldots$$



Niisiis, asume lineaarse regressioonsirge leidmise juurde. Vaatleme ühe muutuja funktsiooni leidmist. Põhimõte on sama mis kahe muutuja funktsiooni puhul, aga valemid on lihtsamad hoomata.

(1) Iga punkti puhul arvutame vea, mida lähendusfunktsioon teeb. Sulgudes on otsitav lähendusfunktsioon (**a**: sirge tõus, **b**: vabaliige), x on konkreetse punkti koordinaat ühel piltidest ja y on sama punkti koordinaat teisel piltidest. Kui teisendus oleks täpne, peaks lähendusfunktsioon andma tulemuseks täpselt y, paraku võib ta anda mingi muu väärtuse.

(2) Summeeritakse kõigi valitud punktide (kokku **n** punkti) puhul tehtud vead, tulemuseks summa **Q**.

(3) Nõutakse et summaarne viga **Q** oleks minimaalne, otsides seda tingimust rahuldavaid sirge tõusu **a** ja vabaliikme **b** väärtusi.

(4) Nõudest et viga oleks minimaalne tuleneb pärast mõningaid matemaatilisi teisendusi et **a** ja **b** leidmiseks tuleb lahendada toodud võrrandsüsteem.

$$\Delta_i = y_i - (ax_i + b) \quad (1)$$

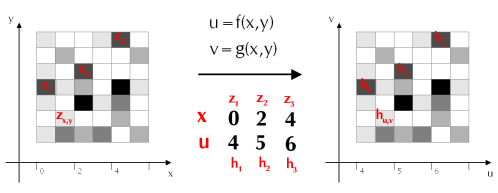
$$Q = \sum_i^n \Delta_i^2 \quad (2)$$

$$\left.\begin{aligned} \frac{\partial Q}{\partial a} &= 0 \\ \frac{\partial Q}{\partial b} &= 0 \end{aligned}\right\} \quad (3)$$

$$\left.\begin{aligned} \left(\sum_i^n x_i^2\right)a + \left(\sum_i^n x_i\right)b &= \sum_i^n x_i y_i \\ \left(\sum_i^n x_i\right)a + nb &= \sum_i^n y_i \end{aligned}\right\} \quad (4)$$

Meie juhul (meil on lineaarfunktsioon, kuid nüüd on rohkem kui üks, nimelt kaks muutujat) on funktsiooni leidmise tee põhimõtteliselt sama, ainult mõnevõrra keerulisem. Õnneks on valmis funktsioonid mõnes tarkvarapaketis juba olemas. Neile valitud punkte ette andes ja neid õigesti kasutades saab leida regressioonsirged, mille abil saab geomeetriliselt moonutatud pildi taastada.

Küsimus: Eespool oleval joonisel oli toodud näitetabel kolme punkti (x, u)- koordinaatide jaoks. Milline oleks lineaarfunktsioon, mis teisendaks x-koordinaadid u-koordinaatideks? Milline oleks funktsioon mis teisendaks u-koordinaadid x-koordinaatideks (seda saab leida ka intuitiivselt)?



Küsimus: eespool oli kirjas "avaldis, mille poolt tehtav viga on vähim võimalikest". Mida tähendab "vähim võimalikest". Mis mõttes "võimalikest"? Alati võib ju ette kujutada mingit matemaatilist avaldist, mis teisendab etteantud arvud veel täpsemini? Või kuidas?

Küsimus: Eespool oleval, ka esimeses küsimuses viidatud joonisel oli toodud näitetabel kolme punkti (x, u)- koordinaatide jaoks. Vaatame nüüd eespool toodud regressioonsirge arvutamise põhimõtte tutvustust, ja selle punkti (1). Võta aluseks joonisel toodud näitetabel ja kirjuta välja selle meetodi rakendamiseks vajalikud (x, y) paarid.

Lisakirjandus: kokkuvõte mitme muutuja lineaarse regressiooni kohta, ingliskeelne loengukonspekt "Geometeric transformations".

Aare.Luts.1@eesti.ee

2. Pildi värvusskaala muutmine.

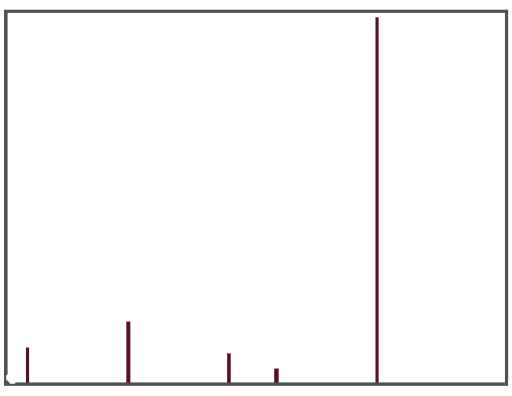
Esimeses teemas oli tegemist digitaalpildi iseloomustussuurustega, neid nimetati ka mitut liiki "lahutusteks". Kui eelmises punktis käsitletud koordinaatteisendused puudutasid pildi geomeetrilist lahutust, siis siinkäsitletav puudutab pildi radiomeetrilist lahutust. Nähtaval kujul avaldub see komponent pildi näiva tonaalsuse ja/või heledusega. Kui räägitakse pildi radiomeetrilisest korrektsioonist, tähendab see pildi nähtava tonaalsuse ja/või heleduse muutmist. Kui räägitakse pildi radiomeetrilisest sarnastamisest teise pildiga, tähendab see nende piltide üldiste tonaalsuste sarnastamist.

Aare.Luts.1@eesti.ee

2.1. Histogrammi olemus.

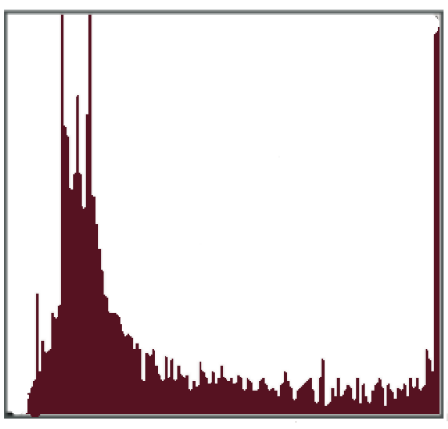
Pildi üldist tonaalsust iseloomustab pildi histogramm. Matemaatilises mõttes näitab histogramm erinevate värvuste/heleduste esinemissagedust pildil, sisuliselt võttes iseloomustab histogramm pildi üldiseloomu (sealhulgas nt pildi tegemisel olnud valgustustingimusi, pildil olevate objektide keskmist värvust).

DISKREETNE histogramm:



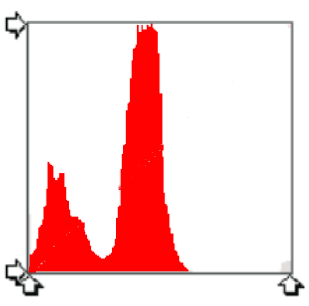
Konkreetse pildi histogramm võib olla nii diskreetne kui ka pidev, seda sõltuvalt pildi iseloomust. Üks diskreetne ja üks pidev

PIDEV histogramm:

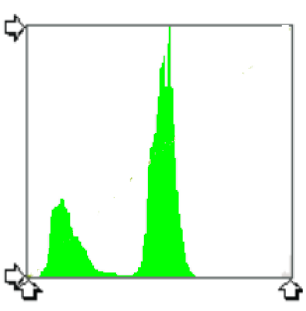


histogramm on näidetes esitatud. Kui valitud pildi tonaalsus moodustub mitmest erinevast elementaarvärvusest, ehk kui pildimaatriks koosneb mitmest (sõltumatust) kihist, siis pildi iga (värvi)kanal moodustab oma (sõltumatu, erineva) histogrammi. Joonistel on toodud valitud pildi R-, G-, B- kanalite histogrammid. Peale nende saab sellisel juhul saab koostada ka eri elementaarvärvuste histogrammide kombinatsioone, millest igaüks iseloomustab mitte ainult selle elementaarvärvuse esinemissagedust, vaid mingi kombineeritud tonaalsuse esinemissagedust pildil.

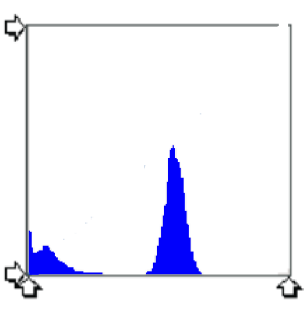
R-histogramm



G-histogramm



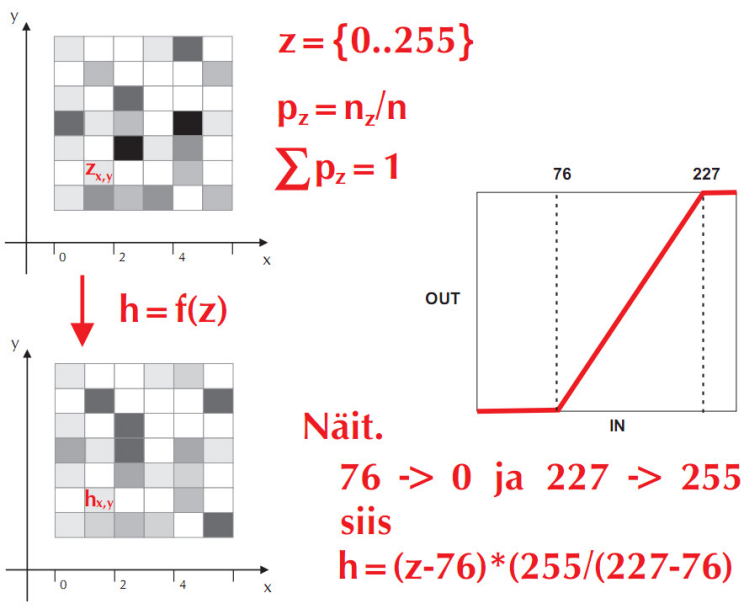
B-histogramm



2.2.

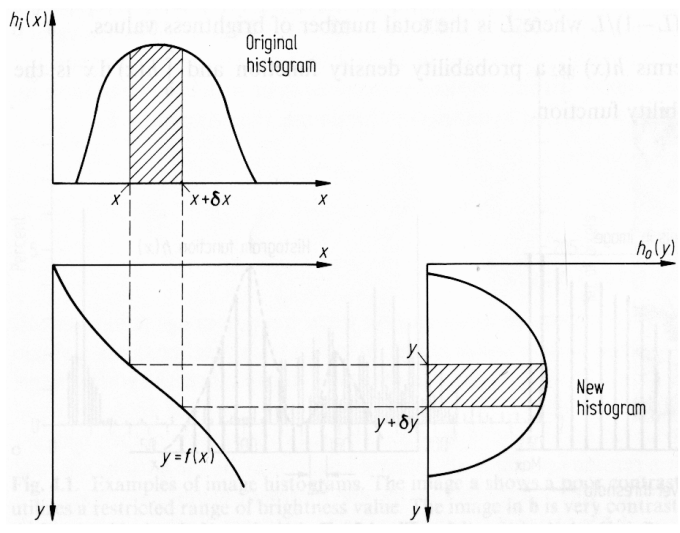
Histogrammiteisenduste olemus.

Pildi värvusjaotuse muutmise teostab värvuste teisendusfunktsioon. Teisendusfunktsiooni rakendamise tulemusena muutuvad nii pildi histogramm,kui ka pildi individuaalsete pikslite värvused. Matemaatiliselt võttes muudab funktsioon järjepanu kõigi pikslite värvusi, võttes argumendiks piksli vana värvuse ja arvutades selle alusel piksli uue värvuse. Kui meil on tegemist halltoonides pildiga, võib rääkida et pigemini ei muudeta pikslite värvusi, vaid muudetakse heledusi. Kuna aga histogramm on pikslite värvustega vahetult seotud, muudab pikslite värvuste muutmine ka histogrammi. Seetõttu ongi otstarbekas rääkida mitte pikslite värvuste teisendamisest, vaid selle asemel rääkida histogrammiteisendusest.

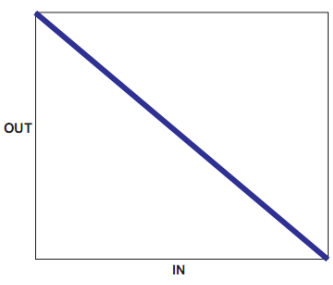


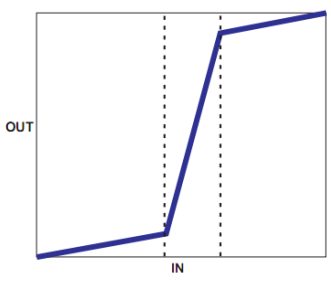
Joonisel on näidatud fragment mingist pildist ja põhimõtteline selgitus, kuidas moodustub (arvutatakse) histogramm ja mida teeb histogrammi teisendusfunktsioon. Z(x,y) on esialgsed heledused kohal (x,y) (võimalikud heledused olgu 0...255), p (Z) on esialgse pildi histogrammi kõrgus kohal Z, antud esituses pole p(z) mitte selle heledusega pikslite arv vaid on selle heledusega pikslite osa (protsent) pikslite koguarvust. Sellise interpretatsiooni korral on kõigi p(z) summa võrdne ühega, põhjusel et ükskõik millist pikslit me ka ei vaataks, mingi heledusega on ta ikka. h(x,y) on teisendusfunktsiooni abil saadavad heledused kohal (x,y), ja h=f(Z) on heleduste teisendusfunktsioon. Olgu näiteks tarvis muuta pildi heledusi nii, et kõik heledused vahemikus [0..76] teisendataks heleduseks 0, kõik heledused vahemikus [227..255] teisendataks heleduseks 255, ja heledused vahemikus [76..227] teisendataks näidatud otspunktidega määratud lõigu poolt. Sellise teisendusfunktsiooni graafiline kuju ja matemaatiline avaldis on samuti toodud vaadeldaval joonisel.

Heleduste teisendusfunktsioon h=f(Z) võib olla mistahes kujuga, seda sõltuvalt vajaliku teisenduse iseloomust. Üks matemaatiliselt ehk üldisem kuju on toodud joonisel. Siin on kirjutis h=f(z) asendatud kirjutisega y=f(x). Põhimõttelist vahet neil kirjutistel pole, oleneb kontekstist, näiteks milliseid tähistusi on enne kasutatud. Sellel joonisel kirjeldab funktsioon y=f(x) "original histogram" teisendust "new histogram"-ks. Joonisel on kujutatud ka üks tähtis omadus, mis kehtib mistahes histogrammiteisenduse korral. Nimelt, mis kujuga ka ei oleks histogrammi teisendusfunktsioon y=f(x), alati kehtib seos h(x)dx =h(y)dy. Oma olemuselt väljendab see pikslite jäävuse seadust: kuidas me pikslite heledusi ka ei teisendaks, kõik esialgsel pildil olevad pikslid saavad ka uuel pildil mingisuguse heleduse. Pole olemas sellist pikslit millel esialgsel pildil oli mingi heledus, aga millel uuel pildil üldse mingit heledust ei ole. Matemaatiliselt on see seos oluline mitmete edasiste valemite tuletamisel.



Küsimus: Mida teevad pildi heledustega neil joonistel kujutatud teisenduskõveratega funktsioonid? Kirjeldada sõnaliselt, näiteks "vasakpoolsel joonisel kujutatud teisendusfunktsioon käitub järgmiselt: 1) Kui esialgne heledus on vahemikus ..., siis ....; 2) ..." (jne).

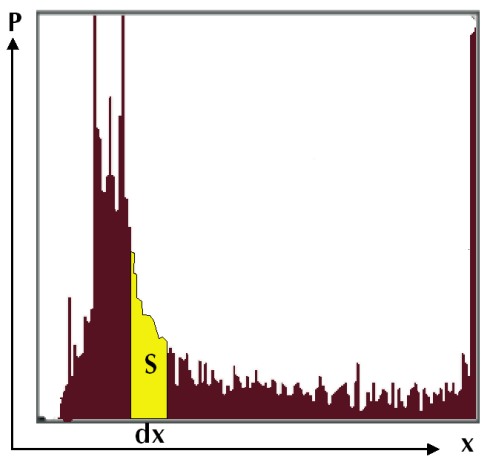




Aare.Luts.1@eesti.ee

2.3. Histogrammi tasandamine.

Inglise keeles on tehte nimetuseks **"histogram equalization".** Tehte eesmärk on matemaatilises mõttes lihtne ja selge. Histogrammi poolt väljendatavaks olulisimaks suuruseks on mingi värvusega (heledusega) pikslite arv, mis avaldub histogrammi kõrgusena mingi tavaliselt x-teljel asuva värvuse/heleduse kohal. Seda kõrgust võib vaadata ka seda värvust omavate pikslite osakaaluna ehk protsendina. Tõenäosusarvutustes tähistatakse tõenäosust tavaliselt p, mistõttu tähistatakse ka histogrammi kõrgust tähega p. Kuna p sõltub värvusest (erineva värvusega on erinev hulk piksleid), siis võib kirjutada p=p(x), tingimusel et värvust tähistatakse x. Histogrammi abil saab arvutada värvuste mingis vahemikus asuvate pikslite koguarvu. Kui tähistame värvuste vahemiku dx, saame S=p(x)*dx. Saadav S on nii värvuste vahemikus dx asuvate pikslite koguarv kui ka selle vahemiku histogrammi alla jääv kogupindala, mis annab ka põhjuse tähistada tulemust tähega S.



Püstitame nüüd nõude et p(y)*dy =const. Siin on x asendatud y-ga, põhjusel et me tahame saada mingit uutmoodi histogrammi ja selleks tuleb teha teatav värvusteisendus. Kui vanad värvused olid tähistatud x, tuleks uusi tähistada teistmoodi, nt y. Otsitavat teisendusfunktsiooni tähistame y=y(x). Joonisel on toodud funktsiooni tuleta mise põhietapid. Kõigepealt, kui pildil on üldse L värvust (heledust) siis on värvustelje diskreet dy avaldatav (L-1)/L. Diskreedi (matemaatilist) põhjendust vt soovi korral lisakirjandusest, aga intutiivselt võib ette kujutada et nii võiks see tõepoolest olla. Näiteks, kui pildil on ainult üks värvus, on diskreet null. Kui värvusi on palju, läheneb diskreet ühele. Mõlemad variandid sobivad. Ja sobib ka see et selliselt alustades saame mõistlikud tulemused.

$\text{Olgu } p_y dy = \text{const} \quad y = 1..L, \text{ siis } dy = (L-1)/L$

$N = \text{pildivälja pikslite arv, seega } p_y(L-1)/L = N/L$

x: vanad heledused, y: uued heledused

vastavad histogrammid p_x ja p_y

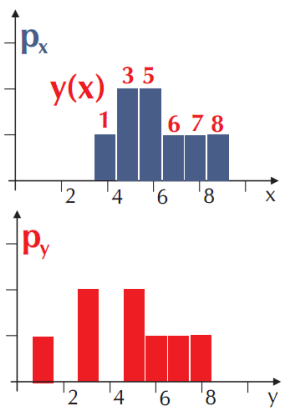
kehtib $p_x dx = p_y dy$

$y = f(x)$: heleduste teisendus

$dy/dx = p_x/p_y = (L-1)/N * p_x$

$\text{Tuleb } y = [(L-1)/N]\int_0^x p_x(u)du$

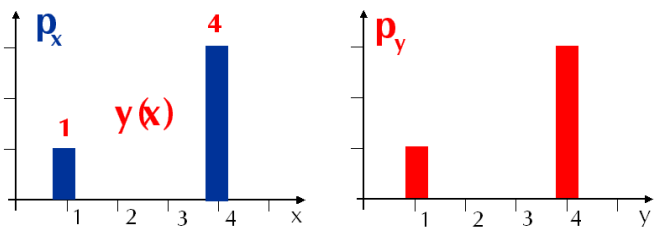
x	1	2	3	4	5	6	7	8	9
p_x				1	2	2	1	1	1
y	0	0	0	1	3	5	6	7	8



Kui kehtiks p(y)*dy =const ja pildil oleks kokku N pikslit, tähendaks see et igasse värvusvahemikku langeks N/L pikslit, seega saab "const" asendada väärtusega N/L. Eespool oli juba märgitud et mistahes histogrammiteisenduse puhul kehtib jäävusseadus ehk p(x)*dx = p(y)*dy. Sellesse asendatakse eestpoolt p(y), mille järel saab avaldada y=y[p(x)]. Selle funktsiooni kuju sisaldab osaintegraale üle esialgsete heleduste (värvuste), integreeritakse nullindana tähistatud värvusest kuni valitud värvuseni. Valemite kõrval olev näitehistogramm illustreerib, kuidas pikslite heledused ja histogramm tervikuna muutuvad. Esialgses pildis on 1 piksel heledusega 4, 2 pikslit heledusega 5, 2 pikslit heledusega 6 ja nii edasi, kokku 8 pikslit. Iga vana histogrammi tulba kohale on märgitud punasega teisendusfunktsioonist tulenev uus väärtus. Kõik need pikslid, mis esialgu omasid punasega märgitud arvu alla jääva tulba heledust, saavad uueks heleduseks punasega märgitud arvu. Näiteks, üks piksel mis esialgu oli heledusega 5, saab uueks heleduseks selle tulba kohal oleva arvu ehk 1, jne. Alumises nurgas on punasega toodud uus histogramm, mis nüüd peaks vastama tingimusele p(y)*dy =const. Tegelikult ta vastab ka, seda nii hästi kui nendest andmetest on võimalik saada. Tulemus ei vasta ideaalile, aga erinevus ideaalist on põhjustatud kahest faktorist. Esiteks tuleneb erinevus sellest et meil on tegemist mitte pidevate vaid diskreetsete suurustega (meenutame, valem tuletati pideva juhu jaoks). Teiseks, erinevus võimendub selle tõttu et sel juhul on meil üldse väga vähe piksleid, väga palju vähem kui peaks olema pideval juhul.

$y = [(L-1)/N]\int_0^x p_x(u)du$

x	0	1	2	3	4
p_x	0	1	0	0	3
y	0	1	0	0	4

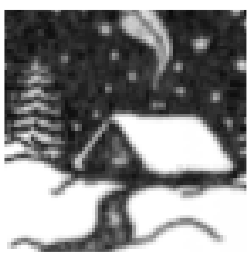


Diskreetse juhu eripäradest tingituna esineb juhtumeid, kus tuletatud teisendusvalem ei muuda üldse midagi, nt kõrvaloleva näite korral.

Küsimused: 1) Milline peaks nägema välja ideaalile vastav "equalized" histogramm? 2) Püüa selgitada, mis mõttes siin näites saadud tulemus siiski läheneb ideaalile, kasvõi selles aspektist et näitad ära et ta on ideaalile lähedasem kui oli esialgne histogramm.

2.4. Histogrammide empiiriline sarnastamine.

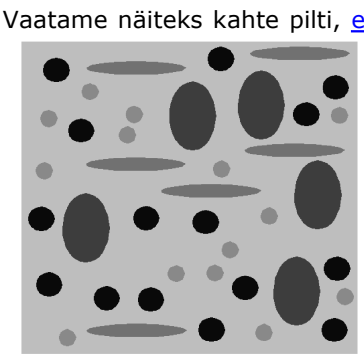
Eelnenud lõigus käsitetud histogrammi tasandamise tehet võib ka praktiliselt proovida, see tehe on olemas paljudes pilditöötlusprogrammides, sealhulgas PhotoStyler-s. Allpool mõned näited, millede põhjal võib vist öelda vähemalt seda et visuaalselt parematek need pildid tasandamise käigus küll ei läinud.

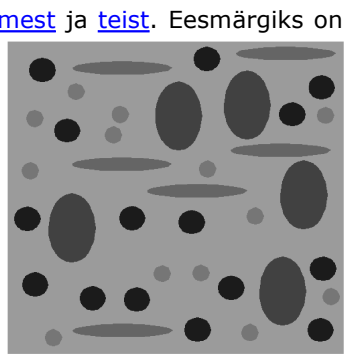


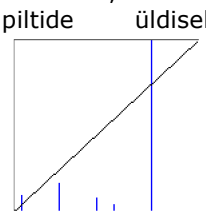
Siin veel üks näide: originaalpilt ja equalized versioon. Ega "equalization" vistiti ei olegi selline tehe, mid tema enda pärast sageli kasutataks. Tema suurim väärtus on ehk mujal: tema kaudu õnnestub sarnastada eri piltide tonaalsused (heledusiseloomud). Alustame siiski kaugemalt, nimelt põhjustest, miks on tarvis pilte tonaalsuse mõttes sarnastada ja seejärel mõnedest käepärastest vahenditest, kuidas saab seda samuti teha, seda küll et mõneti ebatäpsemalt. **Piltide tonaalsuse sarnastamine on oluline** näiteks juhul kui vaadeldavaid

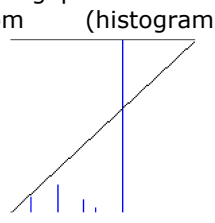
pilte on tarvis võrrelda. Oletagem et pildistatakse teatavat stseeni või teatavaid objekte ja oletagem et on oluline tuvastada piltidel toimunud muutusi. Põhimõtteliselt võib see olla üsna lihtne: kui segavaid faktoreid pole liiga palju, võib muutuste tuvastamiseks pildid üksteisest lihtsalt maha lahutada. Sel viisil leitud erinevus peaks näitama just seda, mis on eri piltidel teistmoodi. Paraku on piltide erinev tonaalsus just üks segavatest faktoritest. Tonaalsuse muutuse põhjusi on palju, nt valgustustingimuste muutus (öö ja päeva vaheldumine).

Vaatame näiteks kahte pilti, esimest ja teist. Eesmärgiks on otsustada kas piltidel on toimunud muutusi või mitte. Esialgsete piltide lahutamine ei anna midagi. Et muutusi (lihtsamini) avastada, tuleb kõigepealt sarnastada piltide üldiseloom (histogrammid).

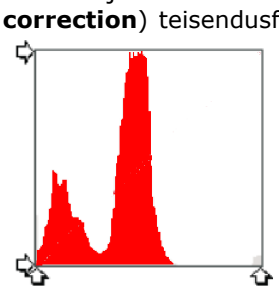




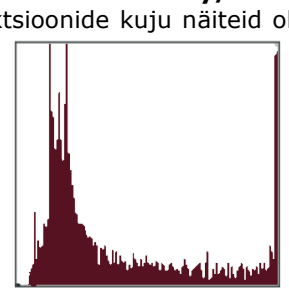




Halltoonides piltide histogrammid on toodud nende kõrval. Sellel konkreetsel juhul läheneme me ülesandele lihtsustustega. Kõigepealt vaatame mõlema pildi histogramme (nt PhotoStyler abil) ja paneme tähele et pildid on erineva kontrastsusega. Kontrastsuse all mõistame seda, kui suur osa kogu võimalikust heledusskaalast on pildis tegelikult kasutust leidnud. Kui histogramm on heledustelje sihis (Photostyler: x-telg) kokku surutud, on pilt madala kontrastsusega ehk hall. Näeme, et vasakpoolse pildi histogramm on x-telje sihis laiemalt asustatud. Küsimuse ilmsemaks selgituseks on toodud veel kaks näitehistogrammi. Punasega toodud histogrammil, mis ei ole seotud üleval näidatud halltoonides pildiga, on ligi 40% võimalikust piirkonnast kasutamata, kasutamata on just heledamad väärtused. Kui histogramm hõlmab kogu heledustelje (nagu paremal näidatud pruuniga antud juhul), on kontrastsus maksimaalselt võimalik. Püüamegi ka halltoonides piltide juures saavutada olukorra, kus mõlema pildi kontrastsus saaks maksimaalseks. Selleks joonistame PhotoStyler histogrammiaknas (aken avaneb menüüst **Tune -> Gray/Color correction**) teisendusfunktsiooni, mis teeks vajaliku teisenduse. Teisendusfunktsioonide kuju näiteid oli



eepool toodud, nüüd tuleb lihtsalt valida sobiv ja see tegelikult joonistada. Joonistamiseks saab kasutada nii x-telje kui ka y-telje sihis liikuvaid liugureid (nagu näha vasakpoolse histogrammi raami juures). Veel kord: punase ja pruuniga histogrammid ei kuulu üleval toodud halltoonides piltidele, punase ja pruuniga toodud histogrammid on toodud lisaks seetõttu et nad on vägagi iseloomulikud. Kui avada nt Photostyler, on kohe näha, kuidas üleval toodud piltide



histogramme eelpool toodud näidete kohaselt muuta.
Olles piltide kontrastsused (enamvähem, me teeme seda käsitsi ja intuitiivselt) sarnastanud, lahutame jällegi ühe pildi teisest. Kui kõik saab tehtud õigesti, on piltide lahutamise tulemus nüüd sootuks teistsugune.

Ülesanne: Teha eelkirjeldatu ise praktiliselt läbi.

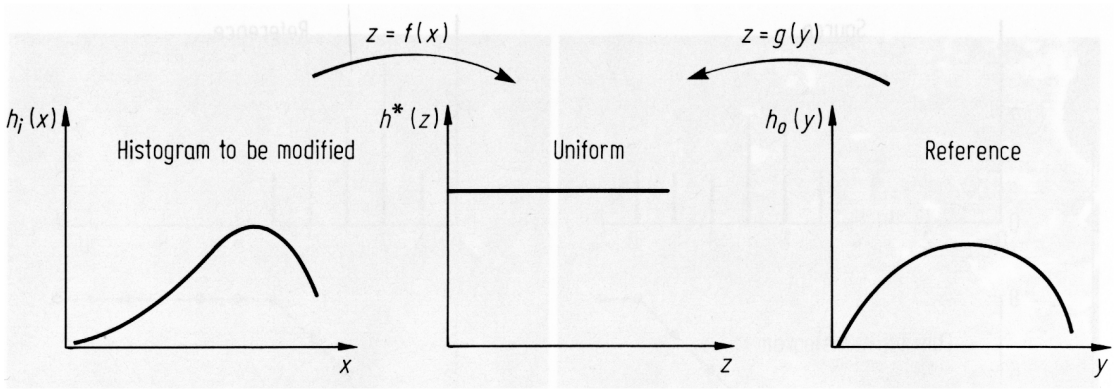
Ebahomogeensete moonutustegurite avastamine ja kõrvaldamine. Eelnenus tegelesime juhtumitega, kus konkreetse piksli tooni muutus ei sõltunud piksli asukohast pildil, tonaalsust muutnud tegur toimis ühtmoodi kogu pildi ulatuses. Kui heledustonaalsuse muutus on tingitud valgustustingimuste muutumisest, ongi tavaliselt nii et muutunud valgustus muudab kogu pildiala ühtmoodi. Kuid alati ei pruugi see nii olla. Tegu võib olla moonutusega, mille omadused sõltuvad asukohast pildil (ebahomogeensed moonutused). Selliste moonutuste kõrvaldamiseks üldist reeglit ei ole, siiski on võimalik nii mõndagi tuvastada ja seejärel ka kõrvaldada. Allpool mõned näited (mõeldud isesesvaks läbimängimiseks). Sellele pildile on ebahomogeenne moonutus kaasa antud, sellele pildile mitte. Viimasel juhul katsuda ebahomogeenne moonutus ise üles leida. Ei tohiks väga keeruline olla kui pildi nurki tähelepanelikut vaadata. Alles lisamoonutuse kõrvaldamise järel on mõtet tegelda histogrammide sarnastamisega. Eesmärgiks on sarnastada mõlemad moonutatud pildid selle pildiga ja otsida seejärel piltidelt võimalikke erinevusi.

2.5. Histogrammide matemaatiline sarnastamine.

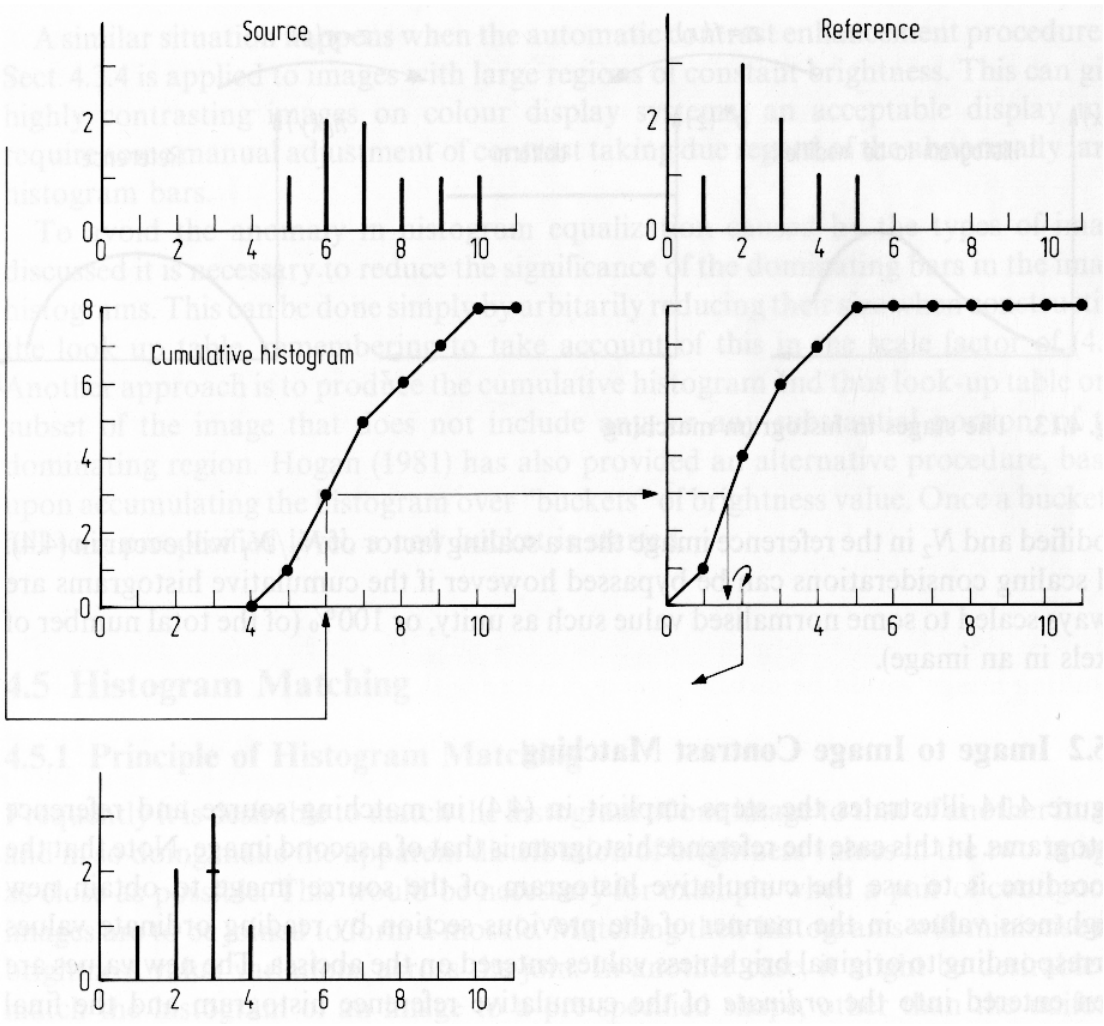
Eespool sai viidatud mõnedele kasulikele poolempiirilistele histogrammiteisendustele. Tegu oli katse ja eksituse meetodiga (kuigi põhimõtteliselt teati, mida tehakse, oli tegelik töö siiski suurelt osalt katsetamine). Seetõttu ei saa seda meetodit üldistada, iga konkreetse juhu jaoks tuleb leida oma meetod.

Hea oleks omada mingit matemaatiliselt paremini kirjeldatavat ja üldisemalt kasutatavat tehnoloogiat. Matemaatiline sarnastamine malliga kasutab eelnevalt kirjeldatud "equalization" tehet oma vaheetapina. Meetodi matemaatiline tuletamine ei ole küll ületamatult keeruline, siiski ei hakata seda siinkohas üksikasjalikult ära tooma. Soovi korral on matemaatikaga võimalik tutvuda nt raamatus *John A. Richards, Remote sensing digital image analysis* (on mitu väljaannet, saadaval ka Tartu Ülikooli raamatukogus). Siin toome ära ainult meetodi põhimõtte ja selle töötamisnäite ühe konkreetse histogrammi korral. Kahjuks tuleb märkida et levinud pilditöötlusprogrammid seda meetodit ei sisalda. See-eest on see olemas kursuse Java-paketis, mille abil on ka plaan mõned näited iseseisvalt läbi töötada.

Siin toodavas üldises, põhimõtet tutvustavas, näites on aluseks kaks histogrammi, "histogram to be modified" ja "reference", viimast nimetame me ka malliks. Alustuseks arvutatakse mõlemast



$y = g^{-1}[f(x)]$ histogrammide tasandatud versioonid, "reference" histogramm teisendub funktsiooniga z=g(y) ja muudetav histogramm teisendub funktsiooniga z=f(x). Soovitud tulemus saadakse nende funktsioonide sobival kooskasutusel, konkreetselt avalduvad soovitud uued heledused y vanade heleduste x kaudu alljärgnevalt. Nüüd vaatame konkreetset, ühe histogrammi teisenemist puudutavat näidet. Teisendatav histogramm on "source" ja mall on "reference". Järelikult, "source" peaks saama võimalikult sarnaseks "reference"-le. Alustuseks arvutatakse mõlema histogrammi "equalized"-versioonid. Seejärel rakendatakse ülaltoodud valemit. Vaatame näiteks esialgse pildi piksleid heledusega 6, neid on esialgses pildis kaks. "Equalized"-



histogrammilt näeme et selle heleduse (x=6) teisendus f(x) on võrdne kolmega. Et saada heledusele 6 vastavat uut heledust, tuleb vaadata g(y) pöördfunktsiooni kohal 3. Näeme et pöördfunkstiooni väärtus on 1 ja 2 vahel, kuid lähemal kahele. Seega, kõik pikslid, mis esialgses pildis olid heledusega 6, saavad uues pildis heleduseks 2. Uuelt histogrammilt näeme et heledusega 2 on kokku 2 pikslit, need on needsamad pikslid mis esialgses pildis omasid heledust 6. Läbi tuleb vaadata esialgse pildi kõi need heledused, millele vastas vähemalt üks piksel. Nende pikslite heledused tuleb teisendada viisil, mis on analoogiline eespool kirjeldatud heledust 6 omanud pikslite teisendamisega.

Küsimus: Vaatame sedasama konkreetset näidet. Miks on uues pildis heledusega 3 kolm pikslit, esialgses pildis pole ju ühtki sellist heledust mida omaksid 3 pikslit? Kust need kolm pikslit siis tulevad?

Teema 3 küsimused on järgmised:

Vastake raamatutes "Pildi geomeetria teisendused" ja "Pildi värvusskaala teisendused" otsesõnu küsitud küsimustele. Vastused arutage rühmas (rühmafoorumis) läbi, ühised vastused vormistage leheküljele "3.teema vastuste ja lahenduste Wiki". Kui nii vastused kui ka lahendused valmis, postitagu rühma üks liige sellekohane teade foorumisse "3.teema vastused valmis". Selle teate järel avatakse seni ainult rühmale näha olev Wiki kogu kursusele ja läheb retsenseerimisele.

3.teema ülesanded koos lisaküsimustega

Vahetekst, lugemiseks ja juhisteks (valgel taustal).
Ülesanded ja küsimused, mis on mõeldud lahendamiseks/vastamiseks (kollasel taustal).

Need ülesanded, mis on allpool mõeldud lahendamiseks Java-paketi abil, võib lahendada ka mingi muu vahendiga, **eeldustel et** retsenseeriv rühm saab neid kontrollida ja et lahendamine saab tähendama uute tehete ja kasutajaliidese tekitamist, mitte pelgalt olemasolevate tehete rakendamist.

1. Pildi histogrammi tekitamine.

Ülesande eesmärgiks on tekitada (Java-paketiga) kujutis sisseloetud pildi histogrammist. See annab juurde histogrammi käsitlemise oskusi, aga on ka kasulik siinsamas allpool, kus tuleb saadud histogramme visualiseerida. Põhimõtteliselt tuleb vajalik pilt sisse lugeda, see kuhugi salvestada (nt mõnda oma massiivi, näited olemas), misjärel tekitada uus pilt, kuhu formeerida histogramm. Histogrammi formeerimine on piisavalt lihtne, kogemuste põhjal saab põhiliseks takistuseks kahe erineva pildi samaaegne kasutamine, aga juba järgmises ülesandes läheb ka seda kogemust vaja.

Lugeda sisse see pilt ja tekitada uus pilt, mile sisuks oleks loetud pidi histogramm. Kontrollimiseks esitada saadud pilt.

2. Pildi geomeetrilised teisendused.

Ülesande eesmärgiks on muuta pildi geomeetria sarnaseks mallile. Sellega modelleeritakse olukorda kus pildi geomeetria on mingitel põhjustel näitepildi omast erinev, piltide (piltidel leiduva) võrdlemiseks on aga tarvis et piltide geomeetriad oleksid sarnased. Tegemist on kahe pildiga: malliga ja pildiga, mis on tekitatud mallist lineaarsete geomeetriliste teisendustega. On tarvis teha pöördteisendus(ed), aga esialgsed teisendused pole teada, mistõttu tuleb rakendada referentspunktide meetodit. See meetod on piisavalt üldine selleks et analoogilist taastamist rakendada ka praktikas ettetulevate ülesannete juures. Lisaks: tavalistes pilditöötluspakettides sellist tehet pole, seega saab see olema miski, mis on ka selles mõttes uus.

Muuta pildi geomeetria sarnaseks malli geomeetriale. Tulemus (pilt) salvestada. Kontrollimiseks esitada saadud pilt. Nagu teoreetilises osas märgitud, on geomeetria taastamiseks vaja valida referentspunktid, määrata referentspunktide koordinaadid ja moodustada funktsioon, mis tekitaks parimal võimalikul viisil koordinaatide vajaliku teisenduse. Ka siin kasutame lineaarfunktsioone. Vajaliku funktsiooni võib ka ise arvutada, aga seda saab lihtsamini teha nt Excel-s, kasutades paketi "Data Analysis" funktsiooni "Regression". Kui seda paketti konkreetses Excel-s ei ole, tuleb pakett aktiviseerida aknas "Add-ins", valides seal "Data Analysis". Kui funktsioon on (Excel abiga) leitud, tuleb see funktsioon programmeerida Java-paketis (arvestada piltide suuruse võimaikku muutust, selleks tuleb defineerida eri suurustega massiivid, vaikimisi on kõik massiivid ühe suurusega), misjärel saab pildi geomeetria taastada. Pildi taastamisel tuleb arvestada ka sellega et uue pildi ükski piksel ei jääks tühjaks (pikslite täitmist vt ka teooriast). Võib olla kasulik tekitada lisa indeksmassiiv, mille ainus ülesanne on näidata, kas mingi piksel on juba täidetud või mitte.

3. Pildi värvustonaalsuse taastamine tavavahenditega.
Eesmärgiks on saavutada olukord kus piltidel leiduva info muutusi oleks võimalik pildide võrdlemise abil tuvastada. Teoreetilises osas anti ülesanne läbi töötada teatav juhtum halltoonides piltidega. Aga oli ka mainitud et värviliste piltide puhul on olukord selle võrra keerulisem et tuleb sarnastada mitte üks histogramm

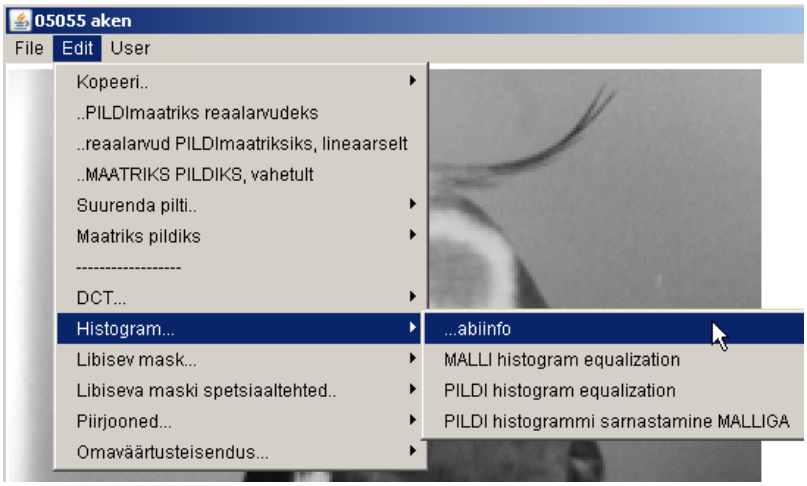
vaid kõigi värvuskanalite histogrammid. Siin ongi tegemist sellise juhtumiga. Piltide sarnastamine ja võrdlus pakutakse teha PhotoStyler vahenditega.

Otsida piltidel pilt 1, pilt 2 ja pilt 3 esinevaid muutusi, kasutades piltide paarikaupa võrdlemist. Arvatavasti tuleb enne võrdlemist piltide histogrammid (enamvähem) sarnastada.

Kontrollimiseks saata võrdlustulemustena saadud pildid koos tekstilisi kommentaare sisaldava failiga.

4. Pildi histogrammi sarnastamine malliga.

Sisaldab kolme ülesannet, kõigi eesmärgiks on veenduda et malliga sarnastamist lubav tehe seda tõepoolest ka teeb, ühtlasi annavad ülesanded veel kord võimaluse harjutada piltide histogrammi soovitud muutmist. Ülesannetes on mõeldud kombineerida PhotoStyler ja Java-paketi võimalusi. Java-paketis avaneb histogrammi sarnastamise tehe kõrvaloleval joonisel näidatud menüü alt (loe ka näidatud "abiinfo"-t).



1) Muuta pildi 1 histogramm sarnaseks pildi 2 histogrammiga. Veenduda et histogrammid tõesti said sarnasteks (sarnasemaks).

2) Muuta pildi 1 histogrammi, näiteks vähendades selle kontrasti. Seejärel pildi histogramm taastada. Veenduda et taastatud pilt tõesti on esialgsega sarnane.

3) Selles ülesandes käsitletakse samu pilte, mida vaadeldi teoreetilise osa ühes ülesandes. Nüüd aga ei sarnastata histogramme käsitsi, nüüd sarnastatakse histogrammid programmiliselt. Sarnastada pilt 1 ja pilt 2 histogrammid, kasutades selleks histogrammide matemaatilise sarnastamise tehet. Tulemusena võib selguda et histogrammid ikka pole päris sarnased. Sel juhul küsimus ja lisaülesanne: millest on tingitud et histogrammid ei saanud esimesel katsel päris sarnasteks (meenutada teooriat, eriti seal toodud konkreetse histogrammi näidet)? Kui silmas pidada just äsja vaadeldud põhjusi ja matemaatilist meetodit oskuslikult kasutada, saab ka sel juhul histogrammid sarnasteks. Lisaülesanne: kasutada matemaatilist meetodit niimoodi (võimalik et mitu korda) et histogrammid saaksid sarnasteks.

Retsenseerimiseks esitada kõik vajalikud materjalid, et retsensent saaks tehtut oma arvutil kontrollida.

Jõudu ja edu!

4.teema õppematerjalid

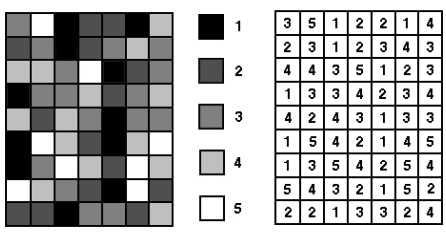
Selles raamatus antakse ülevaade pildi teisendamisest sagedusruumi, nagu ka uue ruumi mõningatest (kasulikest) omadustest. Pildiinfo erinevates ruumides. 1. Fourier' teisendus pideval juhul (FT). 2. Fourier' teisendus diskreetsel juhul (DFT). 3. Sidum ja korrelatsioon. 4. Pildi servade probleem. 5. Diskreetimisvõre vähim lubatav sagedus. 6. DCT teisendus. 7. Sagedusruumi omaduste kasutusnäiteid.

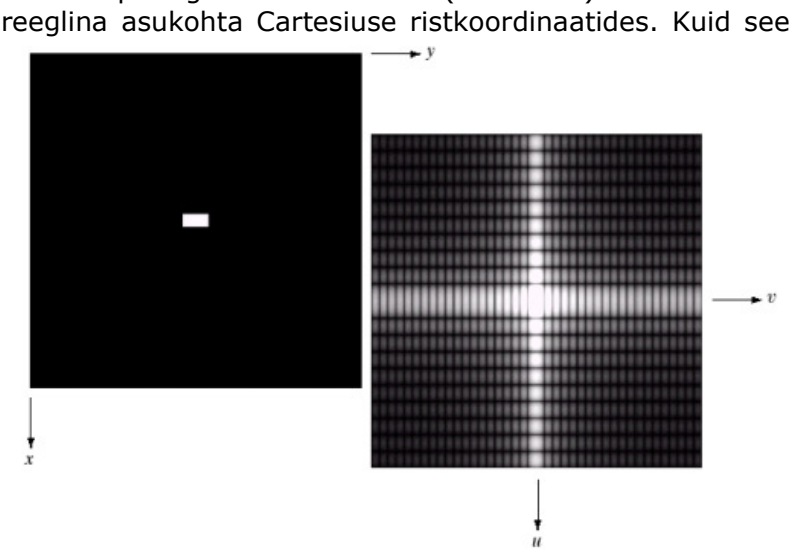
Õpikeskkond: TÜ Moodle
Kursus: Pildiinfo töötlus (LOFY.05.055)
Koosta raamat: 4.teema õppematerjalid
Printed by: Aare Luts
Kuupäev: teisipäev, 22 mai 2012, 08:29

Sisukord

Pildiinfo erinevates ruumides.

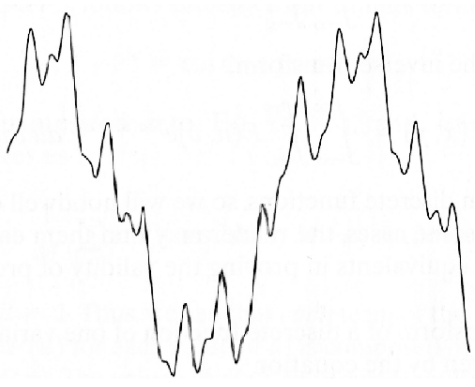
Nagu mainitud esimeses teemas, ei ole digitaalpilt tema vahetul matemaatilisel kujul midagi muud kui arvmaatriks, või siis teatav hulk omavahel seotud arvmaatrikseid. Maatriksites endis pole kusagil kirjas, kuidas on neis sisalduv (info) seotud reaalse maailmaga. Et seostada arvmaatrikseid reaalsete nähtustega, on tarvis lisainfot, kuidas tuleb maatriksites sisalduvaid arve interpreteerida. Näiteks tegeldi värvuste ruumidega ehk seosega maatriksis (maatriksites) teataval kohal (pikslis) sisalduvate arvude ja värvuste vahel. Kui arvmaatriksitega seotud värvuste ruum on määratud, saab ühtlasi ka aimu, mis värvusteks tuleks maatriksites olevad arvud konverteerida. Mida samuti mainiti, oli pildimaatriksite ridade ja veergudega seotud lahutused. Kui need on määratud (maatriksitega kaasa antud), saab leida igale konkreetsele pikslile vastava ala asukoha reaalses ruumis (nt meetrites, arvates etteantud nullpunktist maapinnal). Võib öelda et pilt on antud teatava funktsiooniga V=V (x,y), kus x ja y määravad konkreetsele pikslile vastava ala asukoha koordinaatruumis ja V on selle piksliga seotud väärtus (nt värvus). Vähemalt "tavalisel" pildil tähendavadki x ja y reeglina asukohta Cartesiuse ristkoordinaatides. Kuid see ei pea nii olema. Matemaatikast on teada mitmed muud koordinaatsüsteemid, nt polaarkoordinaadid, kus punkti asukohta määrab paar (kaugus nullpunktist, nurk nullsuuna ja punkti asukoha vahel). Punkti asukoha teisenduseks ühest süsteemist teise on olemas teatavad valemid (punkti asukoht teisendusega tegelikult ei muutu, muutuvad ainult asukohta kirjeldavad arvud ehk koordinaadid). Teatava piksli (konkreetse värvusega seotud punkti) koordinaadid võib anda ka muudes süsteemides, üks teisendus allpool kirjaldatavasse ruumi on toodud ülaloleval pildil. Matemaatiliselt on oluline ainult see et uus koordinaadistik (ruum) rahuldaks teatavaid nõudeid, nt nõutakse ruumi baasvektorite ortonormaalsust. Oluline on mõistagi ka see et me oskaksime ühest ruumist teise minna, nagu ka küsimus, on uus ruum ikka kasulikum kui vana.

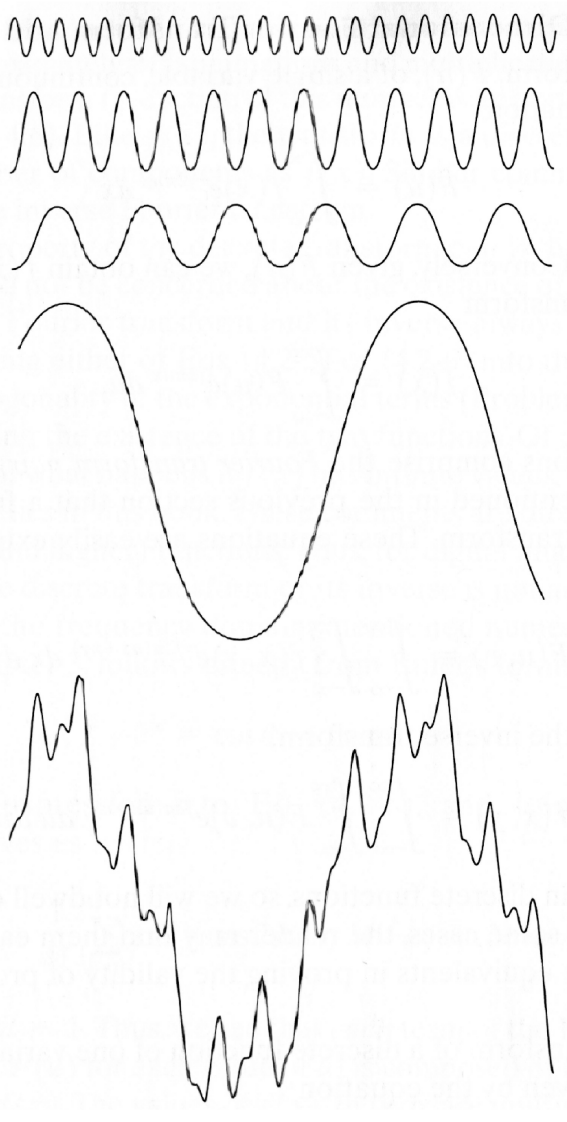




1.Fourier' teisendus pideval juhul (FT).

Olgu meil mingi funktsioon, mille poolt kirjeldatav heleduse muutus nt x-telje sihis on kujutatud pildil. Selle funktsiooni kuju näib üsna ebakorrapärane ja raskesti kirjeldatav. Pannes aga selle funktsiooniga kohakuti erineva sageduse ja erineva amplituudiga harmooniliste võnkumiste graafikud, hakkab tunduma et neid sobival viisil





kombineerides võiks ehk vaadeldava funktsiooni kuju tuletada. Ja nii ongi: kõik teatavatele tingimustele vastavad funktsioonid saab esitada harmooniliste võnkumiste kaalutud summaga. Tähendab, kui meil ennist oli funktsioon argumendist, mida sai mõõta nt meetrites (pildi x-telg seostatuna asukohaga reaalses maailmas), siis nüüd on meil uus telg, x-telje analoog, mida saab mõõta sagedustes. Funktsiooni teatavat sagedusesitust nimetatakse funktsiooni esituseks Fourier' ruumis, vastavat teisendust Fourier' teisenduseks. Sedaliiki sagedusesitusel on veel üks omapära, mis viib ta tavatajust kaugemale: nimelt on selles ruumis esitatud funktsioonidel (piltidel) nii reaal- kui ka imaginaarosa. Imaginaarosa võib komplitseerida nii arusaamist kui ka teisendatud piltide (visuaalset) esitamist, sest üldjuhul tuleks esitada nii reaal- kui ka imaginaarosa, ehk ühest pildist tekiks justkui kaks pilti. Milleks seda igapäevaelu mõistes raskesti hoomatavat uut ruumi vaja on? Üsna üldiselt võiks vastata et sagedusruumis näevad paljud asjad välja teistmoodi, ja seetõttu võib juhtuda et nad on lihtsamini käsitletavad. Esmapilgul võib tunduda et tavaruumis lihtsad objektid (punkt, ruut) näevad uues ruumis välja keerulised, ju siis võib arvata et mõned tavaruumis keerulised objektid on ehk lihtsad. Ülalpool oli toodud üks näide ristküliku kujust Fourier' ruumis. Mõned ütlevad et ta näib nüüd hoopis ilusam. Alljärgnevalt on toodud pideva juhu teisendusvalemid, seda nii otse- kui ka pöördteisenduse jaoks, mõlemad nii ühe- kui ka kahemõõtmelises variandis. Originaalfunktsioone (kas *f(x)* või *f(x,y)*) vaadeldakse kogu arvteljel (kas *x* või *x* ja *y*) ning ka uue, Fourier'- ruumi funktsioonide (kas *F(u)* või *F(u,v)*) argumentideks on kõikvõimalikud

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\,dx \qquad f(x) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ux}\,du$$

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy \qquad f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

sagedused (kas *u* või *u* ja *v*). Tegelikult pole uue ruumi baasfunktsioonideks mitte sagedused vaid (ühemõõtmelisel juhul) moodustised

$$e^{j\theta} = \cos\theta + j\sin\theta$$

exp(-j·2¶·u), kus *j* on imaginaarühik, kuid kuna eksponent kompleksarvust avaldub reaalosa cos ja imaginaarosa sin summana, on baasfunktsioonide vaatlemine sagedustena igati mõistlik.

Aare.Luts.1@eesti.ee

2. Fourier' teisendus diskreetsel juhul (DFT).

Kui pideval juhul oli tegemist nii argumendi *x* kui ka funktsiooni *f(x)* lõpmatu arvu väärtustega, siis diskreetsel juhul saab tekitada lõpliku pikkusega read, mis sisaldavad argumendi *x* kõiki võimalikke väärtusi kui ka funktsiooni *f(x)* kõiki võimalikke väärtusi. Meenutame esimesest teemast, kuidas toimus digitaalsignaali loomine. Pidevsignaalile asetati võre ja võre iga sõlme kohal salvestati pidevsignaali selles kohas olev väärtus. Üldiselt võetakse võre selline et tema kahe mistahes järjestikuse sõlme vahekaugused on ühesugused. Järelikult saab funktsiooni *f(x)* diskreetsel kujul esitada alljärgneva jadana.

$$\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \ldots, f(x_0 + [N-1]\Delta x)\}$$

Selles jadas on seos tegeliku maailmaga sees ainult ühes kohas: ruumi diskreedis Δ*x*, mida aga funktsioonide esitamisel ilmsi enam kusagil ei mainita. Diskreetsel kujul esitatud funktsiooni *f(x)* ainsaks ilmsi esitatavaks argumendiks on dimensioonitu arv

$$f(x) = f(x_0 + x\,\Delta x)$$

x, sisuliselt punkti järjekorranumber jadas. Fourier' teisendusvalemid diskreetse pildi jaoks on toodud alljärgnevalt.

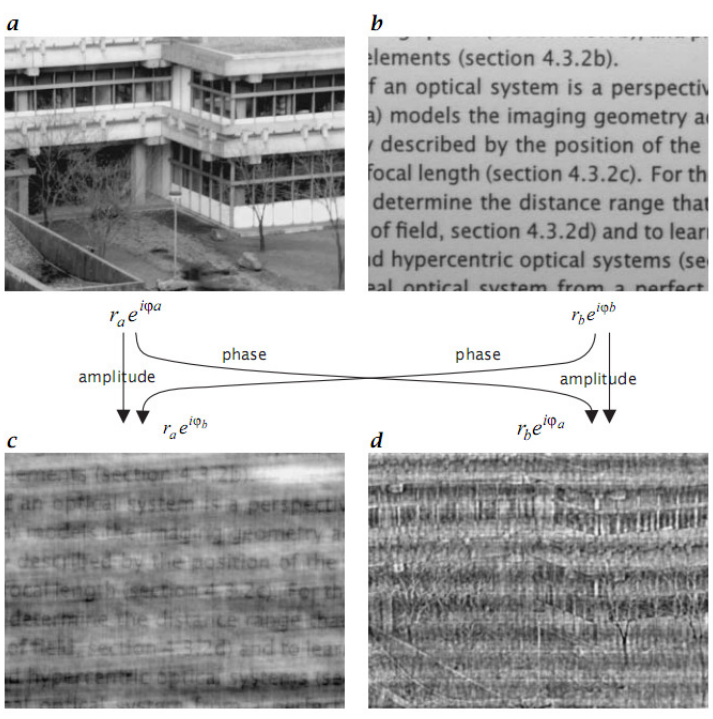
$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

Kui vaadata teisendusvalemeid, saab uue Fourier' ruumi mõnda liiget üsna lihtsasti interpreteerida. Näiteks saab lausa peast läbi arvutada, milline on uue ruumi nullinda liikme

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

F(0,0) sisu. Seejärel on aimatav, mida sisaldaksid järgnevad liikmed, nt mille poolest erinevad liikmed ***F(0,1)*** ja ***F(1,0)***. Liige ***F(M-1,N-1)*** aga kirjeldab esialgsel pildil toimunud kõige kiiremaid muutusi. Ise võiks mõelda, kuidas need (kõige kiiremad) muutused esialgsel pildil (visuaalselt) avalduvad. Ehk kui küsida teiste sõnadega, milline peaks olema esialgne pilt et tema esitusel Fourier' ruumis oleks liige ***F(0,0)*** väike ja liige ***F(M-1,N-1)*** oleks suur?

Pildi Fourier' ruumis avalduva kuju täpsem interpreteerimine võib küll valmistada teatavaid raskusi, kasvõi selle tõttu, et uue "pildi" pikslite "värvused" ***F(u,v)*** sisaldavad nii reaal- kui ka imaginaarosa. Nii reaal- kui ka imaginaarosa on olulised. Järgnevalt üks näide, millise osa pildil leiduvast infost kannab edasi reaal-, milline imaginaarosa. Näidetes olevad pildid on saadud sel viisil et mõlemad esialgu koordinaatruumis olevad pildid on teisendatud Fourier' ruumi. Seejärel on tulemuste reaal- ja imaginaarosad ära vahetatud ja sel viisil saadud kombinatsioonid on teisendatud tagasi koordinaatruumi. Nagu näha, on nii reaal- kui ka imaginaarosa kandnud edasi teatavat spetsiifilist infot esialgsest pildist. Fourier' ruumi teisendatud



"pildi" visualiseerimiseks kasutatakse tavaliselt kompleksarvu moodulit ***¦F(u,v)¦*** või siis "power spectrum" ***P(u,v)***; mõnikord kasutatakse siiski ka faasi ***Φ(u,v)***. Kasulik on teada ka koordinaatruumi ja Fourier' ruumi tähtsamaid seoseid, nagu ka Fourier' ruumi tähtsamaid omadusi. Mõned seostest ja omadustest on toodud selles tabelis.

$$|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2}$$

$$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$$

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$$

3. Sidum ja korrelatsioon.

Üheks väga oluliseks seoseks on sidumi (ingl. convolution) ja elementkaupa korrutise (ingl. pointwise product) ekvivalentsi seadus (convolution theorem). See ütleb et kui meil on kaks funktsiooni (pilti), siis neist samasuguse kolmanda võime tekitada kahel viisil. Esiteks, kolmanda pildi tekitamiseks võib ühele kahest pildist rakendada sidumit teise pildiga, seda koordinaatruumis. Teiseks, samasuguse kolmanda pildi võime tekitada ka sel viisil et kõigepealt viime mõlemad, nii esimese kui ka teise pildi Fourier' ruumi, seejärel korrutame mõlema pildi Fourier' pöörded element elemendi kaupa läbi (**see pole skalaarkorrutis**), ning lõpuks toome korrutise Fourier' ruumist koordinaatruumi tagasi. Teine variant kõlab küll pikemalt, aga mõnel juhul ei pruugi ta sugugi halvem olla. Ning kehtib ka vastupidine: kahe funktsiooni (pildi) sidum Fourier' ruumis on ekvivalentne samade funktsioonide koordinaatruumi kujude elementkaupa korrutisega. Seega võib samaväärse tehte teha nii koordinaat- kui ka sagedusruumis, samaväärse tehte saab teha nii kasutades sidumit kui ka korrutamist, millist varianti eelistada, sõltub ülesandest ja ka kasutaja maitsest.

$$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v)$$

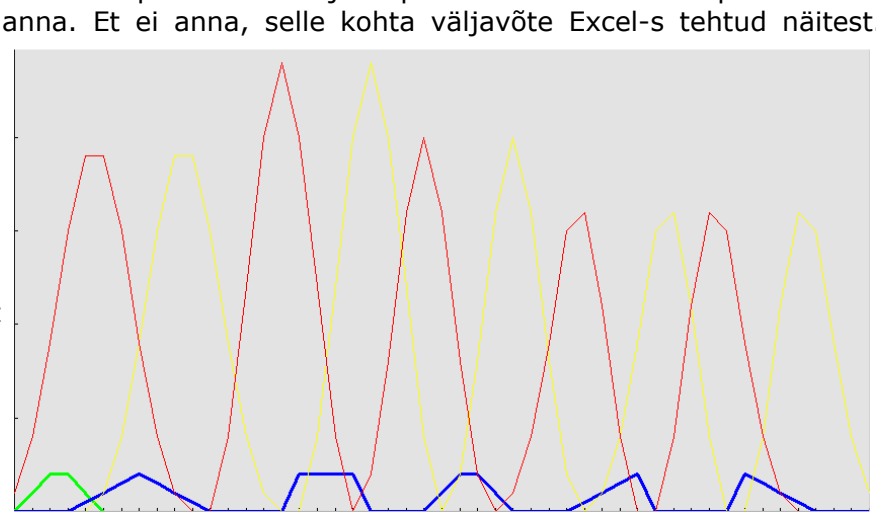
Järgnevalt on toodud piltide (3D funktsioonide) sidumi arvutamise valemid nii pideva kui ka diskreetse juhu jaoks. Sidumi arvutamist võib kujutada ette niimoodi et kui meil on kaks funktsiooni, tuleb kõigepealt teine neist argumenttelgede suhtes peegeldada ja seejärel hakata teda esimese funktsiooni suhtes mööda argumenttelgi liigutama. Funktsioonide igas asendis võetakse nende selles kohas asuvate väärtuste korrutised ja liidetakse kõik sel viisil saadud korrutised kokku. Korrelatsioon erineb sidumist märgi poolest, nüüd teist funktsiooni eelnevalt ei peegeldata. Korrelatsiooni korral võib ära mainida et temast ei tasu oodata seda, mida tema nimi justkui lubaks.

$$f(x, y) * g(x, y) = \int_{-\infty}^{\infty}\int f(\alpha, \beta)g(x - \alpha, y - \beta)\, d\alpha\, d\beta$$

$$f(x, y) * h(x, y) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m, n)h(x - m, y - n)$$

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{\infty}\int f^*(\alpha, \beta)g(x + \alpha, y + \beta)\, d\alpha\, d\beta$$

Nimi "korrelatsioon" võib viidata sellele et temalt võiks oodata abi sarnaste objektide tuvastamisel, näiteks et kui võtta aluseks pilt, kus on hulk mitmesuguseid objekte ja kui võtta teiseks pildiks huvipakkuvat objekti kirjeldav pilt, siis annaks nende piltide siin kirjeldatud korrelatsioon infot teisel pildil oleva objekti paiknemisest esimesel pildil. Sellist infot see korrelatsioon ei anna. Et ei anna, selle kohta väljavõte Excel-s tehtud näitest. Sinisega antud joont võib vaadelda kui lõiget pildist, mis sisaldab mitmesuguseid objekte (erineva kujuga kolmnurki ja trapetseid). Rohelisega antud joont võib vaadelda teise pildina, mis sisaldab ainult üht, huvipakkuvat objekti. Nagu näha, on huvipakkuva objektiga (roheline joon) sarnane fragment sinisel joonel täiesti olemas, aga ei korrelatsioon (punane joon) ega ka sidum (kollane joon) ei näita selgelt ära, kus see fragment asub. "Korrelatsioon" siin ei ole täpselt see mis "korrelatsioon" statistikas, aga kuna seda mõistet kasutatakse ka sidumi kõrval, on otstarbekas mõiste see kasutus vähemalt ära nimetada.

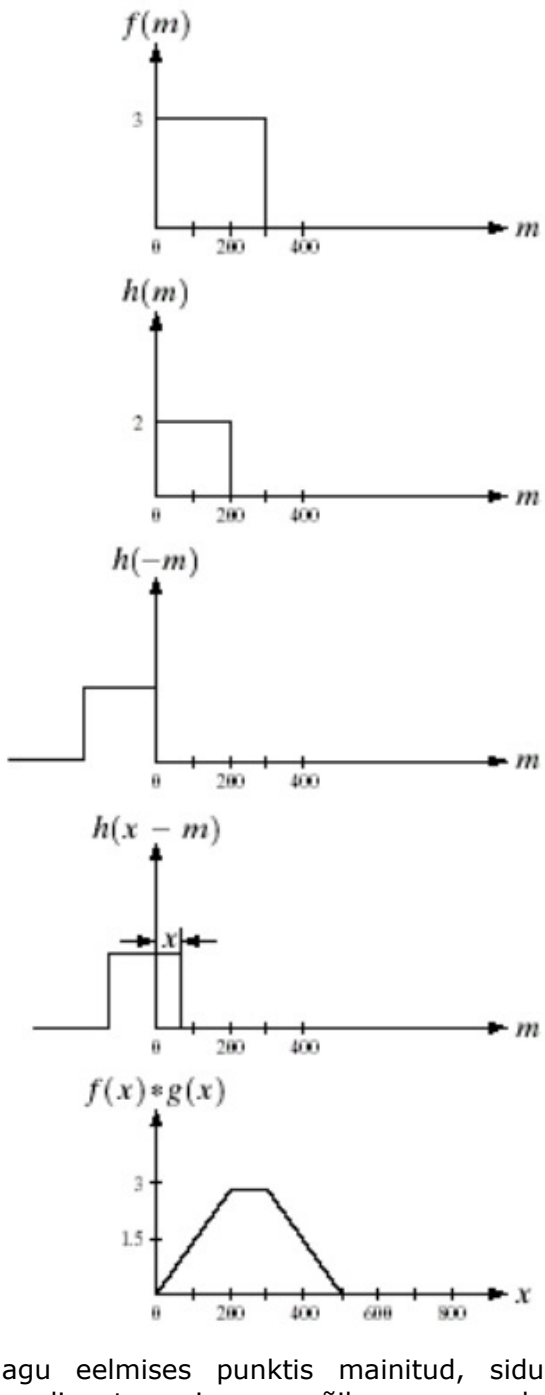


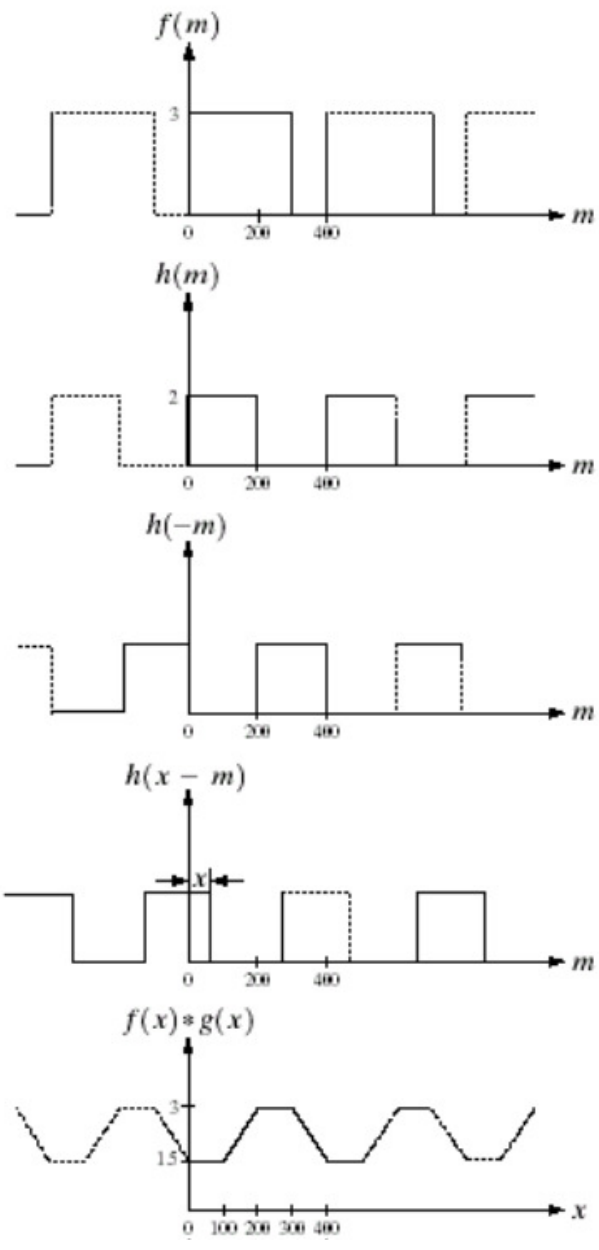
Sidumi ja korrutise ekvivalentsi seadus on oluline nii siis, kui me tahame teda kasutada, kui ka siis, kui me ei taha teda kasutada. Viimasel juhul tuleb meeles pidada et ta toimib hoolimata sellest, tahame me seda või mitte, ja tema tõttu esinevad mitmed efektid, mida ei oskakski ehk esimese hooga arvata, neist mõnest edaspidi.

Aare.Luts.1@eesti.ee

4. Pildi servade probleem.

Olgu meil kaks signaali (nt lõiget piltidest) nagu kujutatud vasakus tulbas oleval joonisel. Mõlema signaali pikkus on 400 punkti, millest esimene signaal *f(m)* on nullist erinev 300 punkti jooksul ja teine signaal *h(m)* on nullist erinev 200 punkti jooksul; arvutatakse nende sidum. Sidumi arvutamise valemit järgides on selge et tulemus ei mahu enam 400 punkti sisse ära. Sidumi korrektse tulemuse mahutamiseks on tarvis rohkem punkte. Kui arvestada ainult signaali nullist erinevat osa, siis on tarvis vähemalt 500 punkti. Kui arvestada kõiki punkte, mille jaoks midagi arvutati (null korda null on ka tulemus, mis siis et null), siis tuleb sidumi tulemuse vajalikuks pikkuseks 800 punkti. Järelikult, kui arvutada kahe suure pildi sidum, on tulemuseks pilt, mis on mõlemast oluliselt suurem. Pilt jääb suuremaga võrreldes enamvähem samaks ainult sel juhul kui teine pilt on väike, aga ka sel juhul kandub osa tulemist endise suurusega pildi servade taha. Veelgi enam, kui siin näites saadud tulemus *m*=400 punkti juures ära lõigata, on tulemuseks ebasümmeetriline kujund, mida esialgsete piltide kujudest ei oskaks kuidagi oodata. Seega peab olema valmis otsustama, kuidas tulemust käsitleda.





Nagu eelmises punktis mainitud, sidumi koordinaatruumis võib asendada korrutisega Fourier' ruumis. Paraku, sellisel juhul muudab äsjakirjeldatud efekt oma kuju. Selle põhjuseks on asjaolu et Fourier' ruumis käsitletakse kõiki funktsioone automaatselt perioodilistena, nende perioodiks saab etteantud funktsiooni see määramispiirkond, mis on etteandmisel kirjeldatud. Olukorda käsitletakse sellisena et kohe pärast kirjeldatud funktsiooni määramispiirkonna lõppu algab funktsioon uuesti ja sedaviisi kordub ikka ja jälle. Äsjakirjeldatud 400-punktise ulatusega funktsioonide puhul tähendaks eelöeldu et pärast 400 punkti algavad mõlemad funktsioonid automaatselt uuesti. Järelikult ei arvutataks Fourier' ruumis korrutist mitte kahe 400-punktise ulatusega funktsiooni Fourier' pööretest, vaid hoopis teistsugustest, perioodiliselt korduvatest funktsioonidest. Seda, mida arvutatakse, saab sellesama ekvivalentsseaduse alusel modelleerida koordinaatruumis, kuid kasutades funktsioone sellistena, nagu Fourier' ruum neid käsitleb. Meil tuleb võtta sidum kahest perioodilisest funktsioonist, mõlemad perioodiga 400. Nagu näha paremas veerus oleva joonise lõpufragmendist, on tulemus eelmisel korral saadust hoopis erinev.

Et arvutada funktsioonide sidumi ekvivalenti Fourier' ruumi kaudu, tuleb funktsioonid ette anda selleks sobitatud kujul (ingl. padded functions). Kui tegemist on kahe funktsiooniga, esimese määramispiirkond A ja teise määramispiirkond B (eespool, A=400 ja B=400), siis tuleb nendest luua uued funktsioonid sel viisil et täiendada mõlemat nullidega kuni määramispiirkonna uue pikkuseni vähemalt A+B-1 (ingl. function padding). Eespool vaadeldud juhul tuleks luua kaks uut funktsiooni, mõlemad määramispiirkonna pikkusega vähemalt 799;

need uued funktsioonid sobivad ka Fourier' ruumi teisendusteks. Joonisel on näha, milline saaks olema uue variandi tulem. On selge et tulemus saab olema küll perioodiline (mis on liiast), aga iga periood eraldi võetuna annab õige kujundi.

Ka järgnevates MathCad abil tehtud näidetes on ilmselt näha et esialgsete funktsioonide määramispiirkondade laiendamine annab tõepoolest teistsuguse tulemuse kui on tulemus, mis saadi ilma määramispiirkondi laiendamata. Aluseks on võetud kaks 3-mõõtmelist funktsiooni, piltide kujul näeksid need funktsioonid välja mustal taustal olevate heledate ruutudena seal, kus

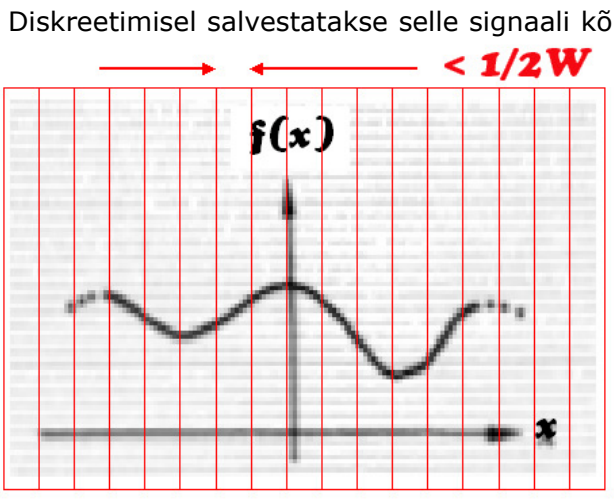
funktsioonide väärtused on nullist suuremad. Arvutatakse nende funktsioonide (piltide) sidum, aga kasutades selleks eelnevalt Fourier' ruumi teisendatud funktsioonide korrutist.

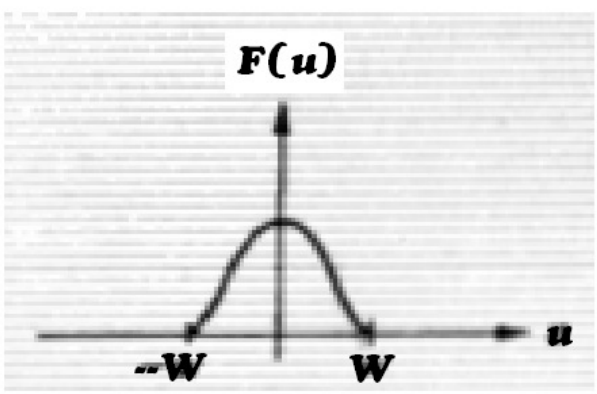
Esimesel juhul võetakse pildid nii suurtena nagu nad on, teisel juhul on mõlemat pilti suurendatud, lisandunud ala on täidetud nullidega (pildil on see ala must). Esimesel juhul on tulemuseks ootamatult ebasümmeetriline kujund, mis erineb oluliselt teisel juhul saadud tulemusest. Teisendusi teostav Mathcad programm on saadaval siit.

5. Diskreetimisvõre vähim lubatav sagedus.

Nagu näidatud esimeses teemas, asetatakse reaalse maailma signaali (analoogsignaali) diskreetmisel temale teatav võre. Näidati ka et kui asetatav võre on ebasobiv, võib juhtuda asju. Pooleli jäi küsimus, milline peaks sobiv võre olema. On ilmne et esialgse signaali täpsemaks salvestamiseks on kasulikum tihedam võre, ehk signaalist tuleks teha rohkem väljavõtteid. Järgnevalt vaatame, kui palju tuleb väljavõtteid teha ehk kui tihedast võrest piisab.

Olgu mingi, nt kõrval kujutatud signaal ***f(x)***. Diskreetimisel salvestatakse selle signaali kõik need väärtused, mis ühtivad sellele asetatud võrega. Protsessi võib ette kujutada sel viisil et me korrutame elementkaupa läbi kaks signaali: esimene on esialgne signaal ise ja teine on "signaal", mis moodustub võrest. Teine signaal on null kõikjal, välja arvatud võre sõlmede asukohtades, kus ta on 1. Järelikult, meil on tegemist kahe signaali elementkaupa korrutistega. Eelnenu põhjal võib öelda et see on ekvivalentne nende kahe signaali Fourier' pöörete sidumiga. Et seda sidumit hinnata, on tarvis mõlema funtsiooni Fourier' kujusid. Signaali "1" Fourier' pööre on deltafunktsioon, kuna aga "1"-d on terve rida, saavad nad Fourier' ruumis asuma naabrist teataval kaugusel du, mis omakorda sõltub signaalide "1" omavahelisest kaugusest koordinaatruumis *dx*, ehk *du* on *dx* funktsioon. Järelikult, muutes *dx* (mis juhtumisi ongi otsitav võre diskreet), saame muuta *du*. Aga me ei tea veel, kuidas on teda vaja muuta.





Vaatame nüüd esialgse, diskreeditava, signaali Fourier' pööret ***F(u)***. Üldiselt on esialgse signaali edastamiseks sageduste ruumis tarvis (lõpmatult) palju elementaarsagedusi. Aga oletagem et see pole nii. Oletagem et esialgse signaali saab teisendada Fourier' ruumi nii et nullist erinevateks jäävad ainult väike osa võimalike elementaarsageduste kordajatest. Selline signaal võiks Fourier' ruumis näha välja nagu kõrvaltoodud pildil, kus olulisim on asjaolu et kordajad on nullist erinevad ainult vahemikus (-**W** ... **W**). Hakkame nüüd ette kujutama selle uue kuju sidumit Fourier' ruumis juba ootavate, teineteisest kaugusel *du* asuvate, deltafunktsioonidega. Sidum deltafunktsiooniga on funktsioon ise, nullpunktiga deltafunktsiooni asukohas. Kuna meil on ees ootamas rida deltafunktsioone, saab iga üksik sidum olema laiusega 2**W** ja asuma kõrvalolevast sidumist kaugusel *du*. Tekib ilmne piirang: kõrvalolevad sidumid ei tohi kattuda, vastasel korral kirjutatakse osa väärtusi üle. Järelikult peab olema *du* > 2**W**. Kuna *du* oli võre otsitava diskreedi (sageduse) sammu *dx* funktsioon (täpsemalt: nad on pöördvõrdelised), saame leida tingimuse *dx* jaoks: *dx* < 1/(2**W**).

Mida suuremate omasagedustega nähtusi esineb esialgses signaalis, seda väiksem peab olema diskreetimisvõre samm ehk seda suurem peab olema diskreetimisvõre sagedus. Et tuua tulemust rohkem igapäevaellu, siis **küsimus**: kui suur peaks olema nn sämplimissagedus et saaks inimkõrvaga kuuldava heli CD-plaadi tarvis kadudeta digitaliseerida? Kui sämplimissagedus välja valitud, siis seda enam ei muudeta (CD pole kummist). Mis juhtub nüüd kui sellelesamale CD-le salvestatakse ka taustal häälitsevad nahkhiired? Kui juhtub midagi ebasoovitavat, kuidas seda ära hoida (meenutan: sämplimissagedus on paigas, seda enam muuta ei saa)?

Aare.Luts.1@eesti.ee

6. DCT teisendus.

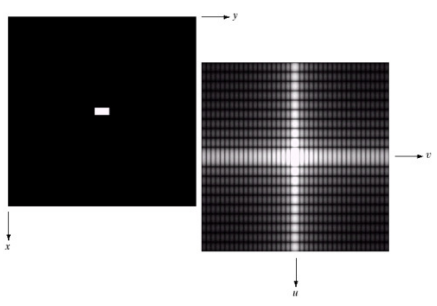
Nii DFT kui DCT teisendavad pildi ruumi, mille baasfunktsioone saab iseloomustada sageduste kaudu. DFT puhul on tegemist imaginaarosa sisaldavate baasfunktsioonidega, DCT puhul eranditult reaalsete baasfunktsioonidega. DCT teisenduse ja pöördteisenduse täpsed kujud ruutpiltide jaoks on toodud pildil. Muutujad *x*, *y*, *u* ja *v* on tuntud, *N*-1 on

$$C(u, v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y)\cos\left[\frac{(2x + 1)u\pi}{2N}\right]\cos\left[\frac{(2y + 1)v\pi}{2N}\right]$$

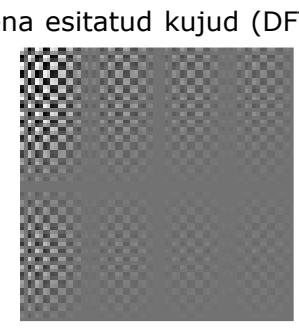
$$u, v = 0, 1, 2, \ldots, N - 1, \text{ and}$$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v)C(u, v)\cos\left[\frac{(2x + 1)u\pi}{2N}\right]\cos\left[\frac{(2y + 1)v\pi}{2N}\right]$$

pikslite arv ühes sihis, *a(u)* ja *a(v)* on teatavad konstandid (saab leida kirjandusest). DCT eeliseks on, näiteks, et tema interpreteerimine on imaginaarosa puudumise tõttu mõneti lihtsam. Näitena on toodud sama pildi DFT ja DCT teisenduse piltidena esitatud kujud (DFT



juhul on esitatud "power spectrum", DCT juhul vahetu pilt). Nagu näha, on sarnasusi aga on ka erinevusi. Siiski omab ka DCT, oma "poolikusest" hoolimata, ikkagi mitmeid sagedusruumist tulenevaid praktiliselt kasulikke omadusi, näiteks saab DCT kasutada piltide sageduskarakteristikute hindamisel



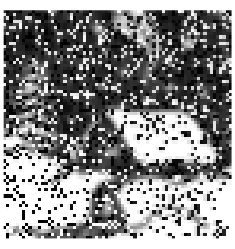
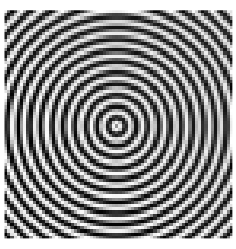
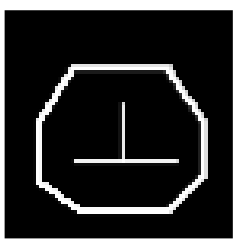
või sagedusliku iseloomuga moonutuste kõrvaldamisel. DCT laialt levinud rakenduseks on .jpg-formaat. DCT puuduseks DFT kõrval on, näiteks, asjaolu et DCT ei paku mitmeid matemaatiliselt kasulikke omadusi, nt ei kehti DCT puhul sidumi ja korrutise ekvivalentsi seadus. DCT ruutpiltide jaoks on realiseeritud ka kursuse praktiliste tööde Java-paketis.

Aare.Luts.1@eesti.ee

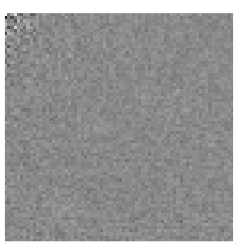
7. Sagedusruumi omaduste kasutusnäiteid.

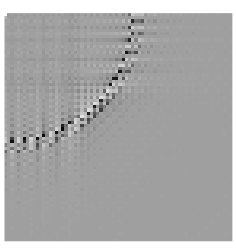
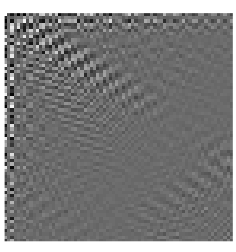
Sagedusruumis paistavad piltidel olevad nähtused teistmoodi, seega pakub sagedusruum uue võimaluse piltidel leiduva info otsimiseks ja töötlemiseks. Seda võimalust ei paku mitte ainult Fourier' ruum, võimalusi pakub ka näiteks DCT-ruum. Allpool kasutamegi näideteks põhiliselt seda ruumi, osalt ka põhjusel et DCT-teisendus on kursuse Java-paketis realiseeritud, DFT aga mitte.

Ülesandeks võib olla nt piltide klassifitseerimine, seda mingi (matemaatilise) kriteeriumi, näiteks iseloomuliku mustri alusel. Selleks tuleks pildid grupeerida nende "sarnasuse" alusel. "Sarnasust" võib hinnata mitmest aspektist, üheks (lisa)võimaluseks on hinnata "sarnasust" sagedusruumis. Näiteks: püüame hinnata järgnevate piltide nö. sarnasuse määra. Pildid, mis koordinaatruumis on nö. sarnased, võivad sagedusruumis olla väga

erinevad, ja vastupidi. Esmapilgul võivad mõisted nagu "piltide sarnasus" näida üsna laialivalguvad, ometigi leiavad ka seda liiki mõisted pilditöötluses rakendust. Näiteks on eelkõige

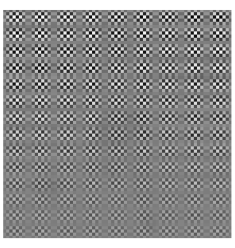


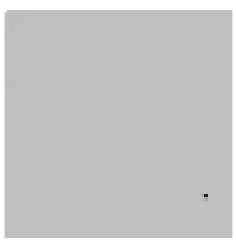



sagedusruumis võimalik defineerida piltide klass "tavaline pilt". Selle klassi alla võib tinglikult liigitada enamuse igapäevaelust tehtud piltidest. Kui rääkida "klassi sarnasuse kriteeriumist", siis selle klassi piltide tüüpkuju pildiruumis on selline, ja klassi tüüpkuju DCT-sagedusruumis on selline. On üsna selge et "tavalise pildi" sagedusruumi tüüpkuju erineb näiteks sellest kujust. Klass "tavaline pilt" ei ole lihtsalt mõiste iseeneses, sellest mõistest tuleneb mõnigi täiesti tarvilik praktiline rakendus, näiteks võib öelda et ainult sellesse klassi kuuluvaid pilte on mõistlik esitada .jpg-kujul. Jpg-formaat kasutab piltide pakkimiseks asjaolu et pildis on suhteliselt vähe kõrge omasagedusega nähtusi (teravad servad, eraldatud jooned, isoleeritud punktid). Kui nii on, on jpg teisendusalgoritm üsna efektiivne, kui aga pilt ei kuulu sobivasse klassi, juhtub üks kahest: kas ei saavutata pildi suuruse vähenemist või siis moonduvad pildil olevad detailid.

ISESEISVALT: Proovida teisendada pilt 1, pilt 2, pilt 3, pilt 4, pilt 5, pilt 6 ja pilt 7 .jpg-kujule. Võrrelda tulemusi nii saadavate failide suuruse kui ka pildi oluliste detailide säilivuse seisukohast, mõelda kuidas on tulemused seotud piltide kujudega sagedusruumis. VIHJE: Ülalnimetatud piltide kujud DCT-ruumis saab arvutada kursuse praktiliste tööde Java-paketi abil.

Piltidel kuju erisusi erinevates ruumides saab kasutada ka, näiteks, teatavat liiki moonutuste kõrvaldamiseks. Olgu meil moonutatud pilt (vasakpoolne), mille kohta on teada ainult seda, et ta peaks olema "tavaline pilt". Koordinaatruumis ei oskagi, kust moonutuse



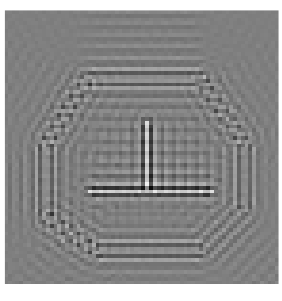
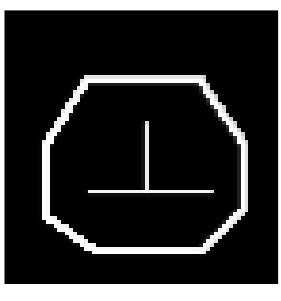
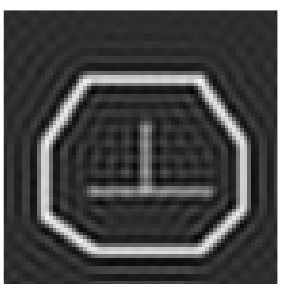


kõrvaldamisega alustada. Sagedusruumis on lihtsam. Pildi taastamiseks kasutame kursuse Java-paketti. Selles saab arvutada moonutatud pildi kuju DCT-ruumis (piltidest keskmine). Selle pildi kuju sagedusruumis saab võrrelda "tavalise pildi" kujuga sagedusruumis (pildidest parempoolne). Keskmisel pildil on täheldatavad kaks märgatavat iseärasust. Täpsemalt on iseärasused näha sagedusruumi teisendusel, mis on esitatud arvutabeli kujul. Arvutabeli saab tekitada, salvestades Java-paketis tekitatud DCT-teisenduse tulemuse .prn-failina. Arvutabelit on mugav sisse lugeda, sirvida ja muuta näiteks MathCad abil. Arvutabelit võib muuta ka tavalise tekstiredaktoriga.

Pildi ligikaudseks taastamiseks tuleb iseärasused kõrvaldada. Teisisõnu, kui midagi paremat ei oska teha, tuleb ilmsete iseärasuste kohtadesse kirjutada naabrusele iseloomulikud arvud. Antud juhul kõrvaldatakse iseärasused väga jämedalt, sest täpsem info moonutuse kohta puudub, lähtuda saab ainult klassi "tavaline pilt" tüüpkujust; mis konkreetselt tehtud, vt MathCad failist. Seejärel saab MathCad abil muudetud tabeli jälle .prn-faili salvestada ja lugeda selle uuesti Java-paketti, kus saab teha selle parandatud tabeli alusel DCT-

pöördteisenduse ehk tuua parandatud tulemuse tagasi koordinaatruumi. Tulemus annab juba midagi aimata, et paremini näha, tuleks teisendada histogrammi või mängida võimalusega "reaalarvud pildimaatriksiks logaritmiliselt".

Näeme, et juba väga umbkaudse teisendusega, lähtudes ainult sellest, kuivõrd on kahjustatud pildi DCT-kuju erinev klassi "tavaline pilt" tüüpkujust, saab ometigi teatud moonutusi suhteliselt edukalt kõrvaldada. Võib öelda ka nii et sagedusruumi saab kasutada (eba) sobiva sageduskarakteristikuga nähtuste eraldamiseks pildilt. Teada tuleb vaid otsitavat (ebasoovitavat) nähtust iseloomustavat sageduskarakteristikut. DFT ruumi sagedussamm ja pildiruumi samm on seotud pöördvõrdeliselt $du \sim 1/dx$. Seega, aeglased muutused pildiruumis (suur samm) vastavad madalatele sagedustele sagedusruumis (väike samm); kiired muutused pildiruumis (väike samm) vastavad suurtele sagedustele (suur samm).

Kvalitatiivne näide: proovime vasakul näha oleva pildi sagedusfiltreerimist. Keskmisel pildil on näha tulemus, kui eemaldati kõik kõrgemad sagedused; Parempoolsel pildil on näha tulemus, kui eemaldati kõik madalamad sagedused. Peale vihjete, millist liiki infot esitavad madalamad ja millist kannavad kõrgemad sagedused, saame hulga laineid ehk arusaamatuid kõrvaltulemusi. Suurt osa neist tulemustest käsitleme selle teema teises pooles.

Aga kõike seda ja ka enamat saab ise läbi teha, näiteks kursuse praktiliste tööde Java-paketi abil, sealsamas saab teha ka sageduslõike, misjärel võib teha pöördteisenduse nägemaks, mis sageduslõikega muutus. DCT kuju saab salvestada ka arvutabeli kujul (.prn-failina), salvestust (arvutabelit) saab vaadata ja töödelda nt MathCad abil, kui selles sisseloetava faili nimi sobivaks muuta. MathCad abil töödeldud tabeli saab uuesti (Java-paketti) sisse lugeda ja seal edasi toimetada.

Aare.Luts.1@eesti.ee

Aare.Luts.1@eesti.ee

Teema 4 küsimused ja ülesanded on järgmised:

1. Vaatame diskreetse Fourier' teisenduse

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

arvutusvalemit. Mis on liikme ***F(0,0)*** sisu? Mille poolest erinevad liikmed ***F(1,0)*** ja ***F(0,1)***? Iseloomusta pilti, mille DFT kujus on väike ***F(0,0)*** ja suur ***F(M-1,N-1)***.

2. Eespool kujutatud valemis oli tegemist kahe muutuja funktsiooniga *f(x,y)*. Vaatame nüüd selle ühemõõtmelist analoogi *f(x)*. Mida näitab *x0*, mida *x* ja mida *Δx*?

$$f(x) = f(x_0 + x\,\Delta x)$$

3. Koordinaat- ja Fourier' ruumi üheks olulisemaks seoseks on sidumi ja korrutise ekvivalentsus. Olgu tegemist mitte kahe- vaid ühemõõtmelise juhuga ning

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

olgu funktsioon *f(x)* kirjeldatav reaga (1, 4, 0, 0, 8, -2, 2), funktsioon *h(x)* reaga (-1, 1). Arvutage nende funktsioonide sidum. Funktsioon *F(u)* rida olgu (5, 8, -1, 4) ja *H(u)* rida (-2, -3, 2, 3) (need pole seotud eelnenud *f(x)* ja *h(x)* -ga). Arvutage *F(u)* ja *H(u)* korrutis viisil, mida nõutakse ekvivalentsuse seaduses.

4. Kui suur peaks olema nn sämplimissagedus et saaks inimkõrvaga kuuldava heli CD-plaadi tarvis kadudeta digitaliseerida? Kui sämplimissagedus välja valitud, siis seda enam ei muudeta (CD pole kummist, ja kõik CD-d peavad olema ühesugused). Mis juhtub kui sellelesamale CD-le salvestatakse ka taustal häälitsevad nahkhiired? Kui juhtub midagi ebasoovitavat, kuidas seda ära hoida (meenutan: sämplimissagedus on paigas, seda enam muuta ei saa)?

5. Teisendada pilt 1, pilt 2, pilt 3, pilt 4, pilt 5, pilt 6 ja pilt 7 .jpg-kujule. Võrrelda tulemusi nii saadavate failide suuruse kui ka pildi oluliste detailide säilivuse seisukohast, mõelda kuidas on tulemused seotud piltide kujudega sagedusruumis. Vastuse osana loetleda pildid, mis sobivad esitamiseks .jpg-kujul ja pildid, mis ei sobi. Millised on pildiinfo sagedusjaotus esimesel, milline teisel juhul?

6. Taastada see pilt. Võib öelda et see on "tavaline pilt" ja ka muus kehtib konspekti punktis 1.7 öeldu.

7. Mis liiki moonutusega on tegemist sellel pildil? Vihje: kuna seda saab suhteliselt lihtsalt kõrvaldada, peaks moonutus selle lihtsusega seotud olema. Võibolla õnnestub isegi mõni sedalaadi moonutus ise tekitada?

Vastused arutage rühmas (rühmafoorumis) läbi, ühised vastused koos lahendustega vormistage leheküljele "4.teema vastuste ja lahenduste Wiki". Kui vastused ja lahendused valmis, postitagu rühma üks liige sellekohane teade foorumisse "4.teema vastused valmis". Seejärel avatakse seni ainult rühmale näha olev Wiki kogu kursusele ja vastused ning lahendused lähevad retsenseerimisele.

5. teema õppematerjalid.

Selles raamatus tutvustatakse libiseva maski tehnoloogiat, rõhuga selle seostel sagedusruumiga. 1. Libiseva maski olemus. 2. Pildi koordinaatruumi libiseva maski seostamine sagedusruumiga. 3. Libiseva maski tehte olemaolevad realisatsioonid. 4. LSI-tehted. 5. Maskide kombineerimine. 6. Libiseva maski tehte pööratavus.

Õpikeskkond:	TÜ Moodle
Kursus:	Pildiinfo töötlus (LOFY.05.055)
Koosta raamat:	5. teema õppematerjalid.
Printed by:	Aare Luts
Kuupäev:	teisipäev, 22 mai 2012, 08:27

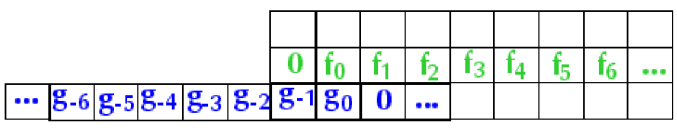
Sisukord

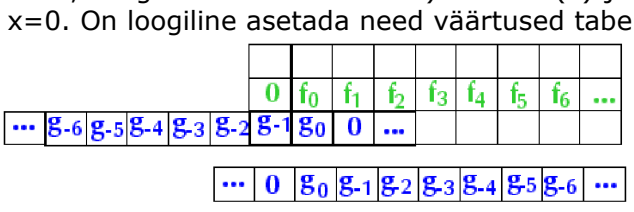
1. Libiseva maski olemus.

Eespool sai tegeldud sidumiga, mis ühemõõtmelisel juhul on defineeritud nagu näha kõrvalolevas valemis. Vaatleme nüüd selle arvutamist ühel konkreetsel juhul, mis on samas oma konkreetsusest hoolimata piisavalt üldine. Kirjutame välja diskreetsed funktsioonid f(x) ja g(x). Kuna nad on diskreetsed, saab nad esitada tabeli(te) kujul. Eeldame et f(x) on nullist erinev ainult argumendi x mittenegatiivsete väärtuste korral, samuti eeldame et g (x) on nullist erinev ainult argumendi x mittepositiivsete väärtuste korral. See ei vähenda käsitluse üldisust, küll aga muudab lihtsamaks tehte käigus toimuva (visuaalse) hoomamise. Kui vaadelda argumentskaalat (diskreetne juhtum, seega x väärtuste tabelit) siis on f(x) ja g(x) on mõlemad nullist erinevad ainult kohas x=0. On loogiline asetada need väärtused tabeli(te)s kohakuti. Funktsiooni f(x) väärtuste tabel saab olema null x negatiivsete väärtuste korral ja funktsiooni g(x) tabel on null argumendi x postitiivsete väärtuste korral, seetõttu on nii x negatiivsete kui ka positiivsete väärtuste korral nullist erinev ainult üks funktsioonidest f(x) või g(x).

$$f(x)*g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)\,d\alpha$$

$$FG = f(x) * g(x) = \int_{-\infty}^{+\infty} f(u)g(x-u)du$$





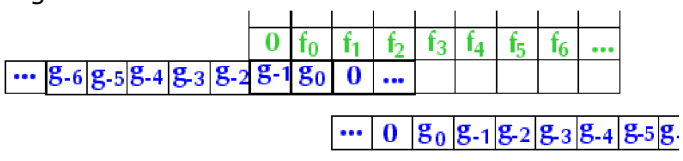
$$FG(0) = f_0 g_0 + f_1 g_{-1} + f_2 g_{-2} + f_3 g_{-3} + \ldots\ldots$$

Sidumi arvutamise valemi järgi tuleb üks funktsioonidest argumendi x nullpunkti suhtes peegeldada, valemis on peegeldatud g(x). Sel juhul saab asetada kohakuti f(x) ja g(-x) väärtused, sidumi arvutamise valemi järgi arvutatakse kohakuti jäävate väärtuste korrutiste summa üle kogu määramispiirkonna. Näiteks, kohal x=0 saab summaks f(0)*g(0) = f(0)*g(0) +f(1)*g(-1) +.... Kohal x=1 oleva summa f(1)*g(1) leidmiseks vaatame üksipulgi arvutusvalemit. Valemisse lähevad kõik f(u) väärtused, kuid juhul u<0 on need väärtused, vastavalt funktsioonide valikule, nullid. Seetõttu on mõtet integreerida (diskreetsel juhul: võtta summa) ainult üle määramispiirkonna u>=0. Funktsioonist g(u) lähevad sel juhul valemisse liikmed, mis sisaldavad ka x valitud väärtust ehk g(x-u), x=1 korral g(1-u). Funktsioonist f(u) lähtudes oli mõtet võtta summa üle liikmete u>=0. Aga, funktsioonide valiku tõttu, on g(u) nullist erinev ainult juhul u<=0. Seega, liige f(0)*g(1-0) on null selle tõttu et g(1) on null. Sellel juhul algavad valemi järgi saadavas reas nullist erinevad liikmed alates kohast f(1)*g(0). Analoogiliselt algavad juhul x=2 nullist erinevad liikmed alates kohast f(2)*g(0).

$$FG(0) = f_0 g_0 + f_1 g_{-1} + f_2 g_{-2} + f_3 g_{-3} + \ldots\ldots$$
$$FG(1) = 0 + f_1 g_0 + f_2 g_{-1} + f_3 g_{-2} + \ldots\ldots$$

Näeme et f(x)*g(x) argumendi x igale konkreetsele väärtusele vastava integraali (summa) arvutamiseks sobib olukord kus f(x) ja g(-x) väärtuste read on teineteisega teatavas asendis kohakuti ja selles asendis arvutatakse f(x) ja g(x) ridade kõigi kohakuti olevate liikmete korrutiste summa. Et leida argumendi x teistsugusele väärtusele vastavat summat, kehtib sama mehhanism, kuid g(-x) rida tuleb eelnevalt nihutada vajalikku kohta. Kui vaatleme väärtusi x ja x+1, tuleb g(-x) rida nihutada ühe võrra. Näite fail isemuutuval kujul on saadaval siit. Järelikult, sidumi arvutamise tehe on oma mehhanismilt sarnane teatava libiseva maskiga, kus teatav mask (siin: funktsioon g(-x)) libiseb mööda teatavat rida (siin: funktsiooni f(x) väärtusi) ja maski igas asendis arvutatakse maski ruutude ja maski konkreetsete ruutude alla jäävate väärtuste korrutiste summa. Sidum on oma tehniliselt olemuselt libiseva maski tehe.

$$FG(2) = 0 + 0 + f_2 g_0 + f_3 g_{-1} + \ldots\ldots$$

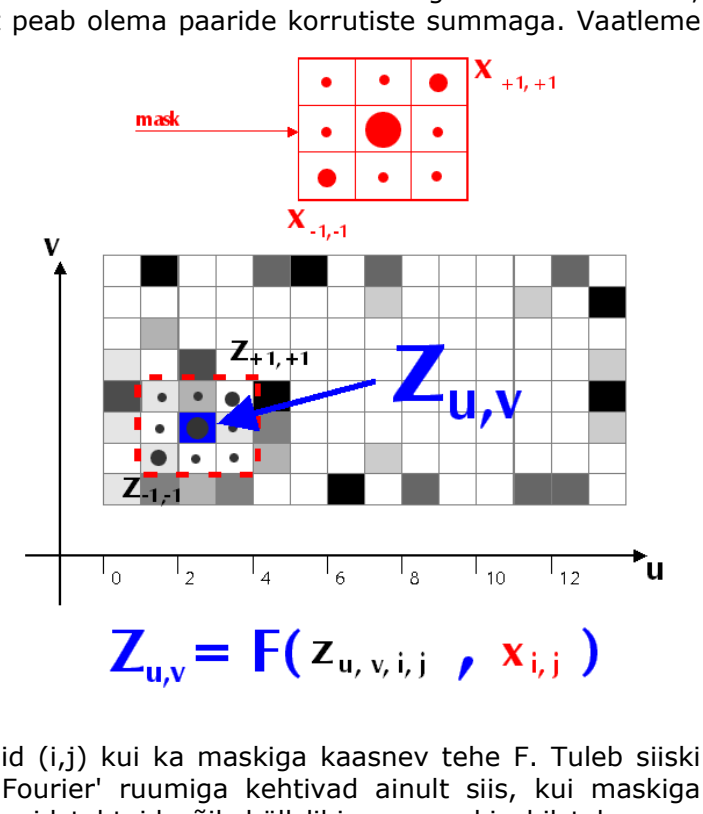


$$FG(0) = f_0 g_0 + f_1 g_{-1} + f_2 g_{-2} + f_3 g_{-3} + \ldots\ldots$$
$$FG(1) = 0 + f_1 g_0 + f_2 g_{-1} + f_3 g_{-2} + \ldots\ldots$$
$$FG(2) = 0 + 0 + f_2 g_0 + f_3 g_{-1} + \ldots\ldots$$
$$FG(3) = 0 + 0 + 0 + f_3 g_0 + \ldots\ldots$$
$$\ldots\ldots\ldots\ldots$$

Sidumi tulemit võib avaldada ka maatrikskujul (mõistagi on seda mõtet teha ainult juhul kui funktsioonid on sobivate omadustega ehk kui kasutatavad maatriksid pole mitte väga suurte mõõtmetega). Siin on tulem avaldatud mitte sidumina f(x) *g(x) vaid (punkt)korrutisena FG, rõhutamaks tehete ekvivalentsust koordinaat- ja Fourier'-ruumis. Nagu näha, on FG vektor, mille elementide väärtused saab arvutada vastavalt maatriksalgebra reeglitele. Tehte esimeseks argumendiks on maatriks, mille igas reas on funktsiooni g(x) rida, kusjuures iga eelmise reaga võrreldes on g(x) rida ühe võrra nihutatud. Tehte teiseks argumendiks on sidumisse kuuluva funktsiooni f(x) väärtuste rida vektorkujul.

$$FG = \begin{pmatrix} g_0 & g_{-1} & \ldots \\ 0 & g_0 & g_{-1} \ldots \\ 0 & 0 & g_0 \; g_{-1} \\ & \ldots\ldots\ldots\ldots \\ & \ldots\ldots\ldots\ldots \end{pmatrix} * \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \ldots \\ \ldots \end{pmatrix}$$

Põhimõtteliselt võib tehet "libisev mask" ka üldistada. Tehte tehnoloogiast lähtudes võib, näiteks, ära unustada nõude et tegemist peab olema paaride korrutiste summaga. Vaatleme joonisel punase katkendjoonega piiratud 3x3 ala, mis võib mingi piirkonna (nt pildi) kohal liikuda, nimetame selle ala maskiks. Siin on tegemist 3x3 maskiga, aga mask võib olla ka teistsuguste mõõtudega. Maski alla jääva "maastiku" koordinaatideks olgu u ja v, maski enda koordinaatideks olgu i ja j. Maski igas asendis rakendatakse tehet F, mis kas asendab maski alla jääva piirkonna maski keskmise ruuduga kohakuti oleva väärtuse uue väärtusega Z(u,v) või siis kirjutab uue väärtuse Z(u,v) mingisse täiendavasse massiivi. Väärtuse Z(u,v) arvutamisel kasutatakse mingit, põhimõtteliselt suvalist, funktsiooni, mille argumentideks on maski kordajad (maski ruutudes asuvad väärtused x (i,j)) ja "maastikul" maski asendis (u,v) maski konkreetsete ruutude alla jäävate maastiku ruutude väärtused z(u,v,i,j). Maski igas asendis (u,v) kehtestuvad nii maskiga määratud lokaalsed koordinaadid (i,j) kui ka maskiga kaasnev tehe F. Tuleb siiski arvestada et eespool näidatud seosed Fourier' ruumiga kehtivad ainult siis, kui maskiga kaasnevaks tehteks on sidum, teistsuguseid tehteid võib küll libiseva maski abil teha, aga seosed Fourier' ruumiga ei pruugi siis enam kehtida.



$$Z_{u,v} = F(z_{u,v,i,j}, x_{i,j})$$

Libiseva maski tehte sedaviisi üldistamise põhjuseks on eelkõige tehnoloogiline sarnasus: kui tehe on kord juba realiseeritud, saab sama realisatsiooni kasutada mitut liiki ülesannete lahendamiseks, valides ainult parajasti sobiva maski ja sobiva tehte.

Küsimus: 1) Kas libiseva maski tehte tulemuse võiks kirjutada tagasi otse maski all olevale "maastikule" või pigemini teatavasse puhvrisse? Kui puhvrisse, siis miks? Meenutan: maski tehte sagedaseks eesmärgiks on muuta "maastikku" ennast ehk kasutada maski tehet pildi teatavaks muutmiseks. Seetõttu tuleb tulem niikuinii varem või hiljem "maastikule" tagasi kirjutada.

Aare.Luts.1@eesti.ee

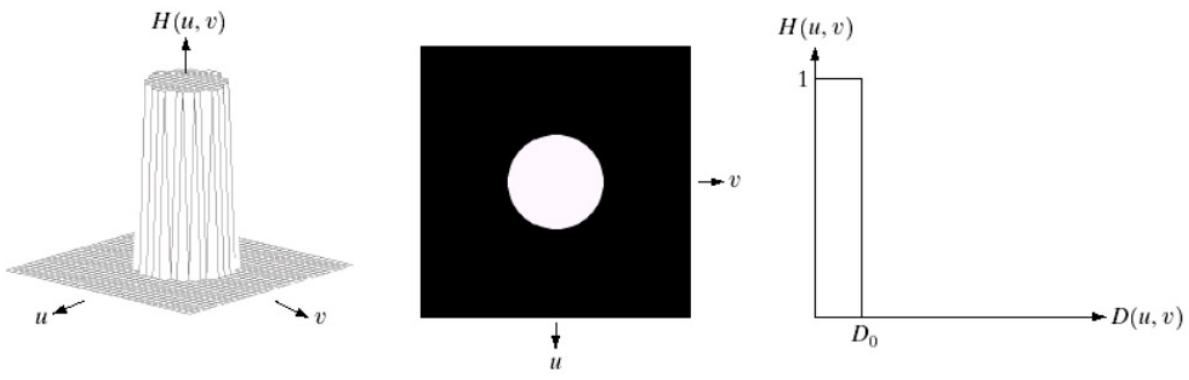
2. Pildi koordinaatruumi libiseva maski seostamine sagedusruumiga.

Nagu mainitud eestpool, on libiseva maski rakendamine samaväärne sidumi arvutamisega, seda juhul kui libiseva maskiga seotud tehteks on arvupaaride korrutiste summa arvutamine. Nagu näha ka eespool olnud seosest, saab sel juhul pildiruumi libiseva maskiga teha

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

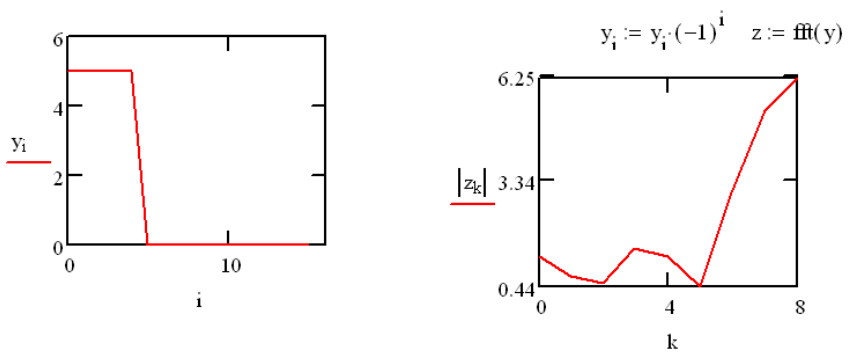
samaväärset tehet mida saaks teha sagedusruumis. Ühtlasi tähendab eelöeldu, et maski rakendamine pildiruumis tähendab alati teatava tehte tegemist sagedusruumis, sõltumata sellest kas me sagedusruumi tehet tahame või mitte. Samuti, tehte (konkreetselt, korrutise) tegemine sagedusruumis tähendab ühtlasi et me justkui rakendame teatavat maski pildiruumis. Et see nii on, on oluline osata hinnata, mida üks või teine teisendus teises ruumis teeb, otstarbekas on teha teisendus selles ruumis kus teisenduse tegemine on lihtsam.

Kui lähtuda soovist teha teatav sagedusruumi teisendus, aga seda teisendust ei taheta teha sagedusruumis (nt põhjusel et taheta teha mitut sel juhul vajalikku DFT-teisendust), tuleb igal konkreetsel juhul sobiv (koordinaatruumi) mask koostada selle põhjal, millist (sagedusruumi) teisendust soovitakse teha. Sobivat, analoogilist tehet teostavat, maski saab leida näiteks MathCad abil (üks sobiva maski arvutamist ja lähendatud maski mõju hindamist teostav tööleht). Siintoodud MathCad näidete juures tuleb arvestada asjaolu et need MathCad näited kujutavad vaid 1D funktsioonide (maskide) püstlõikeid nullpunkti läbiva püsttelje suhtes sümmeetrilistest teisendustest. Sümmeetrilise teisenduse üks 3-mõõtmeline näide on toodud allloleval joonisel. Sedalaadi juhtumid puudutavad väga sageli esinevaid

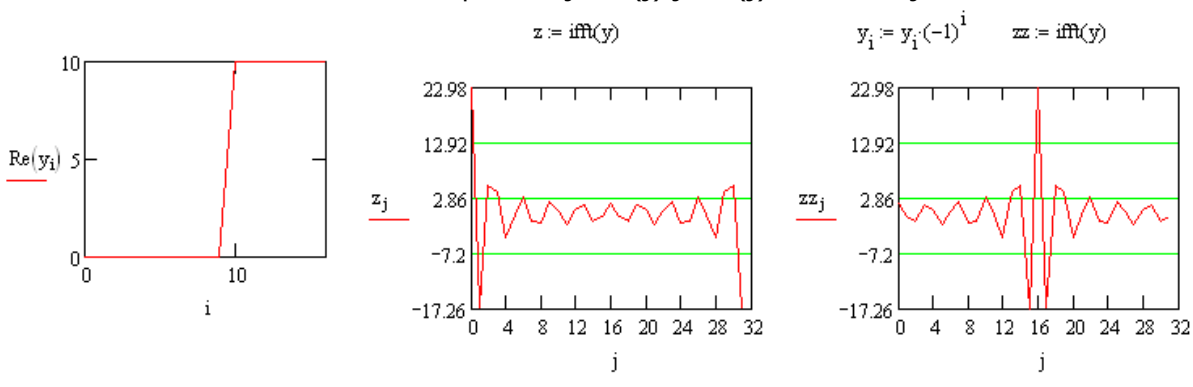


teisendusi, kus kasutatavad maskid on nullpunkti läbiva püsttelje suhtes täielikult sümmeetrilised. Loomulikult on võimalikud ka teisendused, kus kasutatavad maskid ei ole (mingi telje suhtes) sümmeetrilised (enamgi, teatavat liiki tehete jaoks on ebasümmeetrilised maskid lausa möödapääsmatud). Siiski, praegu käsitletav juhtum on praktilises mõttes igati oluline ja, samas, teda kirjeldavad joonised on oluliselt lihtsamad kui ebasümmeetrilistel juhtudel. Juba jooniste lihtsus on oluliseks argumendiks seni, kuni tegemist on mitte erijuhtumite, vaid põhimõtete tutvustamisega. Siin tutvustamegi eelkõige põhimõtteid.

Jooniste õigeks interpreteerimiseks tuleb tähele panna mõningaid (MathCad eripärast tingitud) asjaolusid. Esiteks, Mathcad arvutatav 1-dimensionaalne DFT-teisendus (tehe ***fft***) vajab alati argumendiks vektorit karakteristliku mõõduga 2 astmes m. Tulemuseks annab tehe alati vektori, mille analoogiline mõõt on 2 astmes (m-1) pluss 1. Näiteks: kui DFT-teisenduse argumentvektori y(i) pikkuseks on 16 (i=0...15), on tulemi vektori z(k) pikkuseks 8. Tulem on perioodiline (nagu mainitud eespool, käsitleb DFT juba olemuslikult olukorda nii et kõik funktsioonid on perioodilised). Võib ka öelda et Mathcad tehe ***fft*** annab tulemuseks argumendi DFT-teisendi ühe poolperioodi. Kui teostatakse pöördtehe ***ifft*** (argumendiks on funktsioon DFT-ruumis), tuleb argumentfunktsioon anda ette ühe poolperioodi kaudu. Näites olev funktsioon z(k) ongi esitatud ühe poolperioodi kaudu.



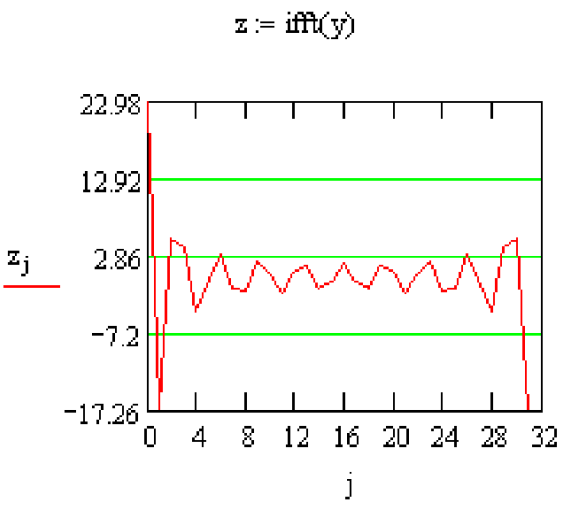
Teiseks, tuleb aru saada, milline osa funktsioonidest tähendab mida. Siin toodud näites on tegemist sagedusruumi (moduleeriva) funktsiooniga y(i), mille reaalosa kuju on näidatud joonisel ja mille imaginaarosa on null ja mille jaoks arvutatakse koordinaatruumis analoogilist modulatsiooni teostavate maskide täpsed kujud z(j) ja zz(j). Maskid z ja zz on oma olemuselt



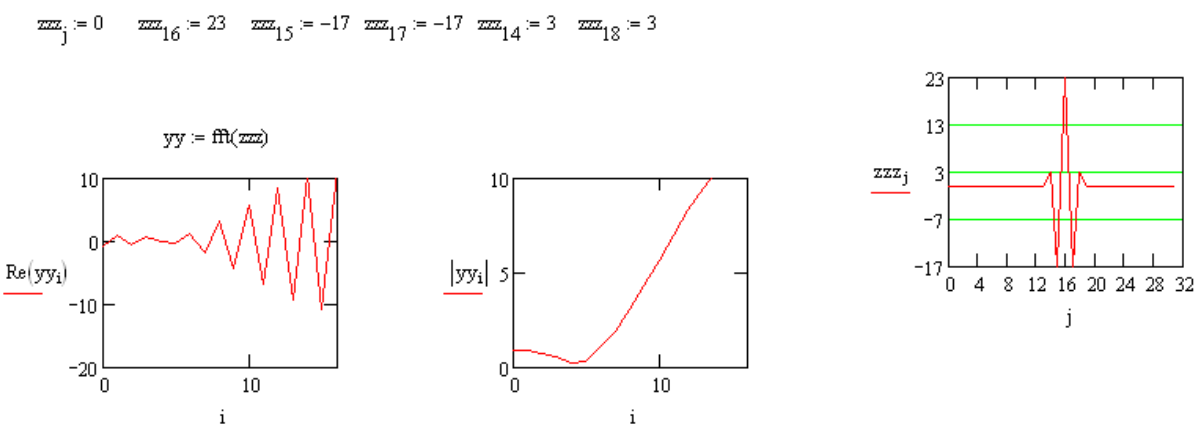
identsed, kuid esitatud kahel erineval viisil. DFT omadustest on teada et tehe yy(i)=y(i)*[-1 astmes i] annab uue vektori yy(i), mille DFT pöördteisendus ***ifft***(yy)on pöördteisenduse ***ifft*** (y) suhtes nihutatud poole perioodi võrra. Tulemid z(j) ja zz(j) ongi pöördteisenduse kaks tulemit, mis on nihutatud teineteise suhtes poole perioodi võrra. Funktsiooni y terminites (selle funktsiooniga siin lõigatakse maha kõik sagedused allpool 10) on õige esimene (mask z, kus suuremad väärtused on maski nullpunktile lähedasematel koordinaatidel), aga z kuju on hulga ebameeldivam lugeda. Nagu mainitud, siin on tegemist nullpunkti läbiva püsttelje suhtes sümmetriliste funktsioonidega. Seetõttu oleks mugav kui ka tehte tulemusena saadav mask näeks välja sümmeetriline. Nagu näha, näib mask zz(j) sümmeetriline. Et seda sümmeetriat saada, võib (teades, et olemuslikult on kõik DFT käsitletavad funktsioonid perioodilised) nihutada funktsiooni poole perioodi võrra, aga siis tuleb täpselt teada, mis on tehtud, nagu ka seda, kuidas saadud maski lugeda.

Kujutatud lõikest tuleb arvulisi väärtusi sisaldav mask kõigepealt koostada, arvestades ka nõuet loodava maski sümmeetria kohta. Siin toodud MathCad näited puudutavad 1D-funktsioone. Kui sel viisil saadavatest tulemustest koostada pilditöötluses tavalisi 2D-maske, tuleb täiendavalt arvestada maskide liitumisest tulenevaid asjaolusid, seda on kirjeldatud allpool. Ülaltoodud konkreetses näites on tegemist 32 elemendist koosneva (pildi)vektoriga z (j). DFT teisendus käsitleb seda vektori ühe perioodina. Järelikult, taustal vaadatakse asja sel viisil nagu korduks funktsioonide z(j) või zz(j) aknas (j=0...31) esitatu aknast väljas ikka ja jälle aknas esitatud kujul. Kuna funktsioonid on sümmeetrilised, peab nullpunkti läbiva püsttelje suhtes olema sümmeetriline ka loodav mask. Et koostada maski funktsiooni z(j) graafiku alusel, tuleb maski nullpunkti väärtuseks panna funktsiooni väärtus nullpunktis j=0, maski muude ruutude väärtused tuleb arvutada selle alusel, kui kaugel asub vaadeldav ruut koordinaatide nullpunktist.

Näiteks, kui me võtame ette et me ei soovi suuremat kui maski laiusega 5, tuleks maski nullpunkti väärtuseks (ümardatult) 23, maski punktide -1 ja 1 väärtusteks tuleks funktsiooni z väärtus kohal 1 ehk ümardatult -17. Nagu näha, punkti -1 väärtust joonisel justkui ei ole, aga siinkohas tulebki arvestada funkstiooni perioodilisust: kohal -1 on funktsioonil sama väärtus kui kohal 31. Punktidega -2 ja 2 on ka selles mõttes lihtne et kohal 2 on funktsiooni z väärtus täpselt olemas, see on ümardatult 3. Kui vaatleme näiteks ka lõikeid, mida on vaja kujuteldava 2-dimensionaalse 3x3 maski koostamiseks, siis punktide (-1,-1); (1,1); (1,-1); (-1,1) kaugused neid läbivate lõigete nullpunktist pole ei 1 ega ka 2, vaid ruutjuur kahest. Sellel kohal funktsioonil z väärtust ei ole. Seetõttu tuleb sobilik väärtus kas interpoleerida (antud juhul -17 ja 3 vahelt, ca -10) või siis panna maski ruutu punktile lähemal asuv funktsiooni z tegelik väärtus (antud juhul -17). Tõsi, viimasel juhul pole mask enam nii sümmeetriline, kui ta saaks olla. Et koostada maski eespool toodud funktsiooni zz graafiku järgi, tuleb arvestada et sel juhul on maski tinglik nullpunkt nihutatud poole perioodi võrra ehk kohale 16. Järelikult, näiteks punkti 1 koordinaadiks on nüüd 17. See-eest on zz(j) graafikult selgesti näha, milline peaks maski läbilõige välja nägema.



Nagu näha funktsiooni zz(j) graafikult, on maski kordajad nullist erinevad maski kogu ulatuses. Teisisõnu, funktsiooni y(i) tekitatud sagedusmodulatsiooni täpse analoogi arvutamiseks koordinaatruumi maski abil peaks mask olema sama suur kui tema all olev pilt. Selliseid maske tavaliselt ei kasutata. Kui meenutada kasvõi eespool käsitletud pildi äärte probleemi, tekitavad liig suured maskid olemuslikult vajaduse suurendada ka tulemusena käsitletavat pilti, mis pole sugugi see, mida oodatud. Libiseva maski tehet kasutatakse üldjuhul ainult siis kui saab kasutada maski, mis on pildist oluliselt väiksem. Sel juhul muutub pildi äärte probleem praktiliselt ebaoluliseks ja ka arvutusteks kuluv aeg on oluliselt väiksem. Niisiis oleks meil tarvis täpselt vajalikku maski lähendada, nt maskiga laiusega 5.



Kasutame selleks funktsiooni zz(j) lähendamist. Tekitame lähendatud funktsiooni zzz(j), mis on nullist erinev ainult maski punktide (-2 kuni 2) kohal ehk, antud juhul, punktide 14 kuni 18 kohal, selle funktsiooni läbilõige on antud joonisel. Kui arvutame selle maski analoogi yy (i) DFT-ruumis, saame tulemuseks sagedusmodulatsiooni selle kuju, mis vastaks meie poolt tegelikult kasutatavale maskile. Kui sagedusmodulatsiooni uus kuju yy(i) sobib (võrdle esialgse kujuga y(i)), võime me kasutada sellist maski. Kui kuju ei sobi, võime püüda maski kordajaid muuta, eesmärgiga saada sobivamat sagedusruumi kuju.

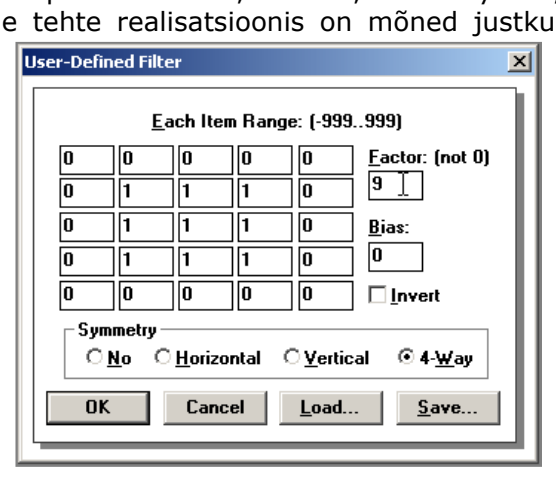
Tähelepanu tasub juhtida ka graafikule, kus kujutatakse yy(i) reaalosa. Sellel graafikul on näha võnkumine, mis yy(i) tavapärase esituse (moodul) juures taandub välja. Võnkumine tekkis vahepeal tehtud teisendusest y(i)=y(i)[-1 astmel i]. Kuid sellel teisendusel oli ka meeldiv pool: kuna ta nihutas funktsiooni z(j) pool perioodi, paistis vajalik mask paremini kätte. Aga ühtlasi on selge et poole perioodi võrra nihutatud koordinaatruumi funktsioonid (kus nullpunkt asub nüüd joonise keskel) võivad tekitada Fourier' ruumi funktsioonide reaal- ja ka imaginaarosades ootamatuid efekte.

Aare.Luts.1@eesti.ee

---

3. Libiseva maski tehte olemaolevad realisatsioonid.

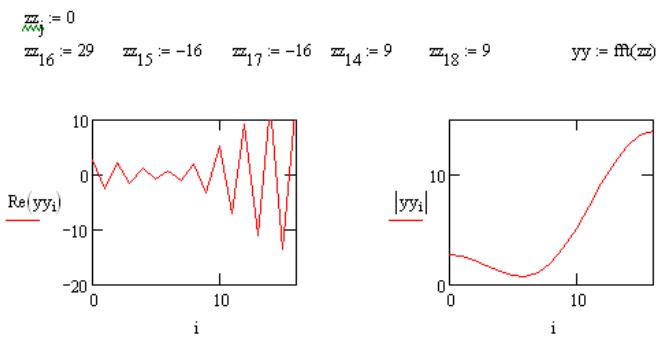
Libiseva maski tehe on realiseeritud nii kursuse Java-paketis kui ka, näiteks, PhotoStyler-s, viimases nimetatakse seda UserDefinedFilter. Selle tehte realisatsioonis on mõned justkui vabalt valitavad parameetrid. Nende parameetrite selgitamiseks tuleb puudutada maskile kehtivaid üldisi nõudeid.



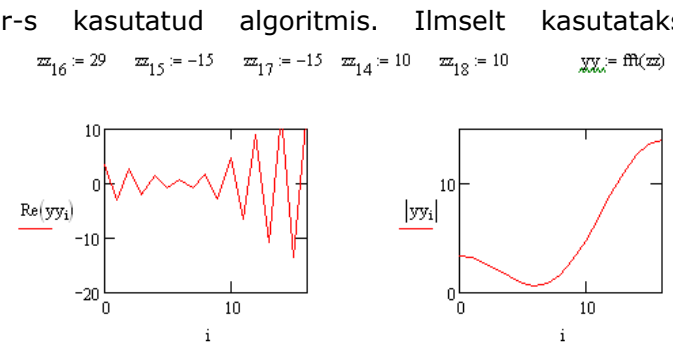
Nagu eespool selgitatud, mask peab sisaldama arvulisi väärtusi, mis sobivad tehte eesmärgiga. Kui mask on mõeldud olema sümmeetriline, peab sümmeetria kajastuma arvudes. Seejärel tuleb jälgida seda, et konstrueeritud mask (koos maskile omistatud tehtega, ja koos tehte rakendamise konkreetse metoodikaga) ei muudaks maski alla jääva "maastiku" (pildi) üldiseloomu ebasoovitavas suunas (muidugi kui pildi üldiseloomu muutmine ei ole eesmärgiks omaette). Konkreetsed reeglid sõltuvad maski abil tehtava tehte eesmärgist. Näiteks võib olla mõistlik vaadata, milline on maskitehte tulemus juhul kui mask liigub "maastiku" tasandiku (pildi samavärvi piirkonna) kohal. Kui sellel juhul on eesmärgiks et tehte tulemus peaks olema maastik ise (pildi sama värvi piirkonda ei muudeta), ja maskiga tehtavaks tehteks on sidum (paariskorrutiste summa), tuleb jälgida et maski kõigi kordajate summa oleks 1. Kui mask võib sisaldada reaalarvulisi kordajaid, tulebki sel juhul jälgida et nende summa oleks 1. PhotoStyler-i tehte maski kordajad on aga igal juhul täisarvud, seetõttu tuleb tema terminoloogias 1 saamiseks kordajate summa millegagi läbi jagada, jagajaks on parameeter "Factor". Vaatame ühte näidet.

Olgu meil teatav pilt, muudame teda maskidega
1) {(9,-16,9); (-16,29,-16); (9,-16,9)} ja
2) {(10,-15,10); (-15,29,-15); (10,-15,10)},
maskide rakendamiseks kasutame PhotoStyler tehet UserDefinedFilter. Esimesel juhul on maski kordajate summa 1, teisel juhul 8, millest võiks arvata et maskid käituvad mõnevõrra erinevalt, aga seda peaks saama ellimineerida "Factor" sobiva valikuga (siin 8). Muus osas paistavad maskid olema lähedased. Kuid, nagu näha kõrvalolevalt pildilt, tulemused on vägagi erinevad. Paraku, sel juhul ei ole erinevus tingitud mitte (niivõrd) maski kordajate

summade erinevusest. Kui maskid annavad oluliselt erineva tulemuse, peavad olema oluliselt erinevad ka nende ekvivalentfunktsioonid sagedusruumis. On üsna lihtne näha (kasvõi juba eespool kasutatud MathCad faili abil, kui seda veidi modifitseerida) et maskide sagedusruumi ekvivalendid erinevad üsna vähe. Seetõttu peab põhjus olema muus.



Antud juhul on põhjus PhotoStyler-s kasutatud algoritmis. Ilmselt kasutatakse täisarvaritmeetikat, koos võimalike ületäitumistega juhtudel, kui tulemid lähevad täisarvudele lubatud piirkonnast välja. See lähenemine avaldab tulemustele mõju eriti juhtudel kui tegemist on vaheldumisi negatiivsete ja positiivsete arvudega. Seega, antud juhul on tulemuste erinevuse (pea)põhjus arvutamisel kasutatud algoritmi erisused. Milleks seda siis üldse nii pikalt arutada? Aga selle pärast et midagi analoogilist võib juhtuda ka mujal, ja on oluline osata tulemusi mitmest küljest kontrollida. Antud juhul kasutasime võimalust kontrollida tulemust sagedusruumi ekvivalentteisenduste arvutamise abil, kust selgus et erinevus ei tohi nii suur olla.



Et efekt oli põhiliselt algoritmiline, sellele annab kinnitust ka samade tehete tegemine kursuse Java-keskkonnas, kus maskitehted on realiseeritud reaalarvaritmeetika abil. Selles saab määrata sobiva libiseva maski (antud juhul tuleb kasutada tehete järjestust "loo uus pilt mõõduga 3x3"-> "loe sisse maski sisaldav .prn-fail"-> "määra mask .prn-faili sisu alusel"-> "loe sisse muudetav pilt"). Uue 3x3 pildi ajutine loomine on vajalik seetõttu et tegelikult muudetava pildi sisselugemisel muudab pakett kõik süsteemsed maatriksid selle pildi mõõtu ja siis ei saa enam maski sisaldavat 3x3 maatriksit sisse lugeda, seetõttu tuleb esmalt luua maski mõõtu ajutine pilt ja mask lugeda sisse enne tegelikku pilti. Java keskkonnas saame mõlema maski puhul lähedased tulemused, mida oli sagedusruumi funktsioonide sarnasusest ka oodata. Veel kord, näite eesmärgiks ei olnud viidata PhotoStyler-i ebatäiuslikkusele. Näite eesmärgiks oli kasutada ebatäiuslikkust selleks et viidata vajadusele tulemusi mitmel viisil kontrollida, nagu ka näidata mõningaid võimalusi, kuidas saab tulemusi kontrollida.

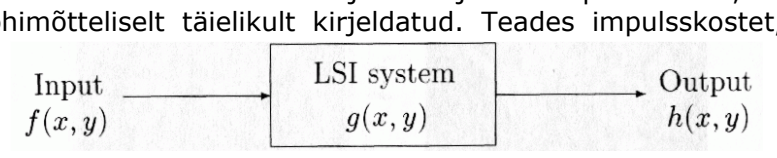
Aare.Luts.1@eesti.ee

4. LSI-tehted.

Libiseva maski sidumtehe kuulub tehete suurde ja olulisse klassi. Põhimõtteliselt võib iga tehet vaadelda "musta kastina", mille sisendjuhtmetele ühendatakse moduleeritav funktsioon ja mille väljundjuhtmetelt saab kätte moduleeritud funktsiooni. LSI (linear space invariant) tehted on "kastide" üks oluline klass. Igasuguste süsteemide toime kirjeldamisel on oluline koht deltafunktsioonil, mille üks definitsioon on toodud kõrval. Nagu näha, integraal suvalise funktsiooni f(x) ja deltafunktsiooni korrutisest on funktsioon f(x) ise deltafunktsiooni kohal. Kui me anname deltafunktsiooni "musta kasti" sisendisse, võime põhimõtteliselt saada mitmesuguseid väljundeid, sõltuvalt "kasti" sisust. Kui kehtib tingimus et väljundiks oleva funktsiooni kuju ei sõltu asukohast ruumis (argumentide väärtustest), loetakse süsteemi iseloomustajaks SI ehk space invariant. Sel juhul on nii et deltafunktsiooni δ sisestamisel süsteemi saame alati tulemuseks sama funktsiooni g, ainsa erinevusega et sel viisil saadud funktsiooni g argumentideks on deltafunktsiooni argumendid.

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)\,dx = f(x_0)$$

Kui me teame sisendisse antud deltafunktsioonist saadud väljundi kuju ehk impulsskostet, on süsteemi ("must kast") toime põhimõtteliselt täielikult kirjeldatud. Teades impulsskostet, saab alati arvutada mistahes teise funktsiooni muundumise süsteemis. See juhtum on esitatud järgmisel joonisel, kus süsteemi sisendisse antakse funktsioon f(x,y); süsteemi kirjeldab teadaolev impulsskoste g (x,y) ja tulemuseks on h(x,y). LSI süsteem peab rahuldama ka lineaarsuse (L) tingimust. Kui anda süsteemi sisendisse korraga mistahes kaks kordajatega a ja b moduleeritud funktsiooni, peab süsteemi vaste olema alati arvutatav mõlema funktsiooni eraldi arvutatud vastete samade kordajatega (a ja b) moduleeritud summaga. Sidumitehe rahuldab nii L kui la SI tingimusi, seega on sidum üks LSI operaatoritest.



$$a \cdot f_1(x,y) + b \cdot f_2(x,y) \Longrightarrow a \cdot h_1(x,y) + b \cdot h_2(x,y)$$

Oletagem nüüd et meil on mingi (optiline) süsteem, mis muundab teda läbivat pilti mingil sellele süsteemile omasel viisil. Kuna süsteemi sisendisse võib sattuda mistahes pilte, võib ka väljund olla mitmesugune. Loomulikult sõltub väljundis saadav (väljundpilt) sisendis olevast pildist, aga väljundpilt sõltub ka süsteemisisesest moonutustest. Tegelikult ei huvita meid pahatihti mitte süsteemi väljund, vaid pigemini see pilt, mis anti moonutava süsteemi sisendisse. Seega tuleks osata moonutust arvutada, siis ehk oskab pildiga toimunud moonutuse ka kõrvaldada. Oletagem et pilti moonutavat süsteemi võib käsitleda LSI-süsteemina. Sel juhul on moonutus kogu pildi ulatuses ühesugune ja kogu pildi moonutust võime vaadelda kui sisendpildi üksikute punktide moonutuste (impulsskostete) summat, kusjuures kõigi sisendpunktide moonutused on sama kujuga (kuna on SI-süsteem).

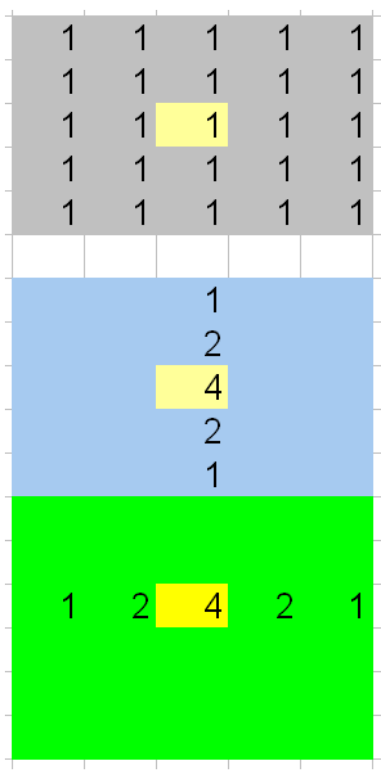
Küsimus: mis tuleks sellise (optilise) süsteemi juures ära mõõta, et hiljem saaks mõõdetut kasutada mistahes sisendpiltide moonutuste arvutamiseks? Kui see "miski" on ära mõõdetud, mis tehtega saaks "miskit" kasutades arvutada mistahes sisendpildist süsteemi väljundis saadava väljundpildi? Üks kiirvihje (lisaks lõpus soovitatud muule lisalugemisele).

5. Maskide kombineerimine.

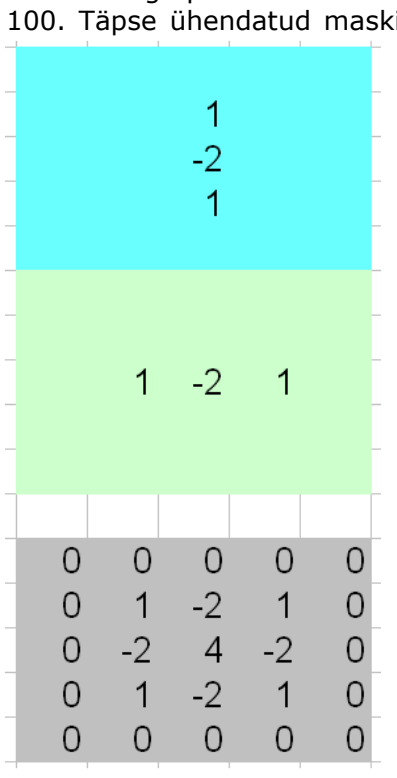
Maskide kombineerimisel tekitatakse kõiki soovitud tehteid korraga teostav mask, kasutades lähteandmetena üksikuid tehteid teostavaid maske. Kui eesmärgiks on 2D-mask, võivad aluseks olla nii 1D-maskid kui ka 2D-maskid. Väga suure tõenäosusega saavutatakse arvutiressursi kokkuhoid, aga see pole enam ehk peamine. Hoopis olulisem võib olla tulemus et kombineeritud maskilt on tehte mõte paremini hoomatav kui üksikutelt maskidelt eraldi võetuna. Samuti, kui eesmärgiks on näiteks sümmeetriline 2D mask ja aluseks on 1D maskid, õnnestub 2D mask teha rohkem sümmeetriliseks kui on 1D maskid eraldi võetuna. Ja loomulikult on maskide kombineerimise valdamine väga oluline teostatavate tehete mõistmise seisukohast.

Maskide kombineerimisel tuleb rangelt paika panna, mida tahetakse teha. Mõneks kombineerimiseks on vajalik sidumtehe, mõnede puhul on see aga ebasobiv. Sobivateks küsimusteks on: 1) "Kas üksikuid maskitehteid võib teha mistahes järjekorras"? 2) "Kas üksikute maskitehete järjestikusel tegemisel on vaja et uuel tehtel kasutataks sisendina alati eelmise tehte tulemusi"? 3) "Kas üksikuid maskitehteid võib teha niimoodi et kõigil tehetel on sisendiks üks ja seesama pilt, ning üksikute tehete tulemused võib lihtsalt kokku liita"? Kui maskitehete järjekord on oluline, viitab see juba asjaolule et ka teise küsimuse vastus on jaatav. Kui järjekord on oluline, aga teise küsimuse vastus ei ole jaatav, on tõenäoselt tegemist mingi segadusega, sel juhul tuleks ülesande üle veel mõelda.

Küsimustele vastamist võib olla abiks kui kasutada kõrvaltoodu laadis näiteid. Siintoodud näites on kaks maski, esimene sinisel ja teine rohelisel taustal. Hallil taustal on kujuteldav fragment pildist, lihtsuse huvides on selleks võetud võrdsete väärtustega ala. Maskide tööd on kasulik läbi mängida. Kui kõik kolm on kõrvuti, on see ehk lihtsam. Läbi on tarvis mängida ainult see, mis peaks juhtuma kollasega märgitud ruudu sisuga. Asja lihtsustab ka asjolu et pildil kollasega märgitud ruudu sisu muudetakse ainult siis kui kas ühe või mõlemate maskide kollased ruudud asuvad pildi kollase ruudu kohal (see ei pea ilmtingimata nii olema, aga üldjuhul on küll nii et maski tulem ehk koste kirjutatakse maski keskmise ruudu kohale). Kui vaadata korraga ühte maski kahest, on ka selge et pole vahet, mis suunas seda esmalt liigutada, lõpuks on selle maski tulem ikka sama (meenutan: sidumitehte tulemus kirjutatakse esmalt mitte kasutatavale pildile vaid puhvrisse, pildile kopeeritakse tulem alles siis kui mask on oma liikumised lõpetanud). Kohale, kus kollased ruudud on kohakuti, satub üks mask kogu oma liikumise kestel ainult ühel korral. Seega polegi kogu liikumist tarvis ilmtingimata läbi mängida. Esimese maski koste kollasesse ruutu on siinantud juhul arvutatav valemiga (1*1 +1*2 +1*4 +1*2 +1*1) =10.



Nüüd saab küsida et mis edasi? Kas enne teise maski rakendamist on ilmtingimata vaja et saadud koste (10) oleks juba pildile tagasi kirjutatatud? Kui jah (tegemist on maskitehete kumulatiivse rakendamisega), on mõistlik ette kujutada näiteks rida, kus asub ka pildi kollane ruut. Selle rea kõik ruudud saavad sel juhul sisaldama 10. Nüüd vaatame, mis teeb teine mask. Nagu eespool, toimib ka see mask kõigile ruutudele ühtmoodi ja piisab täiesti, kui saada selgeks, mida ta teeb näiteks pildi kollase ruuduga. Seda ruutu muudetakse ainult teise maski ühes asendis, ja on lihtne näha et tulemiks on 100. Seega peab ühendatud mask sel juhul olema selline et tema kordajate summa oleks 100. Täpse ühendatud maski saab kätte, võttes mõlema maski sidumi. Näiteks, kui võtta **samasuunaliste** maskide (1,1) ja (1,1) sidum, saame tulemuseks maski (1,2,1). Selles näites on tegemist **erisuunaliste** maskidega, seega tuleb siin seda sidumi arvutamisel arvestada. Näide kahe erisuunalisest maskis ja nende sidumist on toodud kõrval (maskid sinisel ja rohelisel taustal, tulem hallil taustal).



Kui enne teise maski rakendamist pole esimese maski kostet tarvis pildile tagasi kirjutada, on pilt enne teise maski rakendamist endine ehk sisaldab "1"-d. Vaatame jälle kollast ruutu. Selles kohas kirjutatakse puhvrisse jällegi 10, aga kuna puhvris juba on 10, saab tulemiks 20. Miks liita? Aga selle pärast et sidumtehtega liidetakse, ainult et antud juhul mitte vahetult pildil, vaid puhvris. Nagu näha, on tulemid kahel juhul oluliselt erinevad. Sellel juhul saab ühendmaski sel viisil et tehakse maskidega sama mis tehti eelnevalt pildiga, ehk liidetakse maskid kokku. Kuna mõlemad maskid mõjuvad pildi kollasele ruudule ainult siis kui nende kollased ruudud on pildi ruuduga kohakuti, tuleb ka maskid kokku liita sellessamas asendis.

Eespool kirjeldatud juhtumeid võib vaadata ka sagedusruumi poolt. Kuna sidumi rakendamine on ekvivalentne sagedusruumi funktsioonide korrutamisega, on kumulatiivne sidumdamine mitme (erineva) maskiga ekvivalentne sagedusruumi mitme (erineva) funktsiooni korrutamisega. Kui näiteks tehakse kaks järjestikust sidumit (f*g)*h, on sagedusruumi ekvivalenttehteks (FG)H. Kui sidumeid ei ole tingimata tarvis kumulatiivselt rakendada vaid piisab (näiteks) üksikute maskide kostete kokkuliitmisest, tähendaks see sagedusruumis mitte üksikute maskide ekvivalentfunktsioonide korrutamist, vaid summat.

Seoseid sagedusruumiga saab endiselt modelleerida MathCad abil, näiteks kasutades juba eespool käsitletud töölehte. Siikohas tuleb küll arvestada et sellel töölehel tegeldakse 1D funktsioonide modelleerimisega. Kui saadavad 1D-maskid arendatakse 2D-maskideks, tähendab see tehniliselt et 2D maski koste arvutamiseks tuleb tegelikult rakendada mitut 1D-maski, 2D-maski koste saadakse 1D-maskide kostete kombineerimisel. Mil viisil maske kombineeritakse (nt kas kumulatiivselt või sõltumatult), sõltub valikust, nagu eespool kirjeldatud. Kuna MathCad selle töölehe abil saadud maskide lõiked on 1-dimensionaalsed, ja lõigete arvutustulemused kehtivad ka ainult 1D-juhul, saab maskide kombineerimisest saadava 2D-maski lõige igal juhul erinema 1D-lõigetest. 2-dimensionaalse maski kordajad saavad olema sellised nagu (MathCad abil arvutatud) 1D-maskide ühendamisreglitest järeldub. Põhimõtteliselt võib 2D-tehete seoseid arvutada ka MathCad 2D Fourier' teisenduste arvutamise kaudu, kahjuks on eriti tulemuste visualiserimine 1D-juhust palju ebameeldivam.

Küsimus: Eespool oli maskide ühendamisel võimalik kriteerium "kas maskitehete järjekord on oluline". Kas ja millal võiks tehete järjekord olla oluline (näited, põhjendus)?

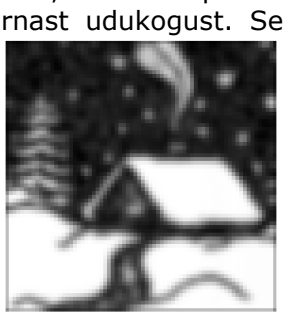
Ülesanne: arvutada ühendmaskid eespool pildil toodud näitemaskide (1,2,4,2,1) kahel juhul, nii siis kui ühendmaskiks on tarvis maskide liitmist kui ka siis, kui on tarvis maskide sidumit.

Ülesanne: arvutada maskide (1,-2,1) ja (1,1,1) sidumtehte kumulatiivse rakendamise tulem Mathcad vahenditega. Vihje(d): üsna mõistlik on lähtuda sidumi ja sagedusruumi korrutise ekvivalentsuse reeglist. Sellest saab reegli(d), kuidas sagedusruumis vajalikke tehteid teha. Loomulikult tuleb esmalt tekitada maskide sagedusruumi analoogid, aga selles osas võib võtta eeskuju juba varem mõnes failis olnud tehetest. Lõpuks, kui kõik vajalikud tehted sagedusruumis tehtud, tuleb tulem jällegi maski kujul esitada.

Aare.Luts.1@eesti.ee

6. Libiseva maski tehte pööratavus.

6. Libiseva maski tehte pööratavus. Mõnedes kriminaalfilmides on ikka ja jälle näiteid, kuidas udukogust kõik vajalikud näod kätte saab. Mõnikord võibolla saab ka, aga sellel on piirid, mõnesid piire siin ja praegu käsitlemegi. Kui udukogust juba juttu tuli, alustamegi õrnast udukogust. Selleks määrime laiali algpildi, kasutades libiseva maski sidumtehteid,



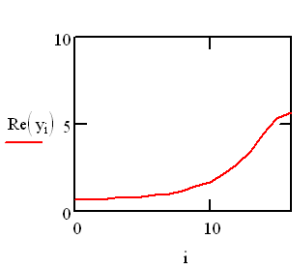
tulemuseks on kõrvalolev pilt. Nagu näha, on pilt tõepoolest mõningal määral laiali määritud ja võib tunduda küsitav kas temast enam kunagi midagi originaalile sarnast saab, eriti kui taastamiseks kasutada ainult libiseva maski sidumtehteid. Moonutuseks kasutati sümmeetrilist maski (2,5,2). Moonutatud pildi all on toodud teistsuguse maskiga sidumtehte tulemus, selle maski eesmärgiks oli laialimääritud pilt taastada. Võib täheldada et saadud pilt on originaalile märgatavalt lähedasem kui oli laialimääritud pilt. Seega, midagi justkui tõepoolest saab teha. Varasemalt

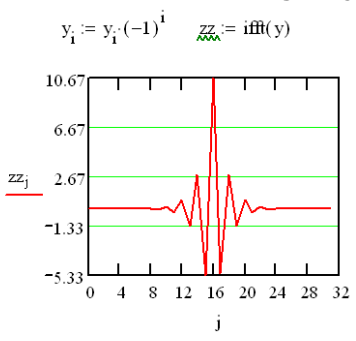
taastasime ka visuaalses mõttes veelgi rohkem moonutatud pilte, aga nende taastamiseks ei kasutatud libisevat maski, seetõttu ei lähe need siinkohas arvesse.

Nagu eestpoolt teada, on libiseva maski tehe pildi koordinaatruumis, h=f*g, ekvivalentne pildi ja maski sagedusruumi ekvivalentide korrutamisega sagedusruumis, H=FG. Seega on vastus küsimusele "kas libiseva maski tehe on pööratav" põhimõtteliselt ühene ja lihtne: kui sagedusruumis saab leida maski analoogfunktsiooni G pöördfunktsiooni 1/G, siis saab vähemalt põhimõtteliselt leida ka funktsioonile 1/G vastava maski ja teha libiseva maski tehte selle maskiga. Tulemuseks oleks sagedusruumis tehe FG (1/G) =F ehk esialgne funktsioon, järelikult saaks esialgse funktsiooni f kätte ka libiseva maski tehtega.

Paraku, eelkirjutatu puhul tuleb tihtipeale arvestada klauslit "põhimõtteliselt". Kui pöördfunktsiooni 1/G ei saa leida, on vastus ühene: seda tehet ei saa pöörata, ei sagedus- ega ka koordinaatruumis. Kui pöördfunktsiooni saab leida, asub esmatähtsaks teguriks (vahepeal toimunud) infokadu. Näiteks võib esialgne tehe olnud selline et "pressis kokku" teatava värvuste piirkonna või siis teatava sageduspiirkonna. Piltlikult, kui lähtepildis olnud piirkond (kolm pikslit väärtustega 20, 30, 40) on esialgse maskitehtega pressitud kokku piirkonnaks (30, 30, 30), siis on esialgses piirkonnas olnud info suure tõenäosusega kaduma läinud ja suure tõenäosusega pole enam kusagilt võtta, kas ja kui palju erinevad peaksid naaberpikslid olema. Mõnikord saab info taastada kaugematest pikslitest, aga mõnikord ka mitte. Põhimõtteliselt võib taastamine võimalik olla, aga tavaliselt on nii et kui taastava funktsiooni argumendiks anda samad arvud, on ka tulemuseks samad arvud, mitte üksteisest erinevad arvud, nagu oli esialgses pildis.

Teiseks võib taastav funktsioon 1/G olemas olla, aga tema ekvivalentmask võib olla väga suur ja selline et ka maski keskpunktist kaugel asuvad kordajad ei ole kaugeltki nullid. Kõrvalolevas näites on samuti tegemist moonutust kõrvaldava funktsiooniga ja ekvivalentmaskiga, mask pole just meeldivalt kitsas. Seekord on siiski sedamoodi et maski keskpunktist eemaldumisel vähenevad kiiresti ka kordajad, seega saab sobiva ja piisavalt väikese maski teatava lähendusega siiski koostada. Kui aga vajalik mask peaks olema peaaegu sama lai kui pilt, ei ole maskitehe teostatav, kasvõi eespool kirjeldatud pildi äärte probleemi tõttu. Kui täpsest, aga laiast maskist koostatakse selle väiksemate mõõtudega lähendus, tuleks alati modelleerida, kui palju saab lähendatud maski mõju täpselt vajaliku maski mõjust erinema. Modelleerimisnäide on toodud nt sellel MathCad töölehel. Kui erinevus saab olema suur, tuleb lähendust parandada või kui see ei õnnestu, tuleb tõdeda et vaadeldavat moonutust (praktiliselt) ei saa libiseva maski tehtega pöörata.





Ülesanne: Selles failis on eelpoolnimetatud maskiga (2,5,2) laialimääritud pilt arvkujul, selle saab sisse lugeda nii kursuse Java-paketti kui ka nt Excel-sse. Leida

mask, millega saaks pildi taastada vähemalt eespool toodud näitega võrreldaval määral. Vihjed: arvatavasti oleks mõistlik alustada eelmistes lõikudes käsitletutega analoogilistest MathCad-teisendustest. Maskist saab arvutada esialgset pilti sagedusruumis moduleerinud funktsiooni. Siis tuleb vastata küsimusele, millist teistsugust moduleerivat funktsiooni oleks vaja et toimunud sagedusmoonutus kõrvaldada. Seejärel saab arvutada sellele funktsioonile vastava maski, mis tuleb ka vormistada, arvestades punktis "maskide kombineerimine" käsitletut. Loomulikult tuleb maski tööd ka praktiliselt katsetada, milleks sobib nt kursuse Java-pakett, aga midagi saab teha isegi Excel-s. Siin failis on antud Excel makro 2D-sidumi arvutamiseks. Kui makrole andmed ette anda, saab (maske) arvkujul katsetada.

Küsimused: Tooge välja vähemalt üks juhtum, kus pildiga tehtud tehet ei saa juba põhimõtteliselt pöörata, ükskõik kas koordinaat- või sagedusruumi näitel. Milliseid järgnevalt nimetatud koordinaatruumis teostatud libiseva maski tehteid saab pöörata, milliseid mitte ja miks? Kui võiks saada, aga on problemaatiline, kirjeldada ka suuremaid probleeme. Maskid on (kõiki kujusid käsitleda püsttelje suunaliste, maski nullpunkti läbivate lõigetena sümmeetrilistest maskidest):
(1,1,1); (5,5,5); (0,-1,0); (1,-1,1); (1,4,16,4,1).

Lisalugemist:
LSI-süsteemide kohta: (1) , (2) , (3) , (4) , Google otsing märksõnadega "linear space invariant systems".
2D-Fourier' teisenduste kohta: (1) .

Aare.Luts.1@eesti.ee

Teema 5 küsimused ja ülesanded on järgmised:

1.1. Kas libiseva maski tehte tulemuse võiks kirjutada tagasi otse maski all olevale "maastikule" või pigemini teatavasse puhvrisse? Kui puhvrisse, siis miks? Meenutan: maski tehte sagedaseks eesmärgiks on muuta "maastikku" ennast ehk kasutada maski tehet pildi teatavaks muutmiseks. Seetõttu tuleb tulem niikuinii varem või hiljem "maastikule" tagasi kirjutada.

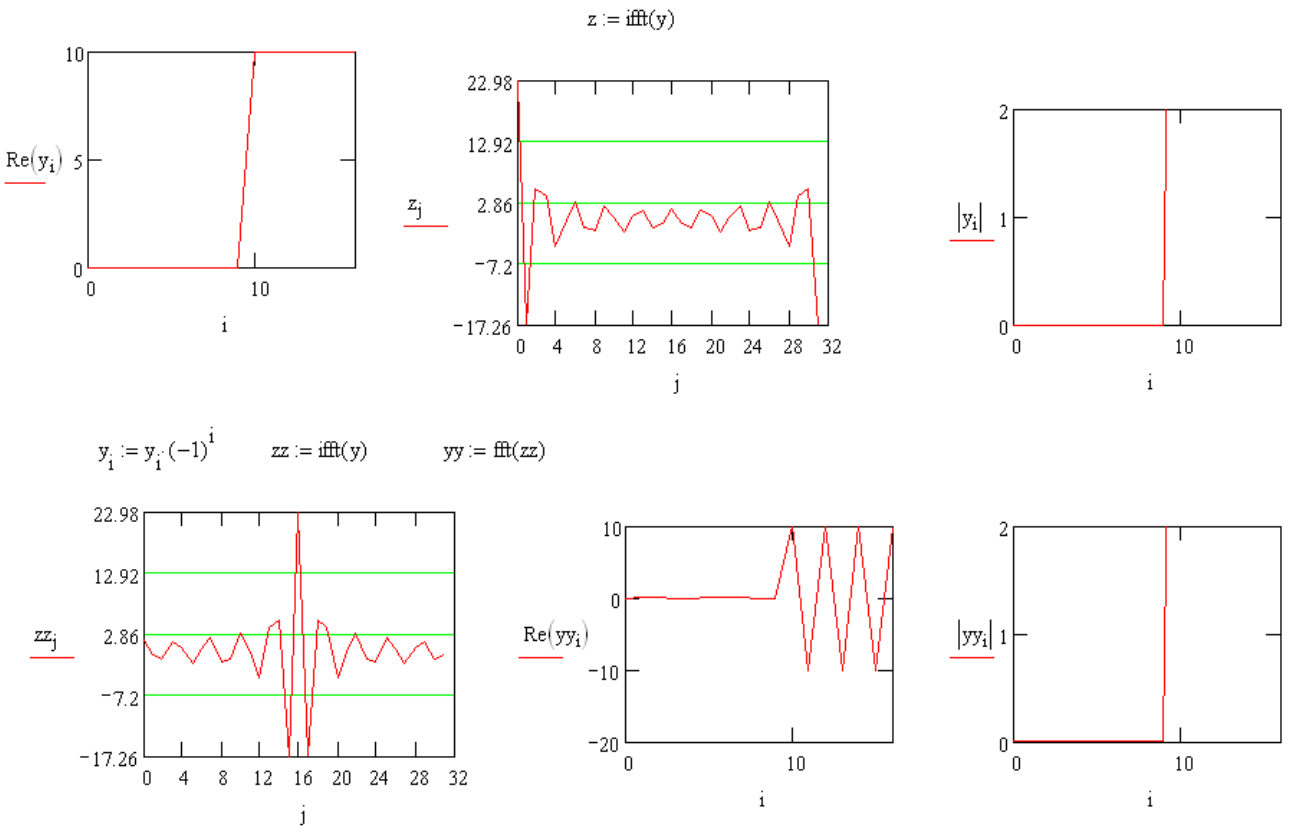
1.2. Olgu meil LSI optiline süsteem. Võib vaadata sedamoodi et süsteemi väljundiks on süsteemisisese moonutuse sidum sisendpildiga. Mis tuleks sellise süsteemi juures ära mõõta, et hiljem saaks mõõdetut kasutada mistahes sisendpiltide moonutuste arvutamiseks? Kui see "miski" on ära mõõdetud, kuidas saaks "miskit" kasutades arvutada mistahes sisendpildist süsteemi väljundis saadava väljundpildi? Üks kiirvihje (lisaks konspekti lõpus soovitatud muule lisalugemisele).

1.3. Maskide ühendusviisi valiku juures oli üks võimalik kriteerium "kas maskitehete järjekord on oluline". Kas ja millal võiks tehete järjekord olla oluline (näited, põhjendus)?

1.4. Tooge välja juhtumeid, kus pildiga tehtud tehet ei saa juba põhimõtteliselt pöörata, vähemalt üks nii koordinaat- kui sagedusruumi tehte näitel. Põhjendusega.

1.5. Milliseid järgnevalt nimetatud koordinaatruumis teostatud libiseva maski tehteid saab pöörata, milliseid mitte ja miks? Kui pöörata võiks saada, aga on problemaatiline, kirjeldada ka suuremaid probleeme. Maskid on (kõiki kujusid käsitleda püsttelje suunaliste, maski nullpunkti läbivate lõigetena sümmeetrilistest maskidest): (1,1,1); (5,5,5); (0,-1,0); (1,-1,1); (1,4,16,4,1).

2.1. Defineerida sagedusruumis kõrgeid sagedusi nulliv funktsioon (joonisel oleva funktsiooni y(i) analoog) ja arvutada sellise



sagedusmodulatsiooniga samaväärset libiseva maski tehet tegev koordinaatruumis toimiv mask (funktsioonide z(j) või zz(j) analoog). Kui vajaliku maski laius osutub olevat suurem kui 5, koostada lähendav mask (z(j) või zz(j) kärbitud versioon) ja arvutada lähendava maski poolt tehtavale tehtele samaväärne sagedusmodulatsioon. Võrrelda esialgu soovitud ja lõpuks, lähendatud maski kaudu, saadud sagedusmodulatsioone. Saavutada olukord kus tegelikult saadud sagedusmodulatsioon ei oleks soovitust liiga erinev.

2.2. Olgu meil teatav pilt, muudame teda maskidega
1) {(9,-16,9); (-16,29,-16); (9,-16,9)} ja
2) {(10,-15,10); (-15,29,-15); (10,-15,10)}.
Konspektis oli kirjas et mõlemad maskid peaksid andma ligilähedase tulemuse. Kontrollida, kas see ikka on nii (**mitte PhotoStyler abil**).

2.3. Olgu meil nii horisontaalne kui ka vertikaalne 1D mask (1,2,4,2,1). Arvutada nende kahe maski ühendmaskid kahel juhul, nii siis kui ühendmaskiks on tarvis maskide liitmist kui ka siis, kui on tarvis maskide sidumit.

2.4. Arvutada samasuunaliste 1D maskide (1,-2,1) ja (1,1,1) sidumtehte kumulatiivse rakendamise tulem Mathcad vahenditega. Vihje(d): üsna mõistlik on lähtuda sidumi ja sagedusruumi korrutise ekvivalentsuse reeglist. Sellest saab reegli(d), kuidas sagedusruumis vajalikke tehteid teha. Loomulikult tuleb esmalt tekitada maskide sagedusruumi analoogid, aga selles osas võib võtta eeskuju juba varem näiteks

selles failis olnud tehetest. Lõpuks, kui kõik vajalikud tehted sagedusruumis tehtud, siis tulem jällegi maski kujul esitada.

2.5. Selles failis on 2D maskiga, mille läbilõige on (2,5,2), laialimääritud pilt arvkujul, selle saab sisse lugeda nii kursuse Java-paketti kui ka nt Excel-sse. Kõrval on laialimääritud pilt pildi kujul. Leida mask, millega saaks pildi taastada vähemalt konspektis toodud näitega võrreldaval määral. Vihjed: arvatavasti oleks mõistlik alustada konspektis käsitletutega analoogilistest MathCad-teisendustest. Maskist saab arvutada esialgset pilti sagedusruumis moduleerinud funktsiooni. Siis tuleb vastata küsimusele, millist teistsugust moduleerivat funktsiooni oleks vaja et toimunud sagedusmoonutus kõrvaldada. Seejärel saab arvutada sellele funktsioonile vastava maski, mis tuleb ka vormistada, arvestades maskide kombineerimise reegleid. Loomulikult tuleb maski tööd ka praktiliselt katsetada, milleks sobib nt kursuse Java-pakett, aga midagi saab teha isegi Excel-s. Siin failis on antud Excel makro 2D-sidumi arvutamiseks. Kui makrole andmed ette anda, saab (maske) arvkujul katsetada.

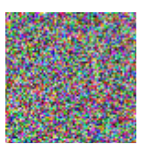
6.teema õppematerjalid.

Selles raamatus vaadeldakse müra olemust ja tema vähendamise meetodeid. Samuti kirjeldatakse piirjoonte esiletoomise põhilisi viise. Müra väiksemaks. Ühtlane müra ja selle kõrvaldamine. Müra kõrvaldamise muud meetodid. Moonutuste kõrvaldamise matemaatilised meetodid. Piirjooned pildile. (Piir)joonte detekteerimise elementaarmaskid. (Piir)joonte maskide kõrvalmõju. Joone detekteerimise muud tehnikad. Joonepunktide (läviväärtuste) määramine. Läviväärtuste määramine lokaalselt.

Õpikeskkond: TÜ Moodle
Kursus: Pildiinfo töötlus (LOFY.05.055)
Koosta raamat: 6.teema õppematerjalid.
Printed by: Aare Luts
Kuupäev: teisipäev, 22 mai 2012, 08:22

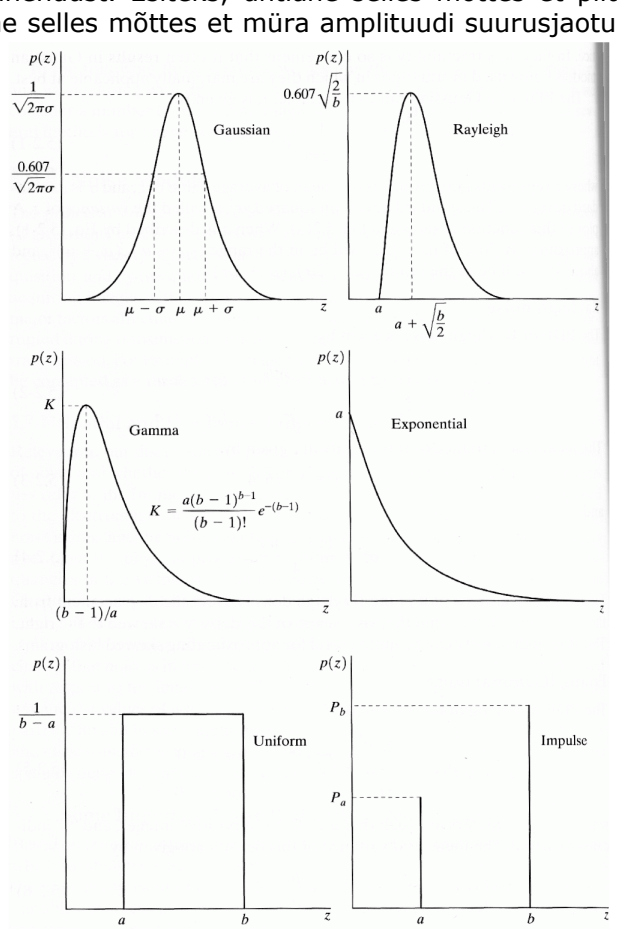
Sisukord

1. Ühtlane müra ja selle kõrvaldamine.

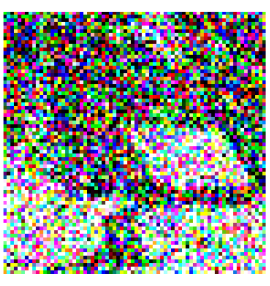


Kõigepealt, mis on müra (pildil)? Ei midagi muud kui et reaalse maailma nähtustest tulenev "puhas" sisu on vähem või rohkem varjutatud juhusliku signaaliga, mis ei tulene pildi aluseks olevatest nähtustest vaid tuleneb mujalt. Signaalitöötluses on müra laialdaselt ja mitmekülgselt käsitletud. Seal eristatakse ka müra erinevaid "värvusi", nt valge müra, roosa müra, pruun müra (vt viiteid nt Wikipedia-st). Siin punktis vaadeldav ühtlane müra on lähim valge müraga, aga siin ei vaadata müra sellest aspektist, mida tehakse signaalitöötluses.

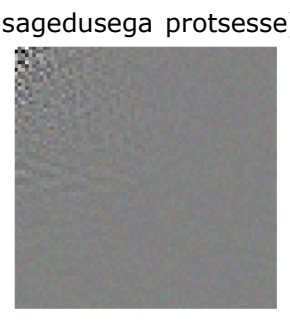
Müra "värvid" tulenevad müra jaotusest sagedusvahemike järgi, pilditöötluses pöörame me tähelepanu eelkõige sellele, kas mürast on (mingil määral) haaratud pildi kõik pikslid, või on mürast puudutatud ainult osa piksleid. Kui puudutatud on kõik pikslid, räägime ühtlasest mürast. "Ühtlasel" võib tegelikult olla vähemalt kaks tähendust. Esiteks, ühtlane selles mõttes et pildi kõik pikslid on temast haaratud. Teiseks, ühtlane selles mõttes et müra amplituudi suurusjaotus on ühtlane. Kui amplituudi suurusjaotus on ühtlane, esineb võrdse tõnäosusega nii suure amplituudiga müra kui ka väikese amplituudiga müra. On iseenesestmõistetav et suure amplituudiga müra moonutab pildil olevat infot rohkem. Müra amplituudi suurusjaotus võib olla mitmesugune. Võrdluseks võib tuua normaaljaotusega müra, kus müra väikesi amplituude esineb sagedamini, täpsemalt vastavalt normaaljaotusele. Seega võib pildil ühtlaselt jaotunud müra olla amplituudi mõttes nii ühtlase jaotusega kui ka normaaljaotusega, neid kahte käsitleme me samal viisil. Müra jaotustest amplituudi järgi käsitleme me erinevalt ainult üht spetsiifilist jaotust amplituudi järgi, sellest edaspidi.



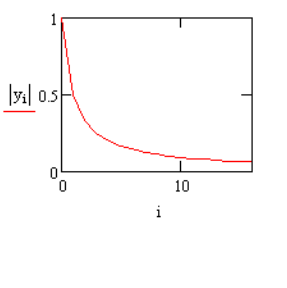
Müra konkreetne liik võib muutuda oluliseks kui kasutatakse pildi taastamise korrektseid valemeid. Mõnest valemist edaspidi. Tõsi, nende valemite juures on ka muid probleeme kui müra liigi määramine, aga kui miski on usaldusväärselt määratud, on üks liige valemis rohkem usaldusväärne ja abiks seegi. Müra mõnda liiki iseloomustavad müra suurusjaotused on toodud joonisel. Nende graafikute *z*-telgedel on müra väärtused ja *p (z)*-telgedel on tõenäosused, millega selle suurusega müra pildil esineb. Joonise alumises reas on ka eespool nimetatud ühtlane ehk "uniform" müra. Tõsi, siin on ta natuke teistsugune kui meie näidetes. Meie näidetes on ühtlase müra minimaalväärtuseks null, siin aga mitte.

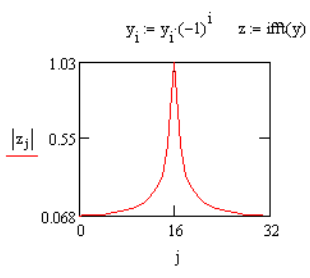



Kõrval toodud näidetes on originaalpildi ühtlase müraga mõjutatud versioonid, esimesel juhul on ühtlase jaotusega müra maksimaalne amplituud väiksem ja teisel suurem. Mida suurem amplituud, seda rohkem on originaalpilt mõjutatud. Müraste piltide DCT-kujudest (näitepildid müraste piltide all) on selge et mida rohkem müra, seda rohkem on pildil kõrgete sagedustega protsesse (DCT piltide ülemine vasak nurk kirjeldab madala sagedusega protsesse, alumine parem nurk kõrge sagedusega protsesse). Et väitele selgemat tõestust saada, võib võtta aluseks olevad mürased pildid 1 (väiksema amplituudiga müra) ja 2 (suurema amplituudiga müra), lugeda need sisse kuruse Java-paketti, arvutada DCT-pöörded ja pöörded arvkujul salvestada. Arvutabelist on lihtne näha, millisel juhul ja kui palju mingid arvud kas suuremad või siis väksemad on.

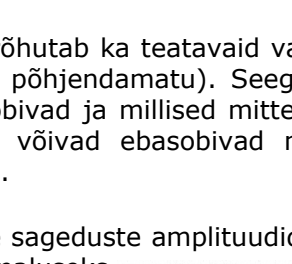


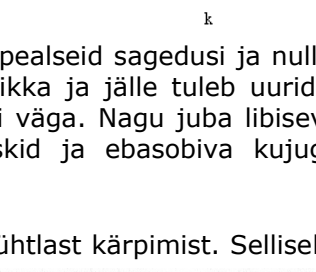

Järelikult on müra olemuslikult kõrge sagedusega protsess, millest järeldub et müra mahasurumiseks sagedusruumis tuleb maha suruda kõrge sagedusega nähtusi ehk kasutada madalpääsfiltrit (ingl. "low-pass filter"). Kuna müra on olemuslikult juhuslik nähtus, ei ole vähemalt ühest pildist lähtudes selle pildi täiuslik taastamine põhimõtteliselt võimalik, ühel eraldi võetud pildil olevat müra saab ainult maha suruda. Lähtudes teadmisest et müra mahasurumiseks tuleb maha suruda pildil olevaid kõrge sagedusega nähtusi, saab arvutada ka maski, mis teeks analoogilise teisenduse otse pildil. Maski konkreetne kuju sõltub funktsioonist, millega sagedusruumis kõrgeid sagedusi maha surutakse, aga alati on pildiruumi analoogiks pildi töötlemine teatava keskmistava maskiga, kus põhirõhk on vaadeldaval pikslil (maski keskel), aga kaasatakse ka naaberikslid. Selle maskiga arvutatakse teatav kaalutud keskmine maski alla jäävate pikslite väärtustest. Olemuslikult tähendab selline tegevus et justkui eeldatakse et iga piksli "õigest" väärtusest on mingi osa alles tema naaberpikslites, ja sedamoodi keskmistades õnnestub mürast rikutud osa mingil määral taastada. Võrdluseks, müra toimib põhimõtteliselt igale pikslile sõltumatult tema mõjust naaberpikslitele, seega taanduvad üle naaberpikslite keskmistades müra erinevad mõjud mingil määral välja.





Nagu mainitud, maski konkreetne kuju sõltub sagedusruumi funktsioonist. Õige on ka vastupidine: kasutatavast maskist tuleneb sagedusruumi analoogfunktsiooni kuju. Kui näiteks mitte kasutada maski, mille kordajate väärtused kahanevad koos kaugusega maski keskpunktist ühtlaselt (nagu oli eelmisel juhul, kuigi siis oli tegemist liig suure maskiga), vaid selle asemel kasutada maski, mille kordajad kahanevad järsult, saame sagedusruumi funktsiooniks sellise, mis küll rõhutab madalaid sagedusi (mis on õige), kuid mis rõhutab ka teatavaid vahepealseid sagedusi ja nullib mõned sagedused sootuks (mis on üldiselt põhjendamatu). Seega, ikka ja jälle tuleb uurida, millised maskid konkreetseks ülesandeks sobivad ja millised mitte nii väga. Nagu juba libiseva maski tehte käsitlemise juures vaadatud, võivad ebasobivad maskid ja ebasobiva kujuga sageduslõiked viia ootamatute tagajärgedeni.

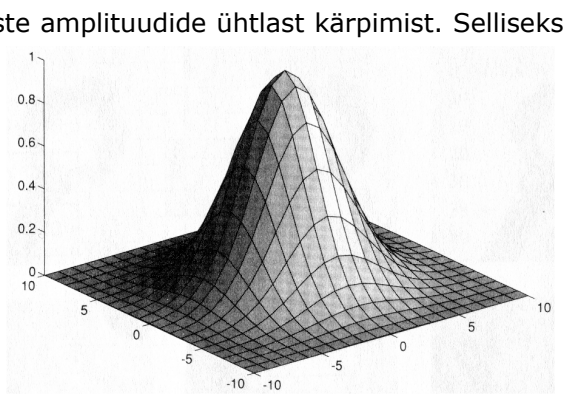




Üldiselt eelistatakse pildil olevate kõrgemate sageduste amplituudide ühtlast kärpimist. Selliseks otstarbeks on üheks heaks võimaluseks keskmistamine Gaussi kõverast tuletatud maskiga.

$$f(x) = \left[1/(2\pi\sigma^2)^{1/2}\right]\exp\left[-x^2/(2\sigma^2)\right]$$

$$F(\omega) = \exp(-\sigma^2\omega^2/2)$$



Sellise tehte üheks heaks omaduseks on see et tema mõju on sarnane nii pildiruumis kui ka sagedusruumis: kui pildiruumis kasutatakse Gaussi maski, on ka sagedusruumis sagedusi muutvaks funktsiooniks Gaussi funktsioon (vt ülaltoodud valemeid). Heaks omaduseks on ka see et Gaussi funktsioon pole kusagil null, seega säilitab ta mingi osa kõigist sagedustest ja, vastavalt eespool käsitletud libiseva maski tehte pööramise tingimustele, selline tehe on vähemalt põhimõtteliselt alati pööratav. Gaussi maski üks puudus tuleneb tema headest omadustest: kuna funktsioon pole kusagil null, on täpne mask väga suur, aga suuri maske tavaliselt ei kasutata. Täpsete maskide asemel kasutatakse Gaussi maskide lähendusi (vt näidet, mis on juba isegi 7*7 mask ehk küllaltki suur), aga need on siiski ainult lähendused.

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Ülesanded: 1) Tekitada võimalus varieeritavate parameetritega müra lisamiseks piltidele. Mingi võimalus on olemas nii Java-paketis kui ka Photostyler-s, aga parameetrite paremaks varieerimiseks oleks kasulik tekitada täiuslikum võimalus. Igati sobilik on kursuse Java-pakett, müra tekitamiseks sobib juhuslike arvude generaatori väljakutsumine, nt kujul *y=(float) Math.random();* (paneb reaalarvulise muutuja *y* väärtuseks juhusliku arvu nullist üheni).

2) Võtta aluseks mingi vabalt valitud pilt, lisada sellele ühtlase jaotusega müra vähemalt 3 tuntavalt erineva amplituudiga. Proovida müra kõrvaldamise filtreid. Müra millise taseme juures on pildis oleva info enamvähem taastamine veel võimalik?

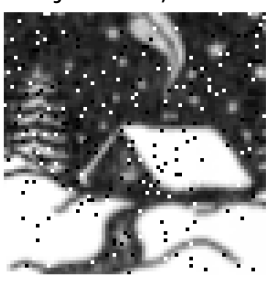
3) Proovida selle pildi müra mahasurumist sümmeetriliste maskidega (1,1,1) (3x3 mask) ja (1,1,1,1,1) (5x5 mask). Millest on tingitud erinevused? Proovida müra mahasurumist Gaussi maski lähendusega (Photostyler sisaldab vastavat otsetehet). Mis muutus ja miks?

4) Proovida selle pildi müra mahasurumist järsu sageduslõikega (järsku sageduslõiget saab teha Java-paketis, kus on olemas nii DCT tehe kui sellele järgnev sageduslõige, koos võimalusega sageduslõigatud pilt pöörd-DCT abil jälle pildikujule tagasi tuua). Millest on tingitud ootamatud efektid?

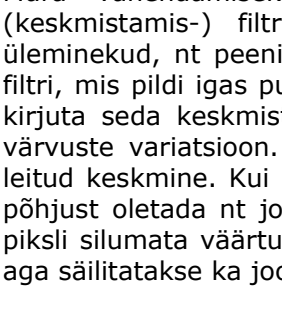
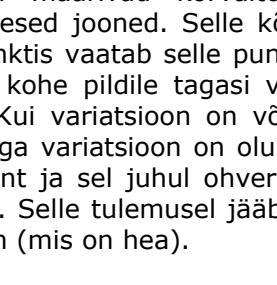
Aare.Luts.1@eesti.ee

2. Müra kõrvaldamise muud meetodid.

Eespool käsitlesime müra, mis, esiteks, on ühtlane selles mõttes et pildi kõik pikslid on temast haaratud ja, teiseks, ühtlane selles mõttes et müra amplituudi suurusjaotus on ühtlane. Impulssmüra erineb ühtlasest mürast selle poolest et tema jaotus amplituudi järgi on binaarne: neil juhtudel, kus müra esineb, on tema amplituud maksimaalne võimalikest. Võib eristada

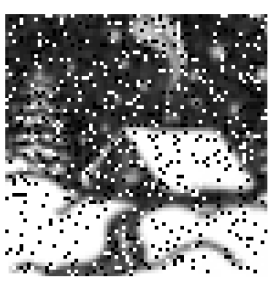



ühepoolse ja kahepoolse amplituudiga müra. Esimesel juhul asendab müra esialgse sisu nt maksimaalse väärtusega, teisel juhul vahelduvalt kas minimaalse või maksimaalse väärtusega. Näitepiltidel on kahepoolse amplituudiga müra. Kuna impulssmüra puhul on osa piksleid mürast täiesti puutumata, võib asja vaadata ka sellest küljest et tegemist ei ole ühtlase müraga, põhjusel et pildi kõik pikslid pole mürast haaratud. See on ka põhjus, miks siin on impulssmüra toodud ühtlasest mürast eraldi. Kuid asja võib vaadata ka sellest küljest et kõik pikslid siiski on mürast

haaratud, ainult et müra binaarne suurusjaotus tingib selle et osal pikslitest müra ei avaldu. Kuna müra on põhimõtteliselt juhuslik nähtus, avaldub müra juhuslikult ja sama juhuslikult võib ta ka mitte avalduda. Mida tõenäosem on müra, seda rohkem piksleid saab mürast mõjutatud.

Impulssmüra kõrvaldamiseks naabruse keskmistamisel põhinevad maskid ei sobi, hoolimata asjaolust et impulssmüraga piltide sageduskujud näivad üsna sarnased ühtlase müraga piltide sageduskujule (vt näitepilte eelmises punktis ja siin). Kuna nüüd avaldub müra teravalt väljendatult pikslikaupa, võiks ju olla et suurima sagedusega protsesse on nüüd mõnevõrra rohkem, seega peaks sobiv sagedusfilter olema kõrgemate sageduste poolt järsema langusega. Samas pole eespool tehtud põhimõtteline eeldus et mingi osa piksli sisust peaks olema säilunud kõigis naaberpikslites, nüüd täidetud. Impulssmürast mõjutatud naaberpikslites on nüüd kadunud kogu esialgne info. Seetõttu ei sobi keskmistamine põhimõtteliselt, kuna keskmistamisega levitatakse infot, mis pole oma olemuselt enam õige. Siiski, võib teha eelduse et mõnedes naaberpikslites on osa piksli esialgset väärtust sisaldavast infost säilunud. Aga nüüd ei saa seda infot kätte keskmistamise abil vaid näiteks sel moel et vaadatakse naaberpiksleid kui tervikut ja jäetakse kõrvale terviku keskväärtusest väga erinevad pikslid, pidades neid riknenuteks (mis impulssmüra puhul võib olla õige). Sobivaks libiseva maskiga seotud tehteks oleks nüüd mediaan.



Toodud näites on toodud impulssmüraga mõõdukalt rikutud pildi taastamiskatse keskmistamisega ja mediaaniga. On näha et mediaaniga saadud tulemus on parem.

Mõlemal juhul, nii ühtlase kui ka impulssmüra puhul, võib müra maha suruda piltide liitmise teel. Nüüd saab rääkida ka info taastamisest, mitte ainult müra mahasurumisest. Kuna müra on olemuselt juhuslik, peaksid erinevate piltide puhul samale pikslile mõjunud mürad kompenseeruma või vähemalt vähenema. Kui müra ei ole mitte liiga intensiivne, lisab iga uus liidetud pilt kasulikku informatsiooni ja vähendab müra mõju. Matemaatiliselt saab näidata et teatavate eelduste korral väheneb müra mõju võrdeliselt ruutjuurega piltide arvust. Paraku saab liita ainult samast objektist ja samades tingimustes võetud pilte. Kui vähemalt üks neist, kas objektid või siis (valgustus)tingimused seeria jooksul muutuvad, ei ole piltide lihtviisline liitmine võimalik. Siiski on hulk juhtumeid kus liitmismeetodit ka praktiliselt kasutatakse, näiteks meditsiinilistes piltides, kus nii objekt kui ka tingimused püütakse hoida muutumatud.

Milliseid müra kõrvaldamise meetodeid me ka ei kasutaks, tuleb alati jälgida et nad rahuldaksid teatavaid üldisi nõudeid: **1)** asukoha säilitamine ehk meetodite kõrvalmõjuks ei tohi olla pildi oluliste objektide nihkumine; **2)** keskväärtuse säilitamine ehk müra kõrvaldamine ühte tooni pildilt (nt ühtlane hall) peaks andma tulemuseks sama tooni pildi (nt sama tooni halli); **3)** isotroopsus ehk meetod peab mõjuma kõigis suundades ühtemoodi. Kui pildil on näiteks erineva kaldenurgaga jooned, peavad joonte paratamatud moonutused olema kõigil joontel ühesugused.

Müra vähendamiseks võib kasutada ka muid kombineeritud filterid. Lihtsad silumis (keskmistamis-) filtrid määrivad kõrvaltoimena paratamatult laiali pildil olevad teravad üleminekud, nt peenikesed jooned. Selle kõrvaltoime vähendamiseks võib teha kombineeritud filtri, mis pildi igas punktis vaatab selle punkti ümbrust ja arvutab nt teatava keskmise, aga ei kirjuta seda keskmist kohe pildile tagasi vaid uurib enne, kui suur on selle punkti ümbruse värvuste variatsioon. Kui variatsioon on võrreldav pildi keskmise müraga, kirjutatakse tagasi leitud keskmine. Kui aga variatsioon on oluliselt suurem, ja tegemist ei ole impulssmüraga, on põhjust oletada nt joont ja sel juhul ohverdatakse müra mahasurumine ja kirjutatakse tagasi piksli silumata väärtus. Selle tulemusel jääb selles punktis müra küll puutumata (mis on halb), aga säilitatakse ka joon (mis on hea).

Ülesanded: 5) Võtta aluseks eespool ülesandes 2 valitud pilt, lisada sellele impulssmüra vähemalt 3 erineva sagedusega. Proovida müra kõrvaldamist keskmistamise ja mediaaniga. Müra millise taseme juures on pildis oleva info enamvähem taastamine veel võimalik?

6) Võtta aluseks eespool ülesandes 2 valitud pilt, lisada sellele ühtlase jaotusega müra vähemalt 3 tuntavalt erineva amplituudiga. Nüüd tekitada sama amplituudiga vähemalt 8 erinevat pilti (müra amplituud kõigil 8 juhul sama, aga pildid ise tulevad erinevad, tingituna müra juhuslikust iseloomust). Nüüd proovida pildi taastamist sama müra amplituudiga saadud erinevate piltide liitmise teel. Müra millise taseme juures ja mitme pildi liitmisel on pildis oleva info enamvähem taastamine veel võimalik?

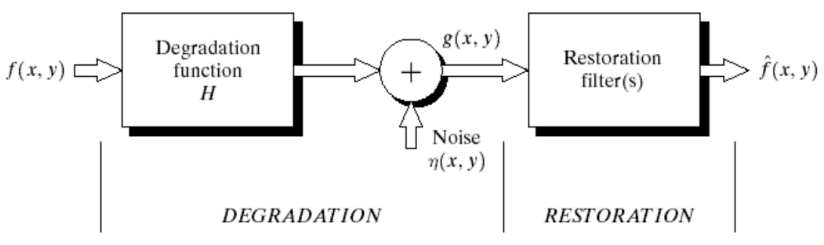
Konkreetseid teemasid puudutavaid kirjutisi võib leida, sisestades otsingumootorisse sobivad fraasid nagu "image noise (reduction)" jne. Mõned sel viisil leitud tulemid on allpool. Kas nad käesolevast konspektist just paremad on, aga igatahes on nad teistsugused ja ehk ka siinkirjutatut täiendavad.
http://en.wikipedia.org/wiki/White_noise
http://en.wikipedia.org/wiki/Colors_of_noise
http://en.wikipedia.org/wiki/Image_noise
http://www.michaelalmond.com/Articles/noise_print.html (noise reduction photographic tools).

Aare.Luts.1@eesti.ee

---

3. Moonutuste kõrvaldamine.

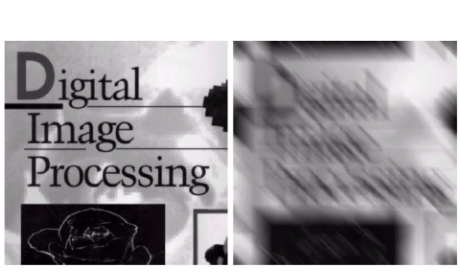
Selles peatükis vaatame moonutuste (ja müra) kõrvaldamist, kasutades selleks sagedusruumis tehtavaid arvutusi. Sellest tuleneb et nende meetodite rakendamine otse pildiruumis (nt maskide abil) ei pruugi võimalik olla. Joonisel on toodud pildiga toimunud moonutuste ja hilisema taastamise skemaatiliselt kujutatud mudel ja selle skeemi esitus valemite abil. Nagu näha,



läheb "moonutuste kasti" sisse originaalpilt *f(x,y)*. Esimeses etapis toimub selle pildi moondumine operaatoriga ***H***. ***H*** mõjule lisandub müra *η(x,y)*, tulemuseks on *g(x,y)*. Järgneb "taastamise kast", mille väljundis saadakse esialgse pildi hinnang *f(x,y)* (mitte esialgne pilt, vaid selle mingis mõttes parim hinnang). Valemitest on näha, kuidas moonutust ette kujutatakse. Esmalt toimub sidum moonutava funktsiooniga *h(x,y)* ja seejärel müra *η(x,y)* lisandumine. Seda moonutust võib kirjeldada ka sagedusruumis, kasutades teadaolevat seost sidumi ja korrutamise vahel, sagedusruumi esitus on antud alumise rea valemiga.

$$g(x,y) = h(x,y) * f(x,y) + n(x,y)$$
$$G(u,v) = H(u,v)F(u,v) + N(u,v)$$

Vaatame näiteks moonutust, mille tulem on toodud kõrvaloleval pildil. Tegemist on liikumisest tingitud moonutusega, antud juhul on liikumine toimunud pildi diagonaalsihis. Kui meil on originaalpilt *f(x,y)*, siis selle liikumisest tingitud moondumise saab kirja panna valemiga mis on toodud kõrval esimesena. Nagu näha, on tegemist teatava integraaliga üle aja, mille jooksul liikumine toimus. Kui me esitame selle valemi Fourier' ruumis, saab seda teha alljärgnevalt näidatud viisil. Fourier' ruumi valemite lõplikuks teisendamiseks on kasutatud nii teisenduse lineaarsust kui ka Fourier' ruumi ja valemite omadusi. Täpsemalt vt (allpool soovitatud) kirjandusest. Tulemuseks saame et esialgu pildiruumi integraali abil kirja pandud moonutuse saab Fourier' ruumis esitada teatava teise integraaliga, mille tulemiks on ***H(u,v)***. Fourier' ruum on kasulik selle poolest et siin kehtib funktsioonide korrutamine: moondunud pildi Fourier' kuju ***G(u,v) =H (u,v)•F(u,v)***, kus paremal poool on vastavalt moonutust teostanud funktsiooni ja esialgse pildi Fourier' kujud.



$$g(x,y) = \int_0^T f[x_0 - x(t), y_0 - y(t)]\,dt$$
$$G(u,v) = F(u,v)\int_0^T \exp[-j2\pi[ux(t) + vy(t)]]\,dt = F(u,v)H(u,v)$$

e.g. uniform horizontal motion:

$$x(t) = at/T; \quad y(t) = 0$$
$$H(u,v) = \int_0^T \exp[-j2\pi u\, at/T]\,dt = \frac{T}{\pi ua}\sin(\pi ua)\exp[-j\pi ua]$$

Kuna kehtib korrutamine, kehtib ka jagamine ehk Fourier' ruumis saab pildi taastada lihtsa jagamise abil, eeldusel mõistagi et ***H(u,v)*** kuju ehk liikumise kiirus, suund ja ajaline kestus on teada.

Ülal toodud moonutuse saab taastamisvalemite abil edukalt kõrvaldada.

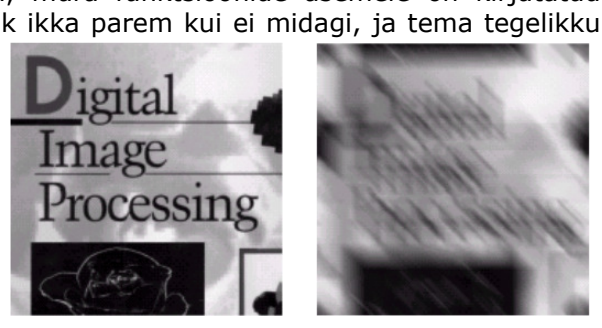
Pildi taastamist (sh müra kõrvaldamist) võib vaadelda sellest aspektist et püstitada ülesanne "millistel tingimustel on moonutunud pildi taastamisel saadud hinnang kõige lähedasem esialgse pildiga". See tingimus matemaatilisel kujul on antud kõrval. Meenutame, taastamise tulemusena ei saa üldjuhul kunagi täpselt esialgset pilti, saab ainult selle hinnangu. Ja võib küsida, millal on hinnang täpsele tulemusele kõige lähedasem. Kui valemitesse moonutust ja müra kirjeldavad funktsioonid sisse panna, saab selle matemaatilises mõttes korrektselt arvutada. Iseasi, on selle matemaatilise tulemusega pärast midagi peale hakata. Matemaatiline tulemus on toodud kõrval, ja sellega tema sellisel kujul ei olegi suuremat peale hakata. Et teda sellisena kasutada, peaksime teadma müra täpset jaotust pildil, aga kui me seda teaksime, võiksime ta otse maha

$$e^2 = E\{[f(x) - \hat{f}(x)]^2\} = \min$$

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma[S_\eta(u,v)/S_f(u,v)]}\right]G(u,v)$$
$$= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + \gamma[S_\eta(u,v)/S_f(u,v)]}\right]G(u,v)$$

$$\hat{F}(u,v) \approx \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]G(u,v)$$

lahutada ja keerulisi valemeid poleks üldse vaja. Teie asi on moonutava funktsiooniga *h(x,y)*, mis võib olla teada (nt eespool toodud juhul ta oligi teada), sel juhul on teada ka moonutava funktsiooni Fourier' kuju ***H(u,v)***. Müra kirjeldavad funktsioonid tuleb aga millegi muuga asendada, ja alumises valemis ongi seda tehtud, müra funktsioonide asemele on kirjutatud konstant ***K***. Loomulikult on see lihtsustus, aga ehk ikka parem kui ei midagi, ja tema tegelikku efekti saab ju piltide peal kontrollida. Kõrval on toodud üks näide, kus pilt oli moonutatud nii liikumise kui ka müra poolt (tõsi, müra ei ole väga intensiivne). Taastamise tulemus näitab et meetod, vähemalt sellisel juhul, justkui töötaks.



Et Wiener filtri rakendamiseks ei peaks Fourier' ruumi minema, selleks on välja arvutatud tema otse pildiruumi maski abil teostatav lähendus. Selle abil pole küll võimalik kõrvaldada funktsiooni *h(x,y)* tehtud moonutust, aga on võimalik vähendada müra. Valemis olevad suurused on järgmised. $\boldsymbol{A}_{i,j}$: filtreeritud (taastatud) piksel, kirjutatakse maski keskpunkti; $\boldsymbol{\mu}_{i,j}$: maskialune (lokaalne) keskväärtus; $\boldsymbol{\sigma}_{i,j}$: maskialuse ala variatsioon; ***s***: kogu pildi (keskmise) müra hinnang, tuleb ise valida ja vajadusel katsetada; $\boldsymbol{N}_{i,j}$: pildil oleva (mürast haaratud) piksli väärtus.

$$A_{i,j} = \mu_{i,j} + \frac{\sigma_{i,j} - s^2}{\sigma_{i,j}}(N_{i,j} - \mu_{i,j})$$

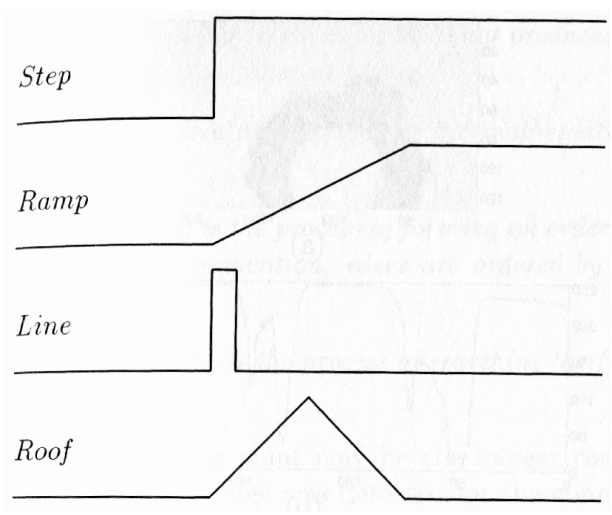
Ülesanne: Proovida müra kõrvaldamist Wiener filter lihtsustatud versiooniga (sellisega, mis ei nõua sagedusruumi kasutamist). Kas on erinevusi, kui võrrelda müra kõrvaldamise tulemustega eespool käsitletud meetoditel?

Loe lisaks:

Gonzalez and Woods, Digital Image processing, ptk. 5 (konspekt, slaidid).

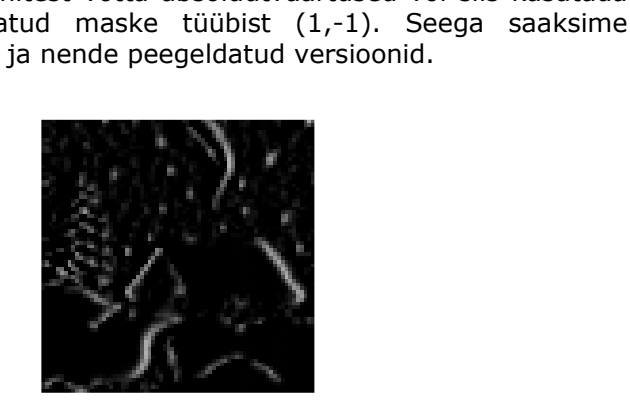
Aare.Luts.1@eesti.ee

4. (Piir)joonte detekteerimise elementaarmaskid.



Juba definitsiooni järgi on piirjoon ala, kus üks nähtus läheb üle teiseks. Piirjooni võib olla mitmesuguseid, aga igal juhul peab see ala olema pildil selgesti eristatav. Mõned võimalikest piirjoonte kujudest nendega risti olevas sihis on toodud kõrvaoleval pildil.

Läheneme piirjoonele matemaatiliselt. Valime piirjoone ühe võimaliku kuju *f(x,y)* ja arvutame esimese ja teise tuletise piirjoont iseloomustava karakteristliku suuruse (nt värvuse) muutumise sihis. Kuna pildi iseloom muutub joonel (ja/või selle ümbruses), aga mitte mööda joont, on sihiks, kus parameetrid muutuvad, joonega risti olev siht.



Tehe, millega matemaatikas uuritakse funktsioonide muutusi, on **funktsiooni tuletis**. Esimene ja teine tuletis piirjoont iseloomustava funktsiooni *f(x,y)* kujust on näidatud joonisel. Nagu näha, saavutab esimene tuletis maksimumi kohas, kus karakteristliku suuruse (nt heleduse) muutuse kiirus on suurim. Teine tuletis on esimese tuletise maksimumi kohas null, kuid omandab suured väärtused kahel pool seda kohta, seal, kus karakteristlik suurus kas hakkab muutuma või siis lõpetab muutumise.

Seega, et leida joonte asukohti **esimese tuletise** põhjal, tuleb otsida alasid, kus esimene tuletis on tavalisest oluliselt suurem, ehk ületab teatava läviväärtuse. Päris null pole esimene tuletis tõenäoselt kusagil, põhjusel et juba olemuselt on peaaegu igal pildil värvuste või heleduste muutused, ja igasugust muutust ei saa pidada jooneks. Joont on põhjust arvata seal, kus tuletise väärtus on tavalisest suurem. Tuletise arvutusvalem on üldiselt teada, seetõttu pole mõtet kogu arvutust läbi teha. Mainiks ainult nii palju et kuna digitaalpildi juhul on diskreet Δx=1 (üks piksel), jääb tuletise arvutamise valemisse ainult murrujoone pealne osa. Sellest valemist tuleneb kohe tehet teostav mask, selleks on $\mathbf{G}_x$=(-1,1).

$$G_x \cong f[i, j+1] - f[i, j]$$

$$G_x = \boxed{-1 \mid 1}$$

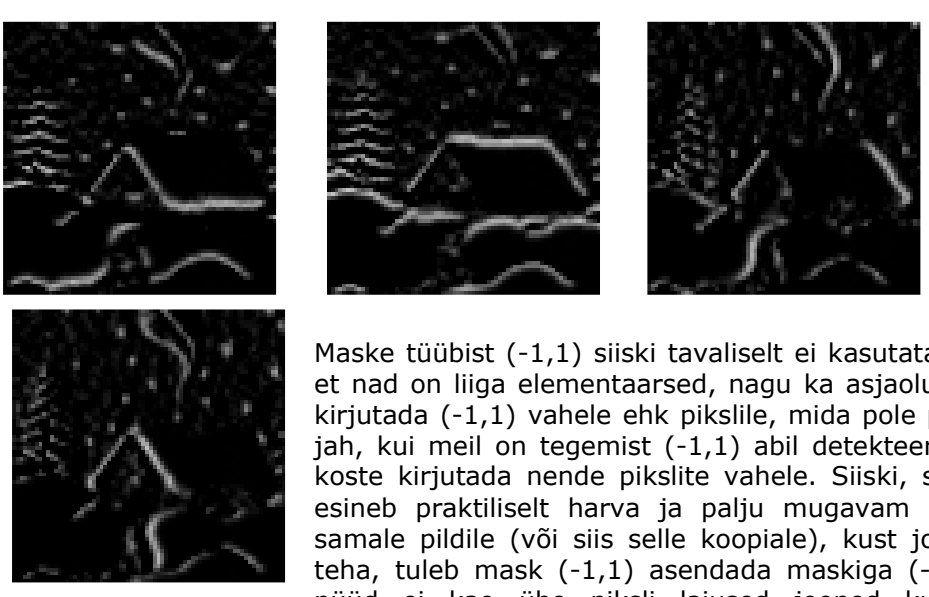
Nagu mainitud, karakteristliku suuruse muutumise funktsiooni vaadeldakse joonega risti olevas sihis. Seetõttu tunneb see mask ära jooned, mis kulgevad vertikaalselt, täpsemalt need jooned, milledel on vertikaalne komponent. Mida suurem vertikaalne komponent on, seda suurem on maskitehte tulem, kui aga vertikaalne komponent puudub (joon on x-telje sihiline), on selle maski tulem null ehk sel juhul joont ei detekteerita. Horisontaalsete joonte detekteerimiseks on tarvis eelmisega 90 kraadi võrra pööratud maski $\mathbf{G}_y$.

$$G_y = \boxed{\begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array}}$$

Eelnevast järeldub et kui me tahame parimal võimalikul viisil detekteerida diagonaaljooni, oleks tarvis ka maske, mis on pööratud 45 kraadi võrra. Selliseid tavaliselt siiski ei kasutata, põhjusel et $\mathbf{G}_x$ ja $\mathbf{G}_y$ koosrakendamisel leitakse mistahes sihis jooned üles, ainukese erinevusega et diagonaaljooned märgitakse nõrgematena. Siiski, kui $\mathbf{G}_x$ ja $\mathbf{G}_y$ tulemid kokku liita, saavad mistahes sihis jooned märgitud ka enamvähem võrdse tugevusega. Lisaks on nende abil võimalik hinnata joone kulgemise nurka α(x,y).

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Maski (-1,1) juures tuleks tähele panna veel seda et ta annab postiivse tulemi juhul kui joone ristlõige on sellise kujuga nagu eespool toodud joonisel ehk kui koordinaadi x kasvamisel karakteristliku funktsiooni väärtus kasvab. Pildi tavalises käsitluses tähendaks see et vasakul pool on tumedam ala ja paremal pool on heledam ala. Iseenesestmõista on joonega tegemist ka siis kui olukord on vastupidi ehk kui toimub heledama ala üleminek tumedamaks. Paraku, sel juhul annab mask (-1,1) negatiivse tulemi. Kuna joone olemasolu määratakse tavaliselt maski tulemi väärtuse järgi, võib nagatiivne tulem tähendada et joone olemasolu ei detekteerita. Et sellist olukorda vältida, tuleb kas maskide tulemitest võtta absoluutväärtused või siis kasutada lisaks maskidele tüübist (-1,1) ka peegeldatud maske tüübist (1,-1). Seega saaksime elementaarsel juhul neli vajalikku maski: $\mathbf{G}_x$, $\mathbf{G}_y$ ja nende peegeldatud versioonid.



Maske tüübist (-1,1) siiski tavaliselt ei kasutata. Põhjuseks on asjaolu et nad on liiga elementaarsed, nagu ka asjaolu et nende koste tuleks kirjutada (-1,1) vahele ehk pikslile, mida pole pildil olemas. Tegelikult jah, kui meil on tegemist (-1,1) abil detekteeritava joonega, tulekski koste kirjutada nende pikslite vahele. Siiski, sellise teravusega jooni esineb praktiliselt harva ja palju mugavam on ka tulem kirjutada samale pildile (või siis selle koopiale), kust jooni otsiti. Et nii saaks teha, tuleb mask (-1,1) asendada maskiga (-1,0,1). On näha et ka nüüd ei kao ühe piksli laiused jooned kuhugi, ainult et nüüd kirjutatakse selliste joonte koste mitte täpselt joonele vaid joone äärde. Ka maskid (-1,0,1) tunnevad ainult üht joonetüüpi neljast, analoogiliselt eeltoodule. Nende maskide kosted ühe konkreetse pildi korral on toodud ülal. Kõigi joonte detekteerimiseks tuleks igas punktis (pikslil) rakendada kõiki nelja maski, nende kostetest valitakse tavaliselt suurim väärtus.

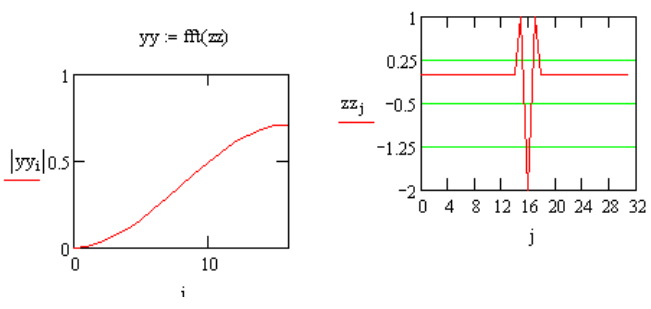
Nüüd vaatame, mis saab, kui kasutada **teist tuletist**. Ka teise tuletise arvutamine pole miski ületamatult keeruline, aga olgu ta täieliku selguse huvides siinkohas siiski ära toodud. Nagu näha, on tulemuseks mask (1,-2,1). Kui võrrelda seda maskiga (-1,0,1), on näha et uus mask on sümmeetriline, erinevalt eelnevast. Sellest järeldub muu hulgas et nt maski (-1,0,1) peegeldatud versioon (1,0,-1) pole nüüd vajalik, mask (1,-2,1) detekteerib nii üleminekud tumedamalt heledamale kui ka üleminekud heledamalt tumedamale. Ka see mask detekteerib ainult neid jooni, millel on olemas teatava sihiline komponent, antud juhul peab olemas olema y-telje sihiline komponent. Kõigis sihtides joonte leidmiseks on ka nüüd tarvis 90 kraadi võrra pööratud maski, aga nüüd pole tarvis teha mitut maskitehet järjest (ja üksikuid tulemusi kokku liita või leida nende maksimumi), vaid nüüd saab koostada maski, mis teeb kõik tehted korraga. Antud juhul on tarvis (korraga) teha mitmes sihis maskide rakendamine ja tulemuste kokkuliitmine. Vastavalt eespool käsitletud maskide kombineerimise reeglitele on ühendmask selline nagu toodud kõrval.

$$\begin{aligned}
\frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} \\
&= \frac{\partial\,(f[i, j+1] - f[i, j])}{\partial x} \\
&= \frac{\partial f[i, j+1]}{\partial x} - \frac{\partial f[i, j]}{\partial x} \\
&= (f[i, j+2] - f[i, j+1]) - (f[i, j+1] - f[i, j]) \\
&= f[i, j+2] - 2f[i, j+1] + f[i, j].
\end{aligned}$$

$$\nabla^2 \approx \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

5. (Piir)joonte maskide kõrvalmõju.

Võtame aluseks teise tuletise kaudu leitud maski (1,-2,1). Juba tuttavate MathCad tehete abil saab näha et sagedusruumis on sellise maski ekvivalendiks kõrgete sageduste rõhutamine. Tegemist on kõrgpääsfiltriga. Tegelikult saanuks sama tulemuse ka Fourier' teisenduse omaduste tabelitest, kus on kirjas et nii pildiruumi esimese kui ka teise tuletise ekvivalentteheteks sagedusruumis on kõrgpääsfiltrid. Nüüd on selge, et meil on põhimõtteline probleem.



Eespool sai tegeldud idealiseeritud joontega, aga tegelikkuses võib pildi heleduste lõige mingis sihis olla näiteks selline kui toodud pildil. Tegelikkuses on mistahes objektid alati varjutatud kas müra või siis mõnede muude nähtustega, ehk idealiseeritud objekte praktiliselt ei esine. Kui pidada silmas ainult müra, on paslik meelde tuletada et ka müra oli oma olemuselt kõrgsageduslik nähtus. Järelikult, otsides jooni suurendame me ühtlasi pildil olevat müra, mis juba ilma suurendamiseta võib joonte äratundmist oluliselt takistada.

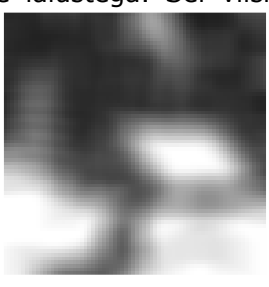
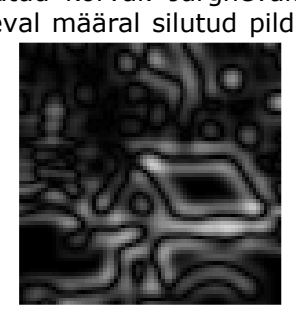
Peab märkima et ega selle metoodika raames müra ja joonte koosmõju probleemile ideaalset lahendust polegi. Kuna müra on kahtlemata segav tegur, tuleks tema mõju vähendada. Seda saab teha näiteks viisil et vähendada müra mõju joone sihis, jättes samas puutumata meetodi võime detekteerida jooni joontega risti olevas sihis. Sellise idee rakendamiseks saab koostada ühendmaski, mis kombineerib müra vähendamise ja joonte detekteerimise. Sellisteks, ka praktikas rakendatavateks, maskideks on näiteks Sobeli operaatorid, mis on toodud ülal. Nagu eespool märgitud, on mitmes mõttes eelistatud müra kõrvaldamise tehteks silumine Gaussi kõvera alusel koostatud maskidega. Ka seda tehet saab kombineerida joonte otsimisega, tulemuseks on nn LoG operaator, mille 3D-kuju ja 2D-mask on toodud joonistel.

$$s_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad s_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$\begin{array}{|ccccc|}\hline 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ \hline \end{array}$$

6. Joone detekteerimise muud tehnikad.

Eeltoodud meetodite (maskidega) võrreldava tulemuse võib saada ka, kasutades mõnevõrra teistsugust lähenemist. Teeme originaalpildist mitu keskmistatud (silutud) varianti, näiteks Gaussi maski erinevate laiustega. Sel viisil saadud kaks pilti on näidatud kõrval. Järgnevalt arvutame nende kahe erineval määral silutud pildi erinevuse ehk lahutame ühe pildi teisest, salvestades mitte saadud vahe enda vaid saadud vahe absoluutväärtuse. Tulemus on näha kolmandal pildil. Nagu näha, võib sel viisil saadud kolmandal pildil tõepoolest täheldada visuaalses mõttes rõhutatud jooni.

Viimatimainitud meetod on teoreetiliselt üsna lihtsasti põhjendatav. Kui me kasutame silumist siis sagedusruumi mõttes kasutame me madalpääsfiltrit. Erinevad silumised on sarnased selles mõttes et nad kõik säilitavad madalate sagedustega nähtused ja erinevad selles mõttes et vähendavad kõrgete sageduste osakaalu erineval määral. Nõrk silumine vähendab kõrgeid sagedusi vähe, kuid tugev silumine vähendab neid palju. Seega, kui me arvutame kahe erinevalt silutud pildi erinevuse siis esimeses lähenduses arvutame me kahe pildi erinevuse, milledel on erineval määral kõrgeid sagedusi. Järelikult saab erinevus sisaldama rohkem kõrgeid sagedusi kui madalaid, ehk sel viisil saadaval pildil peaksid jooned olema rõhutatud, mida me ka täheldame.

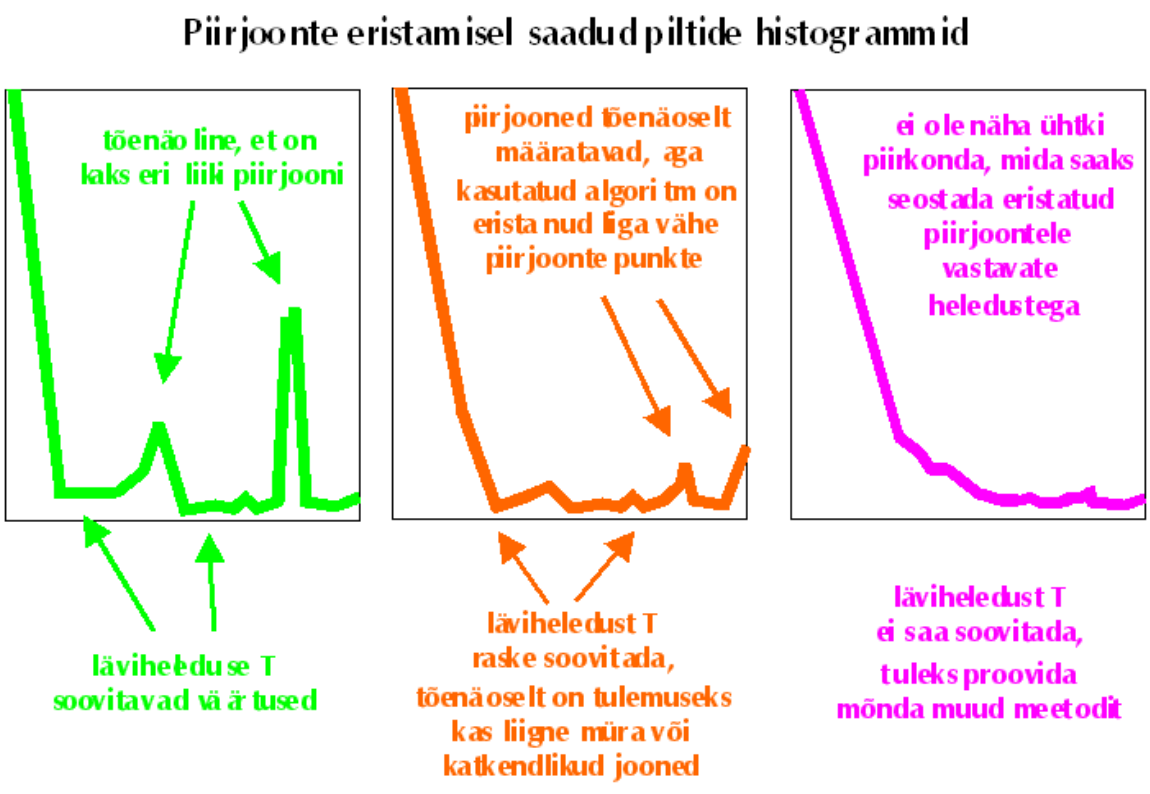
Praktikas võib piirjooni olla palju liike. Näiteks võib piirjoone all mõista ala, kus üht liiki muster asendub teisega. Silmaga on ka sellised piirjooned üsna märgatavad, paraku mitte eelkirjeldatud meetoditega. Kui piirjooneks on näiteks üleminek ühelt mustrilt teisele, siis tuleb üle pildi arvutada mingeid mustreid iesloomustavaid suurusi, sel juhul saab piirjooneks ala, kus arvutatud suuruste väärtused muutuvad. Mustreid käsitletakse edaspidi.

Aare.Luts.1@eesti.ee

7. Joonepunktide (läviväärtuste) määramine.

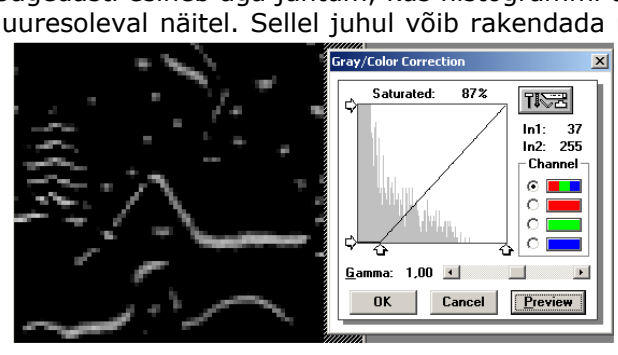
Eelkirjeldatud joonte detekteerimise maskide rakendamise tulemina saadakse üldjuhul uus pilt, kus joonte asukohad on märgitud endisest erineva värvusega (lihtsamal juhul: joonte asukohad on muust pildist heledamad). Selline uus pilt ei ütle midagi üheselt mõistetavat joonte olemasolu või puudumise kohta. Pildil võivad jooned olla visuaalselt aimatavad, aga meil on tegemist mitte visuaalsete, vaid matemaatiliste meetoditega. Matemaatilises mõttes on joon või selle alge olemas siis ja seal, kui ja kus saab üheselt öelda et selles punktis on joon. Tähendab, pildi iga piksli kohta tuleb öelda, kuulub see piksel mõnele joonele või mitte. Esialgu saadud pildi kõrvale tuleb luua selle binaarne analoog, mille iga piksli sisuks on üks kahest: joon või mitte-joon.

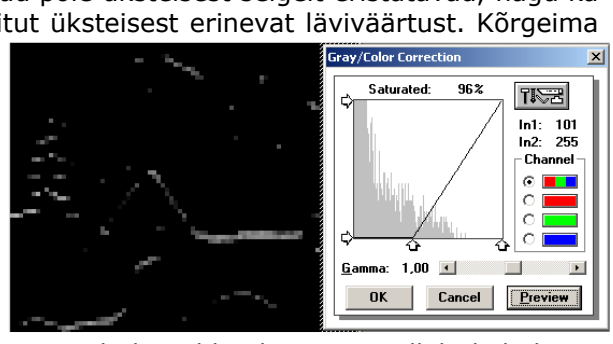
Kuna kasutatud meetodid (maskid) eristasid jooni sel viisil et värvisid joonte oletatavad asukohad teise värviga (tegid heledamaks), on binaarse pildi tekitamiseks loomulik kasutada saadud pildile **läviväärtuste rakendamist**. **Kui piksli väärtus ületab läve, loetakse piksel kuuluvaks joonele**, muidu mitte. Põhiliseks probleemiks on, millised peaksid sobivad läviväärtused olema. Läviväärtuste määramise üheks meetodiks on kasutada pildi histogrammi.



Kuna pildil olevad erinevad nähtused peaksid olema (ka) värvuste (heleduste) poolest eristatavad, ja nähtus "joon" on üks võimalikest nähtustest, peaks selle nähtuse kohal olema histogrammis märgatav tipp. Paraku, see on nii ainult ideaaljuhul, kui 1) jooni on pildis piisavalt palju, et nad histogrammi kujule märgatavat mõju avaldaksid ja 2) joonte detekteerimise meetod (mask) oli piisavalt selektiivne, tekitamaks joonte kohal muust pildist selgelt erinevaid värvusi. Kui need tingimused on täidetud, saab pildi histogramm olema selgelt mitmemodaalne ehk mitme üksteisest selgelt lahkneva tipuga. Iga tipu all saab olema üks pildil olevatest nähtustest, näiteks ühe tipu all "taust" ja teise tipu all "jooned". Sel juhul on loogiline asetada läviväärtus histogrammi tippude vahele.

Sagedasti esineb aga juhtum, kus histogrammi tipud pole üksteisest selgelt eristatavad, nagu ka juuresoleval näitel. Sellel juhul võib rakendada mitut üksteisest erinevat läviväärtust. Kõrgeima



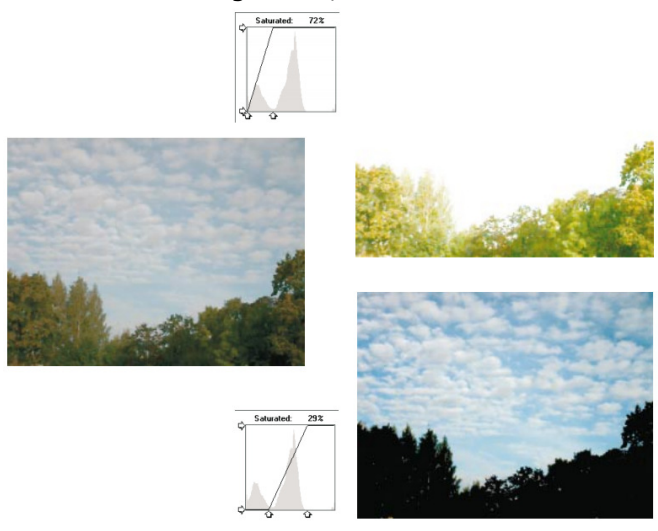


läve puhul eeldatakse et sellel kohal on kindlasti joon. Madalamate lävede kohtadel on joon "võib-olla". Mitme läve kasutamisel saab järgmisel sammul võtta aluseks kõrgeima läve kaudu saadud tulemused ja neist lähtudes hakata jooni nö kasvatama. Joonte "kasvatamisel" loetakse madalamate lävede kaudu saadud joone punktid tõesteks ainult juhul, kui nende naabruses on mõni kõrgema läve kaudu saadud joone punkt. Üks praktiline meetod, mis kasutab nii eespool toodud maske, läviväärtuste määramist kui ka joonte "kasvatamist", on nn **Canny Edge Detector.**

Aare.Luts.1@eesti.ee

8. Läviväärtuste määramine lokaalselt.

Eelmises punktis käsitleti läviväärtuste määramist histogrammi abil. Selline lähenemine põhineb eeldusel et mingid sarnased nähtused (nt jooned) avalduvad pildil mingis mõttes teistmoodi kui teised (omavahel sarnased) nähtused. Kui kasutatakse histogrammi, eeldatakse et "sarnasus" avaldub sarnaste värvuste, või sarnaste heledustena. Üldiselt saab (selliste) sarnasuste alusel otsida mitte ainult jooni, vaid mistahes sarnastena käsitletavaid objekte. Vaatame näiteks kõrval toodud pilti, mille vasakpooses osas on toodud kujutis metsast ja taevast. Selle pildi histogramm on selgelt kahemodaalne, millest võiks järeldada et pildil esineb (vähemalt) kaks teineteisest eristuvat nähtust. Tõepoolest, kui me justkui lõikame pildi histogrammi moodide vahelise lohu kohalt kaheks, tehes seda kahel erineval viisil, saame kaks erinevat pilti, milledest esimene sisaldab nüüd ainult nähtust "mets" ja teine sisaldab ainult nähtust "taevas". Nii saadud pildid on selles mõttes binaarsed et "nähtused" on taustast selgesti ja üheselt eristatud: esimesel juhul saab alati kasutada kriteeriumi "valge=taust; muu värvus=objekt" ja teisel juhul saab nüüd alati kasutada kriteeriumi "must=taust; muu värvus=objekt". Kui me nüüd tahame kasutada mingeid tehteid, mis peaksid rakenduma ainult objektile või ainult taustale, saab sellise kriteeriumi panna tehte esimeseks tingimuseks. Kõige lihtsamal juhul saab nüüd kokku lugeda, millise osa pildist moodustab objekt ja millise osa taust.



Võib olla abiks, kui mitte vaadata kogu pilti tervikuna, vaid pildi erinevaid osasid eraldi. Võib täiesti olla et pildi erinevate osade histogrammid on erinevad, ja et üksikute osade histogrammid ei ole samad mis on pildi kui terviku histogramm. Sellisel juhul võib olla kasulik kui mitte kasutada pildi kui terviku histogrammi, vaid kasutada selle asemel valitud akende alla jäävaid histogramme. Vaatame näiteks kõrvalolevat pilti. Paistab sedaviisi et üldiselt mürasel taustal on mõned justkui homogeensed piirkonnad. Võib püstitada küsimuse et kas need piirkonnad ongi ka tegelikult homogeensed või sisaldavad nad mingeid selgesti eristatavaid alampiirkondi. Et sellele küsimusele vastata, tuleks uurida piirkondi teistmoodi kui uuritakse tervet pilti. Üheks võimaluseks on luua piirkondade kohale aknad ja vaadata akende aluseid eraldi, nt muutes akende alla jäävate piltide histogramme. Kui nii teha, selgub et mõned aknad sisaldavad tõepoolest ka eristuvaid alampiirkondi. Põhimõtteline tehe on tuttav: libisev mask, ainult et nüüd tuleb libiseva maskiga siduda mitu üksteisele järgnevat tehet. Kõigepealt tuleb valida sobiva suurusega mask. Tegelikult saab seda teha automatiseeritult iteratiivselt, näiteks alustades väiksemast maskist, ja seda suurendada maski seni, kuni maski alla jääva ala histogramm hakkab muutuma. Kui histogramm hakkab muutuma, satub aknasse juba ka homogeense piirkonna väliseid piksleid, järelikult on sel juhul jõutud homogeense piirkonna piirile. Sellel piiril on paras valida sobiv tehe, antud juhul on kõigi homogeensete alade histogrammid kitsamad kui pildi kui terviku histogramm, seega sobib antud juhul kui kõigi homogeensete alade histogrammid lihtsalt laiaks venitada.

Eri osad eraldi võetuna on tõenäosemalt sarnase sisuga (sarnane sisu peaks asuma pildil lähestikku), mistõttu peaksid ka pildi väiksemates osades olevad jooned muu sisu taustal paremini välja paistma. Ka sellel juhul sobib alustada väiksemast maskist (aknast), liikuda automaatselt suuremale ja iga maski korral arvutada, kui hea on selle suuruse juures jooni määrata. Kui eeldada et rohkem on parem, tuleb valida see suurus, kus jooni oli kõige rohkem (parem määrata). Maski igas asendis tuleb leida maskialuse ala histogramm ja selle moodid. Kui moode ei saa määrata, siis jooni kas pole või on mask ebasobiv. Liikuda uude asendisse. Kui moodid on leitavad, tuleb leida sobiv läviväärtus, tavaliselt moodide vahel. Seejärel lugeda kokku, kui palju joone punkte sel juhul tuleb. Kuna mask liigub, saab ühtlasi leida, satuvad uues asendis määratud jooned punktid eelnevalt määratutega samasse kohta. Kui satuvad, on hästi. Kui ei satu, on asi kahtlane ja määratud punktid võib lugeda kahtlasteks. "Kahtlaste" punktide hulk oleks ka üheks kriteeriumiks, millega maski sobivat suurust valida. Kõrval üks näide, kus kogu pilti hõlmavat läviväärtust on halb leida, küll saab aga leida lokaalsed läviväärtused.

Kui joontele kuuluvad punktid on määratud, saab kogu kasutatud ahela headust hinnata. Seda tulekski teha, muidu pole objektiivset kriteeriumi meetodite valikuks ja parandamiseks. Sobivad näiteks sellised kriteeriumid: **1)** õigesti määratud joonepunktide arv; **2)** liigselt määratud jiinepunktide arv; **3)** puuduvate joonepunktide arv. Joonte detekteerimise ahela arvuliseks hindamiseks sobiks näiteks tulem, mis on saadud viisil: **1) tulem** *miinus* **2) tulem** *miinus* **3) tulem**.

Küsimused: 1) Millised elementaarmaskid on aluseks Sobeli operaatoritele, millised LoG operaatorile?

2) Sobeli operaatorites on kas read või veerud (-1,-2,-1) ja (1,2,1). Ometigi oli just leitud teise tuletisega sobiv mask (1,-2,1). Kas on tegemist mingi veaga? Kui ei, siis miks?

3) Milliseid jooni detekteerib Sobeli operaator $\mathbf{s}_x$, milliseid $\mathbf{s}_y$?

Ülesanded: 1) Valida meeldiv pilt (võib olla näiteks see), proovida läbi joonte detekteerimise erinevad tehnikad (maskid). Lõpuks tuleb saada (nt läviväärtus(t)e abil saadud) binaarne pilt, kus üks värvus (nt valge) tähistab jooni ja teine värvus (nt must) tähistab ülejäänud pilti. Hinnata erinevate meetodite efektiivsust.

Loe lisaks:
1) Ramesh Jain et al., Machine vision, McGraw-Hill Inc., 1995; 4. ja 5. ptk.
2) Rafael C. Gonzalez et al., Digital Image Processing, Prentice Hall, mitu väljaannet; 2002 väljaandes p. 4.3, 4.4, 4.5, 10.1 (smoothing, sharpening and filtering; point, line and edge detection).
3) Davies, E.R., Machine vision: theory, algorithms, praticalities. ptk.4 (läviväärtuste valiku matemaatilised algoritmid).

Konkreetseid teemasid puudutavaid kirjutisi võib leida, sisestades otsingumootorisse sobivad fraasid nagu "image thresholding techniques", "image edge detection" jne. Mõned sel viisil leitud tulemid on allpool. Kas nad käesolevast konspektist just paremad on, aga igatahes on nad teistsugused ja ehk ka siinkirjutatut täiendavad.
http://en.wikipedia.org/wiki/Canny_edge_detector
http://en.wikipedia.org/wiki/Thresholding_(image_processing)
konspekt ja artikkel läviväärtuste määramisest.

Aare.Luts.1@eesti.ee

Teema 6 küsimused ja ülesanded on järgmised:

1) Tekitada võimalus varieeritavate parameetritega müra lisamiseks piltidele. Mingi võimalus on olemas nii Java-paketis kui ka Photostyler-s, aga parameetrite paremaks varieerimiseks oleks kasulik tekitada täiuslikum võimalus. Igati sobilik on kursuse Java-pakett, müra tekitamiseks sobib juhuslike arvude generaatori väljakutsumine, nt kujul *y=(float)Math.random();* (paneb reaalarvulise muutuja *y* väärtuseks juhusliku arvu nullist üheni).

2) Võtta aluseks mingi vabalt valitud pilt, lisada sellele ühtlase jaotusega müra vähemalt 3 tuntavalt erineva amplituudiga. Proovida müra kõrvaldamise filtreid. Müra millise taseme juures on pildis oleva info enamvähem taastamine veel võimalik?

3) Proovida selle pildi müra mahasurumist sümmeetriliste maskidega (1,1,1) (3x3 mask) ja (1,1,1,1,1) (5x5 mask). Millest on tingitud erinevused? Proovida müra mahasurumist Gaussi maski lähendusega (Photostyler sisaldab vastavat otsetehet). Mis muutus ja miks?

4) Proovida selle pildi müra mahasurumist järsu sageduslõikega (järsku sageduslõiget saab teha Java-paketis, kus on olemas nii DCT tehe kui sellele järgnev sageduslõige, koos võimalusega sageduslõigatud pilt pöörd-DCT abil jälle pildikujule tagasi tuua). Millest on tingitud ootamatud efektid?

5) Võtta aluseks eespool ülesandes 2 valitud pilt, lisada sellele impulssmüra vähemalt 3 erineva sagedusega. Proovida müra kõrvaldamist keskmistamise ja mediaaniga. Müra millise taseme juures on pildis oleva info enamvähem taastamine veel võimalik?

6) Võtta aluseks eespool ülesandes 2 valitud pilt, lisada sellele ühtlase jaotusega müra vähemalt 3 tuntavalt erineva amplituudiga. Nüüd tekitada sama amplituudiga vähemalt 8 erinevat pilti (müra amplituud kõigil 8 juhul sama, aga pildid ise tulevad erinevad, tingituna müra juhuslikust iseloomust). Nüüd proovida pildi taastamist sama müra amplituudiga saadud erinevate piltide liitmise teel. Müra millise taseme juures ja mitme pildi liitmisel on pildis oleva info enamvähem taastamine veel võimalik?

7) Olgu meil mingi mürast haaratud pilt. Kuidas me saaksime hinnata, millise jaotusega (ühtlase- , normaal-, impulss- jne) on pilti rikkunud müra, ja võibolla hinnata ka müra parameetreid (nt normaaljaotusega müra standardhälvet)? Millise iseloomuga piltidel on seda lihtsam teha?

8) Proovida vähemalt kahe eespool kasutatud pildi müra kõrvaldamist Wiener filter lihtsustatud versiooniga (sellisega, mis ei nõua sagedusruumi kasutamist). Kas on erinevusi, kui võrrelda müra kõrvaldamise tulemustega eespool käsitletud meetoditel?

9) Millised elementaarmaskid on aluseks Sobeli operaatoritele, millised LoG operaatorile?

10) Milliseid jooni detekteerib Sobeli operaator $\mathbf{s}_x$, milliseid $\mathbf{s}_y$?

11) Sobeli operaatorites on kas read või veerud (-1,-2,-1) ja (1,2,1). Ometigi oli just leitud teise tuletisega sobiv mask (1,-2,1). Kas on tegemist mingi veaga? Kui ei, siis miks?

12) Uurida (sobivalt valitud akna abil), millised sellel pildil näha olevad homogeensed piikonnad sisaldavad ka selgesti eristuvaid alampiikondi, millised mitte. Vihje: tuleks muuta aknaaluse ala histogrammi.

13) Leida piirjoon nendelt piltidelt (pilt 1, pilt 2).

14) Mida võiks teha et pildil olevaid piirjooni visuaalselt rõhutada, aga seda mitte pildil oleva ülejäänud info olulise kaotamise hinnaga? (ise tehtud) näidetega.

7.teema õppematerjalid.

Selles raamatus kirjeldatakse (piir)joonte koostamist, kirjeldamist ja võrdlemist. Objektide koostamise meetodid. 1. Algoritm pildivälja trasseerimiseks. 2. Objekti (piirjoone) kitsendamine. 3. Algoritm piirjoone jälgimiseks. 4. Algoritm piirjoonte koostamiseks graafi abil. 5. Hough meetod. 6. Parameetrid joonte iseloomustamiseks.

Õpikeskkond:	TÜ Moodle
Kursus:	Pildiinfo töötlus (LOFY.05.055)
Koosta raamat:	7.teema õppematerjalid.
Printed by:	Aare Luts
Kuupäev:	teisipäev, 22 mai 2012, 08:25

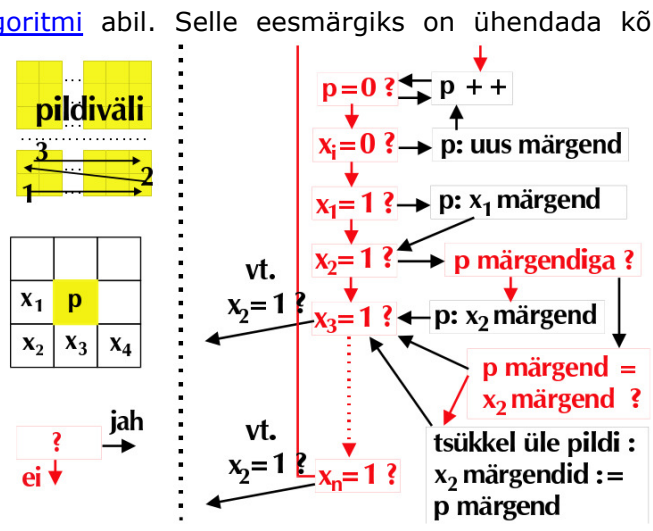
Sisukord

Objektide koostamise meetodid.

Olles pildil objektid (erijuhul:piirjooned) nö. eristanud, on tulemuseks uus pilt, millel on esialgsega võrreldes objektid mingil pildilisel moel rõhutatud. Visuaalses mõttes ongi objektide (piirjoonte) eristamine lõpule viidud, aga mitte matemaatilises mõttes. Selle pildi alusel tuleb nüüd teostada objektide matemaatiline koostamine, millise tehte tulemuseks peab olema mitte (niivõrd) pilt, vaid pildil olevate objektide matemaatiline esitus (tabel, joonte võrrandid vms.). Ainult teatud juhtudel ajab asja ära ka (pseudo)pilt, näiteks selline kus jooned on märgitud mingi kindla värviga, nii et seda värvi pildi ülejäänud osades ei esine. Üldiselt on aga nii et alles võrrandi või tabelitega esitatud objektide (erijuhul:joonte) olemasolul saab hakata objekte (erijuhul:piirjooni) matemaatiliselt uurima.

1. Algoritm pildivälja trasseerimiseks.

Põhimõte on esitatud joonisel toodud algoritmi abil. Selle eesmärgiks on ühendada kõik pikslid, mis vastavad tingimusele **p**<>0. See tingimus võib tähendada mida iganes, ühel lihtsamal juhul on meil tegemist piirjoonte rõhutamise ja läviväärtus(t)e rakendamise tulemusel saadud pildiga, kus piirjoonte pikslid on pildil märgitud valgega ja muud piklsid mustaga. Sellel konkreetsel juhul saab tingimust võtta otseselt: kui piksli väärtus pole 0, on tegemist piirjoonega ja kuulub edasisele vaatlusele, kui piksli väärtus on aga 0, ei paku see piksel huvi. Keerulisematel juhtudel võib tingimus "**p**" olla komplekssem, kui jällegi võtta mõni lihtsamatest juhtudest, võib tingimus "**p**<>0" tähendada et piksli värvus peab olema üks etteantutest või kuuluma värvuste mingisse piirkonda.



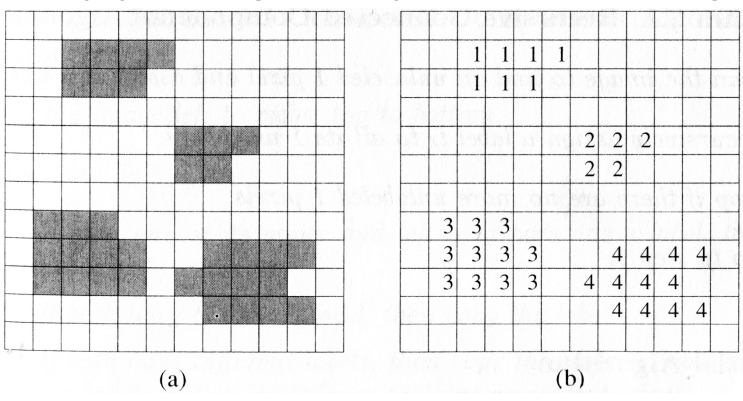
Tegelikult ei pruugi "**p**" tähendada ühtainust pikslit puudutavat tingimust, see võib olla ka vaadeldava piksli ümbrust puudutav tingimus. Ja nii edasi, järjest mitmekülgsemaks ja/või keerulisemaks. Igatahes on oluline et tingimus "**p**<>0" on kogu algoritmi võti, sellest tulenevad kõik edasised valikud. Kui tingimus pole rahuldatud, ei paku piksel huvi ja liigutakse järgmisele pikslile (joonisel: "p++"). Kui tingimus on rahuldatud, võetakse kasutusele mask (p,x_1,x_2,x_3,x_4). Põhimõtteliselt võib see mask olla ka teistsugune, aga esialgu on ta niisugune. Maski kuju tähendab seda et vaadeldavaks on piksel, mis asub maski kohal p, ja sellel kohal vaadeldakse p ümbruse neid piksleid, mis asuvad kohtadel x_1,x_2,x_3,x_4. Neid piksleid, kus maskis on tühikud, ei vaadelda.

Tingimus "x_i=0?" küsib, kas kõik maskiga määratud vaadeldavad pikslid on nullid, ehk, meie tingimuse järgi, esialgu märkimata. Kui kõik pikslid on märkimata, järeldub eespool olnud tingimusest "**p**<>0" et selles punktis algab uus objekt (kuna ümbrus oli seni märkimata, on objekt just uus, mitte mõne olemasoleva jätk). Olukorrast järeldub loogiliselt et see piksel tuleb märgendada uue märgendiga ja liikuda järgmisele pikslile. Kui "x_i<>0" ehk mõni maskiga määratud pikslitest on juba märgendatud, on tegemist olukorraga kus antud punktis jätkub eelnevalt juba alustatud objekt. Nüüd vaadatakse ümbruse piksleid ükshaaval. Kui maski x_1 all asuv piksel on juba märgendatud, on loogiline omistada pikslile p ehk x_1 naaberpikslile sama märgend millega oli märgendatud piksel x_1. Piksel p on sama objekti jätk, milline asus ka pikslil x_1. Kui maski x_1 all olev piksel ei olnud märgendatud, vaadatakse järgmist naaberpikslit, ja kordub sama protsess mis oli piksli x_1 puhul.

Siiski, eeltoodud tsükliga ei saa piirduda. Käsitleda tuleb ka võimalikku olukorda kus piksli p ümbruses oli mitu eelnevalt erinevalt märgendatud objekti. Kui nii on, on piksel p nende erinevate objektide ühenduspunkt. Kuna aga sel juhul saab erinevatest objektidest üks ühendobjekt, peavad selle ühendobjekti kõik pikslid olema märgendatud sama märgendiga. Kui eelnevalt olid märgendid erinevad, tuleb nad nüüd samaks teha. Erinevate objektide võimalikku kohtumist testitakse küsimustega "p märgendiga?" ja "p märgend = x_2 märgend?". Kui peaks selguma et "p märgend <> x_2 märgend", asutaksegi märgendite ühtlustamisele. Seda tuleb teha üle kogu pildi, kuna pole teada, kus märgendatud pikslid täpselt asuvad. Seetõttu vaadatakse sel juhul läbi pildi kõik pikslid ja kõik pikslid, mis olid eelnevalt märgendatud x_2 viisil, kirjutatakse üle p märgendiga.

Maski elementide x_1 ja x_2 alla jäävate pikslitega tehtut tuleb korrata ka teiste maski elementide puhul (üldjuhul kuni x_n, selle maski puhul kuni x_4). Vasakule suunatud viited "vt x_2 =1?" tähendavadki analoogseid tsükleid sellega mis väljus küsimusest "x_2=1?". Viidetega on piirdutud selle tõttu et kõigi tsüklite üksikasjalikud plokkskeemid ei mahu joonisele. Aga kuna kõik tsüklid on analoogilised, pole nende üksikasjaliku lahtikirjutamise järele ka vajadust.

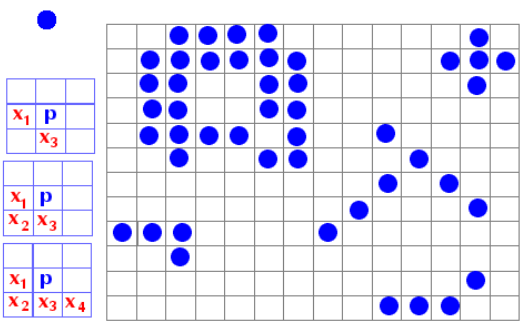
Selgitada tuleb ka tehet "p++". Põhimõtteliselt on selle sisuks "liigu järgmisele pikslile", kuid mitte suvalisele pikslile ja mitte suvalises järjekorras. Et tingimused töötaksid nagu ette nähtud, tuleb tagada et pikslid, millede puhul uuritakse kas nad pole ehk juba märgendatud, tõepoolest saaksid olla juba märgendatud. Järelikult, need pikslid, millega võrreldakse, tuleb läbida enne kui hakatakse vaatlema pikslit p. Sellise maski puhul on pikslite läbimise õige järjekorra tingimus täidetud kui järgitakse järjestust nagu toodud joonise ülal vasakul nurgas: esmalt läbitakse pildi alumine rida vasakult paremale (1), seejärel liigutakse altpoolt järgmise rea vasakusse otsa (2) ja korratakse rea läbimist (3). Nii tehakse pildi kõigi ridadega.



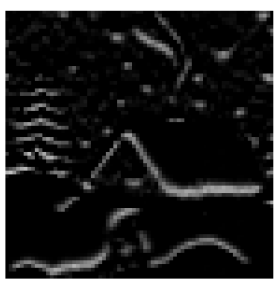
Algoritmi tulemina saame midagi sellist nagu toodud kõrvaloleval joonisel: pildil olevad seostatud objektid on kogu nende ulatuses märgitud sama märgendiga (siin: numbriga) ja eraldiseisvad objektid on märgitud erinevate märgenditega.

Rohkem algoritme ja kommentaare saab Internetist, nt otsingufraasiga "region growing (image segmentation)".

Küsimus: Nagu mainitud, võivad maskid olla mitmesugused. Paraku, erinevate maskide hinnaks võib olla asjaolu et teatava kujuga objekte ei ühendata. Joonisel on toodud mõned objektid (märgitud siniste punktidega piksleid kujutavas ruudustikus) ja mõned maskid. Millised objektidest saab ühendada milliste maskidega, ja milliste objektide jaoks millised maskid ei sobi?

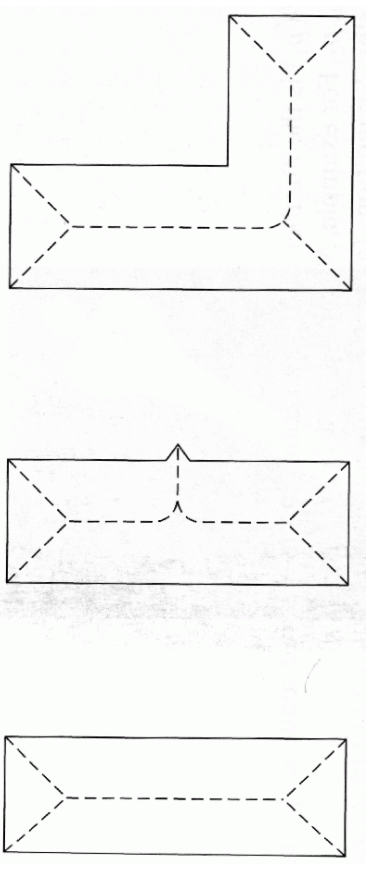


2. Objekti (piirjoone) kitsendamine.



Kui me räägime pildil olevatest objektidest siis üldiselt võib objekt olla mistahes kujuga. Teistsugune on lugu piirjoontega, sel juhul eeldatakse et selle objekti üks mõõde (pikkus) on teistest mõõtmetest oluliselt suurem. Teisisõnu, kõige parem joon on ühe piksli laiune joon. Paraku, piirjoonte rõhutamise teel saadud uutel piltidel on piirjoontena käsitletavad objektid tihtipeale üsnagi hajusad, nt meie poolt eelnevalt saadud ühel pildil.

Tihtipeale püütakse jooned "õhendada" juba enne joonte matemaatilisele kujule viimise algust. Eespool sai mainitud Canny Edge detector-it, mis muu hulgas sisaldab joone "õhendamise" operatsiooni. Jooni võib õhendada ehk arvutada selle "skeleti" ka pärastpoole, olles näiteks eelnevalt koostanud (joonte) objektid algoritmiga, mida on kirjeldatud eelnenud punktis.



Olgu meil drastiliselt jämedad jooned, nagu kujutatud kõrvaloleval joonisel. Nende joonte skeletid (ingl. "skeletons") on joonisel toodud punktiirjoontega. Põhimõtteliselt on skelettide üheks eemärgiks saada jooned (objektid) kujule, mida saab matemaatiliselt lihtsamini kirjeldada. Ja mitte ainult et lihtsamini, vaid ka unikaalsemalt ehk paremini eristatavalt. On loomulik et kui objekt sisaldab peale "kondikava" ka "karvu ja kasukat" ehk kuju kõikvõimalikke (juhuslikke) fluktuatsioone, saab tema matemaatiline kirjeldus sõltuma suurel määral fluktuatsioonidest ehk "kasukast", tihtipeale isegi rohkem kui objekti kondikavast. Kondikavasid peaks olema lihtsam võrrelda. Paraku, see oleneb küll nii objekti konkreetsest kujust kui ka meetodist, aga (välis)kuju väikeste erisuste tugev võimendumine pole välistatud ka siin. Joonisel on selliseks võimendunud näiteks kaks alumist kujundit. Tõsi, üks neist on ideaalne ristkülik ja teisel on selgesti lisandunud viies tipp, aga jääb küsimuseks, on see tipp nüüd nii oluline et sellest totaalselt erinevat skeletti tekitada. Kui nüüd võrrelda esialgseid kujusid ja skelette, on kahe lumise kuju esialgne erinevus ehk väiksem kui tuleb skelettide erinevus. Kui nii on hästi, siis on hea. Kui aga ei peaks nii olema, siis tuleb sellise võimalusega arvestada. Igatahes pakuvad skeletid (lisa)võimaluse objektide kirjeldamiseks, kas nad igal erijuhul sobivad, on iseasi.

Järgnevalt toome algoritmi, mis selliseid skelette tekitab. Nagu suure osa sellesarnaste algoritmide puhul, on ka nüüd eelduseks et pilt on juba selles mõttes binaarne et joonte pikslid on märgitud muudest pikslitest erineva tähisega, kõige lihtsamal juhul olgu joonte pikslid märgitud "1" ja kõik muud pikslid "0". Algoritm on iteratiivne, iteratsiooni igal sammul rakendatakse järjestikku kaht järku. Iteratsioon kordub, kuni mõnel sammul enam ühtki pikslit ei muudeta. Igal sammul muudetakse pikslit, mis jääb maskis oleva "p" alla. Täpsemalt seda kohe ei muudeta vaid kõigepealt märgitakse ära et on vaja muuta, muudetakse alles siis kui terve järk on läbitud, ja siis kõik vastavalt märgitud pikslid korraga. Kasutatakse maski, mis on toodud kõrval ja vaadeldakse maski kordajate alla jäävaid piksleid. Iga sammu esimeses järgus kontrollitakse tingimusi

a) $2 <= \mathbf{N} <= 6$?
b) $\mathbf{T} = 1$?
c) $p_2 * p_4 * p_6 = 0$?
d) $p_4 * p_6 * p_8 = 0$?

x_9	x_2	x_3
x_8	p	x_4
x_7	x_6	x_5

Siin **N** on määratud $\mathbf{N} = p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 + p_9$, kus pi on maski xi alla jääva piksli väärtus (meenutame, sellel korral oli nii et joone pikslid olid "1" ja muud pikslid olid "0"). Järelikult, N on piksli p nullist erinevate naabrite arv. **T** on määratud kui selliste muutuste arv, kus väärtusest "0" saab väärtus "1", seda järjekorras $p_2, \ldots, p_9$. Tingimused c) ja d) ongi korrutised, mis saavad nulliks alati kui vähemalt üks tingimuses nimetatud pikslitest ei ole joonepiksel. Kui kõik tingimused a), b), c) ja d) on piksli p puhul rahuldatud, märgitakse piksel p kui kustutamist vajav ehk selline, mille väärtus muudetakse järgu lõppemise järel nulliks. Kui vähemalt üks tingimustest ei ole täidetud, jääb piksel p selliseks nagu ta on. Kui tingimused a), b), c) ja d) on kontrollitud pildi kõigi pikslite jaoks, ja kustutamist vajavad pikslid on märgitud, on üks järk lõppenud. Seejärel kustutatakse kõik sedaviisi märgitud pikslid (siin: pannakse võrdseks nulliga). Selle järgneb teine järk, mis on esimesega väga sarnane, selle erinevusega et tingimused c) ja d) on nüüd c) $p_2 * p_4 * p_8 = 0$? d) $p_2 * p_6 * p_8 = 0$? Ka teise järgu lõppemisel kustutatakse kõik sedaviisi märgitud pikslid ja sellega loetakse iteratsiooniprotsessi üks samm lõppenuks. Kui sellel sammul ühtki pikslit ei kustutatud, on kogu algoritm lõppenud, vastasel korral minnakse jälle tagasi esimese järgu juurde.

Sarnaseid algoritme on mõistagi mitmeid, nende ja kommentaaride leidmiseks sobib kõige paremini otsing Internetis (otsingusõnad "thinning (algorithm, morphology)", "image skeletonization", "Medial Axis Transform").

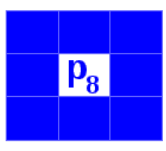
Küsimus: mida kontrollivad tingimused a) ja b)? Selgitada.

Aare.Luts.1@eesti.ee

3. Algoritm piirjoone jälgimiseks.

Kui piirjoon on ühe piksli laiune, ei teki probleeme piirjoone koostamisega selle jälgimise kaudu. Eelnevalt oli objekt (piirjoon) ehk küll juba koostatud, selles mõttes et samad ühendatud objektid olid märgitud sama tähisega ja et lahusolevad objektid olid märgitud erinevate tähistega, aga sellel juhul ei saanud rakendada tunnuseid nagu joone suund, mis on eriti joonte kirjeldamisel ja edaspidisel võrdemisel oluliseks tunnuseks.

Kõigepealt tuleb rääkida piksli naabruse definitsioonidest. Nimelt, koostamisel saadava joone mitmed parameetrid sõltuvad sellest, millist naabrust (naabrushulka) kasutatakse. Piltlikult võib asja vaadata sel viisil et lähtume maskidest, nt kahest kõrvaltoodud maskist $\mathbf{p}_4$ ja $\mathbf{p}_8$. Kui vaadeldakse punktist p (asub maski keskel) lähtuvaid võimalikke ühendusi, saab luua ühendusi ainult nende naaberpikslitega, mis on maskil värvitud sinisteks. Esimesel juhul ($\mathbf{p}_4$) on selliseid piksleid 4, teisel juhul 8. Esimest kogumit nimetatakse kirjanduses tihti $\mathbf{N}_4$(p) ja teist $\mathbf{N}_8$(p), mõlemate puhul kasutatakse hulgateoreetilisi termineid nagu näiteks "kuulub hulka", "hulkade ühendus", "hulkade ühisosa". Samuti saab hulgateoorias kasutada sisult loogilisi tehteid nagu "ja" või "või", mille tulemiks ei ole mitte tavaliste loogikatehete vastused "jah" või "ei", vaid hoopis mingi teistsugune hulk. Võimalikke ühendusi (naaber)pikslitega kirjeldatakse inglise keeles "connectivity".



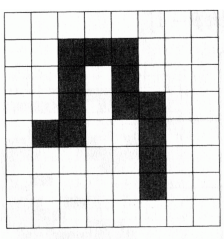
(a) 4-*connectivity*. Two pixels p and q with values from V are 4-connected if q is in the set $N_4(p)$.

(b) 8-*connectivity*. Two pixels p and q with values from V are 8-connected if q is in the set $N_8(p)$.

(c) m-*connectivity* (mixed connectivity). Two pixels p and q with values from V are m-connected if

(i) q is in $N_4(p)$, *or*

(ii) q is in $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ is empty. (This is the set of pixels that are 4-neighbors of both p and q and whose values are from V.)

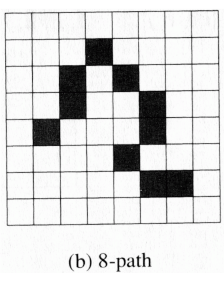


(a) 4-path

Siin ülal on toodud ka üks asjakohane näide vastava sisuga tekstidest. Kui hulk, millesse võimalikud ühendused kuuluvad, on ette antud, hakkavad sellest sõltuma näiteks pikslite kaugused üksteisest. Kõrval on toodud kaks näidet, vasakpoosled näitavad, millised oleksid jooned juhtudel, kui trasseerimiseks kasutatakse naabrust $\mathbf{N}_4$ ja $\mathbf{N}_8$, parempoolsed näitavad arvuliselt,

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

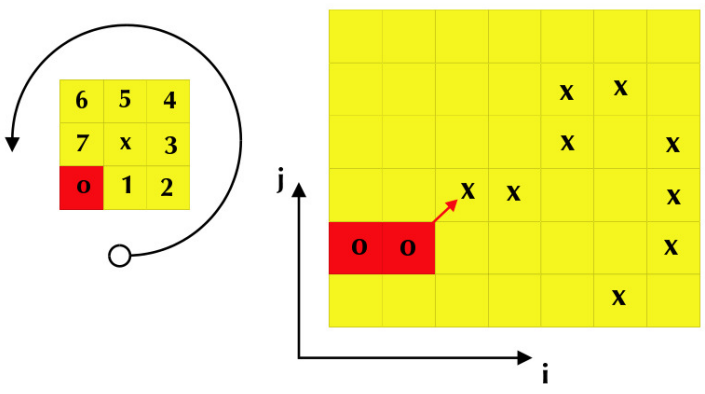
millised on kaugused keskel asuvast pikslist (selle piksli jaoks on kaugused 0). Kui kasutatakse seoste hulka $\mathbf{N}_4$(p), on tulemuseks ülemine näide, kui aga hulka $\mathbf{N}_8$(p), on kaugused kirjas alumises näites.



(b) 8-path

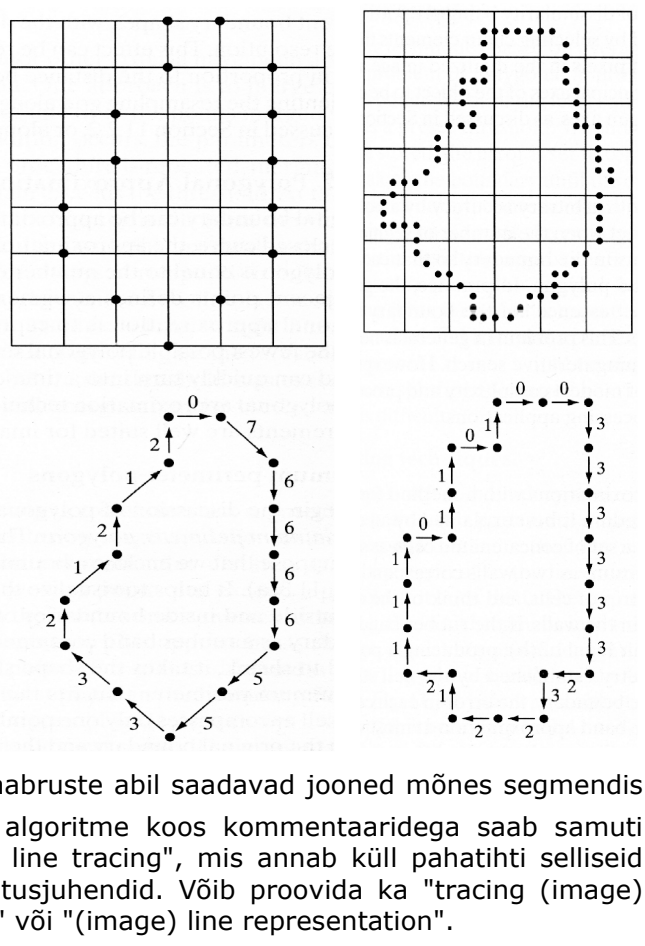
2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Piirjoone jälgimise algoritmis on lubatud naabrushulk kirjas etteantud maskis, siin toodud näites on naabruseks $\mathbf{N}_8$(p). Joone alustamiseks tuleb võtta mingi joonele kuuluv punkt, mis ei ole keeruline, kuna joontele kuuluvad punktid on nii ehk teisiti juba ära märgitud. Siis asetatakse mask keskpunktiga selle joonele kuuluva punkti peale. Järgnevalt otsitakse joone järgmist punkti, läbides selleks alati kindlaksmääratud järjekorra, mis on kirjas ka maskis. Järgmise punkti otsimist alustatakse alati maski vasakust alumisest nurgast (joone punkti suhtes asub see vasakul all). Kindlaksmääratud järjekord tagab et joonel liigutakse alati edasi. Joone järgmine punkt märgitakse sellel suunal, kus esimesena leiti joonele kuuluv punkt. Seejärel liigub ka mask keskpunktiga uue punkti kohale. Antud näites alustati joonise vasakust äärest. Nagu näha, oli järgmine punkt suunal 4. Ülejärgmine punkt saab olema suunal 5, ja nii edasi. On ka näha et tõepoolest läbitakse kõik joone punktid. Kui joon on kinnine, jõutakse lõpuks tagasi punkti, kust märkimist alustati ja sellega loetakse see joon koostatuks. Kui jätku ei leita, on võimalik kas lugeda joon lõpetatuks või siis näiteks otsida laiemast naabrusest, oletades et joon katkeb. Sel juhul tuleb küll olla kindel et joon tõepoolest just katkeb, mitte et algab uus joon, mis ei peagi olema eelnevaga ühenduses.



Joone trasseerimist tuleks alati alustada võimalikult pildi vasakust äärest, siis ei juhtu olukorda et joon lõpeb selle tõttu et minnakse vastu pildi äärt. Selline olukord juhtuks kui kõrvalolevas näites alustada mitte vasakust äärse vaid selle piksli parempoolsest naabrist. Sellisel juhul leitakse esimene uus piksel suunalt 5, aga lõuks jõutakse vastu pildi alumist äärt ja suunal 8 asuv piksel jääbki märkimata. Mittekinniste joonte puhul jääb siiski võimalus et osa punkte jääb märkimata. Et seda vältida, võib joone katkemise järel pöörduda tagasi sellesse punkti, kust katkenud trasserimist alustati, ja proovida uusi suundi, alustades nüüd eelmisel korral kasutatust järgmiselt suunalt. Kui ka siis joont ei leita, võib lugeda selle joone koostatuks, eeldusel ei tunta vajadust otsida katkevuspunkti laiemast naabrusest.

Joone sellise koostamise üheks eeliseks on asjaolu, et sel viisil saadaks mitte ainult järjestatud pikslid vaid ka iga piksli juures oleva joone suunad. Joone suunad (inglise keeles tihti "chain codes") on üks parameeter mille abil saab jooni kirjeldada ja hiljem võrrelda. Joone suund võib olla abiks ka katkenud joone jätkamisel. Kui joone otspunkti laiemas naabruses leitakse mingi uus joonele kuuluv punkt, ja on alust arvata et seni leitud joon ei peaks selles kohas lõppema, on mõistlik võrrelda lõppenud joone viimase segmendi suunda eemal algava uue joone esimese segmendi suunaga. Kui suunad on sarnased, on sama joone jätkumine usutavam. Sama võrdlust võib kasutada ka siis kui kaugemas naabruses on mitu erinevat potentsiaalest jätku. Ka sel juhul on tavaliselt põhjust eelistada seda jätku, mille suund on sarnane katkenud joone viimase lõigu suunaga. Kõrvaloleval joonisel on toodud sarnase joone kaks trasserimistulemust, üks kasutades naabrust $\mathbf{N}_8$ ja teine, kasutades naabrust $\mathbf{N}_4$. Nagu näha, on eri naabruste abil saadavad jooned mõnes segmendis vägagi erineva kujuga. Joonte trasserimise algoritme koos kommentaaridega saab samuti Internetist. Sobivaks fraasiks oleks "(image) line tracing", mis annab küll pahatihti selliseid tulemusi nagu tarkvara (nt Photoshop) kasutusjuhendid. Võib proovida ka "tracing (image) object boundaries" või "(image) line following" või "(image) line representation".

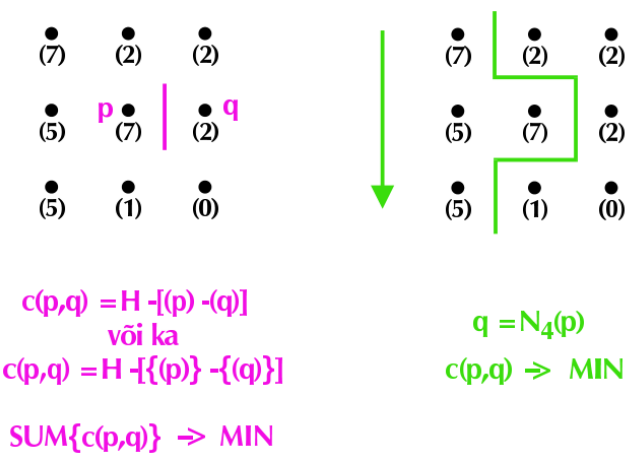


Ülesanne: Koostada pildil olevad jooned ja märkida ära sidusad piirkonnad (eristades objektid nt erinevate värvustega). Vähemalt mõne joone puhul anda tulemuseks suunakoodide tabel (Java paketis saab teha nt sel viisil et objektid hoida pildimassiivides, suunakoodid aga kirjutada reaalarvmassivi, viimase saab hiljem arvkujul ka faili kirjutada). Vastuses anda ka iga joone pikkus ja iga piirkonna punktide arv.

Aare.Luts.1@eesti.ee

4. Algoritm piirjoonte koostamiseks graafi abil.

Põhimõte on toodud joonisel. Selle algoritmi kasutamisel pole vajadust eelnevalt läbi teha pildi piirjoonte rõhutamist, millega tegelesime eelmises teemas ja mille võtmesõnadeks olid sellised nagu "Laplace filter". Samuti pole vaja tegelda läviväärtuste seadmisega viisil, millega tegeldi eelnevalt. Vähemalt põhimõtteliselt peaks meetod suutma tuvastada piirjooni otse pildilt, kasutades selleks suurima kontrastiga alade (teede) otsimist.



Antud algoritm (pealkirjaga "**Global processing via Graph-theoretic techniques**") koostab piirjoone tingimusest, et saadav joon optimiseeriks teatava üldise kriteeriumi. Joonisel toodud juhul on kriteeriumiks kaalude c(p,q) summa minimiseerimine, kus **H** on pildil leiduv max heledus, (p) ja (q) aga vastavalt pikslite p ja q heledused. Kasutatav kriteerium, nagu ka igal sammul vaadeldav naabrus võivad olla mitmesugused, mitte ainult seesugused nagu toodud näites. Nagu näha c(p,q) arvutamise valemist, on tema väärtus seda väiksem mida suurem on p ja q erinevus. Pildi mõttes tähendab suurem erinevus suuremat kontrastsust ehk värvuse järsemat muutmist. Põhimõte sobib igati: parameetrite kiired muutused vastavad tõepoolest (piir)joontele.

Kui rääkida teoreetiliselt, tuleks läbi arvutada joonte asukohtade kõikvõimalikud kombinatsioonid, et siis valida see kombinatsioon, mille puhul c(p,q) summa üle kõigi punktide oli minimaalne. Praktiliselt ei ole seda võimalik teha, kuna kombinatsioonide arv läheb üüratuks. Seetõttu kasutatakse praktikas samm-sammulist lähendust, mis annab igal sammul küll ka teoreetiliselt parima tulemuse, aga ei välista et kogu pilti korraga vaadates võiks leiduda mõni veel parem kombinatsioon. Samm-sammulisel lähendusel alustatakse pildi ühest äärest. Kui on teada, kust äärest jooned tavaliselt algavad (kas tõenäosemad on horisontaalsed või siis vertikaalsed jooned), on mõistagi arukas alustada sobivamast äärest.

Selles näites alustame pildi ülemisest servast. Et leida joone esimene lõik, võrreldakse kõiki erinevusi naaberpikslite väärtuste vahel. Siin on võrrelda paare (7,2) ja (2,2), ja tuleb otsustada, milliste vahelt võiks joon läbi minna. Paari (2,2) vahel pole joont põhjust oletada, nende vahel mingit väärtuste hüpet ei toimu. Küll võib oletada joont (objekti serva) paari (7,2) vahel, seega tõmbame joone esimese lõigu nende vahele. Oleme jõudnud neliku (7,2,5,7) keskpunkti ja nüüd on küsimus, kuhu edasi. Edasi minnakse jällegi selles suunas, mille kaks "seina" on kõige erinevamad, järelikult paremale paari (2,7) vahele. Punktis, kuhu jõutakse, tuleb jällegi otsida teed kõige sobivamaks edasiliikumiseks, selleks saab suund alla paari (7,2) vahelt. Ja samamoodi edasi, kuni jõutakse kas pildi ääreni või ilmneb mõni muu takistus (nt on lõpp-punkti ümbruse kõik pikslid võrdse väärtusega, mistõttu ei ole selles punktis enam minimaalset kaalu andvat suunda). kui pildil ongi ainult üks joon, on protsess sellega lõpetatud. Kui jooni võiks olla veel, tuleb analoogilist protsessi korrata, alustades nüüd mõnest muust punktist.

Kui alguseks on valitud ülemine rida, tuleb liikuda mööda rida paremale ja otsida järgmist paari, mille vahelt võiks alata joon. Selles näites sellist paari enam ei ole. Ainus allesjäänud paar on (2,2), mida läbivat joont pole põhjust oletada. Antud juhul vaadeldakse joone koostamisel ainult punkti p 4-naabreid. Kui vaadelda naabrust $\mathbf{N}_8(p)$, võib joon liikuda ka diagonaalsuundades. Selle näite korral saaksime kaks samaväärset algust: juba kasutatud suuna ülevalt alla ja sellele lisaks suuna paremale diagonaalselt alla pikslilt (2) pikslile (2), diagonaalpikslite (2,7) vahelt. Mõlemal juhul jõuaks joon lõpuks samasse punkti.

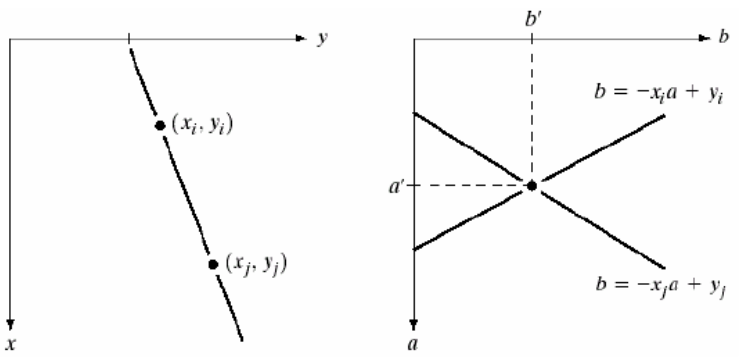
Siin jõudsime küll muu hulgas olukorda, mida olema kord juba kirjeldanud. Nimelt hakkasid meie jooned asuma pikslite vahel, kuhu on tihtipeale halb kirjutada, kuna maatriksis pole vastavat kohta. Aga ega keegi pole öelnud et võrrelda võib vaid naaberpiksleid. Võrrelda võib ka suuremaid piirkondi, nt paaride või kolmikute keskmisi. Mis olekski tihtipeale mõistlik kuna üksikute pikslite väärtused võivad olla juhuslikud hoopis suurema tõenäosusega kui mitme piksli keskväärtused. Selles näites me nii teha ei saa kuna nt esimeses reas saaksime me küll arvutada (7,2) ja (2,2) keskväärtused, aga poleks midagi, millega nende keskväärtusete erinevust võrrelda. Kui meil oleks esimeses reas veel kaks punkti, kokku rida (2,1,7,1,3), saaksime kõrvutiste paaride summadeks (3,8,8,4); keskväärtuse leidmiseks tuleks need lihtsalt kahega jagada. Näeme et suurim erinevus oleks paaride (2,1) ja (1,7) vahel, seega läheks joon sel juhul 2 ja 7 vahelt läbi punkti (1) alla. See pole küll seesama mis otse läbi punkti 7, aga selle lähedalt. Üldjuhul võib võimalikke jooni olla mõistagi palju, jooned võivad puutuda jne. Sellistel juhtudel leiavad kasutust eespool käsitletud meetodid nagu näiteks teineteisega puutuvate objektide ühendamine (märkimine sama tähisega).

Lisakirjanduse otsimiseks võib proovida fraasi "(image) line detection graph techniques".

Aare.Luts.1@eesti.ee

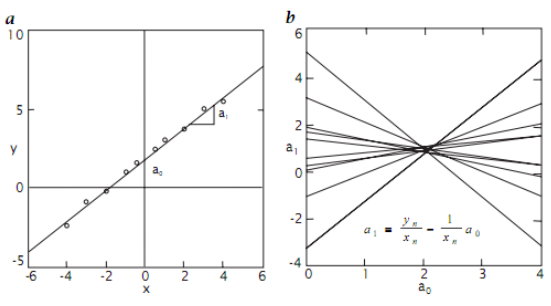
5. Hough meetod.

Eelmises punktis vaatasime meetodit, mis üldiselt ei nõua pildi eeltöötlust. Sellele sarnane on ka siin käsitletav, see on siiski selles mõttes erinev et mingit eeltöötlust ta siiski nõuab, nimelt peab saama iga punkti jaoks öelda, on tegemist joonele kuuluva punktiga või mitte. Kui see on teada siis joonte koostamise teeb juba meetod. Meetod on eelnimetatutest oluliselt erinev nt selles mõttes, et siin ei otsita/koostata mitte igasuguseid (piir)jooni, vaid otsitakse ainult etteantud parameetritega kirjeldatavaid jooni. Tähendab, sirgjoonte jaoks sobib oma meetod, ringide jaoks oma, ellipsite jaoks kolmas jne, kusjuures üldjuhul üks ei sobi teise jaoks.

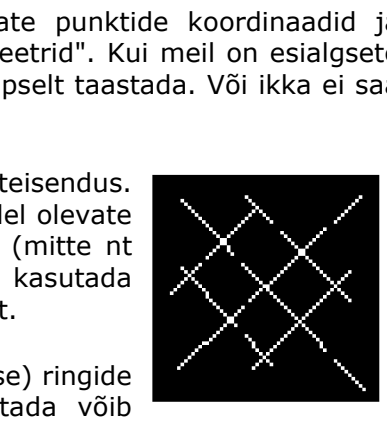


Otsitagu näiteks pildil leiduvaid sirgjooni, st jooni mis on kirjeldatavad võrrandiga y=a(i)·x+b (i), kus (x,y) on mingi joonele kuuluva punkti koordinaat, a(i) on konkreetse sirgjoone tõus ja b(i) on selle sirgjoone vabaliige. a(i) ja b(i) on selle konkreetse sirge jaoks fikseeritud, muutuvad x ja y. Kuna aga on mitte lihtsalt a ja b, vaid a(i) ja b(i), tähendab see et jooni võib olla erinevaid. Mõistagi võib olla ka palju punkte (x,y), milliseid kõiki võib käsitleda kuuluvana mingile joonele. Sellisel juhul minnakse üle ruumi (a,b), nii et b=y(j)-a·x(j). Nüüd on fikseeritud y(j) ja x(j), aga muutujateks on a ja b.

Vaatame mingit eelmistes koordinaatides (x,y) asuvat sirgjoont ja sellel olevaid kahte punkti. Kuna need kaks punkti asuvad samal sirgel, kehtivad nende jaoks samad sirge tõus a ja sirge vabaliige b. Loomulikult, kui me pärast vaatame ruumi (a,b), on selles ruumis oleva sirge x-koordinaadi analoogiks nüüd a ja y-koordinaadi analoogiks on b. Nüüd on sedamoodi et x(j) on käsitletav hoopistükkis sirge tõusuna ja y(j) on käsitletav sirge vabaliikmena. Siiski, kuna ennist asusid mõlemad (x,y) samal sirgel, tähendab eelöeldu et ruumis (a,b) peavad mõlema x(j) ja y (j) sirged lõikuma. Tähendab, kui me vaatame läbi esialgse joone kõik punktid, saame iga x(j) ja y(j) puhul küll erineva sirge ruumis (a,b), aga kõik need sirged lõikuvad ühes ja samas punktis. Kui me iga lõikumise kuidagi ära märgime, saame me sirgete lõikepunktis palju äramärkimisi, mis võib välja paista näiteks sel viisil et see lõikepunkt muutub ruumis (a,b) teistest punktidest heledamaks. Sedalaadi punktide "helestamist" kasutataksegi.

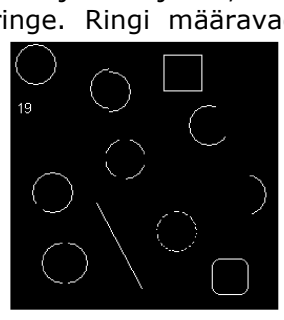


Tsükkel võib toimuda sel viisil et ülemisel tasemel vaadatakse läbi esialgse (x,y) ruumi kõik punktid, mis on märgitud kuuluvana mõnele joonele. Osa neist võib kuuluda siin näites vaadeldud joonele, osa mõnele teisele joonele, aga see pole siinkohas oluline. Kui mingi joonele kuuluv punkt x(j), y(j) on leitud, algab sisemine tsükkel sirge joonistamiseks ruumis (a,b). Selleks toimitakse nagu sirget ikka saab joonistada: võetakse tsükkel üle a kõikvõimalike väärtuste ja iga a väärtuse korral arvutatakse vastav b väärtus, kasutades selleks leitud punkti väärtusi x(j) ja y(j), mis on nüüd (a,b)-ruumis joonistatava sirge tõusuks ja vabaliikmeks. Ruumi (a,b) iga punkt, mida loodav sirge läbib, märgitakse ära. Kui punkt on mõne varasema sirge poolt juba ära märgitud, suurendatakse loendurit ehk pildi mõttes muutub see punkt (a,b) tasandil heledamaks. Kui esialgse (x,y)- ruumi kõik joontele kuuluvad punktid on läbi vaadatud, protsess lõpeb. Tulemuseks on (a,b)-ruum, kus olevaid punkte on ära märgitud nii mitmel korral kui neid punkte läbis ükskõik milline sirge. Sirgete lõikepunkte on aga ära märgitud sagedamini, seega on need arvtabelis suuremate väärtustega ja pildil heledamad. Lõpuks määratakse kõigi (a,b)-ruumi piisavalt heledate punktide koordinaadid ja saadakse selle kaudu teada, milliseid sirgeid pildil rohkem oli.



Mõnikord muretsetakse selle pärast et programm ei jookseks kinni juhtudel kui pildil olevate sirgete tõus on null (tekib jagamine nulliga). Et sellist olukorda vältida, minnakse (x,y)-ruumist üle polaarkoordinaatidesse (q,f). Täpsemaid valemeid vt kirjandusest.

Nagu eepool mainitud, saab Hough teisendust kasutada ka teistsuguste joonte jaoks, küll tuleb iga liigi jaoks koostada omaette algoritm. Vaatame näiteks ringe. Ringi määravad parameetrid on tema keskpunkti koordinaat ja raadius. Seega, kui joonte puhul oli meil kaks parameetrit (tõus ja vabaliige), on nüüd parameetreid kolm. Kui kõiki kolme varieerida, saame me kolmemõõtmelise ruumi. Ülesannet saab siiski taandada kahemõõtmeliseks, näiteks sel viisil et me fikseerime raadiuse ehk otsime korraga ainult ühe raadiusega ringe, lubades küll neil asuda pildil ükskõik kus. Ringi võrrand on teada, kui seda võtta (x,y,x_0,y_0,R)-kujul, saab ringi kõik punktid arvutada, andes ükshaaval ette kõik punktid x vahemikus (x_0-R,x_0+R) ja lahendades x iga väärtuse korral ruutvõrrandi y leidmiseks. Kui me läheme Hough ruumi ja anname R ette, siis võrrandi kuju isegi ei muutu, küll on nüüd otsitavaks y_0 ja selle leidmiseks tuleb anda ette kõik punktid x_0 vahemikus (x-R,x+R). Jällegi, esmalt võetakse vaatluse alla kõik punktid (x,y), mis on eelnevalt märgitud kuuluvana mingile joonele. Iga sellise punkti korral leitakse kõik sellel juhul võimalikud ringid ruumis (x_0,y_0). Tulemuseks on heledamad kohad ruumi (x_0,y_0) nendes punktides, millise keskpunkti ja raadiusega R ringe esialgsel pildil rohkem oli.



Hough teisenduse üheks määravaks eeliseks on tema vähene tundlikkus mürale ja ka objektide katkestustele (vt näiteid). Selles mõttes sarnaneb Hough teisendus väga inimese nägemismeelele, mis suudab samuti näha ka objekte ka juhul, kui nad on kas müra taustal või osaliselt teiste objektide poolt varjatud. Lisakirjandust saab otsingufraasi "Hough transform" abil.

Küsimused: 1. Hough teisenduse all oli öeldud "me saame (a,b)-ruumi sirgete lõikepunktis palju äramärkimisi". Oletagem et esialgses (x,y)-ruumis oli ainult üks sirglõik. Kui palju äramärkimisi me sel juhul täpsemalt saame?

2. Oli öeldud "määratakse kõigi (a,b)-ruumi piisavalt heledate punktide koordinaadid ja saadakse selle kaudu teada (x,y) ruumis olevate sirgete parameetrid". Kui meil on esialgsete sirgete parameetrid teada, peaksime me saama esialgse pildi täpselt taastada. Või ikka ei saa ja miks?

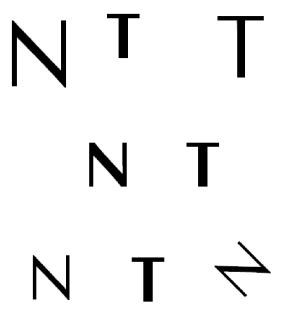
Ülesanded: 1. Java-paketis on realiseeritud lineaarne Hough' teisendus. Alljärgnevalt on toodud pilt mis sisaldab hulga jooni. Leida piltidel olevate joonte võrrandid, kasutades vabalt valitud pilditöötlusmeetodit (mitte nt väljatrükitud pildil olevate joonte mõõtmist joonlauaga), võib kasutada Hough' teisenduse tulemusi. Vihje: loe paketis pakutavat abiinfot.

2. Ehk koostaks Hough teisenduse etteantud diameetriga (vali ise) ringide jaoks ja prooviks tema (head) omadused ise järele? Kasutada võib eespool toodud näitepilte, nendel on raadiuseks 19.

Aare.Luts.1@eesti.ee

6. Parameetrid joonte iseloomustamiseks.

Sageli on ülesandeks võrrelda koostatud objekte (nt. piirjooni) mingite teiste objektidega. Seega tuleb koostatud objekte (piirjooni) iseloomustada mingite lihtsalt käsitletavate parameetritega. Eespool vaadeldud joone suunakoodide rida võib mõnel juhul olla hea küll, aga suurema pikkusega joone puhul koosneb selline rida (mitme)sajast liikmest, milliseid tuleks siis kõiki ükshaaval võrrelda. Oleks kiirem ja lihtsam kui saaks võrrelda mitte mõndasada vaid mõnda parameetrit. Tavaliseks probleemiks on ka asjaolu, et pildil olevad (sama liiki, samasuguse geomeetrilise kujuga) objektid võivad olla eri asendis, eri suurustega jne. Eri asendis joonte puhul tuleb nende suunakoodide tabelid kõigepealt samasse suunda pöörata, milleks tuleb see suund kõigepealt leida. Veel parem kui oleksid mingid pöördenurgast ja suurusest (piisavalt) sõltumatud parameetrid.



Olgu meil mingid piirjooned nagu need, mis on kõrvaloleval joonisel värvitud punasteks. Küsimuseks olgu, millised neist on sama kujuga. Lihtlabaselt võrrelda ei saa, põhjusel et jooned on erinevate suuruste ja erineva asendiga. Oleks hea, kui õnnestuks kõik need jooned mingi algoritmiga ka teoreetiliselt põhjendatult samasse suunda pöörata ja, veel parem, ka kõik samasse punkti nihutada. Kui jooned oleksid samauunalised ja samas puntis, saaks neid võrrelda lihtlabase üksteisest lahutamise abil. Soovitud omadustele lähedane on objekti omaväärtusteisendus (ingl. "principal component analysis", "(line) description by principal components"). Selle teisenduse üksikasjaliku mehhaanika saab leida sellele pühendatud kirjandusest, siinkohas summeerime lähtekohad ja tulemused. Teisendus on juba realiseeritud kursuse Java-paketis, nagu ka MathCad-s.

Teisenduse alustuseks tuleb leida objekti (piirjoont) iseloomustav kovariatsioonimaatriks. Kovariatsioon on ka muidu kasulik, seetõttu sellest veidi pikemalt. Formaalsel kujul on kovariatsioonimaatriksi $\mathbf{C}_u$ valem kirjas joonisel. Tähised: **E** on operaator, mis arvutab selle operaatori all olevate suuruste keskväärtuse; **u** on objekti (joone) punktide vektor, mu nende vektorite keskväärtus. Tähis T tähendab maatriksi transponeerimist, st püstvektorid muudetakse horisontaalvektoriteks ja vastupidi. Transponeerimine on oluline, kuna maatrikstehete tulemused sõltuvad maatriksi asendist. Antud juhul on objektiks joonisel sinisega märgitud joon. See on antud 6 punkti kaudu, seega on suur osa joonest tegelikult ettekujutus, rangelt võttes saame me rääkida ainult etteantud 6 punktist. Selguse huvides on toodud kõik need 6 punkti, esitatuna nende x- ja y-koordinaatide kaudu, $\mathbf{u}_1$ kuni $\mathbf{u}_6$. Nagu näha, on nad kõik kahekomponendilised vektorid. Järgnevalt on toodud vektorite keskväärtuse arvutuskäik ja saadud tulemus. Edasine arvutus läheb valemi järgi ja arvestades maatriksarvutuse reegleid. Selguse huvides on ka arvutuse üksikasjad lisas 1 ja lisas 2 ära toodud, tulemuseks on 2*2 kovariatsioonimaatriks $\mathbf{C}_u$, nagu antud joonisel.

$$C_u = E\left\{(u - m_u)(u - m_u)^T\right\}$$

$$u_1 = \begin{pmatrix}2\\2\end{pmatrix} \quad u_2 = \begin{pmatrix}2\\3\end{pmatrix} \quad u_3 = \begin{pmatrix}3\\4\end{pmatrix}$$

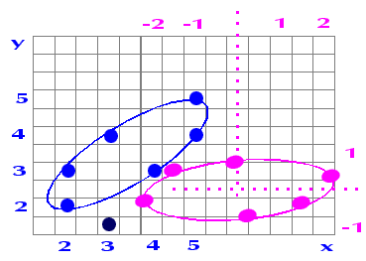
$$u_4 = \begin{pmatrix}5\\5\end{pmatrix} \quad u_5 = \begin{pmatrix}5\\4\end{pmatrix} \quad u_6 = \begin{pmatrix}4\\3\end{pmatrix}$$

$$m_u = 1/6\begin{pmatrix}2+2+3+5+5+4\\2+3+4+5+4+3\end{pmatrix} = \begin{pmatrix}3.5\\3.5\end{pmatrix}$$

$$C_u = 1/K\begin{pmatrix}9.5 & 5.5\\5.5 & 5.5\end{pmatrix}$$

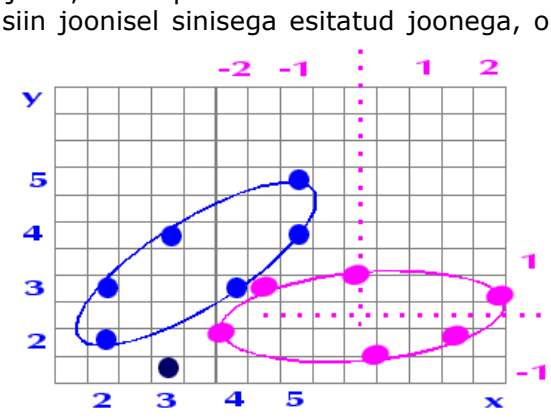
$$\{v\} = G\{u\},\ C_v = \text{diagonal}$$

$$z_i \rightarrow G,\ \left(C_u - \delta_{ij}\right)z_i = 0$$



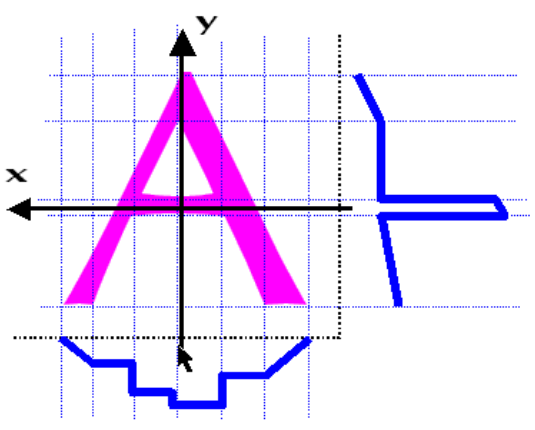
Mis on kovariatsioonimaatriksi sisu? Selle diagonaalelemendid näitavad vastava parameetri variatsiooni ehk muutlikkust. Esimeseks parameetriks on x ja teiseks y, seega on objekti (keskmine) muutlikkus x-suunal suurem kui tema muutlikkus y-suunal. Vaatlusel tundub ka sedaviisi et justkui oleks joon x-suunal rohkem välja venitatud. Mittediagonaalelemendid iseloomustavad x-ja y-suunaliste variatsioonide statistilist seost. Mida suuremad need elemendid on, seda tugevamini on x- ja y-suunaline muutlikkus seotud. Kui elemendid on positiivsed, on seos samasuunaline, kui negatiivsed, erisuunaline. Muutlikkuste seost võib illustreerida sel viisil et seos on positiivne juhul kui objekti punkti x suurenemisega kaasneb tavaliselt ka selle punkti y suurenemine. Sellel joonisel keskeltläbi nii ongi, võrdleme näiteks punkti (2,2) ülejäänud punktidega. Punktide (4,3), (5,4), (3,4), (5,5) kõigi korral on x suurem ja ka y on ka suurem, täpselt nii et "seos on positiivne". Punkti (2,3) korral on x sama, kuid y suurem. Seega läheb see punkt "seose" tavapäraselt rajalt välja, kuid ta ei räägi ka sellele vastu. Seosele räägiks vastu näiteks punkt, mis on märgitud joonisel mustaga, aga sellist punkti meil joonel tegelikult ei ole. Seega on joone kõigi punktide x- ja y-koordinaadid positiivse seosega, mis kovariatsioonimaatriksis väljendub selles et mitte-diagonaalelemendid on positiivsed. Et objekte pöörata ja/või nihutada ja/või skaleerida, tuleb muuta objekti punktide koordinaate. Punktide kordinaatide muutmiseks võib kirja panna seose $\mathbf{v}=\mathbf{G}\cdot\mathbf{u}$, kus **G** on muutmist teostav maatriks, **u** on punktide vanade koordinaatide vektor ja **v** on uute koordinaatide vektor.

Antud juhul otsitakse erilist teisendust, sellist kus teisendusejärgsete punktide asukohtade kovariatsioonimaatriks oleks diagonaalne (diagonaalkujul). Mida tähendab "diagonaalne"? Sellisel juhul on kovariatsioonimaatriksi mitte-diagonaalelemendid nullid ja x- ja x-koordinaatide vahel statistiline seos puudub. Üks jooni, mille punktide koodrinaatide vahel statililine seos puudub, on ring. Kui võrrelda ringi siin joonisel sinisega esitatud joonega, on nende erinevus näiteks selles et siin toodud joon on mingis suunas välja venitatud, ring pole aga üheski suunas välja venitatud. Selles mõttes on tulem ühtaegu unikaalne ja universaalne et selle teisendusega saame me mistahes kujuga joone viia kujule, mis on sarnane mistahes teise joone kujuga, kui me ka selle joonega oleme teinud sama teisenduse. Järelikult, pärast sellist teisendust peaks saama mistahes jooni omavahel võrrelda. Matemaatikast on teada et sobiva maatriksi **G** saab sel viisil et formeerida see esialgse kovariatsioonimaatriksi omavektoritest.



Omaväärtuste ja omavektorite leidmiseks tuleb lahendada omaväärtusülesanne ehk võrrandsüsteem, mis on toodud joonise alumises parempoolses nurgas. Seda süsteemi me üksikasjaliselt lahendama ei hakka, vt asjakohasest kirjandusest. Olulisim on et lõpuks saadakse omaväärtused, antud juhul 2.67 ja 0.33 ning on võimalik pöörata antud kujund teatavasse unikaalsesse suunda ja asukohta. Pööratud ja nihutatud kujund on antud joonisel lillaga, sealjuures tuleb tähele panna et ka koordinaadid on nüüd muutunud, nüüd asub nullpunkt seal kus märgitud lillade punktiirjoontega. Nüüd oleme leidnud esialgsete joone (objekti) omaväärtused ja liigutanud selle teatavasse unikaalsesse asendisse. Sama saame me teha teiste objektidega, misjärel saab sama kujuga objekte leida neid lihtviisil üksteisest maha lahutades. Kui seda eesmärki pole, saab anda (esialgse) hinnangu sarnasuse kohta, kasutades omaväärtusi. Kui omaväärtused on samad, on objektid sarnased. Kui omaväärtuste suhe on sama, on objektid umbes samal määral välja venitatud.

Seega on omaväärtused üks otsitavatest "lihtsatest" parameetritest, milliseid saab kasutada objektide võrdlemisel. Kui tahaks ikka keerulisemaid (aga mõnes mõttes täpsemaid) parameetreid, on ka nende leidmiseks nüüd uued võimalused. Kui objektid on pööratud teatavasse unikaalsesse asendisse, saab leida näiteks nende projektsioonide vektorid, mis saavad nüüd olema võrreldavad. Ühe objekti projektsioonid on antud kõrvaloleval joonisel. Uute parameetrite tuletamise võimalusi on palju. Üheks üldisemaks klassiks on kõikvõimalikud momendid, milliste üldvalem on antud joonisel allpool. Selles valemis olevad x ja y võivad olla koordinaadid, valemis on need võetud astmetesse p ja q. Lisaks on valemis teatav funktsioon f(x,y), mis on tihtipeale 1, aga võib seda ka mitte olla. Seda funktsiooni võib vaadelda asukohast sõltuva kaaluga. Olulised on ka tsentraalmomendid, mida ei arvutata mitte lihtsalt koordinaatide x ja y järgi, vaid erinevuste järgi keskväärtusest. (x miinus x keskväärtus) oleks sisuliselt kaugus pildi keskpunktist. Näitena on toodud momentide kasutamine mõnede sagedasti kasutatud parameetrite leidmiseks, aga nüüd mitte lihtsa valemi, vaid üldistatud valemi järgi. Näiteks on **E** [x] nüüd mitte lihtsalt pildi x-suunaline keskpunkt, vaid kaalufunktsiooniga modifitseeeritud keskpunkt ehk sisuliselt x-suunaline masskese.



$$m_{pq} = SUM_x SUM_y\left\{x^p y^q * f(x,y)\right\}$$

$$w_{pq} = SUM_x SUM_y\left\{(x - E[x])^p * (y - E[y])^q * f(x,y)\right\}$$

$$E[x] = m_{10}/m_{00}$$

Kuidas tuletatud parameetreid kasutatakse, selle kohta vt näiteid kirjandusest. Sobivad fraasid internetiotsinguks "principal component analysis", "Hotelling transform", "object descriptors". Mõned juba kohaletoodud konspektid (1, 2, 3, 4; ühegi kohta ei saa öelda et ta oleks ideaalne).

ÜLESANNE: 1. Lisatud piltidel (1, 2, 3) on kujutatud rida objekte. Teha nende objektide omaväärtusteisendused, võrrelda objekte saadud omaväärtuste alusel. Millised objektidest on sama kujuga? Analüüsida, kuivõrd tundlikud on omaväärtused objekti kuju väikestele muutustele (võtta ette mõni üksik objekt, muuta teda nt mõne piksli nihutamise või kustutamise teel, võrrelda uut tulemust esialgsega).

Soovitatavat (lisa)lugemist:

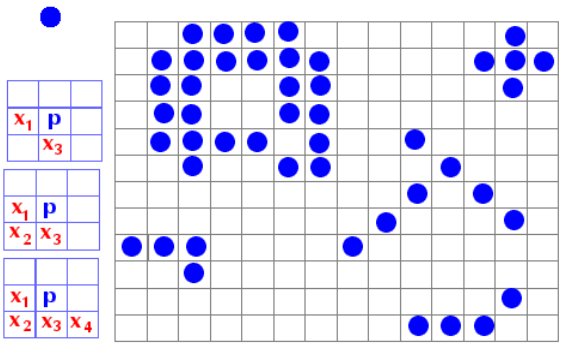
(*) Rafael C.Gonzalez, Richard E.Woods, Digital Image Processing, Addison-Wesley (ptk 10,11).

(*) Ramesh Jain, Rangachar Kasturi, Brian G.Schunck. Machine Vision. McGraw-Hill.

Aare.Luts.1@eesti.ee

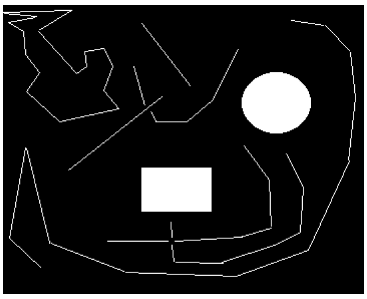
Teema 7 küsimused ja ülesanded on järgmised:

Küsimus 1. Nagu mainitud, võivad maskid olla mitmesugused. Paraku, erinevate maskide hinnaks võib olla asjaolu et teatava kujuga objekte ei ühendata. Joonisel on toodud mõned objektid (märgitud siniste punktidega piksleid kujutavas ruudustikus) ja mõned maskid. Millised objektidest saab ühendada milliste maskidega, ja milliste objektide jaoks millised maskid ei sobi?



Küsimus 2. Piirjoonte õhendamise algoritmi töö määrasid tingimused
a) 2<= **N** <=6 ?
b) **T** =1 ?
c) $p_2*p_4*p_6$ =0 ?
d) $p_4*p_6*p_8$ =0 ?
Mida kontrollivad tingimused a) ja b)? Selgitada.

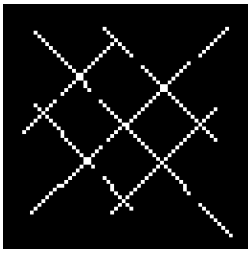
Ülesanne 1: Koostada pildil olevad jooned ja märkida ära sidusad piirkonnad (eristades objektid nt erinevate värvustega). Vähemalt mõne joone puhul anda tulemuseks suunakoodide tabel (Java paketis saab teha nt sel viisil et objektid hoida pildimassiivides, suunakoodid aga kirjutada reaalarvmassivi, viimase saab hiljem arvkujul ka faili kirjutada). Vastuses anda ka iga joone pikkus ja iga piirkonna punktide arv.



Küsimus 3. Hough teisenduse kirjeldamisel oli öeldud "me saame (a,b)-ruumi sirgete lõikepunktis palju äramärkimisi". Oletagem et esialgses (x,y)-ruumis oli ainult üks sirglõik. Kui palju äramärkimisi me sel juhul täpsemalt saame?

Küsimus 4. Hough teisenduse kirjeldamisel oli öeldud "määratakse kõigi (a,b)-ruumi piisavalt heledate punktide koordinaadid ja saadakse selle kaudu teada (x,y) ruumis olevate sirgete parameetrid". Kui meil on esialgsete sirgete parameetrid teada, peaksime me saama esialgse pildi täpselt taastada. Või ikka ei saa ja miks?

Ülesanne 2. Java-paketis on realiseeritud lineaarne Hough' teisendus. Alljärgnevalt on toodud pilt mis sisaldab hulga jooni. Leida piltidel olevate joonte võrrandid, kasutades vabalt valitud pilditöötlusmeetodit (mitte nt väljatrükitud pildil olevate joonte mõõtmist joonlauaga), võib kasutada Hough' teisenduse tulemusi. Vihje: loe paketis pakutavat abiinfot.



Ülesanne 3. Ehk koostaks Hough teisenduse etteantud diameetriga (vali ise) ringide jaoks ja prooviks tema (head) omadused ise järele? Kasutada võib konspektis toodud näitepilte, nendel on raadiuseks 19.

Ülesanne 4. Lisatud piltidel (1, 2, 3) on kujutatud rida objekte. Teha nende objektide omaväärtusteisendused, võrrelda objekte saadud omaväärtuste alusel. Millised objektidest on sama kujuga? Analüüsida, kuivõrd tundlikud on omaväärtused objekti kuju väikestele muutustele (võtta ette mõni üksik objekt, muuta teda nt mõne piksli nihutamise või kustutamise teel, võrrelda uut tulemust esialgsega).

8.teema õppematerjalid

Sellel leheküljel kirjeldatakse morfoloogilisi tehteid ja pildi mustrite tunnussuurusi. 1. Morfoloogilised tehted. 1.1. Põhitehete määratlus. 1.2. Dilation ja erosion. 1.3. Closing ja opening. 1.4. Piirjoone eraldamine. 1.5. Piirjoonega antud ala täitmine. 1.6. Hit-or-miss tehe. 1.7. Objektide skeleti leidmine. 2. Mustrid 2.1. Sarnaste objektide otsimine korrelatsiooni abil. 2.2. Alade piiritlemine, lähtudes morfoloogilistest tehetest. 2.3. Alade piiritlemine, lähtudes mustritest. 2.3.1. Histogrammi kasutamine. 2.3.2. Piirkonna heleduste regulaarsed muutused. 2.4. Pildiinfo tasandid. 2.5. Objektide klassifitseerimine.

Õpikeskkond:	TÜ Moodle
Kursus:	Pildiinfo töötlus (LOFY.05.055)
Koosta raamat:	8.teema õppematerjalid
Printed by:	Aare Luts
Kuupäev:	teisipäev, 22 mai 2012, 08:05

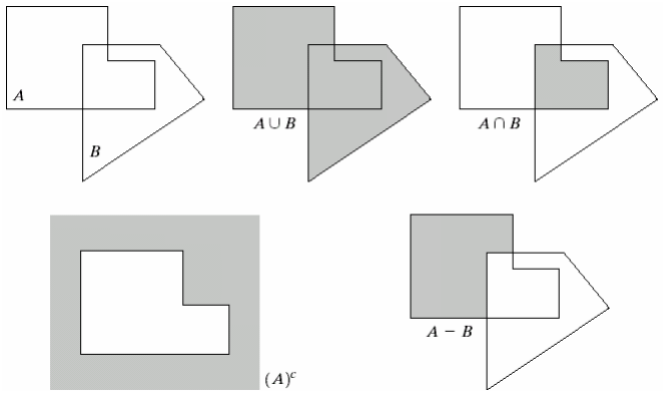
Sisukord

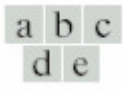
1. Morfoloogilised tehted.

Eespool, libiseva maski tehete käsitlemise juures, sai ära märgitud et libiseva maskiga võib siduda mistahes tehteid. Selgub, et üks seni käsitlemata tehete klass osutub iseäranis kasulikuks. Tõsi, nende tehete juures ei kehti pikalt käsitletud seosed sagedusruumiga. Veel tuleb ära märkida et üldjuhul kasutatakse seda uut tehete klassi eelnevalt binaarsele kujule viidud piltide edasisel töötlemisel. Halltoonides (värvustes) piksleid puudutavad tehted on defineeritud, kuid neid üldjuhul ei kasutata. Järelikult, enne uue klassi tehete kasutamist tuleb pilt muuta binaarseks, mis tähendab tavaliselt seda et pildil on juba eristatud objektide pikslid. Teisisõnu, uue klassi sisuks on tehted objektidega, mitte niivõrd tehted, mis muudavad pildi üldist iseloomu (üldist iseloomu muutvad olid nt histogrammi baasil koostatud tehted). Rangelt võttes ei tähenda pildi binaarsus seda et ta oleks just mustvalge (valged nt objektid ja must nt muu taust), tähtis on et suudetaks kirja panna kriteeriumid, millised pikslid vastavad objektidele (nt piklsid, milliste heledused on suuremad kui 127).

1.1. Põhitehete määratlus.

Uue klassi nimetuseks on morfoloogilised tehted ja matemaatiliselt on nende aluseks hulgateooria abil käsitletavad (loogilised) operatsioonid. Põhitehete tulemid on näidatud joonisel. Ülemisest reast vasakult paremale on näidete sisuks **a)** hulgad **A** ja **B** ise; **b)** hulkade **A** ja **B** ühend (ühend nagu ka kõik teised tulemushulgad on kujutatud halliga); **c)** hulkade ühisosa; **d)** hulga **A** täiendhulk $\mathbf{A}^C$; **e)** tehe "hulk **A** miinus hulk **B**".



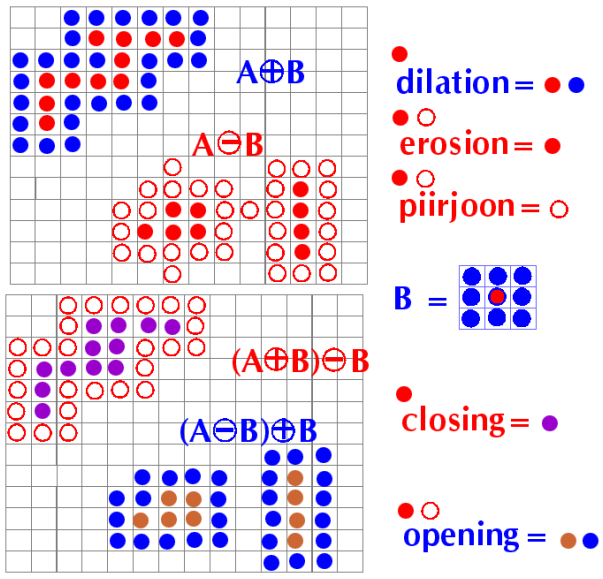


Suure osa edasiste tehete aluseks on nende loogiliste operatsioonide reeglistik. Kõigepealt, "hulk ise" määrab pikslite teatava hulga, hulka kuuluvad pikslid ei pruugi sugugi olla ruumiliselt kõrvuti nagu nad on siin toodud joonisel. Tehniliselt määratakse kuuluvus hulka tavaliselt mingi seda hulka kirjeldava unikaalse tingimusega (nt pikslid, millede heledused on vahemikus 100 kuni 200). Kui võrrelda hulka kuulumise tingimust millegi juba eespool tooduga, siis kõige paremini sobib analoogiks teema 7 punkti 1 all käsitletud tingimus "**p=0?**". Hulkade ühend ja ühisosa määratakse loogiliste tehetega "OR" ja "AND". Kui objektile kuuluvate pikslite logiliseks tunnuseks on "1" ja tausta loogiliseks tunnuseks on "0", siis on tehet *OR* määratud "kui kas **A** või **B** on 1, siis 1, muidu 0"; tehe *AND* on määratud "kui **A** ja **B** on mõlemad 1, siis 1, muidu 0". Täiendhulgas on "1" ja "0" vahetatud. Tehe "**A** *miinus* **B**" on eelmistest erinev, tema reegliks on "kui **A** *miinus* **B** võrdub 1, siis 1, muidu 0". Pildil toodud näites hulga **B** ühe osa asukohas hulk **A** puudub, tavalisel lahutamisel tuleks selles osas (määramatus miinus 1), siin asendatakse "määramatus" väärtusega "0".

Aare.Luts.1@eesti.ee

1.2. Dilation ja erosion.

Järgmise sammuna võtame kohe ette pilditöötluses praktiliselt kasutatavad baasoperatsioonid. Need on "dilation", "erosion", "opening" ja "closing". Operatsioonide tulemid on toodud joonisel. Nagu näha, ongi mängu tulnud mask, formaalselt võttes on mask asunud teise hulga **B** kohale. Mask toimib põhimõtteliselt samuti nagu libiseva maski eelnevates kasutustes. Ta nö libiseb üle pildi ja maski igas asendis muudetakse maski erilise punkti alla jääva piksli väärtust. Tavaliselt on selleks "eriliseks punktiks" maski keskpunkt, aga selleks võib määrata ka maski mõne muu punkti. Siin toodud näidetes on eriliseks punktiks maski keskpunkt, nagu tavaliselt.



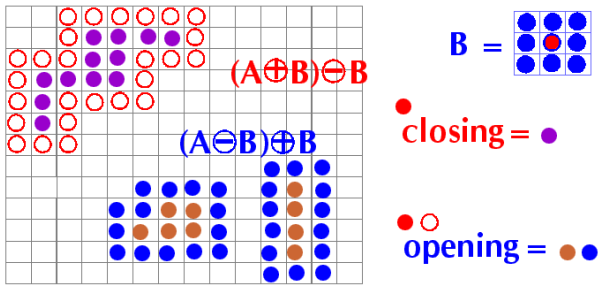
Vaatame detailsemalt tehet "**dilation**", mille käigus saab ühtlasi selgemaks, mida tähendavad näitepildil olevad tähised. Niisiis, alguses on meil mask **B** ja hulk (objekt) **A**, mis kujutab endast pildi vasakus ülemises nurgas üksteise kõrval asuvate punaste ringide hulka. Näitepildi paremal äärel on kõigi tehete selgitus, antud sümbolite abil. Vaadeldava tehte selgitus on tekstiks tõlgituna "punaste ringidega antud objekti *dilation* maskiga on võrdne objektiga, milline on märgitud kas punaste või siniste ringidega". Vaatleme detailsemalt mõnda selle tehte sammu.

Asugu mask **B** oma keskpunktiga pildiruudustiku ülemise rea vasakult esimesel veerul. Selles kohas tuleb arvutada tehe "*OR*". Kui kasvõi üks maski aktiivsetest ("1"-ga märgitud) punktidest satub kohakuti pildil oleva objekti mõne aktiivse ("1"-ga märgitud) punktiga, saab tulemuseks "1", mis kirjutatakse maski aktiivse punkti all olevasse pikslisse. Kui mask asub keskpunktiga ruudustiku ülemise rea vasakult esimesel veerul, ei satu kohakuti ühtki nii maskis kui ka pildil asuvat aktiivset punkti. Järelikult, selles kohas on tulemuseks "0", pildil on see märgitud tühja pikslina. Sama olukord kordub ka siis, kui mask liigub ühe piksli võrra kas paremale või alla (meenutagem, pildil on "aktiivsed" punaste ringidega märgitud pikslid). Kui maski keskpunkt saab asuma pildi ülemise rea vasakult kolmandal ruudul, olukord muutub. Seal pikslil satuvad kohakuti maski **B** alumine parempoolne aktiivne punkt ja pildil oleva objekti teise rea vasakpoolne punkt. Järelikult on tulemuseks "1", mis kirjutatakse pildile, antud näites on "1" märgitud sinise ringiga. tegevuste sama järjestus kordub, kuni kas kogu pilt või siis määratud piirkond on läbitud. Tehte "dilation" tulemiks on uus objekt, mis hõlmab nii esialgsed punaste punktidega märgitud pikslid kui ka uued, siniste ringidega märgitud pikslid. *Dilation* tulemusel silutakse objektide teravad tipud, nagu ka ühendatakse objektid kohtades, kus esineb katkemine. Tulemuseks on väliselt siledam ja sisemiselt paremini ühendatud objekt, tõsi küll sellise hinnaga et objekt on nüüd suurem kui ta oli enne.

Erosion tulem on mõneti vastupidine: lõigatakse maha objekti "karvad" (objekti piirjoonel avalduv müra) ja katkestatakse objekt kohtadest, kus ta oli ainult õrnalt seotud. Näiteks, kui me vaatame trükitud teksti skanneerimise tulemust, näeme me nii tähemärke ümbritsevaid "karvu" kui ka kohti, kus erinevad tähemärgid on ühendunud. *Erosion* abil saab "karvad" kõrvaldatud ja juhuslikud kokkupuuted katkestatud. Samuti kõrvaldatakse üksikud pikslid, mis on tihtipeale mitte tähendusega pikslid vaid müra. Tõsi, (ebasoovitavaks) kõrvaltulemiks on objekti üleüldine vähenemine.

Aare.Luts.1@eesti.ee

1.3. Closing ja opening.



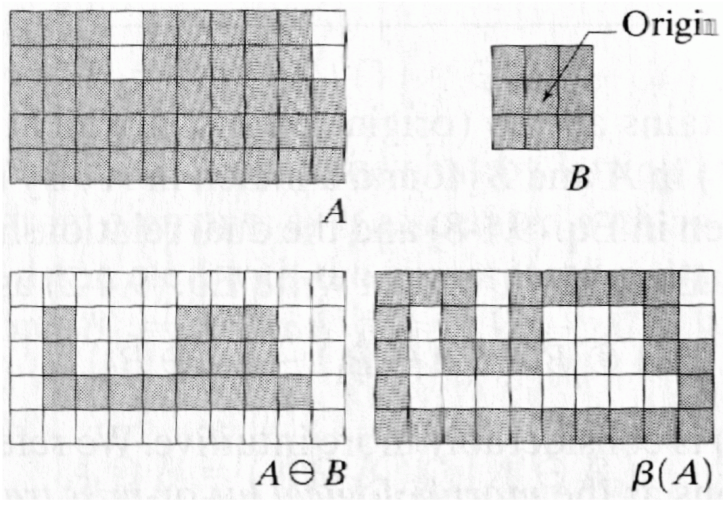
Nii *dilation* kui ka *erosion* tehted omavad üldjuhul ebasoovitavaid kõrvalmõjusid: esimene teeb objektid suuremaks ja teine väiksemaks. Kõrvalmõjude vähendamiseks kasutatakse kombineeritud tehteid: "closing" ja "opening". Esimesel juhul arvutatakse kõigepealt *dilation* tulem ja rekendatakse sellele tulemile *erosion* tehet. Nagu näitest näha, saab sel viisil uue objekti (joonisel märgitud lillade ringidega), mis on esialgse objektiga **A** (joonisel ülemises vasakus nurgas, märgitud punaste ringidega) üsna sarnane. Samas on esialgsele objektile **A** rakendunud soovitud muutused, antud juhul on silutud esialgset objekti iseloomustanud terav tipp. Teisel juhul (*closing*) arvutatakse kõigepealt *erosion* ja rekendatakse seejärel tehet *dilation*. Näites on sel viisil saadud tulem toodud objekti(de)na, mis on märgitud siniste ja kollaste ringidega. Võrreldes esialgse objektiga (joonisel objekt, mis sisaldab nii punaseid ringe kui punaseid rõngaid), on teravad tipud maha lõigatud ja esialgu ainult õrnalt seotud objekt on nüüd lahutatud kaheks eraldi objektiks.

Eelnenud näidetes kasutati kõikjal maski **B** sellist kuju, kus maski suuruseks oli 3•3 ja maski kõik ruudud olid "aktiivsed". See ei pruugi alati nii olla. Võib valida mistahes suuruse ja kujuga maski, sõltuvalt eesmärgist, kui suuri ja millise (spetsiifilise) kujuga objekte tahetakse tehetega muuta.

(Lisa)kirjandus: Gonzalez and Woods, Digital Image Processing, ptk.9. (konspekt)

Aare.Luts.1@eesti.ee

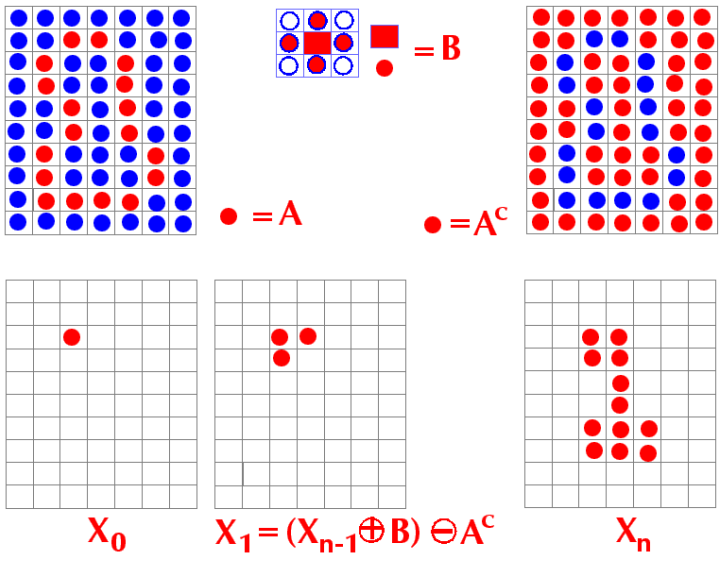
1.4. Piirjoone eraldamine.



Olgu meil olemas mingi sidus piirkond, koostatud näiteks viisil nagu kirjeldatud eelmise teema punktis 1. Kõrvaloleval joonisel on selliseks piirkonnaks hulk **A**, piirkond on sellel märgitud hallide ruutudega. Seda piirkonda piirava joone leidmiseks sobib tehe **β**(**A**) =**A** - (**A** *erosion* by **B**), kus **β**(**A**) on otsitav piirjoon, A on esialgne piirkond ja **B** on mask. Sobiva maski kuju on samuti näidatud joonisel, selle maski kõik punktid on "aktiivsed", ehk maski võrreldakse tema alla jäävate kõigi punktidega. Saadav piirjoon on kujutatud joonise parempoolses alumises osas, piirjoone punktid on märgitud halliga.

1.5. Piirjoonega antud ala täitmine.

Nüüd vastupidine operatsioon. Olgu antud mingi kinnine joon, kas ette antud või leitud näiteks mõnel eelmises teemas kirjeldatud viisil. Kõrvaloleval joonisel on piirjoon näidatud joonise vasakus ülemises osas, piirjoon **A** on märgitud punaste ringidega. Tehtes kasutatakse maski **B**, mille kuju on samuti näidatud. Erinevalt eelmisest, piirjoone eraldamise tehtest on maski diagonaalruudud nüüd mitteaktiivsed ehk nende alla jäävaid piksleid ei vaadata. Ala täimine algab mingist suvalisest otsitavasse alasse kuuluvast punktist.



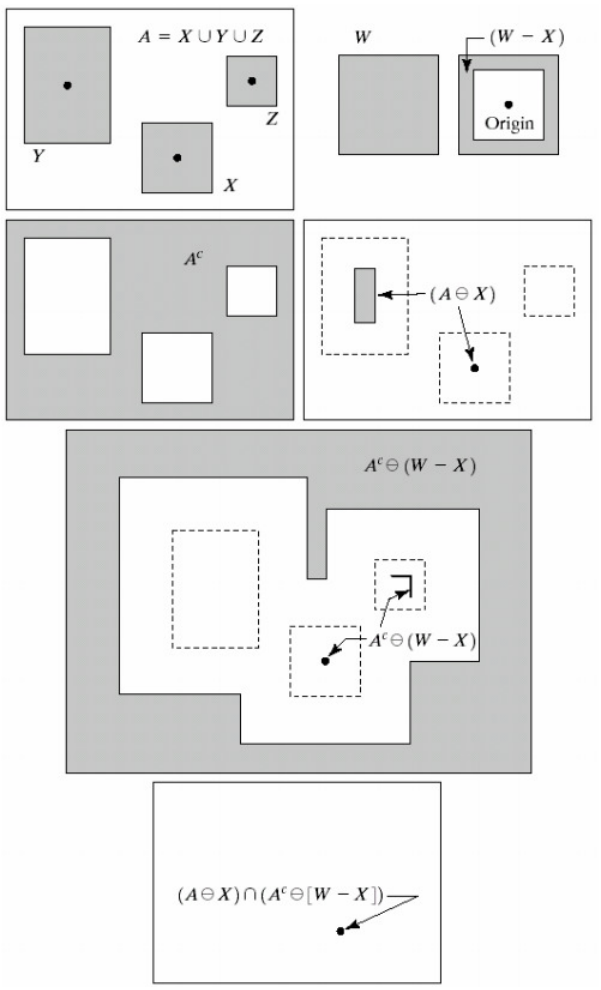
Esimene punkt tuleb määrata mõnel muul viisil, aga sellega ei ole ületamatuid probleeme. Kui piirjoone punktid on antud, võib võtta suvalise punkti ja nihkuda sellest ühe võrra piirkonna sisu poole. Millises suunas sisu asub, seda saab leida näiteks sel viisil et suurendada piirjoone valitud punkti x-koordinaati. Kui sisu asub piirjoonest paremal pool, jõutaks koordinaadi suurendamisel lõpuks piirjoone järgmisele punktile, seda põhjusel et joon on eelduse kohaselt kinnine. Kui koordinaadi suurendamisel piirjoone teist punkti ei leita, peab sisu asuma piirjoonest vasakul pool.

Otsitavat ala märgime tähisega $\mathbf{X}_n$, algolekut on järelikult sobilik märkida $\mathbf{X}_0$. Piirjoon täidetakse iteratiivselt, iga sammuga natukene. Esimese sammu tulem ja kasutatav tehe on näidatud joonisel. Tehteks on ($\mathbf{X}_{n-1}$ *dilation* by **B**) *erosion* by $\mathbf{A}^c$. Hulk $\mathbf{A}^c$ on hulga **A** täiend (ingl. "complement"), see on jonisel kujutatud hulga **A** kõrval.

Esimese sammu tulemiks on kolm punkti ehk kaks lisandunud punkti. Iteratsiooni jätkatakse, kuni uusi punkte enam ei lisandu. Antud juhtumi tulem $\mathbf{X}_n$ on näidatud joonise paremas alumises osas.

1.6. Hit-or-miss tehe.

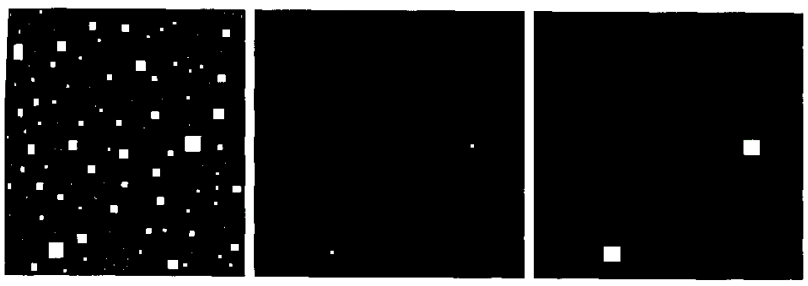
Selle tehte eesmärgiks on leida pildilt objektid, mis on täpselt sama kujuga kui etteantav mask. Näitena võib tuua tööstusrobotid, mis peavad kontrollima toodangu vastavust standardile. Näitena võib tuua ka hea kvaliteediga skänneeritud trükitud teksti,. eeldusel et see sisaldab ainult ühes stiilis tähemärke; sellisel juhul saab vaadeldava tehtega lahendada OCR (optical character recognition) ülesande. Objektide ja malli sobivuse uurimise muid (tõenäosuslikke) meetodeid vaatame allpool, mustrite käsitlemise juures. Nagu öeldud, siin vaadeldav tehe leiab täpsed vastavused.



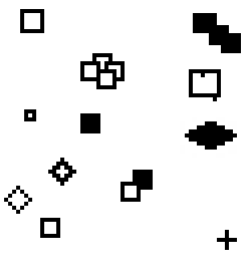
Joonisel on toodud pilt kolme üksteisest erineva objektiga; **X**, **Y** ja **Z**. Objektid on märgitud halliga. Otsitakse, kas ja kus kohal asub pildil objekt **X**. Alustuseks luuakse objekte sisaldava pildi täiendpilt $\mathbf{A}^C$. Siis luuakse hulk **W**, mis kujutab endast hulga X äärtega täiendatud versiooni (sellise saab näiteks dilation abil); samuti luuakse hulk (**W-X**), lahutades äärtega täiendatud **X**-st maha **X** enda. Summaarseks tehteks on {(**A** *erosion* by **X**) *ühisosa* [$\mathbf{A}^C$ *erosion* by (**W-X**)]}. Joonisel on näidatud ka tehte üksikute etappide tulemid. Kõigepealt on toodud (**A** *erosion* by **X**), kus objektide **X**, **Y** ja **Z** esialgsed asukohad on märgitud punktiirjoontega. Nagu näha, on erosion kaotanud täielikult objekti **Z**, objektist **X** on jäänud punkt ja objektist **Y** on jäänud ristkülik.

Järgnevalt on toodud [$\mathbf{A}^C$ erosion by **(W-X)**] tulem. Nüüd asub tühjus objekti **Y** kohal, objekti **X** kohal asub endiselt punkt ja objektist **Z** on jäänud üks nurkadest. Nende kahe tulemi ühisosa ongi tehte vastus, nagu näha on selleks punkt objekti **X** asukohas. Järelikult saadi otsitav tulem (leida täpne vastavus objektiga **X**) punkti kujul objekti **X** asukohas. Näiteks, kui pildil oli mitu objekti **X**, saab nüüd nad lihtsasti pildile tagasi paigutada või siis kasvõi kokku lugeda.

Kõrvaltoodud joonisel on näidatud algpilt, mis sisaldab palju mitme suurusega ruute ja lisaks üksikuid punkte, mis on tõenäoselt müra. Käsitletud tehte tulemusel saadakse et pildil on kaks otsitava suurusega ruutu ja paigutatakse need ruudud oma esialgsetesse asukohtadesse.

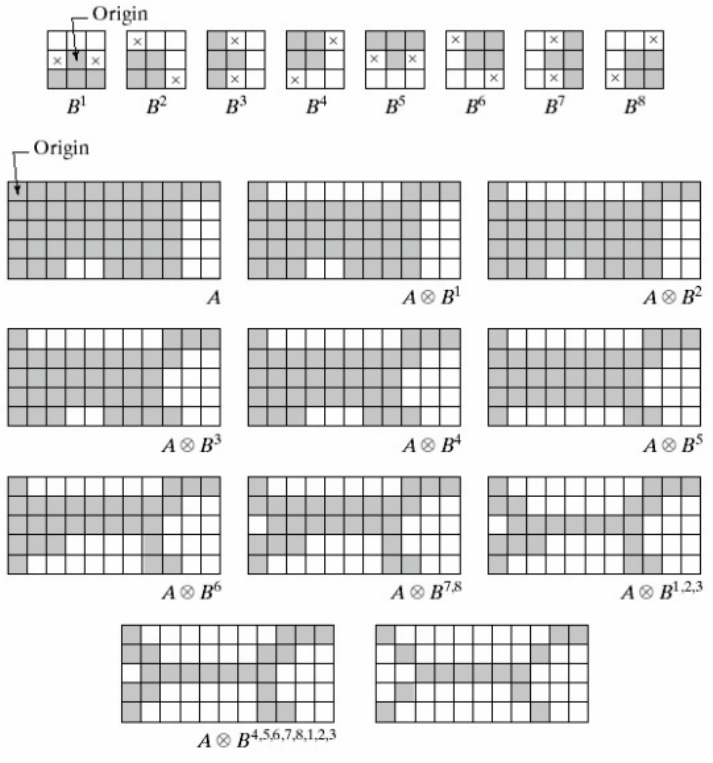


Kui otsida pildilt objekte, mis ei sisalda ainult täielikult täidetud piirkonda, vaid sisaldavad ainult piirjoont (vt kõrvalolev joonis), võib sobiva kujuga objektide otsimiseks piisata ka lihtsast erinevusest abs(**A-B**), kus **A** on pilt ja **B** on otsitav objekt.



1.7. Objektide skeleti leidmine.

Skelett on üks objekti võimalikest karakteristikutest. Oma olemuselt on ta joon. Joone puhul on mitmed temast tuletatud suurused (nt omaväärtused, projektsioonid) täpsemad kui tervet piirkonda kirjeldavad suurused, seetõttu sobivad need (täpsemad) tuletatud suurused objekti kirjaldamiseks paremini. Objektide kirjeldamise eesmärk on ikka endine: sarnaseid objekte samasse klassi paigutada ja erinevaid objekte eristada. Tõsi, objekti skelett võib olla (liiga) tundlik piirkonna väikestele muutustele, analooglist situatsiooni oli mainitud ka eelmise teema all. Siiski on skelett objekti kirjeldamise üks võimalusi, ja sellisena ei saa teda mitte tähelepanu alt välja jätta.

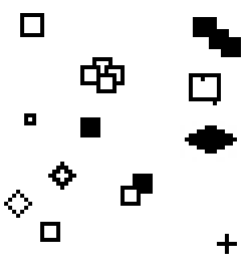


Pakutud algoritm sisaldab mitut sammu. Kõigepealt vajalikest vahenditest. Antud on piirkond **A**, mille skeletti tahetakse leida. Seejärel defineeritakse 8 maski $\mathbf{B}_1$ kuni $\mathbf{B}_8$. Neis maskides on olulised kõik ruudud peale ristiga märgitute. Ristiga märgitud ruutude all pildil võib asuda mistahes sisu, teistes ruutudes peab asuma sama sisu mis nende kohal asuva(te)s maski (de)s. Algoritmi iga üksiku sammu valemiks on **C** = **A** *miinus* (**A** *hit or miss* **B**). Igal sammul saadakse järjekordne lähendus $\mathbf{C}_n$ skeletile **C**.

Tsükli ühe pöörde sisuks on kaheksa järjestikust sammu, igal sammul kasutades vastavalt maske $\mathbf{B}_1$ kuni $\mathbf{B}_8$. Tsüklit korratakse kuni järjekordne pööre ei toonud enam mingeid muutusi. Tsükli lõpul võib ikkagi olla olukord kus saadud joon võib osutuda mitmeti käsitletavaks. Toodud joonisel nii ongi: joonte otsaalad on liiga laiad, neist võib läbi tõmmata mitu erinevat joont, sõltuvalt naabrushulgast, mida joonte tõmbamiseks kasutatakse. Joonisel on valitud üks naabrushulkadest (siin: $\mathbf{N}_m(p)$, naabruste kirjeldusi vt eelmisest teemast) ja teisendatud joont nii et ta sobiks selle naabrusega.

Nagu ka eespool juba märgitud, alati on tulemuse seisukohast määrava tähtsusega nii tehte valik kui ka maski (õige) suurus (ja kuju).

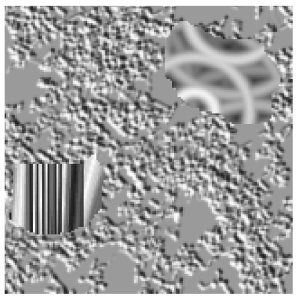
Ülesanne: Leida pildilt alumises vasakus nurgas asuva objekti esinemise kohad. Vihje: Java-paketis on vajalik tehe juba olemas, tuleb vaid leida ja õigesti kasutada. Ka sobiv mask on antud: Seejärel leida pildilt ainult täitmata väikeste ruutude asukohad. Sellel juhul sobiv mask koostada ise. Mõlema juhu vastused esitada piltidena, kus otsitud objektide asukohad on märgitud valgete punktidega mustal taustal (võib ka vastupidi).



Aare.Luts.1@eesti.ee

2. Mustrid.

Mustrid võivad olla parimaks või ka ainsaks kriteeriumiks, mille abil pildil olevaid objekte üldse määratleda. Eespool oli toodud näiteid, kuidas objekte (nähtusi) määrata (piiritleda) histogrammi omaduste alusel. Samuti käsitleti piiritlemist objekti ümbritseva joone abil. Mõnel juhul aga on erinevate objektide histogrammid sarnased ja mõnikord on objektide piirjoon sedavõrd hajus või taustaga lõiguti väga sarnane et seda on raske määrata. Siis võib olla abiks, kui võtta aluseks objektide (sisepiirkonna) mustrid. Ühe objekti piires peaksid mustrid (mustrite karakteristikud) olema sarnased, karakteristikute muutus vihjab teistsugusele mustrile ehk uuele objektile. Kõrvalolev pilt on üks selliseid, kus piirkondi tuleks määrata nende iseloomulike mustrite järgi.



Mustreid puudutav on leitav otsingusõna "(image) pattern recognition" alt. Vasteid tuleb tuhandeid, mis näitab üsna selgesti et see teema on üks keerukamaid, aga ka huvitavamaid. Kes tahab uusimaid trende ja uudiseid, võib vaadata nt ajakirja "Pattern Recognition letters".

Aare.Luts.1@eesti.ee

2.1. Sarnaste objektide otsimine korrelatsiooni abil.

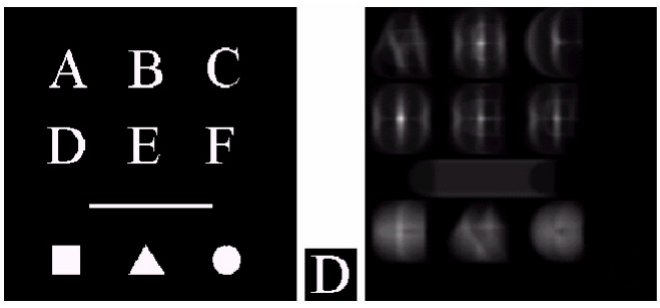
Eespool käsitletud "hit-or-miss" tehe leidis pildilt objektid, mis sobisid malliga (maskiga) täpselt. Tihtipeale on aga nii et täpset sobivust ei ole (mürad, juhuslikud moonutused, ...) ja ka eespool vaadatud morfoloogiliste tehetega (nt opening ja closing) ei õnnestu objekte nii palju parandada et kõik mürad ja juhuslikud erinevused kõrvaldatud saaksid. Sel juhul tuleb kasutada meetodeid, mis küll ei otsi ainult täpset vastavust, aga parima sobivuse korral annavad annavad siiski suurima signaali (nt sel kujul et vastav punkt pildil saab olema kõige heledam). Üheks sedalaadi meetodiks on korrelatsioon.

Eespool oleme vaadanud kahte korrelatsiooni (ingl. "correlation"). Esimene tuli jutuks DFT käsitlemisel, kus kõrvuti sidumiga nimetati ka korrelatsiooni. Nagu seal näidatud sai, ei ole see korrelatsioon objektide äratundmiseks hea. Teine korrelatsioon tuli jutuks omaväärtuste leidmise juures, kus leitud kovariatsioonimaatriks oli edasise teisenduse aluseks. See kovariatsioon sobib, ja korrelatsioon ei ole midagi muud kui normeeritud kovariatsioon. Omaväärtusteisenduse käsitlemise juures oli kovariatsioonimaatriks defineeritud keskväärtuse operaatorite kaudu. Siin on toodud otsene arvutusvalem kahe muutuja korrelatsioonikoefitsiendi $\mathbf{r}_{xy}$ jaoks. Mida suurem see koefitsient tuleb, seda sarnasemalt kaks muutujat käituvad.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Pildi puhul on üheks muutujaks mall ehk mask, mis sisaldab malli. Muutuja "mall" käitumist võib vaadata näiteks sel viisil et me liigume mingis kindlas järjestuses mööda maski ridu ja veerge ja vaatame, kuidas muutub malli heledus (värvus). Teine muutuja on pilt. Kui asetame maski pildi kohale, muutuvad ka maski alla jääva pildi pikslite värvused mingil viisil. Kui maskialuse ala värvused muutuvad sarnaselt maski värvustele, saamegi korrelatsioonikoefitsiendi suure väärtuse. Aga kui muutused on sarnased, ongi tõenäoselt tegu otsitava objektiga.

Siin tuleb sisse tingimus "tõenäoselt". Kui nt *hit-or-miss* tehtega saime me kas täpse vastavuse või ei midagi siis siin saame ainult objektide sarnasuse protsendi. Seejärel jääb igaühe enda otsustada, kas näiteks 95% sarnasust on objekti tuvastamiseks piisav, või tahetakse 99%, või piisab ka näiteks 90%-st. Kõrval on toodud näide, kus otsiti objekti "D" esinemist pildil. Pildi paremal pool on esitatud saadud korrelatsioonikoefitsiendid, normeerituna sel viisil et 100% on pildil esitatud suurima võimaliku heledusega. Nagu näha, on objekti D asukohas tõepoolest parempoolse pildi kõige heledam punkt, aga ka pildi muud alad ei ole kaugeltki mustad. Järelikult, kui otsida objekte korrelatsiooni abil, tuleb analüüsida nii saadud tulemuste läviväärtust (kas 95% on piisav?) kui ka suurima väärtuse erinevust suuruse poolest järgmisest väärtusest (kas 85% ja 95% on piisavalt erinevad et esimesel juhul väita et objekti pole, teisel juhul aga et objekt on?).

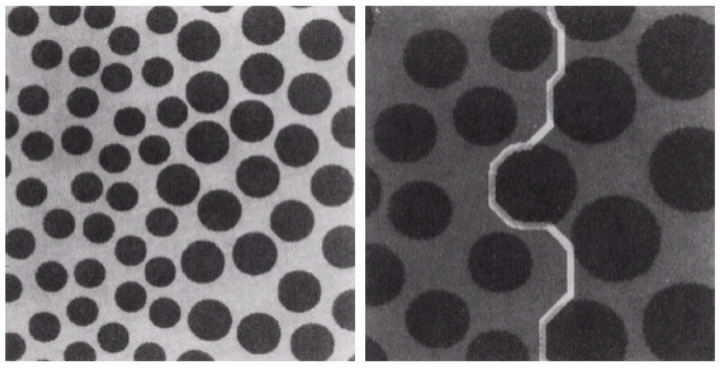


Ülesanne: Leida pildilt vabalt valitud tähemärkide (vähemalt 2 märki) esinemise asukohad, lugeda kokku, mitu märki esines. Tulemuseks anda pilt, kus tähemärkide esinemiste asukohad on märgitud taustast erineva värviga, eri tähemärgid eri värviga. Lisaks anda tekstifail, mis sisaldab arve, mitu korda mingi märk esines, üks arv ühe tähemärgi kohta.

Aare.Luts.1@eesti.ee

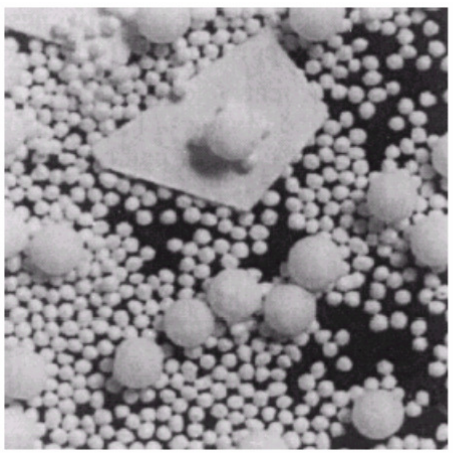
2.2. Alade piiritlemine, lähtudes morfoloogilistest tehetest.

Nagu juba eespool mainitud, mõnikord ei ole piirjoone leidmise tavavahenditega (nt Laplace operaator) midagi peale hakata, mõnikord saab alasid ja neid eristavat joont määrata hoopis paremini kui lähtuda alade sisemisest struktuurist (struktuuri erinevustest). Üks näide on kõrvaltoodud pildil. Visuaalselt saab eristada kaht erinevat ala: üks on täidetud väikeste ringidega, teine suuremate ringidega. Aga kuidas leida nendevahelist piirjoont, kasutades ainult matemaatikat? Üks võimalus on rakendada "opening" tehet, kasutades selles järjest suurenevat maski. Siinsel juhul tuleb eelnevalt määrata et objektid on tumedad ja taust (null) on hele. Tehte aluseks on *erosion*, mis kõrvaldab kõik maskist väiksemad objektid, *opening* tehte teise poole ülesandeks on hoida pilt igal sammul enamvähem muutumatu, aga seda saab ta teha ainult seni, kuni *erosion* ei ole objekte täielikult kustutanud. Maski teatavast suurusest alates on väikeste objektide piirkonnaks tehte tulemiks null, ehk sellel alal on kõik objektid kustutatud.



Olukorda, kus osa pildist on null, on lihtne kontrollida, näiteks kasutades ala trasseerimiseks mingit veel suuremat maski: kui sellise maski all on kõik null, on üks aladest juba tühi. Kui ühel pool on objektid kustutatud, on lihtne. Nüüd tuleb ainult kas rakendada saadud pildile järjest mitu korda *dilation*, kuni suuremad ringid on kõik ühendatud, või siis rakendada üks *closing* viimati kasutatust veel suurema maskiga. Kui rakendada järkjärgulist *dilation*, tuleb iga sammu järel veel suurema maskiga kontrollida, kas pildil on juba piirkondi, mis on ühtlaselt mitte-nullid. Kui selline piirkond leitakse, on suuremate ringide vahed täidetud ja alade eraldamise võib lugeda lõpetatuks.

Kuna nüüd on pildil ainult kaks piirkonda: "null" ja "mitte-null", saab piirjoone leida mistahes eespool käsitletud meetodiga, sealhulgas ka Sobeli operaatori abil. Järgnevalt vaatame näidet, kus ülesandeks on sarnase suurusega objektide kokkulugemine. Olemuslikult, loendada ei saa enne kui objektid on sisuliselt piiritletud. Järelikult, kui me suudame loendada, suudema me ka loendatavad objektid piiritleda. Joonis ise näib üsna lootusetu, siit on isegi silmaga raske midagi kokku lugeda. Järgnev algoritm aga peaks sobima.



Et lugeda kokku erinevate suurustega objektid (leida histogramm suuruste järgi), kasutatakse jällegi *opening* tehet järjest suureneva maskiga. Kasutatava maski mõõt saab nüüd määrama karakteristliku mõõdu ehk histogrammi x-telje väärtuse. Maski iga suurusega tehtud *opening* lahutatakse originaalpildist ja loetakse kokku, mitmes pikslis oli erinevus. See erinevus saab pärast teisendusi histogrammi y-väärtuseks kohal x. Miks nii? Kuna opening objektide suurusi üldiselt ei muuda, ei muutu maskist suuremate objektide korral pildil eriti miski, ja piltide erinevus on väike. Kui aga mask saab suuremaks mõnedest (väiksematest) objektidest, siis kustutab *opening* esimene aste *erosion* need objektid täielikult ja pildil saab selles kohas olema null, mida ei muuda ka *opening* teine samm. Sellisel juhul on piltide erinevus märkimisväärne.

Tuleb siiski märkida et maski kasvamisel saab piltide erinevus ainult suureneda, seetõttu on histogrammi y-väärtuse määramiseks oluline mitte piltide erinevuse väärtus vaid erinevuse muut, kui võrrelda maski eelmise suuruse juures leitud erinevusega.

Ülesanne: Leida sellel pildil erineva suurusega ringide poolt määratud alade piirid. Vastus anda pildina, kus originaalpildile on lisatud mingi erineva värvusega (nt punasega) piirjooned. Ühtlasi loendada, mitu väiksemat ja mitu suuremat ringi pildil on.

Aare.Luts.1@eesti.ee

2.3. Alade piiritlemine, lähtudes mustritest.

Mustrite kirjeldamise karakteristikuid on palju, igal konkreetsel juhul sobivad mõned vähem ja mõned rohkem. Igale spetsiifilisele ülesandele sobib oma meetod, mille valik võib olla kõike muud kui triviaalne. Siin märgitakse ära mõned karakteristikutest, ja illustreeritakse nende tulemeid mõne konkreetse näitega. Ülejäänud karakteristikute (nt Entroopia, Energia, Kontrast, Homogeensus) kohta loe kirjandusest.

2.3.1. Histogrammi kasutamine.

Esimest liiki karakteristikud lähtuvad piirkonna integraalsetest iseloomustussuurustest, näit. histogrammi momendid. Mis on suuruse momendid, sellest oli juba juttu, olgu üldvalemid toodud uuesti. Valemites olevad x ja y on näiteks koordinaadid pildil, aga võivad olla ka mingid muud suurused. Funktsioon **f(x,y)** on kaalufunktsioon, mis võib olla milline tahes, sõltuvalt eesmärgist. **p** ja **q** on astmenäitajad, operaator **E** on keskväärtuse arvutamise tehe. Antud juhul vaatame me ainult üht suurust, heledusväärtust, seega muutujat y sisaldavad liikmed kaovad ja **f(x,y)** asemal saab olema **f(x)**, selle kohale paneme histogrammi. Seega on antud juhul nullindaks momendiks (**p**=0) pildi heleduste keskväärtus, esimene moment iseloomustab histogrammi tipu nihet heleduste keskväärtuse suhtes, teine moment iseloomustab histogrammi laiust ja kolmas moment iseloomustab seda, kuhu poole on histogramm kaldu (momentide üksikasjalikumat kirjeldust vt statistilisi jaotusi käsitlevatest kirjutistest, nt sellest konspektist).

$$m_{pq} = SUM_x SUM_y \{ x^p y^q * f(x,y) \}$$

$$w_{pq} = SUM_x SUM_y \{ (x-E[x])^p * (y-E[y])^q * f(x,y) \}$$

$$E[x] = m_{10}/m_{00}$$

Näiteks on vaadeldud 12 pilti (mustrit), alljärgnevalt esitame nende piltide jaoks arvutatud karakteristikud ja lõpuks püüame kokku võtta, millised mustritest on sarnased ja millised selgesti märgatavalt erinevad. Histogrammide momendid (**p**=0,1,2,3) keskmise heleduse suhtes on alljärgnevad. Momendid on normaliseeritud, s.t. väga suured ja väga väikesed arvud on vastavalt jagatud ja korrutatud et tulemuseks saadud arvud oleksid lihtsa kujuga. Võrdlemiseks pole olulised absoluutsed vaid suhtelised väärtused.

muster_1.bmp: 105, 15, 38, 1829
muster_2.bmp: 56, 22, 9, 297
muster_3.bmp: 126, -7, 42, -64
muster_4.bmp: 124, -4, 37, 465
muster_6.bmp: 142, -16, 23, -72
muster_7.bmp: 150, -10, 23, -623
muster_8.bmp: 187, 7, 126, -15170
muster_9.bmp: 187, -7, 126, -15147
muster_a.bmp: 152, 8, 18, -137
muster_b.bmp: 133, 0.6, 11, -240
muster_c.bmp: 218, -4, 79, -14531
muster_d.bmp: 218, -4, 79, -14514

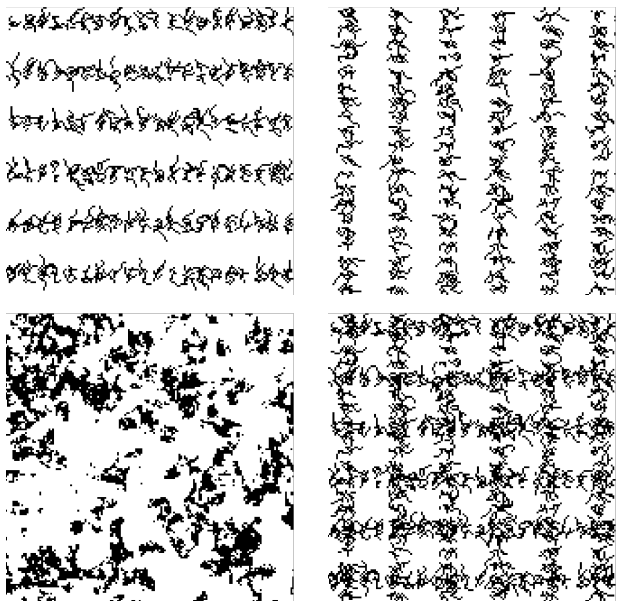
Histogrammi momentide kasutusvaldkond on põhimõtteliselt piiratud histogrammide kasutusvõimalustega. Histogrammi abil ei saa põhimõtteliselt eristada nt mustreid c ja d, mis annavad sama histogrammi kuid millede karakteristlikud suunad on erinevad. Samas on arvata et c ja d erinevad selgesti suuremast osast ülejäänud mustritest (v.a. ehk 8 ja 9), põhjusel et nende 3.momendid on teistest selgesti eristuvad, aga omavahel üsnagi sarnased.

Aare.Luts.1@eesti.ee

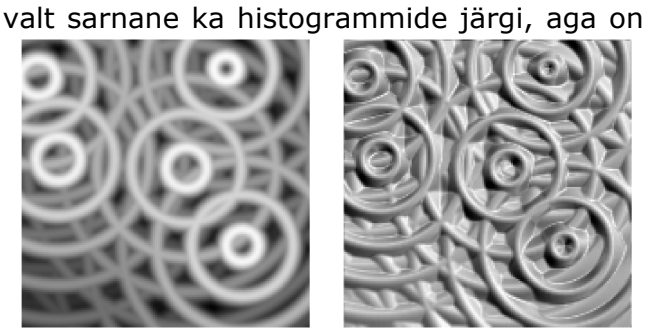
2.3.2. Piirkonna helelduste regulaarsed muutused.

See tunnus sisaldab tavaliselt infot mustri (helelduste) (regulaarse) mikrostruktuuri kohta. Et seda hinnata, võib leida helelduste ruumilise muutumise maatriksi, nim. ka "**gray-level co-occurence matrix**". Tehte sisuks on otsing, kui sageli ja milliseid heledusmuutusi toimub pildi suhteliselt väikeses piirkonnas (tihti isegi naaberpikslite vahel, nt {x,y}...{x+1,y+1}). Maatriksi koostamiseks on mitmeid võimalusi, kui arvestada kõikvõimalikke heledusmuutusi, tuleks 256•256 maatriks, mis on üldjuhul liiga suur. Seetõttu helelduste kõikvõimalikke muutusi vähemalt korraga tavaliselt ei vaadelda. Piirdutakse mõne väljavalitud helelduse (helelduste vahemikuga). Siin vaatleme helelduste ruumilisi muutusi (punkti {x,y} heledus *versus* punkti {x+2;y+2} heledus), iga pildi jaoks on tulemus esitatud kolme arvuna juhtudel (1. arv: heledused erinesid vähem kui 8 ühikut, 2. arv: heledused erinesid vähem kui 30 ühikut, 3. arv: heledused erinesid rohkem).

muster_1.bmp: 43, 61, 152
muster_2.bmp: 44, 73, 138
muster_3.bmp: 18, 50, 189
muster_4.bmp: 68, 143, 45
muster_6.bmp: 30, 68, 157
muster_7.bmp: 18, 52, 186
muster_8.bmp: 186, 0, 70
muster_9.bmp: 169, 0, 87
muster_a.bmp: 36, 41, 178
muster_b.bmp: 173, 47, 36
muster_c.bmp: 200, 0, 56
muster_d.bmp: 200, 0, 56



Selle tunnuse järgi on väga sarnased paarid (c,d) ja ka (8,9). Sarnasepoolsed on ka (1,2,a) ja (6,7). Pildid 4 ja b on teistest kõige erinevamad. Kui vaatame ka histogrammidest tuletatud tunnuseid, näeme et (c,d) ja (8,9) sarnasus leidis kinnitust. Paar (6,7) tundub aktsepteeritavalt sarnane ka histogrammide järgi, aga on selge et ta on erinev näiteks paarist (8,9). Kui pilte vaadata, tundub ka vaadates samamoodi. Järelikult on juba siintoodud tunnuste järgi võimalik mustrite erinevuse ja sarnasuse kohta üht-teist öelda, aga võimalikke tunnuseid on palju, ja iga tunnus võib midagi lisada.

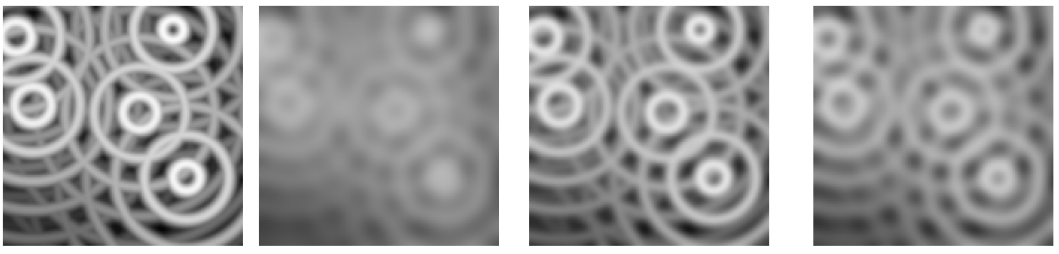


Ülesanne: Koostada programm, millega saaks uurida helelduste ruumilisi muutusi. Nagu eespool näha, üks võimalus on juba läbi arvutatud, aga võimalikke kombinatsioone on palju rohkem. Kombinatsioonide all mõtleme esiteks seda, milliste pikslite heledusi omavahel võrreldakse (siin näites: (x,y) *versus* (x+2,y+2)). Teiseks mõtleme seda, milliseid heledusmuutusi loendatakse. Valida mingi muu kombinatsioon ja võrrelda tulemusi eespool toodud tabeliga. Kui ise ei suuda kombinatsiooni ära otsustada, siis võrrelda nt (x,y) *versus* (x+2,y-2) ja (x,y) *versus* (x-2,y-2). Kas see muudab sarnaste ja erinevate mustrite osas midagi? Võib muuta ka heledusvahemikke. **Vihjeteks:** 1) kui programm on juba valmis, läheb kiiresti; 2) näites kasutatud programm mahtus ühele leheküljele, aga seda pole üldse optimiseeritud, nii et saab ka lühemalt; 3) Java-paketis on mitmed kasulikud funktsioonid juba olemas, nt saab võrdlemise teha sobiva maskitehtega, aga maskitehted on kõik otse programmist välja kutsutavad, loe juhendeid.

Aare.Luts.1@eesti.ee

2.4. Pildiinfo tasandid.

Pildiinfo erinevatel tasandite all mõeldakse piltlikult võttes pildi vaatamist erinevatelt kaugustelt. Lähemalt vaadates paistavad detailid, kaugemalt vaadates suuremamastaabilised omadused. Igal tasandil domineerib teatavat spetsiifilist liiki info, alumistel tasanditel avalduvad detailid ja kõrgematel avaldub info (pildi) üldisem iseloom. Seetõttu võib olla mõtet iseloomustada (võrrelda) pilte (mustreid) mitte ainult ühelt "kauguselt", vaid nii ühelt kui ka mõnelt teiselt tasandilt. Näiteks olgu toodud pilt



ja selle kaugemad tasandid.

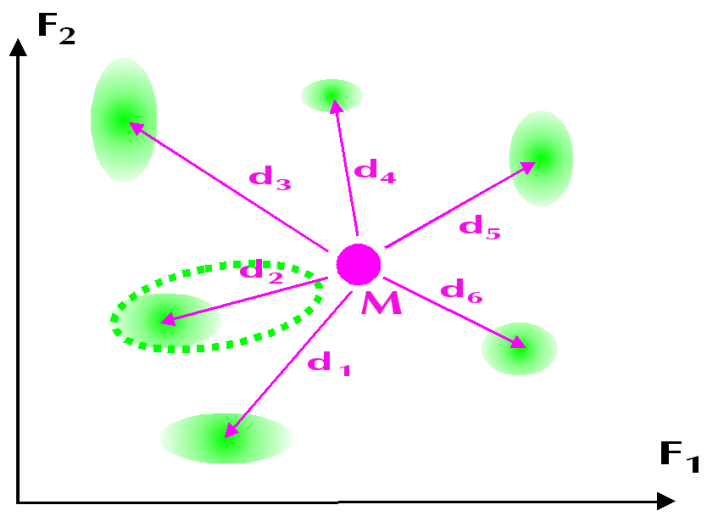
Tasandesitus võimaldab ühtlasi vähendada ka pildi mõõdet, kuna kaugemate tasandite pilte saab esitada väiksema arvu pikslitega. Esimese tasandi pildist kaugematele liikumiseks tuleb modelleerida efekte, mis kaasnevad pildist kaugenemisega. Sellega muutub pilt üldiselt udusemaks, detailide nähtavus väheneb. Sellise efekti analoog on pildi silumine. Nagu silumist (keskmistamist) käsitlevas osas sai mainitud, ei ole kõik keskmistamismeetodid sugugi võrdselt head. Mitmete omaduste pärast eelistatakse sageli Gaussi keskmistamist, ja selles ülesandes eriti. Seega tuleb tasandite konstrueerimiseks pilti kõigepealt keskmistada, misjärel võib soovi korral vähendada ka pildi pikslite arvu.

Aare.Luts.1@eesti.ee

2.5. Objektide klassifitseerimine.

Klassifitseerimine tähendab, et objekti mingite tunnuste abil tuleb objekt määrata ühte (huvipakkuvatest) klassidest, näit. OCR ülesandes tuleb määrata, millise sümboliga on tegemist. Eespool me suure osa sisust sellega tegelesimegi et näitasime, kuidas saab luua objekte iseloomustavaid tunnuseid. Kui ülesandeks on objektide klassifiseerimine (mis on sageli pilditöötluse ülim ülesanne), tuleb valida selle pildi(klassi) jaoks sobivad tunnused, need tunnused (pildi eelneva töötlemisega) luua ja seejärel paigutada tunnused nö tunnuste ruumi. Tunnused võivad olla millised iganes, peaasi et nad suudavad objekte piisavalt iseloomustada ja eristada.

Näiteks, leides tähemärkide x-ja y-teljelised projektsioonid on selge et tähemärgi **I** projektsioonid saavad erinema tähemärgi **O** projektsioonidest. Paigutades x-projektsioonid nt x-tejele ja y-projektsioonid y-teljele, hakkavad (x,y) tasandile tekkima sidusad piirkonnad, iga tähemärk kaldub asuma temale omases piirkonnas, ideaaljuhul (kui tähemärgid on kõik täpselt ühesugused) moodustavad kõik tähemärgid sellel tunnuste tasandil selgesti eristuvad punktid.



Joonisel toodud juhtum ei ole ideaalne, sellel on tunnuste F_1 ja F_2 tasandil moodustunud piirkonnad, iga sidus piirkond tähistab tõenäoselt ühte objekti. Siis on sellele tasandile tekkinud punkt **M**, mis võib iseloomustada uut, senitundmatut objekti. Kui me oleme kindlad et see objekt peab olema üks juba tuntutest, tuleb meil omistada ta ühte olemasolevatest klassidest. On loogiline valida see klass, mis asub punktile **M** kõige lähemal. Aga seejärel tekib kohe võimalus algoritmi (selliseid algoritme nimetatakse ka klassifikaatoriteks) õpetada. Kui punkt **M** omistati klssi d_2, on loogiline oletada et senine klass d_2 ei ole päris õiges kohas. Kuna temasse kuulub nüüd ka punkt **M**, on loogiline seda klassi punkti **M** suunas välja venitada ja loota et järgmisel korral õnnestub klassifitseerimine juba paremini.

Klassifitseerimismeetodite kohta loe kirjandusest juurde. See on nii ulatuslik teema et siin vähegi detailset käsitlust anda läheks ilmselt liiale, põhimõtted on igatahes antud.

Loe (lisaks) :

(*) Ramesh Jain et al., Machine vision, ptk. 7. Viide e-raamatule: http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.htm

(*) Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, ptk 12. Internetist leitud konspekt.

Aare.Luts.1@eesti.ee

Teema 8 küsimused ja ülesanded on järgmised:

1. Leida pildilt alumises vasakus nurgas asuva objekti esinemise kohad. Vihje: Java-paketis on vajalik tehe juba olemas, tuleb vaid leida ja õigesti kasutada. Ka sobiv mask on antud. Seejärel leida pildilt ainult täitmata väikeste ruutude asukohad. Sellel juhul sobiv mask koostada ise. Mõlema juhu vastused esitada piltidena, kus otsitud objektide asukohad on märgitud valgete punktidega mustal taustal (võib ka vastupidi).

2. Leida pildilt vabalt valitud tähemärkide (vähemalt 2 märki) esinemise asukohad, lugeda kokku, mitu märki esines. Tulemuseks anda pilt, kus tähemärkide esinemiste asukohad on märgitud taustast erineva värviga, eri tähemärgid eri värviga. Lisaks anda tekstifail, mis sisaldab arve, mitu korda mingi märk esines, üks arv ühe tähemärgi kohta.

3. Leida sellel pildil erineva suurusega ringide poolt määratud alade piirid. Vastus anda pildina, kus originaalpildile on lisatud mingi erineva värvusega (nt punasega) piirjooned. Ühtlasi loendada, mitu väiksemat ja mitu suuremat ringi pildil on.

4. Koostada programm, millega saaks uurida mustrite heleduste ruumilisi muutusi. Pildid on: 1, 2, 3, 4, 6, 7, 8, 9, a, b, c, d. Nagu konspektis näha, üks võimalus on juba läbi arvutatud, aga võimalikke kombinatsioone on palju rohkem. Kombinatsioonide all mõtleme esiteks seda, milliste pikslite heledusi omavahel võrreldakse (konspektis võrreldi (x,y) *versus* (x+2,y+2)). Teiseks mõtleme seda, milliseid heledusmuutusi loendatakse. Valida mingi muu kombinatsioon ja võrrelda tulemusi konspektis toodud tabeliga. Kui ise ei suuda kombinatsiooni ära otsustada, siis võrrelda nt (x,y) *versus* (x+2,y-2) ja (x,y) *versus* (x-2,y-2). Kas see muudab sarnaste ja erinevate mustrite osas midagi? Võib muuta ka heledusvahemikke. **Vihjeteks:** 1) kui programm on juba valmis, läheb kiiresti; 2) konspekti näites kasutatud programm mahtus ühele leheküljele, aga seda pole üldse optimiseeritud, nii et saab ka lühemalt; 3) Java-paketis on mitmed kasulikud funktsioonid juba olemas, nt saab võrdlemise teha sobiva maskitehtega, aga maskitehted on kõik otse programmist välja kutsutavad, loe juhendeid.

Sissejuhatus viimasesse teemasse, mille sisuks on referaat

Koostatava referaadi sisuliseks eesmärgiks on anda põhjus uurida mingit teemat nii põhjalikult kui selle kursuse käigus võimalik. Seetõttu on pakutavate teemade nimekiri alles teine valik. Esimeseks valikuks peaks olema teema, mis iseendale tundub huvi pakkuvat, ja mille lahtikirjutamise tulemust oleks rõõm ka teistega jagada. Kui teema on ise valitud lemmik, on tulemus üldjuhul parem. Kui oma lemmikteema olemas, tuleks sellest õppejõule teada anda, nii igaks juhuks, võibolla on põhjus mõnda teist teemat soovitada. Parem enne kokku leppida kui pärast vaevelda. Kui siiski peaks juhtuma sedamoodi et ise ei suuda kuidagi teemat välja mõelda, võib mõistagi kasutada ka alljärgnevat nimekirja.

Teemade nimekiri (kordan, see nimekiri peaks olema alles teine valik).

1) Piltide pakkimisalgoritmid.

2) Pildi geomeetriliste moonutuste kõrvaldamise automatiseeritud meetodid (st sellised, mis ei nõua kinnispunktide käsitsi määramist).

3) Canny Edge detector (kindlasti koos ise saadud näitetulemustega).

4) Läviväärtuste automaatne määramine (ise tehtud näidetega).

5) Wavelet-teisendus ja milleks seda kasutatakse, mõni ise tehtud näide ka.

6) Piltide moonutuste kõrvaldamise matemaatilised meetodid.

7) Joonte koostamise meetodid ja algoritmid (ise tehtud näidetega).

8) (Sarnaste/erinevate) piirkondade eraldamine ja ühendamine (region splitting and merging).

9) Objektide kirjeldamine ja võrdlemine.

10) Stereonägemine, roboti juhtimine stereopildi alusel.

11) Liikumise käsitluse meetodid.