Lappeenrannan teknillinen korkeakoulu
Tietotekniikan osasto

Spektrivideo

Diplomityön aihe on hyväksytty Tietotekniikan osaston osastoneuvostossa 15.9.1999.

Tarkastajat: professori, TkT Heikki Kälviäinen
dosentti, FT Jussi Parkkinen
Ohjaaja: dosentti, FT Jussi Parkkinen

Lappeenrannassa 8.11.1999

Panu Koponen
Skinnarilankatu 28 D 11
53850 Lappeenranta
puh. +358-400-954 688

TIIVISTELMÄ

Tekijä:	Panu Koponen
Nimi:	Spektrivideo
Osasto:	Tietotekniikan osasto
Vuosi:	1999
Paikka:	Lappeenranta

Diplomityö. Lappeenrannan teknillinen korkeakoulu.

66 sivua, 29 kuvaa, 10 taulukkoa, 1 liite.

Tarkastajat: prof. Heikki Kälviäinen, dos. Jussi Parkkinen

Hakusanat: värikuva, spektrikuva, videokuvan tiivistys, kuvankäsittely, konenäkö

Työn tavoitteena oli tutkia värispektreistä koostuvien monikomponenttikuvien, spektrikuvien, muodostamien kuvasarjojen tiivistämistä ja esittämistä videona. Erityyppisiä menetelmiä testattiin sekä kuvien välistä että kuvien sisäistä tiivistystä varten.

Tutkittuja tiivistysmenetelmiä olivat liikkeen kompensointi, pääkomponenttianalyysi ja aallokkeet. Liikkeen kompensoinnilla pyrittiin vähentämään peräkkäisten kuvien välillä toistuvaa informaatiota. Pääkomponenttianalyysin ja aallokkeiden avulla pyrittiin säilyttämään kuvan sisällä vain tarpeellinen osa informaatiosta. Parhaat tulokset sekä visuaalisesti että tiivistyssuhteen kannalta tuottivat liikkeen kompensointi ja pääkomponenttianalyysi yhdessä sovellettuina.

Työssä toteutettiin ohjelmat spektrivideon koodaukseen ja dekoodaukseen testattuja tiivistysmenetelmiä käyttäen. Videokuvan esitystä varten dekooderiin lisättiin yksinkertainen käyttöliittymä, jolla spektrivideota voidaan katsella tavallisella tietokoneen näytöllä.

ABSTRACT

Author: Panu Koponen

Title: Multispectral video

Faculty: Department of Information Technology

Year: 1999

Place: Lappeenranta

Diploma thesis. Lappeenranta University of Technology.

66 pages, 29 figures, 10 tables, 1 appendix.

Supervisors: prof. Heikki Kälviäinen, doc. Jussi Parkkinen

Keywords: colour image, multispectral image, video compression, image processing, computer vision

The purpose of this diploma thesis was to study how to compress and present video sequences of multispectral colour images. Several methods were tested for both intra coding and inter coding of pictures in a sequence.

The examined compression methods included motion compensation, principal component analysis, and wavelets. Motion compensation was utilized to reduce repeated information between consecutive pictures in a sequence. Using the wavelets and principal component analysis, an attempt was made to retain only necessary parts of the information in a picture. The best results, judged by compression performance and visual quality, were achieved by combining motion compensation with the principal component analysis.

Besides studying efficient compression methods, software for encoding and decoding multispectral video was implemented. A simple user interface for video presentation in RGB colour mode was included in the decoder application.

Alkusanat

Tämä diplomityö on tehty Lappeenrannan teknillisessä korkeakoulussa Tietojenkäsittelytekniikan laitoksen tutkimuslaboratoriossa vuosien 1998-99 aikana. Työ liittyy laboratorion väritutkimusprojektiin, joka keskittyy spektrivärikuvien tutkimukseen ja analysointiin.

Kiitän työn ohjaajaa ja tarkastajaa dosentti Jussi Parkkista sekä tarkastajaa professori Heikki Kälviäistä arvokkaasta ohjauksesta ja kärsivällisyydestä. Lisäksi kiitän lehtori Arto Kaarnaa lukuisista hyödyllisistä neuvoista työn aikana ja laboratorion koko henkilökuntaa miellyttävästä työympäristöstä.

Sisältö

Symboliluettelo

A/D	Analog-to-Digital
AVIRIS	Airborne Visible InfraRed Imaging Spectrometer
CBP	Coded Block Pattern
CIE	Comission Internationale de l'Éclairage
CMP	Coded Motion Pattern
CPP	Coded Pixel Pattern
CR	Compression Ratio
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FFT	Fast Fourier Transform
GLUT	OpenGL Utility Toolkit
GOP	Group Of Pictures
H.261	ITU:n videoneuvottelustandardi
H.263	ITU:n videoneuvottelustandardi
HDTV	High Definition Television
ITU	International Telecommunications Union
JPEG	Joint Photographic Experts Group
MAE	Mean Absolute Error

MJPEG	Motion JPEG
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
MSI	Multispectral Image
MSD	Mean Spectral Distance
MSV	Multispectral Video
NASA	National Aeronautics and Space Administration
NTSC	National Television System Committee
OpenGL	Open Graphics Library
PCA	Principal Component Analysis
PSNR	Peak Signal-to-Noise Ratio
RGB	Red, Green, Blue
RLE	Run-Length Encoding
SNR	Signal-to-Noise Ratio
VLC	Variable Length Code

$\mathbb{R}$	reaalilukujen joukko
$\mathbb{Z}$	kokonaislukujen joukko

Luku 1

Johdanto

Digitaalisten värikuvien käyttö eri sovelluksissa on nykyisin merkittävä tietojenkäsittelyn osa-alue. Kuvanlukijoiden avulla digitaaliseen muotoon muunnettujen kuvien lisäksi suuri osa kuvista generoidaan suoraan digitaalisten havaintolaitteiden avulla tai synteettisesti tietokoneella.

Yleisin digitaalisen kuvan esitysmuoto on RGB-kuva, jossa kunkin kuvapisteen väri ilmaistaan kolmella komponenttiarvolla: punaisella, vihreällä ja sinisellä. Monissa erikoissovelluksissa kuten kaukokartoituksessa käytetään RGB-esitystä tarkempia monikomponenttikuvia, joissa jokaisen kuvapisteen väriä kuvaa kohteesta heijastunut diskreettinä mitattu värispektri. Tällaisen spektrikuvan komponenttiarvojen lukumäärä voi olla jopa monikymmenkertainen RGB-kuvaan verrattuna.

Digitaalista kuvainformaatiota on helppoa muokata ja käsitellä monipuolisesti. Informaation tallennus ja siirto vaativat kuitenkin tavallisesti huomattavan suuria resursseja. Ongelma korostuu videokuvan tapauksessa, jolloin yhteen videoesitykseen voi kuulua satoja tai tuhansia kuvia. Yleensä kuvat sisältävät runsaasti päällekkäistä ja toistuvaa informaatiota, jota poistamalla voidaan vähentää niiden käsittelyssä tarvittavaa kapasiteettia.

Tämän työn tavoitteena oli tutkia värispektreistä koostuvien spektrikuvasarjojen tiivistämistä ja esittämistä videona. Luvussa 2 esitellään kuvainformaation tiivistyksen yleisiä periaatteita ja joitakin perusmenetelmiä, jotka voidaan yleisesti jakaa tiedon säilyttäviin ja tietoa hukkaaviin menetelmiin. Luku 3 käsittelee värispektrien ja spektrikuvien ominaisuuksia, ja kuvailee yleisesitystä tarkemmin eräitä spektrien tiivistykseen työssä käytettyjä menetelmiä. Videokuvan tunnusomaisia piirteitä ja sen tiivistykseen oleellisesti kuuluvan liikkeen kompensoinnin tekniikkaa käsitellään luvussa 4.

Käytännön toteutusta ja tuloksia on esitelty luvuissa 5 ja 6. Luku 7 sisältää tulosten perusteella tehdyt johtopäätökset sekä pohdintaa työn mahdollisen jatkokehityksen pohjaksi. Liitteenä on julkaistu esimerkkikuvia testeissä käytetystä kuvamateriaalista menetelmäkohtaisten tulosten visuaalista arviointia varten.

Luku 2

Kuvainformaation tiivistys

2.1 Peruslähtökohdat

Digitaalinen kuva muodostuu tietokoneen muistissa tavallisesti kuvapisteiden eli pikseleiden väriarvojen taulukosta. Analoginen kuva voidaan muuntaa digitaaliseen muotoon A/D-muunnoksen avulla, jolloin tuloksena saadaan valitun resoluution mukainen äärellisen kokoisten pikseleiden joukko. Kuvat voidaan luokitella kuvapisteen tallennukseen käytettävien bittien määrän mukaan binäärikuviin (1 bitti/pikseli), harmaasävykuviin (8 bittiä/pikseli) ja varsinaisiin värikuviin ($\geq$ 16 bittiä/pikseli). [14, 18]

Digitaalisessa muodossa esitettävä kuvainformaatio ja erityisesti videokuva, jonka koko voi olla tiivistämättömänä satoja tai jopa tuhansia megatavuja, vaatii huomattavan suurta kapasiteettia sekä tallennuksessa että tiedonsiirrossa. Kuvainformaation tiivistämiselle on yleensä tiettyjä käytännön rajoituksia kuten kuvanlaadun säilyminen mahdollisimman hyvänä, toteutuksen suhteellinen yksinkertaisuus kustannustehokkuuden kannalta sekä reaaliaikajärjestelmissä riittävä nopeus. Tiivistyksellä pyritään määrätyissä rajoissa vähentämään tallennustilan ja kaistanleveyden vaatimuksia laskenta-ajan kustannuksella. [16, 17]

Kuvissa esiintyvä toistuva ja päällekkäinen informaatio, **redundanssi**, voidaan yleisesti jakaa kolmeen ryhmään: [14]

1. Vierekkäisten kuvapisteiden korrelaatiosta johtuva spatiaalinen redundanssi.
2. Kuvapisteen värikomponenttitasojen korrelaatiosta johtuva spektraalinen redundanssi.
3. Videokuvan peräkkäisten kuvien korrelaatiosta johtuva temporaalinen redundanssi.

Kuvan tiivistys pyrkii vähentämään mainittua redundanssia, minkä tuloksena kuvainformaatio vie muistissa yleensä huomattavasti alkuperäistä vähemmän tilaa. Tiivistyksen tehokkuutta mitataan tavallisesti **tiivistyssuhteella** (*compression ratio*, CR): [14]

$$\mathrm{CR} = \frac{\text{bittien määrä alkuperäisessä kuvassa}}{\text{bittien määrä tiivistetyssä kuvassa}} \tag{2.1}$$

2.2 Tiivistyksen lajit

Yleisesti tiedon tiivistykseen käytettävät menetelmät voidaan luokitella kolmeen pääryhmään: [14, 16, 17]

1. **Entropiakoodaus** (*entropy coding*) on tiedon säilyttävä tiivistystapa (*lossless compression*). Tietoa ei katoa tiivistysprosessissa, ja alkuperäinen tieto voidaan tarvittaessa palauttaa eli rekonstruoida täydellisesti, mikä on välttämätöntä esimerkiksi tekstidokumenttien ja numeerisen informaation tallennuksessa. Jononpituuskoodaus ja Huffman-koodaus ovat tyypillisiä esimerkkejä tällaisesta tiivistämisestä, jossa dataa käsitellään vain yksinkertaisena digitaalisena sekvenssinä. Tiivistyssuhde on melko vaatimaton; yleensä 2:1 - 8:1 informaation sisältämän redundanssin määrästä ja tiivistysalgoritmin tehokkuudesta riippuen.

2. **Lähderiippuva koodaus** (*source coding*) on tietoa hukkaava tiivistystapa (*lossy compression*), jossa osa informaatiosta häviää tiivistyksen aikana pysyvästi. Tiivistystehokkuus riippuu tiedon sisällöstä ja ominaisuuksista, jotka pyritään ottamaan koodausvaiheessa huomioon. Tiivistyssuhde voi olla jopa 100:1. Menetelmä soveltuu hyvin esimerkiksi kuvan ja äänen tiivistykseen, jolloin pieniä yksityiskohtia tai häiriöitä ei normaalioloissa voitaisi muutenkaan havaita. Esimerkkejä lähderiippuvassa koodauksessa käytetyistä menetelmistä ovat muunnokset kuten pääkomponenttianalyysi ja aallokkeet.

3. **Hybridikoodauksessa** (*hybrid coding*) voidaan käyttää sekä tiedon säilyttävän entropiakoodauksen että tietoa hukkaavan lähderiippuvan koodauksen menetelmiä. Tällaista yhdistelmätekniikkaa sovelletaan monissa videokoodausstandardeissa.

Tietoa hukkaamattoman tiivistyksen tehokkuutta rajoittaa käsiteltävän tiedon entropia. Entropia on suuri, kun datan esiintymistodennäköisyys on pieni. Vastaavasti suurella todennäköisyydellä esiintyvän datan entropia on vähäinen. Käytännössä entropia riippuu sen estimointiin käytetyn mallin tarkkuudesta. Tällöin tiedon koodaamiseen tarvitaan sitä vähemmän bittejä, mitä tarkemmin se voidaan ennustaa. [17]

Tietoa hukkaavan tiivistyksen aikana syntyviä eroja ja laadun huononemista alkuperäiseen informaatioon verrattuna voidaan mitata subjektiivisilla tai objektiivisilla menetelmillä. Subjektiivisessa menetelmässä valitut testihenkilöt arvioivat esimerkiksi kuvan tiivistysprosessin jälkeen rekonstruoituja kuvia silmämääräisesti. Objektiivisessa arvioinnissa alkuperäisen ja rekonstruoidun informaation välillä lasketaan määrätyt tunnusluvut kuten keskivirhe ja signaalikohinasuhde. Objektiiviset tunnusluvut ovat yksikäsitteisiä ja vertailukelpoisia eri tiivistysmenetelmien kesken, mutta esimerkiksi kuvan tiivistyksessä ne eivät ota huomioon ihmisen näköaistin ominaispiirteitä tai heikkouksia. [3]

2.3 Tiivistyksen vaiheet

Yksittäis- ja videokuvan tiivistyksessä yhden kuvan prosessointi voidaan jakaa tyypillisesti seuraaviin vaiheisiin (kuva 2.1): [16]

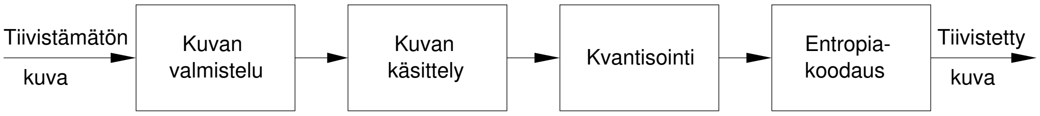
1. Valmisteluvaiheessa kuva muunnetaan tarvittaessa sopivaan digitaaliseen muotoon.

2. Varsinaisessa käsittelyvaiheessa sovelletaan tehokkaita muunnoksia ja esimerkiksi videokuvan tapauksessa liikkeen estimointialgoritmeja. Usein ku-

va käsitellään vakiokokoisissa lohkoissa.

3. Kvantisointivaiheessa edellisen vaiheen tuloksena saadut reaaliset lukuarvot kvantisoidaan kokonaislukutarkkuuteen halutulle välille. Kvantisoinnin suuruus voi vaihdella lukuarvojen merkityksen ja tärkeyden mukaan.

4. Viimeiseen vaiheeseen sisältyy tavallisesti entropiakoodaus, jossa aiempien vaiheiden tuottama informaatio tiivistetään ja tallennetaan ilman tiedon häviämistä.



Kuva 2.1: Kuvan tiivistyksen vaiheet. [16]

Videokuvan tiivistyksen tuloksena saadaan tietyn syntaksin mukainen datavuo, joka tavallisesti sisältää varsinaisen kuvainformaation lisäksi erilaisia tunniste- ja tarkistustietoja.

Purkaminen eli dekoodaus on tiivistyksen käänteisoperaatio, jossa dekooderi rekonstruoi tiivistetyn datan perusteella alkuperäisen tai sitä vastaavan kuvan. **Symmetrisessä tiivistyksessä** koodaus ja dekoodaus vievät suunnilleen saman verran aikaa, mikä on esimerkiksi reaaliaikaisten videoneuvottelujärjestelmien perusedellytys. **Asymmetrisessä tiivistyksessä** dekoodausvaihe on tavallisesti nopeampi ja vaatii vähemmän tehoa kuin koodausvaihe. Asymmetrisyys soveltuu esimerkiksi multimediasovelluksiin, joissa koodaus täytyy suorittaa vain tuotteen valmistusvaiheessa mutta dekoodaus useita kertoja ja nopeasti käytön aikana. Tällöin tehokkaan kooderin rakenteen ja toiminnan monimutkaisuus voi olla moninkertainen dekooderiin verrattuna. [10, 16]

2.4 Perusmenetelmiä

2.4.1 Jononpituuskoodaus

Yksittäiskuvien ja kokonaisten videosekvenssien data sisältää usein toistuvia saman tavun jonoja. Monissa tapauksissa dataa voidaan tällöin tiivistää olennaisesti tallentamalla toistuva tavu ja jonon pituus tavalla, jota kutsutaan jononpituuskoodaukseksi (*run-length encoding*, RLE). Tavujonon alku ilmoitetaan jollakin erityisellä merkinnällä tai merkillä, joka ei esiinny varsinaisen datan osana, ja tämän jälkeen kerrotaan toistettava merkki ja sen lukumäärä (kuva 2.2). Jos jonon alkua merkitään tietyllä yksittäisellä merkillä (esimerkiksi ‘<Esc>’), tarkoittaa kaksi peräkkäistä erikoismerkkiä (‘<Esc><Esc>’) datassa sitä, että merkki esiintyy kyseisessä kohdassa sellaisenaan. [16, 17]

```
**************************abcdefg
              ⇓
       <Esc>*26abcdefg
```

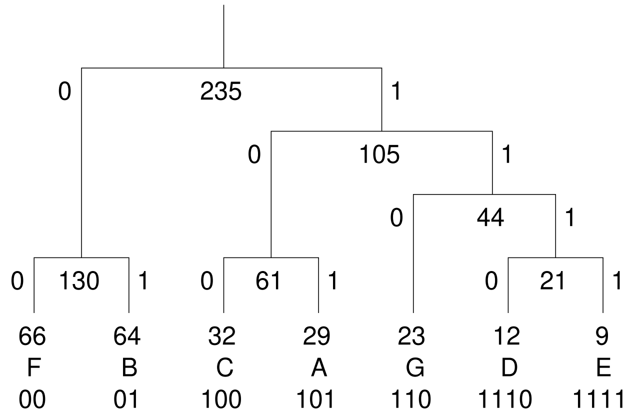
Kuva 2.2: Esimerkki jononpituuskoodauksesta. [17]

Jononpituuskoodauksen erikoistapaus on nollien poisto (*zero suppression*), jossa vain yhden merkin oletetaan esiintyvän tiheästi datavirrassa. Tällöin kolmen tai useamman nollan jono korvataan valitulla erikoismerkillä ja nollien lukumäärällä, joten parhaassa tapauksessa 258 tavun pituinen jono voidaan tallentaa kahdella tavulla. [16]

2.4.2 Huffman-koodaus

Huffman-koodaus on tavupohjaista jononpituuskoodausta kehittyneempi bittitasolla tehtävän entropiakoodauksen menetelmä, jossa jokaiselle datavirrassa esiintyvälle symbolille etsitään vaihtelevanmittainen tallennuskoodi (*variable length code*, VLC). Tavoitteena on saavuttaa hyvä tiivistyssuhde siten, että useimmin esiintyville symboleille käytetään mahdollisimman lyhyitä koodimerkintöjä. [17]

Menetelmässä symbolit järjestetään niiden esiintymistiheyden tai siihen verrannollisen esiintymistodennäköisyyden mukaan. Tämän jälkeen muodostetaan ns. Huffman-puu siten, että joka kierroksella kaksi pienimmän todennäköisyyden symbolia tai haaraa yhdistetään, ja lasketaan niiden todennäköisyyksien summa. Tätä jatketaan niin kauan, kunnes kaikki symbolit on yhdistetty puumaiseksi rakenteeksi. Seuraavaksi jokaista valmiin puun haaraa merkitään bitillä 0 tai 1 yhdenmukaisella tavalla - esimerkiksi vasenta haaraa bitillä 0 ja oikeaa haaraa bitillä 1 - ja kuljetaan kustakin symbolista lähtien puun juureen, jolloin kyseisen symbolin tallennuskoodi muodostuu reitin varrella olevista biteistä käänteisessä järjestyksessä (kuva 2.3). [17]



Kuva 2.3: Esimerkki Huffman-koodauksesta, kun datavirran sisältämien symbolien `A-G` esiintymistiheydet tiedetään. [17]

Tuloksena saadut tallennuskoodit ovat yksikäsitteisiä, eli mikään koodi ei voi esiintyä toisen koodin alkuosana. Dekoodausvaiheessa on käytettävä samaa kooditaulukkoa, joka näin ollen täytyy tallentaa tiivistetyn datan kanssa. Menetelmän haittapuolena symbolien esiintymistiheydet on selvitettävä ennen koodausta. Vaihtoehtoisesti voidaan soveltaa adaptiivista Huffman-koodausta, jossa puurakenne kehittyy ja mukautuu datan prosessoinnin aikana. [17]

2.4.3 Muunnoskoodaus

Muunnoskoodauksessa käsiteltävälle datalle tehdään jokin matemaattinen, yleensä lineaarinen muunnos. Käänteisessä muunnoksessa data palautetaan alkuperäisen esityksen mukaiseksi. Perinteisiä kuvan tiivistyksessä käytettyjä muunnoksia ovat Fourier-muunnokseen perustuva diskreetti kosinimuunnos (*discrete cosine transform*, DCT) ja nopea Fourier-muunnos (*fast Fourier transform*, FFT) [4, 20]. Muunnosten laskentakompleksisuudesta johtuen kuva jaetaan yleensä lohkoihin, jotka käsitellään kukin erikseen. [16]

Muunnoksen lähtötietona on yleisesti datavektori

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathbf{T}}, \tag{2.2}$$

jonka arvojen x_i välinen korrelaatio on merkittävä. Soveltamalla vektoriin $\mathbf{x}$ sopivaa lineaarista muunnosta

$$\mathbf{y} = \mathbf{T}\mathbf{x}, \tag{2.3}$$

missä $\mathbf{T}$ on $n \times n$ -kokoinen muunnosmatriisi, saadaan tuloksena samanpituinen vektori $\mathbf{y}$. Vektorin arvot $y_1, y_2, \ldots, y_n$ ovat ns. muunnoskertoimia, jotka yleensä korreloivat keskenään huomattavasti vähemmän kuin alkuperäiset arvot. Lisäksi informaatio keskittyy tavallisesti suppeaan joukkoon muunnoskertoimia. Alkuperäinen vektori voidaan palauttaa käänteisellä muunnoksella. [4]

Muunnos on yksiulotteinen, jos se kohdistuu yhdessä ulottuvuudessa n-komponenttiseen vektoriin kuten kuvan riviin tai sarakkeeseen. Kaksiulotteinen muunnos kohdistuu yleensä tasossa $n \times n$ -kokoiseen lohkoon ja kolmiulotteinen muunnos $n \times n \times n$ -kokoiseen kuutioon. Useat moniulotteiset muunnokset voidaan jakaa erillisiin yksiulotteisiin vaiheisiin. [14]

Yleisiä tehokkaiden muunnosten tavoiteltavia ominaisuuksia ovat: [14]

1. Tuloksena saadut muunnoskertoimet ovat mahdollisimman korreloimattomia, ja informaatio keskittyy harvoihin kertoimiin.

2. Muunnoksen kantafunktiot, jotka muodostavat muunnosmatriisin arvot, ovat kuvainformaatiosta riippumattomia.

3. Muunnos voidaan toteuttaa käytännössä nopeasti.

Tiivistyksen saavuttaminen

Normaalisti muunnos sellaisenaan ei johda tiedon tiivistymiseen, vaan tavoitteena on saada aikaan korreloimattomia muunnoskertoimia siten, että mahdollisimman suuri osa informaatiosta on keskittynyt harvoihin kertoimiin. Tällöin monet kertoimista ovat arvoltaan hyvin pieniä, ja ne voidaan kvantisoida ja koodata jotain entropiakoodauksen menetelmää käyttäen. Suuriarvoisella kvantisoijalla tehtävä karkea kvantisointi parantaa tiivistystehoa mutta voi samalla huonontaa kuvanlaatua. [14]

Koodattavien muunnoskertoimien valintaan on useita mahdollisuuksia. Yksinkertaisessa tapauksessa säilytetään vain määrätyllä alueella sijaitsevat muunnoskertoimet ja nollataan muut, minkä jälkeen jäljellejääneet arvot kvantisoidaan ja tallennetaan. Kvantisoijan suuruus voi olla vakio kaikille arvoille tai vaihdella kertoimen paikan ja merkityksen mukaan. [14]

Alueellisen valinnan sijasta tallennettavat kertoimet voidaan myös valita asetetun kynnysarvon tai lukumäärän perusteella. Tällöin vältytään virheeltä, joka syntyy, jos rajatun alueen ulkopuolelle jää informaatioarvoltaan merkittäviä kertoimia. Haittapuolena on kuitenkin se, että säilytettävistä arvoista täytyy tallentaa myös sijaintitiedot. [14]

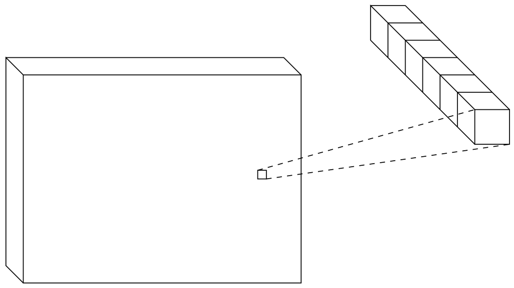
Luku 3

Spektrikuvan tiivistys

3.1 Spektrikuvan luonne

Tietojenkäsittelyn sovelluksissa tärkein digitaalisen kuvan esitysmuoto on RGB-kuva, jossa kuvapisteen väri määritetään kolmella komponenttiarvolla: punaisella, vihreällä ja sinisellä. Tietyissä tilanteissa tarvitaan kuitenkin värin tarkempaa komponenttiesitystä, joka saadaan mittaamalla kuvattavasta kohteesta heijastuneen sähkömagneettisen säteilyn energiajakauma määrätyltä aallonpituusväliltä. Tuloksena saadussa spektrikuvassa jokaisen kuvapisteen värin määrittää kiinteänmittainen värispektri. Tällöin kuvan spektrien saman tason komponentit eli kanavat muodostavat spektritasoja, joita voidaan haluttaessa tarkastella erillisinä (kuva 3.1). [15]

Spektrikuva soveltuu hyvin tarkoituksiin, joissa mitattavat aallonpituudet sijaitsevat kokonaan tai osittain näkyvän valon aallonpituusvälin ulkopuolella. Kuvasta



Kuva 3.1: Spektrikuvan rakenne. [15]

voidaan myös tarvittaessa muodostaa RGB-esitys visualisointia varten. Eräs tärkeimpiä spektrikuvien sovelluskohteita on kaukokartoitus, josta tyypillisiä käytännön esimerkkejä ovat NASA:n julkaisemat satelliitti- ja lentokuvat [1].

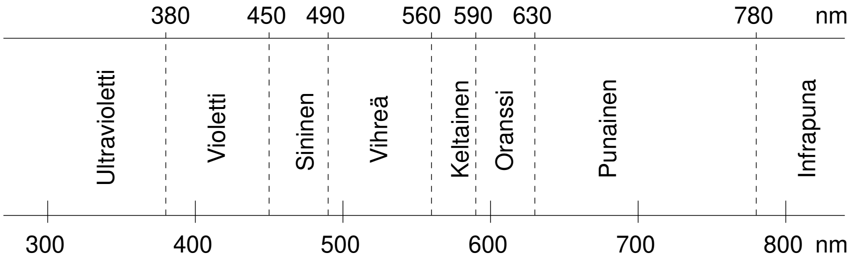
3.2 Spektrin ominaisuudet ja väriesitys

3.2.1 Näkyvän valon spektri

Näkyvä valo on sähkömagneettisen säteilyn kapea taajuusalue eli spektri välillä $4.3 \cdot 10^{14} - 7.5 \cdot 10^{14}$ Hz. Tällä alueella jokainen taajuus vastaa tiettyä väriä. Taajuuden f, aallonpituuden λ ja valon nopeuden c (tyhjiössä $c = 3 \cdot 10^8$ m/s) välisestä riippuvuudesta [1]

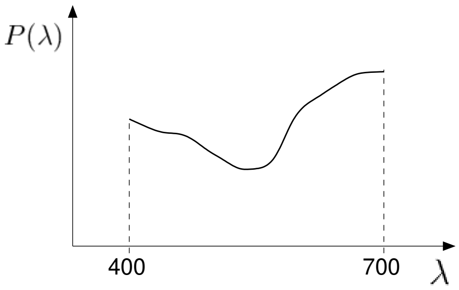
$$\lambda f = c \quad \Leftrightarrow \quad \lambda = \frac{c}{f} \tag{3.1}$$

voidaan laskea kyseisiä taajuuksia vastaavat aallonpituudet (kuva 3.2).



Kuva 3.2: Näkyvän valon spektrin tärkeimmät värikaistat. [6]

Valonlähteen tuottaman valon energiajakauma $P(\lambda)$ kuvaa säteilyn energiaa valon kunkin aallonpituuden λ kohdalla (kuva 3.3). [2, 5]



Kuva 3.3: Esimerkki näkyvän valon energiajakaumasta.

Erilaisia energiajakaumia on periaatteessa ääretön määrä - käytännössä jakauma mitataan tavallisesti diskreettinä äärelliselle määrälle aallonpituuksia. Väriaistimus muodostuu silmään tulevasta näkyvän valon alueella olevan kohteesta heijastuneen säteilyn spektristä. Kohteen väriin vaikuttavat sekä pinnan että valonlähteen ominaisuudet, ja useat erilaiset spektrin energiajakaumat voivat näyttää silmälle samalta väriltä. [2, 15]

3.2.2 Väriaistimus ja RGB-värimalli

Väriaistimuksen trikromaattisuusteorian mukaan ihmissilmä havainnoi herkimmin punaista, vihreää ja sinistä valoa. Teoriaa tukevien kokeellisten tulosten perusteella havainnointivasteen huiput osuvat aallonpituuksille 440 nm (“sininen”), 545 nm (“vihreä”) ja 580 nm (“punainen”). Vastemittausten mukaan silmän herkkyys siniselle valolle on huomattavasti heikompi kuin vihreälle tai punaiselle valolle, ja vaste vakiovoimakkuuden valolle on huipussaan 550 nm:n aallonpituudella. [2, 19]

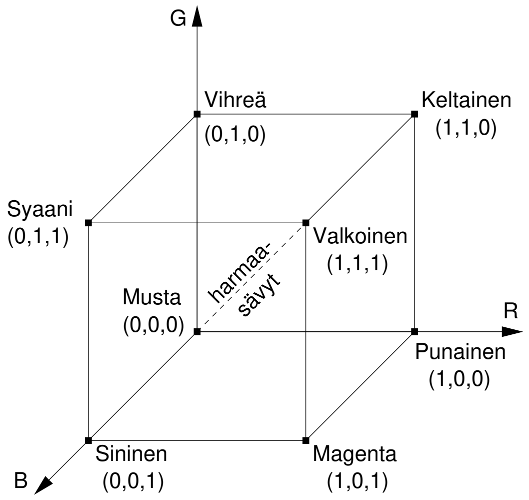
Väriaistimusteoriaan perustuvan RGB-värimallin mukaan tietty väri C_λ voidaan määrittää kolmen päävärin, punaisen (*red*, R), vihreän (*green*, G) ja sinisen (*blue*, B), painotettuna summana

$$C_\lambda = r_\lambda \mathrm{R} + g_\lambda \mathrm{G} + b_\lambda \mathrm{B}, \tag{3.2}$$

missä päävärien osuutta kuvaavat kertoimet r_λ, g_λ ja b_λ saadaan kokeellisesti määritellyistä värinsovitusfunktioista kaikille näkyvän valon väreille aallonpituusvälillä 360-830 nm, kun valon voimakkuus on vakio [2]. Näin muodostuvan RGB-mallin väriasteikko esitetään tavallisesti yksikkökuutiona (kuva 3.4).

3.2.3 XYZ-värimalli

Kansainvälinen valaistuskomissio CIE (*Comission Internationale de l’Éclairage*) on kehittänyt värien täsmällistä määritystä varten kolme keinotekoista pääväriä, jotka tunnetaan nimillä X, Y ja Z. Vastaavien värinsovitusfunktioiden $\bar{x}_\lambda$, $\bar{y}_\lambda$ ja $\bar{z}_\lambda$



Kuva 3.4: RGB-mallin väriasteikko. [5]

avulla voidaan mikä tahansa väri esittää näiden kolmen standardivärin yhdistelmänä. [2, 19]

Kun värispektrin energiajakauma on $P(\lambda)$, päävärien X, Y ja Z tarvittavat kertoimet ovat [2]

$$x_\lambda = k \int P(\lambda)\, \bar{x}_\lambda \, d\lambda, \quad y_\lambda = k \int P(\lambda)\, \bar{y}_\lambda \, d\lambda, \quad z_\lambda = k \int P(\lambda)\, \bar{z}_\lambda \, d\lambda, \tag{3.3}$$

missä vakio k on [2]

$$k = \frac{100}{\int P_w(\lambda)\, \bar{y}_\lambda \, d\lambda}, \tag{3.4}$$

kun $P_w(\lambda)$ on valitun standardivalonlähteen mukaisen valkoisen valon spektrin energiajakauma. Integroinnit suoritetaan tavallisesti yhteenlaskuina, koska mitatut energiajakaumat ovat diskreettejä [2]. Tarkkojen väriarvojen laskemiseen tarvitaan 1 nm:n välein mitattu spektri, mutta käytännön sovelluksissa 5 nm:n mittaustarkkuus välillä 380-780 nm on usein riittävä. Lisäksi ihmissilmä havaitsee käytännössä vain välillä 400-700 nm olevia aallonpituuksia [11].

Standardipäävärien avulla väri C_λ voidaan määrittää summana

$$C_\lambda = x_\lambda \mathrm{X} + y_\lambda \mathrm{Y} + z_\lambda \mathrm{Z}. \tag{3.5}$$

Koska tietty väri saadaan sekä RGB- että XYZ-mallissa päävärien lineaarisena yhdistelmänä, muunnos XYZ $\rightarrow$ RGB on vastaavasti myös lineaarinen.

3.3 Spektrimuunnokset

3.3.1 Pääkomponenttianalyysi

Pääkomponenttianalyysi (*principal component analysis*, PCA) on yleinen diskreetin tilastollisen aineiston analysointimenetelmä, joka perustuu datan kovarianssimatriisista laskettaviin ominaisvektoreihin. Menetelmässä lähtötietona olevista datavektoreista $\mathbf{x}$ muodostetaan kovarianssimatriisi [11]

$$\mathbf{C} = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^{\mathbf{T}}], \tag{3.6}$$

missä n-komponenttisten datavektoreiden $\mathbf{x}$ muodostaman vektorijoukon matemaattinen odotusarvo $\mu = E[\mathbf{x}]$. Matriisin $\mathbf{C}$ ominaisarvot ovat λ_i $(i = 1, \ldots, n)$, joille pätee

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0, \tag{3.7}$$

ja vastaavat ominaisvektorit ovat $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$. Tällöin $\lambda_1 = \sigma^2$ on ensimmäisen pääkomponentin varianssi, ja $\mathbf{u}_1$ osoittaa suurimman varianssin suunnan. [11]

Kovarianssimatriisille voidaan laskea estimaatti

$$\hat{\mathbf{C}} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^{\mathbf{T}}, \tag{3.8}$$

missä $\hat{\mu}$ on datavektoreiden estimoitu odotusarvo. Vaihtoehtoisesti voidaan käyttää myös autokorrelaatiomatriisia

$$\mathbf{R} = E[\mathbf{x}\mathbf{x}^{\mathbf{T}}] \qquad (\mathbf{C} = \mathbf{R} - \mu\mu^{\mathbf{T}}) \tag{3.9}$$

tai sen laskennallista estimaattia

$$\hat{\mathbf{R}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^{\mathbf{T}}. \tag{3.10}$$

Koska matriisit $\mathbf{C}$ ja $\mathbf{R}$ ovat symmetrisiä, niiden ominaisvektorit muodostavat ortogonaalisen vektorijoukon [11]. Tällöin voidaan suuruusjärjestykseen lajiteltujen ominaisarvojen perusteella valita ominaisvektorit, jotka määrittävät aliavaruuden kannan. Datan koostuessa spektreistä pyritään tallennettavien ominaisvektorien määrä ja aliavaruuden kannan koko valitsemaan niin, että kanta säilyttää mahdollisimman paljon alkuperäisten spektrivektorien informaatiosta. Tämä voidaan toteuttaa valitsemalla ominaisvektorit suurimpien ominaisarvojen mukaan, ja sovelluksesta riippuen jo muutama ominaisvektori voi olla tarpeeksi riittävän tarkan väriesityksen tuottamiseksi [12, 13].

Pääkomponenttianalyysissa suoritettava PCA-muunnos on spektrien energian säilymisen kannalta optimaalinen: kun jonkin muunnoksen tuloksena saatavista kertoimista säilytetään vain tietty osa, se sisältää PCA-muunnoksessa enemmän energiaa kuin missään muussa muunnoksessa. Menetelmän huonona puolena on laskennallinen hitaus ja kantafunktioiden riippuvuus käsiteltävästä kuvasta, mikä edellyttää kovarianssimatriisin laskemista kuvakohtaisesti. [14]

3.3.2 Aallokkeet

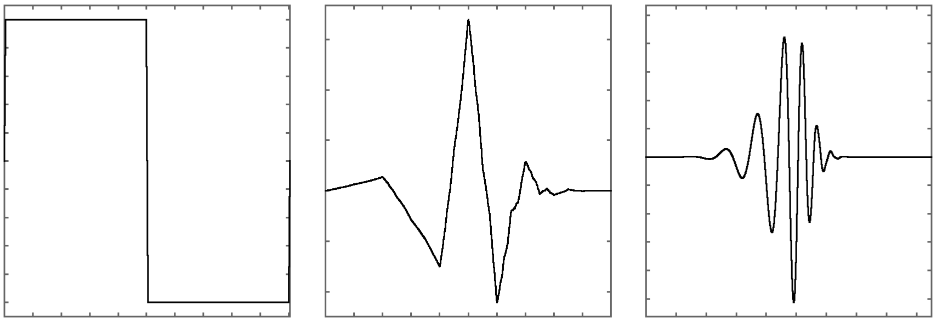
Aalloke (*wavelet*) on äärellisen pituinen, usein epäsäännöllinen ja epäsymmetrinen aaltomuoto, jonka funktion Ψ keskiarvo on nolla ajan t suhteen: [8]

$$\int_{-\infty}^{+\infty} \Psi(t)\, dt = 0 \tag{3.11}$$

Aallokefunktioista yksinkertaisin on paloittain määritelty Haarin aalloke [8]

$$\Psi(t) = \begin{cases} 1 & \text{kun } 0 \leq t < \frac{1}{2} \\ -1 & \text{kun } \frac{1}{2} \leq t < 1 \\ 0 & \text{muuten} \end{cases} \tag{3.12}$$

joka tunnetaan myös osana Daubechies-aallokeperhettä (kuva 3.5). Daubechies-aallokkeiden soveltamisesta värispektrien tiivistämiseen on tutkimuksissa saavutettu myönteisiä tuloksia [7].



Kuva 3.5: Daubechies-aallokkeita: 2. asteen Daubechies (Haar), 4. asteen Daubechies ja 20. asteen Daubechies. [9]

Jatkuva muunnos

Jatkuvassa aallokemuunnoksessa (*continuous wavelet transform*, CWT) käsiteltävä signaali muunnetaan aallokefunktion Ψ skaalattujen ja aika-akselilla siirrettyjen versioiden summaksi [8, 20]

$$W[f](s,u) = (f, \Psi_{s,u}) = \int_{-\infty}^{+\infty} f(t)\Psi_{s,u}(t)\,dt, \tag{3.13}$$

missä

$$\Psi_{s,u}(t) = \frac{1}{\sqrt{|s|}} \Psi\left(\frac{t-u}{s}\right) \tag{3.14}$$

on ns. emoaalloke, $f(t)$ on signaalifunktio, ja skaalaus- ja siirtymäparametrit $s, u \in \mathbb{R},\ s \neq 0$. Tuloksena saatavat muunnoskertoimet ovat skaalauksen ja siirtymän funktioita. Käänteismuunnoksessa aallokkeen siirrettyjen ja skaalattujen versioiden yhdistelmänä voidaan rekonstruoida alkuperäinen signaali. Prosessissa syntyvän ylimääräisen redundanssin vuoksi jatkuva muunnos on sellaisenaan sopimaton tiivistämiseen, ja sitä käytetäänkin pääasiassa signaalien analysoinnissa esimerkiksi epäjatkuvuuskohtien etsimiseen.

Diskreetti muunnos

Diskreetti aallokemuunnos (*discrete wavelet transform*, DWT) on periaatteeltaan muuten vastaavanlainen kuin jatkuva muunnos, mutta skaalausten ja siirtymien mahdolliset arvot on rajattu kahden potensseihin, jolloin [8, 20]

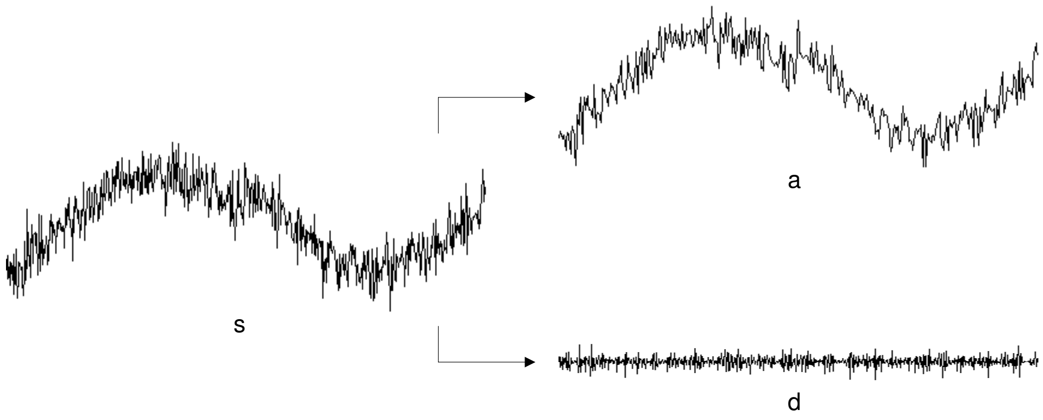
$$W[f](m,n) = (f, \Psi_{m,n}) = \int_{-\infty}^{+\infty} f(t)\Psi_{m,n}(t)\,dt, \tag{3.15}$$

missä

$$\Psi_{m,n}(t) = s_0^{-\frac{m}{2}}\Psi(s_0^{-m}t - nu_0), \tag{3.16}$$

kun $s_0 > 1$, $u_0 > 0$ ja $m, n \in \mathbb{Z}$. Yleensä käytetään arvoja $s_0 = 2$ ja $u_0 = 1$.

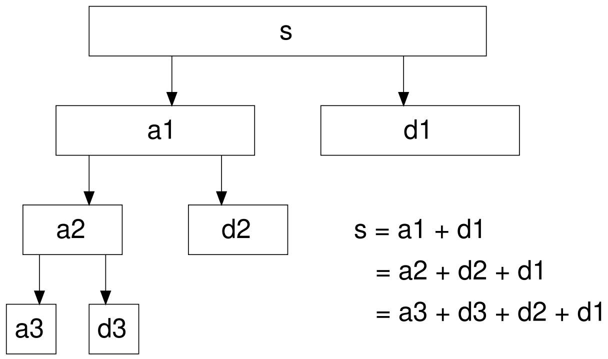
Diskreetti aallokemuunnos voidaan suorittaa tehokkaammin kuin jatkuva muunnos, ja signaali on rekonstruoitavissa käytännössä lähes yhtä tarkasti. Muunnoksen tuloksena saatavat kertoimet jakautuvat pienen ja suuren taajuuden kertoimiin, joiden molempien lukumäärä on prosessiin sisältyvän alinäytteistyksen tuloksena puolet alkuperäisen signaalin kertoimien määrästä. Pienen taajuudet kertoimet muodostavat signaalista suuren mittakaavan approksimaation, kun taas suuren taajuuden kertoimet sisältävät tietoa pienen mittakaavan yksityiskohdista (kuva 3.6). [9]



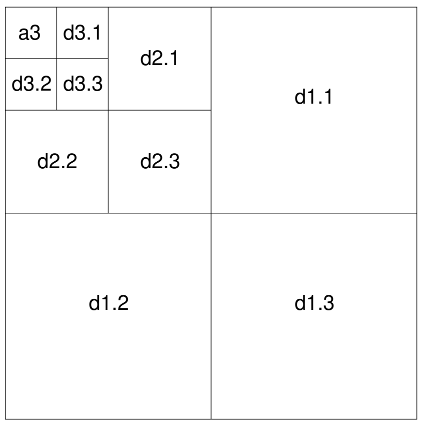
Kuva 3.6: Signaalin s jakautuminen approksimaatioon a ja yksityiskohtiin d yksitasoisessa diskreetissä aallokemuunnoksessa. [9]

Monitasoisessa muunnoksessa approksimaatiokertoimet toimivat syötteenä seuraavan tason muunnokselle, jolloin kertoimet muodostavat puumaisen hierarkian (kuva 3.7). Käänteisessä muunnoksessa eli synteesissä alkuperäinen signaali voidaan rekonstruoida approksimaatio- ja yksityiskohtakertoimien perusteella. [9]

Yksittäiskuvien tiivistyksessä muunnos voidaan tehdä kuvalohkoittain tai koko kuvalle kerralla, jolloin jälkimmäisessä tapauksessa vältytään diskreettiin kosinimuunnokseen perustuville menetelmille tyypillisiltä virheiltä lohkojen reunoilla. Muunnoksen jälkeen kertoimet voidaan kvantisoida ja koodata jollain entropiakoodauksen menetelmällä muunnostasoittain (kuva 3.8). [18]



Kuva 3.7: Signaalin s monitasoinen muunnos. [9]



Kuva 3.8: Kaksiulotteisen lohkon monitasoinen muunnos. [8]

Luku 4

Videokuvan tiivistys

4.1 Videokuvan ominaisuudet

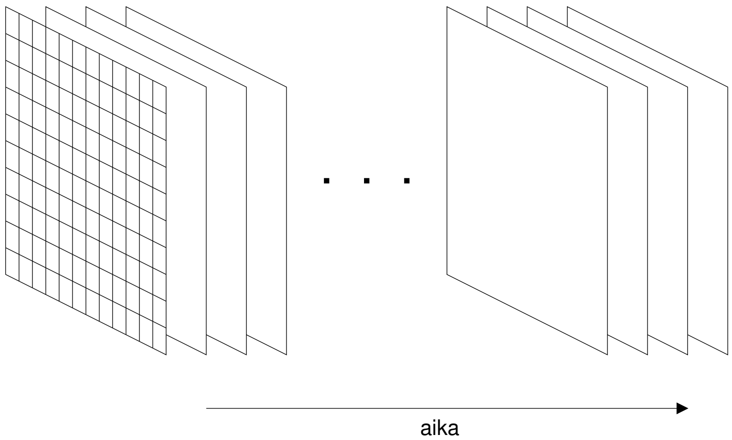
Videokuvan käsittely asettaa merkittäviä vaatimuksia järjestelmän kapasiteetille. Esimerkiksi RGB-muodossa esitettävä video, jonka resoluutio on 320×240 kuvapistettä ja kuvanopeus 30 kuvaa sekunnissa, tarvitsee tiivistämättömänä yli 6.9 miljoonaa tavua yhden sekunnin mittaisen näytteen siirtoon tai tallennukseen sellaisenaan. Tästä syystä useimmissa järjestelmissä käsitellään tiivistettyä videokuvaa. Taulukossa 4.1 on esitelty tällaisen videokuvan tyypillisiä nykysovelluksia ja niiden ominaisuuksia. [18]

Tavallinen videosekvenssi koostuu tasaisesti lyhyin aikavälein esiintyvistä yksittäisistä kuvista, jotka määrätyllä kuvanopeudella toistettaessa synnyttävät vaikutelman jatkuvasta liikkeestä. Yksinkertaisessa tapauksessa videosekvenssin jokainen kuva koodataan ns. intrakoodausmenetelmillä sekvenssin muista kuvista riip-

Taulukko 4.1: Tiivistetyn videokuvan sovelluksia. [18]

Sovellus	Siirtonopeus	Standardi	Resoluutio	Kuvanopeus (kuvaa/s)
Analoginen video-puhelin	5-10 Kbps	-	170×128	2-5
Videoneuvottelu, pieni siirtonopeus	26-64 Kbps	H.263	128×96 176×144	15-30
Digitaalinen videopuhelin	64-128 Kbps	H.261	176×144 352×288	10-20
Videoneuvottelu, suuri siirtonopeus	$\geq$ 384 Kbps	H.261	352×288	15-30
Vuorovaikutteinen multimedia	1-2 Mbps	MPEG-1	352×240	15-30
Digitaalinen televisio (NTSC)	3-10 Mbps	MPEG-2	720×480	30
HDTV	15-80 Mbps	MPEG-2	1200×800	30-60

pumatta, mistä MJPEG-tiivistys (*Motion JPEG*) on yleinen käytännön esimerkki. Tehokkaat videokoodaustekniikat kuten MPEG (*Moving Picture Experts Group*) poistavat myös peräkkäisten kuvien välistä temporaalista redundanssia. Kummassakin tapauksessa kuvat koodataan yleensä vakiokokoisiin alueisiin jaettuina (kuva 4.1). [10]



Kuva 4.1: Videosekvenssin rakenne. [10]

4.2 Liikkeen kompensointi

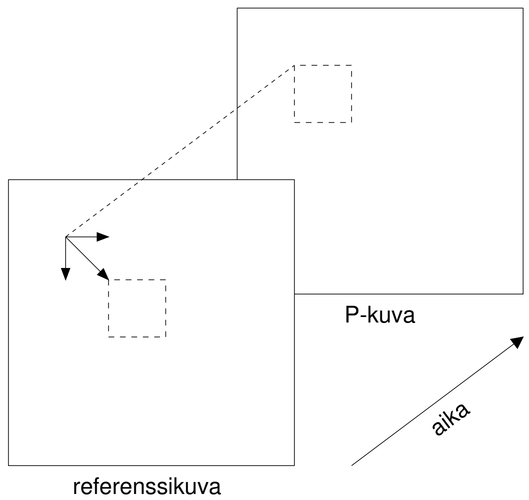
4.2.1 Yleisperiaate

Normaalin videosekvenssin kuvat sisältävät usein alueita, käytännön toteutuksissa tyypillisesti vakiokokoisia lohkoja, jotka toistuvat kuvasta toiseen joko sellaisenaan tai vain vähän muuttuneina. Yksinkertaisessa tapauksessa alue säilyy täysin muuttumattomana samassa paikassa kahden peräkkäisen kuvan välillä, jolloin referenssinä toimivan kuvan aluetta voidaan käyttää sellaisenaan seuraavassa kuvassa. Mahdolliset muutokset voidaan laskea erotuksena kuvien välillä ja tallentaa halutulla tarkkuudella. Tietoa hukkaavassa tiivistyksessä vähäiset muutokset voidaan jättää virherajojen puitteissa kokonaan huomiotta, mikä osaltaan parantaa tiivistystehokkuutta. [10]

Kun muutos aiheutuu alueen liikkumisesta kuvassa, voidaan liikettä kompensoida tallentamalla liikevektori, joka määrittää alueen siirtymän vaaka- ja pystysuunnassa peräkkäisten kuvien välillä. Koska muutoksia aiheuttavat kameran liikkeen lisäksi myös kuvan objektien liikkuminen ja valaistuksen muuttuminen, kullakin rajatulla alueella on tavallisesti oma liikevektorinsa. [10]

4.2.2 Kuvatyypit

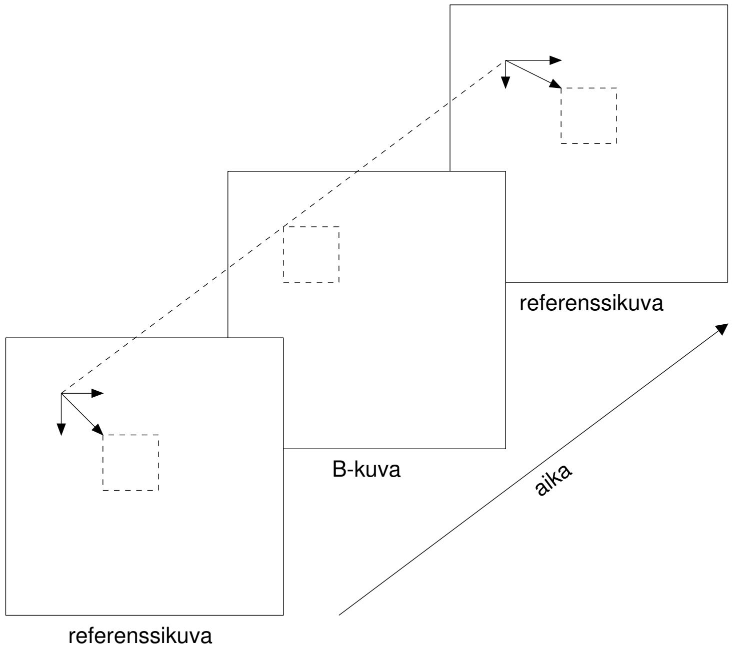
Perinteinen menetelmä liikkeen kompensointiin on jakaa videosekvenssi toistuviin ja määrämittaisiin I- ja P-kuvien ryhmiin, mitä käytetään esimerkiksi H.261-videoneuvottelustandardissa. Tällöin I-kuva (*intra-coded picture*) koodataan yksittäiskuvan tapaan ainoastaan kuvan sisäistä redundanssia hyödyntäen. P-kuva (*predictive-coded picture*) koodataan hyödyntäen kuvan lohkojen vastaavuuksia lähimmän edeltävän I- tai P-tyyppisen referenssikuvan kanssa, ja tarvittaessa tallennetaan vain lohkojen siirtymät ja erot referenssikuvaan verrattuna (kuva 4.2). [10]



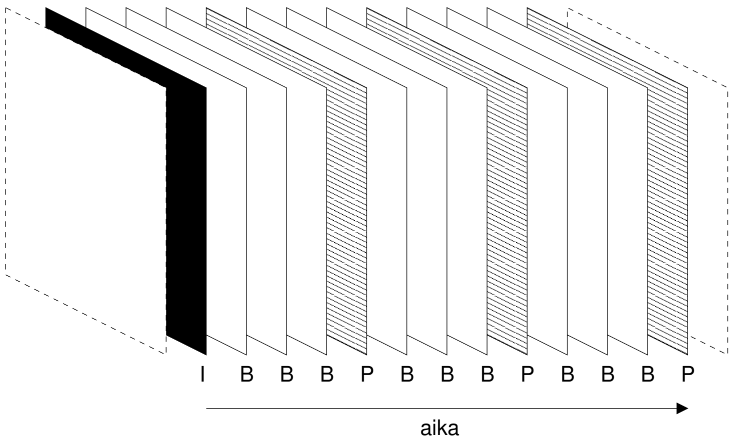
Kuva 4.2: Liikkeen kompensointi P-kuvassa. [10]

MPEG- ja H.263-standardeissa videosekvenssin kuva voi olla tyypiltään myös B-kuva (*bidirectionally predictive-coded picture*), jolloin kuvan lohko määräytyy lähimmän edeltävän ja seuraavan I- tai P-tyyppisen referenssikuvan perusteella (kuva 4.3). Tällaisessa videossa P- tai B-kuvan yksittäinen lohko voidaan tarvittaessa koodata myös muun tyyppisenä - B-kuvassa P- tai I-tyyppisenä lohkona ja P-kuvassa I-tyyppisenä lohkona - jos referenssikuvista ei löydetä tyydyttävää vastaavuutta. [10, 16]

Eri tavoin koodattuja kuvia sisältävä videosekvenssi voidaan jakaa hierarkisesti kiinteänmittaisiin kuvaryhmiin (*group of pictures*, GOP), jotka muodostuvat määrätyssä järjestyksessä esiintyvistä I-, P- ja B-tyyppisistä kuvista. Valinnaisesti tällainen kuvaryhmä voi olla suljettu kokonaisuus, jolloin ryhmän ulkopuolisia kuvia ei käytetä referenssikuvina (kuva 4.4). [10, 16]



Kuva 4.3: Liikkeen kompensointi B-kuvassa. [10]



Kuva 4.4: MPEG-videosekvenssin tyypillinen kuvaryhmä. [10]

4.3 Liikkeen estimointi

4.3.1 Lähtökohdat

Liikkeen estimointi on koodausvaiheen prosessi, jossa määritetään kuvalohkojen siirtymien suuruus. Kooderi etsii käsiteltävälle lohkolle referenssikuvasta tarpeeksi hyvää vastinetta ja sellaisen löytyessä tallentaa videosekvenssiin liikevektorien arvot, jotka dekooderi tulkitsee ja käyttää kuvan rekonstruoinnissa. Kaikkien mahdollisten siirtymäarvojen testaus takaa parhaan lopputuloksen mutta ei aina ole laskenta-ajan kannalta järkevää. Liikkeen estimointi onkin usein laskennallisesti koodausvaiheen raskain osa ja suurimpia kooderin ja dekooderin asymmetrisyyteen vaikuttavia tekijöitä. [10]

Kooderia toteutettaessa liikkeen estimointiin käytettävät algoritmit voidaan usein valita vapaasti, kunhan tuloksena saadut siirtymäarvot ovat asetettujen rajojen sisällä ja ne tallennetaan videoformaatin standardissa määritellyllä tavalla. Lohkopohjaisessa liikkeen koodauksessa tavoitteena on tavallisesti minimoida käsiteltävän kuvan lohkon ja referenssikuvan lohkon välinen eroavaisuus tai maksimoida niiden välinen korrelaatio. Tällöin lohkokohtainen liikevektori pätee kyseisen lohkon kaikille kuvapisteille. [10]

4.3.2 Vertailutekniikat

Keskimääräinen virhe

Yksinkertaisin ja yleisin tapa $m \times n$ -kokoisten lohkojen vertailuun on laskea koodattavan kuvan lohkon kuvapisteiden $V_p(x+i, y+j)$ ja referenssikuvan lohkon kuvapisteiden $V_r(x+dx+i, y+dy+j)$ välinen keskimääräinen virhe (*mean absolute error*) [10]

$$\mathrm{MAE} = \frac{1}{m \times n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |V_p(x+i, y+j) - V_r(x+dx+i, y+dy+j)|, \quad (4.1)$$

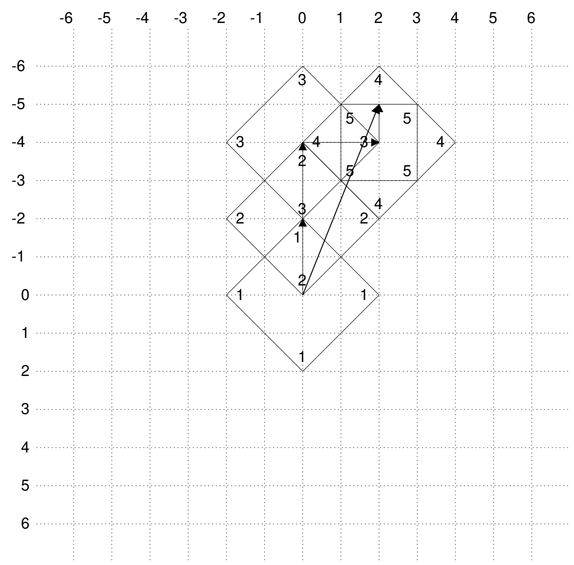
missä x ja y ovat lohkon vasemman yläkulman koordinaatit, i ja j ovat kuvapisteiden koordinaatit lohkon sisällä ja dx ja dy ovat siirtymäarvot. Vaihtoehtoisesti voidaan vertailussa käyttää keskimääräistä neliövirhettä (*mean square error*, MSE), jolloin kahden kuvapisteen välinen erotus korotetaan neliöön ennen yhteenlaskua.

Täydellinen haku

Täydellisessä haussa (*full search*) jokaista lohkoa kohti testataan kaikki mahdolliset siirtymäarvot annettujen rajojen sisällä ja lasketaan niiden mukaan lohkojen välinen keskimääräinen virhe. Koska täydellisen haun laskentakompleksisuus on verrannollinen suurimman hakualueen sivun pituuden neliöön, siirtymäarvoille täytyy asettaa kohtuulliset rajat. Käytännössä ± 16 kuvapisteen hakuetäisyys on todettu tavallisten videosekvenssien koodauksessa useimmiten riittäväksi. Hakualueen koosta riippumatta liikkeen estimoinnissa ei aina löydetä tyydyttävää vastinetta käsiteltävälle kuvalohkolle, jolloin kyseinen lohko täytyy koodata intrakoodausmenetelmiä soveltaen. [10]

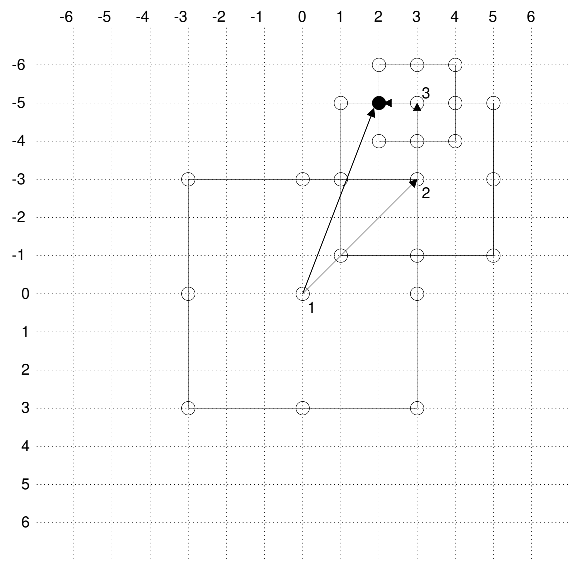
Vaihtoehtoisia hakumenetelmiä

Koska täydellinen haku on laskennallisesti vaativa operaatio, liikkeen estimointia varten on kehitetty lukuisia nopeampia mutta samalla epätäydellisempiä algoritmeja. **Kaksiulotteinen logaritminen haku** (*2-D logarithmic search*) vertailee viittä mahdollista siirtymää kahden kuvapisteen välein (kuva 4.5). Jos paras vastine sijaitsee hakualuetta rajoittavan suorakulmion jossakin kulmapisteessä, siirretään hakualueen keskipiste kyseiseen pisteeseen ja toistetaan haku (kuvassa vaiheet 1-4). Kun paras vastine saadaan nykyisen hakualueen keskipisteessä, tehdään viimeinen hakukierros 3×3 kuvapisteen kokoisen alueen sisällä (kuvassa vaihe 5). Viimeisen kierroksen perusteella valitaan parhaan vertailutuloksen antava siirtymä, mikäli kuvalohkojen erot ovat tällöin sallituissa rajoissa. [10]



Kuva 4.5: Kaksiulotteinen logaritminen haku. [10]

Kolmivaiheinen haku (*three-step search*) vertailee siirtymiä hakualuetta rajoittavan neliön keskipisteessä, kulmapisteissä sekä sivujen keskipisteissä (kuva 4.6). Ensimmäisen hakukierroksen jälkeen hakualueen keskipiste siirretään aina edellisellä kierroksella parhaan vertailutuloksen tuottaneeseen pisteeseen, ja samalla alueen rajoja pienennetään yhden kuvapisteen verran joka suunnassa (kuvassa vaiheet 1-3). [10]



Kuva 4.6: Kolmivaiheinen haku. [10]

Useimmissa vaihtoehtoisista algoritmeista liikkeen oletetaan muodostuvan ortogonaalisista siirtymistä. Todellisuudessa luonnollisen videokuvan sisältämään liikkeeseen kuuluu myös kiertoa ja skaalausta, mutta näiden huomioonottaminen kasvattaa liikkeen estimointiprosessin monimutkaisuutta merkittävästi. Yksinkertaisten siirtymien käsittelyä pidetään yleensä riittävänä, jos kuvan lohkot ovat tarpeeksi pieniä. [10]

Luku 5

Työn toteutus

Menetelmien testausta varten työssä ohjelmoitiin Unix-ympäristössä toimivat C-kieliset komentorivipohjaiset ohjelmat spektrivideon koodaukseen ja dekoodaukseen. Dekooderiin sisältyvä yksinkertainen käyttöliittymä videokuvan katseluun RGB-näytöllä toteutettiin OpenGL- ja GLUT-funktiokirjastojen ohjelmointirajapintoja hyödyntäen.

5.1 Kooderi

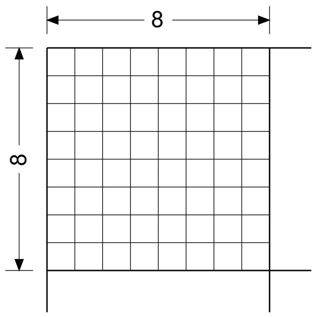
5.1.1 Yleinen toiminta

Kooderin syötetiedostot ovat tiivistämättömiä spektrikuvia, jotka on tallennettu yksinkertaistetussa MSI-muodossa (*multispectral image*, [15]). MSI-tiedoston otsaketiedoissa ilmoitetaan kokonaislukutarkkuudella kuvan leveys ja korkeus, spektrin kanavien lukumäärä sekä mitatun aallonpituusvälin alku ja loppu. Spektrikanavien arvot tallennetaan tiedostoon tavuina, joiden arvot ovat välillä 0-255 (kuva 5.1).

```
multispectral_image() {
  string(“MSI\n”);
  string(“width height channels start end\n”);
  channel_values(height × width × channels);
}
```

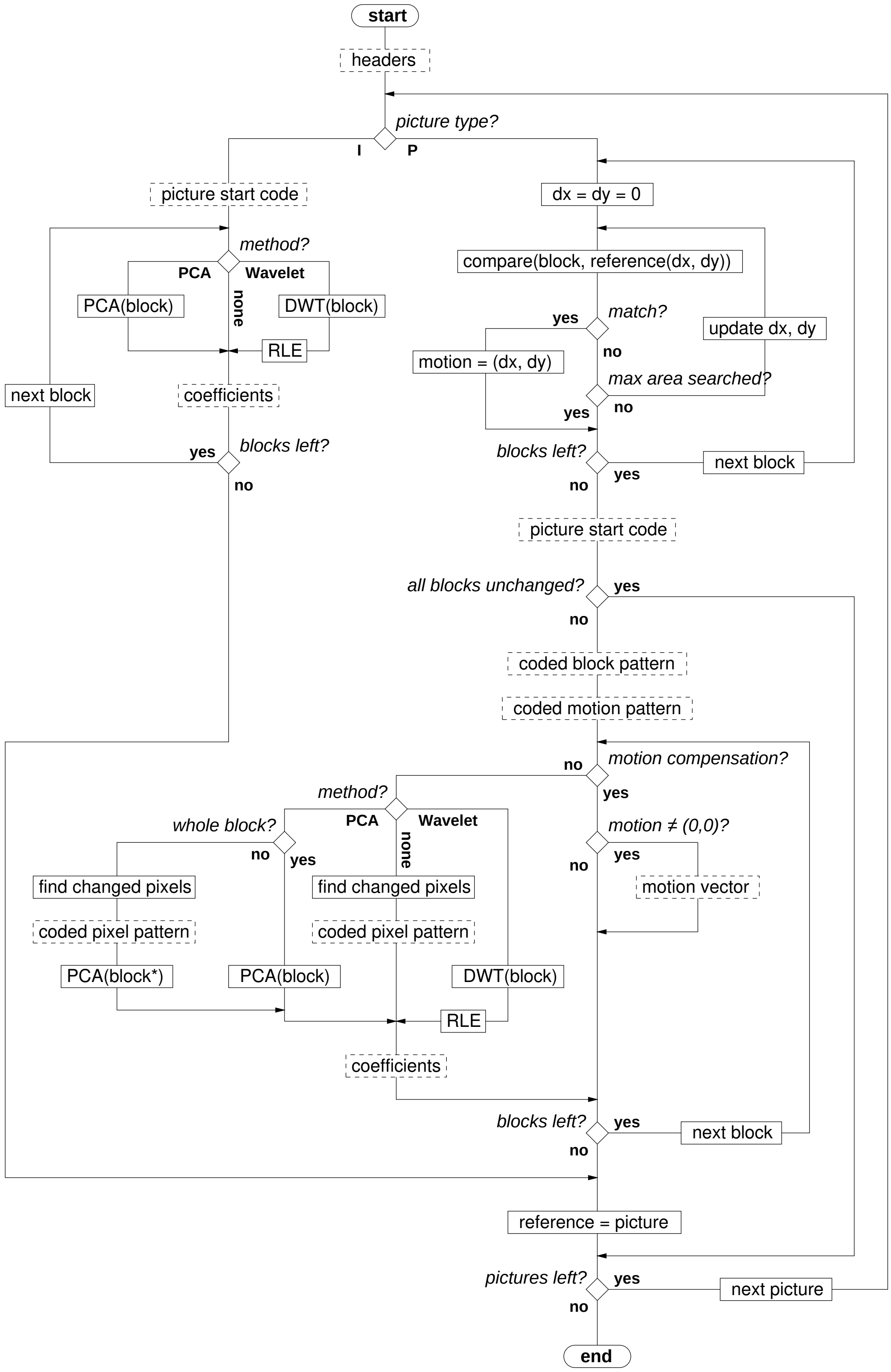
Kuva 5.1: Proseduuri `multispectral_image()`.

Ohjelman komentoriviparametreilla voidaan ilmoittaa syöte- ja tulostiedostot sekä koodaukseen käytettävä menetelmä mahdollisine parametreineen. Kaikissa tapauksissa kuvat käsitellään vasemmalta oikealle ja ylhäältä alas lohkoittain, joiden koko on 8×8 kuvapistettä (kuva 5.2).



Kuva 5.2: Kuvan koodauksen perusyksikkö.

Kuvasarjojen lisäksi kooderia voidaan käyttää myös yksittäisten MSI-muotoisten spektrikuvien tiivistämiseen, jolloin videosekvenssin tulkitaan koostuvan yhdestä kuvasta. Tällöin tiivistysprosessista jää luonnollisesti liikkeen estimoinnin suoritusvaihe kokonaan pois. Kuvassa 5.3 on esitetty kooderin toiminta-algoritmi vuokaavion muodossa.

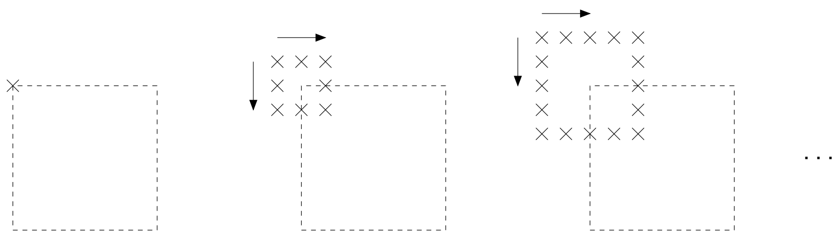


Kuva 5.3: Kooderin toiminta-algoritmi.

5.1.2 Kuvien välinen koodaus

Menetelmästä riippumatta ensimmäinen vaihe kuvalohkon koodauksessa on liikkeen estimointi, joka suoritetaan sarjan kaikille kuville ensimmäistä lukuunottamatta. Prosessissa kullekin käsiteltävän kuvan lohkolle etsitään vastinetta edeltävästä referenssikuvasta maksimihakuetäisyyden rajoissa.

Käytettyä hakutekniikkaa on havainnollistettu kuvassa 5.4, jossa katkoviivalla merkitylle kuvalohkolle etsitään vastinetta referenssikuvassa testattavan lohkon rastilla merkityn vasemman yläkulman mukaan vasemmalta oikealle ja ylhäältä alas. Hakuetäisyys kasvaa joka kierroksella yhden kuvapisteen verran vaaka- ja pystysuunnassa. Jos lohkolle löytyy hyväksyttävä vastine hakurajojen sisällä, tallennetaan kyseisestä lohkosta vain liikevektori, jota käytetään liikkeen kompensoinnissa. Oletuksena etsintä tapahtuu maksimissaan ± 15 kuvapisteen etäisyydellä alkuperäisestä sijainnista, mikä vastaa esimerkiksi H.261-videokoodausstandardin käytäntöä [10].



Kuva 5.4: Täydellisen haun vaiheet.

Ohjelmassa vastinelohkon testaus on toteutettu vertaamalla aallonpituuksia, joilla silmän herkkyys punaisen, vihreän ja sinisen värin astimukselle on trikromaattisuusperiaatteen mukaisesti huipussaan. Jos verrattavien lohkojen kaikkien kuvapisteiden spektreillä näiden kanava-arvojen erotukset ovat sallittujen virherajojen sisällä, hyväksytään kyseinen referenssikuvan lohko vastineeksi.

Videosekvenssin aloittava I-kuva koodataan itsenäisesti sarjan muista kuvista riippumatta. Seuraavissa P-kuvissa liikkeen referenssinä toimii aina sarjan edellinen I- tai P-kuva. Tällöin videotiedostoon tallennetaan kuvakohtainen lohkomaski (*coded block pattern*, CBP), jonka bitit ilmaisevat edeltävään referenssikuvaan verrattuna muuttuneet lohkot. Jos kuvan kaikki lohkot säilyvät ennallaan samassa paikassa, tallennetaan vain merkintä muuttumattomasta kuvasta, jolloin dekoodausvaiheessa toistetaan sama kuva uudelleen.

5.1.3 Kuvan sisäinen koodaus

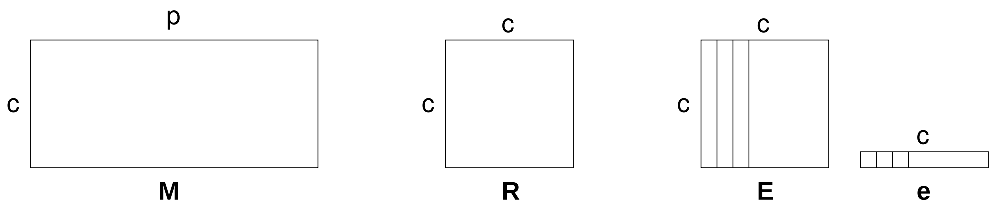
Jos lohkolle ei liikkeen estimoinnissa löydetä tyydyttävää vastinetta tai kyseessä on I-kuva, koodataan menetelmästä ja kuvatyypistä riippuen lohkon kaikkien tai ainoastaan referenssikuvassa samassa paikassa sijaitsevaan lohkoon verrattuna muuttuneiden kuvapisteiden spektrit - maksimissaan 64 kappaletta - valitulla kuvan sisäisen koodauksen menetelmällä. Jälkimmäisessä tapauksessa videotiedostoon tallennetaan lohkokohtainen pikselimaski (*coded pixel pattern*, CPP), jonka bitit ilmaisevat lohkon sisällä muuttuneet ja edelleen koodatut kuvapisteet.

Pääkomponenttianalyysi

Pääkomponenttianalyysissa I-kuvan jokaiselle lohkolle ja P-kuvan muuttuneille lohkoille tehdään PCA-muunnos. Lähtötilanteessa lohkon kaikki tai vain muuttuneet spektrit järjestetään kaksiulotteiseen matriisiin M. Tämän jälkeen muodostetaan autokorrelaatiomatriisi

$$\mathbf{R} = \mathbf{M} \cdot \mathbf{M}^{\mathbf{T}}, \tag{5.1}$$

jolle lasketaan ominaisvektorit $\mathbf{E}$ ja ominaisarvot $\mathbf{e}$ (kuva 5.5).

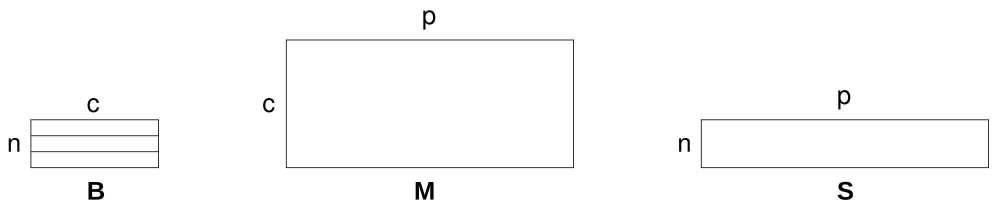


Kuva 5.5: Koodattavan lohkon spektreistä muodostetulle matriisille $\mathbf{M}$ lasketaan autokorrelaatiomatriisi $\mathbf{R}$, jolle etsitään ominaisvektorit $\mathbf{E}$ ja ominaisarvot $\mathbf{e}$. Kuvassa c on spektrin komponenttien lukumäärä ja p on spektrien lukumäärä.

Määrätyn pääkomponenttien lukumäärän n mukaisesti suurimpien ominaisarvojen perusteella valituista ominaisvektoreista muodostetaan n-ulotteisen aliavaruuden kanta $\mathbf{B}$, ja lasketaan sitä vastaavat uudet kertoimet

$$\mathbf{S} = \mathbf{B} \cdot \mathbf{M}, \tag{5.2}$$

jotka kvantisoidaan ja tallennetaan kannan $\mathbf{B}$ kanssa (kuva 5.6).



Kuva 5.6: Kertomalla matriisi $\mathbf{M}$ suurimpia ominaisarvoja vastaavista ominaisvektoreista muodostetulla kannalla $\mathbf{B}$ saadaan uuden aliavaruuden mukaiset kertoimet $\mathbf{S}$.

Dekoodausvaiheessa rekonstruoidaan matriisi $\mathbf{M}'$ laskemalla kannan käänteismatriisin $\mathbf{B}^{-1}$ ja kertoimien $\mathbf{S}$ tulo (kuva 5.7):

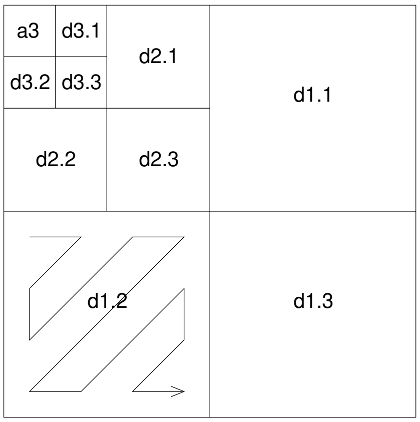
$$\mathbf{M}' = \mathbf{B}^{-1} \cdot \mathbf{S} \tag{5.3}$$

Kuva 5.7: Kannan käänteismatriisin $\mathbf{B}^{-1}$ ja kerroinmatriisin $\mathbf{S}$ tulosta saadaan rekonstruoitu matriisi $\mathbf{M}'$.

Aallokkeet

Aallokkeita käyttävässä tiivistysmenetelmässä kuvan jokaiselle muuttuneelle lohkolle kokonaisuudessaan tehdään yksiulotteisten muunnosten yhdistelmänä kolmiulotteinen diskreetti aallokemuunnos. Tuloksena saadut kertoimet kvantisoidaan määrätyllä kvantisoijan arvolla ja tallennetaan muunnostasoittain siksak-järjestyksessä jononpituuskoodattuna (kuva 5.8).

Koska diskreetissä muunnoksessa käsiteltävän datavektorin pituuden tulee olla kahdella jaollinen jokaista muunnostasoa kohti, tämä voi spektrin pituudesta riippuen aiheuttaa ongelmia spektrisuunnassa. Vaihtoehtona tällaisessa tilanteessa on esimerkiksi spektrivektorin jatkaminen joko lineaarisesti viimeisten arvojen perusteella tai nolla-arvoja käyttäen. Työssä ongelma ratkaistiin niin, että koodausvaiheessa tehdään muunnos spektrisuunnassa niin monta kertaa kuin käsiteltävän vektorin pituus täyttää edellä mainitun ehdon - kuitenkin enintään kaksiulotteisen



Kuva 5.8: Aallokemuunnoskertoimien koodaus siksak-järjestyksessä. [8]

muunnoksen maksimikierrosten verran eli 3 muunnostasoa, mitä pidetään yleensä käytännön suosituksena. Dekoodausvaiheessa lohko rekonstruoidaan suorittamalla vastaavat käänteisoperaatiot.

5.2 Dekooderi

Toteutettu dekooderi purkaa kooderin tuottaman tiivistetyn videosekvenssin ja valinnan mukaan esittää sen näytöllä RGB-muodossa tai kirjoittaa rekonstruoidut kuvat erillisiksi MSI-tiedostoiksi. Vaihtoehtoisesti ohjelmaa voidaan käyttää myös tiivistämättömien MSI-kuvien katseluun yksittäin tai kuvasarjana. Esitystä varten spektrien perusteella lasketaan ensin kuvapisteiden XYZ-arvot CIE:n värinsovitusfunktioiden ja valitun standardivalonlähteen mukaisten parametriarvojen avulla, ja tämän jälkeen tehdään lineaarinen muunnos XYZ $\rightarrow$ RGB.

Videokuvan katseluun vaikuttavia parametreja kuten suurennuskerrointa ja kuvanopeutta voidaan säätää komentoriviparametrien lisäksi esityksen aikana valikkopohjaisen käyttöliittymän tai näppäinkomentojen avulla. Normaalin RGB-esityksen lisäksi voidaan myös tarkastella vain tietyn aallonpituuden spektritasoa harmaasävyesityksenä.

Oletuksena dekooderi rekonstruoi videosekvenssin kaikki kuvat ja tallentaa ne RGB-muodossa muistiin ennen katselua. Ohjelmassa on myös optio koodatun videotiedoston lukemiseen ja kuvien rekonstruointiin vain tarpeen mukaan esityksen aikana, mutta RGB-muunnoksen ja kuvan tiivistykseen käytettyjen menetelmien käänteisoperaatioiden vaatimasta laskenta-ajasta johtuen kuvanopeus jää dekooderin nykyisessä versiossa huonommaksi kuin etukäteen muistiin tallennettua videoesitystä katseltaessa.

5.3 Videoformaatin syntaksi

Kooderin tulostiedosto on tiivistetty videosekvenssi MSV-formaatissa (*multispectral video*), jonka syntaksi on esitetty MPEG-standardissa käytetyllä C-kieleen pohjautuvalla pseudokielellä kuvissa 5.9 - 5.12.

Tiedoston alussa sijaitsevat tekstimuotoiset otsikkotiedot kuten tunnistemerkkijono ja yleisten videoparametrien arvot. Otsikkotietoja seuraa tavumuodossa koodattu kuvadata, jonka käsittely määräytyy kuvan tyypin perusteella (kuvat 5.9 ja 5.10).

Sekvenssin aloittavan I-kuvan kaikki lohkot on koodattu itsenäisesti tiettyä kuvan sisäisen tiivistyksen menetelmää käyttäen. Lähinnä testausmielessä lohkojen arvot voidaan myös tallentaa sellaisenaan yksittäisen MSI-kuvan tapaan, jos koodausvaiheessa ei käytetä mitään erityistä menetelmää sisäiseen tiivistykseen (kuva 5.11).

```
video_sequence() {
  sequence_header();
  while (picture_start_code()) {
    if (picture_start_code == R) { /* repeat previous */
      continue;
    }
    if (picture_start_code == I) {
      intra_picture();
    }
    if (picture_start_code == P) {
      predicted_picture();
    }
  }
}
```

Kuva 5.9: Proseduuri `video_sequence()`.

```
sequence_header() {
  string(“MSV\n”);
  string(“width height channels start end\n”);
  string(“picture_coding_type ”);
  if (picture_coding_type == PCA) {
    string(“principal_components ”);
  }
  if (picture_coding_type == WAVELET) {
    string(“wavelet_type quantizer_value ”);
  }
  string(“frame_rate\n”);
}
```

Kuva 5.10: Proseduuri `sequence_header()`.

P-kuvien rekonstruointiin kuuluu keskeisenä osana liikkeen kompensointi. Videotiedostossa kuvan alkuun on tallennettu edelliseen kuvaan verrattuna muuttuneiden ja otsikkotiedoissa ilmoitetulla tavalla koodattujen lohkojen maski (CBP). Menetelmästä riippuen koodatun lohkon alussa voi olla lisäksi koodattujen kuvapisteiden maski (CPP), jolloin ainoastaan lohkon muuttuneiden kuvapisteiden spektrit on tallennettu. Jos ns. liikemaskin (*coded motion pattern*, CMP) nojalla edellisestä kuvasta periytyvän vastinelohkon siirtymäarvot ovat nollasta poik-

```
intra_picture() {
  forearch (block) {
    if (picture_coding_type == NONE) {
      channel_values(BLOCKSIZE_SQ × channels);
    }
    if ((picture_coding_type == PCA) ||
        (picture_coding_type == PCA2)) {
      base_matrix(principal_components × channels);
      score_matrix(principal_components × BLOCKSIZE_SQ);
    }
    if (picture_coding_type == WAVELET) {
      rle_vector();
    }
  }
}
```

Kuva 5.11: Proseduuri `intra_picture()`.

keavia, luetaan sekvenssistä myös lohkon suhteellista sijaintia referenssikuvassa ilmaisevan liikevektorin arvot (kuva 5.12).

```
predicted_picture() {
  coded_block_pattern();
  coded_motion_pattern();
  forearch (block) {
    if (coded_block_pattern[block]) {
      if (picture_coding_type == NONE) {
        coded_pixel_pattern();
        forearch (pixel) {
          if (coded_pixel_pattern[pixel]) {
            channel_values(channels);
          }
        }
      }
      if (picture_coding_type == PCA) {
        coded_pixel_pattern();
        base_matrix(principal_components × channels);
        score_matrix(principal_components × pixels);
      }
      if (picture_coding_type == PCA2) {
        base_matrix(principal_components × channels);
        score_matrix(principal_components × BLOCKSIZE_SQ);
      }
      if (picture_coding_type == WAVELET) {
        rle_vector();
      }
    } else {
      if (coded_motion_pattern[block]) {
        motion_vector(); /* default (0,0) */
      }
    }
  }
}
```

Kuva 5.12: Proseduuri `predicted_picture()`.

Luku 6

Tulokset

6.1 Testiaineisto

Tiivistysmenetelmiä testattiin kahdella erityyppisellä spektrikuvasarjalla, joista on esitetty esimerkkikuvia liitteessä I. Lyhyempi kuvasarja muodostuu alunperin RGB-muotoisesta 60 kuvan tietokoneanimaatiosta, joka on muunnettu spektrikuviksi etsimällä kullekin kuvapisteelle sen RGB-arvoja vertailuarvoiltaan lähimpänä oleva Munsellin standardiväreistä mitattu spektri. Pidempi testiaineisto koostuu NASA:n ylläpitämän julkisen AVIRIS-arkiston materiaalista muodostetusta 218 spektrikuvan ilmakuvasarjasta. Taulukossa 6.1 on vertailtu eräitä testikuvasarjojen ominaisuuksia.

Kaikissa testeissä videokoodausvaiheeseen sisältyi peräkkäisten kuvien välinen liikkeen estimointi maksimihakuetäisyyden rajoissa sekä tämän jälkeen parametrina annetun menetelmän mukainen kuvan sisäinen koodaus muuttuneille kuva-

Taulukko 6.1: Testiaineiston ominaisuuksien vertailu.

	Kuvasarja 1	Kuvasarja 2
tyyppi	tietokoneanimaatio	ilmakuvasarja
värit	kirkkaita, hyvin erilaisia	tummia, samansävyisiä
liike	suurta, epäsäännöllistä	säännöllistä, yhdensuuntaista
resoluutio	320×224	352×288
kuvia	60	218
spektrikanavia	10	11
aallonpituusväli	420-672 nm	400-700 nm
tiivistämätön koko	41.02 MB	231.85 MB

lohkoille kokonaisuudessaan tai vain muuttuneiden kuvapisteiden spektrien osalta.

6.2 Testitulokset

6.2.1 Vertailun tunnusluvut

Tiivistystulosten numeerista arviointia varten kunkin testin tuloksena saadusta MSV-videosekvenssistä rekonstruoidut kuvat kirjoitettiin dekooderilla MSI-muodossa erillisiksi spektrikuviksi. Tämän jälkeen kuvasarjan muodostavien alkuperäisten MSI-kuvien ja rekonstruoitujen kuvien välillä laskettiin tarvittavat vertailuarvot, joista johdettiin sekvenssikohtaiset tunnusluvut.

Tiivistysmenetelmän tehokkuutta mitattiin laskemalla alkuperäisten MSI-kuvien ja koodatun MSV-videon tiedostokoon perusteella tiivistyssuhde CR (sivu 8, kaava 2.1). Virheen arviointiin käytettiin perustunnuslukuina absoluuttista keskivirhettä (*mean absolute error*)

$$\mathrm{MAE} = \frac{1}{p \times c} \sum_{i=1}^{p \times c} |o_i - r_i| \tag{6.1}$$

ja keskimääräistä neliövirhettä (*mean square error*)

$$\mathrm{MSE} = \frac{1}{p \times c} \sum_{i=1}^{p \times c} (o_i - r_i)^2, \tag{6.2}$$

joissa p on kuvapisteiden lukumäärä, c on spektrin kanavien lukumäärä, o_i on alkuperäinen kanava-arvo ja r_i on rekonstruoitu kanava-arvo.

Kuvapistekohtaisen virheen huippuarvoa kuvaa maksimivirhe

$$\mathrm{MAX} = \max \sum_{i=1}^{c} |o_i - r_i|. \tag{6.3}$$

Spektreihin tiivistysprosessissa syntyvää virhettä mitattiin erityisesti laskemalla kuvien keskimääräinen spektrietäisyys (*mean spectral distance*)

$$\mathrm{MSD} = \frac{1}{p} \sum_{i=1}^{p} \sqrt{\sum_{j=1}^{c} (o_j - r_j)^2}. \tag{6.4}$$

Lisäksi videosekvenssille laskettiin signaalikohinasuhde (*signal-to-noise ratio*)

$$\text{SNR} = 10\log_{10}\frac{\frac{1}{p\times c}\sum_{i=1}^{p\times c} o_i^2}{\text{MSE}} \tag{6.5}$$

ja vastaava huippuarvo (*peak signal-to-noise ratio*)

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\text{MSE}}. \tag{6.6}$$

6.2.2 Yleiset parametrit

Kuvien välisessä liikkeen estimoinnissa käytettiin kaikissa testeissä vastinelohkon maksimihakuetäisyytenä ± 15 kuvapistettä vaaka- ja pystysuunnassa. Vertailuaallonpituuksien arvojen maksimierotuksen vaikutusta kokonaisten lohkojen ja yksittäisten spektrien vertailussa tutkittiin maksimierotuksen eri arvoilla ilman spatiaalisten tiivistysmenetelmien käyttöä, ja testauksen tulokset on listattu taulukoissa 6.2 ja 6.3.

Taulukko 6.2: Kuvasarja 1, maksimierotuksen *diff* vertailu.

diff	CR	MAX	MAE	MSE	MSD	SNR	PSNR
5	1.46	428	0.51	38.50	0.22	45.22	50.75
10	1.60	428	6.88	546.63	2.66	33.67	39.20
15	1.90	513	21.10	2120.72	8.00	28.01	33.54
20	2.12	513	32.53	3870.88	12.23	25.44	30.96
25	2.27	555	39.44	5138.90	14.82	24.16	29.68

Molempien testiaineistojen kohdalla käytettiin varsinaisissa menetelmätesteissä

Taulukko 6.3: Kuvasarja 2, maksimierotuksen *diff* vertailu.

diff	CR	MAX	MAE	MSE	MSD	SNR	PSNR
2	40.74	72	4.30	72.31	1.59	29.50	49.00
4	53.55	94	11.71	329.92	4.08	23.23	42.73
6	68.21	121	20.65	800.52	7.05	19.54	39.04
8	85.57	138	28.85	1363.41	9.77	17.26	36.76
10	103.30	153	37.04	2124.02	12.46	15.41	34.91

PSNR-arvoon nojautuen pienintä maksimierotuksen arvoa, jotta kuvan sisäisten tiivistysmenetelmien vaikutus tunnuslukuihin olisi selvemmin havaittavissa. Ilmakuvasarjassa spektrien tasaisuudesta ja koko kuvan alueella hyvin lähellä toisiaan olevista värisävyistä johtuen käytännössä järkevän maksimierotuksen raja tulee vastaan huomattavasti nopeammin kuin kirkasvärisessä tietokoneanimaatiossa, jossa ei numeeristen tunnuslukujen huononemisesta huolimatta esiintynyt visuaalisesti kovin häiritseviä värivirheitä vielä suurimmallakaan testatulla maksimierotuksen arvolla.

6.2.3 Menetelmä 1: PCA-muunnos

Ensimmäisessä menetelmässä koodattavien kuvien muuttuneille lohkoille tehdään pääkomponenttianalyysin mukainen PCA-muunnos. Visuaalista RGB-esitystä vertailtaessa 3-4 pääkomponentilla koodatun videokuvan erot alkuperäiseen kuvasarjaan verrattuna ovat molemmissa testikuvasarjoissa käytännössä mahdottomia havaita. Lyhyemmässä kuvasarjassa kahden pääkomponentin mukainen muunnos aiheuttaa lievää kuvan kirkkauden vaihtelua, mutta varsinaisia värivirheitä ei vielä esiinny. Yhteen pääkomponenttiin perustuvan muunnoksen tuloksena värit ovat jo selvästi alkuperäisiä haaleampia, kirkkaus vaihtelee tuntuvasti ja videokuvassa on monin paikoin havaittavissa värivirheitä ja ns. blokkiintumista lohkojen reunojen mukaan, mikä on myös numeeristen vertailuarvojen perusteella pääteltävissä (taulukko 6.4).

Taulukko 6.4: Kuvasarja 1, pääkomponenttien lukumäärän *pc* vertailu.

pc	CR	MAX	MAE	MSE	MSD	SNR	PSNR
4	2.98	837	16.65	361.17	6.55	34.99	40.51
3	3.92	871	22.74	843.09	8.82	31.53	37.05
2	5.76	1323	41.10	3259.56	15.90	25.62	31.14
1	10.82	1393	93.18	16834.57	34.42	18.92	24.44

Pidemmässä kuvasarjassa kuvanlaatu ja vertailun tunnusluvut säilyvät hyvinä pienelläkin pääkomponenttien lukumäärällä (taulukko 6.5). Tähän vaikuttavat osaltaan merkittävästi kuvissa tapahtuvan liikkeen säännöllisyys ja sisäisesti koodattavien muuttuneiden lohkojen pieni määrä lyhyempään kuvasarjaan verrattuna.

Taulukko 6.5: Kuvasarja 2, pääkomponenttien lukumäärän *pc* vertailu.

pc	CR	MAX	MAE	MSE	MSD	SNR	PSNR
4	81.17	78	12.23	195.08	4.49	25.13	44.63
3	101.78	78	12.23	195.16	4.49	25.13	44.63
2	136.47	78	12.29	196.78	4.51	25.09	44.60
1	207.06	174	15.42	317.58	5.65	22.93	42.44

6.2.4 Menetelmä 2: PCA-muunnos kokonaisille lohkoille

Ensimmäisen menetelmän variaationa testattiin tiivistystapaa, jossa kuvan muuttuneet lohkot käsitellään täydellisesti PCA-muunnoksella, eli yksittäisten kuvapisteiden vastaavuutta referenssikuvan kuvapisteiden kanssa ei tutkita, jos kokonaisen lohkon liikkeen estimointi epäonnistuu. Tulokset ovat käytännössä sekä visuaalisesti että numeerisesti samankaltaisia kuin edellä, mutta vaillinaisemman temporaalisen tiivistyksen vuoksi tiivistystehokkuus jää hieman huonommaksi (taulukot 6.6 ja 6.7).

Taulukko 6.6: Kuvasarja 1, pääkomponenttien lukumäärän *pc* vertailu.

pc	CR	MAX	MAE	MSE	MSD	SNR	PSNR
4	2.45	837	16.32	324.84	6.41	35.56	41.08
3	3.27	840	22.42	809.07	8.70	31.77	37.29
2	4.90	1323	40.79	3247.97	15.78	25.64	31.16
1	9.79	1457	91.50	16507.98	33.87	19.00	24.52

Taulukko 6.7: Kuvasarja 2, pääkomponenttien lukumäärän *pc* vertailu.

pc	CR	MAX	MAE	MSE	MSD	SNR	PSNR
4	67.18	67	10.66	137.50	3.98	26.49	46.00
3	85.69	67	10.66	137.51	3.98	26.49	46.00
2	118.28	67	10.73	139.70	4.00	26.42	45.93
1	190.89	200	14.20	274.31	5.26	23.47	42.98

6.2.5 Menetelmä 3: Aallokkeet

Kolmannessa menetelmässä verrattiin 2. ja 4. asteen Daubechies-aallokkeisiin perustuvien aallokemuunnosten soveltamista eri kvantisointiarvoilla. Tulosten perusteella tiivistystehokkuus jää muita menetelmiä huonommaksi erityisesti materiaalissa, jossa kokonaisten lohkojen liikkeen estimoinnnin onnistumisprosentti on pieni ja osittain muuttuneita lohkoja on paljon.

Lyhyemmässä kuvasarjassa 2. asteen Daubechies-aallokkeen eli Haarin aallokkeen tuottamien muunnoskertoimien kvantisoijan kasvattaminen tuottaa melko nopeasti suuria värivirheitä RGB-esitykseen ja näkyy vastaavasti myös numeerisissa virhearvoissa. Korkeamman asteen Daubechies-aalloketta sovellettaessa voidaan käyttää suurempia kvantisointiarvoja hienoisesti paremmalla tiivistyssuhteella. Molemmissa tapauksissa PSNR-arvon laskeminen alle 30:een näkyy käytännössä hyvin häiritsevinä värivirheinä videokuvassa (taulukko 6.8).

Taulukko 6.8: Kuvasarja 1, aallokkeen *wave* ja kvantisoijan *quant* vertailu.

wave	*quant*	CR	MAX	MAE	MSE	MSD	SNR	PSNR
2	2	1.81	176	16.08	292.56	6.30	36.14	41.66
2	4	2.20	1023	29.93	1047.10	11.63	30.47	35.99
2	6	2.67	1031	42.52	2188.96	16.61	26.97	32.49
2	8	3.14	1229	54.79	4302.04	21.67	23.33	28.85
4	12	1.99	278	21.02	512.42	8.21	33.76	39.28
4	20	2.47	766	31.94	1233.50	12.44	29.43	34.95
4	28	2.95	1002	42.31	2325.59	16.75	25.80	31.32
4	36	3.42	1253	52.71	3881.06	21.30	23.12	28.64

Pidemmässä kuvasarjassa, jossa temporaalisen tiivistyksen merkitys on huomattavasti spatiaalista ja spektraalista tiivistystä suurempi, tiivistyssuhteen arvot ovat lähempänä PCA-muunnokseen perustuvien menetelmien tuloksia (taulukko 6.9). Myös yleiset värisävyt säilyvät suurellakin tiivistyssuhteella paremmin kuin lyhyemmässä kuvasarjassa. Haarin aalloketta käytettäessä videokuva alkaa kuitenkin blokkiintua selvästi jo toiseksi pienimmällä kvantisoijan arvolla, ja suurimmalla testatulla kvantisoijan arvolla kuvan resoluutio on käytännössä pienentynyt verrannollisena lohkon sivun pituuteen. Neljännen asteen Daubechies-aallokkeeseen perustuvassa muunnoksessa kvantisoijan kasvu näkyy vähemmän dramaattisena kuvan yksityiskohtien katoamisena, mutta samalla RGB-esitykseen ilmaantuu alkuperäisestä kuvasarjasta poikkeavaa kirkkauden vaihtelua.

Taulukko 6.9: Kuvasarja 2, aallokkeen *wave* ja kvantisoijan *quant* vertailu.

wave	*quant*	CR	MAX	MAE	MSE	MSD	SNR	PSNR
2	2	72.48	74	13.67	217.20	5.13	24.35	43.85
2	4	142.22	542	23.16	662.86	8.36	19.81	39.32
2	6	213.61	758	30.05	1149.10	10.71	17.49	36.99
2	8	270.44	814	35.26	1608.67	12.59	15.97	35.48
4	12	142.12	532	19.50	499.96	7.00	21.11	40.61
4	20	192.93	1031	26.00	941.93	9.24	18.01	37.51
4	28	219.58	1679	31.08	1434.93	11.12	15.66	35.16
4	36	233.28	1853	36.83	1987.77	13.04	14.39	33.90

Luku 7

Pohdinta

Testimateriaalin oleelliset erot käyvät tuloksia tarkasteltaessa konkreettisesti ilmi: epäsäännöllistä laajalla alueella esiintyvää liikettä sisältävä lyhyt tietokoneanimaatio tiivistyy selvästi huonommin kuin pitkä ilmakuvasarja, jossa liikkeen suunta on ennustettavissa ja muutokset ovat hyvin säännöllisiä. Tuloksia arvioitaessa on kuitenkin huomattava, että lyhyemmän testivideon kohdalla liikkeen estimoinnissa spektrien vertailuun käytettyjen kanavien maksimierotusta kasvattamalla voitaisiin tiivistyssuhdetta todennäköisesti parantaa vielä huomattavasti ilman varsinaisia häiritseviä visuaalisia virheitä. Animaation huonommat tiivistysarvot johtunevat osittain myös keinotekoisesta muunnoksesta spektrikuviksi, jonka tuloksena kuvissa esiintyy lähes jatkuvaa väreilyä, ja liikkeen kompensointi ei toteudu niin hyvin kuin aitojen spektrikuvien tapauksessa voisi olla mahdollista.

Kuvanlaadun ja tiivistyssuhteen perusteella parhaita tuloksia tuottaa kuvan sisäiseen tiivistykseen käytetty pääkomponenttianalyysi yhdistettynä kuvien väliseen liikkeen kompensointiin. Testiaineiston perusteella jo 2-3 pääkomponenttia voi olla PCA-muunnoksessa riittävä määrä tyydyttävän rekonstruointituloksen saavuttamiseen. Pääkomponenttianalyysin huonona puolena on kuitenkin sen vaatimasta laskenta-ajasta johtuva hitaus. Ongelmaa voisi yrittää kiertää muodostamalla muunnoksessa käytettävän aliavaruuden kannan esimerkiksi Munsellin standardivärien spektrien perusteella ja käyttämällä tätä samaa kantaa kaikkien lohkojen muunnoksessa. Tällöin aikaavievä lohkokohtaisen autokorrelaatiomatriisin laskeminen sekä ominaisvektorien ja -arvojen etsiminen jäisivät pois. Jos oletetaan videosekvenssin kuvien sisältävän pääasiassa luonnonvärejä, menettelyssä ei välttämättä syntyisi kovin suurta virhearvojen kasvua.

Aallokkeita soveltavan tiivistyksen tulokset eivät tässä työssä ole parhaita mahdollisia etenkään aiemmin julkaistuihin yksittäiskuvien tiivistämistä käsitteleviin tutkimuksiin verrattuna, mikä johtuu suurelta osin siitä, että vain osittain muuttuneissa lohkoissa yksittäisten kuvapisteiden temporaalinen redundanssi edelliseen kuvaan verrattuna jää käytännön syistä hyödyntämättä. Lisäksi 8×8 kuvapisteen lohkokoko on todennäköisesti liian pieni aallokkeiden potentiaalin täysipainoiseen hyödyntämiseen. Videosekvenssin aloittavan I-kuvan koodaus yhtenä suurena lohkona aallokkeita käytettäessä olisi tosin mahdollista toteuttaa melko helposti, mutta sen vaikutus videon kokonaistiivistyssuhteeseen olisi kuitenkin varsin vähäinen. Virhearvojen melko nopea kasvu muunoskertoimien kvantisoijan kasvaessa johtunee osaltaan vakioarvoisen kvantisoijan käyttämisestä lohkon kaikille muunnoskertoimille. Kvantisoimalla kertoimet eri arvoilla muunnostason ja kertoimen informaatioarvon mukaan saattaisi tässä tapauksessa tuottaa parempia tuloksia, mutta yleispätevän kvantisoijataulukon määrittäminen lohkoille vaatisi lisäselvityksiä. Haasteeksi jää edelleen liikkeen kompensoinnin ja aallokkeiden sovittaminen toimimaan tehokkaammin yhdessä.

Liikkeen estimointiprosessissa käytettävä väriaistimukseen pohjautuva vertailumenetelmä toimii testien perusteella tyydyttävästi, vaikkakin vertailun kriteerinä toimivat maksimiarvot saattavat vaatia hienosäätöä aineiston luonteesta riippuen. Työn aikana saatiin joissakin tilanteissa hyväksyttäviä visuaalisia tuloksia jopa ainoastaan yhden tai kahden kanavan vertailulla, mikä luonnollisesti paransi tiivistyssuhdetta jonkin verran. Lopulliset testit suoritettiin kuitenkin objektiivisuuden vuoksi kaikkien kolmen päävärin aallonpituushuippujen vertailua soveltaen. Tavallisen videokoodauksen mukainen liikkeen estimointi erikseen laskettujen kirkkausarvojen perusteella kasvattaisi ohjelmiston laskentakompleksisuutta entisestäänkin ja vaatisi värierojen koodausta erikseen, mutta voisi olla yleispätevämpi tapa vertailun suorittamiseen.

Kuvien välisessä tiivistyksessä yleisesti loogisia kehityskohteita olisivat liikkeen estimoinnin maksimietäisyyden suurentaminen ja kahdessa suunnassa tapahtuvan estimoinnin toteuttaminen eli B-kuvien käyttöönotto, joilla molemmilla on tutkitusti suora ja melko nopea vaikutus tiivistystehokkuuteen. Tällä hetkellä ohjelmistossa käytetty videosekvenssin syntaksi ja tietorakenteet rajoittavat näiden muutosten välitöntä implementointia, mutta ohjelmiston mahdollisen toiminnalllisen laajentamisen yhteydessä olisi tarkoituksenmukaista ottaa mainitut ominaisuudet huomioon.

Vaikka numeerisia tunnuslukuja voikin käyttää viitteenä tiivistyksen onnistumisesta, lopullinen tuloksen hyväksyminen vaatii edelleen subjektiivista visuaalista arviointia. Spektrien luonteesta riippuen esimerkiksi liikkeen estimoinnissa käytettävä kanava-arvojen vertailu voi erilaisilla testiaineistoilla vaatia parametrien muuttamista oletusarvoistaan ennen tyydyttävän tuloksen saavuttamista, mikä tässäkin työssä kävi ilmi. Täysin objektiivisten ja absoluuttisten arviointimittausten kehittäminen on kuitenkin vaativa prosessi, johon on perinteisen videokoodauksen puolellakin vasta viime aikoina pyritty löytämään ratkaisuja.

Videokuvan visuaalisessa arvioinnissa lievää epävarmuutta aiheuttivat RGB-esitystä varten tehtävässä värimuunnoksessa syntyvät pyöristystarkkuudesta johtuvat virheet, jotka voivat tunnuslukujen perusteella hyvässäkin videokuvassa aiheuttaa yksittäisiä häiritseviä värivirheitä. RGB-esitykseen vaikuttaa myös värikomponenttiarvojen laskennassa käytetyn standardivalonlähteen tyyppi ja siitä riippuvat parametrit. Yksittäisiä spektritasoja harmaasävyesityksenä tarkasteltaessa virheet näkyvät monissa tapauksissa huomattavasti lievempinä kuin täysimittaisen RGB-muunnoksen jälkeen. Lopullisen visualisoinnin kannalta RGB-esityksen laatu on tietenkin hyvin merkittävä.

Parhaissakin työssä saavutetuissa tuloksissa tiivistyssuhdetta olisi vielä varaa konkreettisesti parantaa käyttämällä tiivistyksen viimeisessä vaiheessa esimerkiksi Huffman-koodausta tai vastaavaa bittipohjaista entropiakoodauksen menetelmää ja muuttamalla videoformaatin syntaksi kokonaisuudessaan yhtenäiseen bittivirtaan perustuvaksi esimerkiksi MPEG-videokoodausstandardin tavoin. Käytännössä tämä vaatisi järkevästi toteutuakseen adaptiivisen Huffman-koodauksen soveltamista ja samalla todennäköisesti ohjelmiston tämänhetkisen ytimen suunnittelua ja toteuttamista suurimmaksi osaksi uudelta pohjalta.

Ohjelmiston nopeus reaaliaikaisiin sovelluksiin ei tällä hetkellä ole tyydyttävällä tasolla, mikä johtuu tiivistyksessä käytettävien matemaattisten muunnosten vaatimasta laskentatehosta sekä visuaalista esitystä varten tarvittavasta RGB-muunnoksesta. Jonkin verran asiaan voitaisiin vaikuttaa kehittämällä toteutusta edellämainituin tavoin. Nopeus ei kuitenkaan ollut alunperin etusijalla ohjelmien kehitysvaiheessa, vaan varsinainen painopiste oli peruskonseptin toimivuuden testaamisessa ja erityyppisten menetelmien toteutusvaihtoehtojen vertailemisessa.

Kokonaisuutena työn voidaan katsoa onnistuneen melko tyydyttävästi, kun otetaan huomioon, että työssä pyrittiin yhdistämään yksittäisten spektrikuvien tiivistyksen ja perinteisen videokoodauksen menetelmiä toimivaksi kokonaisuudeksi aiemmista tutkimusalueen sovelluksista poikkeavalla tavalla. Vaikka käytännön toteutuksen optimaalisuudessa on vielä parantamisen varaa, suoritetut testit osoittavat, että ohjelmiston nykyversiollakin voidaan sopivissa käyttötarkoituksissa saavuttaa kohtuullisia tuloksia.

Lähteet

[1] E. C. Barrett, L. F. Curtis: Introduction to Environmental Remote Sensing, Second Edition. London: Chapman and Hall, 1982. pp. 3-25. ISBN 0-412-23090-9.

[2] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes: Computer Graphics: Principles and Practice, Second Edition. Reading: Addison-Wesley Publishing Company, 1990. pp. 574-587. ISBN 0-201-12110-7.

[3] P. Fränti: Block Coding in Image Compression. Department of Computer Science. University of Turku: Turku 1994. Research Reports R-94-12. pp. 6-16. ISSN 1235-6727. ISBN 951-29-0326-1.

[4] A. Gersho, R. M. Gray: Vector Quantization and Signal Compression. Boston: Kluwer Academic Publishers, 1992. pp. 225-257. ISBN 0-7923-9181-0.

[5] D. Hearn, M. P. Baker: Computer Graphics, Second Edition. Englewood Cliffs: Prentice-Hall, 1994. pp. 565-573. ISBN 0-13-159690-X.

[6] R. W. G. Hunt: Measuring Colour. Chichester: Ellis Horwood Limited, 1987. pp. 17-31, 74-104. ISBN 0-7458-0125-0.

[7] A. Kaarna: Application of Wavelets to Color Spectra. Proceedings of the 1997 Finnish Signal Processing Symposium. Tampere University of Technology, Pori School of Technology and Economics. Pori: Porin korkeakoulu,

TTKK, TKKK, 1997. Julkaisusarja nro A16. pp. 148-152. ISSN 1238-321X. ISBN 952-9607-15-6.

[8] S. G. Mallat: A Wavelet Tour of Signal Processing. San Diego: Academic Press, 1998. pp. 7-17, 302-312, 497-508. ISBN 0-12-466605-1.

[9] M. Misiti, Y. Misiti, G. Oppenheim, J-M. Poggi: Wavelet Toolbox User's Guide. Natick: The MathWorks Inc., 1996.

[10] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, D. J. LeGall: MPEG Video Compression Standard. New York: Chapman & Hall, 1996. pp. 1-32, 51-103, 237-262, 283-312, 395-439. ISBN 0-412-08771-5.

[11] J. Parkkinen: Subspace Methods in Two Machine Vision Problems. Department of Computer Science and Applied Mathematics. Kuopio: University of Kuopio, 1989. Natural Sciences, Original Reports 1/1989. pp. 8-13, 33-40. ISSN 0781-0873. ISBN 951-780-759-7.

[12] J. Parkkinen, J. Hallikainen, T. Jääskeläinen: Characteristic Spectra of Munsell Colors. Journal of the Optical Society of America A, 1989. Vol. 6, No. 2, pp. 318-322. ISSN 0740-3232.

[13] J. Parkkinen, T. Jääskeläinen: Color Representation Using Statistical Patter Recognition. Applied Optics, 1987. Vol. 26, No. 19, pp 4240-4245. ISSN 0003-6935.

[14] M. Rabbani, P. W. Jones: Digital Image Compression Techniques. Bellingham: SPIE, 1991. pp. 3-10, 102-128. ISBN 0-8194-0648-1.

[15] A. Saarinen: Spektrikuvan tiivistys ja esitys. Diplomityö. LTKK, Tietotekniikan osasto, 1994. pp. 6-14, 62-63.

[16] R. Steinmetz, K. Nahrstedt: Multimedia: Computing, Communications and Applications. Upper Saddle River: Prentice-Hall, 1995. pp. 113-174. ISBN 0-13-324435-0.

[17] J. Weiss, D. Schremp: Putting data on a diet. IEEE Spectrum, 1993. Vol. 30, No. 8, pp. 36-39. ISSN 0018-9235.

[18] R. Westwater, B. Furht: Real-Time Video Compression: Techniques and Algorithms. Boston: Kluwer Academic Publishers, 1997. pp. 1-13. ISBN 0-7923-9787-8.

[19] G. Wyszecki, W. S. Stiles: Color Science: Concepts and Methods, Quantitative Data and Formulae, Second Edition. New York: John Wiley & Sons, 1982. pp. 117-175. ISBN 0-471-02106-7.

[20] A. I. Zayed: Handbook of Function and Generalized Function Transformations. Boca Raton: CRC Press, 1996. pp. 531-555, 585-586. ISBN 0-8493-7851-6.

Liite I

Testiaineisto

Molemmista testeissä käytetyistä kuvasarjoista on aluksi esitetty esimerkkikuvat, joiden yhteydessä on lyhyesti kerrottu ko. aineiston ominaisuuksista. Menetelmätestien visuaalisia tuloksia on tämän jälkeen havainnollistettu esimerkein kuvasarjan ja käytetyn menetelmän mukaisessa järjestyksessä. Jokaista testiä kohti on kuvasarjasta valittu sama kuva.

Kuvateksteissä on ilmoitettu suluissa menetelmäkohtaisten parametrien arvot kyseisen testin suorituksen aikana. Tiivistyssuhde on laskettu alkuperäisten MSI-kuvatiedostojen ja tiivistetyn MSV-videotiedoston koon perusteella.

Kuvateksteissä esiintyvien lyhenteiden selitykset:

CR	=	tiivistyssuhde
pc	=	pääkomponenttien lukumäärä
wave	=	aallokkeen aste
quant	=	kvantisoijan arvo

Kuva I.1: Kuvasarja 1, tietokoneanimaatio, resoluutio 320 $\times$ 224, 60 kuvaa. Koko kuvan alueella esiintyy jatkuvia, nopeita ja epäsäännöllisiä muutoksia, jotka aiheutuvat leijuvista kukan terälehdistä sekä käden ja itse “kameran” liikkumisesta.

Kuva I.2: Kuvasarja 2, ilmakuvasarja, resoluutio 352 $\times$ 288, 218 kuvaa. Kuvien sisältämä liike muodostuu säännöllisen lentoreitin mukaisesta maiseman vähittäisestä muuttumisesta.

Kuva I.3: Kuvasarja 1, menetelmä 1 (*pc* = 4). CR = 2.98.

Kuva I.4: Kuvasarja 1, menetelmä 1 (*pc* = 3). CR = 3.92.

Kuva I.5: Kuvasarja 1, menetelmä 1 (*pc* = 2). CR = 5.76.

Kuva I.6: Kuvasarja 1, menetelmä 1 (*pc* = 1). CR = 10.82.

Kuva I.7: Kuvasarja 1, menetelmä 2 (*pc* = 4). CR = 2.45.

Kuva I.8: Kuvasarja 1, menetelmä 2 (*pc* = 3). CR = 3.27.

Kuva I.9: Kuvasarja 1, menetelmä 2 (*pc* = 2). CR = 4.90.

Kuva I.10: Kuvasarja 1, menetelmä 2 (*pc* = 1). CR = 9.79.

Kuva I.11: Kuvasarja 1, menetelmä 3 (*wave* = 2, *quant* = 2). CR = 1.81.

Kuva I.12: Kuvasarja 1, menetelmä 3 (*wave* = 2, *quant* = 4). CR = 2.20.

Kuva I.13: Kuvasarja 1, menetelmä 3 (*wave* = 2, *quant* = 6). CR = 2.67.

Kuva I.14: Kuvasarja 1, menetelmä 3 (*wave* = 2, *quant* = 8). CR = 3.14.

Kuva I.15: Kuvasarja 1, menetelmä 3 (*wave* = 4, *quant* = 12). CR = 1.99.

Kuva I.16: Kuvasarja 1, menetelmä 3 (*wave* = 4, *quant* = 20). CR = 2.47.

Kuva I.17: Kuvasarja 1, menetelmä 3 (*wave* = 4, *quant* = 28). CR = 2.95.

Kuva I.18: Kuvasarja 1, menetelmä 3 (*wave* = 4, *quant* = 36). CR = 3.42.

Kuva I.19: Kuvasarja 2, menetelmä 1 (*pc* = 4). CR = 81.17.

Kuva I.20: Kuvasarja 2, menetelmä 1 (*pc* = 3). CR = 101.78.

Kuva I.21: Kuvasarja 2, menetelmä 1 (*pc* = 2). CR = 136.47.

Kuva I.22: Kuvasarja 2, menetelmä 1 (*pc* = 1). CR = 207.06.

Kuva I.23: Kuvasarja 2, menetelmä 2 (*pc* = 4). CR = 67.18.

Kuva I.24: Kuvasarja 2, menetelmä 2 (*pc* = 3). CR = 85.69.

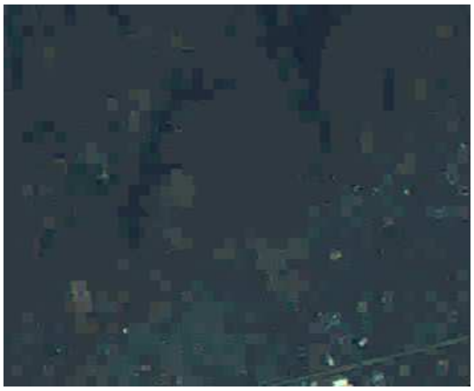
Kuva I.25: Kuvasarja 2, menetelmä 2 (*pc* = 2). CR = 118.28.

Kuva I.26: Kuvasarja 2, menetelmä 2 (*pc* = 1). CR = 190.89.

Kuva I.27: Kuvasarja 2, menetelmä 3 (*wave* = 2, *quant* = 2). CR = 72.48.

Kuva I.28: Kuvasarja 2, menetelmä 3 (*wave* = 2, *quant* = 4). CR = 142.22.

Kuva I.29: Kuvasarja 2, menetelmä 3 (*wave* = 2, *quant* = 6). CR = 213.61.



Kuva I.30: Kuvasarja 2, menetelmä 3 (*wave* = 2, *quant* = 8). CR = 270.44.

Kuva I.31: Kuvasarja 2, menetelmä 3 (*wave* = 4, *quant* = 12). CR = 142.12.

Kuva I.32: Kuvasarja 2, menetelmä 3 (*wave* = 4, *quant* = 20). CR = 192.93.

Kuva I.33: Kuvasarja 2, menetelmä 3 (*wave* = 4, *quant* = 28). CR = 219.58.

Kuva I.34: Kuvasarja 2, menetelmä 3 (*wave* = 4, *quant* = 36). CR = 233.28.