Matias Yli-Hillilä

Fast Fourier Transform -algoritmit digitaalisessa kuvankäsittelyssä

Tietotekniikan kandidaatintutkielma

31.12.2020

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Matias Yli-Hillilä

Yhteystiedot: maanyli@student.jyu.fi

Ohjaaja: Sanna Mönkölä

Työn nimi: Fast Fourier Transform -algoritmit digitaalisessa kuvankäsittelyssä

Title in English: Fast Fourier Transform algorithms in digital image processing

Työ: Kandidaatintutkielma

Sivumäärä: 29+0

Tiivistelmä: Tutkimuksessa esitellään Fast Fourier Transform -algoritmien (FFT) pääpiirteet ja niiden käyttökohteita digitaalisessa kuvankäsittelyssä.

Avainsanat: FFT, Fourier-muunnos, digitaalinen kuvankäsittely, digitaalinen kuvanprosessointi, numeerinen analyysi

Abstract: This study exhibits the main points of Fast Fourier Transform algorithms (FFT) and some of their use cases in digital image processing.

Keywords: FFT, Fourier transform, digital image processing, numerical analysis

Termiluettelo

digitaalinen kuva	(kaksi)ulotteisen kuvan tietokone-esitys, eli tallennettu bitteinä
digitaalinen kuvankäsittely	*(digital image processing)* algoritmien käyttäminen digitaalisen kuvan käsittelyyn
Fourier-muunnos	integraalimuunnos, joka muuntaa signaalin aikatasolta taajuustasolle
diskreetti Fourier-muunnos	kuin Fourier-muunnos, mutta äärellinen diskreetti summa
konvoluutio	operaatio, jolle syötetään systeemin pistejakaumafunktio sekä jokin yksittäinen (normalisoitu) impulssi, ja ulos saadaan täsmälleen, miten impulssi on muuttunut
Fast Fourier Transform -algoritmit	harvoihin matriiseihin perustuvien algoritmien joukko, jotka tekevät diskreetin Fourier-muunnoksen nopeammin ja tarkemmin kuin edellämainittu "suora" implementaatio
butterfly	laskentatoimitus, joka liittää pienemmät DFT:t yhteen tai jakaa suuren DFT:n pienempiin osiin
permutaatiomatriisi	Matriisi, jolla kertomalla voi muuttaa toisen matriisin rivien tai sarakkeiden järjestystä
bit-reversal permutation	permutaatio, joka "peilaa" bittimuodossa olevan datajoukon jokaisen alkion s.e. ne "alkavat lopusta"
twiddle factor	FFT-algoritmissa esiintyvät trigonometriset vakiokertoimet
decimation-in-time (DIT)	tapa tehdä FFT-jako; käytännössä valitaan joka n:s piste yhteen osioon
decimation-in-frequency (DIF)	tapa tehdä FFT-jako; käytännössä jaetaan data n:nään osaan
flop	Floating Point Operation eli liukulukulaskutoimitus

Lyhenteet

DFT	Diskreetti Fourier-muunnos
FFT	Fast Fourier Transform
DCT	Discrete Cosine Transform
IDCT	Inverse Discrete Cosine Transform

Sisältö

1 Johdanto

1.1 Fourier-muunnos ja signaalinkäsittely

Fourier-muunnos on useilla aloilla erittäin tärkeä matemaattinen integraalimuunnos, joka voidaan tulkita siten, että se muuntaa ajan yli esitetyn datan taajuuden yli esitetyksi (Gomes 2015). Fourier-muunnokseen liittyy olennaisesti myös *käänteinen Fourier-muunnos*, joka voidaan tulkita datavirran muunnoksena taajuustasolta aikatasolle. Fourier-muunnoksella on useita käytännön sovellusten kannalta hyödyllisiä ominaisuuksia.

Yleensä Fourier-muunnosta käytetään muuttamaan jokin hankalasti ratkaistava ongelma sellaiseen muotoon, jossa ongelma on helpompi ratkaista. Havainnollistava esimerkki on ääninauhoite, jossa on häiriöitä. Fourier-muunnos muuntaa nauhoitteen sisältämät taajuudet jakaumaksi, josta voidaan helposti havaita ja eristää häiriötaajuudet. Muunnetusta signaalista poistetaan kyseiset häiriötaajuudet, ja sen jälkeen signaali muunnetaan alkuperäiseen muotoon käänteisellä Fourier-muunnoksella. Lopputuloksena on (lähestulkoon) häiriötön nauhoite.

Vaikka teoriassa useimmat signaalit ovat jatkuvia[1], suurin osa signaaleista mitataan, tallennetaan ja toistetaan diskreetissä muodossa. Siksi on oleellista yleistää Fourier-muunnos diskreettiin aikaan, jolloin puhutaan *Fourier-sarjasta*. Jos tämä sarja rajataan äärelliseksi, saadaan *diskreetti Fourier-muunnos* (DFT), ja samaa logiikkaa soveltaen *käänteinen diskreetti Fourier-muunnos* (DTFT). DFT:tä voidaan käyttää johonkin todelliseen jaksolliseen otokseen, ja DFT sekä DTFT voidaan kuvata matriisimuodossa. Tämä matriisimuoto muodostaa *Fast Fourier Transform -algoritmien*[2](FFT) pohjan.

FFT-algoritmien hyödyt ovat nopeus ja tarkkuus: niiden laskennallinen aikavaativuus on $\mathcal{O}(n \log n)$, siinä missä DFT:n laskemisen aikavaativuus on $\mathcal{O}(n^2)$ (Pickering 1986) (Van Loan 1992); ja toisaalta kirjoittajan (1978):n mukaan DFT:n melun suhde signaalipisteiden määrään N on $\mathcal{O}(N)$ tai $\mathcal{O}(N^2)$ riippuen pyöristystavasta, mutta FFT:lle $\mathcal{O}(log_2(N))$ tai $\mathcal{O}([log_2(N)]^2)$.

1. Kvanttimekaanisella tasolla tämä ei ole totta, mutta tämä tutkimus keskittyy makrotason ilmiöihin.
2. Suoraan suomennettuna "Nopea Fourier–muunnos-algoritmi".

1.2 Digitaalinen kuvankäsittely

Digitaaliset kuvat ovat tietokoneelle tallennettuja 2D-rasterikuvia[3], joihin on yleensä sovellettu jotain pakkausalgoritmia. Digitaaliset kuvat voivat esiintyä sellaisenaan tai kuvien joukkona, joka on tallennettu tai joka piirretään reaaliajassa. Esimerkiksi elokuvat ovat valmiiksi tallennettujen kuvien joukkoja, kun taas vaikkapa lääketieteessä käytetään *tähystimiä* ihmisten sisäelimien tarkasteluun reaaliaikaisesti.

Kuvia on usein tarve prosessoida, eli käsitellä algoritmeilla. Edellämainitut pakkausalgoritmit ovat (tärkeä) esimerkki kuvankäsittelystä. Muita esimerkkejä ovat kuvien "yksinkertaistaminen" automatisaation mahdollistamiseksi, kuvien koon muuttaminen, kuvien kääntäminen, kirkkauden ja kontrastin säätö, ja niin edelleen.

Sovelluskohteesta riippuen algoritmeilla on erilaisia vaatimuksia: esimerkiksi reaaliaikaisessa renderöinnissä on tärkeää, että algoritmi on tehokas (skaalautuva) ja laitteistoon sopiva. Ongelmia voi aiheutua, jos algoritmi vaikkapa tarvitsee liian paljon muistia, joutuu tekemään paljon alustustöitä, tai toimii välillä hitaammin. Toisaalta ei-reaaliaikaisissa kohteissa saatetaan haluta painottaa sitä, että lopputulos vie mahdollisimman vähän tilaa edellämainittujen kustannuksella.

1.3 Tutkielman rakenne

Tämän tutkielman tavoitteena on esitellä lukijalle FFT-algoritmien pääasialliset käyttötavat digitaalisessa kuvankäsittelyssä. Johdanto-osiossa esitellään Fourier-muunnokset, FFT-algoritmit sekä digitaalinen kuvankäsittely pääkohdiltaan. Toisessa luvussa tarkastellaan FFT-algoritmien ja digitaalisen kuvankäsittelyn matemaattista pohjaa. Kolmannessa luvussa tutustutaan varsinaisiin FFT-algoritmien sovelluksiin digitaalisessa kuvankäsittelyssä. Lopulta neljännessä luvussa tehdään yhteenveto tutkielmasta.

3. On olemassa myös vektorikuvia, ja teoriassa digitaalisia kuvia voitaisiin tallentaa muutenkin kuin bittimuodossa.

2 Fourier-muunnos ja Fast Fourier Transform -algoritmit

Tämä luku perustuu teoksiin Van Loan (1992) ja Smith (1997).

Fourier-muunnos on matemaattinen operaatio, jota käytetään laajasti matemaattisessa analyysissä ja signaalinkäsittelyssä. Matemaattisessa analyysissä sitä sovelletaan mm. osittaisdifferentiaaliyhtälöiden ratkaisemisessa, ja signaalinkäsittelyssä useilla sovellusalueilla, joilla on tarpeen analysoida signaalia sen taajuuden suhteen. Signaalit voivat olla esimerkiksi puhetta, kuvia, tai elektronisen mittalaitteiden tuottamaa tietoa. Tämän tutkielman pääpainopiste on digitaalisessa kuvankäsittelyssä käytettyjen algoritmien matemaattisessa taustassa, mutta tutkielman loppuosassa tuodaan esille myös muita sovelluskohteita.

Fourier-muunnos perustuu matematiikan alalla tunnettuun teoriaan, jonka mukaan funktio voidaan tietyin edellytyksin esittää sini- ja kosinimuotoisten funktioiden integraalina tai summana. Digitaalisessa kuvankäsittelyssä käsiteltäviä datapisteitä on paljon, joten datapisteistö on luontevaa esittää yhteenä muuttujana, vektorina.

2.1 Matemaattiset määritelmät

2.1.1 Fourier-muunnos ja käänteinen Fourier-muunnos

Smithin (1997):in mukaan signaaleja voidaan esittää aika- ja taajuustasolla. Fourier-muunnoksen ominaisuudet kertovat, miten signaalin muutos yhdellä tasolla vaikuttaa toisella tasolla. Muunnoksen avulla signaali voidaan saattaa muotoon, jota on helpompi käsitellä jonkin tietyn ongelman näkökulmasta: esimerkiksi signaalin taajuutta on helpompi muuttaa, kun signaali on esitetty taajuustasolla.

(Jatkuva) Fourier-muunnos $\hat{f}$ voidaan määritellä seuraavasti[1]:

$$\hat{f}(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i x k} dx \tag{2.1}$$

1. Määritelmät vaihtelevat eri sovellusalueilla, koska eri määritelmillä voidaan yksinkertaistaa erilaisten laskutoimitusten kirjoitusasua; muunnos toimii olennaisesti samalla tavalla määritelmästä riippumatta.

Fourier-muunnos muuntaa aikasignaalin taajuussignaaliksi. Osa signaaleista kuitenkin mitataan taajuustasolla, ja yleensä käsitelty signaali halutaan ilmaista sen alkuperäisessä mittausmuodossa. Tämän takia tarvitaan operaatio, joka muuntaa signaalin takaisin taajuustasolta aikatasolle. Tämä operaatio on aina mahdollista tehdä, ja operaatiota kutsutaan nimellä *käänteinen Fourier-muunnos*.

Muunnettu data voidaan muuttaa täsmälleen alkuperäiseen muotoonsa käyttämällä käänteistä Fourier-muunnosta:

$$f(k) = \int_{-\infty}^{\infty} \hat{f}(k) e^{2\pi i x k} dk \tag{2.2}$$

(Käänteinen) Fourier-muunnos voidaan tehdä jatkuville signaaleille. Yleisesti käytetyt tietokoneet kuitenkin tallentavat datan digitaalisessa muodossa, eli data on äärellisessä ja numeroituvassa (diskreetissä) muodossa. Tämän takia tutkielmassa käsitellään Fourier-muunnoksen yleisempää muotoa, *diskreettiä Fourier-muunnosta* (DFT, Discrete Fourier Transform), jota voidaan soveltaa diskreetteihin signaaleihin ja joka voidaan esittää matriisimuodossa.

DFT voidaan määritellä seuraavasti[2]:

$$y_k = \sum_{j=0}^{n-1} \omega_n^{kj} x_j \qquad \begin{aligned} y &= [y_0, \ldots, y_{n-1}]^T \\ x &= [x_0, \ldots, x_{n-1}]^T \\ \omega_n &= \cos(2\pi/n) - i\sin(2\pi/n) = e^{-2\pi i/n} \end{aligned} \tag{2.3}$$

Tässä y_k ja x_k ovat yksittäisiä datapisteitä. Signaalinkäsittelyssä kuitenkin käsitellään yleensä joukkoa datapisteitä, jolloin on luontevaa käyttää diskreetin muunnoksen matriisimuotoa:

$$y = F_n x; \qquad \begin{aligned} F_n &= (f_{pq}) \\ f_{pq} &= \omega_n^{pq} = e^{-2\pi q p i/n} \end{aligned} \tag{2.4}$$

2. Jatkuvan muunnoksen tapaan diskreetille muunnokselle on olemassa useampia yhtäpitäviä määritelmiä, joista valitaan sovellusalueeseen asianmukaisin.

DFT-matriisin koko on $n \times n$. Yleensä DFT:llä tarkoitetaan em. matriisimuotoa.

Käänteinen diskreetti Fourier-muunnos voidaan muodostaa diskreetin Fourier-muunnoksen operaatiomatriisiin perustuen:

$$x_n = \frac{1}{n} \sum_{k=0}^{n-1} e^{2\pi qpi/n} y_k \tag{2.5}$$

2.1.2 Kosinimuunnos

Fourier-muunnoksen määritelmästä nähdään, että se voidaan jakaa kosini- ja sinikomponentteihin. Jättämällä sinikomponentti pois päädytään *kosinimuunnokseen*. Kosinimuunnosta käytetään esimerkiksi JPG-kuvien pakkaamisessa (Smith 1997). Kosinimuunnoksen määritelmä on siis

$$\hat{f}_{cos}(k) = \int_{-\infty}^{\infty} f(t) \cos(2\pi xk) dx. \tag{2.6}$$

Diskreetissä tapauksessa kosinimuunnos voidaan määritellä monella hieman erilaisella tavalla. Se, mitä määritelmää kannattaa käyttää, riippuu sovelluskohteesta. Yleisimmin käytetty muoto tunnetaan nimellä DCT-II (lyhenne tulee sanoista *discrete cosine transform*):

$$y_k = \sum_{j=0}^{m-1} \cos\left(\frac{k(2j+1)\pi}{2m}\right) x_j \tag{2.7}$$

2.1.3 Konvoluutio

Signaalinkäsittelyssä käytetään yleisesti Fourier-muunnoksen lisäksi operaatiota nimeltä *konvoluutio* (Smith 1997). Konvoluutio on operaatio, jolle syötetään systeemin *pistejakaumafunktio* (point spread function)[3] sekä jokin yksittäinen (normalisoitu) impulssi, ja ulos saadaan täsmälleen, miten impulssi on muuttunut. Koska signaalit voidaan jakaa yksittäisiin impulssikomponentteihin, pistejakaumafunktio (ja konvoluutio) kuvaa, kuinka systeemi reagoi signaaleihin yleisesti.

3. Joskus pistejakaumafunktiosta käytetään nimitystä impulssivaste, mutta kuvankäsittelyssä pistejakaumafunktio-ilmaisu on yleisempi.

Smith (1997):n mukaan kahden funktion f ja g konvoluutio määritellään seuraavasti:

$$y(t) = x(t) \star h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \qquad y[n] = x[n] \star h[n] = \sum_{j=0}^{M-1} h[j]x[i-j] \qquad (2.8)$$

Konvoluution yhteys Fourier-muunnokseen on muun muassa siinä, että joitakin konvoluutioalgoritmeja voidaan toteuttaa FFT-algoritmeja hyväksikäyttäen (Van Loan 1992).

2.2 Fast Fourier Transform -algoritmit

DFT:n laskeminen voidaan toteuttaa suoraviivaisesti algoritmiksi, mutta suoraviivainen algoritmi on hidas ja epätarkka: sen laskennallinen vaativuus on $\mathcal{O}(n^2)$.[4] Sen sijaan käytetään FFT-algoritmeja, joiden laskennallinen vaativuus on $\mathcal{O}(n\log(n))$. Laskennallinen vaativuus kertoo, kuinka monta laskutoimitusta jonkin ongelman ratkaisu (Fourier-muunnoksen tekeminen) vaatii suhteessa ongelman kokoon (datapisteiden määrään).

Joissain tapauksissa prosessoriarkkitehtuuri voi vaikuttaa algoritmin valintaan ja toteutukseen. Yksinkertaisin esimerkki on moniydinprosessorit, joiden tapauksessa laskenta on liukuistettava/putkitettava (*pipeline*). Jos prosessori kykenee käsittelemään vektoreita, laskenta saattaa usein helpottua.

Kaikki FFT- ja DCT-algoritmit ovat hajota–ja–hallitse-algoritmeja. Hajota–ja–hallitse-algoritmit jakavat ratkaistavan ongelman pienempiin osaongelmiin, ratkaisee osaongelmat, ja yhdistää ratkaisut. Eri algoritmit jakavat laskettavat matriisit erikokoisiin osiin. FFT-algoritmit voidaan jakaa kahteen pääluokkaan: kantalukualgoritmit (*radix algorithms*) ja alkulukualgoritmit (*prime factor algorithms*). Alkulukualgoritmien tapauksessa jaettujen matriisien koot n_1 ja n_2 ovat keskenään jaottomat ja siten erisuuret. Kantalukualgoritmien tapauksessa jaetut matriisit voivat olla samaa tai eri kokoa, eikä jaon tarvitse tapahtua yhtä suuriin osiin joka jaolla.

Käytettävän algoritmin valintaan vaikuttaa monta seikkaa sekä laitteisto- että ohjelmistotasolla:

Ensinnäkin datan on oltava algoritmin jakotapaan nähden sopivan kokoinen: seitsemän datapisteen

4. Vektoriprosessorilla vaativuudeksi voidaan saada $\mathcal{O}(n)$, mutta niiden käyttö nykyään on harvinaista.

joukkoa ei voi sellaisenaan jakaa kahtia, joten kantaluvun 2 algoritmeja ei voida soveltaa kyseiseen datajoukkoon.

Valinnassa on otettava huomioon toimintaympäristön tapa tallentaa vektorit ja matriisit: onko matriisit tallennettu rivi- vai sarakejärjestyksessä; mikä on datapisteiden tallennusjärjestys; kompleksisten vektorien tallennusjärjestys; ja niin edelleen.

Eri algoritmit palauttavat ja käsittelevät muunnetun datan eri tallennusjärjestyksessä. Joidenkin algoritmien kohdalla muunnetun datan datapisteet eivät ole enää alkuperäisessä järjestyksessä lopussa tai muunnoksen aikana, ja eri algoritmit tallentavat datan joko riveittäin tai sarakkeittain.

Monesti algoritmin valintaan vaikuttavat muistinkäytön haasteet. Joissain tapauksissa ei ole mahdollista käyttää erillistä "työtilaa", jolloin muunnos on tehtävä suoraan alkuperäisen datan päälle. Luonnollisesti tätä ei voida tehdä, jos alkuperäinen data halutaankin säilyttää. Joskus myös muunnoksen välivaiheet halutaan tallentaa.

Väärä algoritmivalinta saattaa aiheuttaa sen, että peräkkäiset operaatiot joko vaativat viittauksia toisistaan kaukana oleviin muistialueisiin tai viittaukset hyppäävät seuraavaan muistilohkoon, mikä on hidasta (*stride issue*). Tärkein valinta muistiviittausten suhteen on se, tehdäänkö muunnos riveittäin vai sarakeittain.

Algoritmit käyttävät laskennassaan painokertoimia, jotka voidaan joko laskea etukäteen tai muunnoksen aikana. Jos algoritmi on toteutettu väärin, samat kertoimet saatetaan joutua laskemaan monta kertaa, ja toisaalta kertoimien laskutapa vaikuttaa laskennon nopeuteen ja tarkkuteen.

2.2.1 Cooley–Tukey-algoritmikehys

FFT-algoritmeista tunnetuin on *Cooleyn-Tukeyn algoritmi* (CT-algoritmi). CT-algoritmit ovat kantalukualgoritmeja; yksinkertaisimmassa tapauksessa puhutaan *kantaluvun 2 CT-algoritmista* (*radix-2 Cooley-Tukey algorithm*).

CT-algoritmi voidaan esittää seuraavana matriisitulona:

$$F_n = A_t \ldots A_1 P_n^T \tag{2.9}$$

Tässä P_n on nk. *"bitinkääntöpermutaatio"* (bit-reversing permutation), ja sen tehtävä on järjestää data siten, että se voidaan suoraviivaisesti jakaa pienempiin osiin, joita voidaan erikseen käsitellä DFT:llä (so. FFT-algoritmilla); A_q-matriisille ei ole vakiintunutta nimeä. Van Loan (1992) kutsuu A_q-matriisien kokonaisuutta termillä *combination phase*[5], joten se voitaisiin kääntää esimerkiksi *"yhdistelmämatriisiksi"*. Se vastaa edellä kuvatun DFT:n summalauseketta.

Kantaluvun 2 "yhdistelmämatriisin"varsinainen määritelmä on hieman monimutkaisempi:

$$\begin{aligned} A_q &= I_r \otimes B_L & L &= 2^q \\ B_L &= \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix} & r &= n/L \\ \Omega_{L/2} &= diag(1, \omega_L, \ldots, \omega_L^{L/2-1}) & \omega_L &= e^{-2\pi i/L} \end{aligned} \tag{2.10}$$

Määritelmää kannattaa lähteä tutkimaan tutuista lähtökohdista käsin: ω_L-termi määriteltiin jo aikaisemmin. $L = 2^q$ kuvaa, monesko kahtiajako on meneillään.

B_L tunnetaan erityisnimellä *perhonen* (butterfly). Sen toimintatapa on helpointa osoittaa kertomalla vektoria $z = \begin{bmatrix} z_T \\ z_B \end{bmatrix}$[6] perhosella:

$$\begin{aligned} y = \begin{bmatrix} y_T \\ y_B \end{bmatrix} = B_L z &= \begin{bmatrix} I_1 & \Omega_{L/2} \\ I_1 & -\Omega_{L/2} \end{bmatrix} \begin{bmatrix} z_T \\ z_B \end{bmatrix} \\ &= \begin{bmatrix} z_T + \Omega_{L/2} z_B \\ z_T - \Omega_{L/2} z_B \end{bmatrix} \end{aligned} \tag{2.11}$$

5. Sivu 21, Algorithm 1.3.1.

6. T(op), B(ottom)

CT-algoritmin tapauksessa kyseessä on erityisesti perhosen *aikadesimaatio*-versio (decimation-in-time, DIT). Joissakin FFT-algoritmeissä käytetään *taajuusdesimaatio*-perhosta (decimation-in-frequency, DIF), joka on DIT-perhosen transpoosi. Nimet juontuvat siitä, miten signaalinkäsittelyssä aika- tai taajuusriippuvainen muuttujavektori voidaan jakaa osiin; Van Loan (1992) mukaan terminologiasta voi löytää lisätietoa Cochran ym. (1967):ltä.

Operaattori $\otimes$ on *Kroneckerin tulo*, joka vastaa sitä, että ensimmäisen matriisin jokaista komponenttia kerrotaan oikealta toisella matriisilla. Esimerkiksi tapauksessa $r = 2$:

$$
\begin{aligned}
I_2 \otimes B_L &= \begin{bmatrix} I_2 & 0 \\ 0 & I_2 \end{bmatrix} \otimes \begin{bmatrix} I_1 & \Omega_{L/2} \\ I_1 & -\Omega_{L/2} \end{bmatrix} = \begin{bmatrix} I_2 \begin{bmatrix} I_1 & \Omega_{L/2} \\ I_1 & -\Omega_{L/2} \end{bmatrix} & 0 \\ 0 & I_2 \begin{bmatrix} I_1 & \Omega_{L/2} \\ I_1 & -\Omega_{L/2} \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \begin{bmatrix} I_1 & \Omega_{L/2} \\ I_1 & -\Omega_{L/2} \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} I_1 & \Omega_{L/2} \\ I_1 & -\Omega_{L/2} \end{bmatrix} \end{bmatrix} \\
&= diag(B_L, B_L)
\end{aligned}
\tag{2.12}
$$

Itse CT-algoritmin voi toteuttaa monella tavalla riippuen siitä, millaisia rajotteita käytettävään ohjelmistoon, laitteistoon ja ongelmaan liittyy.

Van Loan (1992) esittelee muun muassa seuraavan ylikirjoittavan CT-algoritmitoteutuksen[7]:

Olkoon $x \in \mathbb{C}^n$ ja $n = 2^t$. Tällöin seuraava algoritmi ylikirjoittaa $x : n$ $F_n x$:llä:

for $q = 1 : t$

L $\leftarrow 2^q$; r $\leftarrow n/L$; $L_* \leftarrow L/2$

for $j = 0$ to $L_* - 1$

$\omega \leftarrow \cos(2\pi j/L) - i\sin(2\pi j/L)$

7. Sivu 44, Algorithm 1.6.1

```
    for k = 0 : r − 1
      τ ← ω · x(kL + j + L*)
      x(kL + j + L*) ← x(kL + j) − τ
      x(kL + j) ← x(kL + j) + τ
    end
  end
end
```

2.2.2 Muut FFT-algoritmikehykset

Muut FFT-algoritmikehykset pohjaavat pääasiassa erilaisiin matriisituloihin sekä konvoluutioteoreemaan. Van Loan (1992) esittelee kahdeksan erilaista jakotapaa:

(DIT)-Cooley-Tukey:	$F_n = A_t \dots A_1 P_n$	(DIF)-Cooley-Tukey:	$P_n A_1^T \dots A_t^T$
(DIT)-Pease:	$F_n = H_t \dots H_1 P_n$	(DIF)-Pease:	$P_n H_1^T \dots H_t^T$
(DIT)-Transposed Stockham	$S_t \dots S_1$	(DIF)-Transposed Stockham:	$S_1^T \dots S_t^T$
(DIT)-Stockham:	$G_t \dots G_1$	(DIF)-Stockham:	$G_1^T \dots G_t^T$

$$
\begin{aligned}
A_q &= I_r \otimes B_L \\
H_q &= \begin{bmatrix} I_{n/2} & (\Omega_{L/2} \otimes I_r) \\ I_{n/2} & -(\Omega_{L/2} \otimes I_r) \end{bmatrix} \Pi_n^T \\
S_q &= (I_r \otimes B_L)(\Pi_{2r} \otimes I_{L/2}) \\
G_q &= (B_L \otimes I_r)(\Pi_L \otimes I_r) \\
L &= 2^q \\
r &= n/L
\end{aligned}
$$

Tässä etuliite DIT on lyhenne termistä *decimation-in-time* ja DIF termistä *decimation-infrequency*. Termit viittaavat siihen, että DIT-algorimit tekevät FFT-jaon ajan suhteen aikariippuvaiselle sisääntulodatalle x, ja DIT-algorimit taajuuden suhteen taajuusriippuvaiselle muunnetulle ulostulodatalle y. Käytännössä ero on siinä, että DIT tekee FFT-jaon valitsemalla joka n:nnen

datapisteen, kun taas DIF jakaa datan n:nään osaan.

Van Loan (1992) esittelee konvoluutioon (2.2.5) perustuvat kaksi algoritmia: Raderin algoritmi ja Bluesteinin algoritmi. Bluesteinin algoritmi tunnetaan myös nimellä *taajuusmuuttuva z-muunnos* (Chirp z-transform)[8] Tässä esitellään Bluesteinin algoritmi[9]:

Olkoon $T_n(a)$ Toeplitz-matriisi, jolle $a_j = \bar{\omega}_2^{j^2} n$, $j = -n+1 : n-1$. Tällöin seuraava algoritmi korvaa x:n $(x \in \mathbb{C}^n)$ $F_n x$:llä:

for j=0 : n-1

$a(j) \leftarrow e^{j^2 \pi i/n}$

$a(-j) \leftarrow a(j)$

$u(j) \leftarrow \bar{a}(j)x(j)$

end

$v \leftarrow T_n(a)u$ (lasketaan erillisellä algoritmilla)

$y \leftarrow \bar{a}(0 : n-1).*v$

Suurin osa laskennasta tapahtuu Toeplitz-matriisin muodostamisessa, johon Van Loan (1992) esitteli algoritmin[10], jonka vaativuus on FFT-algoritmeja vastaava.

2.2.3 Käänteismuunnos

Diskreetin käänteismuunnoksen kaavasta (2.1.1) seuraa, että käänteismuunnos saavutetaan yksinkertaisesti korvaamalla kaikki ω_n-kertoimet niiden konjugaateilla $\bar{\omega}_n$ ja kertomalla lopullinen muunnos termillä $1/n$, missä n on datapisteiden määrä.

8. Termi "sirinä" (chirp) tarkoittaa signaalia, jonka taajuus joko nousee tai laskee ajan myötä; z-muunnos on muunnos, joka muuntaa diskreetin aikasignaalin taajuussignaaliksi. Diskreetti Fourier-muunnos on z-muunnoksen erikoistapaus.

9. Sivu 210, Algorithm 4.2.3

10. Sivu 209, Algorithm 4.2.2

2.2.4 Kosinimuunnos

Van Loan (1992) esittelee seuraavan DCT–II-muunnosalgoritmin[11]:

Olkoon $x(0:m-1) \in \mathbb{R}^n$ ja $m = 2p$. Tällöin seuraava algoritmi ylikirjoittaa x:n sen DCT-II:lla:

$f \leftarrow F_m \begin{bmatrix} x(0:2:m-1) \\ E_p x(1:2:m-1) \end{bmatrix}$

for $k = 0:m-1$

$y_k \leftarrow Re(\omega_{4m}^k f_k)$

end

Tässä E_p kääntää kerrottavan matriisin sarakkeet päinvastaiseen järjestykseen[12].

2.2.5 Konvoluutio

Konvoluutiolle pätee $H_n(h)g = F_n^{-1} diag(F_n h) F_n x$. Tämän tuloksen avulla voidaan muodostaa esimerkiksi seuraava kantaluvun 2 algoritmi[13]:

Olkoon $g, h \in \mathbb{C}^n$ ja $n = 2^t$. Nyt seuraava algoritmi korvaa g:n $H_n(h)g$:llä:

for $q = t:-1:1$

$g \leftarrow A_q^T g$

$h \leftarrow A_q^T h$

end

$g \leftarrow g.*h$

for $q = 1:t$

$g \leftarrow \bar{A}_q g$

---

11. Sivu 243, Algorithm 4.4.6
12. Esim. $E_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
13. Sivu 207, Algorithm 4.2.1

end

$g \leftarrow g/n$

2.2.6 Moniulotteinen data

Kaikki ylläoleva koskee yksiulotteista dataa, mutta digitaalisessa kuvankäsittelyssä käsitellään yleensä 2D- tai 3D-dataa. Periaatteessa riittää muuntaa data peräjälkeen ulottuvuus toisensa jälkeen, mutta algoritmien toteuttamisessa on oltava yksiulotteista tapausta huolellisempi.

2.3 Muut muunnokset

Edellämainittujen lisäksi löytyy lukuisia muitakin muunnoksia, joita käytetään signaalinkäsittelyssä: esimerkiksi (diskreetit) aallokemuunnokset, Karhunen–Loève-muunnos ja Laplacen muunnokseen perustuva z-muunnos (Smith 1997).

Muiden kuin FFT:n ja DCT:n käsittely ohitetaan tässä tutkielmassa, koska kuvankäsittelyn kannalta FFT ja etenkin DCT ovat tärkeimmät () ja koska useimmat eroavat Fourier-muunnoksesta melko paljon eikä niitä voida laskea FFT-algoritmien avulla. Lisätietoa muunnoksista löytyy esimerkiksi (Yaroslavsky 2014):n artikkelista.

3 Signaalinkäsittely

Digitaaliset signaalit ovat funktioita, jotka välittävät tietoa systeemistä äärellisellä tiheydellä Priemer (1991). Signaalinkäsittely on signaalien analysointia ja muokkaamista. Smith (1997):n mukaan joitakin signaalinkäsittelyn keskeisiä työkaluja ovat DFT (etenkin FFT) ja konvoluutio. Young, Driggers ja Jacobs (2008) esittävät, että signaalinkäsittelyn perustyökalut ovat Fourier-muunnos, aallokemuunnos (wavelet transform) sekä äärellisen impulssivasteeseen ja Fourier-muunnokseen perustuvat suodattimet.

Kuvankäsittely on yksi signaalinkäsittelyn alakategoria. Kuvasignaali on siitä erityinen, että sen sisältämä tieto on *kooditettu*(encode) tilaan eikä aikaan. Tämän takia DFT:n "tulkinta"on kuvien tapauksessa hieman erilainen Smith (1997).

3.1 FFT signaalinkäsittelyssä

Tämä osio perustuu teokseen Smith (1997).

DFT:tä käytetään usein yhdellä kolmesta tavasta: signaalien taajuusspektrin laskemiseen, systeemien taajuusvasteen selvittämiseen impulssivasteesta tai toisinpäin, tai osana monimutkaisempia signaalinkäsittelytekniikoita. Signaalin taajuuspektri sisältää tietoa kyseisestä signaalista, kun taas impulssivaste ja taajuusvaste ovat täydellisiä kuvauksia vastaavista järjestelmistä.

Suodattimet ovat tärkeä osa signaalinkäsittelyä: niitä käytetään signaalien erottelemiseen ja entisöintiin (virheiden vähentämiseen). Halutunlainen lineaarinen suodatin voidaan suunnitella FFT:tä hyödyntäen. Suoraviivaisin tapa suodattaa signaali on konvolvoida sitä FFT:n avulla luodun suodattimen kanssa. Muunlaisiakin suodattimia toki on.

3.2 FFT kuvankäsittelyssä

Lineaarisia suodattimia käytetään kuvankäsittelyssä muun muassa ääriviivojen laadun parantamiseen, häiriön vähentämiseen, valaistuksen tasapainottamiseen, ja hämärryksen (*blur*) ja liikkeen aiheuttaman konvoluution korjaamiseksi (Smith 1997).

Tässä osiossa käsitellään FFT:n joitakin sovelluksia kuvankäsittelyssä mukaanlukien videokuva. Tämän tutkielman lista ei ole millään tavalla kattava.

Joissakin sovelluksissa käytetään DCT:tä - nämä sovellukset on sisälletty tässä, koska DCT on helpointa toteuttaa FFT:n avulla, ja toisaalta koska lohkoihin käytettävä DCT on käytetyin kuvan ja videon kooditukseen (encoding) käytetty muunnos (Bovik 2000).

3.2.1 JPEG

JPEG on väitetysti yksi suosituimmista kuvaformaateista (Abuzaher ja Al-Azzeh 2017; Hou ym. 2018; Liu ym. 2018)), mutta tieteellistä tutkimusta väitteestä on vaikea löytää. Verkkosivun *Media | 2019 | The Web Almanac by HTTP Archive* (2020) vuonna 2019 julkaistun raportin mukaan Internetin kuvahauista 60 % ja kuvatavuista 65 % on JPEG-muodossa, kun taas toisen verkkosivun W3Techsin *Historical trends in the usage statistics of image file formats for websites, December 2020* (2020) mukaan 31.12.2020 72.3 % Internetin 10 miljoonalla suosituimmalla sivulla oli JPEG-kuvia.

Häviöllinen JPEG-kuvanpakkaus perustuu DCT:hen (Wallace 1992). Kuvat jaetaan ensin 8×8-lohkoihin ja jokainen lohko käsitellään erikseen DCT:llä. Harmaakuville riittää yksi käsittely DCT:llä - värikuvien tapauksessa voidaan joko käsitellä eri värit (esimerkiksi RGB) erikseen tai "lomittain"käsittelemällä eri värejä eri lohkoissa. Saatu taajuusjakauma pakataan ensin kvantisoimalla (häviöllinen prosessi) ja sitten entropiakoodauksella (häviötön prosessi).

Kuvanpakkaus puretaan tekemällä edelliset askeelet päinvastaisessa järjestyksessä ja korvaamalla DCT sen käänteismuunnoksella (*IDCT*).

3.2.2 Videostandardit

Videokuva koostuu jonosta kuvia, jotka näytetään katsojalle nopealla tahdissa. Yleensä videokuva pakataan, koska muuten se vie liian paljon tilaa. Tällä hetkellä suosituin pakkausstandardi on H.264, joka tunnetaan myös nimellä MPEG osa 10 (Punchihewa ja Bailey 2020). Se ja sen edeltäjät H.262 ja H.263 perustuvat H.261-standardiin (Sullivan ja Wiegand 2005).

H.261 ja sen seuraajat perustuvat osittain DCT:hen (Wiegand ym. 2003) (Girod, Steinbach

ja Faerber 1995). Standardeissa sitä käytetään kuvien pakkaamiseen jakamalla kuvat 8x8-lohkoihin (H.261-H.263) tai mahdollisesti 4x4-lohkoihin (H.264), jotka syötetään DCT:lle (Girod, Steinbach ja Faerber 1995) (Wiegand ym. 2003). Jokaista yksittäistä videokuvaa ei kannata pakata (Sullivan ja Wiegand 2005), ja standardeissa käytetäänkin DCT:n lisäksi muitakin pakkauskeinoja (Sullivan ja Wiegand 2005) (Wiegand ym. 2003) (Girod, Steinbach ja Faerber 1995).

H.261-H.263 tapauksessa ainoastaan käänteisen DCT:n (IDCT) tarkkuudelle on annettu parametreja (Sze, Budagavi ja Sullivan 2014). H.264:n tapauksessa sekä DCT:lle että IDCT:lle annetaan kokonaislukuapproksimaatiot, joita standardin toteutuksissa on käytettävä (Wiegand ym. 2003).

Myös melko uudessa ja suosiota keräävässä HEVC-standardin (High Efficiency Video Coding) toteutuksissa tulee käyttää määrättyjä IDCT:n approksimaatioita lohkokoilla 4×4, 8×8, 16×16 ja 32×32, tai vaihtoehtoisesti diskreettiä käänteistä 4×4-sinimuunnosta (Sze, Budagavi ja Sullivan 2014).

Vaikka DCT onkin osa kaikissa H.261–H.264-standardeja, sen käyttötapa niissä ei ole olennaisesti muuttunut (lohkokokoa lukuunottamatta) ja suurin osa kyseisten koodekkien parannuksista on liittynyt liikekorjattuun ennustukseen (motion-compensated prediction) (Sullivan ja Wiegand 2005).

3.2.3 Muut sovellukset

R.W. Cox ja Raoqiong (1999) ehdottivat, että aikakehittyvän taajuuden z-muunnosta (chirp z-transform) magneettiresonanssikuvien (NMR image) nopeaan pyörittämiseen 2D- ja 3D-tasoissa tarkasti. DFT on kyseisen muunnoksen erityistapaus. Cox ja Tong esittelevät, miten z-muunnos voidaan laskea FFT:tä hyödyntäen. NMR-kuvien pyörittäminen on tärkeää kuvattavien potilaiden liikkeen takia.

Li, Mueller ja Ernst (2004) vertasivat z-muunnosta ja FFT-kirjaston avulla toteutettua signaalien *uudelleennäytteistämistä* (resampling). Uudelleennäytteistämistä tarvitaan kun kuvia suurennetaan, käännetään, siirretään, tai väännetään; sekä muun muassa CT-, PET- ja MRI-kuvien kohdentamisessa. Tutkielmassa todettiin, että z-muunnos ja FFT-kirjasto saavuttivat samankaltaisen tarkkuuden, mutta FFT-kirjasto oli huomattavasti nopeampi. Kuvien käsittelyssä kuvat siirrettiin taajuustasolle

FFT-muunnoksella, muokattiin matemaattisilla operaatioilla ja muunnettiin takaisin.

Lve Huang ym. (2020) esittelivät, miten FFT:tä voidaan hyödyntää liikkuvan kuvan sumentumisen vähentämiseksi. Esitellyssä algoritmissa kuvaan käytettiin FFT:tä kaksi kertaa peräkkäin, jotta saatiin tarkka arvio siitä, kuinka paljon kuva oli sumentunut. Tämän jälkeen kuvasta poistettiin tilastollisella optimointialgoritmilla ylimääräiset taajuudet. Menetelmä toimi tutkielman testeissä hyvin.

Zahedi ja Ghadi (2015) yhdistivät FFT:n ja Gabor-suodattimen sormenjälkikuvien tunnistamisessa. Gabor-suodatin on parempi löytämään rikkinäiset ja haarautuvat "harjanteet", kun taas FFT on parempi yhdistämään "katkenneet" harjanteet ja täyttämään sormenjälkikuvan reikiä. Menetelmässä sormenjälkikuva jaetaan osioihin ja jokaiseen osioon sovelletaan joko FFT:tä tai Gabor-suodatinta riippuen siitä, kumpi "korjaa"osion paremmin. Tutkielma löysi, että menetelmä saavutti parempia tuloksia kuin asiantuntija, tai vain FFT:n tai Gabor-suodattimen käyttö.

Gil-Jimenez ym. (2005) käyttivät FFT:tä liikennemerkkien *luokitteluun* (classification). Aluksi kuvista eristetään kaikki muodot, jotka saattavat olla liikennemerkkejä, ja sen jälkeen nämä alueet esikäsitellään. Esikäsitellyt alueet muunnetaan FFT:llä ja verrataan valmiiksi laskettuihin tunnettujen liikennemerkkien FFT-kuviin. Menetelmää sovellettiin noin 300 kuvan testikuvajoukkoon ja sillä saavutettiin kohtuullisen hyviä tuloksia.

Tian ym. (2018) pyrkivät parantamaan kasvontunnistusta liittämällä FFT:n avulla lasketun taajuusjakauman harvassa luokittelijassa. Aikaisemmissa menetelmissä luokittelijaa opetettiin pelkästään muokkaamattomien kuvien avulla, mutta tutkielmassa kehitettiin uusi pisteytystapa, joka hyödynsi kuvien taajuusjakaumaa. Menetelmä todettiin tehokkaaksi ja vakaaksi.

Kruse, Rother ja Schmidt (2017) pyrkivät poistamaan vähentämään kuvien sumentumista eli *dekonvolvoimaan* kuvia. Aikaisemmissa iteratiivissa dekonvoluutiomenetelmissä joka iteraatiolla FFT:tä sovellettiin kuvasta suodattimilla ja muutenkin laskettuihin funktioihin, jotka yhdistettiin uudeksi funktioksi joka syötettiin käänteiselle FFT:lle jota käytettiin seuraavan iteraation laskemiseksi. Tutkielmassa osa funktioista korvattiin konvoluutioneuroverkoilla, eli käänteisen FFT:n syöte muuttui.

Wang ym. (2006) käyttivät FFT:tä osana häiriönpoistoa kuvissa. Tutkielmassa muokataan

aikaisempaa menetelmää, jossa lasketaan kuvan kaikkien pikselien painotettu keskiarvo, jota käytetään häiriöiden poistamiseen. Aikaisempi menetelmä muunnetaan muotoon, jossa esiintyy vain summia ja konvoluutio. FFT:n osuus menetelmässä on muuntaa konvoluutio yksinkertaiseksi kertolaskuksi.

Kuvien kohdistamisessa eli niiden *koordinaattijärjestelmien yhdentämisessä* (image registration) on olennainen kuvankäsittelytehtävä (Brown 1992). FFT:tä voidaan käyttää tähän tarkoitukseen, kuten (Averbuch ja Keller 2002) esittelee. Averbuchin ja Kellerin algoritmissa lasketaan kahden kuvan korrelaatio Fourier-muunnosta hyödyntäen ja sen jälkeen iteratiivisesti siirretään kuvia Fourier-muunnoksiin perustuvilla operaatioilla, kunnes loppuehto täyttyy.

Zear, Singh ja Kumar (2018) käyttivät diskreetin aallokemuunnoksen (DWT), DCT:n ja singulaariarvohajotelman (SVD) yhdistelmää terveydenhuollossa käytettävien kuvien *vesileimaamiseen* (watermarking). Esitellyssä menetelmässä vesileimattava kuva syötettiin DWT:lle. DWT:n ulostulosta valittiin yksi taajuusalue, joka syötettiin DCT:lle ja sitten SVD:lle. Tähän ulostuloon lisättiin kolme erilaista vesileimatyyppiä, joista yhdessä käytettiin DCT:tä ja SVD:tä. Vesileimattu taajuusalueeseen käytettiin sen jälkeen järjestyksessä käänteistä SVD:tä, käänteistä DCT:tä ja käänteistä DWT:tä. Menetelmän vesileimat todettiin huomaamattomammiksi, kestävämmiksi ja "tilavammiksi" kuin jos kuvat olisi vesileimattu pelkästään DWT:llä, DCT:llä tai SVD:llä.

Li ja Wyrwicz (2018) tekivät laitteistopohjaisen FFT-algoritmitoteutuksen ja raportoivat suunnitelmasta ja toteutuksesta rinnakkaiselle 2D-FFT:lle FPGA-piireissä.

4 Yhteenveto

Tässä tutkielmassa käsiteltiin Fourier-muunnoksen, sen tehokkaan toteutuksen ja kuvankäsittelyn yhteyttä.

Fourier-muunnos on matemaattisella tasolla hyvin tunnettu ja tutkittu menetelmä ja sen käyttöönottoon on kehitetty useita eri algoritmeja. Uusin FFT-kirjallisuus vaikuttaa keskittyvän FFT-algoritmien hyödyntämiseen osana monimutkaisempia algoritmeja. Laitteistopohjaisia FFT-algoritmitoteutuksia kehitetään myös jonkin verran - vaikka ne eivät olleetkaan tutkielman keskiössä, yksi esimerkki mainittiin.

Tutkielmassa viitattujen tutkimuksista nousi esille se, että niissä käytettiin aina jotain muuta menetelmää Fourier-muunnokseen pohjautuvien menetelmien lisäksi. Konvoluutio tai FFT-algoritmi yksinään eivät ole kuvankäsittelyä, vaan enemmänkin välttämättömiä astinkiviä, joita tarvitaan monimutkaisempien menetelmien mahdollistamiseksi.

Fourier-muunnosta ei myöskään käytetty uusien kuvien luomiseen, vaan nimenomaan olemassaolevien kuvien käsittelyyn tavalla tai toisella.

Yleisesti ottaen Fourier-muunnosta käytetään myös kuvankäsittelyssä siihen, että jokin ongelma muutetaan sellaiseen muotoon, jossa sen omainaispiirteet on ilmaistu selkeällä ja käsiteltävällä tavalla. Ongelma ratkaistaan yksinkertaisessa mutta ihmisepäystävällisessä muodossa ja muunnetaan takaisin ihmisystävälliseen muotoon.

Lähteet

Abuzaher, Mazen, ja Jamil Al-Azzeh. 2017. “JPEG Based Compression Algorithm” [kielellä en]. 4 (4): 4.

Alt, René. 1978. “Error propagation in fourier transforms”. *Mathematics and Computers in Simulation* 20, numero 1 (maaliskuu): 37–43. ISSN: 0378-4754, viitattu 7. maaliskuuta 2019. doi:10.1016/0378-4754(78)90052-6.

Averbuch, A., ja Y. Keller. 2002. “FFT based image registration”. Teoksessa *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing,* 4:IV–3608–IV–3611. Toukokuu. doi:10.1109/ICASSP.2002.5745436.

Bovik, Alan C. 2000. *Handbook of Image and Video Processing.* Academic Press.

Brown, Lisa Gottesfeld. 1992. “A survey of image registration techniques”. *ACM Computing Surveys* 24, numero 4 (joulukuu): 325–376. ISSN: 0360-0300, viitattu 24. heinäkuuta 2020. doi:10.1145/146370.146374. https://doi.org/10.1145/146370.146374.

Cochran, W. T., J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, D. E. Nelson, C. M. Rader ja P. D. Welch. 1967. “What is the fast Fourier transform?” Conference Name: Proceedings of the IEEE, *Proceedings of the IEEE* 55, numero 10 (lokakuu): 1664–1674. ISSN: 1558-2256. doi:10.1109/PROC.1967.5957.

Gil-Jimenez, P., S. Lafuente-Arroyo, H. Gomez-Moreno, F. Lopez-Ferreras ja S. Maldonado-Bascon. 2005. “Traffic sign shape classification evaluation. Part II. FFT applied to the signature of blobs”. Teoksessa *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* 607–612. ISSN: 1931-0587. Kesäkuu. doi:10.1109/IVS.2005.1505170.

Girod, Bernd, Eckehard G. Steinbach ja Niko Faerber. 1995. “Comparison of the H.263 and H.261 video compression standards”. Teoksessa *Standards and Common Interfaces for Video Information Systems: A Critical Review,* 10282:102820D. International Society for Optics / Photonics, lokakuu. Viitattu 31. joulukuuta 2020. doi:10.1117/12.227952. https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10282/102820D/Comparison-of-the-H263-and-H261-video-compression-standards/10.1117/12.227952.short.

Gomes, Jonas. 2015. *From Fourier analysis to wavelets.* IMPA monographs. Pages: 1216. Cham, Switzerland: Springer. ISBN: 978-3-319-22075-8.

Historical trends in the usage statistics of image file formats for websites, December 2020. 2020. Viitattu 31. joulukuuta. https://w3techs.com/technologies/history_overview/image_format/all.

Hou, Dongdong, Haoqian Wang, Weiming Zhang ja Nenghai Yu. 2018. “Reversible data hiding in JPEG image based on DCT frequency and block selection” [kielellä en]. *Signal Processing* 148 (heinäkuu): 41–47. ISSN: 0165-1684, viitattu 31. joulukuuta 2020. doi:10.1016/j.sigpro.2018.02.002. http://www.sciencedirect.com/science/article/pii/S0165168418300483.

Kruse, Jakob, Carsten Rother ja Uwe Schmidt. 2017. “Learning to Push the Limits of Efficient FFT-Based Image Deconvolution”, 4586–4594. Viitattu 30. heinäkuuta 2020. https://openaccess.thecvf.com/content_iccv_2017/html/Kruse_Learning_to_Push_ICCV_2017_paper.html.

Li, A., K. Mueller ja T. Ernst. 2004. “Methods for efficient, high quality volume resampling in the frequency domain” [kielellä en]. Teoksessa *IEEE Visualization 2004,* 3–10. Austin, TX, USA: IEEE Comput. Soc. ISBN: 978-0-7803-8788-1, viitattu 31. joulukuuta 2020. doi:10.1109/VISUAL.2004.70. http://ieeexplore.ieee.org/document/1372173/.

Li, Limin, ja Alice M. Wyrwicz. 2018. “Parallel 2D FFT implementation on FPGA suitable for real-time MR image processing” [kielellä en]. Publisher: AIP Publishing LLCAIP Publishing, *Review of Scientific Instruments* 89, numero 9 (syyskuu): 093706. ISSN: 0034-6748, viitattu 30. heinäkuuta 2020. doi:10.1063/1.5019846. https://aip.scitation.org/doi/abs/10.1063/1.5019846.

Liu, Zihao, Tao Liu, Wujie Wen, Lei Jiang, Jie Xu, Yanzhi Wang ja Gang Quan. 2018. “DeepN-JPEG: a deep neural network favorable JPEG-based image compression framework”. Teoksessa *Proceedings of the 55th Annual Design Automation Conference,* 1–6. DAC ’18. New York, NY, USA: Association for Computing Machinery, kesäkuu. ISBN: 978-1-4503-5700-5, viitattu 31. joulukuuta 2020. doi:10.1145/3195970.3196022. https://doi.org/10.1145/3195970.3196022.

Lve Huang, Lushen Wu, Wenyan Xiao ja Qingjin Peng. 2020. “Deblurring approach for motion camera combining FFT with alpha-confidence goal optimization” [kielellä en]. *Optica Applicata* 50 (2). ISSN: 0078-5466, 1899-7015, viitattu 4. elokuuta 2020. doi:10.37190/oa200202. http://opticaapplicata.pwr.edu.pl/files/pdf/2020/no2/optappl_5002p185.pdf.

Media | 2019 | The Web Almanac by HTTP Archive. 2020. Viitattu 31. joulukuuta. https://almanac.httparchive.org/en/2019/media#image-formats.

Pickering, Morgan. 1986. *An introduction to fast Fourier transform methods for partial differential equations, with applications.* Electronic & electrical engineering research studies. Letchworth: Research Studies Press. ISBN: 978-0-471-91261-3.

Priemer, Roland. 1991. *Introductory Signal Processing* [kielellä en]. Google-Books-ID: QBT7nP7zTLgC World Scientific. ISBN: 978-9971-5-0919-4.

Punchihewa, A., ja D. Bailey. 2020. “A Review of Emerging Video Codecs: Challenges and Opportunities”. Teoksessa *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ),* 1–6. ISSN: 2151-2205. Marraskuu. doi:10.1109/IVCNZ51579.2020.9290536.

R.W. Cox ja Tong Raoqiong. 1999. “Two- and three-dimensional image rotation using the FFT”. *IEEE Transactions on Image Processing* 8, numero 9 (syyskuu): 1297–1299. ISSN: 1057-7149. doi:10.1109/83.784442.

Smith, Steven W. 1997. *The Scientist and Engineer's Guide to Digital Signal Processing.* Viitattu 30. lokakuuta 2019. http://www.dspguide.com/.

Sullivan, G.J., ja T. Wiegand. 2005. “Video Compression - From Concepts to the H.264/AVC Standard” [kielellä en]. *Proceedings of the IEEE* 93, numero 1 (tammikuu): 18–31. ISSN: 0018-9219, viitattu 31. joulukuuta 2020. doi:10.1109/JPROC.2004.839617. http://ieeexplore.ieee.org/document/1369695/.

Sze, Vivienne, Madhukar Budagavi ja Gary J. Sullivan, toimittaneet. 2014. *High Efficiency Video Coding (HEVC): Algorithms and Architectures* [kielellä en]. Integrated Circuits and Systems. Cham: Springer International Publishing. ISBN: 978-3-319-06894-7 978-3-319-06895-4, viitattu 22. joulukuuta 2020. doi:10.1007/978-3-319-06895-4. http://link.springer.com/10.1007/978-3-319-06895-4.

Tian, Chunwei, Qi Zhang, Guanglu Sun, Zhichao Song ja Siyan Li. 2018. “FFT Consolidated Sparse and Collaborative Representation for Image Classification” [kielellä en]. *Arabian Journal for Science and Engineering* 43, numero 2 (helmikuu): 741–758. ISSN: 2191-4281, viitattu 30. heinäkuuta 2020. doi:10.1007/s13369-017-2696-7. https://doi.org/10.1007/s13369-017-2696-7.

Wallace, G. K. 1992. “The JPEG still picture compression standard”. Conference Name: IEEE Transactions on Consumer Electronics, *IEEE Transactions on Consumer Electronics* 38, numero 1 (helmikuu): xviii–xxxiv. ISSN: 1558-4127. doi:10.1109/30.125072.

Van Loan, C. 1992. *Computational Frameworks for the Fast Fourier Transform.* Frontiers in Applied Mathematics. Society for Industrial / Applied Mathematics, tammikuu. ISBN: 978-0-89871-285-8, viitattu 24. helmikuuta 2019. doi:10.1137/1.9781611970999. https://epubs.siam.org/doi/book/10.1137/1.9781611970999.

Wang, J., Y. Guo, Y. Ying, Y. Liu ja Q. Peng. 2006. “Fast Non-Local Algorithm for Image Denoising”. Teoksessa *2006 International Conference on Image Processing,* 1429–1432. Lokakuu. doi:10.1109/ICIP.2006.312698.

Wiegand, T., G. J. Sullivan, G. Bjontegaard ja A. Luthra. 2003. “Overview of the H.264/AVC video coding standard”. Conference Name: IEEE Transactions on Circuits and Systems for Video Technology, *IEEE Transactions on Circuits and Systems for Video Technology* 13, numero 7 (heinäkuu): 560–576. ISSN: 1558-2205. doi:10.1109/TCSVT.2003.815165.

Yaroslavsky, Leonid P. 2014. *Fast Transforms in Image Processing: Compression, Restoration, and Resampling* [kielellä en]. Review Article. ISSN: 2356-6655 Pages: e276241 Publisher: Hindawi Volume: 2014, heinäkuu. Viitattu 15. joulukuuta 2020. doi:https://doi.org/10.1155/2014/276241. https://www.hindawi.com/journals/aee/2014/276241/.

Young, S. Susan, Ronald G. Driggers ja Eddie L. Jacobs. 2008. *Signal Processing and Performance Analysis for Imaging Systems.* Norwood, UNITED STATES: Artech House. ISBN: 978-1-59693-288-3, viitattu 31. joulukuuta 2020. http://ebookcentral.proquest.com/lib/jyvaskyla-ebooks/detail.action?docID=456883.

Zahedi, Morteza, ja Ozra Rostami Ghadi. 2015. “Combining Gabor filter and FFT for fingerprint enhancement based on a regional adaption method and automatic segmentation” [kielellä en]. *Signal, Image and Video Processing* 9, numero 2 (helmikuu): 267–275. ISSN: 1863-1711, viitattu 30. heinäkuuta 2020. doi:10.1007/s11760-013-0436-3. https://doi.org/10.1007/s11760-013-0436-3.

Zear, Aditi, Amit Kumar Singh ja Pardeep Kumar. 2018. “A proposed secure multiple watermarking technique based on DWT, DCT and SVD for application in medicine” [kielellä en]. *Multimedia Tools and Applications* 77, numero 4 (helmikuu): 4863–4882. ISSN: 1573-7721, viitattu 30. heinäkuuta 2020. doi:10.1007/s11042-016-3862-8. https://doi.org/10.1007/s11042-016-3862-8.