Ilkka Ollari

DIGITAALISTEN KUVAVÄÄRENNÖSTEN TUNNISTAMINEN JPEG-PAKKAUSTA HYÖDYNTÄEN

Tietojenkäsittelytieteen pro gradu -tutkielma
10.8.2010

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS

SISÄLTÖ

TIIVISTELMÄ

Ollari, Ilkka

Tietojenkäsittelytieteen pro gradu -tutkielma / Ilkka Ollari

Jyväskylä: Jyväskylän yliopisto, 2010, 100 s.

Pro gradu -tutkielma

Digitaalisten valokuvien muokkaamisesta on tullut hyvin helppoa ja yleistä, mikä on nostanut esille huolen valokuvien uskottavuudesta. Ongelman ratkaisuksi on esitetty erilaisia tekniikoita, joilla valokuvien aitoutta voidaan arvioida. Tarkastelen tässä tutkielmassa niin sanottuja passiivisia tekniikoita, jotka eivät edellytä kuvasta minkäänlaista ennakkotietoa, kuten vesileimaa. Vertailen viittä tällaista tekniikkaa laatimani neljä kohtaa käsittävän kriteeristön pohjalta. Vertailtavia ominaisuuksia ovat tarkkuus, tunnistetut muokkaustyypit, rajoitukset sekä vakaus vastatoimia vastaan. Vertailu osoittaa, että eräs merkittävimmistä tekniikoiden ongelmista on niiden huono toleranssi kuvan häviöllisen pakkaamisen suhteen. Koska häviöllisistä tallennusmuodoista käytetyin on JPEG, olen toteuttanut neljä valokuvien aitouden arviointimenetelmää, jotka hyödyntävät JPEG-pakkauksen ominaispiirteitä. Kolme tekniikoista on lähes sellaisenaan esitelty aiemmin alan kirjallisuudessa. Neljäs on oma muunnelmani yhdestä edellä mainituista. Kaksi menetelmistä hyödyntää JPEG-pakkauksen ytimeen kuuluvia diskreettiä kosinimuunnosta ja kvantisointia. Toiset kaksi puolestaan tarkastelevat JPEG-pakkauksen aiheuttamaa kuvan laatikoitumista. Tutkielman tarkoituksena on selvittää, ovatko useamman tekniikan avulla saavutettavat tulokset oleellisesti luotettavampia, kuin yksittäisellä menetelmällä saavutettavat. Esitän myös testitulokset, jotka olen saanut testaamalla tekniikoita laatimani kuva-aineiston suhteen. Testitulosten perusteella voidaan todeta, että useamman tekniikan avulla kuvan aitoudesta saadaan kokonaisvaltaisempi kuva, ja hieman useampia tilanteita saadaan todella katettua. Joiltain osin tarkasteltujen tekniikoiden toimintaedellytykset poikkeavat toisistaan, mutta kaikkien tekniikoiden kanssa törmätään kuitenkin lopulta samaan rajoitukseen. Merkittävin tekijä kuvaväärennösten onnistuneen tunnistamisen kannalta näyttäisi olevan ennen ja jälkeen kuvan muokkaamisen käytetyt pakkauslaadut. Onnistunut tunnistaminen edellyttää, että muokkaamisen jälkeinen laatu on ainakin jonkin verran muokkaamista edeltävää laatua korkeampi. Määriteltäessä JPEG-pakkauksen laatu asteikolla 1-100, näyttäisi 1-20 yksikön ero riittävän. Vaadittava ero on pienin pakkauslaadun korkeimmilla tasoilla. Muita vaikuttavia tekijöitä ovat muokkauksen tyyppi, kuvan sisältö sekä muokatun alueen koko.

AVAINSANAT: digitaalinen kuva, kuvankäsittely, kuvaväärennös, digitaalisten valokuvien autentikointi, digital image forensics, JPEG

1 JOHDANTO

Digitaalitekniikka on mullistanut valokuvien käyttöä monella tavalla. Digitaalisten kameroiden yleistymisen myötä valokuvia syntyy jatkuvasti valtavia määriä, ja internetin välityksellä entistä suurempi osa kuvista päätyy varsin suuren ihmisjoukon saataville. Samanaikaisesti myös kuvien erilainen käsittely on yleistynyt, mikä on nostanut esille huolen valokuvien uskottavuudesta. Aina ei voida itsestään selvästi olettaa, että valokuva olisi luotettava todiste todellisista tapahtumista.

Kuvien käsittely sinänsä ei ole mitenkään uusi ilmiö. Valokuvia on käsitelty lähestulkoon yhtä kauan kuin niitä on ollut olemassa. Aiemmin pimiöissä tapahtunut kuvien käsittely oli kuitenkin varsin työlästä ja hidasta, ja vaati huomattavan paljon ammattitaitoa. (Farid 2009b) Digitaalitekniikka on tuonut kuvankäsittelyn entistä useampien ihmisten ulottuville ja tarjoaa lähes rajattoman valikoiman keinoja kuvien muokkaamiseen. Kuvankäsittely ei ole enää vain ammattilaisten yksinoikeus, eikä edellytä ammattimaisia välineitä. Suhteellisen vaativakin kuvankäsittely onnistuu mainiosti tavallisella kotitietokoneella, ja varsin laadukkaita ja helppokäyttöisiä kuvankäsittelyohjelmia on nykyään saatavilla jopa täysin maksutta. Kohtuullisen monipuolisia ilmaisia kuvankäsittelyohjelmia ovat muun muassa Gimp ja Paint.NET. Varmasti kaikkein tunnetuin kuvankäsittelyohjelmista on kuitenkin Adoben Photoshop, josta on arkikielessä muodostunut eräänlainen synonyymi kuvankäsittelylle.

Valokuvien voima ja yhteiskunnallinen merkitys on joka tapauksessa huomattava. Valokuvia voidaan käyttää muun muassa todisteina oikeuskäsittelyissä, uutisoinnin apuna tiedostusvälineissä, propagandatarkoituksissa sekä tieteellisten tulosten raportoinnissa. Muokattujen kuvien käytöllä edellä mainituissa tilanteissa voisi olla hyvin vakavia seurauksia.

Varsin usein väärennettyjä kuvia voi olla mahdotonta erottaa aidoista pelkästään silmämääräisen arvion perusteella. Siksi valokuvien aitouden selvittämiseen tarvitaan entistä tehokkaampia välineitä. Ongelmaan on kaksi lähtökohtaisesti erilaista lähestymistapaa. Perinteisempää mallia edustavat niin sanotut *aktiiviset menetelmät.* Tärkein aktiivisten menetelmien ryhmä on *vesileimatekniikat*. (mm. Zhang, Ren, Ping, He & Zhang 2008) Vesileimojen ajatuksena on se, että kuvan ottamisen hetkellä kuvaan upotetaan jonkinlainen normaalitarkastelussa näkymätön tunnus, jonka avulla kuvan aitous voidaan myöhemmin todentaa (Mohanty 1999). Vesileimatekniikoita on tutkit-

tu suhteellisen paljon, ja niiden avulla kuvien aitous voidaankin varmistaa jo varsin luotettavasti. Vesileimatekniikoiden oleellisin rajoite on kuitenkin niiden soveltuvuus vain vesileimalla varustettuihin kuviin (mm. Popescu 2005, 8). Tällaisten kuvien osuus olemassa olevista kuvista lienee häviävän pieni, sillä vesileiman lisäämisen mahdollisuus kuluttajamarkkinoilla olevissa kameroissa on varsin poikkeuksellinen. Vesileiman lisääminen voisi toki tapahtua myös erillisellä ohjelmalla kuvan ottamisen jälkeen. Tällöin on kuitenkin olemassa se mahdollisuus, että kuvaa muokataan ennen vesileiman asettamista.

Tässä tutkielmassa keskityn uudempaa suuntausta edustaviin, niin sanottuihin *passiivisiin menetelmiin.* Tällaisten menetelmien lähtökohtana on se, että tutkittavasta valokuvasta ei pääsääntöisesti tarvita minkäänlaista ennakkotietoa, vaan kuvan aitouden arviointi perustuu yleensä jonkinlaiseen tilastolliseen päättelyyn.(Zhang ym. 2008) Tämän tyyppisten arvioiden tekemisen mahdollistaa se, että tunnetaan tiettyjä ominaisuuksia, jotka ovat yhteisiä kaikille digitaalisille valokuville. Toisaalta tiedetään myös se, millä tavoin tietyt käsittelytoimenpiteet muuttavat kuvaa. Käytännössä tämä tarkoittaa yleensä sitä, että muokkauksen yhteydessä normaalit kuvan säännönmukaisuudet rikkoutuvat, tai vastaavasti kuvaan muodostuu uusia, epäluonnollisia säännönmukaisuuksia. Vesileimatekniikoihin verrattuna passiivisten autentikointimenetelmien avulla saavutettavat tulokset ovat kuitenkin jonkin verran epävarmempia.

Passiivisia menetelmiä sekä niiden tutkimusta tarkoittamaan on englanninkielisessä kirjallisuudessa omaksuttu termi *digital image forensics* (mm. Fridrich 2009). Tällä viitataan valokuvien rikostekniseen tutkintaan. Tässä tutkielmassa käytän termejä valokuvien aitouden arvioiminen, valokuvien autentikointi ja kuvaväärennösten tunnistaminen. Tutkimusalue on varsin tuore; ennen 2000-lukua julkaistuja tutkimuksia ei juurikaan ole saatavilla. Viime vuosina aihe on kuitenkin herättänyt runsasta mielenkiintoa.

Tutkielman alkupuolella luon katsauksen yleisimpiin ja varteenotettavimpiin valokuvien autentikointimenetelmiin. Menetelmien vertailua varten esitän neljä kohtaa käsittävän arviointikriteeristön. Yleisen toimintaperiaatteen lisäksi tarkasteltavia ominaisuuksia ovat menetelmien tarkkuus, tunnistettavat muokkaustyypit, rajoitukset sekä vakaus vastatoimia vastaan. Vertailun tuloksena voidaan havaita, että osa menetelmistä yltää jopa lähelle 100 prosentin tunnistustarkkuutta, mikäli tarkasteltavien kuvien laatu on riittävän

hyvä. Vertailu kuitenkin osoittaa myös sen, että eräs merkittävimmistä aitouden arviointia haittaavista tekijöistä on kuvien häviöllinen pakkaaminen. Eri autentikointimenetelmät vaihtelevat jossain määrin sen suhteen, kuinka herkkiä ne ovat kuvan laadun heikkenemiselle, mutta silti vain hyvin vähäinen heikennys kuvanlaadulle tulee yleensä kyseeseen.

Johtuen valokuvien alati kasvavasta tilantarpeesta, häviöllinen pakkaaminen on kuitenkin täysin rutiininomainen käytäntö. Monet digitaalikamerat eivät esimerkiksi tarjoa lainkaan mahdollisuutta tallentaa kuvia pakkaamattomassa muodossa. Käytetyin standardi kuvien häviölliseen pakkaamiseen on JPEG (Miano 1999, 35). Vaikka JPEG-pakkaus vaikeuttaakin usein valokuvien aitouden arvioimista, voivat sen ominaispiirteet myös tarjota erilaisia mahdollisuuksia valokuvien autentikointitarkoituksiin.

Tässä tutkielmassa esittelen neljä toteuttamaani valokuvaväärennösten tunnistamiseen tarkoitettua tekniikkaa, jotka hyödyntävät eri tavoin JPEG-pakkausta. Tekniikoista kolme on aiemmin lähes sellaisenaan esitelty alan kirjallisuudessa. Neljäs on oma muunnelmani yhdestä näistä. Menetelmistä kaksi hyödyntää JPEG-pakkauksen ytimeen kuuluvia diskreettiä kosinimuunnosta ja kvantisointia. Toiset kaksi puolestaan hyödyntävät JPEG-pakkauksen kuvaan aiheuttamaa laatikoitumista.

Tutkielman tarkoituksena on selvittää, onko useammalla menetelmällä saavutettavat tulokset ratkaisevasti luotettavampia, kuin yksittäisellä menetelmällä saavutetut. Lisäksi esitän joitakin mahdollisuuksia menetelmien yhteistoiminnalle.

Esitän myös testitulokset, jotka on saatu testaamalla toteutettuja autentikointimenetelmiä testiaineiston suhteen. Itse muodostamani testiaineisto koostuu valokuvista, jotka vaihtelevat käytetyn muokkaustyypin ja pakkauslaadun suhteen. Testien perusteella voidaan todeta, että useamman autentikointimenetelmän käyttö tarjoaa kokonaisvaltaisemman arvion kuvan aitoudesta ja kuvaan mahdollisesti kohdistuneista käsittelytoimenpiteistä. Menetelmien toimintaedellytyksissä on joitakin eroavaisuuksia, ja voidaan todeta, että eri menetelmät todella soveltuvat jossain määrin erilaisiin tilanteisiin. Toisin sanoen useamman menetelmän avulla saadaan katettua hieman useampia tilanteita. Toisaalta testien perusteella käy myös ilmi, että kaikkien tarkasteltujen menetelmien kanssa törmätään samaan ilmiöön. Kaikkein merkittävin tekijä väärennösten onnistuneen tunnistamisen kannalta näyttäisi olevan ennen ja jälkeen muokkaamisen käytetyt pakkauslaadut.

Tarkemmin sanoen muokkaamisen jälkeen käytetyn pakkauslaadun tulisi olla ainakin hieman ennen muokkaamista käytettyä laatua korkeampi. Vaadittava ero näyttäisi riippuvan myös pakkauslaadun lähtötasosta. Tarkasteltaessa JPEG-pakkauksen laatua asteikolla 1-100, riittää korkeimmilla laatutasoilla (yli 90) jopa 1-4:n yksikön ero onnistuneisiin tunnistustuloksiin. Alhaisemmilla tasoilla (60-70) tarvitaan usein jopa 20 yksikön ero pakkauslaaduissa. Taulukkoon 14 (s. 76) on listattu vaadittavat laatuerot lähtötasosta riippuen. Taulukoissa 9-12 (s. 67-68) on esitetty varsinaiset testitulokset. Myös näistä taulukoista käy ilmi laatuerojen vaikutus eri lähtötasoilla.

Muita lopputulokseen vaikuttavia tekijöitä ovat käytetyn muokkauksen tyyppi, kuvan sisältö ja muokatun alueen koko. Näiden tekijöiden vaikutus korostuu erityisesti silloin, kun ennen ja jälkeen muokkauksen käytetyt pakkauslaadut ovat lähellä toisiaan.

Tämän tutkielman toisessa luvussa esittelen yleisiä digitaalisiin kuviin ja kuvankäsittelyyn liittyviä seikkoja ja käsitteitä. Kolmannessa luvussa esittelen digitaalisten valokuvien aitouden arviointiin kehitettyjä tekniikoita ja esitän joukon kriteereitä niiden vertailuun. Neljännessä luvussa esittelen toteuttamani valokuvien autentikointitekniikat ja toteutukseen liittyvät tekniset yksityiskohdat. Viidennessä luvussa esittelen testitulokset, jotka toteutettujen menetelmien avulla on saavutettu. Kuudennessa luvussa esitän yhteenvedon koko tutkielmasta.

2 DIGITAALISET KUVAT JA KUVANKÄSITTELY

2.1 Kuvatiedon esittäminen digitaalisessa muodossa

Kuva voidaan ajatella kaksiulotteisena funktiona *f(x, y)*, jossa *x* ja *y* ovat spatiaalisia koordinaatteja, ja *f*:n arvo vastaa kuvan kirkkautta tai *intensiteettiä* missä tahansa pisteessä *x*, *y*. Kun sekä *x* ja *y,* että *f*:n arvot ovat äärellisiä, diskreettejä arvoja, sanotaan kuvaa digitaaliseksi. (Gonzales & Woods 2002, 54) Tarkkaan ottaen edellä mainittu funktio edellyttäisi värikuvien osalta lisämäärittelynä värikanavan (esim. *fr(x, y), fg(x, y), fb(x, y)*). Yhden funktion esitys voisi olla esimerkiksi $f:[1:X] \times [1:Y] \rightarrow [0:B]$, missä *B* vastaisi arvoa, joka sisältää jokaista värikanavaa koskevan informaation. Kunkin yksittäisen värikanavan arvo olisi tällöin johdettava *B*:n arvosta.

Digitaalisen kuvatiedon esittämiseen on kaksi erilaista periaatetta. Kuvia voidaan esittää *vektorigrafiikkana* tai *bittikarttoina*. Vektorikuvat sisältävät matemaattisista funktioista ja koordinaateista koostuvia piirto-ohjeita, joiden perusteella kuva rakennetaan. Bittikarttakuva sen sijaan esitetään ikään kuin matriisina, jossa kukin solu edustaa yhtä kuvapistettä, eli *pikseliä.* Pikseli on kuvan pienin mahdollinen rakenneyksikkö ja sillä on aina jokin väriarvo, tai kuten sanottu, värikuvan tapauksessa useita väriarvoja. Vektorimuotoisten kuvien etuna on se, että niiden sisältämiä elementtejä on helppo lisätä, poistaa ja muotoilla. Myös erilaiset geometriset operaatiot, kuten koon muuttaminen, onnistuvat kuvan laadun kärsimättä. Vektorigrafiikkaa ei kuitenkaan voida käyttää hyvin monimutkaisen kuvainformaation, kuten valokuvien esittämiseen. Bittikarttakuvat puolestaan soveltuvat hyvin valokuvien esittämiseen, mutta vektorikuviin verrattuna niiden käsittely on monessa suhteessa ongelmallisempaa. (Miano 1999, 1-3)

Kuvan värien esittämiseen on olemassa erilaisia *värimalleja*. Kaikkein yleisin näistä on niin sanottu *RGB-malli,* jossa kaikki käytettävissä olevat värit muodostetaan punaisen, vihreän ja sinisen osakomponentin avulla. Komponenttien mahdolliset arvot riippuvat käytetystä *näytetarkkuudesta*, eli yhtä komponenttia kuvaavien bittien lukumäärästä. Kaikkein tavallisimmin käytetty näytetarkkuus on kahdeksan bittiä, joka tarkoittaa sitä, että kukin osakomponentti voi saada 256 eri arvoa. Näin ollen yhden pikselin väriarvo kuvataan yhteensä 24 bitin avulla. Värien kuvaamiseen voidaan käyttää suurempaakin bittimäärää, mutta johtuen ihmissilmän rajallisesta kyvystä erottaa värejä, tälle ei välttämättä ole erityisiä perusteita. (Miano 1999, 5-9)

Muita suhteellisen yleisiä värimalleja ovat *YCbCr* sekä väritulostuksessa tyypillisesti käytetty *CMYK*. YCbCr-mallissa värit kuvataan yhden kirkkauskomponentin sekä sinisen ja punaisen värikomponentin avulla. Näistä *Y*, eli kirkkauskomponentti on kaikkein merkittävin ja se yksistään vastaa kuvan täydellistä mustavalkoesitystä. CMYK-mallissa värien kuvaamiseen käytetään neljää värikomponenttia, jotka ovat syaani, aniliininpunainen, keltainen ja musta. Edellä luetelluista värimalleista CMYK on *subtraktiivinen*, muut ovat *additiivisia*. Additiivisissa värimalleissa suuremmat värikomponenttien arvot lisäävät kuvan valoisuutta. Toisin sanoen mitä suuremmat värikomponenttien arvot ovat, sitä lähempänä väri on valkoista. Subtraktiivisissa malleissa tilanne on päinvastainen, eli suuret arvot lisäävät värin tummuutta. (Miano 1999, 6-8).

2.2 Tässä tutkielmassa käytetyt merkintätavat

Käytän tässä tutkielmassa edellä jo esiteltyä tapaa viitata kuvan väriarvoihin *x*- ja *y*-koordinaattien avulla. Piste $f(0, 0)$ viittaa kuvan vasempaan yläkulmaan. Jatkossa esiintyvissä merkinnöissä *x*-koordinaatti ilmoitetaan aina ennen *y*-koordinaattia. Funktion arvo puolestaan on oletuksena kokonaisluku väliltä 0-255.

Muun muassa erilaisten algoritmien kuvaukseen on käytetty matemaattisia kaavoja, jotka ovat yleensä sellaisenaan lainattu lähdeaineistosta. Joitakin kaavoja olen kuitenkin muotoillut yhdenmukaisuuden säilyttämiseksi. Esimerkiksi joissakin tapauksissa kuvan *x*- ja *y*-koordinaattien esitysjärjestys on alkuperäislähteissä ollut käänteinen.

2.3 Kuvan taajuusesitys

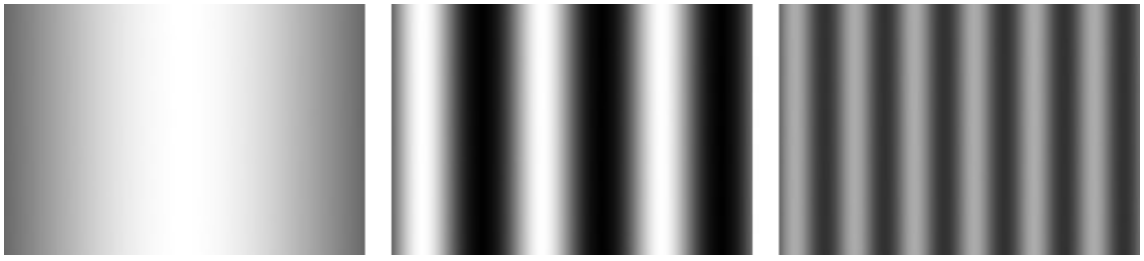
Digitaaliset kuvat voidaan esittää pikseli-informaation sijaan myös taajuuksina. Tämä voi olla tarkoituksenmukaisempi muoto esimerkiksi joidenkin kuvankäsittelytoimenpiteiden, kuten kuvan pakkaamisen kannalta. Kuvan taajuusesitys kuvaa käytännössä sitä, kuinka nopeasti kirkkaus vaihtelee kuvan sisällä. Matalat taajuudet vastaavat hitaita kirkkauden vaihteluita. Kaikkein hitaimmin vaihteleva taajuuskomponentti vastaa kuvan kirkkauden keskiarvoa. Korkeat taajuudet puolestaan vastaavat nopeampia kirkkauden vaihteluita. (Gonzales & Woods 2002, 156)

Kuvan spatiaalinen taajuus voidaan hahmottaa paremmin, tutkimalla seuraavaa funktiota:

$$f(x,y) = 128 + A\sin\left(\frac{2\pi ux}{N-1} + \phi\right) \quad .$$

Jos edellinen funktio piirretään kuvaksi, muodostuu sinimuotoinen x-akselin suuntainen variaatio. *Y*-koordinaatin arvot eivät ole varsinaisesti merkityksellisiä. Funktion tuloksena saadaan reaaliluku, joka vastaa pisteen *(x, y)* kirkkautta. Käytännössä tulos on pyöristettävä kokonaisluvuksi. Vaihtelu tapahtuu arvon 128 molemmin puolin. Amplitudi *A* määrittelee värähdysliikkeen laajuutta, eli tässä tapauksessa kirkkauden vaihtelua. *N* on kuvan leveys pikseleissä. *u* puolestaan määrittelee sen, kuinka monta täyttä sykliä kuvan alueelle mahtuu. ϕ määrittelee vaiheen, eli syklin aloituskohdan. Jos edellisille parametreille annetaan arvot $N = 100$, $u = 3$, $A = 127$ ja $\phi = 0$, muodostuu kuvaan kolme täyttä sykliä, ja taajuudeksi saadaan 0,03 sykliä / pikseli. Kuvan kirkkaus puolestaan vaihtelee välillä 0-255. (Efford 2000, 189)

Kuvassa 1 on kolme erilaista kuviota, jotka on luotu edellisen yhtälön avulla, parametreja vaihdellen. Tällä tavoin luodut kuvat eivät tietenkään sellaisenaan näytä kovinkaan luonnollisilta, mutta useita funktioita summaamalla voidaan muodostaa kuinka monimutkaisia kuvia hyvänsä. Siten esimerkiksi terävät reunat on mahdollista esittää, kun eri taajuisia siniaaltoja summataan riittävästi. Monimutkaisempia kuvia muodostettaessa on tietenkin otettava huomioon värähdysliikkeen suunta myös y-akselin suhteen. (Efford 2000, 191-192)



Kuva 1: Kolme sinifunktion avulla muodostettua kuvaa eri parametreilla. Vasemmalla A = 127 ja u = 0,5. Keskellä A = 127 ja u = 3. Oikealla A = 60 ja u = 6.

Kuvan taajuusesitys perustuu Ranskalaisen matemaatikon Joseph Fourier'n 1800-luvulla esittämään teoreemaan, jonka mukaan jokainen jaksoittainen funktio voidaan esittää sini ja kosinifunktioiden summana (Gonzales & Woods 2002, 148). Näiden niin sanottujen

perusfunktioiden summaa kutsutaan *Fourier'n sarjaksi*. Funktioiden painokertoimia puolestaan kutsutaan *Fourier-kertoimiksi*. (Efford 2000, 191)

Myös funktiot, jotka eivät ole jaksoittaisia, voidaan kuvata sini- ja kosinifunktioiden painotettuna integraalina. Tällöin puhutaan niin sanotusta *Fourier-muunnoksesta*, joka on vaikuttanut ratkaisevasti muun muassa digitaalisen signaalikäsittelyn alaan. (Gonzales & Woods 2002, 148)

Diskreetti yksiulotteinen Fourier-muunnos voidaan laskea seuraavalla kaavalla:

$$F(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)[\cos(\frac{2\pi ux}{N})-i\sin(\frac{2\pi ux}{N})] \quad .$$

Kaavassa *F(u)* on muunnoksen tuloksena saatava kompleksiluku ja *f(x)* alkuperäisen signaalin arvo kohdassa *x*. Kaavassa *i* tarkoittaa imaginääriosaa. (Gonzales & Woods 2002, 151)

Kun muunnoksen kohteena on kuva, tulee muunnos tehdä kahdessa ulottuvuudessa. Tällöin Fourier-muunnos *N x N* kokoiselle kuvalle saadaan eksponenttimuodossa esitettynä seuraavasti:

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-i2\pi(ux+vy)/N} \quad .$$

Yhtälössä *f(x, y)* vastaa pikselin arvoa sijainnissa *x*, *y*. Tuloksena saatava kompleksiluku *F(u, v)* puolestaan kertoo sen, kuinka paljon *u*:n ja *v*:n määrittelemää taajuutta esiintyy kuvassa. (Efford 2000, 195) Tulos lasketaan siis kaikille *u*:n ja *v*:n arvoille. Jokaiseen tulokseen puolestaan summataan funktion *f(x, y)* saamat arvot. Sekä *u*:n ja *v*:n, että *x*:n ja *y*:n arvoalueet määräytyvät kuvan ulottuvuuksien, eli pikseleiden lukumäärän perusteella. (Gonzales & Woods 2002, 151-155)

Käänteinen Fourier-muunnos puolestaan saadaan seuraavasti:

$$f(x,y)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} F(u,v)e^{i2\pi(ux+vy)/N} \quad .$$

Kuten voidaan havaita, oleellisin eroavaisuus edelliseen yhtälöön on itse asiassa eksponentin etumerkissä. Käänteisen muunnoksen tuloksena saadaan pikseliarvo *f(x, y). F(u,*

v) puolestaan on kompleksiluku, joka vastaa Fourier-kerrointa kohdassa *(u, v)*. (Efford 2000, 195)

Fourier-muunnos on laskennallisesti varsin raskas toimenpide. Yksiulotteisen muunnoksen aikavaativuus on $O(N^2)$ ja kaksiulotteisen $O(N^4)$. Fourier-muunnoksen laskemiseksi on kuitenkin kehitetty ns. nopea Fourier-muunnos (Fast Fourier transform, FFT). Kaksiulotteisen muunnoksen tapauksessa voidaan ensinnäkin hyödyntää sitä seikkaa, että Fourier-muunnos voidaan suorittaa vaaka- ja pystysuunnassa erillisinä yksiulotteisina muunnoksina. Tällöin jälkimmäisen muunnoksen lähtöarvoina käytetään ensimmäisen muunnoksen tulosta. Tämä pudottaa kaksiulotteisen muunnoksen aikavaativuuden luokkaan $O(N^3)$. Yksiulotteista muunnosta puolestaan voidaan nopeuttaa suorittamalla se rekursiivisesti. Tämä on mahdollista sen vuoksi, että Fourier-muunnos, jonka pituus on *N*, voidaan laskea kahden *N/2* mittaisen muunnoksen summana. Tällä tavoin yksiulotteisen muunnoksen aikavaativuudeksi saadaan $O(N \log_2 N)$ ja kaksiulotteisen $O(N^2 \log_2 N)$. Rekursiivinen toteutus kuitenkin edellyttää, että kuvan ulottuvuudet ovat kahden potensseja. Tällöin kuvaa voidaan joutua joko pienentämään, tai jatkamaan ylimääräisillä nolla-arvoilla. (Efford 2000, 197-198)

Fourier-muunnoksen lisäksi usein käytettyjä muunnoksia ovat muun muassa kosinimuunnos ja aallokemuunnos. Näitä käsitellään lisää kuvan pakkauksen yhteydessä.

2.4 Digitaalisten kuvien tiedostomuotoja

Bittikarttakuvien eräs haittapuoli on niiden vaatima tallennustila. Käytetty tila riippuu kuvan pikseleiden sekä kunkin pikselin värin ilmaisemiseen käytettyjen bittien lukumäärästä. Ongelman ratkaisuksi on kehitetty erilaisia pakkaustapoja, joiden avulla kuvien vaatimaa tilaa voidaan pienentää. (Miano 1999, 3-4) Pakkausmenetelmät voidaan jakaa *häviöllisiin* ja *häviöttömiin*. Häviöttömät pakkausmenetelmät säilyttävät kaiken alkuperäisen kuvainformaation, kun taas häviölliset pakkausmenetelmät kadottavat osan kuvainformaatiosta. (Miano 1999, 12-13) Häviöllisten pakkausmenetelmien aiheuttama heikennys kuvan laadulle riippuu kuvan koodaamiseen käytettyjen bittien lukumäärästä. (Bovik 2009, 429)

2.4.1 Häviöttömät tiedostomuodot

Häviöttömään pakkaukseen käytetään muun muassa *sanakirjamenetelmiä*, *juoksunpituuskoodausta* (myös jononpituuskoodaus, run lenght encoding) ja *Huffmanin koodausta.* Sanakirjamenetelmät perustuvat siihen, että kuvasta etsitään toistuvia bittijonoja, jotka korvataan lyhyemmillä koodeilla. Tätä varten bittijonoista ja niihin viittaavista koodeista tulee muodostaa eräänlainen sanakirja. Mitä pidempiä muodostetut sarjat ovat, ja mitä useammin ne esiintyvät pakattavassa datassa, sitä parempi pakkausteho saavutetaan. Juoksunpituuskoodauksessa peräkkäin toistuvat väriarvot korvataan siten, että väriarvo esitetään vain kerran, jonka jälkeen ilmaistaan se, kuinka pitkään kyseistä arvoa toistetaan. Huffman koodaus puolestaan perustuu siihen, että kiinteän bittimäärän sijaan väriarvot koodataan vaihtelevan mittaisilla bittijonoilla. Useammin esiintyvät arvot ilmaistaan lyhyemmillä koodeilla. (Miano 1999, 11)

Muun muassa *BMP* (Microsoft Windows / OS/2 bitmap), *GIF* (Graphics Interchange Format) ja *PNG* (Portable Network Graphics) ovat häviöttömiä kuvaformaatteja (Miano 1999, 10). BMP on Windows-käyttöjärjestelmien pääasiallinen kuvamuoto. BMP-kuvissa voidaan käyttää enimmillään 8 bitin näytetarkkuutta. Riippuen käytetystä näytetarkkuudesta BMP-kuvissa on mahdollista käyttää juoksunpituuskoodausta, mutta käytännössä tämä on varsin harvinaista. (Miano 1999, 23) GIF-kuvissa käytetään LZW-pakkausalgoritmia (Lempel–Ziv–Welch), joka on sanakirjamenetelmä. Gif-kuvat voivat sisältää ainoastaan 256 eri väriä. (Miano 1999, 171) PNG-kuvissa käytetään Deflate-pakkausmenetelmää, jossa yhdistyvät sanakirjapakkaus ja Huffman-koodaus. PNG-kuvissa pikseleiden väriarvot voidaan kuvata jopa 16 bitin näytetarkkuudella. Tällöin siis yhden pikselin kaikkien värikanavien kuvaamiseen käytetään peräti 48 bittiä. (Miano 1999, 189, 215-221)

2.4.2 JPEG-standardi

Häviöllisistä kuvaformaateista käytetyin on epäilemättä *JPEG* (Joint Photographic Experts Group). Erinomaisen pakkaustehonsa vuoksi JPEG on myös selkeästi suosituin tiedostomuoto valokuvien tallentamiseen (Miano 1999, 35). Useimmissa taskukameroissa JPEG onkin, ellei peräti ainoa mahdollinen, niin ainakin vaihtoehtoinen kuvien tallennusmuoto.

JPEG-kuvissa käytetään YcbCr-värimallia ja suurin mahdollinen näytetarkkuus on 12-bittiä. (Miano 1999, 37-44)

Tarkkaan ottaen JPEG-standardi ei kuitenkaan määrittele varsinaista tiedostomuotoa, vaan joukon erilaisia pakkaustekniikoita. Standardissa määritellään erikseen häviötön ja häviöllinen pakkausmalli. (Miano 1999, 35) Häviöttömistä menetelmistä käytössä ovat muun muassa Huffman-koodaus ja juoksupituuskoodaus (Miano 1999, 10). Virallisen tiedostomuodon puuttuessa JFIF (JPEG File Interchange Format) on vakiinnuttanut asemansa JPEG-tiedostojen käytetyimpänä tiedostomuotona (Miano 1999, 40).

JPEG:in häviöllisen osan muodostavat *diskreetti kosinimuunnos* (discrete cosine transformation, DCT) ja *kvantisointi*. Kosinimuunnoksen avulla kuvasta luodaan taajuusesitys. Kuva jaetaan tässä yhteydessä 8 x 8 pikselin alueisiin, ja muunnos kohdistuu erikseen jokaiseen lohkoon. Diskreetti kosinimuunnos voidaan laskea seuraavasti:

$$F(u,v)=\frac{C[u]C[v]}{4}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos[\frac{(2x+1)u\pi}{16}]\cos[\frac{(2y+1)v\pi}{16}] \quad 0\le u,v\le 7 \quad ,$$

jossa $C[u]=\begin{cases}\frac{1}{\sqrt{2}} & u=0,\\ 1 & u\neq 0\end{cases}$.

Kaavassa *f(x, y)* tarkoittaa pikseliarvoa ja *F(u, v)* tuloksena saatavaa reaaliarvoista painokerrointa. Kuten Fourier-muunnos, myös kaksiulotteinen kosinimuunnos voidaan laskea yksiulotteisten muunnosten tulona. Kosinimuunnoksen laskemisessa ei sen sijaan tarvita kompleksilukuja. (Bovik 2009, 425) Ennen kosinimuunnosta pikseliarvot muunnetaan arvoalueelta [0, 255] alueelle [-128, 127] (Popescu 2005, 71).

Kunkin lohkon muunnoksen tuloksena saadaan matriisi, joka sisältää taajuuskomponenttien painokertoimia (dct-coefficients). Muunnos on suoritettava jokaiselle värikanavalle erikseen. Tyypillisesti matalia taajuuksia vastaavat kertoimet ovat suurempia, ja niiden merkitys kuvainformaatiolle on myös suurin. (Bovik 2009, 425-426)

Kvantisointi puolestaan tarkoittaa yksinkertaisesti kertoimien jakamista toisella luvulla ja pyöristämistä lähimpään kokonaislukuun. Tällöin osa kertoimista saa tavallisesti arvon nolla. Mitä suurempia jakajia käytetään, sitä enemmän nolla-arvoja muodostuu. Ja-

kajien arvot sisältyvät ns. *kvantisointimatriiseihin*, joiden koko on myös 8 x 8. Jokaista taajuutta vastaa siis yksilöllinen kvantisointiarvo. Lisäksi eri värikanaville voi olla erilliset kvantisointimatriisit. Tyypillisesti Cr- ja Cb-komponentit käyttävät samaa matriisia. Tämä johtuu siitä, että kyseiset värikanavat kvantisoidaan yleensä karkeammin kuin Y-kanava. Kvantisointi voidaan esittää seuraavan kaavan avulla:

$$qX(m,n) = \left[\frac{X(m,n)}{q(m,n)} \right] \quad ,$$

jossa *[a]* tarkoittaa pyöristämistä lähimpään kokonaislukuun. *X(m, n)* on reaaliarvoinen painokerroin ja *q(m, n)* on kvantisointiarvo, joka on kokonaisluku väliltä 1-255. Tuloksena saadaan kvantisoitu painokerroin *qX(m, n)*, joka on nyt siis kokonaisluku. (Bovik 2009, 428)

JPEG-standardi ei määrittele kvantisointiin käytettäviä arvoja, mutta se sisältää muutamia kokeellisesti valittuja matriiseja, jotka tarjoavat hyvän tasapainon kuvanlaadun ja pakkaustehon välillä. Juuri kvantisoinnin seurauksena osa kuvainformaatiosta menetetään. (Miano 1999, 77-90) Käytettävät kvantisointimatriisit määrittelevät siis pääasiassa sen, kuinka paljon paljon kuvan laatu kärsii JPEG-pakkauksen johdosta. Useissa käytännön sovelluksissa pakkauslaatu määritellään kokonaislukuarvolla väliltä 1-100. Tällöin kvantisointimatriisin arvot skaalataan tietyn kaavan mukaan. Kun laatu on 100, eli paras mahdollinen, käytetään kvantisointimatriisin arvoja sellaisenaan. (Bovik 2009, 429-430)

On huomattava, että yhdellä kokonaislukuarvolla ilmaistu pakkauslaatu ei välttämättä ole täysin yksikäsitteinen eri sovellusten välillä. Kuten todettu, täsmällinen pakkauslaatu määritellään viime kädessä kvantisointimatriisien perusteella. Kuvassa 2 näkyy osa valokuvasta, joka on tallennettu Gimp-kuvankäsittelyohjelmalla, käyttäen kolmea eri pakkauslaatua. Kuvia on suurennettu ilmiön selkeyttämiseksi. Käytetyt tiedostomuodot ovat PNG, JPEG laadulla 80 ja JPEG laadulla 60. Kahdessa oikeanpuoleisimmassa kuvassa on huomattavissa tyypillisiä JPEG-pakkauksen aiheuttamia ilmiöitä, kuten laatikoitumista ja yksityiskohtien katoamista. Laajempi esimerkki JPEG-pakkauksen vaikutuksista löytyy muun muassa osoitteesta *http://en.wikipedia.org/wiki/Jpeg*.

Kuva 2: Kuva tallennettu kolmea eri pakkauslaatua käyttäen. Käytetyt tiedostomuodot ovat vasemmalta oikealle häviötön PNG, JPEG laadulla 80 ja JPEG laadulla 60.

JPEG-tiedostoa purettaessa edellä kuvatut vaiheet suoritetaan käänteisesti. Huffman-koodaus puretaan, suoritetaan dekvantisointi, eli kvantisoidut kertoimet kerrotaan alunperin kvantisointiin käytetyillä luvuilla ja lopulta suoritetaan käänteinen kosinimuunnos (IDCT, inverse discrete cosine transform) Käänteisen kosinimuunnoksen jälkeen muodostuneet pikseliarvot muunnetaan takaisin arvoalueelle [0, 255]. (Popescu 2005, 72)

Käänteinen diskreetti kosinimuunnos lasketaan seuraavasti:

$$f(x,y)=\sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C[u]C[v]}{4}F(u,v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}\quad 0\le x,y\le 7\quad ,$$

jossa *F(u, v)* on painokertoimen arvo, joka on nyt kokonaisluku, ja *f(x, y)* vastaa puolestaan tuloksena saatavaa pikseliarvoa.

Taulukoissa 1-5 on esitettynä vaihe kerrallaan pikseliarvojen diskreetti kosinimuunnos, kvantisointi ja käänteinen diskreetti kosinimuunnos takaisin pikseliarvoiksi. Taulukossa 1 näkyy esimerkki 8 x 8 -kokoisesta pikselilohkosta. Taulukossa 2 kyseisille arvoille on suoritettu diskreetti kosinimuunnos. Taulukko 3 on esimerkki kvantisointimatriisista. Taulukossa 4 taulukon 2 arvot on kvantisoitu käyttäen taulukon 3 arvoja. Lopulta taulukossa 5 taulukon 4 arvot on muunnettu jälleen pikseliarvoiksi käänteisen diskreetin kosinimuunnoksen avulla. Taulukoiden arvoista voidaan havaita kvantisointiin käytettyjen arvojen merkitys. Pienillä arvoilla kvantisoidut kertoimet muuttuvat suhteellisen vähän. Sen sijaan suuria kvantisointiarvoja käytettäessä kertoimet voivat muuttuu hyvinkin ratkaisevasti. Kuten todettu, osa kertoimista saa myös tyypillisesti arvon nolla.

Kuvassa 3 on taulukoiden 1 ja 5 arvot esitettynä kuvan muodossa. Samat arvot on asetettu kaikille värikanaville, joten kuvat ovat harmaasävyisiä. Vasen kuva vastaa alkuperäisiä arvoja ja oikealla näkyvä kuva kosinimuunnoksen ja kvantisoinnin läpi käyneitä arvoja. Kuten kuvasta voidaan havaita, pikseleiden arvot ovat muuttuneet jonkin verran.

Taulukko 1: Esimerkki 8 x 8 -kokoisesta pikselilohkosta.

85	71	71	71	71	71	71	85
71	125	85	85	85	85	125	71
71	85	152	125	125	152	85	71
71	85	125	152	152	125	85	71
71	85	125	152	152	125	85	71
71	85	152	125	125	152	85	71
71	125	85	85	85	85	125	71
85	71	71	71	71	71	71	85

Taulukko 2: Taulukon 1 pikseliarvoille suoritetun diskreetin kosinimuunnoksen tulos.

765	0	-106,3	0	-34	0	7,2	0
0	0	0	0	0	0	0	0
-106,3	0	125	0	-2,9	0	-32,5	0
0	0	0	0	0	0	0	0
-34	0	-2,9	0	61	0	8,8	0
0	0	0	0	0	0	0	0
7,2	0	-32,5	0	8,8	0	77	0
0	0	0	0	0	0	0	0

Taulukko 3: Esimerkki kvantisointimatriisista.

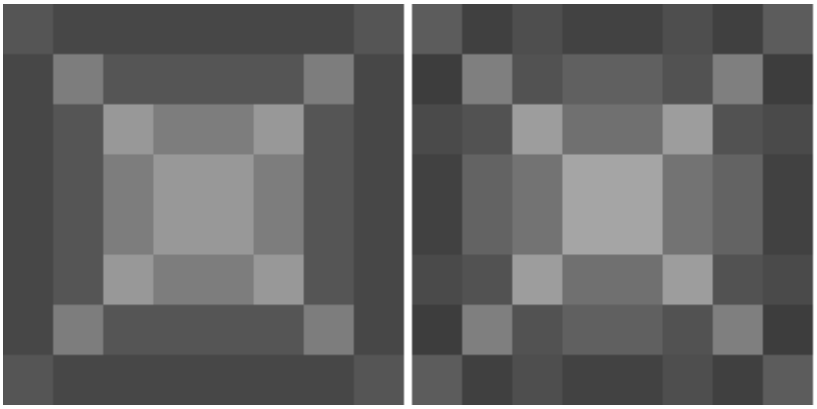
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Taulukko 4: Taulukon 2 painokertoimet kvantisoinnin ja dekvantisoinnin jälkeen. Kvantisointiin käytettiin taulukossa 3 näkyviä arvoja.

768	0	-110	0	-24	0	0	0
0	0	0	0	0	0	0	0
-112	0	128	0	0	0	0	0
0	0	0	0	0	0	0	0
-36	0	0	0	68	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	120	0
0	0	0	0	0	0	0	0

Taulukko 5: Taulukon 4 arvot muunnettuna takaisin pikseliarvoiksi käänteisen diskreetin kosinimuunnoksen avulla.

92	61	74	65	65	74	61	92
64	127	82	99	99	82	127	64
78	82	157	115	115	157	82	78
66	96	112	165	165	112	96	66
66	96	112	165	165	112	96	66
78	82	157	115	115	157	82	78
64	127	82	99	99	82	127	64
92	61	74	65	65	74	61	92



Kuva 3: Taulukoiden 1 ja 5 pikseliarvot kuvan muodossa. Vasemmalla ovat alkuperäiset pikseliarvot. Oikealla näkyvät arvot ovat läpikäyneet diskreetin kosinimuunnoksen ja kvantisoinnin. Kvantisoinnissa käytettiin taulukon 3 arvoja. Kuvat on suurennettu 8 x 8 pikselin kokoisista lohkoista.

JPEG-standardi mahdollistaa myös eri resoluution käyttämisen eri värikanaville. Tämä tarkoittaa sitä, että kaikilta värikanavilta ei välttämättä tallenneta samaa määrää pikseleitä. Käytännössä tämä tarkoittaa sitä, että alhaisemmalla taajuudella näytteistettyjen kanavien osalta samaa pikseliarvoa monistetaan useamman pikselin alueelle. Yleensä Y-kanava tallennetaan käyttäen korkeinta mahdollista resoluutiota. (Miano 1999, 41-44)

JPEG2000 on alkuperäisen JPEG-standardin kehittäjien suunnittelema pakkausstandardi, jonka kehitystyössä lähtökohtana on ollut muun muassa parannettu pakkaustehokkuus. Kosinimuunnoksen sijaan JPEG2000 perustuu *aallokemuunnokseen* (wavelet transform). (Bovik 2009, 445) Aallokemuunnoksessa kuva jaetaan suodattimien avulla rekursiivisesti neljään osaan. Muodostetut osat vastaavat kuvan eri taajuusalueita. (Bovik 2009, 131)

2.5 Digitaalisen valokuvan muodostuminen

Digitaalisen valokuvan tallentamisprosessi on melko monimutkainen ja vaihtelee huomattavasti eri kameramallien välillä. Kuitenkin tietyt komponentit ja prosessointivaiheet ovat yhteisiä lähes kaikille digitaalisille kameroille. Useimmissa kameroissa on yksittäinen *CCD* (charge-coupled device), tai *CMOS* (complementary metal oxide semiconductor) -kenno. Kenno koostuu pikseleistä, jotka tallentavat valoa muuntamalla fotoneita elektroneiksi. Näin kertynyt sähköinen varaus vahvistetaan ja muunnetaan sitten digitaaliseksi signaaliksi. (Fridrich 2009)

Useimmiten kuvakennot eivät kuitenkaan rekisteröi väriä, joten tavallisesti kukin kennon pikseli on varustettu värisuotimella, joka päästää läpi ainoastaan yhtä väriä, yleensä punaista, vihreää tai sinistä. Lievästi yksinkertaistaen voidaan sanoa, että ainoastaan yksi näistä väreistä tallennetaan yhden pikselin alueelta. Täydellisen värikuvan muodostamiseksi kaksi muuta väriä on arvioitava viereisten näytteiden perusteella. Tästä toimenpiteestä käytetään nimeä *interpolaatio*. On kuitenkin olemassa myös sellaisia kuvakennoja, joissa värisuodatinta ei tarvita, joskin tämä on harvinaista. (Popescu 2005, 45-46, 67)

Lopulta kuvaan voidaan kohdistaa erinäistä prosessointia, kuten värien korjailua ja mahdollisesti kohinan poistoa sekä tarkennusta. Etenkin taskukameroissa kuva tallennetaan usein käyttäen jotakin häviöllistä tiedostomuotoa, tyypillisesti JPEG:iä. (Fridrich 2009)

2.6 Digitaalinen kuvankäsittely

Kuvankäsittely (image processing) on melko laaja käsite, ja sen alle voidaan lukea hyvin monenlaisia aihealueita, kuten kuvan ehostaminen, restaurointi ja pakkaus (Gonzales & Woods 2002, 26). Kielitoimiston sanakirja määrittelee kuvankäsittelyn seuraavasti: *"(sähkömagneettisiksi merkeiksi muunnetun) kuvan elektroninen käsittely vars. tietokoneen avulla"*. Gonzalesin ja Woodsin (2002, 2-3) mukaan usein käytetty määritelmä kuvankäsittelylle on *prosessi, jonka syötteenä ja tuloksena on kuva*. He kuitenkin toteavat, että määritelmä saattaa olla liian suppea, sillä kuvankäsittelyn tuloksen ei välttämättä tarvitse olla kuva, vaan se voi olla myös esimerkiksi kuvasta johdettua numeerista informaatiota. Toisin sanoen kuvankäsittelyn tarkoituksena ei välttämättä aina ole kuvan muuttaminen.

2.6.1 Kuvankäsittelyoperaatioiden jaottelua

Bovikin (2009, 43) mukaan kuvankäsittelyoperaatiot voidaan jakaa *pisteoperaatioihin ja geometrisiin operaatioihin*. Pisteoperaatiot kohdistuvat ainoastaan yksittäisten pikseleiden arvoihin. Tämä tarkoittaa sitä, että operaatiot eivät vaikuta esimerkiksi pikselin sijaintiin tai sen naapuripikseleiden arvoihin. Tästä johtuen pisteoperaatioiden vaikutus kuvaan on ainakin jossain määrin rajallinen. Pisteoperaatiot eivät esimerkiksi vaikuta kuvan elementtien sijaintiin tai muotoon. Tyypillisiä pisteoperaatioita ovat muun muassa kuvan kirkkauden, värien tai kontrastin säätäminen.

Geometriset operaatiot sen sijaan vaikuttavat ainoastaan pikseleiden sijaintiin. Tyypillisiä geometrisiä operaatioita ovat muun muassa kuvan kiertäminen ja koon muuttaminen.

2.6.2 Kuvamanipulaatio

Kuvankäsittely on käsitteenä laajempi ja neutraalimpi, kuin *kuvan manipulointi*, joka viittaa selkeämmin jonkinlaiseen vilpilliseen toimintaan. Kielitoimiston sanakirja antaa muun muassa seuraavat määritelmät verbille *manipuloida*: *"käsitellä vilpillisesti t. omavaltaisesti, peukaloida"*. Tällä voidaan ymmärtää tarkoitettavan jokseenkin samaa kuin väärentämisellä.

Digitaalinen kuvankäsittely tarjoaa periaatteessa rajattomat mahdollisuudet kuvien muokkaamiseen. Käytännössä kuitenkin valokuvien manipuloimiseen käytetään tavalli-

sesti melko rajallista joukkoa toimenpiteitä. Esimerkiksi Mahdianin ja Saicin (2008) mukaan tyypillisimpiä valokuvien manipulointiin käytettyjä operaatioita ovat kuvan elementtien poistaminen ja lisääminen. Elementeillä tarkoitetaan esimerkiksi kuvassa esiintyviä ihmisiä tai esineitä. Toisaalta Zhang ym. (2008) kuitenkin toteavat, että varsin usein väärentäjät joutuvat turvautumaan useisiin erilaisiin muokkaustapoihin uskottavan lopputuloksen aikaansaamiseksi.

Ei ole kuitenkaan aivan yksinkertaista määritellä sitä, missä kulkee harmittoman kuvankäsittelyn ja varsinaisen kuvamanipulaation raja. Tämä puolestaan liittyy oleellisesti kysymykseen siitä, mitä aidolla tai väärennetyllä valokuvalla tarkoitetaan. Yksi tulkinta voisi olla se, että aito valokuva on suora kameran tallentama tuotos. Tämä saattaa kuitenkin olla liian tiukka tulkinta, sillä kuten todettu, kaikkien kuvankäsittelytoimenpiteiden tarkoituksena ei ole kuvan sisällön vääristely. Kaikki kuvankäsittelytoimenpiteet eivät myöskään ole nimenomaisesti kuvankäsittelyohjelmalla tapahtuvan muokkaamisen seurausta. Esimerkiksi jo digitaalisten kameroiden signaaliketjuun voi sisältyä erilaisia prosessointivaiheita, kuten värien korjailua, kohinan poistoa ja kuvan terävöitystä (Fridrich 2009). Tämän kaltaisilla toimenpiteillä ei varmasti useimmiten ole oleellista vaikutusta kuvan keskeiseen sisältöön tai tulkintaan. Kuvan keskeisen sisällön määrittelykään ei kuitenkaan ole kontekstista riippumatonta. Täysin yksiselitteisen ja kontekstista riippumattoman määritelmän esittäminen valokuvan aitoudesta on melko hankalaa. Kuten todettu, yksi vaihtoehto voisi olla suora kameran tallentama tuotos.

Yksikäsitteisen määritelmän puuttumisesta huolimatta voidaan kuitenkin todeta, että valokuvaa voidaan pitää väärennettynä ainakin siinä tapauksessa, että kuvaan on lisätty tai siitä on poistettu elementtejä, tai elementtien sijaintia, kokoa, muotoa taikka asentoa on muutettu. Varsinaisesta väärentämisestä ei sen sijaan yleensä voida puhua sellaisten operaatioiden kohdalla, jotka eivät juurikaan edellytä kuvankäsittelijän subjektiivista harkintaa. Tällaisia ovat ainakin kuvan pakkaaminen ja kuvan resoluution muuttaminen. On tietenkin selvää, että tällaisetkin operaatiot saattavat kadottaa sellaisia kuvan yksityiskohtia, jotka voivat olla jossakin yhteydessä merkityksellisiä.

Valokuvien autentikointitekniikoiden näkökulmasta ongelmallisinta on se, että erilaisia muokkaustapoja ei useinkaan ole mahdollista erottaa toisistaan. Täysin harmittomat käsittelytoimenpiteet saattavat aiheuttaa sen, että kuva tulkitaan väärennetyksi, tai arvion

tekeminen ei ole lainkaan mahdollista. Tällaisia toimenpiteitä voidaan mahdollisesti käyttää myös peittämään varsinaisia muokkausjälkiä.

3 KATSAUS VALOKUVIEN AUTENTIKOINTITEKNIIKOIHIN

Tässä luvussa esittelen viisi passiivista tekniikkaa digitaalisten valokuvien aitouden arvioimiseen. Olen pyrkinyt valitsemaan käsittelyyn kaikkein varteenotettavimpia tekniikoita, joita alan kirjallisuudessa on esitetty. Ennen passiivisten tekniikoiden esittelyä luon lyhyen katsauksen vesileimatekniikoihin.

3.1 Tekniikoiden arviointikriteerit

Esitän kunkin käsitellyn tekniikan pääasiallisen toimintaperiaatteen ja arvioin tyypillisiä ongelmia, joita tekniikoiden toteutukseen liittyy. Tarkoitukseni ei ole kuitenkaan käsitellä toteutusten tarkkoja teknisiä yksityiskohtia. Tekniikoiden arviointiin käytän kriteereitä, jotka voidaan jakaa neljään pääkohtaan. Seuraavaksi käyn läpi kyseiset kriteerit.

Tunnistuksen tarkkuus. Menetelmien tunnistustarkkuutta voidaan tarkastella kahdesta eri näkökulmasta. Ensinnäkin voidaan arvioida yleisesti menetelmän kykyä tehdä oikeita tunnistuksia suhteessa virheellisesti tunnistettuihin aitoihin kuviin (false positive) ja tunnistamatta jääneisiin muokattuihin kuviin (false negative). Toiseksi voidaan arvioida alueellista tarkkuutta, eli sitä, kuinka tarkasti menetelmät kykenevät paikantamaan ne alueet, joihin muokkaus kohdistuu. Tällaisten arvioiden tekeminen ei kuitenkaan ole aivan yksinkertaista, sillä asiaan vaikuttavat hyvin monet tekijät, kuten käytetyn muokkauksen tyyppi ja aste, sekä tietenkin käytetyn autentikointitekniikan erityispiirteet. Tekniikoita ei ole myöskään testattu täysin yhteneväisin periaattein, vaan kehittäjien suorittamat testit ja tulosten esitystavat poikkeavat jonkin verran toisistaan. Useimmissa tapauksissa menetelmiä on kuitenkin testattu häviöllisesti pakattujen kuvien suhteen useilla eri pakkauslaaduilla. Kaikissa tapauksissa käytetty tiedostomuoto on tällöin JPEG, joissain tapauksissa lisäksi JPEG2000. Pyrin mahdollisuuksien mukaan esittämään jonkinlaisia numeerisia arvioita eri tekniikoiden tarkkuudesta, mikäli tällaisia tietoja on saatavilla. Yhtenäisten testien puuttuessa tuloksia on kuitenkin syytä tulkita varovaisesti. Testien tarkemmat yksityiskohdat ja tulokset löytyvät alkuperäislähteistä.

Menetelmien tarkkuuteen vaikuttavat myös tietyt tekniset ratkaisut. Usein muokkausjälkien esiintymistä tutkitaan jakamalla käsiteltävä kuva kiinteän kokoisiin lohkoihin. Tavallisesti lohkon koon merkitys on se, että pienestä lohkosta muokkausjälkien tunnistaminen voi olla hyvin epävarmaa ja virhealtista. Suuremmasta lohkosta tunnistus voidaan

tehdä luotettavammin, mutta alueellinen tunnistustarkkuus puolestaan kärsii. Lohkon koko ei kuitenkaan suoraan kerro tunnistuksen alueellista tarkkuutta, sillä lohkot voivat yleensä olla myös osittain päällekkäisiä. Muokatun alueen rajausta voidaan näin tarkentaa useammista lohkoista saatavien tulosten perusteella. Tämä tietenkin lisää laskennallisia kustannuksia. (esim. Chen, Fridrich, Lukáš & Goljan 2007)

Tunnistetut muokkaustyypit. Tässä luvussa esitellyt pikselitason tekniikat rajoittuvat muutamien nimenomaisten muokkaustyyppien tunnistamiseen. Loput tekniikat puolestaan ovat siinä mielessä yleisiä, että ne kykenevät ainakin periaatteessa tunnistamaan lähes kaikki mahdolliset kuvaan kohdistuvat muutokset. Kuitenkaan aivan kaikkia muutoksia ei varmastikaan ole mahdollista havaita näilläkään tekniikoilla. Esimerkiksi rajaus ei aiheuta kuvaan sellaisia tilastollisia muutoksia, jollaisten havaitsemiseen käsitellyt menetelmät yleensä perustuvat. Tekniikoiden kehittäjät eivät kuitenkaan ole kovinkaan kattavasti käsitelleet sitä, mitkä muokkaustyypit ovat todella tunnistettavissa, ja mitkä eivät. Myöskään tässä yhteydessä en pyri esittämään kattavaa listausta. Voidaan kuitenkin todeta, että kaikkein oleellisimmin kuvan sisältöä muuttavat toimenpiteet, kuten kuva-alueiden korvaaminen, ovat yleensä tunnistettavissa.

Rajoitukset. Pyrin antamaan arvion siitä, kuinka hyvin käsitellyt menetelmät soveltuvat erilaisiin käytännön tilanteisiin. Tähän vaikuttavat menetelmien erilaiset rajoitukset, kuten sidonnaisuus tiettyyn kuvaformaattiin, suuret laskennalliset vaatimukset, ynnä muut vastaavat seikat.

Vakaus vastatoimia vastaan. Esiteltyjen tekniikoiden toimintaa voidaan pyrkiä häiritsemään erilaisten vastatoimien avulla. Pyrin arvioimaan sitä, kuinka helposti tällaisia vastatoimia on kehitettävissä. Joitakin tekniikoita vastaan sellaisia on jo kehitetty.

3.2 Vesileimatekniikat

Aktiivisten autentikointitekniikoiden osalta keskityn tässä tutkielmassa vesileimatekniikoihin. Niiden lisäksi aktiivisiin tekniikoihin luetaan ainakin digitaaliset allekirjoitukset (Zhang ym. 2008). Jätän digitaaliset allekirjoitukset käsittelyn ulkopuolelle, sillä viimeaikainen tutkimus näyttää keskittyneen voittopuolisesti vesileimoihin. Digitaaliset allekirjoitukset eroavat vesileimoista siinä, että autentikointiin käytettyä dataa ei upoteta kuvainformaatioon, vaan se pidetään erillisessä tiedostossa tai kuvan otsikkotiedoissa (Rey

& Dugelay 2002). Tosin kyseisiä käsitteitä käytetään toisinaan myös päällekkäin (Mohanty 1999).

Vesileimalla tarkoitetaan tiettyä dataa, esimerkiksi otsikkoa tai tunnistetta, joka upotetaan kohdeobjektiin siten, että myöhemmin vesileima voidaan tunnistaa ja sen perusteella kohdeobjektista voidaan tehdä erinäisiä päätelmiä. Kohteena voi kuvien lisäksi olla muunkinlaista sisältöä, kuten videota tai audiota. (Mohanty 1999, 4) Eri mediat voivat kuitenkin asettaa käytetylle tekniikalle jossain määrin erilaisia vaatimuksia (Podilchuk & Delp 2001). Yleensä vesileimajärjestelmissä käytetään salausavaimia vesileiman muuttamisen tai poistamisen estämiseksi (Hartung & Kutter 1999). Sen sijaan vesileima, toisin kuin salaus yleensä, ei estä kohteena olevan sisällön tarkastelua (Podilchuk & Delp 2001).

Mohantyn (1999, 5) mukaan vesileimat voidaan jakaa lukuisiin eri kategorioihin niiden eri ominaisuuksien perusteella. Kaikkein oleellisin jakoperuste liittyy vesileimojen käyttötarkoitukseen. *Kestäviä* (robust) vesileimoja on perinteisesti käytetty tekijänoikeuksien suojaamiseen ja sisällön seuraamiseen. Sisällön autentikointiin puolestaan on tavattu käyttää *heikkoja* (fragile) vesileimoja. Kestävien vesileimojen perustavana ajatuksena on se, että niiden tulisi säilyä mahdollisimman hyvin isäntäsignaalissa siihen kohdistuvista muutoksista huolimatta. Heikkojen vesileimojen toiminta perustuu siihen, että kuvaan kohdistetut muutokset muuttavat myös vesileimaa (Kundur & Hatzinakos 1999).

Kestävien ja heikkojen vesileimojen lisäksi on kehitetty näiden kahden tyypin välimuoto, josta käytetään englanninkielisessä kirjallisuudessa termiä *semi-fragile*. Käytännön sovellusten näkökulmasta heikkojen vesileimojen suurin ongelma on niiden kykenemättömyys erottaa häviöllistä pakkausta ynnä muita ”sallittuja” operaatioita varsinaisesta kuvan muokkaamisesta. (Rey & Dugelay 2002) Muun muassa Lin ym. (2000) ovat esittäneet toteutuksen, joka yhdistelee ominaisuuksia kestävistä ja heikoista vesileimoista. Se kykenee paikantamaan kuvan muokatut alueet, mutta lisäksi se sallii kestävien vesileimojen tapaan jonkin verran mm. häviöllistä pakkaamista.

Vesileimajärjestelmän toiminnassa on erotettavissa kolme erillistä vaihetta. Ensimmäinen vaihe käsittää vesileimasignaalin luomisen. Vesileima riippuu yleensä avaimesta ja vesileimainformaatiosta. Vesileimainformaatio voi olla täysin satunnaista, mutta se voi myös riippua esimerkiksi kohteena olevan kuvan sisällöstä. Toisessa vaiheessa vesilei-

ma upotetaan isäntäsignaaliin. Toisinaan kahta ensimmäistä vaihetta ei lasketa erillisiksi, etenkin jos vesileiman sisältö riippuu kuvainformaatiosta. Toisen vaiheen tuloksena saadaan vesileimalla varustettu kuva. (Hartung & Kutter 1999) Vesileiman upottamiseen käytetty tapa voi vaihdella huomattavasti eri toteutuksissa. Holliman ja Memon (2000, 432) ovat kuitenkin esittäneet yleisen mallin, jolla vesileiman upottamista voidaan havainnollistaa. Malli voidaan esittää seuraavalla yhtälöllä:

$$X' = E_K(X, W).$$

Yhtälössä X on alkuperäinen kuva, X' on vesileimattu kuva, W on vesileimainformaatio, K on upottamiseen käytetty avain ja E on upottamiseen käytetty funktio.

Kolmannessa vaiheessa vesileimasignaali tunnistetaan ja puretaan vesileimatusta kuvasta. Tässä yhteydessä kuva voidaan ajatella ei-toivottuna signaalina tai kohinana, ja vesileima varsinaisena signaalina. (Cox & Miller 1997) Hollimanin ja Memonin (2000) mallin mukaan vesileiman purkaminen voidaan esittää seuraavasti:

$$\hat{W} = D_{K'}(\hat{X}').$$

Yhtälössä $\hat{X}'$ on mahdollisesti korruptoitunut, vesileimattu kuva, K' on vesileiman purkamiseen käytetty avain, D on vesileiman purkamiseen käytetty funktio ja $\hat{W}$ on purettu vesileimainformaatio.

Arvio kuvan aitoudesta saadaan vertaamalla alkuperäistä sekä purettua vesileimaa keskenään. Vertailun tarkemmat yksityiskohdat sekä sen perusteella saatava informaatio riippuvat käytettävästä vesileimajärjestelmästä. Useissa järjestelmissä esimerkiksi muokattujen alueiden sijainti on pääteltävissä. (Rey & Dugelay 2002) Joissakin järjestelmissä on myös mahdollista arvioida muutosten astetta (mm. Kundur & Hatzinakos 1999). Muutamissa toteutuksissa on myös tietyin edellytyksin mahdollista palauttaa karkea versio kuvan alkuperäisestä sisällöstä, joskin tämä on melko harvinaista (Rey & Dugelay 2002).

Vesileimatekniikoiden avulla kuvien autentikointi voidaan tehdä sangen luotettavasti, mutta niiden käyttöön liittyy kuitenkin tiettyjä rajoituksia. Arvioitaessa vesileimatekniikoiden soveltuvuutta, on huomioitava ajateltu käyttötarkoitus. Eräs huomioitava tekijä voi olla se, että koska vesileimainformaatio todella upotetaan itse kuvainformaatioon

eikä esimerkiksi otsikkotietoihin, jättävät vesileimat aina jonkinlaisen jäljen kuvaan. Vaikka jäljet ovatkin normaalitarkastelussa käytännöllisesti katsoen huomaamattomia, voivat ne estää vesileimojen käytön joissakin erityisissä tarkoituksissa, joissa kuvien tarkkuudella on poikkeuksellisen suuri merkitys. Näin voi olla esimerkiksi sen vuoksi, että kuvia joudutaan suurentamaan huomattavia määriä. Tällainen sovellusalue voi olla esimerkiksi lääketiede. (Fridrich 2002) Erääksi ratkaisuksi ongelmaan on esitetty niin sanottu pyyhkiytyvä vesileima. Ajatuksena on se, että vesileima olisi mahdollista poistaa kuvasta purkamisen yhteydessä ja tuloksena saataisiin tällöin alkuperäinen kuvainformaatio. On kuitenkin huomattava, että tällainen ratkaisu saattaisi tarjota varsin houkuttelevan kanavan erilaisille väärinkäytöksille. (Rey & Dugelay 2002)

Käytännön näkökulmasta kaikkein ilmeisin ongelma vesileimatekniikoiden soveltuvuudessa valokuvien autentikointiin on se, että vesileiman lisäämisen mahdollisuus markkinoilla olevissa kameroissa on hyvin poikkeuksellinen. Jotta valokuvien aitous voitaisiin tehokkaasti taata vesileimatekniikoiden avulla, lienee välttämättömänä edellytyksenä se, että vesileima lisätään kuvaan jo ennen kuin se tallennetaan kameran muistikortille tai vastaavalle tallennusmedialle (Farid 2009a). On toki mahdollista käyttää erillistä ohjelmaa vesileiman asettamiseen siinä vaiheessa kun kuva on jo tallennettu. Tällöin on kuitenkin aina olemassa se mahdollisuus, että kuvaa käsitellään ennen vesileiman asettamista.

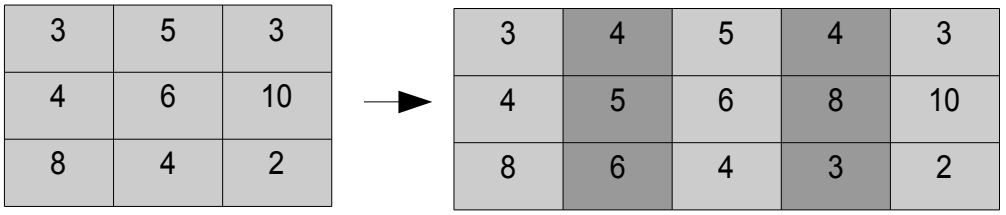
Vaikka vesileimatekniikat eivät yleistyisikään tavallisiin kuluttajakameroihin, niillä saattaisi silti olla joitakin erityisiä sovelluskohteita. Vesileimatekniikalla varustettuja kameroita voitaisiin mahdollisesti käyttää ainakin tilanteissa, joissa kuvien aitous on erityisen tärkeää. Eräänä sovelluskohteena voisi olla vaikkapa rikostutkinta.

3.3 Pikselitason tekniikat

Pikselitason autentikointitekniikat tarkastelevat nimensä mukaisesti erilaisia pikselitasolla esiintyviä ilmiöitä. Niiden toiminta ei perustu esimerkiksi olettamuksiin kameran teknisistä ominaisuuksista, vaan enemmänkin erilaisten muokkaustoimenpiteiden aiheuttamien jälkien tunnistamiseen.

3.3.1 Uudelleen näytteistyksen tunnistaminen

Varsin usein kuvaa muokattaessa joudutaan jotakin kuvan aluetta kiertämään tai venyttämään, jolloin alueen koko muuttuu. Esimerkiksi kuvaan lisättävän henkilön mittasuhteet täytyy mukauttaa kuvan muuhun sisältöön, kuten muihin henkilöihin. Tällaiset operaatiot edellyttävät kuva-alueen *uudelleen näytteistämistä* (resampling), mikä puolestaan yleensä aiheuttaa kuvaan tiettyä jaksoittaista riippuvuutta vierekkäisten pikseleiden välille. Tällaista riippuvuutta ei yleensä esiinny luonnollisesti, joten sitä voidaan käyttää hyväksi tunnistettaessa jälkiä mm. edellä mainituista muokkaustoimenpiteistä. Esimerkkinä voidaan ajatella yksiulotteista signaalia x, joka skaalataan ylöspäin kertoimella kaksi, jolloin saadaan uusi signaali y. Tällöin y:n parittomat näytteet saavat alkuperäiset x:n arvot ja parillisten näytteiden arvoksi muodostuu niiden viereisten näytteiden keskiarvo. Niinpä uudelleen näytteistetty signaali voidaan tunnistaa toteamalla, että sen joka toinen näyte korreloi täydellisesti naapuriarvojensa kanssa. Sama ajatus laajenee hyvin yksinkertaisesti kaksiulotteiseen kuva-avaruuteen. (Farid 2009a) Uudelleen näytteistyksen periaatetta on havainnollistettu kuviossa 1. Kuviossa on kuvitteellinen pikselikehikko, jota on suurennettu sivusuunnassa. Tummennetuilla sarakkeilla olevat arvot on saatu laskemalla viereisten näytteiden keskiarvot.



Kuvio 1: Kuvitteellinen pikselikehikko uudelleen näytteistettynä.

Popescu (2005, 31-37) on testannut menetelmää vaihtelevissa olosuhteissa. Tuloksiin vaikuttaa hyvin monenlaiset tekijät, mutta esitän tässä yhteydessä vain kaikkein tärkeimmät havainnot. Menetelmän tarkkuuteen vaikuttaa muun muassa se, onko muokattua aluetta suurennettu, pienennetty vai kallistettu. Tunnistustarkkuus on lähellä täydellistä silloin, kun kyseessä on yli prosentin suurennus tai yli yhden asteen kallistus. Pienentämisen osalta tunnistustarkkuus näyttäisi olevan parhaimmillaan, kun pienennyksen aste on 5-20 prosenttia. Tunnistustarkkuus heikkenee huomattavasti, kun pienennyksen aste lähenee 50 prosenttia. Pienennyksen asteella tarkoitetaan sitä, kuinka paljon kuvan reso-

luutiota pienennetään sekä pysty, että vaakasuunnassa. 50 prosentin pienennys merkitsee siis itse asiassa sitä, että pienennyksen jälkeen pikseleiden määrä on 25 prosenttia alkuperäisestä. Testien perusteella menetelmä on suhteellisen vakaa kuvaan lisättyä kohinaa vastaan. Sen sijaan etenkin JPEG-pakkaus näyttäisi olevan menetelmän heikkous. Tunnistustarkkuus on vielä hyvä silloin, kun pakkauksen laatu on 97:n ja 100:n välillä. Kuitenkin laadun ollessa 90, tunnistustarkkuus putoaa jo lähes sattuman tasolle. Tämä johtuu JPEG-pakkauksen kuvaan aiheuttamista muutoksista. Etenkin JPEG-pakkauksesta aiheutuvien laatikoitumisjälkien vahva jaksoittaisuus vaikeuttaa oleellisesti uudelleen näytteistyksestä aiheutuvan jaksoittaisuuden havaitsemista.

Menetelmään liittyy myös muita rajoituksia. Ensinnäkin tietyt uudelleen näytteistyksen asteet eivät aiheuta lainkaan jaksoittaista riippuvuutta pikseleiden välille. Toinen rajoitus on tietenkin se, että menetelmän avulla ei voida tunnista sellaisia muokkaustoimenpiteitä, joihin ei liity uudelleen näytteistämistä. Toisaalta samasta syystä havaitun muokkauksen tyyppi on mahdollista rajata, ja myös uudelleen näytteistyksen aste on tietyin rajoituksin pääteltävissä. (Popescu 2005, 16)

Gloe, Kirchner, Winkler ja Böhme (2007) ovat esittäneet vastatekniikan, jolla uudelleen näytteistyksen jäljet voidaan peittää varsin tehokkaasti. Menetelmä perustuu karkeasti ottaen siihen, että uudelleen näytteistyksen yhteydessä kunkin pikselin sijaintiin kohdistetaan lievää satunnaista vaihtelua. Periaatteellisella tasolla ajatus on hyvin yksinkertainen, mutta tällaiset toimenpiteet aiheuttavat kuvaan helposti tietynlaista ”värinää”. Oleellista Gloen ym. tekniikassa on se, että se pyrkii jättämään mahdollisimman vähän havaittavia jälkiä sopeuttamalla muutokset kuvan paikallisiin piirteisiin.

3.3.2 Monistettujen alueiden tunnistaminen

Eräs tyypillisimpiä muokkaustapoja on peittää jokin kuvan alue, esimerkiksi henkilö tai esine siten, että peitettävän alueen päälle kopioidaan jokin toinen alue muualta samasta kuvasta. Tällaisia alueiden automaattinen tunnistaminen on ajatuksena melko yksinkertainen, mutta ongelmana on se, että muokatun alueen koko, muoto ja sijainti ovat tuntemattomia. Kaikkien mahdollisten vaihtoehtojen tutkiminen olisi kuitenkin laskennallisesti mahdotonta. (Farid 2009a) Toinen ongelma kyseisessä lähestymistavassa on se, että monistettu alue ei välttämättä ole täysin identtinen alkuperäisen alueen kanssa. Täl-

löin on ratkaistava ovatko kaksi aluetta riittävässä määrin samanlaisia ja toisaalta millä perusteella kyseinen päätös tehdään. (Zhang ym. 2008)

Popescu (2005, 91-102) on esittänyt toteutuksen, jossa kuva jaetaan ensin kiinteän kokoisiin lohkoihin, jonka jälkeen lohkoista erotetaan keskeisimmät piirteet niin sanotulla pääkomponenttianalyysillä. Pääkomponenttianalyysi on muun muassa hahmontunnistuksessakin käytetty tilastollinen menetelmä, jonka avulla suuri joukko muuttujia pyritään korvaamaan pienemmällä joukolla uusia muuttujia, jotka säilyttävät mahdollisimman paljon alkuperäisten muuttujien informaatiosta (Yunfei & Ping 2007). Tällä tavoin epäoleelliset, esimerkiksi kohinan tai pakkauksen aiheuttamat eroavaisuudet voidaan sivuuttaa. Täysin selväksi ei kuitenkaan käy se, minkä tasoiset ja tyyppiset muutokset jäävät tällöin huomioimatta ja mitkä tunnistetaan. Esimerkiksi kopioitavan alueen väriin tai asentoon kohdistuvien muutosten merkitystä ei ole käsitelty Popescun tutkimuksessa. Pääkomponenttianalyysin jälkeen lohkot järjestetään piirteidensä perusteella siten, että samankaltaiset lohkot päätyvät vierekkäin. Lohkojen järjestäminen on laskennalliselta kannalta koko menetelmän vaativin vaihe. Koko algoritmi on aikavaativuudeltaan luokkaa $O(N_t N \log N)$, jossa N_t on pääkomponenttianalyysillä redusoitujen alueiden dimensio ja N kuvan pikseleiden lukumäärä.

Popescun (2005, 93-100) testien mukaan menetelmän tunnistustarkkuus JPEG-muotoisten kuvien suhteen on kohtuullisen hyvä (n. 75 prosenttia) jopa pakkauslaadulla 50, silloin, kun tunnistuksessa käytetään suurehkoa lohkokokoa (160 x 160 pikseliä). Hyvin pieniä (32 x 32 pikseliä) lohkoja käytettäessä pakkauslaadun tulee sen sijaan olla lähes täydellinen. Tulokset eivät merkittävästi poikkea JPEG2000-muotoisten kuvien osalta, joskin pieniä eroavaisuuksia on. Menetelmä on myös sangen vakaa kuvaan lisättyä kohinaa vastaan. Pienimpiä lohkokokoja lukuun ottamatta tunnistustarkkuus on lähellä täydellistä. Kaikissa tilanteissa väärien hälytysten määrä on melko alhainen, korkeimmillaankin vain noin 3 prosenttia.

Kyseisen menetelmän käyttöala rajoittuu luonnollisesti tilanteisiin, joissa alueita on kopioitu yhden kuvan sisällä paikasta toiseen. Tämän muokkaustavan tyypillinen käyttötarkoitus on etenkin elementtien peittäminen kuvassa. Mahdollista on toki myös monistaa jotakin kuvassa esiintyvää elementtiä. Tällöin tosin elementtien yhdennäköisyys saattaa olla liian ilmeistä, jopa silmämääräisen arvion perusteella.

3.4 Kameraan pohjautuvat tekniikat

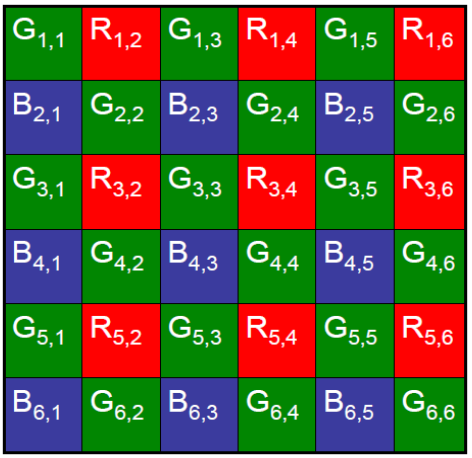
Kameran optiikka sekä eri vaiheissa kuvan prosessointiin käytetyt komponentit jättävät valokuvaan oman jälkensä, jota voidaan käyttää paitsi muokkausjälkien havaitsemiseen, myös tietyn kuvan ottamiseen käytetyn kameran yksilöimiseen.

Esitellyt tekniikat on tarkoitettu nimenomaan digitaalisella kameralla otettujen valokuvien tarkasteluun. Toisin sanoen käyttöalan ulkopuolelle jäävät esimerkiksi perinteisellä filmikameralla otetut kuvat, jotka on myöhemmin skannattu digitaaliseen muotoon.

3.4.1 Värisuotimen interpolaatio

Popescu ja Farid (2005) ovat esitelleet menetelmän, jossa värisuotimen interpolaatiota (CFA interpolation) voidaan käyttää hyväksi kuvan muokkausjälkien havaitsemisessa. Tavallisimmin värisuotimissa käytetään niin sanottua Bayer-matriisia, jossa vihreästä valosta tallennetaan 50 prosenttia, sinisestä ja punaisesta puolestaan 25 prosenttia kustakin. Bayer-matriisin rakenne on esitetty kuviossa 2 (Gallagher & Chen 2008, 1). Koska puuttuvat näytteet joudutaan arvioimaan naapuriarvojen perusteella, aiheuttaa interpolointi tietynlaista jaksoittaista korrelointia pikseleiden välille. Kuvaa muokattaessa tällaiset korrelaatiot hyvin todennäköisesti katoavat tai muuttuvat. Interpolointiin käytettäviä algoritmeja on kuitenkin monenlaisia, toiset yksinkertaisempia, toiset monimutkaisempia. Menetelmän haasteena on se, että lähtökohtaisesti ei ole tiedossa, minkälaista algoritmia interpolointiin on käytetty ja mitkä pikselit korreloivat keskenään.

Pakkaamattomien kuvien tunnistamisessa interpolaatiomenetelmä on Popescun (2005, 66-67) testien mukaan erittäin tarkka. Interpolaatioalgoritmista riippuen tunnistustarkkuus on jopa 97-100 prosenttia. Testauksessa käytetyillä asetuksilla vääriä hälytyksiä ei esiintynyt lainkaan. Myöskään kevyt JPEG-pakkaus ei ole tunnistustarkkuuden kannalta kohtalokasta, vaikkakin interpolaatioon käytetty algoritmi vaikuttaa jonkin verran tähänkin seikkaan. Esimerkiksi pakkauslaadulla 70 menetelmän tunnistustarkkuuden vaihtelu on jo huomattavaa (6-56 prosenttia riippuen interpolaatioalgoritmista). Sen sijaan eräs menetelmän suurimmista heikkouksista on se, että se ei sovellu lainkaan JPEG2000-menetelmällä pakattujen kuvien autentikointiin. Kyseinen pakkausmetodi muodostaa kuvaan jälkiä, joita ei voi erottaa interpolaatiosta.



Kuvio 2: Bayer-matriisin rakenne. (Gallagher & Chen 2008, 1)

Menetelmä ei myöskään luonnollisesti sovellu sellaisilla kameroilla otettuihin kuviin, joissa ei ole värisuodinta. Useimmissa digitaalikameroissa sellainen on, mutta poikkeuksen muodostaa ainakin Sigma Corporationin kehittämä Foveon x3 -kenno, joka tallentaa kolmikerroksisen rakenteensa ansiosta kaikki kolme pääväriä jokaisen pikselin alueelta (Popescu 2005, 67).

Mahdollinen vastatekniikka menetelmää kohtaan voisi olla muokatun alueen interpolointi ohjelmallisesti. Tällöin väärentäjän tulee kuitenkin tuntea kuvan interpolointiin alunperin käytetty algoritmi. Interpolaation aiheuttamat korrelaatiot on myös mahdollista tuhota kuvan kokoa pienentämällä. (Popescu 2005, 59, 68)

3.4.2 Kohinaan perustuvat tekniikat

Kaikissa digitaalisissa valokuvissa esiintyy tietty määrä *kohinaa*, eli satunnaista ja ei-toivottua vaihtelua pikseleiden kirkkaudessa. Kohinaa aiheuttavat erinäiset kameran komponentit ja kuvan prosessointivaiheet. Osa kohinasta on täysin satunnaista ja vaihtelee otoksittain, osa puolestaan on kamerakohtaista ja otoksesta toiseen pysyvää. Kohinaa on mahdollista hyödyntää eri tavoin valokuvien autentikoinnissa. (Fridrich 2009)

Fridrichin (2009, 26-27) mukaan oleellisin osa kohinasta johtuu kuvakennosta. Hän käyttää siitä termiä *photo-response nonuniformity* (PRNU). Ilmiö johtuu kennon pikseleiden lievästi erilaisesta kyvystä muuttaa fotoneita elektroneiksi. Tämä puolestaan joh-

tuu epätäydellisestä valmistusprosessista sekä kennon valmistusmateriaalina käytetyn piin luonteenomaisesta epätasalaatuisuudesta. Jokaiseen kuvaan muodostuu kennon jättämä heikko kohinakuvio. Kuvio voidaan ajatella eräänlaisena sormenjälkenä, jonka perusteella kuvan aitoudesta voidaan tehdä päätelmiä, sekä lisäksi tietyn kuvan ottamiseen käytetty kamera voidaan yksilöidä. Fridrichin mukaan tällaisella sormenjäljellä on monia hyödyllisiä ominaisuuksia. Se on ensinnäkin satunnaisesta luonteestaan johtuen yksilöllinen jokaiselle kennolle. Sormenjälki myös esiintyy kaikissa kennoissa sekä kaikissa kuvissa, riippumatta kameran optiikasta tai asetuksista, eikä se juurikaan muutu ajan tai olosuhteiden vaikutuksesta. Lisäksi sormenjälki kestää kohtuullisen hyvin erilaisia prosessointitoimenpiteitä, kuten pakkausta.

PRNU-jälki voidaan ajatella ikään kuin kameran tahattomasti kuvaan jättämänä vesileimana (Chen ym. 2007). Erona on tietenkin se, että tässä tapauksessa ”vesileiman” sisältö on täysin satunnainen, eikä sen tarkkaa sijaintia tunneta lähtökohtaisesti. Jäljen erottamiseksi kuvasta muodostetaan sopivan suodattimen avulla kohinaton versio. Arvio kohinasta saadaan tällöin vertaamalla alkuperäistä ja kohinatonta kuvaa (Chen ym. 2007). Koska kaikissa kuvissa on myös täysin satunnaista kohinaa, ei PRNU-komponentin erottaminen onnistu yksittäisestä kuvasta. Hyvä arvio sormenjäljestä vaatii jopa 20-50 muokkaamatonta kuvaa (Fridrich 2009).

Muokattujen alueiden tunnistaminen PRNU-jäljen avulla perustuu siihen, että jäljen esiintymistä tutkitaan kuvan eri alueilla. Jos jälki puuttuu tietyltä alueelta, voidaan alue päätellä muokatuksi. Menetelmällä ei voida kuitenkaan havaita sellaisia muokkaustoimenpiteitä, jotka eivät poista PRNU-jälkeä. Tällaisia ovat esimerkiksi kirkkauden, värien tai kontrastin säätäminen. (Chen ym. 2007) On tietenkin huomattava, että edellä mainittujen operaatioiden osalta varsin oleellista on muutosten aste. On luultavaa, että tekijät viittaavat nimenomaan tilanteisiin, joissa muutokset ovat suhteellisen lieviä.

Chen ym. (2007) ovat testanneet menetelmänsä tarkkuutta JPEG-pakattujen kuvien suhteen. Pakkauksen laadun ollessa 90, menetelmä tunnisti 85 prosentissa tapauksista vähintään kaksi kolmasosaa muokatuista alueista. Laadulla 75 vastaava luku oli 73 prosenttia. Menetelmän tunnistusherkkyyteen voidaan vaikuttaa eräänlaisella kynnysarvolla. Maltillisemmilla arvoilla muokatut alueet voivat jäädä tunnistamatta ja herkemmillä arvoilla aiheettomia tunnistuksia voi esiintyä. Tekijöiden mukaan lähes kaikki virheet

tunnistuksessa liittyivät kuitenkin tilanteisiin, joissa kuva sisälsi hyvin tummia ja yleensä rakenteeltaan monimutkaisia alueita. PRNU-jälki ei tavallisesti esiinny tällaisilla alueilla. Tunnistustarkkuuteen vaikuttaa myös käytetty lohkokoko. Pienistä lohkoista on vaikea arvioida luotettavasti PRNU-jäljen esiintymistä, ja suuret lohkot puolestaan heikentävät tunnistuksen alueellista tarkkuutta. Hyvä tasapaino saadaan tekijöiden mukaan 64 x 64 – 128 x 128 pikselin kokoisilla lohkoilla.

Useiden muokkaamattomien kuvien tarve heikentää menetelmän yleiskäyttöisyyttä huomattavasti, ja rajoittaa sen tilanteisiin, joissa joko kuvan ottamiseen käytetty kamera tai muita kyseisellä kameralla otettuja kuvia on saatavilla. Tekniikka ei ole myöskään erityisen nopea, sillä yksittäisen kuvan tarkastaminen voi kestää joitakin minuutteja. (Chen ym. 2007)

Tekijät eivät ole juurikaan käsitelleet mahdollisia menetelmäänsä kohdistuvia vastatekniikoita. Ei kuitenkaan vaikuttaisi epärealistiselta olettaa, että PRNU-jälki olisi kohtuullisen helposti väärennettävissä muokattuun kuvaan, mikäli myös väärentäjällä on käytössään joukko samalla kameralla otettuja aitoja kuvia.

Mahdian ja Saic (2009) ovat esittäneet kohinaan perustuvan menetelmän, joka ei vaadi ennakkotietoa tutkittavasta kuvasta tai sen ottamiseen käytetystä kamerasta. Tekniikka perustuu siihen havaintoon, että kohinataso on yleensä tasainen kaikilla kuvan alueilla. Niinpä paikallisia muutoksia kohinan tasossa voidaan käyttää hyväksi muokattujen alueiden tunnistamisessa. Menetelmä jakaa kuvan kohinatasoltaan samankaltaisiin alueisiin. Kohinan estimointiin käytetään aallokemuunnosta, jolla kuvan korkeat taajuudet voidaan erottaa. Tekijöiden mukaan menetelmä on hyödyllinen etenkin tilanteissa, joissa muokkausjälkiä pyritään peittämään lisäämällä kuvaan paikallisesti kohinaa. Kohinan taso saattaa muuttua myös, mikäli kuvaan lisätään elementtejä vaikkapa toisista kuvista tai ohjelmallisesti generoimalla.

Mahdian ja Saic ovat testanneet menetelmänsä kykyä arvioida kohinan keskihajontaa tietyllä kuvan alueella, mutta he eivät suoranaisesti esitä tutkimustuloksia siitä, kuinka tarkasti heidän menetelmänsä kykenee tunnistamaan väärennettyjä kuvia. Testien perusteella keskihajonta on arvioitavissa melko tarkasti, jos tarkasteltavissa kuvissa on käytetty maltillista pakkaustasoa (esim. JPEG-pakkauksen taso 90). Tarkkuuteen vaikuttaa kohinan vaihtelun aste sekä se, kuinka suuri on tarkasteltava alue. Jos kohinan vaihtelu

on huomattavaa ja tarkasteltava alue on suurehko (128 x 128 pikseliä), ei voimakkaampikaan pakkaus välttämättä haittaa.

Mahdianin ja Saicin mukaan heidän menetelmänsä suurin ongelma on se, että myös täysin aidot kuvat voivat sisältää alueita, joissa kohinan taso poikkeaa toisistaan. Tällaiset tapaukset voivat aiheuttaa vääriä hälytyksiä.

3.5 Yhteenveto autentikointimenetelmien ominaisuuksista

Taulukkoon 6 on tiivistetty edellä esiteltyjen tekniikoiden tärkeimmät ominaisuudet. Ehkäpä kaikkein tärkein arviointikriteeri on tekniikoiden tarkkuus. Kuten todettu, tarkkuuden arviointi ei kuitenkaan ole aivan yksinkertaista, johtuen muun muassa tekniikoiden kehittäjien epäyhtenäisistä testauskäytännöistä. Tästä johtuen esitettyjä arvioita on tulkittava maltillisesti. Myöskään tunnistettuja muokkaustyyppejä ei ole kovinakaan tarkasti käsitelty kaikissa alkuperäislähteissä.

Taulukko 6: Yhteenveto tarkasteltujen autentikointimenetelmien tärkeimmistä ominaisuuksista.

Tekniika	Tunnistetut muokkaustyypit	Tarkkuus		Rajoitukset
		Pakkaamattomat kuvat	JPEG-pakatut kuvat	
Uudelleen näytteistyksen tunnistaminen	Kuvan alueiden pienetäminen, suurentaminen tai kallistaminen.	Lähes 100%	Laadulla 97 tarkkuus on pääosin hyvä, laadulla 90 jo lähes sattuman tasolla.	Kaikki uudelleen näytteistyksen asteet eivät ole tunnistettavissa.
Monistettujen alueiden tunnistaminen	Alueen monistaminen saman kuvan sisällä.	Lähes 100%	Lähes 100% kun laatu on yli 80. Laadulla 50 n. 75%.	
Värisuotimen interpolaatio	Yleinen	97-100%	Lähes 100% kun laatu on yli 96. Laadulla 70 6-56% riippuen interpolaatio-algoritmista.	Ei sovellu JPEG2000-muotoisille kuville. Ei sovellu kameroille, joissa ei ole värisuodinta.
PRNU-jälki	Yleinen, pois lukien värien, kirkkauden ja kontrastin muutokset.	Ei ilmoitettu	85% prosenttia laadulla 90, 73% laadulla 75. (tunnistettu väh. 2/3 muokatuista alueista)	Vaatii suuren määrän muokkaamattomia kuvia, jotka on otettu samalla kameralla kuin käsiteltävä kuva. Laskennallisesti raskas.
Kohinan tason arviointi	Muutokset kohinan tasossa, etenkin kohinan paikallinen lisääminen.	Vertailuun soveltuvia testituloksia ei ole esitetty.		Normaalit kohinan tason vaihtelut saattavat aiheuttaa vääriä hälytyksiä.

4 TOTEUTETTUJEN AUTENTIKOINTIMENETELMIEN YKSITYISKOHDAT

Tässä luvussa esittelen neljä toteuttamaani valokuvien autentikointiin tarkoitettua menetelmää, joista kolme on lähes sellaisenaan esitelty aiemmin alan kirjallisuudessa. Neljäs on oma muunnokseni yhdestä edellä mainituista. Ensimmäisen menetelmän ovat alunperin esitelleet He, Lin, Wang ja Tang (2006). Menetelmä perustuu siihen, että kuvaan kahdesti suoritettu kvantisointi voidaan tunnistaa, ja sen avulla voidaan tehdä erinäisiä päätelmiä. Samasta menetelmästä on kirjoitettu myös tuoreempi tutkimus (Lin, He, Tang & Tang 2009). Oma muunnelmani perustuu jossain määrin kyseiseen menetelmään. Siinä ajatuksena on tutkia kuvan diskreetin kosinimuunnoksen painokertoimien poikkeuksellisen matalia arvoja. Kolmannen toteutettavan menetelmän ovat esitelleet Li, Yuan ja Yu (2009) ja se perustuu JPEG-pakkauksen aiheuttaman laatikoitumisen tunnistamiseen ja hyödyntämiseen. Myös neljännen menetelmän ovat esitelleet Li ym. (2008). Siinäkin ajatuksena on tunnistaa laatikoitumista, mutta pikseleiden sijasta siinä hyödynnetään kuvan kosinimuunnoksen painokertoimia.

Toteutetut menetelmät voidaan toimintaperiaatteidensa perusteella jakaa kahteen eri ryhmään. Hen ym. (2006) menetelmän ja siihen perustuvan oman muunnelmani toiminta perustuu kuvan kosinimuunnoksen painokertoimien tarkasteluun. Myös kvantisointi on oleellisessa roolissa näissä tekniikoissa. Lin ym. (2008 ja 2009) menetelmissä ydinajatus puolestaan on JPEG-pakkauksen muodostaman laatikoitumisen tunnistaminen. Itse asiassa myös toinen näistä menetelmistä tarkastelee kuvan kosinimuunnoksen painokertoimia. Kosinimuunnoksen painokertoimia hyödyntävillä menetelmillä viittaan kuitenkin jatkossa nimenomaan kahteen ensin mainittuun.

Tarkoitukseni ei ole esittää tapaa, jolla käsitellyt yksittäiset autentikointimenetelmät voitaisiin yhdistää yhtenäiseksi, saumattomaksi kokonaisuudeksi. Sen sijaan voidaan ajatella, että menetelmät muodostavat eräänlaisen analyysityökalun. Näin ollen analyysin suorittaja voisi suorittaa erilaisia testejä ja muodostaa niiden tuloksia tulkitsemalla kokonaiskuvan. Osa toteutetuista menetelmistä voisi toki ainakin periaatteessa soveltua kohtuullisesti myös kuvien täysin automatisoituun käsittelyyn, mutta osa edellyttää välttämättä ihmisen suorittamaa tulkintaa.

Tarkoitukseni on kuvata menetelmät sellaisella tarkkuustasolla, että lukijalle välittyy käsitys niiden oleellisimmista yksityiskohdista. Sen sijaan jätän käsittelemättä suuren määrän epäolennaisempia, jossain määrin rutiiniluonteisiin tehtäviin liittyviä yksityiskohtia.

Kuten todettu, kaikki tekniikat hyödyntävät nimenomaan JPEG-pakkausta. Näin ollen voitaisiin päätellä, että muut kuin JPEG-muotoiset kuvat olisi luonnollista jättää kokonaan tarkastelun ulkopuolelle. Näin ei kuitenkaan ole asian laita. Lähtökohtaisena edellytyksenä on se, että alkuperäinen kuva on ennen mahdollisia muokkaustoimenpiteitä ollut JPEG-muotoinen. Kuvan viimeisimmällä, eli muokkauksen jälkeisellä tallennusmuodolla ei sen sijaan ole niinkään ratkaisevaa merkitystä. Itse asiassa aitouden arvioinnin kannalta on jopa parempi, jos tarkasteltava kuva on viimeisimmäksi tallennettu häviötöntä tiedostomuotoa käyttäen. Tällöin aiempien pakkauskertojen aiheuttamat jäljet säilyvät paremmin. On tietenkin kokonaan toinen asia, kuinka tyypillinen tällainen tilanne todellisuudessa on. Muut häviölliset tiedostomuodot eivät kuitenkaan luultavasti tule kyseeseen muokkauksen jälkeisenä muotona. Esimerkiksi JPEG2000-pakkauksessa ei välttämättä käytetä 8 x 8 pikselin lohkojakoa (Skodras, Christopoulos & Ebrahimi 2001). On varsin luultavaa, että tällöin käsitellyt autentikointimenetelmät eivät toimisi odotetulla tavalla.

Kaikki jatkossa esiintyvät väärennösten tunnistukseen liittyvät tuloskuvat sekä erilaiset graafit on luotu toteuttamassani testausympäristössä.

4.1 Toteutuksen teknisistä yksityiskohdista

Autentikointimenetelmien toteutuksessa on käytetty Microsoftin .NET-ympäristöä (versio 3.5) ja C#-kieltä. .NET-ympäristön vakiokirjastojen lisäksi mitään ulkopuolisia ohjelmakirjastoja ei ole käytetty.

Liitteessä 3 on esitetty pseudokoodina eri autentikointimenetelmien keskeisimmät algoritmit. Algoritmeja on jossain määrin yksinkertaistettu, jotta kaikkein tärkeimpien vaiheiden hahmottaminen olisi helpompaa.

Kaikki testiaineistoon kuuluvat kuvat on tallennettu kahdeksan bitin näytetarkkuutta käyttäen. Kaikki toteutetut menetelmät tarkastelevat ainoastaan kuvan Y-värikanavaa. JPEG-kuvissa Y-kanava sisältää tyypillisesti kaikkein merkittävimmän osan informaa-

tiosta. Y-komponentti saadaan muodostettua RGB-väriarvojen pohjalta seuraavalla kaavalla:

$$Y = 0{,}299\,R + 0{,}587\,G + 0{,}114\,B \quad .$$

Kaavassa R tarkoittaa punaisen, G vihreän ja B sinisen värikomponentin arvoa (Miano 1999, 6).

4.2 Kaksoiskvantisoinnin tunnistaminen

Ensimmäisen toteutettavan menetelmän ovat alunperin esitelleet He ym. (2006). Menetelmä hyödyntää ilmiötä nimeltä kaksoiskvantisointi. Menetelmän kuvaus perustuu edellä mainitun tutkimuksen lisäksi Linin ym. (2009) tuoreempaan tutkimukseen.

4.2.1 Menetelmän kuvaus

Popescu ja Farid (2004) ovat havainneet, että kahdesti samaan kuvaan suoritettu kvantisointi aiheuttaa kuvaan eräänlaista jaksoittaista säännönmukaisuutta. Ilmiö on havaittavissa kuvan diskreetin kosinimuunnoksen painokertoimien histogrammista. Histogrammi muodostetaan vastaavalla tavalla kuin kuvan valoisuusarvojen histogrammi, sillä erotuksella, että tässä tapauksessa tutkitaan kosinimuunnoksen painokertoimien arvojen esiintymistaajuuksia. Kuten kohdassa 2.4.2 todettiin, kvantisointi tarkoittaa itse asiassa lukujen jakamista keskenään. Jakajasta käytetään usein nimitystä *kvantisointiaskel* (quantization step). Koska kertoimet pyöristetään jakamisen jälkeen lähimpään kokonaislukuarvoon ja kerrotaan jälleen käytetyllä kvantisointiaskeleella *q1*, on seurauksena se, että kaikki kertoimet ovat arvoltaan joko nollia tai *q1*:n monikertoja. Jos sama signaali kvantisoidaan uudelleen, jakautuvat kertoimet uudella tavalla, riippuen nyt käytetystä kvantisointiaskeleesta *q2*.

Popescun ja Faridin alkuperäinen ajatus oli yksinkertaisesti se, että mikäli kuva havaitaan kahdesti pakatuksi, voi olla syytä epäillä sen aitoutta. Kaksinkertainen pakkaus ei tietenkään yksistään ole erityisen pätevä peruste luokitella kuvaa väärennetyksi. Se voi kuitenkin antaa aihetta tarkemmille tutkimuksille. Hen ym. (2006) esittämä tekniikka jalostaa samaa ajatusta huomattavasti pidemmälle. He kuitenkin tarkastelevat ilmiötä tietyllä tavalla päinvastaisesta näkökulmasta. Heidän mukaansa ainoastaan kerran kvanti-

soitujen alueiden voidaan epäillä olevan väärennettyjä, mikäli muut kuvan alueet on kvantisoitu kahdesti.

Normaalitapauksessa kaikkien kertoimien jakautuminen histogrammin eri lokeroihin riippuu siis kvantisointiaskelista *q1* ja *q2*. Jos taas kuvaa käsitellään ensimmäisen kvantisoinnin jälkeen, riippuukin muokatun alueen kertoimien jakautuminen enää askeleesta *q2*. On nimittäin varsin epätodennäköistä, että muokattu alue vastaisi painokertoimien jakautumisen suhteen kuvan alkuperäisiä alueita. Esimerkiksi pakkaamattomasta kuvasta kopioitu tai ohjelmallisesti generoitu alue ei sisällä lainkaan alkuperäisiä pakkausjälkiä. Kaksoiskvantisoinnin aiheuttamaa ilmiötä ei myöskään synny, mikäli muokatun alueen kosinimuunnoksessa käytetyt lohkot eivät ole täysin kohdakkain ensimmäisessä pakkauksessa käytettyjen lohkojen kanssa. (He ym. 2006)

Tarkkaan ottaen menetelmän toimintaperiaate ei edellytä kahta kvantisointikierrosta. Oleellista on vain se, että kertaalleen kvantisoitua kuvaa käsitellään. Tämän jälkeen kuva voidaan tallentaa yhtä hyvin häviölliseen kuin häviöttömäänkin muotoon. Itse asiassa häviöttömän tallennusmuodon käyttäminen muokkaamisen jälkeen helpottaa muokattujen ja muokkaamattomien alueiden erottamista. Tämä johtuu siitä, että kaikki alkuperäiset kertoimet ovat tällöin *q1*:n monikertoja, kun taas muokatun alueen kertoimet jakautuvat täysin satunnaisesti. Sen sijaan kahden kvantisointikierroksen tilanteessa molemmat käytetyt kvantisointiaskeleet vaikuttavat lopputulokseen. On esimerkiksi selvää, että jos *q1* ja *q2* ovat yhtä suuret, minkäänlaista eroa ei voida havaita kertoimien jakautumisessa.

4.2.2 Toteutuksen vaiheet

Ensimmäinen vaihe menetelmän toteutuksessa on diskreetin kosinimuunnoksen painokertoimien kokoaminen histogrammeiksi. Kertoimet on mahdollista lukea varsin nopeasti suoraan JPEG-tiedostosta, joten erillistä kosinimuunnosta ei välttämättä tarvitse suorittaa. Jos käsitellään muuta, kuin JPEG-tiedostoa, Lin ym. (2009) ehdottavat, että tiedosto muunnetaan JPEG-muotoon parhaalla mahdollisella pakkauslaadulla. He eivät ota kantaa siihen, onko muunnoksella vaikutusta tunnistustarkkuuteen. Kosinimuunnos on kuitenkin mahdollista suorittaa myös erikseen, ilman varsinaista JPEG-muunnosta. Tällöin kertoimet saadaan muodostettua ilman uutta kvantisointia. Omassa toteutuksessani käytetään itseasiassa tätä menettelyä riippumatta siitä, onko kyseessä JPEG-muotoi-

nen kuva vai ei. Tämä ei tietenkään suoritustehokkuuden kannalta ole paras mahdollinen tapa. Johtuen JPEG-tiedostojen varsin monimutkaisesta rakenteesta, en kuitenkaan kokenut järkeväksi käyttää aikaa JPEG-pakkauksen purkualgoritmin laatimiseen. .NET-ympäristön vakiokirjastojen avulla JPEG-tiedostoja on toki mahdollista lukea, mutta tällöin päästään käsiksi ainoastaan lopullisiin pikseleiden arvoihin, eikä lainkaan tässä tapauksessa kiinnostaviin kosinimuunnoksen painokertoimien arvoihin.

Edellä kuvattu poikkeaminen alkuperäisestä toteutuksesta aiheuttaa erään käytännön eroavaisuuden histogrammien rakenteeseen. Kun painokertoimet luetaan suoraan JPEG-tiedostosta, niitä ei ole *dekvantisoitu*, eli kerrottu uudelleen niitä vastaavalla kvantisointiarvolla. Tämä vaihe nimittäin tapahtuu vasta JPEG-kuvaa purettaessa. Tästä johtuen kertoimien arvot eivät tässä vaiheessa noudata erityistä jaksollisuutta, vaan ne voivat olla peräkkäisiä kokonaislukuja. Jaksollisuus ilmenee siis vain, mikäli arvot ovat kvantisoitu kahdesti. Omassa toteutuksessani arvot on laskettu puretun kuvainformaation perusteella, joten siinä vaiheessa kertoimien arvot on jo dekvantisoitu. Tästä johtuen kertoimet ovat siis käytetyn kvantisointiarvon monikertoja. Näin ollen kertoimien histogrammissa on siis joka tapauksessa aukkoja, riippumatta siitä esiintyykö kaksoiskvantisointia. Kaksoiskvantisoinnin aiheuttama jaksollisuus on kuitenkin havaittavissa normaalin jaksollisuuden ohessa.

Kuhunkin histogrammiin kootaan yhtä taajuutta vastaava kerroin jokaisesta kosinimuunnoksessa muodostetusta lohkosta. Jokaisessa lohkossa on yhteensä 64 kerrointa kohdissa (0, 0) – (7, 7). Jokaista kolmea värikanavaa varten voidaan myös muodostaa omat histogrammit. Näin ollen voidaan periaatteessa muodostaa yhteensä 64 x 3, eli 192 eri histogrammia. Olen kuitenkin rajannut omassa toteutuksessani tarkastelun Y-komponenttiin. Lisäksi korkeimpia taajuuksia vastaavien kertoimien arvot ovat kvantisoinnin seurauksena usein nollia, joten niiden osalta ei välttämättä saada arvioitua käytettyä kvantisointiaskelta.

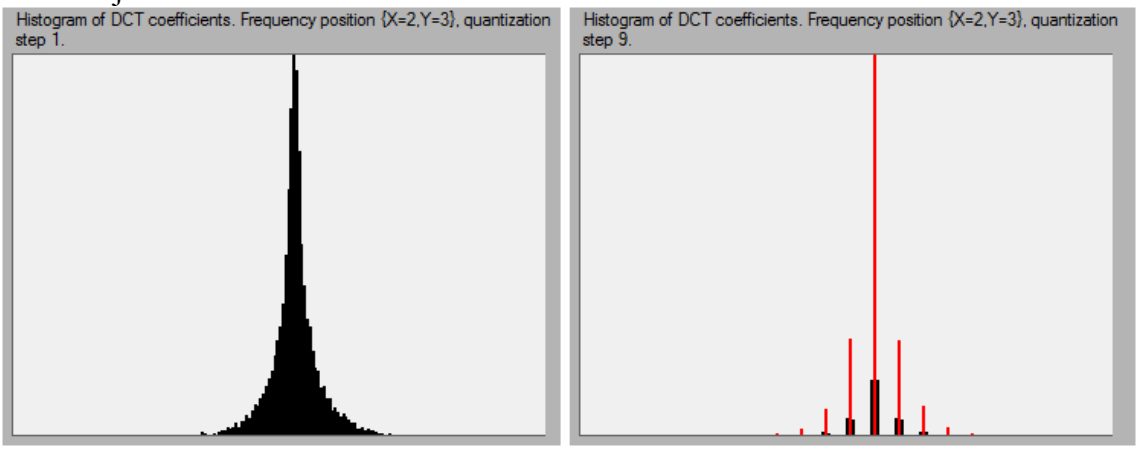
Seuraavaksi on arvioitava jokaisesta histogrammista kvantisoinnin aiheuttama jaksollisuus. Linin ym. (2009) mukaan jakson pituutta voidaan arvioida ensinnäkin kokeilemalla, kuinka hyvin eri pituiset jaksot kerryttävät histogrammin korkeita huippuja. Laskenta aloitetaan tällöin histogrammin korkeimmasta huipusta. Jakson pituutta voidaan arvioida myös Fourier-muunnoksen avulla. Tätä mahdollisuutta Linin ym. (2009) sen parem-

min kuin Hen ym. (2006) artikkeleissa ei kuitenkaan kuvata erityisen tarkasti. Muun muassa Ye, Sun ja Chang (2007) ovat käsitelleet aihetta tarkemmin. Heidän mukaansa histogrammin Fourier-muunnoksen tehospektrissä esiintyvien huippujen lukumäärää lisättynä yhdellä kertoo jakson pituuden. Fourier-muunnoksen tehospektri puolestaan määritellään seuraavasti:

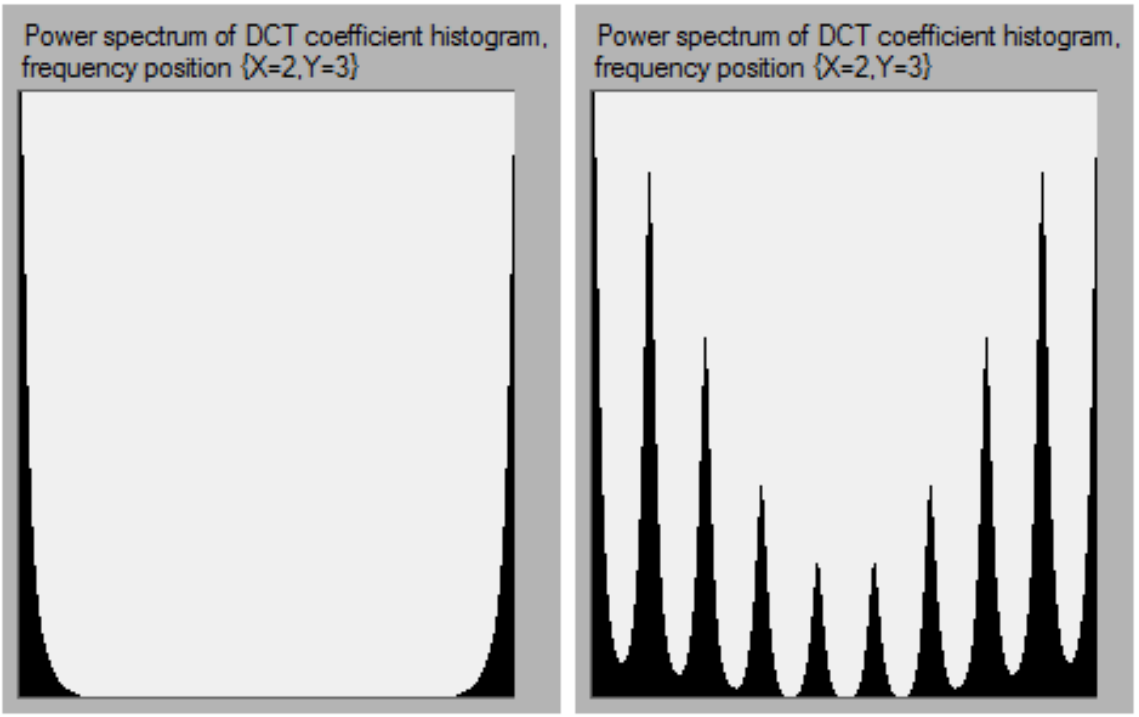
$$P(u) = \left| F(u)^2 \right| = R^2(u) + I^2(u) \quad ,$$

jossa *R* tarkoittaa kunkin Fourier-kertoimen reaaliosaa ja *I* imaginaariosaa (Gonzales & Woods 2002, 152).

Kuten sanottu, kaikista histogrammeista ei välttämättä saada järkevää arviota jakson pituudesta. Siinä tapauksessa kyseinen histogrammin täytyy jättää pois käsittelystä. Kuviossa 4 näkyy pakkaamattoman (PNG) ja pakatun (JPEG, laatu 80) kuvan diskreetin kosinimuunnoksen painokertoimien histogrammit. Kuviossa 5 puolestaan kyseisistä histogrammeista on muodostettu Fourier-muunnoksen tehospektrit. Pakkaamattoman kuvan tapauksessa tehospektrissä ainoat huiput näkyvät kuvan reunoilla, kun taas pakatussa kuvassa on nähtävissä useita huippuja. Kyseisessä kuviossa on kahdeksan huippua, mikä tarkoitta sitä, että vastaava kvantisointiaskel on 9. Kaikissa tapauksissa kuvio ei kuitenkaan ole yhtä siisti ja helposti tulkittava. Tämän vuoksi tehospektriä on syytä siistiä suodattamalla pienimpiä arvoja pois. Kuviossa 4 on merkitty oikean puoleiseen histogrammiin punaisella ne huiput, jotka noudattavat havaittua jaksollisuutta, eli ovat yhdeksällä jaollisia.



Kuvio 4: Kuvan diskreetin kosinimuunnoksen painokertoimista muodostettu histogrammi. Vasemmalla on pakkaamattomasta PNG-kuvasta ja oikealla JPEG-kuvasta (laatu 80) muodostettu histogrammi.



Kuvio 5: Kuvan diskreetin kosinimuunnoksen painokertoimien histogrammista muodostettu Fourier-muunnoksen tehospektri . Vasemmalla tehospektri on pakkaamattomasta ja oikealla JPEG-pakatusta kuvasta.

Menetelmän ydinajatus on se, että aidot ja muokatut lohkot kerryttävät histogrammin lokeroita eri tavoin. Aidot lohkot kerryttävät korkeita huippuja, kun taas muokatut lohkot kerryttävät histogrammin lokeroita jossain määrin satunnaisemmin. Jos oletetaan, että tietty jakso p (period) alkaa histogrammin lokerosta s_0 ja loppuu lokeroon s_0+p-1 , voidaan aidolle lohkolle, joka kerryttää kohtaa s_0+i , laskea todennäköisyys seuraavalla kaavalla:

$$P_a(s_0+i)=h(s_0+i)/\sum_{k=0}^{p-1} h(s_0+k) \quad ,$$

jossa *h(k)* tarkoittaa esiintymien lukumäärää histogrammin lokerossa *k*. Toisin sanoen kyseinen luku vastaa tietyn arvoisten painokertoimien lukumäärää histogrammissa. Esitetty laskukaava ei ole itseasiassa aivan täsmällinen tapa laskea kyseistä todennäköisyyttä. Kaava antaa nimittäin korkeampia arvoja kaikkien huippujen kohdalla. Täsmällisemmän laskutavan käyttö kuitenkin edellyttäisi, että tunnettaisiin kvantisointiarvot *q1* ja

q2. Itse asiassa *q2*:n arvo saadaan suoraan JPEG-tiedostosta, mutta *q1* joudutaan arvioimaan.

Muokatun lohkon todennäköisyys puolestaan saadaan seuraavasti:

$$P_m(s_0+1)=1/p \quad .$$

Bayesilaisen päättelyn perusteella sekä muokatulle, että aidolle lohkolle voidaan laskea posterioritodennäköisyydet seuraavasti:

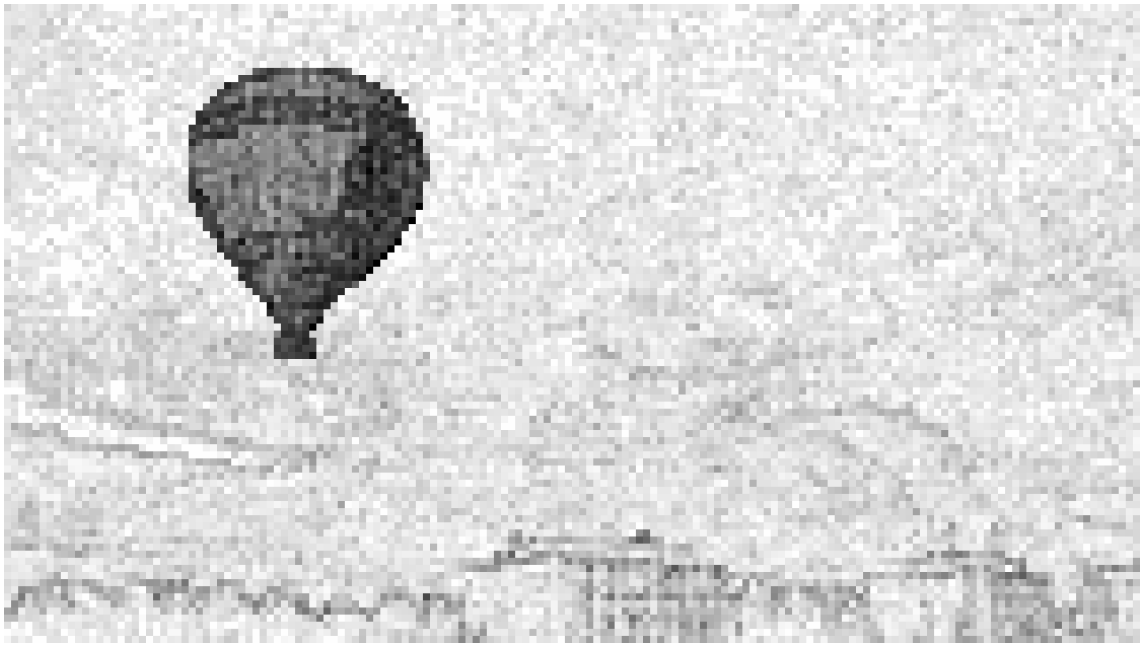
$$P(muokattu|s_0+i)=P_m/(P_m+P_a) \quad ,$$

$$P(aito|s_0+i)=P_a/(P_m+P_a) \quad .$$

Yhden lohkon koko todennäköisyys saadaan lopulta yhdistämällä jokaisen histogrammin perusteella saadut todennäköisyydet.

Alkuperäisessä toteutuksessa tulosten tulkinnassa on hyödynnetty koneoppimista. Tältä osin oma toteutukseni poikkeaa alkuperäisestä. Tämän vuoksi saavutetut tulokset eivät tietenkään ole täysin vertailukelpoisia. Menetelmän pääasiallinen toimintaperiaate vastaa kuitenkin alkuperäistä toteutusta.

Kuvassa 6 on esimerkki aidosta ja väärennetystä valokuvasta. Vasemmalla puolella on alkuperäinen kuva ja oikealla kuvaan on liitetty kuumailmapallo toisesta kuvasta. Alkuperäinen kuva on tallennettu JPEG-muotoon laadulla 80. Muokkauksen jälkeen kuva on tallennettu häviöttömään PNG-muotoon. Kuvassa 7 kyseinen väärennös on tunnistettu Hen ym. (2006) kaksoiskvantisointiin perustuvalla menetelmällä. Muun muassa kuvan rakennusten rajat näkyvät vaimeasti tuloskuvassa, mutta väärennetty alue erottuu silti varsin selvästi yhtenäisenä tummana alueena.



Kuva 7: Kuvan 6 väärennös tunnistettu Hen ym. (2006) kaksoiskvantisointiin perustuvalla menetelmällä.

4.3 Kuvan diskreetin kosinimuunnoksen painokertoimien poikkeuksellisen alhaisten arvojen tunnistaminen

Tässä osiossa kuvailtavaa autentikointimenetelmää ei tietääkseni ole aiemmin esitetty alan kirjallisuudessa. Kyseessä on tietyssä mielessä yksinkertaistettu versio Hen ym. (2006) menetelmästä.

4.3.1 Menetelmän kuvaus

Nyt esittämäni menetelmä hyödyntää sitä seikkaa, että kvantisoinnin seurauksena kutakin kosinimuunnoksen taajuutta vastaavien painokertoimien arvot saavat tietyn alarajan, jota suurempia niiden tulee olla. Tästä huolimatta painokertoimen arvo voi tietenkin aina olla myös nolla. Kohdassa 2.4.2 kuvaillun kvantisointimenettelyn perusteella voidaan ajatella tilannetta, jossa painokerroin c kvantisoidaan arvolla q. Kvantisoinnin tuloksena saadaan

$$c' = [c/q] \quad ,$$

jossa *[a]* tarkoittaa pyöristämistä lähimpään kokonaislukuun. Kun c' kerrotaan jälleen kvantisointiarvolla q, saadaan dekvantisoitu arvo

$$c'' = c'q \quad .$$

Koska c' on kokonaisluku, voidaan edellisestä päätellä, että itse asiassa q määrittelee siis painokertoimien arvon alarajan.

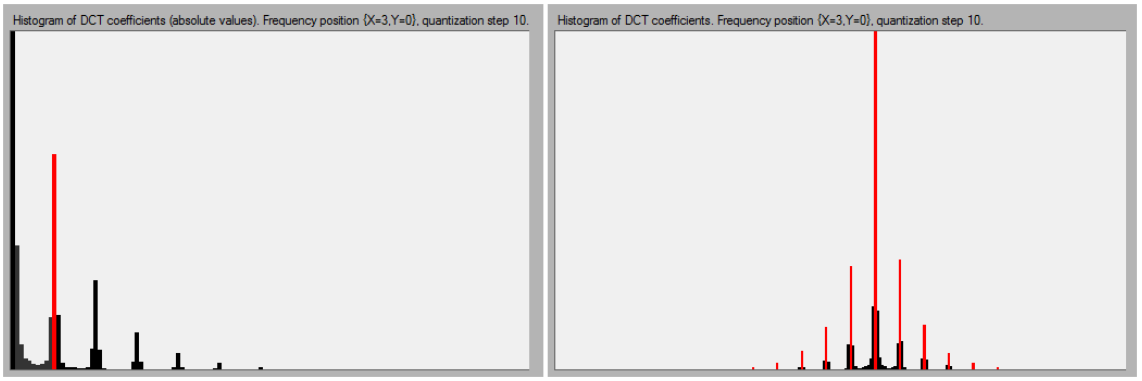
Välttämättömänä edellytyksenä esittämäni menetelmän toiminnalle on se, että tarkasteltava kuva on ennen muokkaamista tallennettu heikommalla laadulla kuin muokkaamisen jälkeen. Menetelmä hyödyntää sitä seikkaa, että muokattujen alueiden painokertoimet voivat olla arvoltaan pienempiä kuin kuvan alkuperäisten alueiden kertoimet. Kuten edellä todettiin, kvantisointiarvo q määrittelee itse asiassa alarajan painokertoimien arvoille. Näin ollen muokattujen alueiden painokertoimet voivat olla alkuperäisen kuvan kertoimia pienempiä siinä tapauksessa, että $q2 < q1$. Kuten todettu, tarkkaan ottaen kaikki kertoimet saavat tietenkin arvoja, jotka ovat viimeksi käytetyn kvantisointiarvon monikertoja. Jos kvantisointiarvot ovat lähellä toisiaan, on tietenkin mahdollista, että sekä muokattujen, että alkuperäisten alueiden kertoimet pyöristyvät saman alarajan kohdalle.

4.3.2 Toteutuksen vaiheet

Kuten edellä kuvaillussa Hen ym. (2006) menetelmässä, tässäkin tapauksessa toteutus aloitetaan muodostamalla histogrammit kuvan kosinimuunnoksen painokertoimista. Päädyin kuitenkin tässä yhteydessä käyttämään kertoimien itseisarvoja histogrammin muodostamiseen. Painokertoimien histogrammi on tavallisesti jakautunut siten, että suurin osa arvoista on nollia ja loput arvot ovat käytetyn kvantisointiarvon monikertoja. Arvot voivat siis olla myös negatiivisia, mutta histogrammi on kvantisointiaskeleiden pituuden suhteen symmetrinen nollan molemmin puolin. Käyttämällä itseisarvoja voidaan hyödyntää kaikki arvot, mutta tarkastelu voidaan rajata histogrammin positiiviselle puolelle. Tällöin negatiiviset ja positiiviset arvot kerääntyvät siis samaan lokeroon. Tästä ei ole kuitenkaan haittaa, sillä oleellista on nimenomaan kvantisointiaskeleen pituus, eli se, kuinka etäällä histogrammin pylväät ovat toisistaan.

Muodostuneen histogrammin korkein huippu näyttäisi siis olevan aina nollan kohdalla. Olen havainnut, että seuraavaksi korkein huippu osuu kohtaan, joka on tarkasteltavaa taajuutta vastaavien painokertoimien vähimmäisarvo. Normaalissa tapauksessa kyseisen huipun ja nollan välillä ei tulisi olla lainkaan esiintymiä. Mikäli kuvaa on kuitenkin kä-

sitelty, ja tämän jälkeen on käytetty korkeampaa pakkauslaatua, eli pienempää kvantisointiarvoa, muodostuu kyseiselle välille esiintymiä. Tämän lähestymistavan vahvuus on se, että kvantisointiarvojen alkuperäistä alarajaa vastaava huippu on suhteellisen helppo havaita. Huipun paikantamiseksi on siis yleensä vain haettava histogrammin toiseksi korkein huippu. Jos histogrammissa on vain yksi huippu, merkitsee se sitä, että joko kyseistä signaalia ei ole lainkaan kvantisoitu tai sitten kaikki arvot ovat nollia. Tietenkin olisi mahdollista tarkastella kaikkia arvoja, jotka eivät noudata kvantisointiarvon mukaista jakaumaa. Näyttäisi kuitenkin siltä, että tällöin virheiden määrä kasvaa. Kuviossa 8 näkyy kaksi histogrammia, jotka on muodostettu kuvan diskreetin kosinimuunnoksen painokertoimista. Histogrammit ovat muodostettu samasta kuvasta, mutta vasemman puoleisessa histogrammissa on käytetty itseisarvoja, kun taas oikean puoleisessa myös negatiiviset arvot ovat mukana. Vasemmassa kuvassa on merkitty punaisella kohta, joka vastaa painokertoimien arvojen alarajaa. Kyseisen kohdan ja nollan välillä on harmaalla merkitty arvot, jotka ovat poikkeuksellisen pieniä. Oikean puoleisessa kuviossa punaisella on merkitty kohdat, jotka noudattavat arvioidun kvantisointiarvon (10) mukaista jakaumaa.



Kuvio 8: Kuvan diskreetin kosinimuunnoksen painokertoimista muodostettu histogrammi. Vasemmalla puolella on käytetty itseisarvoja, oikealla myös negatiiviset arvot ovat mukana. Vasemmassa kuvassa on merkitty punaisella kohta, joka vastaa painokertoimien arvojen alarajaa, oikealla puolella punaisella on merkitty kaikki pylväät, jotka noudattavat arvioidun kvantisointiarvon (10) mukaista jakaumaa .

Kuten aiemmin totesin, en ole voinut hyödyntää JPEG-tiedostoista suoraan saatavia painokertoimia, joten olen käyttänyt sen sijaan uuden kosinimuunnoksen avulla laskettuja arvoja. Heikentyneen suoritusajan lisäksi tässä menettelyssä on se ongelma, että siitä ai-

heutuu lievää epätarkkuutta. Mahdollisten pyöristys- tai muiden laskuvirheiden johdosta kaikki arvot eivät osu aivan oikeille kohdilleen, vaan joitakin haja-arvoja osuu viereisiin arvoihin. Toisin sanoen arvot eivät noudata aivan täsmällisesti kvantisoinnin määrittelemää jakaumaa. Virheet ovat melko pieniä, joten tämä ei vaikuttaisi olevan erityisen merkittävä ongelma, joskin se on otettava huomioon. Käytännössä olen huomioinut asian siten, että poikkeavan kertoimen painoarvo riippuu sen etäisyydestä oletettuun, eli kvantisoinnin määrittelemää jaksollisuutta noudattavaan arvoon. Jos merkitään käytettyä kvantisointiarvoa, eli painokertoimien alarajaa *q*:lla ja tarkasteltavaa painokerrointa *c*:llä, saadaan kyseistä painokerrointa vastaava painoarvo *w* seuraavasti:

$$w = \begin{cases} 0, & kun\ c \geq q, \\ Min(q - c, c), & muutoin \end{cases}.$$

Painoarvo on siis sitä suurempi, mitä kauempana *c*:n arvo on *q*:sta tai nollasta. Kyseinen arvo lasketaan jokaiselle kertoimelle yhden 8 x 8 kokoisen JPEG-lohkon osalta. Lohkon saama kokonaisarvo on siis sen kaikkien kertoimien painoarvojen summa. Kyseinen summa lasketaan luonnollisesti kaikille lohkoille.

Kuvassa 9 on kuvan 6 väärennös tunnistettu diskreetin kosinimuunnoksen painokertoimien poikkeuksellisen matalia arvoja havaitsemalla. Hen ym. (2006) menetelmään verrattuna muokattu alue ei näy tuloskuvassa aivan niin yhtenäisenä, mutta toisaalta kuvan normaalit yksityiskohdat eivät myöskään pahemmin tule esiin.

Kuva 9: Kuvan 6 väärennös tunnistettu kuvan diskreetin kosinimuunnoksen painokertoimien poikkeuksellisen matalia arvoja havaitsemalla.

4.4 JPEG-pakkauksen aiheuttaman laatikoitumisen tunnistaminen pikselitasolla

Kolmannen toteutettavan menetelmän ovat kehittäneet Li ym. (2009). Menetelmän hyödyntää JPEG-pakkauksen kuvaan aiheuttamaa laatikoitumista.

4.4.1 Menetelmän kuvaus

Kuten todettu, JPEG-pakkauksessa käytetään 8 x 8 pikselin lohkojakoa. Diskreetti kosinimuunnos suoritetaan erikseen kullekin lohkolle. Tämä aiheuttaa lohkojen välille eräänlaista epäjatkuvuutta, joka on havaittavissa heikkoina pysty- sekä vaakasuuntaisina katkoksina lohkojen rajoilla. Ilmiön säännöllisyys tarjoaa hyvän lähtökohdan kuvien aitouden arvioimiseen. Mikäli lohkot eivät jollakin kuvan alueella noudata normaalia kahdeksan pikselin jaksollisuutta tai lohkoja ei havaita lainkaan, on kuvan aitoutta syytä epäillä.

4.4.2 Toteutuksen vaiheet

Menetelmän ensimmäisenä vaiheena on tunnistaa JPEG-lohkojen muodostama kehikko. Englanninkielisessä kirjallisuudessa kehikosta käytetään usein termiä *BAG* (block artifact grid, mm. Li ym. 2009). Tässä yhteydessä käytän kuitenkin yksinkertaisesti ter-

miä JPEG-kehikko. Ongelmana on tietenkin JPEG-lohkojen ja muiden kuvassa esiintyvien rajojen erottaminen toisistaan. JPEG-lohkoista tiedetään kuitenkin se, että niiden rajat noudattavat kahdeksan pikselin jaksollisuutta. Kehikon erottaminen suoritetaan erikseen vaaka- ja pystysuunnassa. Lopulta erotetut kehikot yhdistetään. Vaakasuuntaiset rajat erotetaan laskemalla vierekkäisten pikseleiden väliset erot seuraavalla kaavalla:

$$d(x, y) = |2\mathrm{f}(x, y) - f(x, y-1) - f(x, y+1)| \quad .$$

Yhtälössä *f(x, y)* vastaa pikselin arvoa sijainnissa *x*, *y* ja *d(x, y)* tuloksena saatua pikseleiden arvojen eroa. Tässä yhteydessä lohkojakoon ei vielä kiinnitetä huomiota, joten *x* ja *y* ovat globaaleja koordinaatteja. Laskutoimitus suoritetaan siis kuvan kaikille pisteille. Edellisestä kaavasta voidaan kuitenkin päätellä, että pois joudutaan tietenkin jättämään pisteet, jotka sijaitsevat kuvan reunoilla. Esimerkiksi *y*:n arvolla 0 piste (x, *y* – 1) sijaitsee kuvan rajojen ulkopuolella, joten sillä ei ole määriteltyä arvoa. Yhtälön tuloksena saadaan kokonaisluku, joka ei välttämättä mahdu arvoalueella 0-255. Tämän vuoksi seuraavien vaiheiden arvoaluekin saattaa olla laajempi. Tulos voidaan näin ollen muuntaa vasta lopuksi alueelle 0-255.

Tietyn arvon ylittävät erot jätetään huomiotta, sillä JPEG-lohkojen rajat ovat tavallisesti melko heikkoja. Lin ym. ehdottavat kynnysarvoa 50, joskaan tämä ei ole ehdoton raja. JPEG-lohkojen rajat ovat kuitenkin yleensä huomattavasti tätä heikompia.

Seuraava vaihe on, että muodostetun kuvan jokaisen pisteen arvoon lisätään viereisten 16 pisteen arvot vasemmalta ja oikealta puolelta. Luku 16 vastaa kahden naapurilohkon leveyttä. Tämän toimenpiteen tarkoituksena on vahvistaa heikkoja vaakaviivoja. Lisäys tapahtuu seuraavan yhtälön mukaisesti:

$$e_s(x, y) = \sum_{i=x-16}^{16} d(i, y) \quad .$$

Seuraavaksi kaikkien pisteiden arvoja tasoitetaan vähentämällä niistä paikallinen mediaani. Tämä voidaan määritellä seuraavan yhtälön avulla:

$$e(x, y) = e_s(x, y) - Mid[\{e_s(x, i) | y-16 \le i \le y+16\}] \quad ,$$

jossa merkinnällä *Mid[A{}]* tarkoitetaan joukon *A* mediaania. Mediaani lasketaan jälleen kahden naapurilohkon alueelta molemmin puolin (2 x 16 + 1 pistettä). Alueen leveys ei tässäkään tapauksessa ole ehdoton, mutta tärkeintä on käyttää arvoa, joka on jaollinen lohkon leveydellä. Jos alue on liian suuri, pienet muokatut alueet saattavat jäädä huomaamatta. Pienellä arvolla taas kuvan rajojen vaikutus voi olla liian suuri.

Jotta JPEG-lohkojen rajoja voitaisiin vahvistaa suhteessa muihin rajoihin, suoritetaan vielä yksi mediaanisuodatus seuraavasti:

$$g_h(x,y) = Mid\left[\left\{e(x,i) \middle| i = y-16, y-8, y, y+8, y+16\right\}\right] \quad .$$

Lopputuloksena saadaan kuva g_h (horizontal), joka sisältää JPEG-kehikon vaakasuuntaiset rajat. Käytetty algoritmi muodostaa rajat kunkin lohkon ylä- ja alareunaan, eli ensimmäiselle (y = 0) ja viimeiselle riville (y = 7). Normaalitilanteessa rajoja ei siis pitäisi muodostua muualle (y = 1, … ,6).

Pystysuuntaiset rajat sisältävä kuva g_v (vertical) saadaan muodostettua täysin vastaavalla tavalla kuin vaakasuuntainen kuva. Lopulta koko kehikon sisältävä kuva saadaan yhdistämällä molemmat kuvat:

$$g(x,y) = g_h(x,y) + g_v(x,y) \quad .$$

4.4.3 Päätelmien teko JPEG-kehikon perusteella

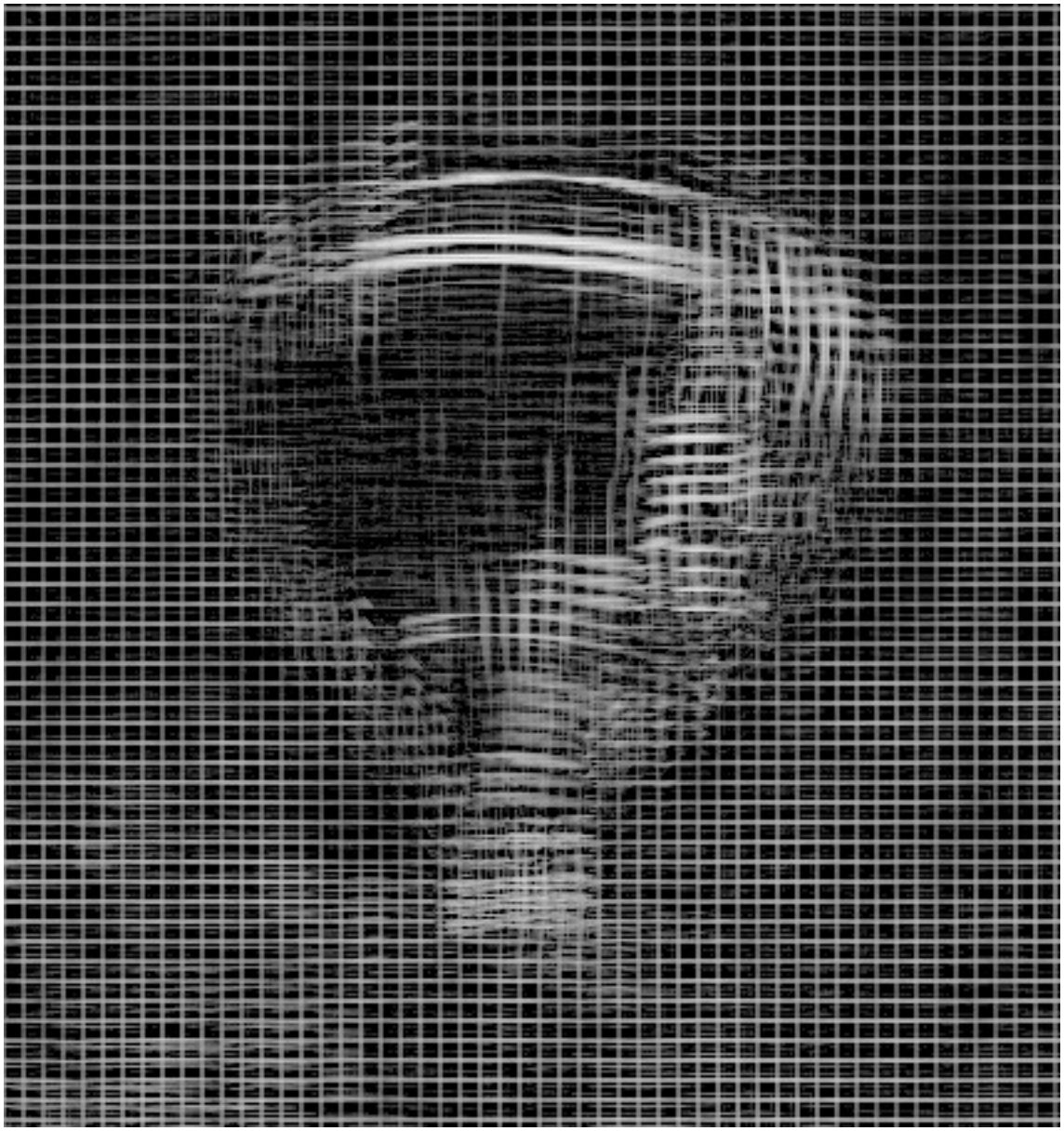
Kuvasta erotettua JPEG-kehikkoa voidaan käyttää monenlaisten päätelmien tekemiseen. Ensinnäkin koko kehikko tai sen osa saattaa olla väärin kohdistettu. Kehikon tulisi normaalisti alkaa vasemmasta yläkulmasta ja noudattaa kahdeksan pikselin jaksollisuutta. Jos reunat sijaitsevat koko kuvan alueella väärissä kohdissa, voidaan päätellä, että kuvaa on rajattu. Jos taas ilmiö on paikallinen, on luultavaa, että kyseinen kohta on kopioitu joko toisesta kuvasta, tai muualta samasta kuvasta. (Li ym. 2009)

Pakattuja kuvia yhdisteltäessä on tietenkin mahdollista, että kuvaan lisättävä alue kohdistetaan siten, että sen JPEG-kehikko on linjassa muun kuvan kanssa. Tällöin mitään poikkeavaa ei luonnollisesti havaita. Kehikkojen kohdistaminen rajoittaa kuitenkin huomattavasti kuvan muokkaajan toimintamahdollisuuksia. Lisättävää aluetta ei esimerkiksi voida pienentää, suurentaa, tai kallistaa ilman, että kehikko tuhoutuu. Tällöin on varsin

todennäköistä, että muokatun kuvan visuaalinen ilme kärsii, eikä lopputulos ole välttämättä uskottava.

Yleisesti ottaen voidaan todeta, että mitä voimakkaammin kuva on pakattu, sitä selvemmin JPEG-kehikko on erotettavissa. Kuitenkin vaikka kuva olisi täysin pakkaamaton, saadaan tuloksena jonkinlainen kehikko, joskin varsin epäselvä. Tämä johtuu siitä, että luonnollisissa valokuvissa on aina hyvin paljon yksityiskohtia. Jos erotettu kehikko on hyvin epäselvä, on luultavaa, että kyseessä on joko pakkaamaton tai hyvin lievästi pakattu kuva tai sen alue. Jälleen on loogista päätellä, että mikäli erotettu kehikko on epäselvä koko kuvan alueelta, kyseessä on pakkaamaton kuva. Jos taas kehikko on epäselvä vain tietyllä alueelta, kyseinen alue on luultavasti kopioitu esimerkiksi pakkaamattomasta kuvasta. Kehikon puuttuminen voi merkitä myös sitä, että kuvaa on esimerkiksi pienennetty, suurennettu tai kierretty. Tällainen on varmasti hyvin tyypillistä esimerkiksi tapauksissa, joissa jokin alue liitetään toisesta kuvasta.

Kuvassa 10 näkyy JPEG-kehikko, joka on erotettu kuvan 6 väärennetyltä alueelta Lin ym. (2009) menetelmällä. Kuumailmapallon kohdalla kehikko on varsin sotkuinen, sillä kyseinen alue ei sisällä pakkausjälkiä.



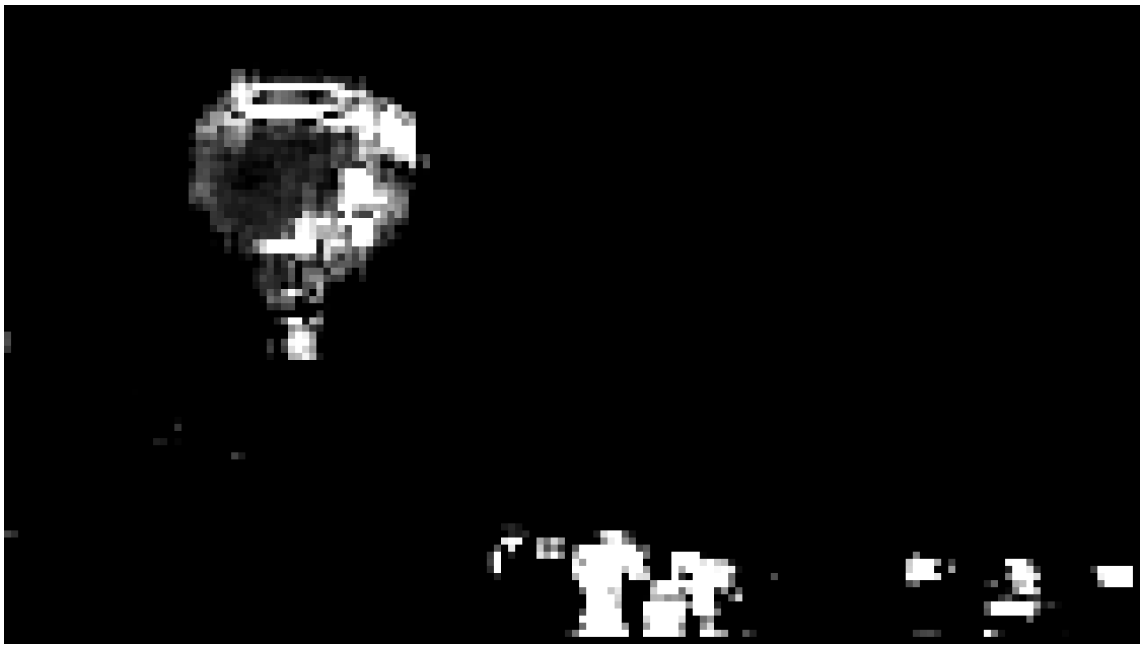
Kuva 10: Kuvan 6 väärennetyn alueen JPEG-kehikko tunnistettu Lin ym. (2009) menetelmällä.

Toisinaan JPEG-kehikon silmämääräinen arviointi saattaa olla hieman hankalaa. Tästä syystä Li ym. ehdottavat automaattista merkkauskäytäntöä muokattujen alueiden yksilöimiseksi. Lohkon alueelta jokaiselle riville lasketaan summa kyseisen rivin pikseleiden arvoista. Sama suoritetaan sarakkeille. Sen jälkeen valitaan pienin reunasarakkeen (x = 0 tai 7) ja reunarivin (y = 0 tai 7) arvo, ja näistä vähennetään suurin keskirivin (y = 1, …, 6) ja suurin keskisarakkeen (x = 1, ..., 6) arvo. Täsmällisemmin ilmaistuna menettely on seuraava:

$$b = Max\left\{\sum_{i=1}^{6} a(x,i) | 1 \le x \le 6\right\} - Min\left\{\sum_{i=1}^{6} a(x,i) | x = 0,7\right\}$$
$$+ Max\left\{\sum_{i=1}^{6} a(i,y) | 1 \le y \le 6\right\} - Min\left\{\sum_{i=1}^{6} a(i,y) | y = 0,7\right\} .$$

Kukin lohko väritetään näin saavutetulla arvolla *b*. Tuloksena saatavassa kuvassa valkoisella merkityt alueet ovat todennäköisesti muokattuja ja mustat alueet aitoja. Jos koko JPEG-kehikko on väärässä kohdassa, saadaan tuloksena lähes täysin valkoinen kuva. Haluttaessa kehikon kohdistus voidaan kuitenkin ottaa huomioon merkkausoperaation yhteydessä. Tällöin on laskettava kehikon poikkeama normaalista sijainnista.

Kuvassa 11 näkyy lopullinen tuloskuva, joka on saatu edellisessä vaiheessa erotetun kehikon perusteella. Muokattu alue löytyy aivan oikein, mutta kuvan rakennusten kohdalla esiintyy melko suuria virhetunnistusalueita. Kuvan sisällön vaikutus on kyseessä olevan menetelmän kohdalla kohtuullisen suuri. Käytännössä täysin puhdasta JPEG-kehikkoa ei saada erotettua kuin kaikkein yksinkertaisimmilta kuvan alueilta. Tällaisia ovat esimerkiksi alueet jotka sisältävät pilvetöntä taivasta. Hyvin monimutkaisilla alueilla kehikon erottaminen on epävarmempaa. Kahdeksan pikselin jaksollisuutta noudattavat kuvan yksityiskohdat aiheuttavat virheellisesti tunnistettuja rajoja.



Kuva 11: Lopullinen tuloskuva, joka on saatu tunnistamalla kuvan 6 väärennös Lin ym. (2009) menetelmällä.

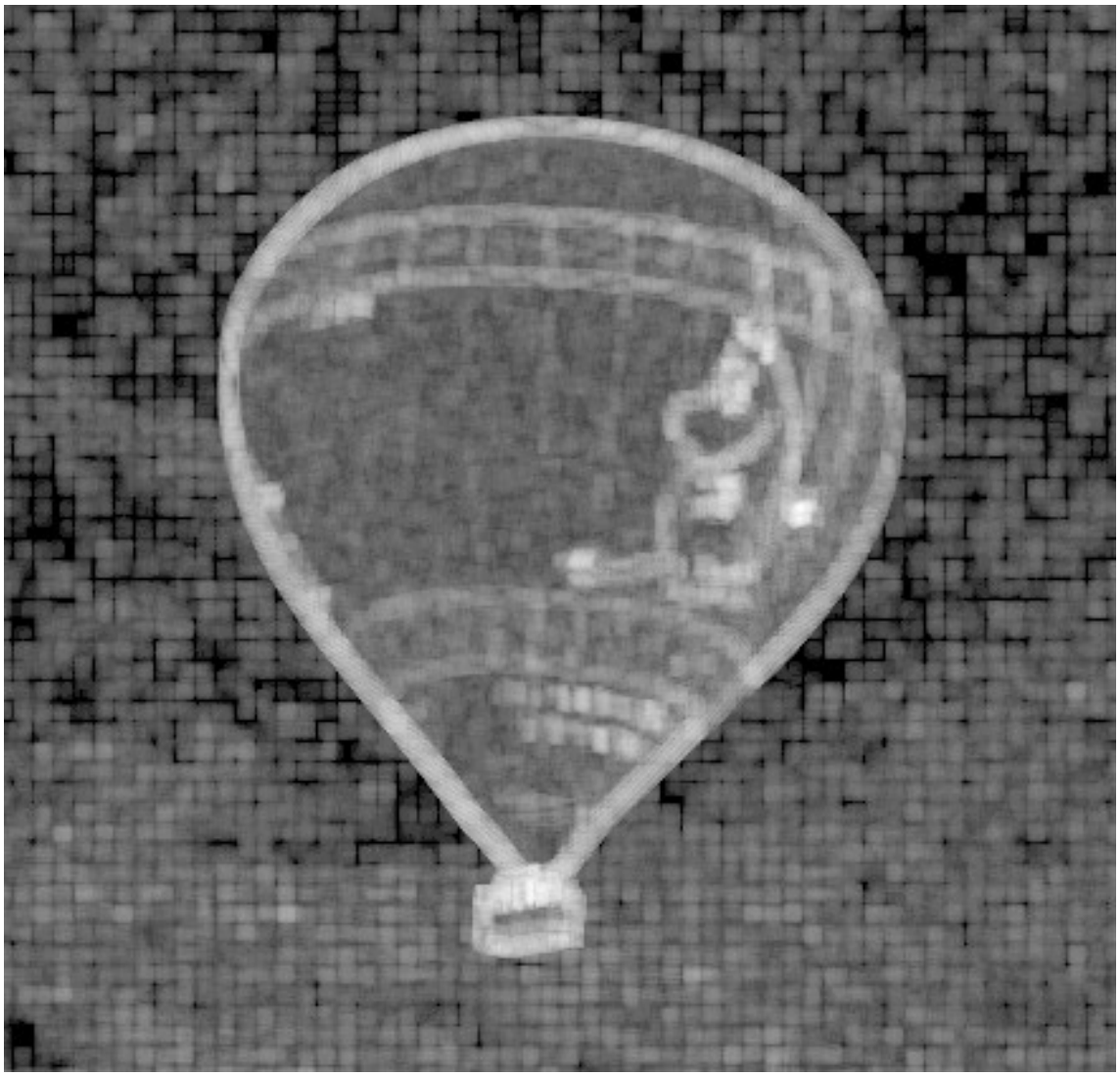
4.5 JPEG-pakkauksen aiheuttaman laatikoitumisen tunnistaminen kuvan diskreetin kosinimuunnoksen painokertoimia arvioimalla

Myös neljännen toteutettavan menetelmän toiminta perustuu JPEG-kehikon tunnistamiseen. Tässäkin tapauksessa menetelmän ovat esitelleet Li ym. (2008).

4.5.1 Menetelmän kuvaus

Nyt esiteltävän menetelmän tapauksessa tarkastelu kohdistuu pikseleiden sijaan kuvan diskreetin kosinimuunnoksen painokertoimiin. Menetelmä pohjaa siihen havaintoon, että kvantisoinnin seurauksena kunkin JPEG-lohkon korkeimpia taajuuksia vastaavien painokertoimien arvot ovat tyypillisesti nollia. Tämä pätee kuitenkin vain siinä tapauksessa, että kosinimuunnoksessa käytetään alkuperäistä 8 x 8 pikselin lohkojakoa. Sen sijaan, jos alkuperäisen lohkojaon reunoja osuu uusien lohkojen sisälle, saavat korkeimpienkin taajuuksien painokertoimet nollasta poikkeavia arvoja. Arvio JPEG-kehikosta saadaan siten, että kuvan jokaisen pikselin kohdalla lasketaan 8 x 8 kokoinen kosinimuunnoslohko. Oletuksena siis on, että korkeimpien taajuuksien kertoimet ovat nollia. Karkeasti ottaen menetelmän ajatuksena on laskea kunkin pikselin kohdalla poikkeama tästä oletuksesta. Menetelmän kehittäjät laskevat korkeiksi taajuuksiksi kaikki ne, jotka sijaitsevat kosinimuunnoslohkon oikeassa tai alareunassa. Tällöin siis kyseistä painokerrointa vastaavassa sijainnissa ainakin *x*- tai *y*-koordinaatti on 7. Kun lohkojen rajat ovat kohdakkain, poikkeaman tulisi olla kaikkein pienin. Reuna-alueilla poikkeama taas kasvaa. Yhden pisteen saama kokonaisarvo muodostuu siis yhdistämällä jokaisen korkean taajuuden poikkeamat. Kehikosta voidaan muodostaa kuva skaalaamalla saadut poikkeamien arvot välille 0-255. Alkuperäisessä toteutuksessa poikkeamien arvot lasketaan hieman monimutkaisemmalla tavalla. Siinä oletetuista painokertoimien arvoista lasketaan käänteisen kosinimuunnoksen avulla pikseliarvot, joita verrataan todellisiin pikseleiden arvoihin. Omassa toteutuksessani tarkastelu tapahtuu siis suoraan painokertoimien tasolla, joten käänteistä kosinimuunnosta ei tarvita. Käyttämäni lievästi yksinkertaistettu tapa näyttäisi kuitenkin toimivan varsin hyvin. Alkuperäisessä tutkimuksessa esitettyjen JPEG-kehikoiden selkeys ei esimerkiksi vaikuttaisi olevan sen parempi kuin omalla toteutuksellani aikaansaatavien.

Kuvassa 12 näkyy Lin ym. (2008) menetelmällä tunnistettu JPEG-kehikko kuvan 6 väärennetyltä alueelta. Laatikoitumisen puuttuminen on helposti havaittavissa kuumailmapallon alueelta.

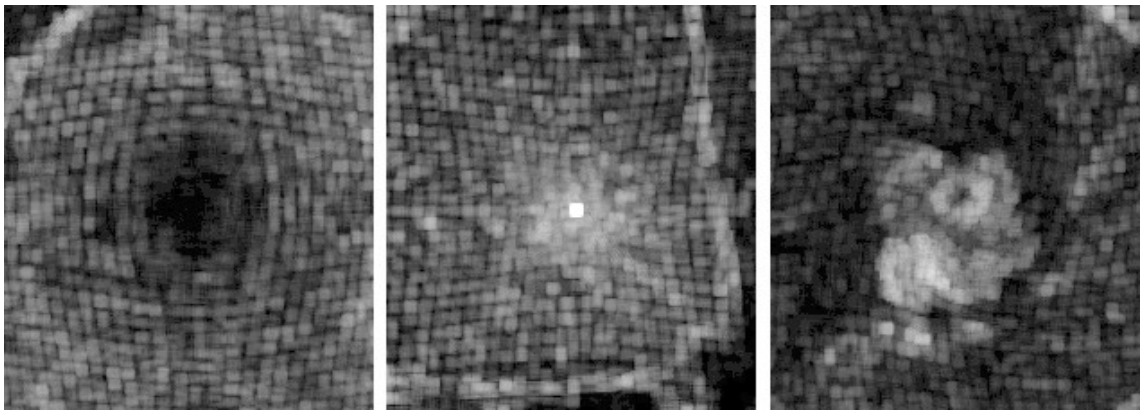


Kuva 13: Kuvan 6 väärennetyn alueen JPEG-kehikko tunnistettu Lin ym. (2008) menetelmän avulla.

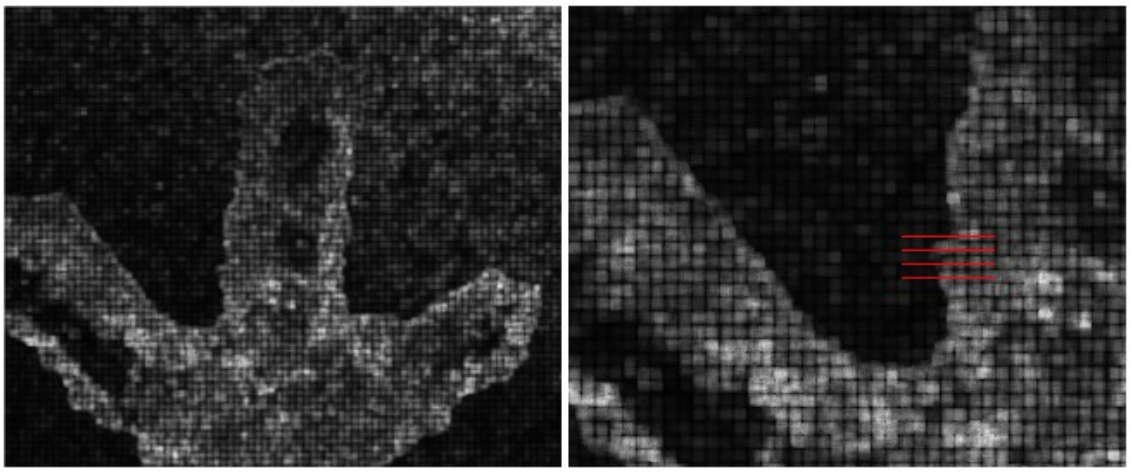
Muodostuneen kehikon perusteella voidaan tehdä päätelmiä samaan tapaan kuin edellisen menetelmän yhteydessä. Tällä menetelmällä muodostettu kehikko voi kuitenkin tarjota myös joitakin viitteitä käytetyn muokkauksen tyypistä. Kuvassa 14 näkyy kolme esimerkkiä geometrisista vääristymistä. Vasemmalla kuvan aluetta on venytetty ulospäin, keskellä sisäänpäin ja oikealla on käytetty eräänlaista pyörre-tehostetta.

Kuvassa 15 on esimerkki JPEG-kehikosta, jossa voidaan havaita väärin kohdistettuja lohkoja. Kuvan oikean puoleisessa suurennoksessa on ilmiön havainnollistamiseksi merkitty punaisin viivoin alkuperäisten lohkojen vaakasuuntaisia rajoja. Vaaleammalla alueella, joka kuuluu kuvan muokattuun osaan, lohkojen kohdistus ei selvästikään osu viivojen kanssa kohdalleen.

Edellä kuvailtujen ilmiöiden havaitseminen on tietenkin sitä vaikeampaa, mitä voimakkaampaa pakkausta muokkauksen jälkeen käytetään. Esimerkkikuvissa muokkauksen jälkeinen tallennusmuoto on häviötön PNG.



Kuva 14: Kolme esimerkkiä geometrisista vääristymistä, joita voidaan havaita Linin ym. (2008) menetelmällä.



Kuva 15: Väärin kohdistettu JPEG-kehikko. Oikealla puolella on punaisin viivoin merkitty kuvan alkuperäisten lohkojen vaakasuuntaisia rajoja.

Menetelmän kehittäjät eivät kuitenkaan ole esittäneet tapaa, jolla epäilyttävät alueet voitaisiin paikantaa automaattisesti. Toisinaan kehikon tulkinta saattaa olla hieman hankalaa, ja vaatia kohtuullisen yksityiskohtaista tarkastelua. Edellisiin verrattuna kyseessä oleva menetelmä on myös laskennallisesti huomattavasti raskaampi. Tekijät mainitsevatkin jatkotavoitteikseen laskennan määrän vähentämisen, automaattisen tulosten tulkinnan toteutuksen ja saavutettavan kehikon tarkkuuden parantamisen. Esitän seuraavaksi mahdollisen ratkaisuvaihtoehdon ensin mainittuun tavoitteeseen.

4.5.2 JPEG-kehikon osittainen tunnistaminen

Lin ym. (2008) menetelmän raskaus johtuu siitä, että 8 x 8 pikselin kokoinen diskreetti kosinimuunnos lasketaan lähes jokaisen kuvan pikselin alueelta. Itse asiassa kuvan oikeasta ja alareunasta jätetään pois seitsemän pikseliä, koska näissä kohdissa kokonaista 8 x 8 pikselin lohkoa ei saada muodostettua. Yleensä suurin osa kuva-alasta ei kuitenkaan ole erityisen kiinnostavaa. Olisi siis varsin käytännöllistä, mikäli epäilyttävät alueet voitaisiin alustavasti paikantaa jollakin tavalla, ja jättää muut alueet kokonaan tarkastelun ulkopuolelle.

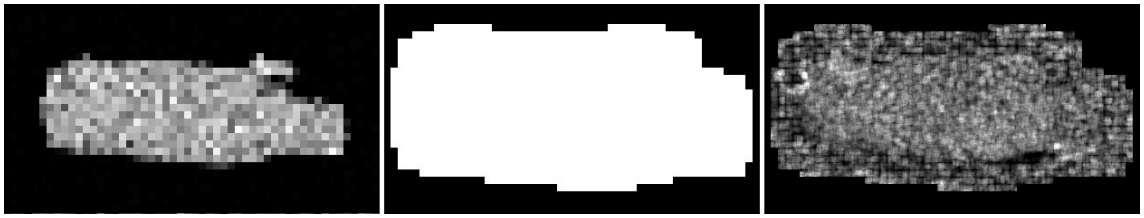
Yksi vaihtoehto on käyttää muita edellä esiteltyjä menetelmiä epäilyttävien alueiden tunnistamiseen. Näin ollen raskas JPEG-kehikon tunnistaminen rajataan vain kriittisille alueille. Epäilyttävien alueiden valinta jää siis käytännössä muiden menetelmien vastuulle, mutta JPEG-kehikon avulla voidaan näin mahdollisesti saada lisävalaistusta esimerkiksi muokkaustavasta. Ratkaisu on siinä mielessä melko luonteva, että menetelmien lähtökohtaiset oletukset mm. pakkauslaatujen suhteen ovat joka tapauksessa useimmiten samat. Toisin sanoen jos muokattu alue on havaittavissa JPEG-kehikon poikkeamien perusteella, se on yleensä havaittavissa myös kuvan kosinimuunnoksen painokertoimia tarkastelemalla. Jälkimmäisellä tavalla tunnistus tapahtuu vain nopeammin.

Esimerkkinä toteutin yhdistelmän, jossa muokatut alueet tunnistetaan ensin kuvan kosinimuunnoksen painokertoimien matalia arvoja tunnistamalla. Näin saatavan tuloskuvan perusteella voidaan valita ne alueet, joilta JPEG-kehikko halutaan erottaa. Jotta alueet voidaan määritellä täsmällisesti, täytyy tuloskuva muuttaa jonkin kynnysarvon perusteella kaksiväriseksi. Kynnysarvo voidaan määritellä esimerkiksi siten, että lohkon arvon tulee olla vähintään 2,5 kertaa kuvan kirkkauden keskiarvo, jotta se valitaan mu-

kaan. Kyseinen kynnysarvo on valittu kokeellisesti ja se voi hyvin olla jokin muukin. Tällä tavoin saavutetaan eräänlainen kartta, joka kertoo sen, mitä alueita alkuperäisestä kuvasta otetaan mukaan tarkasteluun ja mitkä alueet voidaan ohittaa. JPEG-kehikkoa muodostettaessa, voi olla kuitenkin syytä laajentaa tarkastelualaa hieman epäilyttävien lohkojen ulkopuolelle. Toteutuksessani laajensin tarkastelualaa neljän lohkon verran, eli 4 x 8 pikseliä kaikkiin suuntiin. Tällöin JPEG-kehikon poikkeamia voidaan verrata paremmin ympäröiviin alueisiin. Tarkasteltavaa aluetta rajaamalla menetelmän vaatima laskenta-aika putoaa yleensä huomattavasti. Kohdistamalla tarkastelu vain epäilyttäviin alueisiin saadaan itseasiassa myös helpotettua tulosten tulkintaa.

Kuvassa 16 on jälleen esimerkki aidosta ja väärennetystä valokuvasta. Kuvassa 17 näkyy kaikki välitulokset, jotka on saavutettu edellä kuvatulla yhdistelmätekniikalla. Vasemmanpuoleisin tulos on saavutettu kosinimuunnoksen painokertoimien matalia arvoja tunnistamalla. Keskellä sama tulos on laajennettu ja muutettu kaksiväriseksi. Oikealla kyseiseltä alueelta on erotettu JPEG-kehikko Linin ym. (2008) menetelmällä. Pelkkä painokertoimien matalien arvojen tunnistus kyseessä olevaan kuvaan kestää noin sekunnin. Esittämälläni yhdistelmätekniikalla aikaa kuluu noin neljä sekuntia. Sen sijaan jos koko kuvan alueelta tunnistetaan JPEG-kehikko, aikaa kuluu noin 35 sekuntia. Tähän suhteutettuna nopeutus on merkittävä. On kuitenkin huomattava, että saavutettava nopeutus riippuu oleellisesti muokatun alueen ja mahdollisten virheellisesti tunnistettujen alueiden koosta.

Kuva 16: Alkuperäinen ja väärennetty kuva. Oikealla puolella pullo on poistettu kuvasta monistamalla taustaa.



Kuva 17: Kuvan 16 väärennös tunnistettu menetelmällä, joka yhdistää kosinimuunnoksen matalien painokertoimien arvojen havaitsemisen ja JPEG-kehikon erottamisen. Vasemmanpuoleisin kuva on ensin mainitun menetelmän tulos. Keskellä sama tulos on laajennettu ja muutettu kaksiväriseksi. Oikealla kyseiseltä alueelta on tunnistettu JPEG-kehikko. Kuvat on lähennetty muokattuun alueeseen.

4.6 Menetelmien toimintaedellytyksistä

Kuten aiemmin todettiin, kaikkien käsiteltyjen autentikointimenetelmien lähtökohtainen toimintaedellytys on se, että tarkasteltava kuva on ennen mahdollisia muokkaustoimenpiteitä ollut JPEG-muotoinen. Tarkemmin määriteltynä toiminnan edellytykset liittyvät 8 x 8 pikselin lohkoissa tapahtuvaan diskreettiin kosinimuunnokseen ja sen perusteella saatavien painokertoimien kvantisointiin. Kuten todettua, diskreetti kosinimuunnos sellaisenaan ei muuta kuvainformaatiota, vaan tämä on vasta kvantisoinnin seurausta. Näin ollen siis häviötön JPEG-muoto ei voi tulla kyseeseen. Näyttäisi myös siltä, että kaikkein korkeimmilla pakkauslaaduilla (99-100) kuvaan ei välttämättä muodostu niin selkeitä pakkausjälkiä, että autentikointimenetelmät voisivat niitä hyödyntää. Muokkausta seuraavan tallennusmuodon ei välttämättä tarvitse olla JPEG, vaan jokin häviötön muoto käy yhtä hyvin.

On suhteellisen selvää, että muokkaustoimenpiteiden yhteydessä JPEG-kuvia käsitellään tyypillisesti raakana pikseli-informaationa. Tämä johtuu siitä, että taajuusinformaation manipulointi ei vaikuta yksittäisiin pikseleihin. Toisin sanoen, jotta JPEG-kuvia voitaisiin käsitellä pikselitasolla, täytyy taajuusinformaatio ensin muuntaa pikseli-informaatioksi. Alkuperäisen pakkauksen jäljet kuitenkin säilyvät kuvassa myös pikselitasolla. Näin ollen muokkausta seuraavan kosinimuunnoksen yhteydessä saatavien painokertoimien arvot muodostuvat jo kertaalleen kvantisoidun kuvainformaation perusteella. Kuvan alkuperäisillä alueilla muodostuneet painokertoimet vastaavat siis ensimmäisen pakkauksen yhteydessä saavutettuja arvoja.

JPEG-pakkausta hyödyntävien autentikointimenetelmien hyvä puoli on se, että ne eivät aseta vaatimuksia käsiteltävän kuvan alkuperälle. Tämä johtuu siitä, että aitouden arviointiin käytettävä informaatio muodostuu nimenomaan JPEG-pakkauksen ansiosta. Sen sijaan esimerkiksi kuvan ottamiseen käytetyn kameran ominaisuudet eivät ole merkityksellisiä. Kuva voi ainakin periaatteellisella tasolla olla yhtä hyvin digitaalisella kuin perinteisellä filmikamerallakin otettu. Jälkimmäisessä tapauksessa kuva olisi tietenkin digitoitava ja tallennettava JPEG-muotoon. Mahdolliset käsittelytoimet, jotka tapahtuvat ennen tallennusta JPEG-muotoon, eivät tietenkään tällöin tulisi havaituiksi.

Koska menetelmät hyödyntävät JPEG-pakkauksen aiheuttamia jälkiä, on tietenkin tärkeää, että tällaiset jäljet säilyvät myös kuvan viimeisimmän tallennuksen yhteydessä. Oleellista on etenkin alkuperäisen 8 x 8 pikselin lohkojaon säilyminen. Lohkojaon tuhoaminen on kuitenkin valitettavan helppoa, sillä siihen riittää esimerkiksi kuvan kallistaminen tai koon muutos. Kosinimuunnoksen painokertoimia hyödyntävät menetelmät edellyttävät myös, että lohkojen kohdistus on täsmälleen oikea. Toisin sanoen lohkojaon tulee alkaa tarkalleen kuvan vasemmasta ylänurkasta. JPEG-kehikon tunnistamiseen perustuvilla menetelmillä sen sijaan tällaista edellytystä ei ole. Kyseiset menetelmät olettavat ainoastaan, että kehikko noudattaa 8 x 8 pikselin jaksollisuutta.

JPEG-kehikon tunnistamiseen perustuvien menetelmien ongelma puolestaan on se, että ne toimivat huonosti, ellei kuvan viimeisin tallennuslaatu ole suhteellisen korkea. Kuten Li ym. (2008) toteavat, muokkauksen jälkeinen JPEG-pakkaus vaikuttaa kehikon tunnistamiseen kahdella tavalla. Ensinnäkin viimeisin JPEG-pakkaus luo kuvaan oman kehikkonsa. Näin ollen on vaikeampaa erottaa kuvaan lisättyjä alueita, jotka eivät alunperin sisällä laatikoitumisjälkiä. Toiseksi voimakas pakkaus kadottaa kuvan yksityiskohtia. Tämä puolestaan tarkoittaa sitä, että normaalista 8 x 8 pikselin jaksollisuudesta poikkeavat rajat heikkenevät. Näin on etenkin silloin jos aiemmat rajat ovat suhteellisen heikkoja. Toisin sanoen tällöin on vaikeampaa havaita tilanteita, joissa kuvaan liitetään alueita, jotka sisältävät valmiiksi laatikoitumisjälkiä. Kaikkein optimaalisin tilanne on sellainen, jossa viimeisin tallennusmuoto on häviötön.

5 TESTITULOKSET

Tässä luvussa esittelen testitulokset, jotka on saatu testaamalla edellisessä luvussa esiteltyjä autentikointimenetelmiä muodostamani kuva-aineiston suhteen.

5.1 Testiaineiston muodostaminen

Jotta valokuvaväärennösten tunnistamiseen tarkoitetun menetelmän todellista toimivuutta voidaan arvioida, on suoritettava testejä, joita varten tarvitaan testiaineistoa. Testiaineiston kokoamiseksi on olemassa erilaisia vaihtoehtoja. Ensinnäkin internetin eri sivustoilta olisi mahdollista kerätä helposti suhteellisen suuri joukko muokattuja kuvia. Tällöin ongelmana olisi kuitenkin se, että kuvien käsittelyhistoriaa ei tunneta tarkasti. Tiedossa ei ole esimerkiksi se, kuinka monesti kuva on pakattu, mitä pakkauslaatuja missäkin vaiheessa on käytetty ja mitkä osat kuvasta tarkkaan ottaen ovat muokattuja.

Kyetäkseni kunnolla kontrolloimaan edellä mainittuja muuttujia, päädyin tuottamaan itse testiaineistoni. Tämän lähestymistavan ongelmana taas on ainakin se, että aineiston tuottaminen on melko hidasta.

Testiaineiston pohjana on käytetty 18 eri valokuvaa. Kukin kuva on ensimmäisessä vaiheessa tallennettu JPEG-muotoon, käyttäen neljää eri pakkaustasoa, jotka ovat 90, 80, 70 ja 60. Kuhunkin näistä kuvista on suoritettu jonkinlainen manipulaatio, minkä jälkeen kuva on tallennettu uudelleen käyttäen viittä eri tallennusmuotoa, jotka ovat häviötön PNG sekä JPEG tasoilla 90, 80, 70 ja 60. Uskoisin, että käytetty vaihteluväli kattaa kohtuullisen hyvin käytännössä kyseeseen tulevat tallennuslaadut. Tasolla 60 kuvan laatu heikkenee jo varsin huomattavasti. En myöskään usko, että tiheämpi, esimerkiksi viiden yksikön välien käyttö olisi tuonut merkittävää lisävalaistusta tuloksiin. Käytetyillä pakkauslaatukombinaatioilla varsinaisen aineiston kooksi muodostuu 360 kuvaa. Olen kuitenkin näiden lisäksi ottanut testiaineistoon muutamia yksittäisiä testitapauksia, joissa edellä kuvatusta menettelystä on poikettu. Olen tällä tavoin pyrkinyt tutkimaan joitakin erikoistapauksia. Käsittelen näitä tapauksia erikseen. Olen pyrkinyt valitsemaan testiaineistoon sisällöltään mahdollisimman erityyppisiä kuvia. Lisäksi olen pyrkinyt käyttämään mahdollisimman monia tyypillisimpiä muokkaustapoja. Muokkaustyyppien vaikutusta tuloksiin käsitellään kohdassa 5.4 ja kuvan sisällön vaikutusta kohdassa 5.5. Voidaan todeta, että molemmilla tekijöillä on selvästi vaikutusta tuloksiin, mutta vaiku-

tusten tarkempi erittely vaatisi huomattavasti kattavamman testiaineiston. Tässä yhteydessä asian käsittely jää väistämättä suhteellisen pintapuoliseksi.

Kuvien muokkaamiseen ja tallentamiseen on käytetty GIMP-kuvankäsittelyohjelmaa (versio 2.6.8). Valokuvien ottamiseen on käytetty Panasonicin Lumix DMC-TZ3 ja Lumix DMC-FX8 -kameroita. Kuvien alkuperäinen tallennusmuoto on JPEG ja resoluutio on tyypillisesti 2048 x 1563, 2560 x 1440, 3072 x 2304 tai 3328 x 1872 pikseliä. Ennen varsinaista käsittelyä kaikkien kuvien resoluutio on pienennetty 50 prosenttiin alkuperäisestä, jotta mahdolliset JPEG-pakkauksen aiheuttamat jäljet poistuisivat. Osaa kuvista on myös rajattu tai kallistettu. Lin ym. (2009) ovat käyttäneet hieman vastaavaa menettelyä testiaineiston kokoamisessa. Varsinaiseen testiaineistoon kuuluvat kuvat ovat nähtävillä pienennetyssä muodossa liitteessä 1. Käytettyjen muokkaustoimenpiteiden selitykset puolestaan löytyvät liitteestä 2.

5.2 Tulosten tulkinta

Koska toteutetut autentikointimenetelmät esittävät tuloksensa kuvien muodossa, on syytä pohtia hieman tulosten tulkintaa. Muokattujen alueiden olisi tarkoitus näkyä tuloskuvissa yhtenäisinä tummina tai vaaleina alueina. Periaatteellisena lähtökohtana tulosten tukinnassa tulisi olla se, että todellisessa tilanteessa tulosten tarkastelija ei tiedä sisältääkö kuva väärennettyjä alueita, ja jos sisältää, missä ne sijaitsevat. Tämän vuoksi tulosten tulisi olla riittävän selkeästi tulkittavia. Toisin sanoen väärennetyn alueen pitäisi erottua selvästi muista alueista. Tämä tarkoittaa myös sitä, että virheellisesti tunnistettuja alueita ei saisi juurikaan esiintyä. Käytännössä kuvan yksityiskohdat aiheuttavat kuitenkin jonkin verran satunnaisia virheitä ja häiriötä tunnistustuloksiin. Yleensä varsinaiset muokatut alueet ovat kuitenkin suhteellisen selkeästi erotettavissa tuloskuvista, sillä ne ovat tyypillisesti yhtenäisempiä kuin virhealueet. Toisinaan todellisia ja virheellisiä tunnistuksia on kuitenkin vaikea erottaa toisistaan. Jos muokattujen osien tunnistaminen ei ole täysin selvää, olen tulosten raportoinnissa tulkinnut tällaisen tapauksen epäonnistuneeksi. Esitän myöhemmin kuvaesimerkkejä tällaisista tapauksista.

Määrittelin tulosten tulkintaa varten viisi eri kategoriaa, jotka on lueteltu taulukossa 7. Ensimmäinen kategoria tarkoittaa sitä, että tulos ei ole lainkaan tulkittavissa. Toisin sanoen tuloskuvasta ei erotu mitään yksittäistä aluetta. Toiseen kategoriaan kuuluvissa tapauksissa muokattu alue on havaittavissa, mutta virheellisesti tunnistettujen alueiden

vuoksi se ei erotu riittävästi, jolloin virhetulkinnan vaara on olemassa. Kolmannessa kategoriassa muokatut alueet on kohtuullisesti tunnistettavissa, ilman merkittävää virhetulkinnan mahdollisuutta. Neljännessä ja viidennessä kategoriassa muokatut alueet erottuvat jo hyvin selkeästi, ja virhetulkinnan todennäköisyys on enää häviävän pieni.

Taulukko 7: Testiaineiston perusteella saatujen tulosten tulkinnassa käytettyjen koodien merkitykset.

Koodi	Merkitys
1	Ei tulkittavissa
2	Virhetulkinnan mahdollisuus
3	Kohtuullisesti tulkittavissa
4	Selkeästi tulkittavissa
5	Erittäin selkeästi tulkittavissa

5.3 Testitulokset

Taulukossa 8 on eritelty saavutetut tulokset kunkin testiaineiston kuvan osalta, käyttäen edellä kuvattua kategoriajakoa. Tarkastelu rajoittuu tässä kuviin, joiden alkuperäinen pakkauslaatu oli 70. Tulokset on esitetty kolmelle muokkauksen jälkeiselle tallennusmuodolle, jotka ovat JPEG laaduilla 80 ja 90 sekä häviötön PNG. Kaikkia kolmea laatua vastaavat tulokset sijaitsevat taulukon samassa solussa pilkuilla eroteltuna. Tulokset on eroteltu myös autentikointimenetelmän mukaan. Taulukon tulkinta saattaa ensisilmäyksellä tuntua hieman hankalalta, mutta siinä esitetyt tiedot kuvaavat kuitenkin suhteellisen yksityiskohtaisella tavalla saavutettuja tuloksia.

Taulukko 8: Eri autentikointimenetelmillä saavutetut tulokset testiaineiston kuvien mukaan jaoteltuna. Taulukon tulokset koskevat kuvia, joiden pakkauslaatu ennen muokkausta oli 70. Muokkauksen jälkeiset tallennusmuodot ovat JPEG laaduilla 80, 90 ja häviötön PNG. Tuloksen ilmaisevien koodien merkitykset on lueteltu taulukossa 7.

Kuva	**Autentikointimenetelmä ja tunnistuksen tulos pakkauslaaduilla 80, 90 ja häviötön tallennus.**		
	Kaksoisvantisoinnin tunnistus	**Matalien painokertoimien arvojen tunnistus**	**JPEG-kehikon tunnistus pikselitasolla**
1	4, 5, 5	5, 5, 5	1, 2, 3
2	2, 4, 5	1, 4, 5	1, 1, 1
3	3, 5, 5	5, 5, 5	1, 1, 2
4	2, 4, 5	4, 5, 5	1, 1, 5
5	2, 4, 4	2, 4, 4	1, 1, 1
6	2, 4, 4	4, 4, 4	2, 2, 2
7	3, 4, 5	1, 4, 5	1, 1, 1
8	1, 1, 1	1, 1, 1	1, 1, 1
9	3, 5, 5	3, 5, 5	1, 1, 3
10	1, 5, 5	1, 5, 5	1, 1, 1
11	1, 2, 2	1, 2, 2	1, 1, 1
12	2, 2, 2	2, 1, 1	1, 1, 1
13	4, 5, 5	3, 5, 5	1, 2, 4
14	1, 4, 4	1, 3, 3	1, 1, 1
15	2, 2, 2	2, 2, 2	1, 1, 1
16	1, 3, 3	2, 3, 3	1, 1, 1
17	1, 3, 3	1, 3, 3	1, 1, 1
18	3, 4, 4	2, 4, 4	1, 1, 1

Taulukossa 9 on esitetty toteutettujen autentikointimenetelmien yhdistetyt onnistumisprosentit pääasiallisen testiaineiston suhteen. Tulokset on eritelty ennen ja jälkeen muokkauksen käytettyjen pakkauslaatujen mukaan. Taulukoissa 10-12 on esitetty eri menetelmien yksilölliset onnistumisprosentit. Mukana tarkastelussa ovat kuvan kosinimuunnoksen painokertoimia hyödyntävät menetelmät sekä Lin ym. (2009) JPEG-kehikon tunnistamiseen pikselitasolla perustuva menetelmä. Lin ym. (2008) toista JPEG-kehikon tunnistamiseen perustuvaa menetelmää ei ole otettu vertailuun mukaan, koska sille ei ole toistaiseksi toteutettu tapaa tulosten automaattiseen tulkintaan. Kyseisen menetelmä toki soveltuu sellaisenaankin väärennösten tunnistamiseen, mutta toistaiseksi tu-

losten tulkinta on monissa tapauksissa turhan työlästä. Helposti tulkittavia tilanteita ovat ainakin sellaiset, joissa muokatun kuvan viimeisin tallennusmuoto on häviötön. Joka tapauksessa tässä tutkielmassa tarkastelen Lin ym. (2008) menetelmää lähinnä sen tarjoaman lisäinformaation näkökulmasta.

Taulukko 9: Toteutettujen autentikointimenetelmien yhdistetyt onnistumisprosentit testiaineiston suhteen. Vasemmanpuoleiseen sarakkeeseen on merkitty kuvan pakkauslaatu ennen muokkausta. Ylimmällä rivillä on kuvan pakkauslaatu muokkauksen jälkeen.

Q1 / Q2	60	70	80	90	Häviötön
60	0	11	78	78	78
70	0	0	44	78	78
80	0	0	0	78	78
90	0	0	0	0	83

Taulukko 10: Hen ym. (2006) kaksoiskvantisointiin perustuvan menetelmän onnistumisprosentit testiaineiston suhteen. Vasemmanpuoleiseen sarakkeeseen on merkitty kuvan pakkauslaatu ennen muokkausta. Ylimmällä rivillä on kuvan pakkauslaatu muokkauksen jälkeen.

Q1 / Q2	60	70	80	90	Häviötön
60	0	11	72	78	78
70	0	0	33	78	78
80	0	0	0	72	78
90	0	0	0	0	78

Taulukko 11: Kuvan diskreetin kosinimuunnoksen painokertoimien matalien arvojen havaitsemiseen perustuvan menetelmän onnistumisprosentit testiaineiston suhteen. Vasemmanpuoleiseen sarakkeeseen on merkitty kuvan pakkauslaatu ennen muokkausta. Ylimmällä rivillä on kuvan pakkauslaatu muokkauksen jälkeen.

Q1 / Q2	60	70	80	90	Häviötön
60	0	0	78	78	78
70	0	0	39	78	78
80	0	0	0	78	78
90	0	0	0	0	83

Taulukko 12: Lin ym. (2009) JPEG-kehikon tunnistamiseen pikselitasolla perustuvan menetelmän onnistumisprosentit testiaineiston suhteen. Vasemmanpuoleiseen sarakkeeseen on merkitty kuvan pakkauslaatu ennen muokkausta. Ylimmällä rivillä on kuvan pakkauslaatu muokkauksen jälkeen.

Q1 / Q2	60	70	80	90	Häviötön
60	0	0	0	6	22
70	0	0	0	0	17
80	0	0	0	0	11
90	0	0	0	0	0

Esitetyt tunnistusprosentit eivät välttämättä yksistään ole erityisen kiinnostavia. En esimerkiksi esitä lainkaan menetelmien kokonaistarkkuutta kuvaavaa yksittäistä tunnistusprosenttia. Tällainen luku ei olisi kovinkaan kuvaava, sillä tulokseen vaikuttavat varsin monet tekijät. Onnistumisprosentit riippuvat oleellisesti siitä, minkälaisia tapauksia aineistoon valitaan. On myös jokseenkin hankalaa määritellä sitä, minkälainen testiaineisto olisi jollain tavalla mahdollisimman edustava. Tärkeämpää on varmasti se, minkälaisia havaintoja tulosten perusteella voidaan tehdä eri menetelmien toimintaedellytyksistä. Testitulosten perusteella tärkein havainto on epäilemättä se, että kaikkein olennaisinta kuvaväärennösten onnistuneelle tunnistamiselle on se, millaisia pakkauslaatuja on käytetty ennen ja jälkeen kuvan muokkaamisen. Muokkaamisen jälkeen käytetyn laadun tulisi olla ainakin hieman ennen muokkaamista käytettyä laatua korkeampi. Usein kymmenen yksikön ero pakkauslaadussa näyttäisi riittävän. Toisaalta merkitystä näyttäisi olevan myös laatujen lähtötasolla, eli sillä, mihin kohtaan asteikkoa kyseinen väli osuu. Kun laadut olivat 60 ja 70, vain kaksi tapausta testiaineistosta tunnistettiin oikein. Laaduilla 70 ja 80 oikeita tunnistuksia oli jo 8. Laaduilla 80 ja 90 tunnistettiin oikein 14 tapausta. Itse asiassa yhtä poikkeusta lukuun ottamatta tätä suurempaa onnistumisastetta ei saavutettu millään pakkauslaatujen yhdistelmällä. Neljän kuvan osalta tunnistus epäonnistui lähes aina, käytetyistä pakkauslaaduista riippumatta. Nämä epäonnistumiset selittyvät kuvan sisällön ja muokkaustyyppien vaikutuksella. Yleisesti ottaen muokkauksen tyypin, kuvan sisällön ja muokatun alueen koon vaikutukset ovat kuitenkin ratkaisevampia silloin, kun käytetyt pakkauslaadut ovat lähellä toisiaan. Esimerkkitapauksena voidaan tarkastella kuvaa 18, jossa näkyvän henkilöauton rekisterinumero on väärennetty. Kuvissa 19 ja 20 kyseinen väärennös on tunnistettu Hen ym. (2006) ja matalien painokertoimien arvojen tunnistamiseen perustuvilla menetelmillä. Molemmissa tapauksissa

alkuperäinen kuva oli tallennettu JPEG-muotoon laadulla 70. Muokkauksen jälkeiset tallennusmuodot ovat vasemmalta oikealle JPEG laadulla 80, JPEG laadulla 90 sekä häviötön PNG. Kahdessa oikeanpuoleisessa tapauksessa tulos on melko selkeä, ja kyseisten tallennusmuotojen välillä ei ole merkittävää eroa. Sen sijaan ensimmäisessä tapauksessa tuloksen tulkinta ei ole aivan yksiselitteistä, sillä tuloskuvissa esiintyy paljon häiriötekijöitä. Tuloksissa näkyy kyllä selkeä keskittymä väärennetyllä alueella, mutta ilman minkäänlaista ennakkotietoa kuvasta, tulosten tulkinta saattaisi olla epävarmaa.

Kuva 18: Kuvan henkilöauton rekisterinumero on väärennetty.

Kuva 19: Kuvan 18 väärennös tunnistettu Hen ym. (2006) menetelmällä. Alkuperäinen kuva oli tallennettu JPEG-muotoon laadulla 70. Muokkauksen jälkeiset tallennusmuodot ovat vasemmalta oikealle JPEG laadulla 80, JPEG laadulla 90 ja häviötön PNG.

Kuva 20: Kuvan 18 väärennös tunnistettu matalien painokertoimien arvojen tunnistamiseen perustuvalla menetelmällä. Alkuperäinen kuva oli tallennettu JPEG-muotoon laadulla 70. Muokkauksen jälkeiset tallennusmuodot ovat vasemmalta oikealle JPEG laadulla 80, JPEG laadulla 90 ja häviötön PNG.

Oikeiden tunnistustulosten perusteella kuvan diskreetin kosinimuunnoksen painokertoimia hyödyntävät tekniikat ovat selkeästi JPEG-kehikon tunnistukseen perustuvia menetelmiä luotettavampia. Useimmissa tapauksissa Hen ym. (2006) kaksoiskvantisointiin perustuva menetelmä sekä siihen perustuva oma muunnelmani toimivat lähes samalla tavoin. Testiaineistossa oli kuitenkin joitakin tapauksia, joissa menetelmien välillä esiintyi eroja. Osa tapauksista oli tunnistettavissa vain toisella menetelmistä. Myös tuloksen selkeydessä oli eroja. Toimivampi menetelmä vaihteli tapauksittain. Suoritettujen testien perusteella en osaa eritellä tarkemmin niitä olosuhteita tai tekijöitä, jotka vaikuttivat menetelmien keskinäiseen paremmuuteen.

Lin ym. (2009) pikselipohjainen JPEG-kehikon tunnistusmenetelmä tunnisti monissa tapauksissa muokatun alueen oikein, mutta autentikoinnin onnistuminen kaatui tyypillisesti liiallisiin virhetunnistuksiin.

5.4 Muokkaustyypin vaikutus tuloksiin

Kuten aiemmin todettiin, digitaalinen kuvankäsittely tarjoaa lähes rajattomat mahdollisuudet kuvien muokkaamiselle. Olen tässä yhteydessä rajannut tarkastelun kaikkein tyypillisimpiin muokkaustapoihin. Tällaiset muokkaustoimenpiteet ovat toteutettavissa tavalla tai toisella useimmissa vähänkään edistyneemmissä kuvankäsittelyohjelmissa. Pääpaino on kuvan elementtien poistamisessa ja liittämisessä, sillä kuten todettu, tällaiset toimenpiteet ovat varmasti kaikkein todennäköisimmin omiaan muuttamaan kuvan sisältöä harhaanjohtavasti. Taulukossa 13 on lista käsitellyistä muokkaustyypeistä, sekä tieto siitä, kuinka monessa testiaineiston kuvassa kutakin muokkaustyyppiä on käytetty. Tarkemmat selvitykset muokkaustoimenpiteistä löytyvät liitteestä 2. Osa muokkaustyypeistä vaatii lisämäärittelyksi jonkinlaisia parametrejä. Esimerkiksi voidaan määritellä se, kuinka paljon kohinaa lisätään tai kuinka oleellisesti kontrastia taikka värejä muutetaan. Tällaisilla tekijöillä on epäilemättä merkitystä lopputuloksen kannalta, mutta en ole tässä yhteydessä voinut sen tarkemmin huomioida tätä seikkaa. Parametrien arvot on valittu silmämääräisesti siten, että lopputulos näyttää kohtuullisen uskottavalta, eikä muokkausta voi helposti todeta paljaalla silmällä.

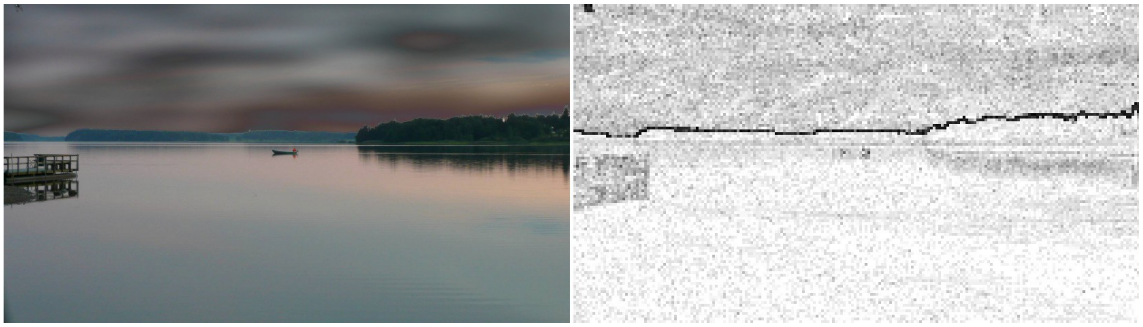
Taulukko 13: Testiaineistossa käytettyjen muokkaustyyppien lukumäärät.

Muokkaustyyppi	Lukumäärä
Elementin liittäminen toisesta kuvasta	4
Elementin liittäminen samasta kuvasta / alueen peittäminen kopioimalla taustaa	4
Kohinan lisäys, pehmennys tai terävöitys	2
Geometrinen vääristys	2
Muutos väreihin	2
Muutos kontrastiin	2
Sisällön generointi ohjelmallisesti	2

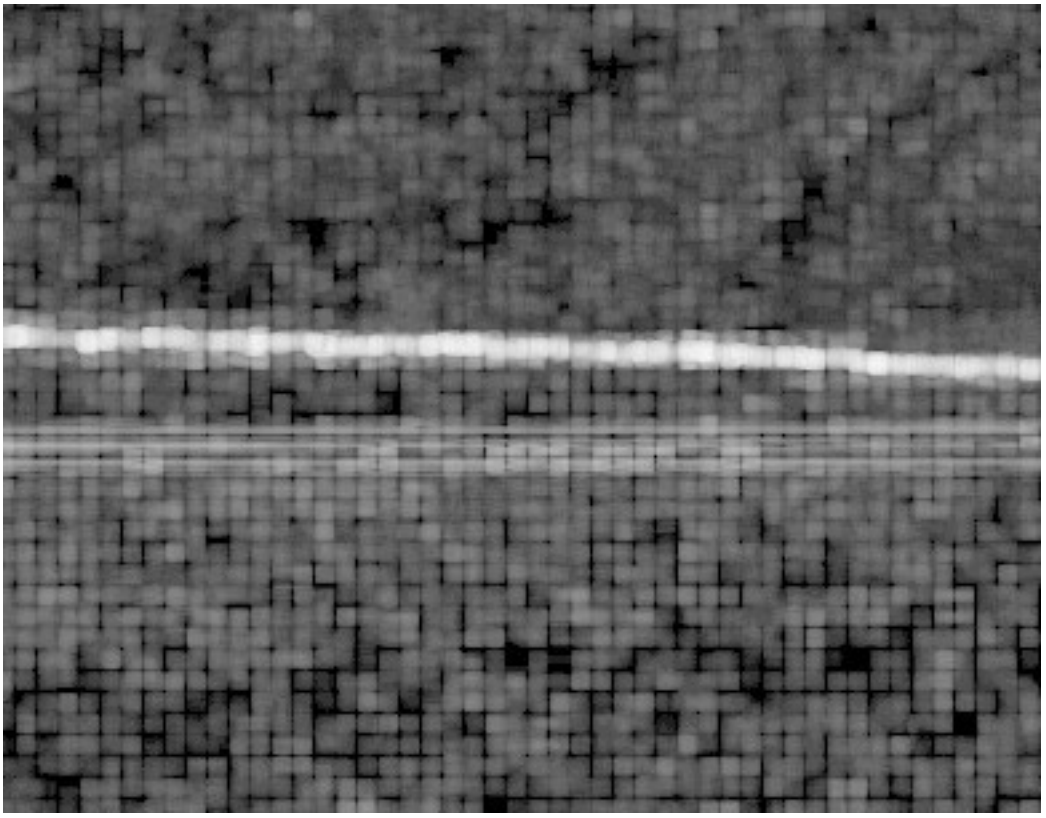
Kuten oletettua, kaikkein varmimmin tunnistettavissa olivat tapaukset, jossa kuvan elementtejä on lisätty tai poistettu. Yhtä lailla selkeitä tapauksia olivat ainakin geometriset vääristykset ja kohinan lisäys. Näissä tapauksissa tunnistuksen selkeys liittyy epäilemät-

tä siihen seikkaan, että kyseiset muokkaustyypit muuttavat kaikkein oleellisimmin kuvan rakennetta. Sen sijaan vaikkapa värien tai kontrastin muutokset saattavat säilyttää kuvan rakenteen kohtuullisen hyvin ennallaan. Vaikutus riippuu tietenkin muutoksen voimakkuudesta. Ainakaan lievät muutokset eivät välttämättä muuta ratkaisevasti kuvan pisteiden suhdetta toisiinsa. Vaikuttaisi siltä, että tällaisissa tapauksissa muokkausjäljet ovat usein parhaiten havaittavissa muokatun ja aidon alueen rajalla. Tämä saattaa kuitenkin aiheuttaa tulkintaongelmia, sillä tuloskuvassa näkyvä tunnistettu alue voi jäädä varsin pieneksi, tai se voidaan tulkita kuvan rajojen aiheuttamaksi virhetunnistukseksi.

Vastaava ongelma oli havaittavissa tapauksissa, jossa kuvaan generoitiin sisältöä ohjelmallisesti. Tämä johtuu siitä, että testitapauksissa generoitu sisältö ei korvannut täydellisesti kuvan alkuperäistä sisältöä, vaan alueet ikään kuin sekoittuivat keskenään. Tästä on nähtävissä esimerkki kuvassa 21. Kuvan maisemaan on ohjelmallisesti generoitu pilviä. Pilvet eivät kuitenkaan täydellisesti korvaa alkuperäistä kuvaa, vaan ne on sekoitettu alkuperäiseen kuvaan. Oikealla puolella on tuloskuva, joka on saatu Hen ym. (2006) menetelmällä. Tunnistetut alueet rajoittuvat muokatun ja alkuperäisen alueen rajalle. Tällaisen tuloksen tulkinta ei ole aivan ongelmatonta. Lisävalaistusta voidaan saada erottamalla kuvasta JPEG-kehikko. Näin on tehty kuvassa 22. Tuloksesta voidaan huomata, että muokatulla alueella kehikko on jonkin verran epäselvempi, kuin alkuperäisellä alueella.



Kuva 21: Kuvaan on generoitu ohjelmallisesti pilviä taivaalle. Oikealla väärennös on tunnistettu Hen ym. (2006) menetelmällä. Kuvan alkuperäinen tallennusmuoto oli JPEG laadulla 70 ja muokkauksen jälkeen häviötön PNG.



Kuva 22: Kuvan 21 JPEG-kehikko tunnistettu Lin ym. (2008) menetelmällä. Muokatulla alueella, eli valkoisen rajan yläpuolella kehikko on jonkin verran epäselvempi kuin aidolla alueella. Kuva on rajattu aidon ja muokatun alueen rajalle.

On tietenkin huomioitava, että testiaineiston muodostuksessa oleellisin huomio on kiinnitetty pakkauslaatujen variaatioihin. Muokkaustyyppien osalta aineisto on sen sijaan melko suppea. Yksittäisten poikkeustapausten ylikorostumisen vuoksi en esitä muokkaustyypin mukaan jaoteltuja testituloksia. Tarkempien johtopäätösten tekeminen edellyttäisi tältä osin kattavampaa aineistoa. Esimerkkitapaukset voivat ehkä kuitenkin antaa viitettä siitä, kuinka monimuotoisesta ilmiöstä on kyse.

5.5 Kuvan sisällön vaikutus tuloksiin

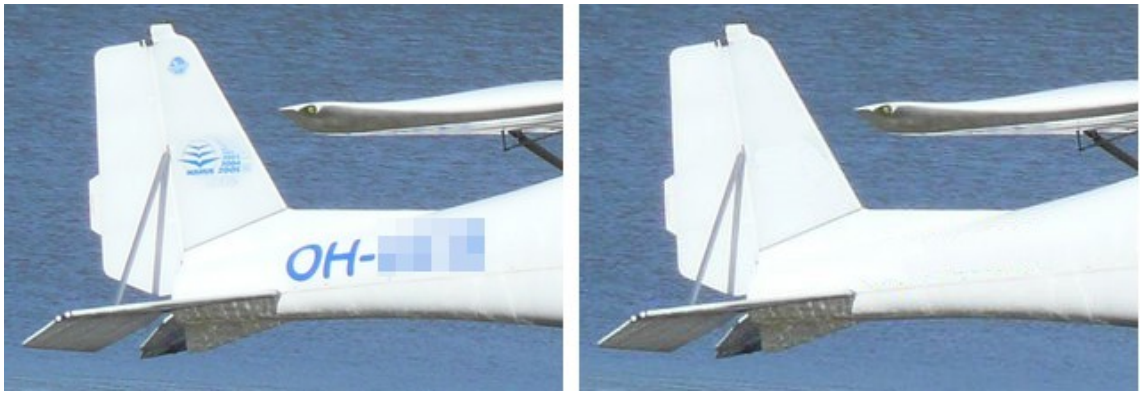
Eräs valokuvien autentikoinnin kannalta merkittävä tekijä on kuvan sisältö. Merkitystä voi olla etenkin kuvan sisällön monimutkaisuudella ja sillä, missä määrin kuva sisältää teräviä reunoja, eli nopeita kirkkauden vaihteluita. Tällaisten tekijöiden merkitys on erityisen selvä, jos mietitään sitä, kuinka JPEG-kuva on rakennettu. Kuten aiemmin todettiin, diskreetti kosinimuunnos muuntaa kuvan sinimuotoisiksi osakomponenteiksi. Täl-

lainen esitys puolestaan soveltuu kaikkein luonnollisimmin pehmeiden väriliukujen esittämiseen. Sen sijaan terävien reunojen muodostamiseksi joudutaan summaamaan useita eri taajuuksia. Esimerkiksi kuvassa 23 näkyvä auringonlaskua vasten kuvattu puu sisältää kontrastiltaan hyvin voimakkaita rajoja. Tällaiset alueet aiheuttavat tyypillisesti jonkin verran häiriötä autentikointimenetelmien tuloksiin. Esimerkkitapauksessa ongelmallista on myös se, että muokattu alue on varsin pieni, jolloin se erottaminen häiriöiden joukosta on entistä vaikeampaa. Esimerkkikuvassa näkyvät tulokset, kun väärennös on tunnistettu diskreetin kosinimuunnoksen painokertoimien matalia arvoja havaitsemalla. Pakkauslaadut ennen ja jälkeen muokkauksen olivat 70 ja 80 sekä 70 ja 90. Kuten voidaan havaita, ensimmäisessä tapauksessa muokatun alueen luotettava erottaminen on häiriöiden vuoksi mahdotonta. Jälkimmäisessä tapauksessa alue erottuu kohtuullisesti kirkkautensa vuoksi.

Kuva 23: Maisemaan on lisätty sorsapoikue toisesta valokuvasta. Väärennös on tunnistettu diskreetin kosinimuunnoksen painokertoimien matalia arvoja havaitsemalla. Keskellä kuvan pakkauslaadut ennen ja jälkeen muokkauksen olivat 70 ja 80, oikealla 70 ja 90.

Ongelmallisia voivat olla myös sellaiset alueet, jotka eivät sisällä lainkaan yksityiskohtia. Tällainen voi olla tilanne esimerkiksi kuvan ylivalottuneilla alueilla. Tällöin kyseisen alueen koko sisältö saattaa muodostua täysin yhtenäisestä valkoisesta väristä. Kaikki toteutetut menetelmät hyödyntävät erilaisia JPEG-pakkauksen aiheuttamia jälkiä, ja on selvää, ettei sellaisia löydy täysin yhtenäisiltä värialueilta. Tällainen tilanne on esimer-

kiksi kuvassa 24. Lentokoneen peräosasta on poistettu tunnus sekä logo, mutta mikään käsitellyistä autentikointimenetelmistä ei havaitse alueella mitään poikkeavaa.



Kuva 24: Lentokoneen peräosasta on poistettu tunnus sekä logo.

Tulosten tulkinnassa voidaan jossain määrin hyödyntää tyypillisten ongelma-alueiden tuntemusta ja yksinkertaista päättelyä. Esimerkiksi varsin tavallista on, että muun muassa puiden ja pensaiden oksat, sekä lautojen tai tiilien muodostamat kuviot rakennusten seinissä saattavat aiheuttaa virhetunnistuksia. Jos tuloksen oikeellisuus epäilyttää, voi olla hyödyllistä verrata kuvan merkattuja alueita muuhun kuvan sisältöön. Jos kuvasta löytyy sisällöltään täysin vastaavia alueita, jotka eivät ole merkattuja, ei kyseessä luultavasti ole virhetunnistus. Tällaisten tulkintaperiaatteiden toteutus automaattisesti voi olla suhteellisen hankalaa, joten ne soveltunevat lähinnä ihmisten suorittamaan tarkasteluun.

5.6 Erikoistapaukset

Varsinaisessa testiaineistossa kuvien pakkauslaadut rajoittuivat välille 60-90 ja vaihtelu tapahtui kymmenen yksikön väleissä. Tämän lisäksi testasin muutaman yksittäistapauksen avulla tilanteita, joissa pakkauslaatujen ero oli pienempi ja lähtötaso korkeampi. Testitapauksessa käytin muokkaustyyppinä alueen kopioimista saman kuvan sisällä. Taulukkoon 14 on listattu korkeimpien pakkauslaatujen osalta onnistuneeseen tunnistukseen vaadittavat erot muokkausta edeltävän ja seuraavan tallennuksen pakkauslaaduissa. Kuten taulukosta voidaan havaita, jopa yhden yksikön ero pakkauslaadussa voi olla riittävä onnistuneeseen tunnistukseen. Näin oli kuitenkin vain lähtötasoilla 97 ja 98. Vaadittava ero näyttäisi kasvavan lähtötason alentuessa. Toisaalta lähtötasoilla 99 ja 100 onnistunutta tunnistusta ei saatu lainkaan, vaikka muokkauksen jälkeinen tallennusmuo-

to oli häviötön PNG. Oletettavasti kaikkein korkeimmilla pakkauslaaduilla kuvaan ei muodostu kyllin selkeästi havaittavia muutoksia, jotta niitä voitaisiin hyödyntää kuvan aitouden arvioinnissa. Testauksessa käytettiin kosinimuunnoksen painokertoimia hyödyntäviä autentikointitekniikoita. On kuitenkin huomattava, että kyseiset tulokset on saatu yhden kuvan ja yhden muokkaustyypin perusteella.

Taulukko 14: Onnistuneeseen väärennöksen tunnistukseen vaadittavat erot pakkauslaaduissa.

Pakkauslaadun lähtötaso	Tarvittava ero pakkauslaadussa
80	7
90	4
95	2
96	2
97	1
98	1
99	Ei tulosta
100	Ei tulosta

Testien perusteella kuvan kosinimuunnoksen painokertoimia hyödyntävät tekniikat toimivat yleensä JPEG-kehikon tunnistamiseen perustuvia tekniikoita luotettavammin. On kuitenkin olemassa tilanteita joissa ensin mainitut menetelmät eivät toimi lainkaan.

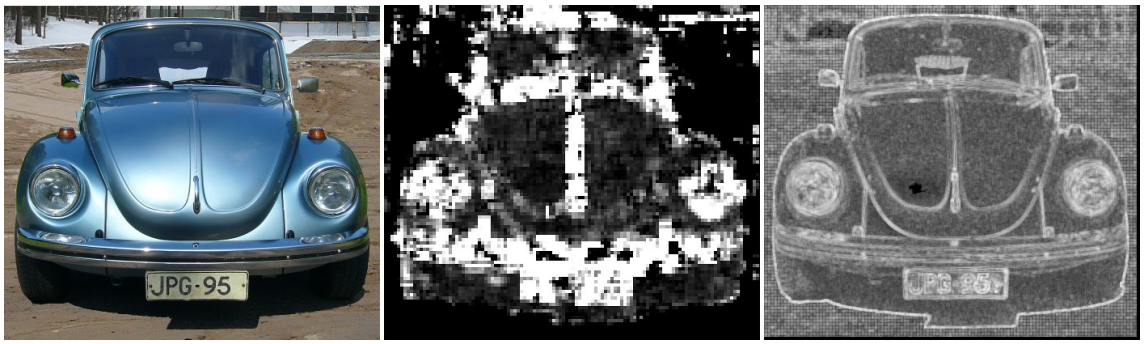
Usein kuvan muokattujen alueiden voidaan olettaa olevan suhteellisen pieniä suhteessa koko kuva-alaan. On kuitenkin täysin mahdollista, että muokattu alue kattaa itse asiassa suurimman osan kuvan alasta. Tällaisessa tilanteessa alkuperäisessä tallennuksessa käytettyjen kvantisointiarvojen arvioiminen ei välttämättä onnistu. Epäonnistuminen johtuu siitä, että arviointiin käytetään koko kuvan sisältämä informaatio. Tämä puolestaan on välttämätöntä sen vuoksi, että lähtökohtaisesti ei tiedetä missä muokatut alueet sijaitsevat. Jos muokattu alue kattaa suurimman osan kuvasta, on selvää, että sen vaikutus kvantisointiarvojen arviointiinkin on suurin. Muokattu alue ei välttämättä sisällä lainkaan kvantisointijälkiä, tai jos sisältää, on lohkojen kohdistus mitä luultavimmin väärä. Tästä johtuen arviot kvantisointiarvoista vastaavat todennäköisesti pakkaamattomasta kuvasta saatuja arvioita.

Kuten aiemmin todettiin, kuvan kosinimuunnoksen painokertoimia hyödyntävien tekniikoiden lähtöoletuksena on se, että alkuperäinen JPEG-lohkojen kohdistus ei ole muuttu-

nut muokkausta edeltävän ja muokkausta seuraavan tallennuksen välillä. Jos näin käy, eivät kyseiset tekniikat toimi lainkaan.

JPEG-kehikon tunnistamiseen perustuvat menetelmät eivät sen sijaan ole yhtä riippuvaisia alkuperäisten lohkojen oikeasta kohdistuksesta tai muokatun alueen koosta.

Kuvassa 25 on esimerkki tapauksesta, jossa muokattu alue kattaa suurimman osan kuva-alasta. Keskimmäisessä kuvassa väärennös on tunnistettu Lin ym. (2009) menetelmällä. Oikealla kuvasta on erotettu JPEG-kehikko Lin. Ym (2008) menetelmällä. Molemmilla menetelmillä väärennös on tunnistettavissa varsin selvästi. Sen sijaan kuvan kosinimuunnoksen painokertoimia hyödyntävät menetelmät antavat tyhjän tuloskuvan kyseisessä tapauksessa.

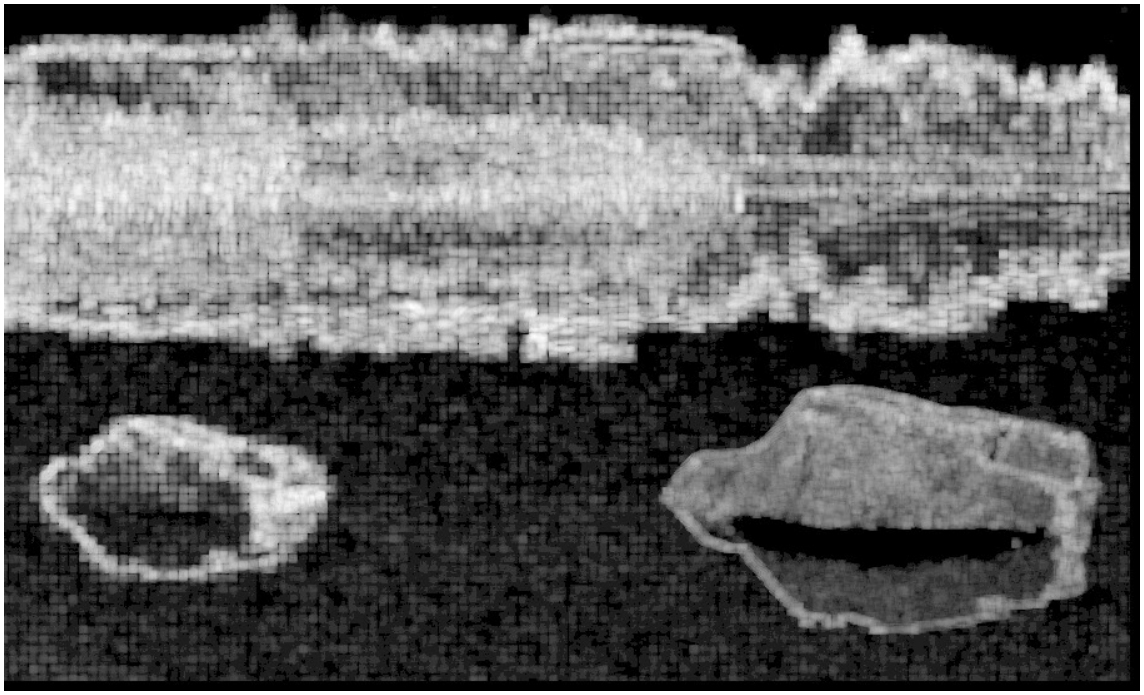


Kuva 25: Kuvan henkilöauto on siirretty väärään taustaan. Keskellä väärennös on tunnistettu Lin ym. (2009) menetelmällä ja oikealla kuvasta on erotettu JPEG-kehikko Lin ym. (2008) menetelmällä. Alkuperäisen kuvan tallennusmuoto oli JPEG laadulla 70. Muokkauksen jälkeinen tallennusmuoto oli häviötön PNG.

Kuvassa 26 on esimerkki tilanteesta, jossa kuvaa on rajattu ennen muokkausta. Tämä aiheuttaa sen, että alkuperäinen JPEG-pakkauksen lohkojako muuttuu. Kuvan kosinimuunnoksen painokertoimia hyödyntävät autentikointimenetelmät eivät kykene tunnistamaan tällaista väärennöstä. Kuvassa 28 näkyy Lin ym. (2008) JPEG-kehikon tunnistukseen perustuvalla menetelmällä saatu tulos edellisestä kuvasta. Rajauksesta huolimatta tuloskuvasta on selvästi havaittavissa, että muokatun alueen, eli oikeanpuolimmainen kiven kohdalla JPEG-kehikko puuttuu kokonaan. Esimerkin tapauksessa kuvan alkuperäinen tallennusmuoto oli JPEG ladulla 80 ja muokkauksen jälkeinen muoto oli häviö-

tön PNG. Jos viimeisimpänä tallennusmuotona käytetään JPEG:iä, ei tilanne ole tietenkään yhtä selkeä.

Kuva 26: Kuvassa oikeanpuolimmainen kivi on liitetty toisesta kuvasta. Ennen muokkausta kuvaa on rajattu.

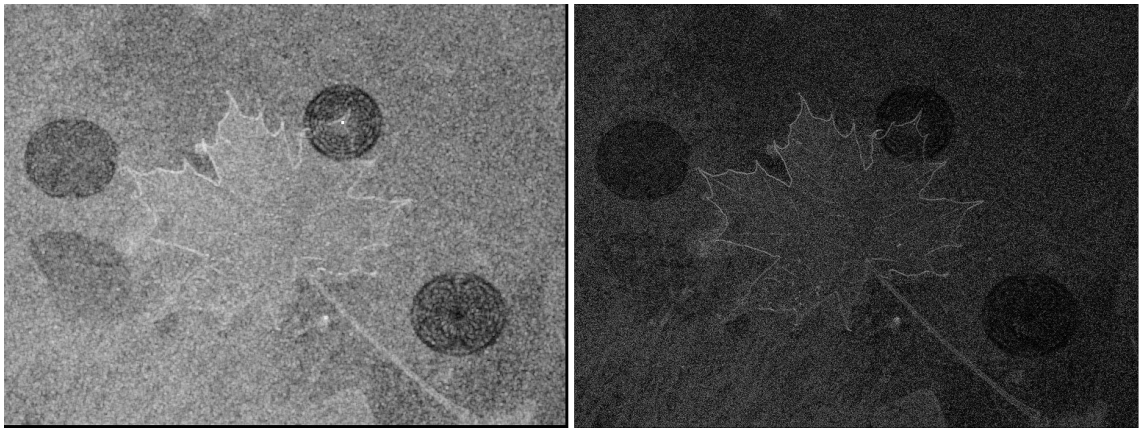


Kuva 27: Lin ym. (2008) menetelmällä erotettu JPEG-kehikko kuvasta, jota on rajattu ennen muokkausta. Kuvan tallennusmuoto ennen muokkausta oli JPEG laadulla 80 ja muokkauksen jälkeen häviötön PNG.

Olen havainnut, että Lin ym. (2008) menetelmän avulla on mahdollista paljastaa joitakin muokkausjälkiä myös tilanteissa, joihin sitä ei varsinaisesti ole tarkoitettu. Menetelmä voi paljastaa poikkeavia jälkiä jopa kuvista, jotka eivät sisällä pakkausjälkiä lainkaan. Toisaalta menetelmä soveltuu kaikista muista käsitellyistä menetelmistä poiketen tunnistamaan muokkausjälkiä myös tilanteissa, joissa muokkauksen jälkeinen pakkauslaatu on muokkausta edeltävää laatua heikompi. Tämä on kuitenkin melko rajallista ja soveltuu vain hieman korkeammille pakkauslaaduille. Tällä tavoin havaittavien jälkien tunnistamiseen ei varmastikaan ole kovin helposti kehitettävissä automaattista tunnistustapaa, vaan niiden tulkinta on puhtaasti analysoijan tapauskohtaista harkintaa. Kuva 28 on häviötön PNG-kuva, johon on kohdistettu neljä erilaista geometrista vääristystä (venytys ulospäin sekä sisäänpäin, pyörre sekä aaltotehoste). Kolmessa tapauksessa vääristysten vaikutusala on määritelty ympyrän muotoiseksi ja yhdessä tapauksessa se on hieman epäsäännöllisempi. Kuvassa 29 on kaksi tulosta, jotka on aikaansaatu Lin ym. (2008) menetelmällä sekä laskemalla naapuripisteiden väliset erot. Etenkin ympyrän muotoiset alueet erottuvat varsin selvästi tuloskuvissa. Kuviot erottuvat helposti jo pelkästään epäluonnollisen symmetrisyytensä vuoksi. Mikään kuvan sisällössä ei myöskään selitä ky-

seisiä kuvioita. Kuten tuloskuvasta näkyy, kuviot on erotettavissa kohtuullisen hyvin myös laskemalla naapuripisteiden väliset erot vastaavasti kuin Lin ym. (2009) JPEG-kehikon tunnistamiseen pikselitasolla perustuvassa menetelmässä. Kosinimuunnoksen painokertoimiin perustuva tapa antaa kuitenkin hieman selkeämpiä tuloksia etenkin silloin, jos kuvan viimeinen tallennuslaatu on heikompi kuin alkuperäinen tallennuslaatu. Esimerkin PNG-kuvan jäljet ovat vielä heikosti erotettavissa, kun kuva tallennetaan JPEG-muotoon laadulla 80.

Kuva 28: Kuvaan on kohdistettu geometrisia vääristyksiä neljään eri kohtaan.



Kuva 29: Kuvan 28 geometriset vääristykset tunnistettu Lin ym. (2008) menetelmällä (vas.) sekä laskemalla naapuripikseleiden väliset erot (oik.). Vääristykset näkyvät säännöllisinä tummina alueina tuloskuvissa.

5.7 Menetelmien suoritusajat

Nykyisillä digitaalikameroilla tuotettavat kuvat voivat olla resoluutioltaan huomattavan suuria. Lisäksi kuvien käsittelyoperaatiot ovat usein varsin monimutkaisia. Sen vuoksi on syytä tarkastella hieman käsiteltyjen autentikointimenetelmien suoritusaikoja. Suoritusajan merkitys riippuu epäilemättä autentikointimenetelmän käyttökontekstista. Suoritusaika voi olla hyvinkin kriittinen tekijä, mikäli tarkoitus on käsitellä automatisoidusti hyvin suuria määriä kuvia. Sen sijaan yksittäisten kuvien tarkastelussa suoritusaika saattaa olla lähes merkityksetön tekijä.

Suoritusajat vaihtelevat melko oleellisesti eri menetelmien välillä. Taulukossa 15 on nähtävissä kaikkien esiteltyjen menetelmien suoritusajat kolmen eri kokoisen kuvan käsittelyssä. Suoritusaikaan ei sisälly kuvan lataaminen kovalevyltä keskusmuistiin, vaan ainoastaan itse algoritmin suoritus ja tuloskuvan esittäminen. Testauksessa käytetyssä tietokoneessa oli Intel Core 2 Duo E6750 (2.66 GHz) suoritin ja 4 gigatavua keskusmuistia.

Taulukko 15: Toteutettujen autentikointimenetelmien suoritusajat eri kokoisten kuvien käsittelyssä.

Autentikointimenetelmä	**Kuvan resoluutio ja suoritusaika sekunneissa**		
	500 x 500	**1280 x 720**	**3328 x 1872**
Kaksoiskvantisoinnin tunnistus	0,65	5	32
Kuvan kosinimuunnoksen painokertoimien matalien arvojen tunnistus	0,3	1,16	7,5
JPEG-kehikon tunnistus pikselitasolla	1,95	7,2	52
JPEG-kehikon tunnistus kuvan kosinimuunnoksen painokertoimien perusteella	10	49	266

Kuten tuloksista näkyy, Hen ym. (2006) kaksoiskvantisointiin perustuva menetelmä sekä siihen pohjautuva oma muunnelmani toimivat huomattavasti nopeammin kuin JPEG-kehikon erottamiseen perustuvat menetelmät. Ensin mainittujen menetelmien osalta suoritusnopeutta olisi vielä mahdollista parantaa, mikäli kosinimuunnoksen painokertoimet luettaisiin suoraan JPEG-tiedostosta. Toisaalta taulukon 15 tuloksiin ei ole laskettu mukaan aikaa, joka kuluu kuvan lataamiseen tietokoneen keskusmuistiin. Painokertoimien lukeminen JPEG-tiedostosta vaikuttaisi suurelta osin juuri tähän vaiheeseen, sillä kuvaa ei tällöin tarvitse välttämättä ladata kokonaisuudessaan keskusmuistiin, eikä taajuuksia muuntaa pikselimuotoon (Lin ym. 2009). Oman versioni suoritusaika on vielä kohtuullisesti Hen ym. menetelmää lyhyempi. Tämä selittyy yksinkertaisemmalla toteutuksella.

Li ym. (2009) raportoivat JPEG-kehikon tunnistamiseen perustuvan menetelmänsä suoritusajaksi 24 sekuntia kun kuvan koko on 256 x 256 pikseliä. Omassa toteutuksessani suoritusaika on n. 10 sekuntia 500 x 500 pikselin kokoiselle kuvalle. Li ym. arvelevat oman toteutuksensa heikon suoritusajan johtuvan MATLAB-ympäristön tehottomista lajittelualgoritmeista, joita tarvitaan mediaanisuodatuksissa.

Lin ym. (2008) kehittämä JPEG-kehikon tunnistus kosinimuunnoksen painokertoimien perusteella on selkeästi hitain menetelmä. Tekijät raportoivat toteutuksensa suoritus-

ajaksi 20 sekuntia 512 x 512 -kokoiselle kuvalle. Omassa toteutuksessani vastaavan kuvan käsittely kestää hieman yli kymmenen sekuntia. 3328 x 1872 -kokoisella kuvalla suoritusaika on jo yli neljä minuuttia.

6 YHTEENVETO

Olen esitellyt tässä tutkielmassa joukon digitaalisten kuvaväärennösten tunnistamiseen tarkoitettuja menetelmiä. Tutkielman kolmannessa luvussa suoritin menetelmien vertailun neljä kohtaa käsittävän kriteeristön pohjalta. Vertailtavia ominaisuuksia olivat menetelmien tarkkuus, tunnistettavat muokkaustyypit, rajoitukset sekä vakaus vastatoimia vastaan. Vertailun perusteella voidaan havaita, että monet menetelmistä yltävät lähelle sadan prosentin tunnistustarkkuutta, jos tarkasteltavan kuvan laatu on riittävän hyvä. Eräs merkittävimmistä aitouden arviointia vaikeuttavista tekijöistä näyttäisi kuitenkin olevan kuvan häviöllinen pakkaaminen. Tämän vuoksi tutkielman varsinainen painoalue on kohdistettu menetelmiin, jotka toimivat häviöllisestä pakkauksesta huolimatta. Häviöllisistä kuvaformaateista ylivoimaisesti käytetyin on JPEG, mistä syystä valitsin erityistarkasteluun nimenomaan JPEG-pakkausta hyödyntäviä tekniikoita. Tutkielman neljännessä luvussa olen esitellyt neljä toteuttamaani autentikointimenetelmää, joista kolme on lähes sellaisenaan esitelty alan tutkimuskirjallisuudessa. Toteutetuista autentikointimenetelmistä kaksi hyödyntää JPEG-pakkauksen ytimeen kuuluvia diskreettiä kosinimuunnosta ja kvantisointia. Toiset kaksi menetelmää puolestaan hyödyntävät JPEG-pakkauksen kuvaan aiheuttamaa laatikoitumista. Luonnollisesti perusoletuksena menetelmien toiminnalle on se, että alkuperäinen kuva on JPEG-muotoinen. Sen sijaan kuvan viimeisin tallennusmuoto voi olla yhtä hyvin myös häviötön.

Tutkimuksen tarkoituksena oli selvittää, onko useampaa eri tekniikka hyödyntämällä mahdollista saavuttaa oleellisesti luotettavampia tuloksia kuin yksittäisellä tekniikalla. Testien perusteella voidaan havaita, että joissakin tilanteissa käytetyt tekniikat todella toimivat eri tavoin. Toisin sanoen useampaa tekniikkaa hyödyntämällä voidaan kattaa suurempi joukko tilanteita, kuin yksittäisellä tekniikalla. Valitettavasti käytettyjen tekniikoiden perustavat toimintaedellytykset ovat silti jotakuinkin samat. Tekniikasta riippumatta kaikkein oleellisin tekijä kuvaväärennösten onnistuneen tunnistamisen kannalta näyttäisi olevan se, minkälaisia pakkauslaatuja käytetään ennen ja jälkeen kuvan muokkaamisen. Onnistumisen edellytyksenä on se, että jälkimmäinen pakkauslaatu on edellistä ainakin jonkin verran korkeampi. Vaikutusta on kuitenkin myös laatujen lähtötasolla, eli sillä, mihin kohtaan asteikkoa pakkauslaadut sijoittuvat. Pakkauslaadun ollessa yli 90, voi jopa 1-4:n yksikön ero riittää. Alhaisemmilla laatutasoilla (60-70) saatetaan tar-

vita jopa 20 yksikön ero pakkauslaatujen välillä. Etenkin silloin, kun pakkauslaadut ovat suhteellisen lähellä toisiaan, korostuu muiden vaikuttavien tekijöiden, kuten muokkauksen tyypin, kuvan sisällön ja muokatun alueen koon merkitys. Mainittujen tekijöiden merkityksen tarkempi määrittely vaatisi kuitenkin kattavamman testiaineiston, kuin mitä tämän tutkimuksen yhteydessä on käytetty. Tällöin voitaisiin tutkia tarkemmin kutakin muuttujaa yksitellen. Tämänkin tutkimuksen perusteella voidaan kuitenkin tehdä joitakin johtopäätöksiä.

Muokkaustyyppien osalta oleellista näyttäisi olevan se, kuinka ratkaisevasti kuvan rakenne muuttuu. Oleellisimmin kuvan rakennetta muuttavat ainakin kuva-alueiden kopioiminen ja erilaiset geometriset vääristykset. Kuva-alueiden kopioimisen kannalta ei ole merkitystä sillä, onko liitetty alue peräisin samasta vai eri kuvasta. Sen sijaan kuvan rakennetta vähemmän muuttavien muokkaustapojen osalta väärennösten tunnistaminen voi olla epävarmempaa. Tällaisia muokkaustyyppejä ovat ainakin värien tai kontrastin muuttaminen. Täsmällisten johtopäätösten tekemistä tällaisissa tapauksissa vaikeuttaa se, että yleensä kyseisten muokkaustyyppien vaikutuksen suuruus on määriteltävistä erilaisin asetuksin. Kaiken kaikkiaankin digitaalisen kuvankäsittelyn eri mahdollisuudet ovat niin valtaisat, ettei niiden vaikutuksia ole aivan yksinkertaista määritellä.

Kuvan sisällön merkitys puolestaan tulee esiin siinä, että hyvin monimutkaiset ja voimakkaita reunoja sisältävät kuva-alueet saattavat aiheuttaa häiriöitä autentikointimenetelmien tuloksiin. Tällöin muokattujen alueiden tunnistaminen voi olla vaikeaa, etenkin jos muokattu alue on hyvin pieni.

On myös huomattava, että esiteltyjen tekniikoiden toimintaedellytykset ovat jossain määrin rajalliset. Tärkein edellytys on se, että alkuperäisen pakkauksen aiheuttamat jäljet ovat edes jossain määrin tallella vielä kuvan viimeisimmän tallennuksen jälkeen. Pakkausjälkien tuhoamiseen on kuitenkin varsin helppoja keinoja. Helpoin tapa on esimerkiksi pienentää tai kallistaa kuvaa muokkaamisen jälkeen. Tämä onnistuu helposti useimmilla kuvankäsittelyohjelmilla. Myös kuvan rajaaminen on kohtalokasta kosinimuunnoksen painokertoimia hyödyntäville tekniikoille. JPEG-kehikon tunnistamiseen perustuvien menetelmien osalta tämä ei ole ongelmallista. Näiden tekniikoiden osalta sen sijaan on erityisen tärkeää, että kuvan viimeisin tallennuslaatu on suhteellisen korkea. Paras tulos saavutetaan, kun viimeisimmäksi käytetään häviötöntä tallennusmuotoa.

Käsitellyt menetelmät voidaan jakaa myös käyttötarkoituksensa puolesta kahteen eri ryhmään. Kuvan diskreetin kosinimuunnoksen painokertoimia tarkastelevat menetelmät voisivat soveltua jossain määrin myös täysin automatisoituun kuvien tarkastustoimintaan. Tällöin menetelmien tuloksen täytyisi olla binääriarvoinen, eli antaa joko kyllä- tai ei-vastaus suhteessa kuvan aitouteen. Kuvan muodossa esitetyt tulokset hyödyttävät ainoastaan ihmisiä. Tuloksen muuntaminen binääriarvoiseksi voisi mahdollisesti perustua yhtenäisten ja selvästi erottuvien alueiden etsimiseen tuloskuvasta.

Sen sijaan JPEG-kehikon tunnistamiseen perustuvat menetelmät soveltunevat parhaiten ihmisen toimesta tapahtuvaan tarkempaan analysointiin. Ne voivat mahdollisesti antaa sellaista lisäinformaatiota esimerkiksi käytetyn muokkauksen laadusta, jota kaksi muuta esiteltyä menetelmää eivät tarjoa. Lisäksi JPEG-kehikon tunnistamiseen perustuvista menetelmistä ainakin Lin ym. (2008) diskreettiä kosinimuunnosta hyödyntävä toteutus olisi sellaisenaan turhan raskas soveltuakseen suuren kuvamäärän automaattiseen tarkastamiseen. Kyseiselle menetelmälle ei myöskään toistaiseksi ole esitetty toteutusta, joka havaitsisi automaattisesti kuvan epäilyttävät alueet. Lin ym (2009) pikselitasolla toimiva toteutus voisi periaatteessa soveltua myös automaattiseenkin käsittelyyn, mutta ongelmana on virhetunnistusten suuri määrä.

Kuten Chen ym. (2007) toteavat, kuvaväärennösten tunnistaminen on erittäin vaikea ongelma, johon ei ole yhtä yleispätevää ratkaisua. Tässä tutkielmassa esitetyt tulokset ja ratkaisut koskevat kyseisen ongelmakentän yhtä rajattua aluetta, eivätkä ne tarjoa aukotonta ratkaisua koko ongelmaan. Muokkausjälkien esiintyminen voidaan oikeissa olosuhteissa todentaa melko luotettavasti, mutta tällaisten jälkien puuttuminen ei kuitenkaan ole aukoton todiste kuvan aitoudesta.

Kun käytetään sekä pakkaamattomien, että pakattujen kuvien käsittelyyn tarkoitettuja menetelmiä, saadaan varmasti katettua kohtuullisen suuri joukko erilaisia olosuhteita. Vaikuttaisi kuitenkin siltä, että kaikesta huolimatta jäljelle jää joitakin tilanteita, jotka pysyvät toistaiseksi kaikkien autentikointimenetelmien ulottumattomissa. Vaikka tässä tutkielmassa esitellyt menetelmät ovat nimenomaan pakattujen kuvien käsittelyyn tarkoitettuja, törmätään niidenkin kanssa lopulta samaan ongelmaan kuin muidenkin menetelmien kanssa. Häviöllinen pakkaaminen hävittää aina kuvan yksityiskohtia, mikä tar-

koittaa eittämättä myös sitä, että kuvan aitouden arvioinnin kannalta tärkeitä vihjeitä menetetään.

Kuten Farid (2009a) toteaa, valokuvien autentikointitekniikat eivät varmastikaan koskaan kykene täysin ratkaisemaan ongelmaa valokuvien aitoudesta. Ne voivat kuitenkin tehdä uskottavien väärennösten tekemisestä entistä vaikeampaa ja hitaampaa. Ihmisten käyttämä harkinta on varmasti edelleen tärkeässä roolissa. Toisaalta vaikka autentikointimenetelmät eivät vielä tarjoaisikaan automaattisesti täysin yksikäsitteistä vastausta kuvan aitoudesta, voivat ne silti tarjota tärkeitä vihjeitä ihmisen suorittaman arvion tueksi. Kokonaisarvio saattaa rakentua hyvin monista erilaisista vihjeistä.

Kuvaväärennösten tunnistustekniikat tarjoavat varmasti hyvin monia jatkotutkimuksen aiheita. Tällaisia ovat ainakin uusien autentikointimenetelmien kehittäminen ja olemassa olevien menetelmien jatkokehittäminen. JPEG-pohjaisten menetelmien osalta hyödyllistä olisi varmasti tutkia mahdollisuuksia väärennösten tunnistamiseen tilanteissa, joissa muokkauksen jälkeinen tallennuslaatu on muokkausta edeltävää laatua heikompi.

LÄHTEET

Bovik, A. 2009. The Essential Guide to Image Processing. London: Academic Press.

Chen, M., Fridrich, J., Lukáš, J., & Goljan, M. 2007. Imaging sensor noise as digital X-ray for revealing forgeries. Teoksessa Information hiding, Lecture Notes in Computer Science. Saint Malo, France: Springer Berlin / Heidelberg, 342-358.

Cox, I. J., & Miller, M. L. 1997. Review of watermarking and the importance of perceptual modeling. Teoksessa Bernice E. Rogowitz & Thrasyvoulos N. Pappas (toim.) Human Vision and Electronic Imaging II, Perception and Watermarking, February 10-13. San Jose, CA, USA: SPIE, 92-99.

Efford, N. 2000. Digital Image Processing: A Practical Introduction Using Java. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Farid, H. 2009a. Image forgery detection. Signal Processing Magazine, IEEE 26(2), 16-25.

Farid, H. 2009b. Seeing is not believing. Spectrum, IEEE 46(8), 44-48.

Fridrich, J. 2002. Security of fragile authentication watermarks with localization. Teoksessa Edward J. Delp III & Ping W. Wong (toim.) Proc. SPIE Photonic West, Electronic Imaging 2002, Security and Watermarking of Multimedia Contents, April 29, 2002. San Jose, California: SPIE, 691-700.

Fridrich, J. 2009. Digital image forensics. Signal Processing Magazine, IEEE 26(2), 26-37.

Gallagher, A. C., & Tsuhan Chen. 2008. Image authentication by detecting traces of demosaicing. Teoksessa Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, June 23-28. Los Alamitos, CA, USA: IEEE Computer Society, 1-8.

Gloe, T., Kirchner, M., Winkler, A., & Böhme, R. 2007. Can we trust digital image forensics? Teoksessa Rainer Lienhart, Anand R. Prasad, Alan Hanjalic, Sunghyun Choi, Brian P. Bailey & Nicu Sebe (toim.) MULTIMEDIA '07: Proceedings of the

15th International Conference on Multimedia, Augsburg, Germany, September 24-29. New York, NY, USA: ACM Press, 78-86.

Gonzales, R. C. & Woods, R. E. 2002. Digital Image Processing, 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Hartung, F. & Kutter, M. 1999. Multimedia watermarking techniques. Proceedings of the IEEE 87(7), 1079-1107.

He, J., Lin, Z., Wang, L. & Tang, X. 2006. Detecting Doctored JPEG Images Via DCT Coefficient Analysis. Teoksessa Ales Leonardis, Horst Bischof & Axel Pinz (toim.) Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, May 7-13. Graz, Austria: Springer, 423-435.

Holliman, M. & Memon, N. 2000. Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes. Image Processing, IEEE Transactions on 9(3), 432-441.

Li, W., Yu, N. & Yuan Y. 2008. Doctored JPEG image detection. Teoksessa Proceedings of the 2008 IEEE International Conference on Multimedia and Expo, ICME 2008. IEEE International Conference on Multimedia and Expo, Hannover, Germany, June 23-26. 253-256.

Li, W., Yuan, Y. & Yu, N. 2009. Passive detection of doctored JPEG image via block artifact grid extraction. Signal Processing, Elsevier North-Holland, Inc. 89(9), 1821-1829.

Lin, Z., He, J., Tang, X. & Tang, C. 2009. Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis. Pattern Recognition 42(11), 2492-2501.

Kundur, D. & Hatzinakos, D. 1999. Digital watermarking for telltale tamper proofing and authentication. Proceedings of the IEEE 87(7), 1167-1180.

Mahdian, B. & Saic, S. 2008. Blind methods for detecting image fakery. Security Teoksessa Technology, 2008. ICCST 2008. 42nd Annual IEEE International Carnahan Conference on, San Francisco, USA, October 22-24. IAENG, 280-286.

Mahdian, B. & Saic, S. 2009. Using noise inconsistencies for blind image forensics. Image and Vision Computing 27(10), 1497-1503.

Miano, J. 1999. Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP. New York: ACM Press/Addison-Wesley Publishing Co.

Mohanty, S. P. 1999. Digital Watermarking: A Tutorial Review. Unversity of South Florida, Dept of Computer Science and Engineering.

Podilchuk, C. I. & Delp, E. J. 2001. Digital watermarking: Algorithms and applications. Signal Processing Magazine, IEEE 18(4), 33-46.

Popescu, A. C. 2005. Statistical Tools for Digital Image Forensics. Darthmouth College, Department of Computer Science, Ph.D. Dissertation.

Popescu, A. C., & Farid, H. 2004. Statistical tools for digital forensics. Teoksessa 6th International Workshop on Information Hiding, Toronto, Canada, May. Toronto, Canada: Springer, 128-147.

Popescu, A. C. & Farid, H. 2005. Exposing digital forgeries in color filter array interpolated images. Signal Processing, IEEE Transactions on 53(10), 3948-3959.

Rey, C. & Dugelay, J. 2002. A survey of watermarking algorithms for image authentication. EURASIP Journal on Applied Signal Processing 2002(6), 613-621.

Skodras, A., Christopoulos, C., & Ebrahimi, T. 2001. The JPEG 2000 still image compression standard. Signal processing Magazine, IEEE 18(5), 36-58.

Yunfei, J. & Ping, G. 2007. Comparative studies of feature extraction methods with application to face recognition. Teoksessa Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on, Montreal, Quebec, Canada, October 7-10. 3627-3632.

Zhang, Z., Ren, Y., Ping, X.-J., He, Z.-Y. & Zhang, S.-Z. 2008. A survey on passive-blind image forgery by doctor method detection.

LIITE 1

Taulukko 16: Testiaineistona käytetyt kuvaväärennökset

LIITE 2

Taulukko 17: Testiaineiston valokuville suoritettujen muokkaustoimenpiteiden kuvaukset.

Kuva	Muokkaustoimenpiteet
1	Linnun jalanjälki on monistettu ja käännetty vaakasuunnassa ympäri.
2	Oikeanpuoleinen kivi on liitetty toisesta kuvasta. Kyseessä on sama kivi lievästi eri kulmasta.
3	Kuumailmapallo on liitetty toisesta kuvasta.
4	Kukan terälehtiin on kohdistettu erilaisia geometrisia vääristyksiä, kuten pullistuksia ja aaltotehosteita.
5	Henkilöauton rekisterinumero on ensin poistettu kopioimalla taustaa. Sen jälkeen uusi rekisterinumero on luotu liittämällä kirjaimet ja numerot kahdesta eri kuvasta.
6	Laiturin lautoihin on lisätty kohinaa sekä kohdistettu pehmennystä ja terävöitystä (kaikki toimenpiteet on suoritettu eri kohtiin).
7	Lehden väri on muutettu punertavasta vihreäksi.
8	Lentokoneen peräosasta on taustaa kopioimalla poistettu tunnus ja logo.
9	Kuvaan on kohdistettu erilaisia geometrisia vääristyksiä, kuten pullistuksia ja aaltotehosteita.
10	Muutamia kuvan alueita on siirretty hieman pois paikaltaan.
11	Joidenkin pihlajanmarjojen väriä on muutettu.
12	Pilvien kontrastia on lisätty.
13	Rannalta on taustaa kopioimalla poistettu pullo.
14	Kuvaan on lisätty sorsapoikue toisesta kuvasta.
15	Taivaalle on generoitu pilviä ohjelmallisesti. Pilvet eivät korvaa täydellisesti kuvan alkuperäistä sisältöä, vaan alueet on sekoitettu keskenään.
16	Kuvaan on generoitu linssiheijastus ohjelmallisesti.
17	Vasemmalla näkyvän rakennuksen muutamiin osiin sekä oikealla näkyvälle nurmikolle on lisätty kohinaa.
18	Kukan keskiosan kontrastia on lisätty.

LIITE 3

Tässä liitteessä esitetään eri autentikointitekniikoiden keskeisimmät algoritmit pseudokoodina. Koodissa käytetään seuraavia tietotyyppejä:

- int: kokonaisluku
- double: reaaliluku
- complex: kompleksiluku
- int[]: kokonaislukutaulukko (vastaavasti muille tyypeille, kuten reaaliluvuille)
- int[,]: Kaksiulotteinen kokonaislukutaulukko (vastaavasti reaaliluvuille)
- Image: yleinen tyyppi bittikarttakuvalle (Periaatteessa kaksiulotteinen kokonaislukutaulukko, jossa eri värikanavat ovat eroteltavissa.)
- DctMatrix: kuvan diskreettiä kosinimuunnosta vastaava tyyppi (Periaatteessa kaksiulotteinen taulukko reaalilukuja)

Aliohjelmien nimissä on aluksi määritelty palautustyyppi ja lopuksi tarvittavat parametrit pilkuilla eroteltuna. Kuvien autentikointialgoritmit palauttavat tyypillisesti tuloskuvan kaksiulotteisen taulukon muodossa. Palautettu taulukko voitaisiin sen jälkeen muuntaa kuvan muotoon. Palautetun taulukon ulottuvuudet vastaavat yleensä kuvan dct-lohkojen lukumäärää. Tuloskuva voidaan ennen esittämistä skaalata vastaamaan alkuperäisen kuvan ulottuvuuksia. Tässä yhteydessä ei ole huomioitu kuvan reuna-alueita, jotka jäävät yli kahdeksan pikselin lohkojaosta.
Koodin lyhentämiseksi joitakin muitakin epäoleellisempia yksityiskohtia on karsittu. Esimerkiksi diskreettiä kosinimuunnosta ja Fourier-muunnosta ei ole kuvattu pseudokoodina, koska niiden laskukaavat on esitelty tutkielmassa aiemmin.
Joitakin muuttujia on selvyyden vuoksi määritelty aliohjelmien sisällä, vaikka käytännössä ne olisi ehkä syytä antaa aliohjelmille parametreina.

```
/*** Kaksoiskvantisointi (luku 4.2) ***//
// Aliohjelma laskee kaksoiskvantisointiin perustuvan arvion kuvan aitoudesta.
// Palautusarvona on kaksiulotteinen reaalilukutaulukko, jossa kukin solu vastaa
// yhdelle dct-lohkolle laskettua aitousarviota.
// Parametrina annetaan tarkasteltava kuva.
double[,] DoubleQuantizationTest(Image img)
{
    // Suoritetaan diskreetti kosinimuunnos (vain kuvan Y-komponentti)
    DctMatrix dct = DCT(img);
    // Määritellään lohkokoko
    int blockWidth = 8, blockHeight = 8;
    // Lasketaan lohkojen lukumäärä vaaka- ja pystysuunnassa
    int blocksHorizontally = img.Width / blockWidth;
    int blocksVertically = img.Height / blockHeight;

    // Luodaan tulostaulukko
    double[,] result = new double[blocksHorizontally, blocksVertically];

    // Käydään läpi kaikki taajuudet dct-lohkon sisällä
    for (int y = 0; y < blockHeight; y++)
    {
        for (int x = 0; x < blockWidth; x++)
        {
            // Luodaan painokertoimien histogrammi taajudelle (x, y)
            int[] h = GetCoefficientHistogram(dct, x, y);

            // Arvioidaan kvantisaatioaskeleen pituus histogrammin perusteella
            int qStep = EstimateQuantizationStep(h);

            // Käydään läpi kaikki dct-lohkot
            for (int y2 = 0; y2 < blocksVertically; y2++)
            {
                for (int x2 = 0; x2 < blocksHorizontally; x2++)
                {
                    // Haetaan taajuutta (x, y) vastaava painokerroin lohkosta (x2, y2)
                    double currentCoef = dct.GetCoefficient(x2, y2, x, y);

                    // Lasketaan arvio nykyisen painokertoimen perusteella
                    double estimate = EstimateAuthenticity(currentCoef, h, qStep);

                    // Lisätään arvio tulostaulukkoon
                    result[x2, y2] += estimate;
                }
            }
        }
    }
    return result;
}

// Aliohjelma paluttaa diskreetin kosinimuunnoksen painokertoimien histogrammin.
// Parametrina annetaan kuvan kosinimuunnosta vastaava tyyppi sekä tarkasteltavan
```

```
// taajuuden koordinaatit.
int[] GetCoefficientHistogram(DctMatrix dct, int x, int y)
{
    // Haetaan kaikista dct-lohkoista kokonaisluvuksi pyöristetty taajuus
    // sijainnista (x, y)
    int[] coefValues = dct.GetCoefficientsAtPosition(x, y);

    Sort(coefValues);  // Järjestetään arvot
    int minValue = coefValues[0];  // Tallennetaan pienimmän painokertoimen arvo
    int[] quantities;  // Luodaan taulukko, johon kerrointen lukumäärätiedot tallennetaan
    int previousValue = coefValues[0];

    // Määritellään muuttuja, joka vastaa käsiteltävän painokertoimen esiintymien
    // lukumäärää
    int valueCount = 1;
    foreach (int i in coefValues)  // Käydään läpi kaikki painokertoimet
    {
        // Jos nykyinen arvo on sama kuin edellinen, lisätään tätä arvoa vastaavaa summaa
        if (i == previousValue) valueCount++;
        else
        {
            // Muutoin lasketaan nykyiselle painokertoimelle oikea indeksi
            int indx = i - minValue;
            quantities[indx] = valueCount;
            valueCount = 1;  // Asetetaan esiintymien lukumäärä yhteen
            previousValue = i; // Tallennetaan nykyinen arvo muuttujaan
        }
    }
    // Tallennetaan lopuksi viimeisimmät lukumäärätiedot
    if (valueCount > 0) quantities[previousValue - minValue] = valueCount;

    return quantities;
}


int EstimateQuantizationStep(histogram)
{
    // Suoritetaan Fourier-muunnos histogrammin arvoille
    complex[] ft = FourierTransform(histogram);

    // Muodostetaan Fourier-muunnoksen tehospektri
    double[] ps = GetPowerSpectrum(ft);

    // lasketaan huippujen lukumäärä
    int peaks = GetPeakCount(ps);

    // palautetaan huippujen lukumäärä lisättynä yhdellä
    return peaks + 1;
}


double EstimateAuthenticity(double coefficientValue, int[] h, int qStep)
{
    int lowerLimit = GetLowerLimit(h, period); // Etsitään jakson alaraja
    int upperLimit = GetUpperLimit(h, period); // Etsitään jakson yläraja

    // Etsitään histogrammin lokero, joka vastaa nykyisen painokertoimen arvoa
    int index = GetIndexForValue(h, coefficientValue);

    // Summataan histogrammin arvot tarkastelujakson alueelta
    double sum = 0.0;
    for (int i = lowerLimit; i < upperLimit; i++)
    {
        sum += h[i];
    }

    // Lasketaan todennäköisyys sille, että tarkasteltava painokerroin kuuluu
    // kuvan aitoon alueeseen
    double pUnchanged = h[index] / sum;
    double pTampered = 1.0 / period;

    return pUnchanged / (pTampered + pUnchanged);
}




/*** Kuvan kosinimuunnoksen matalien painokertoimien tunnistaminen (luku 4.3) ***/
```

```
// Aliohjelma laskee kuvan diskreetin kosinimuunnoksen matalien arvojen tunnistamiseen
//perustuvan arvion kuvan aitoudesta.
// Palautusarvona on kaksiulotteinen reaalilukutaulukko, jossa kukin solu vastaa
// yhdelle dct-lohkolle laskettua aitousarviota.
// Parametrina annetaan tarkasteltava kuva.
double[,] DetectLowValuedCoefficients(Image img)
{
    DctMatrix dct = DCT(image);
    int blockWidth  = 8, blockHeight = 8;
    int blocksHorizontally = img.Width / blockWidth;
    int blocksVertically = img.Height / blockHeight;
    double[,] result = new double[blocksHorizontally, blocksVertically];

    // Luodaan taulukko, johon tallennetaan painokertoimien arvioidut alarajat.
    int[,] coefficientLowerLimits;

    // Käydään läpi kaikki taajuudet dct-lohkon sisällä
    for (int y = 0; y < blockHeight; y++)
    {
       for (int x = 0; x < blockWidth; x++)
       {
            // Luodaan painokertoimien histogrammi taajudelle (x, y).
            // Histogrammi muodostetaan painokertoimien itseisarvoista.
            h = GetCoefficientHistogram(dct, x, y);

            // Arvioidaan painokertoimen alaraja histogrammin perusteella
            int lowerLimit = EstimateCoefficientLowerLimit(h);
            coefficientLowerLimits[x, y] = lowerLimit;
       }
    }

    // Käydään läpi kaikki dct-lohkot
    for (int y = 0; y < blocksVertically; y++)
    {
       for (int x = 0; x < blocksHorizontally; x++)
       {
            // Haetaan dct-lohko paikasta (x, y)
            double[,] dctBlock = dct.GetBlock(x, y);
            double sum = 0.0;

            // Käydään läpi kaikki taajuudet dct-lohkon sisällä
            for (int y2 = 0; y2 < blockHeight; y2++)
            {
               for (int x2 = 0; x2 < blockWidth; x2++)
               {
                   // Haetaan taajuutta (x2, y2) vastaava painokerroin lohkosta (x, y)
                   // ja muunnetaan se itseisarvoiseksi
                   int c = Abs(dct.GetCoefficient(x, y, x2, y2));
                   int limit = coefficientLowerLimits[x2, y2];

                   // Ohitetaan painokertoimet, jotka ovat arvoltaan pienempiä kuin 2
                   // tai suurempia kuin arvioitu alaraja
                   if (c > 2 && c < limit)
                   {
                       // Lasketaan käsiteltävän kertoimen etäisyys alarajasta ja
                       // nollasta ja lisätään näistä pienempi arvo tulostaulukkoon
                       result[x2, y2] += Min(limit - c, c);;
                   }
               }
            }
       }
    }
    return result;
}

// Aliohjelma palauttaa arvion painokertoimien alarajasta
// Aliohjelmalle annetaan parametrina kokonaislukutaulukko, joka vastaa kuvan
// kosinimuunnoksen painokertoimien histogrammia
int EstimateCoefficientLowerLimit(int[] h)
{
    int max = h[1];
    int maxIndex = 1;
    // Haetaan histogrammin toiseksi korkein huippu. Aloitetaan indeksistä 1, koska
    // indeksissä 0 sijaitsee yleensä korkein huippu
    for(int i = 1; i < h.Length; i++)
    {
        int tmp = h[i];
        // Verrataan nykyistä arvoa ja aiempaa suurinta arvoa
```

```
            // Jos nykyinen arvo on suurempi, sen arvo ja indeksi tallennetaan muuttujiin
            if (tmp > max)
            {
                max = tmp;
                maxIndex = i;
            }
        }
        // Palautetaan taulukon toiseksi suurimman arvon indeksi.
        return maxIndex;
}


/*** JPEG-kehikon erottaminen pikselitasolla (luku 4.4) ***/
// Aliohjelma paluttaa kaksiulotteisen kokonaislukutaulukon, joka vastaa kuvasta
//erotettua JPEG-kehikkoa.
// Aliohjelmalla annetaan parametrina tutkittava kuva.
int[,] ExtractBaGridPixel(Image img)
{
    // Erotetaan vaaka- ja pystysuuntaiset rajat kuvasta
    int[,] h = ExtractHorizontalEdges(img);
    int[,] v = ExtractVerticalEdges(img);

    // Yhdistetään naapuripisteiden arvot
    h = AccumulateColumns(h);

    // Vähennetään paikallinen mediaani pisteiden arvoista
    h = ReduceMedianHorizontal(h);

    // Suoritetaan mediaanisuodatus
    h = MedianFilterHorizontal(h);

    // Edelliset vaiheet suoritetaan pystysuuntaisille rajoille
    v = AccumulateRows(v);
    v = ReduceMedianVertical(v);
    v = MedianFilterVertical(v);

    // Yhdistetään vaaka- ja pystysuuntaiset kehikot
    int[,] result = Add(h, v);

    return result;
}


// Neljä seuraavaa aliohjelmaa on esitetty vain vaakasuuntaisten rajojen erottamisen
// osalta. Pystysuuntaisten rajojen erottaminen tapahtuu täysin vastaavalla tavalla.

// Aliohjelma paluttaa kaksiulotteisen kokonaislukutaulukon, joka vastaa kuvasta
// erotettuja vaakasuuntaisia rajoja.
// Aliohjelmalla annetaan parametrina tutkittava kuva (img).
int[,] ExtractHorizontalEdges(Image img)
{
    int[,] result = new int[img.Width, img.Height];
    int treshold = 50;
    for (int y = 1; y < img.Height - 1; y++)
    {
        for (int x = 0; x < img.Width; x++)
        {
            // Haetaan arvo pisteelle paikassa (x, y)
            // sekä sen ylä- ja alapuolisille pisteille
            int v1 = img.GetYValue(x, y - 1);
            int v2 = img.GetYValue(x, y);
            int v3 = img.GetYValue(x, y + 1);
            // Lasketaan naapuripisteeiden ero
            int difference = Abs(2 * v2 - v1 - v3);

            // Arvo nollataan, mikäli sen arvo ylittää määritellyn rajan
            if (difference > treshold) difference = 0;
            if (difference < 0) difference = 0;

            result[x, y] = difference;
        }

    }
    return result;
}


// Aliohjelma yhdistää kuvan naapuripisteiden arvot määritellyn toimintasäteen sisällä.
// Aliohjelmalle annetaan parametrina kaksiulotteinen kokonaislukutaulukko, joka vastaa
```

```
// kuvasta erotettuja vaakasuuntaisia rajoja.
int[,] AccumulateColumns(int[,] img)
{
    int[,] result = new int[img.Width, img.Height];
    int span = 16; // Määritellään tarkastelusäde
    for (int y = 0; y < img.Height; y++)
    {
        for (int x = 0; x < img.Width; x++)
        {
            int sum = 0;
            for (int x2 = x - span; x2 <= x + span; x2++)
            {
               // Jos ollaan kuvan rajojen sisällä, lisätään summaan pisteen (x2, y)
               // väriarvo
               if (x2 >= 0 && x2 < img.Height) sum += img[x2, y];
            }
            result[x, y] = sum;
        }
    }
    return result;
}

// Aliohjelma vähentää kuvan pisteiden arvoista paikallisen mediaanin.
// Aliohjelmalle annetaan parametrina kaksiulotteinen kokonaislukutaulukko, joka vastaa
// kuvasta erotettuja vaakasuuntaisia rajoja.
int[,] ReduceMedianHorizontal(int[,] img)
{
    int[,] result = new int[img.Width, img.Height];
    int span = 16;
    for (int y = span; y < img.Height - span; y++)
    {
         for (int x = 0; x < img.Width; x++)
         {
            // Kerätään tarkastelualueen arvot väliaikaiseen taulukkoon
            int[] values = new int[span * 2 + 1];
            int i = 0;
            for (int y2 = y - span; y2 <= y + span; y2++)
            {
               values[i] = img[x, y2];
               i++;
            }
            // Haetaan arvojoukon mediaani ja vähennetään se pisteen (x, y) väriarvosta
            double median = Median(values);
            int colorValue = img[x, y];
            colorValue -= median;
            if (colorValue < 0) colorValue = 0;
            result[x, y] = colorValue;
         }
    }
    return result;
}

// Aliohjelma suorittaa mediaanisuodatuksen kuvan pisteiden arvoille.
// Aliohjelmalle annetaan parametrina kaksiulotteinen kokonaislukutaulukko, joka vastaa
// kuvasta erotettuja vaakasuuntaisia rajoja.
int[,] MedianFilterHorizontal(int[,] img)
{
    int[,] result = new int[img.Width, img.Height];
    for (int y = 16; y < img.Height - 16; y++)
    {
         for (int x = 0; x < img.Width; x++)
         {
            // Haetaan mediaani määritelyjen kuvapisteiden arvojen joukosta ja asetetaan
            // se pisteen (x, y) arvoksi.
            double m = Median(img[x, y - 16], img[x, y - 8], img[x, y], img[x, y + 8],
            img[x, y + 16]);
            result[x, y] = m;
         }
    }
    return result;
}
```

```
// Aliohjelma paluttaa kaksiulotteisen kokonailukutaulukon, joka vastaa karttaa kuvan
// epäilyttävistä alueista.
```

```
// Aliohjelmalle annetaan parametrina kuva, joka vastaa kuvan JPEG-kehikkoa.
int[,] MarkSuspiciousBlocks(Image img)
{
    int blockWidth = 8, blockHeight = 8;
    int[,] result = new int[img.Width / blockWidth, img.Height / blockHeight];

    // Luodaan taulukot, joihin tallennetaan keskirivien ja keskisarakkeiden summat
    int[] midRowSums = new int[blockHeight - 2];
    int[] midColumnSums = new int[blockWidth - 2];
    // Määritellään muuttujat, joihin tallennetaan reunarivien ja -sarakkeiden summat
    int topRowSum = 0, leftColumnSum = 0, bottomRowSum = 0, rightColumnSum = 0;

    // Käydään läpi kaikki dct-lohkot
    for (int y = 0; y < img.Height - blockHeight; y += blockHeight)
    {
        for (int x = 0; x < cm.Width - blockWidth; x += blockWidth)
        {
            int sum = 0;
            for (int y2 = 1; y2 < blockHeight; y2++) // Vasen reunasarake
            {
               sum += img.GetGrayValue(x, y + y2));
            }
            leftColumnSum = sum;

            // ... Suoritetaan summaus lopuille reunasarakkeille ja -riveille vastaavaan
            // tapaan

            sum = 0;
            for (y2 = 1; y2 < blockHeight - 1; y2++)
            {
               for (x2 = 1; x2 < blockWidth - 1; x2++) // Keskirivit
               {
                    sum += img.GetGrayValue(x + x2, y + y2));
               }
            }

            // ... Suoritetaan summaus keskisarakkeille vastaavaan tapaan

            // Haetaan suurimmat keskisarakkeiden ja -rivien summat
            int midColumnMax = Max(midColumnSums);
            int midRowMax = Max(midRowSums);

            // Lasketaan erotukset suurimpien keskisarakkeiden ja pienimpien
            // reunasarakkeiden sekä suurimpien keskirivein ja pienimpien reunarivein
            // arvoista. Saadut erotukset lasketaan yhteen.
            int b = midColumnMax - Min(rightColumnSum, leftColumnSum) + midRowMax -
            Min(topRowSum, bottomRowSum);

            result[x, y] = b;
        }
    }
    return result;
}


/*** JPEG-kehikon erottaminen dct-painokertoimien perusteella (luku 4.5) ***/

// Aliohjelma palauttaa kaskiulotteisen reaalilukutaulukon, joka vastaa kuvasta
// erotettua JPEG-kehikkoa.
// Aliohjelmalle annetaan parametrina tarkasteltava kuva sekä mahdollisesti
// tarkasteltavaa aluetta rajaava kuva, eli maski.
double[,] ExtractBAGridDCT(Image img, Image mask)
{
    double[,] result = new double[img.Width, img.Height];

    // Käydään läpi kaikki kuvan pisteet, jotka ovat vähintään kahdeksan pikselin
    // etäisyydellä kuvan vasemmasta ja alareunasta.
    for (int y = 0; y < img.Height - blockHeight; y++)
    {
        for (int x = 0; x < img.Width - blockWidth; x++)
        {
            // Jos maskina käytetty kuva ei ole null-arvoinen, ja sen pisteessä (x, y)
            // on arvo 0, ohitetaan nykyinen piste ja siirrytään seuraavaan pisteeseen.
            if (mask != null && mask.GetGrayValue(x, y) == 0) continue;

            // Haetaan 8 x 8 pikselin lohko paikasta x, y
            double[,] pixelBlock = img.GetBlock(x, y, blockWidth, blockHeight);

            // Suoritetaan lohkolle diskreetti kosinimuunnos
```

```
            double[,] dctBlock = DCT(pixelBlock);
            double sum = 0.0;

            // Käydään läpi kaikki dct-lohkon taajuudet
            for (int y2 = 0; y2 < blockHeight; y2++)
            {
               for (int x2 = 0; x2 < blockWidth; x2++)
               {
                   // Jos jompi kumpi taajuuden sijaintia vastaava koordinaatti on 7
                   // lisätään vastaavan painokertoimen itseisarvo summaan
                   if (y2 == 7 || x2 == 7)
                   {
                       sum += Abs(dctBlock.GetCoefficient(x2, y2));
                   }
               }
            }
            // Tallennetaan summa pistelle (x, y)
            result[x, y] = sum;
        }
    }
    return result;
}

// Aliohjelma muuttaa matalien painokertoimien tunnistamiseen perustuvalla algoritmilla
// saavutettavan tuloskuvan mustavalkoiseksi ja laajentaa tulosaluetta halutun määrän.
// Parametrina annetaan tuloskuva, jossa kukin dct-lohko on merkitty tietyllä
// harmaasävyllä.
Image FilterMarkedBlockValues(Image img)
{
    int blockWidth  = 8, blockHeight = 8;

    // Määritellään tuloksen laajentamisessa käytettävä alue. Tässä alueen koko
    // on neljä lohkoa.
    int stretchArea = 4;

    Image result1 = new Image(img.Width, img.Height);
    Image result2 = new Image(img.Width, img.Height);

    // Lasketaan käsiteltävän kuvan kirkkauden keskiarvo.
    int mean = GetMean(img);

    // Luodaan ensin laajentamaton tuloskuva.
    // Käydään läpi kaikki dct-lohkot.
    for (int y = 0; y < img.Height / blockHeight; y++)
    {
        for (int x = 0; x < img.Width / blockWidth; x++)
        {
            // Haetaan kuvasta väriarvo lohkolle, joka alkaa pisteestä (x, y)
            int colorValue = img.GetGrayValue(x * blockWidth, y * blockHeight);
            int fillValue;
            if (colorValue > mean * 2.5) fillValue = 255;
            else fillValue = 0;

            // Täytetään tuloskuvasta dct-lohkon kokoinen alue alkaen pisteestä (x, y).
            // Täyttövärin määrää muuttuja fillValue.
            result1.FillArea(x, y, blockWidth, blockHeight, fillValue);
        }
    }

    // Suoritetaan tulosalueen laajennus.
    // Käydään läpi kaikki dct-lohkot.
    for (int y = 0; y < img.Height / blockHeight; y++)
    {
        for (int x = 0; x < img.Width / blockWidth; x++)
        {
            // Ohitetaan pisteet, joiden arvo on 0.
            if (result1.GetGrayValue(x, y) == 0) continue;

            //Määritellään täyttöalueen rajat huolehtien, että kuvan rajoja
            // ei ylitetä.
            int x1 = Max(0, x * blockWidth - (stretchArea * blockWidth));
            int x2 = Min(result1.Width - 1, x * blockWidth + (stretchArea *
            blockWidth));
            int y1 = Max(0, y * blockHeight - (stretchArea * blockHeight));
            int y2 = Min(result1.Height - 1, y * blockHeight + (stretchArea *
            blockHeight));

            // Täytetään lopullisesta tuloskuvasta alue, joka alkaa pisteestä (x1, y1).
            // Alueen koko määrittyy muuttujien x1, x2, y1 ja y2 perusteella.
```

```
            // Täyttöväri on 255, eli valkoinen (asetetaan kaikille värikanaville).
            result2.FillArea(x1, y1, x2 - x1, y2 - y1, 255);
        }
    }
    return result2;
}
```