Superresoluutio

Patrik Lod

26. huhtikuuta 2015

Pro gradu -tutkielma
Helsingin yliopisto
Matematiikan ja tilastotieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Matematiikan ja tilastotieteen laitos

Tekijä — Författare — Author
Patrik Lod

Työn nimi — Arbetets titel — Title
Superresoluutio

Oppiaine — Läroämne — Subject
Matematiikka

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	Maaliskuu 2015	44 s.

Tiivistelmä — Referat — Abstract

Digitaalisella kuvalla on resoluutio, joka määrittää sen koon. Resoluutiota voi suurentaa ja samalla parantaa kuvan laatua menetelmällä, jota kutsutaan superresoluutioksi. Superresoluutioalgoritmeja on useita ja ne toimivat eri tavalla. Tässä tutkielmassa käydään läpi kolmen eri superresoluutioalgoritmin toiminta läpi ja näytetään, miten nämä algoritmit toimivat kahdelle kuvalle sekä verrataan niitä keskenään.

Tutkielmassa ensiksi käydään läpi tarvittavaa taustateoriaa. Luku kaksi alkaa määrittelemällä digitaalisen kuvan ja kuinka tietokone käsittelee kuvia. Samalla esitellään kuviin liittyvää peruskäsitteistöä kuten (R,G,B)-väriavaruus. Seuraava kappale kertoo, miten värikuva muunnetaan (R,G,B)-väriavaruudesta harmaasävykuvaksi, koska kaikki tässä työssä esiteltävät algoritmit suoritetaan harmaasävykuville. Tämän jälkeen luvussa kaksi esitellään tässä työssä käytettävät lähdekuvat ja kuinka ne on otettu. Ensiksi esitellään, millä kameralla kuvat on otettu, missä formaatissa ja missä tiloissa. Sitten kerrotaan hieman kuvattavista kohteista ja esitellään valitut kuvat, joille algoritmit suoritetaan.

Luvussa kaksi esitellään kuvien jälkeen työssä käytettävät superresoluutioalgoritmit. Ensiksi esitellään kaksi yksinkertaisempaa menetelmää superresoluution saavuttamiseksi. Nämä molemmat ovat kaksiulotteisia interpolaatioita diskreetille datalle, joten niitä voi käyttää digitaalisille kuville. Ensiksi esitellään Bilineaarinen interpolaatio, joka yksinkertaisesti laskee painotetun keskiarvon interpoloitaville pikseleille lähimmästä neljästä pikselistä. Seuraavaksi esitellään bicubic-interpolaatio, joka on kaksiulotteinen jatke cubic-interpolaatiolle. Bicubic-interpolaatio toimii hyvin samankaltaisesti bilineaarisen interpolaation kanssa, mutta saa aikaan hieman tasaisempia tuloksia. Kolmantena algoritmina esitellään harvoihin esityksiin perustuva superresoluutioalgoritmi, joka hyödyntää opetettavia kirjastoja. Luvussa kaksi esitellään nopeasti, mitä ovat opetettavat kirjastot, harvat esitykset ja monet muut algoritmissa tarvittavat komponentit. Luvun kaksi lopussa esitetään SSIM samankaltaisuusmitta, jolla algoritmien tuloksia arvioidaan

Luvussa kolme esitellään algoritmeilla saadut tulokset luvussa kaksi esitetyille kuville. Luvussa kolme on myös kirjattu taulukkoon samankaltaisuusmitan antamat tulokset jokaiselle algoritmille. Luvussa neljä on analysoitu näitä tuloksia ja tehty johtopäätöksiä algoritmien paremmuusjärjestyksestä. Huomaamme, että perus interpolaatiot eivät yllä yhtä hyviin tuloksiin kuin viimeisenä esitelty algoritmi, jonka tulokset ovat yllättävän hyviä. Nopeudessa taas interpolaatiot, varsinkin bilineaarinen interpolaatio, todetaan olevan paljon nopeampia kuin harvoihin esityksiin perustuva superresoluutioalgoritmi. Huomioimme myös, mitä isommiksi kuvat suurennetaan sitä heikommin ne toimivat. Huomaamme myöskin sen, kuinka bilineaarinen interpolaatio toimi kaksinkertaisella suurennoksella paremmin kuin bicubic-interpolaatio, mutta nelinkertaisella suurennoksella taas bicubic-interpolaatio toimi paremmin näistä kahdesta.

Avainsanat — Nyckelord — Keywords
Superresoluutio, interpolaatio, kuvien suurentaminen, kuvankäsittely, resoluutio

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpulan tiedekirjasto

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

Luku 1

Johdanto

Superresoluutio on kuvankäsittelymenetelmä, jonka avulla pyritään kasvattamaan kuvan resoluutiota. Menetelmän tarkoitus on saada aikaiseksi suuriresoluutioisempi kuva, joka on myös tarkempi, eli kuva sisältää enemmän yksityiskohtia. Tavallisilla kuvankäsittelyohjelmilla kuvan koon suurentaminen tekee kuvasta suttuisen. Kuvan resoluutio kasvaa, mutta kuvan tarkkuus pysyy samana. Perinteiset superresoluutiomenetelmät hyödyntävät useampaa kuvaa, jotka on otettu hyvin tiheässä ajassa, esimerkiksi videokuvan peräkkäiset kuvat. Näiden peräkkäisten kuvien pienistä eroista on mahdollista saada tietoa valmistettavan suuriresoluutioisen kuvan yksityiskohdista, mitkä eivät näy pieniresoluutioisissa kuvissa. Menetelmät, jotka saavat suuriresoluutioisen kuvan aikaiseksi vain yhdestä pieniresoluutioisesta kuvasta käyttävät usein hyödyksi kirjastoa, jossa on useita pieniresoluutio-suuriresoluutio kuvapareja. Tätä kirjastoa hyödyntäen menetelmä yrittää löytää kuvasta samankaltaisia objekteja kun kirjaston pieniresoluutioisissa kuvissa ja miimikoida niiden suuriresoluutioisten kuvien yksityiskohtia. Haasteena on tietysti tietokoneen kyky hahmottaa kuvista yksityiskohtia, sillä tietokoneet käsittelevät kuvia eri tavoin kuin ihmiset.

Superresoluutiomenetelmistä on hyötyä tilanteissa, joissa kameran laatu ei ole kovin hyvä. Esimerkiksi suuri osa valvontakameroista, jotka ovat päällä päivittäin jatkuvasti tarvitsisivat suuren tallennustilan, jos ne kuvaisivat suurella resoluutiolla. Suuriresoluutioinen video tai valokuva vie enemmän tilaa kuin pieniresoluutioinen. Tämän takia valvontakamerat usein kuvaavat pieniresoluutioista kuvaa. Kun valvontakameroiden kuvista pitäisi nähdä jotain voidaan superresoluutiomenetelmillä suurentaa kuvan resoluutiota sekä kuvan tarkkuutta. Myöskin kännykät, jotka kuvaavat pienellä resoluutiolla, voivat kuvata riittämättömän kokoisia kuvia. Tällöin näitäkin kuvia voi suurentaa superresoluutiomenetelmillä.

Tässä pro gradu -tutkielmassa esitellään kolme menetelmää kuvien suurentamiseen sekä hiukan niitä varten tarvittavaa teoriaa ja vertaillaan näiden tuottamia tuloksia kes-

kenään. Luvussa kaksi esitellään aluksi, mille kuville tässä työssä sovelletaan superresoluutiomenetelmiä ja miten nämä kuvat ovat otettu. Samassa kappaleessa esitellään myös, miten tietokone hahmottaa kuvan ja esitellään kuviin liittyvää käsitteistöä kuten, mikä on pikseli, mitä tarkoittaa resoluutio tai miten pikselin väri määräytyy. Seuraavissa kappaleissa esitellään itse superresoluutioalgoritmit, joista kaksi ensimmäistä ovat superresoluutio aihealueen perusmenetelmiä ja kolmas on hiukan modernimpi algoritmi, joka käyttää hyväksi opetettua kirjastoa, joka myös esitellään nopeasti. Luvun kaksi lopussa esitellään algoritmi, jolla voidaan mitata kahden kuvan rakenteellista samankaltaisuutta ja miksi siitä on hyötyä kuvankäsittelyalgoritmien suoritusten arvioimisessa.

Luvussa kolme näytetään tulokset, jotka saimme käyttämällä luvussa kaksi esiteltyjä menetelmiä aiemmin esiteltyihin kuviin. Alkuperäisistä kuvista on tehty alinäytteistämällä pienennökset ja superresoluutioalgoritmeilla kuvat on palautettu alkuperäiseen kokoon, jonka jälkeen niiden onnistumista mitataan luvussa kaksi esitetyllä kuvien rakenteellista samankaltaisuutta mittaavalla algoritmilla.

Lopuksi luvussa neljä on esitetty johtopäätökset, mitä ollaan tehty rakenteellisen samankaltaisuusmitan tuloksista ja suurennetuista kuvista.

Luku 2

Aineistot ja menetelmät

2.1 Digitaalinen kuva

Digitaalinen kuva on kaksiulotteinen diskreetti kuvaus $I : \Omega \to \mathbb{R}$, missä $\Omega \in \mathbb{R}^2$. Kuvaus $I(m, n)$ tarkoittaa kuvan intensiteettiä kohdassa (m, n) $(m = 1, 2, \ldots, M; n = 1, 2, \ldots, N)$, eli harmaan sävyä mustavalkokuvassa ja väriä värikuvassa. Digitaalinen kuva on diskretisaatio jatkuvasta kaksiulotteisesta signaalista kameran sensorien avulla.

Tietokone käsittelee kuvaa $M \times N$-matriisina eli disktreettinä datana. Indeksit m ja n tarkoittavat kuvan rivejä ja sarakkeita. Piste (m, n) viittaa kuvan yhteen pikseliin eli kuvan pienimpään osaan, jolla on numeroarvo tai useampi numeroarvo.

Yksinkertaisimmassa tapauksessa jokaisella pikselillä on yksi numeroarvo, joka vastaa kuvan intensiteettiä siinä kohdassa. Näistä arvoista luodaan katseltava kuva värikarttojen avulla, mikä muuttaa numeroarvon sitä vastaavaksi väriksi. Mustavalkokuvissa pikselin numeroarvo vastaa harmaan sävyä täysin mustasta (arvo nolla) valkoiseen (maksimiarvo).

Värikuvissa yhden numeroarvon sijasta joka pikselissä onkin kolme väriarvoa. Useimmiten värikuvissa käytetään (R,G,B) värikarttaa, jossa kuvan jokaista pikseliä vastaa kolme eri väriarvoa: yksi punaista, yksi vihreää ja yksi sinistä kohden. Tällöin kuva tallentuu kolmiulotteisena matriisina, jonka voi mieltää kolmena kaksiulotteisena matriisina, jossa jokainen taso vastaa yhtä värikanavan väriä.

Kuva voi olla myös kolmiulotteinen, esimerkiksi lääketieteessä voidaan luoda rekonstruktio sisäelimestä kolmiulotteisena kuvana. Myöskin voidaan ajatella aika neljänneksi ulottuvuudeksi, jolloin voidaan kuvata vaikka sykkivää sydäntä. Tällöin kuva olisi neliulotteinen matriisi, jossa kolme koordinaattiakselia vastaisi avaruudellisia dimensioita ja neljäs aikaa. Tässä työssä tarkastellaan kuvia vain kaksiulotteisessa tapauksessa.

Tämän työn kannalta hyvin tärkeä kuvaa määrittelevä ominaisuus on resoluutio. Resoluutio kertoo kuvan tarkkuuden ja se voidaan määritellä koostuvan kolmesta osasta: avaruudellisesta resoluutiosta, ajallisesta resoluutiosta ja bitti resoluutiosta.

Avaruudellisella resoluutiolla kaksiulotteisissa kuvissa tarkoitetaan rivien (M) kertaa sarakkeiden (N) määrän pikselien muodostamaa kuvamatriisia. Avaruudellinen resoluutio tai pikseliresoluutio kertoo, kuinka moneen pikseliin jatkuvan kuvan diskretisaatio tehdään. Resoluutio ilmoitetaan usein $N \times M$ muodossa, esimerkiksi HD-kanavan videokuvassa yhden kuvan resoluutio on 1920×1080.

Ajallisella resoluutiolla tarkoitetaan videokuvissa käytettävää kuvataajuutta, eli kuinka monta kuvaa näytetään sekunnissa. Mitä korkeampaa kuvataajuutta käytetään, sitä useamman kuvan video sisältää ja sitä enemmän tallennustilaa video tarvitsee. Televisiolähetyksissä kuvataajuus on 25 kuvaa sekunnissa.

Bittiresoluutiolla tarkoitetaan sitä, kuinka paljon tallennustilaa varataan yhdelle pikselille kuvassa. Tämä viittaa suoraan siihen, kuinka monta eri sävyä kuvassa voidaan esittää. Binäärisessä kuvassa (2-bittinen kuva) on vain kaksi väriä: musta ja valkoinen. Harmaasävykuva eli puhekielessä mustavalkokuva yleensä kykenee esittämään 256 harmaan eri sävyä eli kyseessä on tällöin 8-bittinen kuva. Värikuvat usein ovat 24-bittisiä kuvia.

Tässä työssä resoluutiosta puhuttaessa tarkoitetaan avaruudellista resoluutiota.

2.2 Värikuvan muunnos harmaasävykuvaksi

Tässä tutkielmassa käytettävät superresoluutio-algoritmit suoritetaan harmaasävykuville, mutta kaikki lähdekuvat on kuvattu värikuvina, joten ne täytyy muuntaa harmaasävykuviksi. Harmaasävykuvien käyttäminen kuvankäsittelyalgoritmien analysoimisessa on yleistä, sillä muunnos harmaasävykuvaksi yksinkertaistaa eli vähentää informaatiota kuvassa. Vaikka harmaasävykuvassa on vähemmän informaatiota kuin värikuvassa, niin se sisältää suurimman osan oleellisesta informaatiosta kuten reunat, alueet, muodot, hahmot ja niin edelleen. Vähemmän informaation ansiosta algoritmien toiminta on nopeampaa harmaasävykuville kuin värikuville.

(R,G,B) värikuvan muuttaminen harmaasävykuvaksi onnistuu muunnoksella

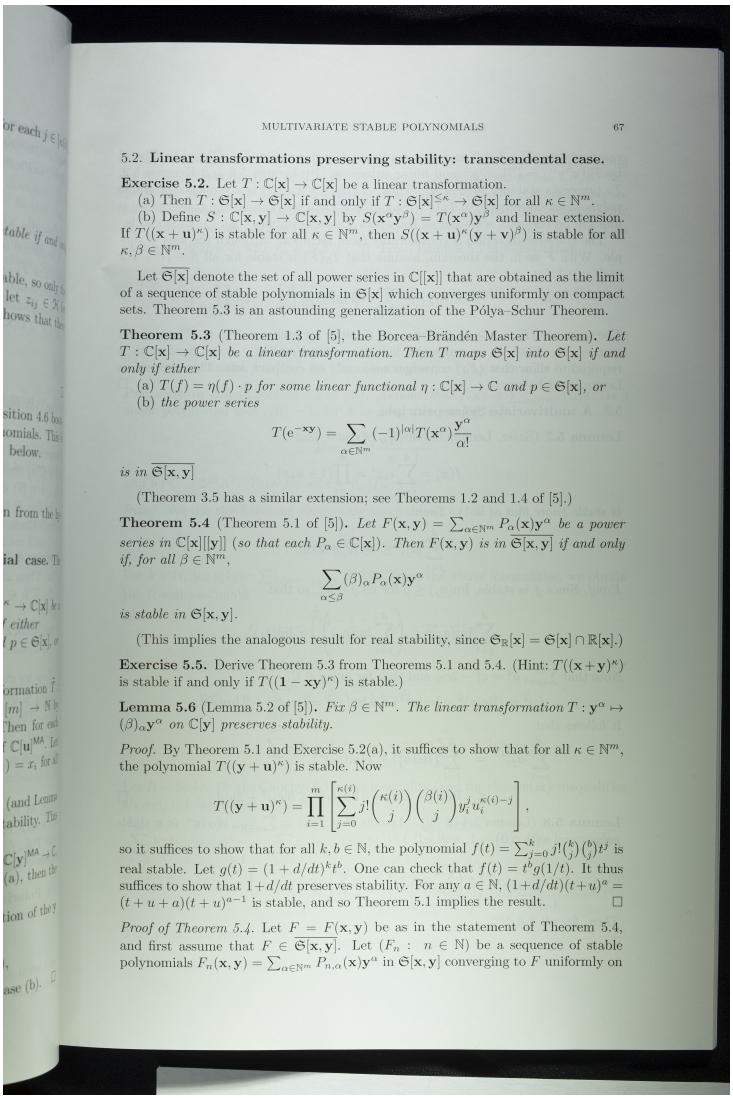
$$I_h(n,m) = \alpha I_v(n,m,r) + \beta I_v(n,m,g) + \gamma I_v(n,m,b), \tag{2.1}$$

missä indeksi (n,m) osoittaa pikselin sijainnin harmaasävykuvassa ja indeksi (n,m,c), missä $c \in \{r,g,b\}$ ja r,g ja b tarkoittavat punaista, vihreää ja sinistä väriä, osoittaa yhden värikanavan värikuvan pikselisijainnissa (n,m). Kaavasta (2.1) huomaa, että muunnos on vain painotettu summa värikanavien arvoista. Kertoimet α, β ja γ on valittu vastaamaan ihmissilmän herkkyyttä punaiselle, vihreälle ja siniselle värille. Ihmissilmä on herkin vihreälle värille ja herkempi punaiselle kuin siniselle värille, joten kertoimet on valittu sen mukaan, että harmaasävykuvan sävyjen suhteellinen intensiteetti vastaisi värikuvan värien intensiteettiä. Yleisesti käytetään arvoja $\alpha = 0,2989, \beta = 0,5870$, ja $\gamma = 0,1140$.

2.3 Tutkielmassa käytettävät kuvat

Tässä osiossa esitellään työssä käytettävät lähdekuvat. Kuviksi on valittu osa tekstiä, joka näkyy kuvassa 2.1 ja kuva sammakosta, joka näkyy kuvassa 2.2, joka on tulostettu 3D-tulostimella. Molemmat kuvat on otettu Canonin EOS 5D mark II kameralla ja linssinä on käytetty makro-planar 2/50 ZE:tä. Kuvat on otettu Helsingin yliopiston Kumpulan kampuksen tiloissa, Matematiikan ja Tilastotieteen laitoksen laboratoriotiloissa. Kuvauslaboratoriossa kuvat otettiin esineistä, jotka sijoitettiin pöydälle valkoisen paperin päälle ja tätä pöytää valaistiin kahdella valaisimella pöydän molemmin puolin. Otetut kuvat ovat kooltaan 3744×5616 pikselin kokoisia värikuvia. Kuvat ovat 8-bittisiä tif-formaatissa olevia otoksia.

Ensimmäinen kuva on otos matematiikan kirjasta, joka sisältää tekstiä sekä useita erityylisiä kaavoja. Tekstiä kuvatessa on käytetty 1/60 sekunnin valotusaikaa, F80 aukkoa, 50 mm polttoväliä sekä iso 100 herkkyyttä. Kuva tekstistä sisältää paljon teräviä reunoja ja isoja kontrastinvaihdoksia ja siitä näkee, miten superresoluutioalgoritmit vaikuttavat teräviin reunoihin. Teräväreunaisuudella kuvankäsittelyssä tarkoitetaan sitä, että kuvassa näkyvät reunat ilmenee tietokoneelle kuvan matriisissa suurena erona vierekkäisten kuvapikselien arvoina. Varsinkin tekstissä tämä korostuu, sillä valkoisen taustan arvo on 255, mustan tekstin arvo on 0 ja useat kirjaimet koostuvat suorista viivoista.



MULTIVARIATE STABLE POLYNOMIALS 67

5.2. **Linear transformations preserving stability: transcendental case.**

Exercise 5.2. Let $T : \mathbb{C}[\mathbf{x}] \to \mathbb{C}[\mathbf{x}]$ be a linear transformation.
(a) Then $T : \mathfrak{S}[\mathbf{x}] \to \mathfrak{S}[\mathbf{x}]$ if and only if $T : \mathfrak{S}[\mathbf{x}]^{\leq \kappa} \to \mathfrak{S}[\mathbf{x}]$ for all $\kappa \in \mathbb{N}^m$.
(b) Define $S : \mathbb{C}[\mathbf{x}, \mathbf{y}] \to \mathbb{C}[\mathbf{x}, \mathbf{y}]$ by $S(\mathbf{x}^\alpha \mathbf{y}^\beta) = T(\mathbf{x}^\alpha)\mathbf{y}^\beta$ and linear extension. If $T((\mathbf{x}+\mathbf{u})^\kappa)$ is stable for all $\kappa \in \mathbb{N}^m$, then $S((\mathbf{x}+\mathbf{u})^\kappa(\mathbf{y}+\mathbf{v})^\beta)$ is stable for all $\kappa, \beta \in \mathbb{N}^m$.

Let $\overline{\mathfrak{S}[\mathbf{x}]}$ denote the set of all power series in $\mathbb{C}[[\mathbf{x}]]$ that are obtained as the limit of a sequence of stable polynomials in $\mathfrak{S}[\mathbf{x}]$ which converges uniformly on compact sets. Theorem 5.3 is an astounding generalization of the Pólya–Schur Theorem.

Theorem 5.3 (Theorem 1.3 of [5], the Borcea–Brändén Master Theorem)**.** *Let $T : \mathbb{C}[\mathbf{x}] \to \mathbb{C}[\mathbf{x}]$ be a linear transformation. Then T maps $\mathfrak{S}[\mathbf{x}]$ into $\mathfrak{S}[\mathbf{x}]$ if and only if either*
(a) *$T(f) = \eta(f) \cdot p$ for some linear functional $\eta : \mathbb{C}[\mathbf{x}] \to \mathbb{C}$ and $p \in \mathfrak{S}[\mathbf{x}]$, or*
(b) *the power series*

$$T(\mathrm{e}^{-\mathbf{xy}}) = \sum_{\alpha \in \mathbb{N}^m} (-1)^{|\alpha|} T(\mathbf{x}^\alpha) \frac{\mathbf{y}^\alpha}{\alpha!}$$

is in $\overline{\mathfrak{S}[\mathbf{x}, \mathbf{y}]}$

(Theorem 3.5 has a similar extension; see Theorems 1.2 and 1.4 of [5].)

Theorem 5.4 (Theorem 5.1 of [5])**.** *Let $F(\mathbf{x}, \mathbf{y}) = \sum_{\alpha \in \mathbb{N}^m} P_\alpha(\mathbf{x})\mathbf{y}^\alpha$ be a power series in $\mathbb{C}[\mathbf{x}][[\mathbf{y}]]$ (so that each $P_\alpha \in \mathbb{C}[\mathbf{x}]$). Then $F(\mathbf{x}, \mathbf{y})$ is in $\overline{\mathfrak{S}[\mathbf{x}, \mathbf{y}]}$ if and only if, for all $\beta \in \mathbb{N}^m$,*

$$\sum_{\alpha \leq \beta} (\beta)_\alpha P_\alpha(\mathbf{x})\mathbf{y}^\alpha$$

is stable in $\mathfrak{S}[\mathbf{x}, \mathbf{y}]$.

(This implies the analogous result for real stability, since $\mathfrak{S}_{\mathbb{R}}[\mathbf{x}] = \mathfrak{S}[\mathbf{x}] \cap \mathbb{R}[\mathbf{x}]$.)

Exercise 5.5. Derive Theorem 5.3 from Theorems 5.1 and 5.4. (Hint: $T((\mathbf{x}+\mathbf{y})^\kappa)$ is stable if and only if $T((\mathbf{1}-\mathbf{xy})^\kappa)$ is stable.)

Lemma 5.6 (Lemma 5.2 of [5])**.** *Fix $\beta \in \mathbb{N}^m$. The linear transformation $T : \mathbf{y}^\alpha \mapsto (\beta)_\alpha \mathbf{y}^\alpha$ on $\mathbb{C}[\mathbf{y}]$ preserves stability.*

Proof. By Theorem 5.1 and Exercise 5.2(a), it suffices to show that for all $\kappa \in \mathbb{N}^m$, the polynomial $T((\mathbf{y}+\mathbf{u})^\kappa)$ is stable. Now

$$T((\mathbf{y}+\mathbf{u})^\kappa) = \prod_{i=1}^{m} \left[\sum_{j=0}^{\kappa(i)} j! \binom{\kappa(i)}{j} \binom{\beta(i)}{j} y_i^j u_i^{\kappa(i)-j} \right],$$

so it suffices to show that for all $k, b \in \mathbb{N}$, the polynomial $f(t) = \sum_{j=0}^{k} j! \binom{k}{j} \binom{b}{j} t^j$ is real stable. Let $g(t) = (1 + d/dt)^k t^b$. One can check that $f(t) = t^b g(1/t)$. It thus suffices to show that $1 + d/dt$ preserves stability. For any $a \in \mathbb{N}$, $(1+d/dt)(t+u)^a = (t+u+a)(t+u)^{a-1}$ is stable, and so Theorem 5.1 implies the result. □

Proof of Theorem 5.4. Let $F = F(\mathbf{x}, \mathbf{y})$ be as in the statement of Theorem 5.4, and first assume that $F \in \overline{\mathfrak{S}[\mathbf{x}, \mathbf{y}]}$. Let $(F_n : n \in \mathbb{N})$ be a sequence of stable polynomials $F_n(\mathbf{x}, \mathbf{y}) = \sum_{\alpha \in \mathbb{N}^m} P_{n,\alpha}(\mathbf{x})\mathbf{y}^\alpha$ in $\mathfrak{S}[\mathbf{x}, \mathbf{y}]$ converging to F uniformly on

Kuva 2.1

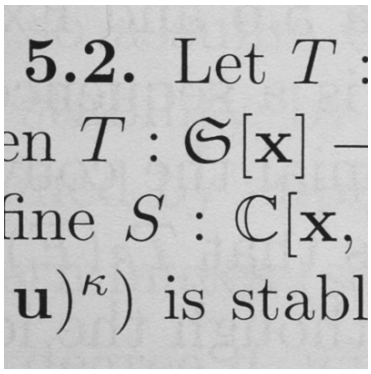
Toiseksi kuvaksi on valittu otos sammakoista, koska kuvan sammakot sisältävät pieniä yksityiskohtia, jotka ovat seurausta 3D-tulostuksesta, joka tulostaa esineet kerroksittain. Kuva sammakoista on otettu 1/60 sekunnin valotusajalla iso 100 herkkyydellä. Kuvassa on erikokoisia ja erivärisiä sammakoita. Sammakot on aseteltuna paperiarkille, josta erottuu hyvin niiden langettamat varjot.

Molemmista kuvista on valittu pieni osakuva, jolle algoritmit suoritetaan. Tähän on kaksi syytä. Ensinnäkin algoritmien suoritus on nopeampaa pienemmille kuville ja toiseksi superresoluutiomenetelmillä suurennettuja kuvia on helpompi verrata pienillä kuvilla,

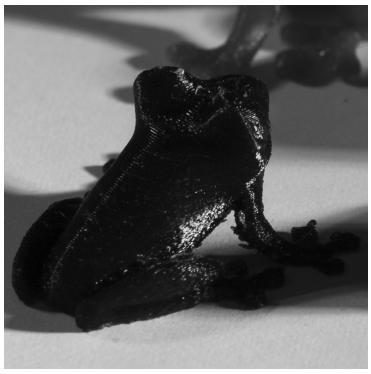
Kuva 2.2

joista näkee helpommin pienemmät eroavaisuudet. Lisäksi nämä osakuvat on muunnettu mustavalkokuviksi algoritmeja varten kaavalla (2.1). Tekstistä on valittu osakuvaksi osio, josta löytyy paljon vaihtelua. Osakuva sisältää pieniä sekä isoja kirjaimia, mutta myös erikoismerkkejä, kaunokirjaimia ja lihavoituja kirjaimia. Osakuvan 2.3a koko on 400x400 pikseliä ja se on tallennettu png-muodossa.

Osakuvaksi toisesta kuvasta, eli kuvasta sammakoista, on valittu vasemmanpuoleisin selin kameraan päin oleva musta sammakko. Kyseisestä sammakosta erottuu paljon yksityiskohtia ja koska se on kuvan etualalla, johon kamera kohdistettiin, niin osakuva on hyvin terävä. Sammakko mahtui parhaiten 800x800 pikseliin, joten se valittiin osakuvan 2.3b kooksi. Sammakko-osakuva tallennettiin png-kuvamuodossa.



(a)



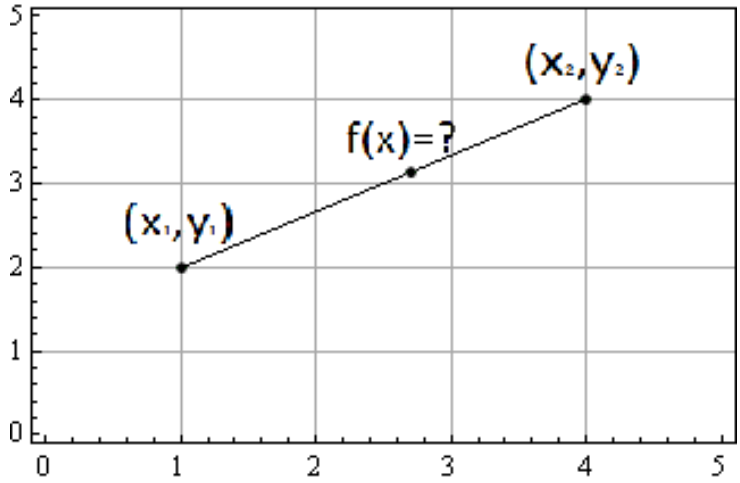
(b)

Kuva 2.3: Vasemmalla osakuva tekstistä (a) ja oikealla osakuva 3D-tulostetusta sammakosta (b)

2.4 Bilineaarinen interpolaatio

Bilineaarinen interpolaatio on lineaarisen interpolaation kasiulotteinen laajennos tasoon. Lineaarisella interpolaatiolla saamme suoran funktion kahden tiedetyn pisteen välille. Jos tiedämme pisteet $(x_1, y_1) \in \mathbb{R}^2$ ja $(x_2, y_2) \in \mathbb{R}^2$, niin voimme ratkaista kaikilla $x \in [x_1, x_2]$ arvon $y = f(x)$. Tämä ratkaisu $(x, f(x))$ sijaitsee pisteiden (x_1, y_1) ja (x_2, y_2) välisellä suoralla. Ratkaisemme muuttujaa x vastaava y arvo yhtälöstä

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}.$$



Kuva 2.4: Kuvassa bilineaarinen interpolaatio kahden pisteen välille.

Tämä onnistuu helposti kertomalla yhtälö vasemmanpuoleisemmalla jakajalla ja lisäämällä y_1. Tällöin ratkaisuksi saadaan

$$y = y_1 + (y_2 - y_1)\frac{x - x_1}{x_2 - x_1}.$$

Merkitään pistettä $y_1 = f(x_1)$ ja pistettä $y_2 = f(x_2)$, jolloin suoran yhtälö voidaan kirjoittaa

$$f(x) = f(x_1) + f(x_2)\frac{x - x_1}{x_2 - x_1} - f(x_1)\frac{x - x_1}{x_2 - x_1}$$

ja tehdään muuttujan vaihto $t = \frac{x - x_1}{x_2 - x_1}$, jolloin

$$f(t) = tf(x_2) + (1 - t)f(x_2)$$

antaa painotetun arvon suoralta muuttujan arvoilla $t \in [0, 1]$. Esimerkiksi $t = 0.5$ antaa arvon keskeltä suoraa, joka kulkee y_1:stä y_2:seen.

Bilineaarinen interpolaatio toimii tasossa siten, että se hyväksikäyttää lineaarista interpolaatiota vaakasuoraan ja pystysuoraan laskeakseen kaikki arvot neljän pisteen muodostaman neliön välille. Olkoon neljä pistettä $A_{11} = (x_1, y_1)$, $A_{12} = (x_1, y_2)$, $A_{21} = (x_2, y_1)$ ja $A_{22} = (x_2, y_2)$ joiden arvot tiedämme. Haluamme ratkaista pisteen $P = (x, y)$ arvon, kun $x \in [x_1, x_2]$ ja $y \in [y_1, y_2]$. Laskemme ensiksi lineaarisen interpolaation x-akselin suuntaisesti pisteiden A_{11} ja A_{21} välille sekä pisteiden A_{12} ja A_{22} välille. Merkitään $B_1 = (x, y_1)$ ja $B_2 = (x, y2)$. Tällöin

$$f(B_1) = \frac{x_2 - x}{x_2 - x_1}f(A_{11}) + \frac{x - x_1}{x_2 - x_1}f(A_{21})$$

ja

$$f(B_2) = \frac{x_2 - x}{x_2 - x_1} f(A_{12}) + \frac{x - x_1}{x_2 - x_1} f(A_{22}).$$

Nyt laskemme lineaarisen interpolaation y-akselin suuntaisesti pisteiden B_1 ja B_2 välille. Tällöin saamme pisteen (x, y) arvoksi

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(B_1) + \frac{y - y_1}{y_2 - y_1} f(B_2).$$

Tästä voimme johtaa kaavan

$$\begin{aligned} f(x,y) &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(A_{11}) + \frac{x - x_1}{x_2 - x_1} f(A_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(A_{12}) + \frac{x - x_1}{x_2 - x_1} f(A_{22}) \right) \\ &= \frac{1}{(x_2 - x_1)(y_2 - y_1)} [(y_2 - y)((x_2 - x) f(A_{11}) + (x - x_1) f(A_{21})) \\ &\quad + (y - y_1)((x_2 - x) f(A_{12}) + (x - x_1) f(A_{22}))], \end{aligned}$$

joka ratkaisee mille tahansa pisteelle arvon, joka sijaitsee neljän pisteen muodostaman nelikulmio sisällä. On hyvä huomata, että samaan tulokseen pääsee aluksi interpoloimalla y-akselin suuntaan ja sitten vasta x-akselin suuntaan.

2.5 Bicubic interpolaatio

2.5.1 Cubic interpolaatio

Bicubic interpolaatiota varten on hyvä käydä läpi, miten cubic interpolaatio toimii. Tehdään interpolointi kuten lähteessä [2]. Cubic interpolaatiolla sovitetaan jatkuva funktio datajoukkoon, joka sisältää pistepareja $(x, f(x))$. Kun lineaarinen interpolaatio sijoitti pisteparit suoralle, niin cubic splini sovittaa pisteparit kolmannen asteen yhtälöön

$$y = f(x) = ax^3 + bx^2 + cx + d. \tag{2.2}$$

Kahden pisteen väliin sijoitus on hyvin helppo. Meidän tarvitsee vain tietää arvot reunoilla, eli kun $x = 0$ ja $x = 1$. Lasketaan ensiksi funktion (2.2) derivaatta, joka on

$$f'(x) = 3ax^2 + 2bx + c.$$

Seuraavaksi lasketaan funktion ja sen derivaatan arvot pisteissä $x = 0$ ja $x = 1$:

$$\begin{aligned} f(0) &= d \\ f(1) &= a + b + c + d \\ f'(0) &= c \\ f'(1) &= 3a + 2b + c. \end{aligned} \tag{2.3}$$

Tästä yhtälöryhmästä ratkaistaan kertoimet a, b, c ja d. Tällöin yhtälöryhmän voi kirjoitaa muodossa

$$\begin{aligned} a &= 2f(0) - 2f(1) + f'(0) + f'(1) \\ b &= -3f(0) + 3f(1) - 2f'(0) - f'(1) \\ c &= f'(0) \\ d &= f(0). \end{aligned}$$

Nyt sijoittamalla yllä lasketut vakiot a, b, c ja d yhtälöön (2.2) voimme interpoloida kaikki arvot pisteiden $x = 0$ ja $x = 1$ väliltä.

Koska haluamme interpoloida useamman pisteen väliin käyrän, niin emme tiedä funktion derivaattaa tiedettyjen pisteiden kohdalla. Yksi vaihtoehto olisi käyttää derivaatan arvona nollaa joka pisteessä, mutta käyttämällä derivaattana suoraa edellisen pisteen ja seuraavan pisteen välillä saamme aikaan paljon sileämpiä käyriä. Tätä kutsutaan *Catmull-Rom spliniksi*. Oletetaan siis, että tiedämme neljän pisteen arvot y_0, y_1, y_2 ja y_3 kohdissa $x = -1$, $x = 0$, $x = 1$ ja $x = 2$. Nyt määritellään uudella tavalla funktion ja sen derivaatan arvot pisteissä $x = 0$ ja $x = 1$. Tällöin uusiksi arvoiksi saamme

$$\begin{aligned} f(0) &= y_1 \\ f(1) &= y_2 \\ f'(0) &= \frac{y_2 - y_0}{2} \\ f'(1) &= \frac{y_3 - y_1}{2}. \end{aligned}$$

Sijoittamalla nämä arvot yhtälöryhmään (2.3) ja ratkaisemalla vakiot saamme

$$\begin{aligned} a &= -\tfrac{1}{2}y_0 + \tfrac{3}{2}y_1 - \tfrac{3}{2}y_2 + \tfrac{1}{2}y_3 \\ b &= y_0 - \tfrac{5}{2}y_1 + 2y_2 - \tfrac{1}{2}y_3 \\ c &= -\tfrac{1}{2}y_0 + \tfrac{1}{2}y_2 \\ d &= y_1. \end{aligned}$$

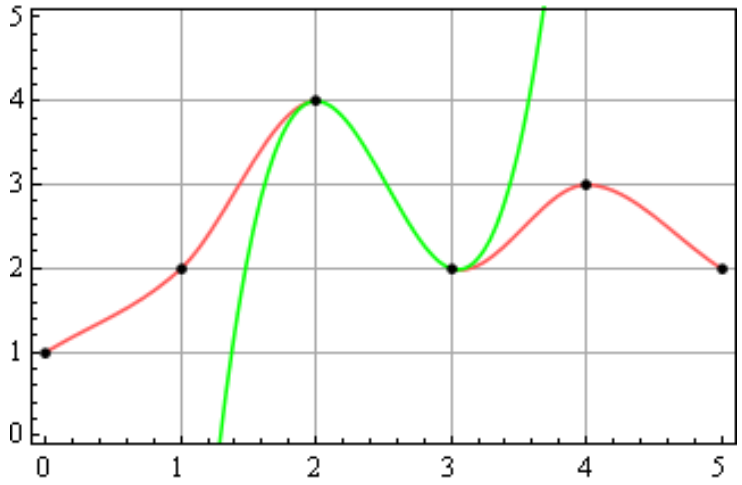
Lopuksi sijoitamme ratkaistut vakiot yhtälöön (2.2) ja saamme interpolaatiofunktioksi

$$f(y_0, y_1, y_2, y_3, x) = (-\tfrac{1}{2}y_0 + \tfrac{3}{2}y_1 - \tfrac{3}{2}y_2 + \tfrac{1}{2}y_3)x^3 + (y_0 - \tfrac{5}{2}y_1 + 2y_2 - \tfrac{1}{2}y_3)x^2 + (-\tfrac{1}{2}y_0 + \tfrac{1}{2}y_2)x + y_1,$$

jolla voimme interpoloida välin $x \in [0, 1]$ kaikki arvot. Sijoittamalla funktioon neljä vierekkäistä pistettä ja liikkumalla yksi piste kerrallaan voimme interpoloida pistejoukon kaikkien pisteiden välit.

Ensimmäisen ja viimeisen alkion kohdalla täytyy huomioda, miten valitsee edellisen tai seuraavan pisteen arvon. Yksi tapa on valita reunoilla ylimeneväksi arvoksi sama arvo kuin reunapisteessä.

Kuvasta 2.5 näkee, vaikka emme tiedäkään derivaattaa pistejoukkoa interpoloidessamme, niin silti jokaisella kahden pisteen välillä saatu interpolaatio on identtinen kahden pisteen cubic interpolaation kanssa, jolloin tiedettiin derivaatta.



Kuva 2.5: Kuvassa vihreällä cubic interpolaatio kahdelle pisteelle ja punaisella viiden pisteen joukolle [2].

2.5.2 Bicubic interpolaatio

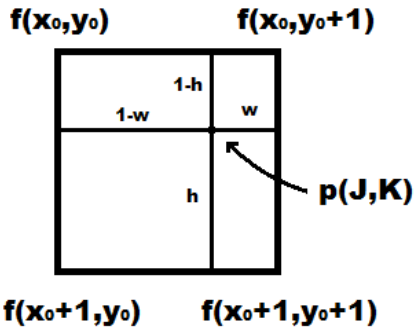
Bicubic interpolaatio on cubic interpolaation laajennos toiseen ulottuvuuteen [3]. Tässä käyttämällä 16 tiedettyä pistettä voimme interpoloida kaikki arvot neljän sisimmän pisteen muodostaman neliön sisältä. Bicubic interpolaatio tarvitsee neljä keskimmäistä pistettä $f(x_0, y_0)$, $f(x_0 + 1, y_0)$, $f(x_0, y_0 + 1)$, $f(x_0 + 1, y_0 + 1)$ ja osittaisderivaatat $\frac{\partial f}{\partial x} = f_x$ ja $\frac{\partial f}{\partial y} = f_y$ sekä poikittaisderivaatta $\frac{\partial^2 f}{\partial y \partial x} = f_{xy}$. Osittaisderivaatat lasketaan suorina edellisestä ja seuraavasta pisteestä, joten derivaatta ei ole tarkka arvo. Sen ei tarvitsekaan olla tarkka. Mitä paremman arvion derivaatalle laskee sitä tarkempi interpolaatiosta tulee, mutta interpoloidusta käyrästä tulee aina sileä [4].

Lasketaan osittaisderivaatat pisteelle (x_0, y_0), ensiksi x-akselin suuntaan, jolloin

$$f_x(x_0, y_0) = \frac{f(x_0 + 1, y_0) - f(x_0 - 1, y_0)}{(x_0 + 1) - (x_0 - 1)} = \frac{f(x_0 + 1, y_0) - f(x_0 - 1, y_0)}{2}.$$

Samalla kaavalla laskettuna y-akselin suuntaan saadaan y:n suhteen lasketuksi osittaisderivaataksi

$$f_y(x_0, y_0) = \frac{f(x_0, y_0 + 1) - f(x_0, y_0 - 1)}{(y_0 + 1) - (y_0 - 1)} = \frac{f(x_0, y_0 + 1) - f(x_0, y_0 - 1)}{2}.$$



Kuva 2.6: Interpoloitava piste $p(J, K)$ alkuperäisen neljän pisteen sisältä.

Lopuksi laskemme poikittaisderivaatan

$$\begin{aligned} f_{xy}(x_0, y_0) &= \frac{[f(x_0 - 1, y_0 - 1) + f(x_0 + 1, y_0 + 1)] - [f(x_0 + 1, y_0 - 1) + f(x_0 - 1, y_0 + 1)]}{[(x_0 + 1) - (x_0 - 1)] * [(y_0 + 1) - (y_0 - 1)]} \\ &= \frac{[f(x_0 - 1, y_0 - 1) + f(x_0 + 1, y_0 + 1)] - [f(x_0 + 1, y_0 - 1) + f(x_0 - 1, y_0 + 1)]}{4}. \end{aligned}$$

Nämä osittaisderivaatat lasketaan samalla tavalla muillekin kulmapisteille. Näiden arvojen avulla voimme määritellä interpoloidun arvon mille tahansa pisteelle $p(x, y)$, joka sijaitsee valitun neljän pisteen muodostaman neliön sisällä. Pisteen arvo ja sen osittaisderivaatat ovat kirjan [4] mukaan:

$$\begin{aligned} p(J, K) &= \sum_{m=0}^{3} \sum_{n=0}^{3} a_{mn}(1 - w)^m (1 - h)^n \\ p_x(J, K) &= \sum_{m=0}^{3} \sum_{n=0}^{3} a_{mn}(1 - w)^{m-1} (1 - h)^n \\ p_y(J, K) &= \sum_{m=0}^{3} \sum_{n=0}^{3} a_{mn}(1 - w)^m (1 - h)^{n-1} \\ p_{xy}(J, K) &= \sum_{m=0}^{3} \sum_{n=0}^{3} a_{mn}(1 - w)^{m-1} (1 - h)^{n-1}. \end{aligned} \tag{2.4}$$

Tarvitsemme interpoloitavan pisteen osittaisderivaattoja siihen, että voimme ratkaista kertoimet a_{mn}.

Ratkaistaan kertoimet a_{mn} käyttämällä tiedettyjä pisteitä. Oletetaan, että w ja h ovat jatkuvia välillä $[0, 1]$ ja merkitään kulmapisteitä $f(x_0, y_0)$, $f(x_0 + 1, y_0)$, $f(x_0, y_0 + 1)$ ja

$f(x_0+1, y_0+1)$ yksikköneliön kulmapisteinä $f(0,0)$, $f(1,0)$, $f(0,1)$ ja $f(1,1)$. Nyt koska $p(x,y) = f(x,y)$ kulmapisteissä voimme laskea yhtälöiden (2.4) arvot neljässä kulmapisteessä, jolloin me saamme 16 yhtälöä. Saamme neljä yhtälöä funktion arvoille

$$\begin{aligned}
f(0,0) &= a_{00} \\
f(1,0) &= a_{00} + a_{10} + a_{20} + a_{30} \\
f(0,1) &= a_{00} + a_{01} + a_{02} + a_{03} \\
f(1,1) &= \sum_{m=0}^{3}\sum_{n=0}^{3} a_{mn} = a_{00} + a_{01} + a_{02} + a_{03} + a_{10} + a_{11} + a_{12} + a_{13} + a_{20} + a_{21} \\
&\quad + a_{22} + a_{23} + a_{30} + a_{31} + a_{32} + a_{33}
\end{aligned}$$

ja neljä yhtälöä poikittaisderivaatoille

$$\begin{aligned}
f_x(0,0) &= a_{10} \\
f_x(1,0) &= a_{10} + 2a_{20} + 3a_{30} \\
f_x(0,1) &= a_{10} + a_{11} + a_{12} + a_{13} \\
f_x(1,1) &= \sum_{m=1}^{3}\sum_{n=0}^{3} a_{mn} m
\end{aligned}$$

ja eljä yhtälöä pystyderivaatoille

$$\begin{aligned}
f_y(0,0) &= a_{01} \\
f_y(1,0) &= a_{01} + a_{11} + a_{21} + a_{31} \\
f_y(0,1) &= a_{01} + 2a_{02} + 3a_{03} \\
f_y(1,1) &= \sum_{m=0}^{3}\sum_{n=1}^{3} a_{mn} n
\end{aligned}$$

ja viimeiset neljä yhtälöä poikittaisderivaatoille

$$\begin{aligned}
f_{xy}(0,0) &= a_{11} \\
f_{xy}(1,0) &= a_{11} + 2a_{21} + 3a_{31} \\
f_{xy}(0,0) &= a_{11} + 2a_{12} + 3a_{13} \\
f_{xy}(0,0) &= \sum_{m=1}^{3}\sum_{n=1}^{3} a_{mn} mn.
\end{aligned}$$

Tämä 16 yhtälön yhtälöryhmä muodostaa yleisen matriisiyhtälön $M\alpha = F$, missä

$$\alpha = \begin{bmatrix} a_{00} & a_{10} & a_{20} & a_{30} & a_{01} & a_{11} & a_{21} & a_{31} & a_{02} & a_{12} & a_{22} & a_{32} & a_{03} & a_{13} & a_{23} & a_{33} \end{bmatrix}^T,$$

$$F = \begin{matrix} [f(0,0) & f(1,0) & f(0,1) & f(1,1) & f_x(0,0) & f_x(1,0) & f_x(0,1) & f_x(1,1) \\ f_y(0,0) & f_y(1,0) & f_y(0,1) & f_y(1,1) & f_{xy}(0,0) & f_{xy}(1,0) & f_{xy}(0,1) & f_{xy}(1,1)]^T \end{matrix}$$

ja

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 0 & 2 & 4 & 6 & 0 & 3 & 6 & 9 \end{bmatrix}$$

Matriisiyhtälö $M\alpha = F$ on helppo ratkaista. Lasketaan ensiksi matriisin M käänteismat-

riisi M^{-1}. Käänteismatriisiksi saadaan

$$M^{-1} = \left[\begin{array}{cccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\
-3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\
9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\
-6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\
2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
-6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\
4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1
\end{array}\right]$$

Kerroinmatriisi M ja sen käänteismatriisi ovat aina samat riippumatta siitä, missä pisteissä interpolaatiota laskee, joten riittää ratkaista vain kerran M^{-1} ja käyttää sitä aina interpoloitaessa. Yhtälön $M\alpha = F$ ratkaisu saadaan kertomalla yhtälöä matriisilla M^{-1}, jolloin kertoimiksi saadaan $\alpha = M^{-1}F$.

Lasketuilla α:n arvoilla voidaan interpoloida kaavan

$$p(J,K) = \sum_{m=0}^{3}\sum_{n=0}^{3} a_{mn}(1-w)^m(1-h)^n$$

avulla mille tahansa pisteelle arvo neliön sisällä. Kuvia interpoloidessa tarvitsee vain käydä läpi kuva siirtämällä interpoloitavaa "neliötä"eli päivittämällä vektorin F arvoja. Matriisi M^{-1} on kerran laskettu ja sen avulla voi laskea uudet arvot vektorille α.

2.6 Harvoihin esityksiin perustuva superresoluutioalgoritmi

Tässä kappaleessa esittelemme superresoluutioalgoritmin, joka pyrkii parantamaan kuvan resoluutiota käyttämällä opetettuja kirjastoja harvoille esityksille, eikä pelkästään interpoloimalla uusia pikseleitä suuriresoluutioiseen kuvaan, kuten aiemmin esitetyt algoritmit. Tätä varten tarvitsee hieman perehtyä opetettaviin kirjastoihin. Tuloksissa viittaamme tähän algoritmiin merkinnällä *PESR* (Peleg-Elad-superresoluutio) algoritmin kehittäjien mukaan.

2.6.1 Kirjasto-oppiminen

Tässä työssä käytetään kirjastoa, jolla voidaan esittää signaaleja harvan esitysmatriisin avulla. Kirjaston ideana on, että se sisältää atomeja, joiden harvana lineaarikombinaationa voidaan esittää annettu signaali. Merkitään kirjastoa matriisilla $D \in \mathbb{R}^{n \times k}$ ja signaalia, esimerkiksi kuvaa, vektorilla $y \in \mathbb{R}^n$. Kaksiulotteinen kuva voidaan esittää vektorina, jossa on matriisin alkiot järjestyksessä sarakkeittain. Nyt tämä signaali y voidaan esittää täsmällisesti $y = Dx$ tai approksimaatiolla $y \approx Dx$, ehdolla $\| y - Dx \|_2 \leq \epsilon$. Tässä harva vektori $x \in \mathbb{R}^k$ sisältää signaalivektoria y edustavien atomien kertoimet. Tätä esitysvektoria kutsutaan harvaksi, koska se sisältää enimmäkseen nollia. Tämän takia signaalin esitysvektori x vie vähemmän muistia, mutta siitä on helppo rakentaa alkuperäinen signaali y tai sen approksimaatio.

On olemassa valmiita kirjastoja, joista voi valita sopivimman tai sitten sen voi opettaa itse käyttötarkoituksen mukaisesti, jolloin on mahdollista saada parempia tuloksia. Ideana on, että valitaan suuresta määrästä signaaleja koostuva harjoitusjoukko. Tällä harjoitusjoukolla opetetaan kirjasto, joka palauttaa parhaimman tuloksen jokaiselle signaalille harvalla esityksellä. Merkitään harjoitusjoukkoa $Y = \{y_i\}_{i=1}^N$ ja harvojen esitysvektorien joukkoa $X = \{x_i\}_{i=1}^N$. Tällöin kirjaston opettamisongelman voi esittää muodossa

$$\min_{D,X} \{\| Y - DX \|_F^2\} \text{ ehdolla } \forall i, \| x_i \|_0 \leq T_0. \tag{2.5}$$

Ehto $\| x_i \|_0 \leq T_0$, missä T_0 on jokin ennalta määrätty arvo, takaa sen, että jokaiselle signaalille y_i löytyy harva esitys x_i joka tuottaa approksimaation $y_i \approx Dx_i$. Tämän ongelman ratkaisuun käytetään K-SVD algoritmia, jonka aihe menee pitkälle ohi tämän työn aihepiirin. Tarkemmin K-SVD algoritmista ja ongelman (2.5) ratkaisusta voi lukea artikkelista [6].

2.6.2 Superresoluutioalgoritmi

Tässä osiossa esitellään superresoluutioalgoritmin perustoiminta, tarkemmin algoritmin toiminta on kerrottu artikkelissa [7]. Tämä algoritmi käyttää kahta kirjastoa, yhtä pieniresoluutioisten kuvien harvaa esitystä varten ja toista suuriresoluutioisten kuvien harvaa esitystä varten. Kirjastot on opetettu niin, että niillä pystyy harvana esityksenä esittää kuvan osia. Merkitään pieniresoluutioisten kuvien osien kirjastoa D_l ja suuriresoluutioisten kuvien osien kirjastoa D_h. Algoritmin toiminta perustuu siihen, että pieniresoluutioisen kuvan osasta ratkaistaan sen harva esitys α_l, jonka avulla luodaan suuriresoluutioisen kuvan vastaavan osan harva esitys α_h. Tästä harvasta esityksestä α_h kirjaston D_h avulla rakennetaan suuriresoluutioisen kuvan osa, joka vastaa pieniresoluutioisen kuvan osaa.

Tällä algoritmilla yritetään ratkaista ongelmaa, jossa oletetaan, että annettu pieniresoluutioinen kuva on saatu suuriresoluutioisesta kuvasta käyttämällä jotakin tuntematonta sumennusoperaattoria ja skaalausoperaattoria. Merkitään pieni- ja suuriresoluutioisia kuvia vektoreina $z_l \in \mathbb{R}^{N_l}$ ja $y_h \in \mathbb{R}^{N_h}$, missä $N_h = q^2 N_l$ ja $q > 1$ on jokin skaalauskerroin ja $q \in \mathbb{N}$. Tuntematonta sumennusoperaattoria merkitään $H \in \mathbb{R}^{N_h \times N_l}$ ja merkitään skaalausoperaattoria, joka skaalaa kuvaa molemmilta akseleilta kertoimen q verran, merkillä $Q \in \mathbb{R}^{N_h \times N_l}$. Oletetaan, että pieniresoluutioinen kuva on saatu yhtälöstä

$$z_l = QHy_h + v, \tag{2.6}$$

missä v on prosessissa aiheutunutta kohinaa. Sumennusytimelle H yksi suosituimmista vaihtoehdoista on bicubic suodatin, joten tässä työssä oletetaan, että sitä on käytetty. Koska yritämme ratkoa muotoa (2.6) olevaa ongelmaa, ratkaisemme eroavuuskuvan $y_{hl} = y_h - y_l$ ja sitten laskemme $\hat{y}_h = \hat{y}_{hl} + y_l$ lopulliseksi suuriresoluutioisen kuvan rekonstruktioksi, eli pidämme pieniresoluutioisen kuvan yksityiskohdat ja ratkaisemme vain puuttuvat yksityiskohdat.

Koska yritämme ratkaista pieniresoluutioisen kuvan osista suuriresoluutioista kuvaa, niin niiden pitää olla samankokoisia resoluutioltaan. Tämän takia interpoloimme bicubic interpolaatiolla pieniresoluutioisen kuvan z_l suuriresoluutioiseksi kuvaksi $y_l \in \mathbb{R}^{N_h}$. Nyt voimme valita samansuuruisia osia $p^k = R_k y$ kuvasta $y \in \mathbb{R}^{N_h}$. Olkoon lineaarinen operaattori R_k sellainen, että se valitsee k keskisen $\sqrt{N_h} \times \sqrt{N_h}$-kokoisen osakuvan. Tavoitteena on arvata $p_h^k = R_k Y_h$ pieniresoluutioisesta osasta $p_l^k = R_k y_l$. Suuriresoluutioinen kuva muodostetaan keskiarvoistamalla joukon $\{p_h^k\}_{k\in\Omega}$ päällekkäin menevät osat, missä Ω sisältää osakuvien keskipisteet ja osakuvat menevät päällekäin keskenään. Paremman tuloksen saa, kun maksimoi päällekäin menevien osien koon, mutta se hidastaa algoritmin suoritusta.

Koska pieniresoluutioisissa kuvien osissa on vähän yksityiskohtia, niin oletamme että, niiden harvoihin esityksiin riittää alimääritelty ortonormaali kirjasto $D_l^{N_l \times m_l}$. Nyt voimme

laskea pieniresoluutioiset harvat esitykset kuvan osien ja kirjaston avulla kaavalla

$$\alpha_l = (D_l)^T p_l. \tag{2.7}$$

Varmistaaksemme, että esitykset ovat harvoja käytämme kynnystä, jolloin esityksen α_l harvuuskaava $s_l \in \{-1, 1\}^{m_l}$ lasketaan ehdolla

$$s_{l,j} = \begin{cases} 1, & |\alpha_{l,j}| > \lambda \\ -1, & muulloin \end{cases}, \forall j = 1, \ldots, m_l, \tag{2.8}$$

missä λ valitaan jokaiselle osalle p^k erikseen. Kynnys λ valitaan siten, että kynnyksen alittavien $\alpha_{l,j} \leq \lambda$ termien neliöiden summa toteuttaa ehdon

$$\sum_{j=1}^{m_l} |\alpha_{l,j}|^2 : (|\alpha_{l,j}| \leq \lambda) \leq N_l \tau^2,$$

missä τ on valittu sen mukaan, kuinka tarkka pieniresoluutioisen kuvan harva esitys on.

Kun pieniresoluutioisten kuvien osien kirjasto valittiin olemaan alimääritelty, niin suuriresoluutioisten kuvien osia varten valitaan täydellinen tai ylimääritelty kirjasto, jotta saamme mahdollisimman tarkkoja esityksiä. Seuraavaksi yritämme ratkaista pieniresoluutioisista harvuus kaavoista $s_l \in \{-1, 1\}^{m_l}$ suuriresoluutioiset $s_h \in \{-1, 1\}^{m_h}$. Yritämme löytää tilastolliset riippuvuussuhteet harvuuskaavasta s_l ja yrittää ratkaista vastaavat riippuvuussuhteet harvuuskaavaan s_h.

Yhdestä harvuuskaavasta $s \in \{1, -1\}^m$ pystyy arvioimaan tilastollisia riippuvuussuhteita Boltzmann koneella

$$\mathrm{P}(s) = \frac{1}{Z} e^{b^T s + \frac{1}{2} s^T W s},$$

missä $b \in \mathbb{R}^m$ on harha vektori ja $W \in \mathbb{R}^{m \times m}$ on vuorovaikutusmatriisi. Harha vektori korjaa vuorovaikutusmatriisin tuloa oikeaan suuntaan, jos tiedetään mihin suuntaan se vääristää tulosta. Käytämme muunnosta tästä koneesta, jotta löytäisimme tilastolliset riippuvuussuhteet pieniresoluutioisen ja suuriresoluutioisen kuvaparin väliltä. Käytämme rajoitettua Boltzmannin konetta ehdollisena todennäköisyytenä

$$\mathrm{P}(s_h | s_l) = \frac{1}{Z} e^{b_h^T s_h + s_h^T w_{hl} s_l} = \prod_{j=1}^{m_h} \Phi((b_{h,j} + w_{hl,j}^T s_l) s_{h,j}),$$

missä $b_h \in \mathbb{R}^{m_h}$ on harha vektori suuriresoluutioiselle harvuuskaavalle, $w_{hl} \in \mathbb{R}^{m_h \times m_l}$ on vuorovaikutusmatriisi, joka yhdistää pieni- ja suuriresoluutioiset harvuuskaavat ja $\Phi(x) = (1 + e^{-2x})^{-1}$ on eräs sigmoid-funktio

Käytämme rajoitettua Boltzmannin konetta, koska se on yksinkertainen eksponentiaalinen malli kahden binäärisen vektorin väliltä, jonka takia käytämme binäärisiä harvuuskaavoja s_h ja s_l, eikä esitysvektoreita α_h ja α_l. Laskemme todennäköisyyksiä suuriresoluutioisen harvuuskaavan termeille $s_{h,j}$ ehdolla s_l kaavalla

$$\mathrm{P}(s_{h,j} = 1|s_l) = \Phi(b_{h,j} + w_{hl,j}^T s_l), \quad \forall j = 1, \ldots, m_h.$$

Suuriresoluutioisen osakuvan esitys α_h saadaan ratkaistun harvuuskaavan s_h ja ratkaistun pieniresoluutioisen osakuvan esityksen α_l avulla ehdolla

$$\alpha_{h,j} = \begin{cases} u_j, & s_{h,j} = 1 \\ 0, & s_{h,j} = -1 \end{cases}, \forall j = 1, \ldots, m_h,$$

missä $u \in \mathbb{R}^{m_h}$ on gaussinen jakauma annetulla α_l siten, että $u|\alpha_l \sim N(C_{hl}\alpha_l, \Sigma_{hl})$, jossa $C_{hl} \in \mathbb{R}^{m_h \times m_l}$ ja $\Sigma_{hl} \in \mathbb{R}^{m_h \times m_h}$. Tästä johdetaan ehdollinen odotusarvo esitykselle α_h kaavalla

$$\mathrm{E}(\alpha_{h,j}|s_{h,j} = 1, \alpha_l) = c_{hl,j}^T \alpha_l, \quad \forall j = 1, \ldots, m_h.$$

Nyt tällä tilastollisella mallilla laskemme esityksen α_h jokaisen termin pienimmän neliösumman erotuksen arviolla ($PNSE$) esityksestä α_l ja kaavasta s_l,

$$\alpha_{h,j} = \mathrm{E}(\alpha_{h,j}|s_l, \alpha_l) = \sum_{s_h \in \Gamma_j} \mathrm{E}(\alpha_{h,j}|s_h, s_l, \alpha_l)\mathrm{P}(s_h|s_l, \alpha_l) \stackrel{*}{=} \sum_{s_h \in \Gamma_j} \mathrm{E}(\alpha_{h,j}|s_{h,j} = 1, \alpha_l)\mathrm{P}(s_h|s_l)$$

(2.9)

$$= \mathrm{E}(\alpha_{h,j}|s_{h,j} = 1, \alpha_l)\mathrm{P}(s_{h,j} = 1|s_l) = (c_{hl,j}^T \alpha_l)\Phi(b_{h,j} + w_{hl,j}^T s_l),$$

missä $\Gamma_j = \{\gamma \in \mathbb{R}^{m_h} : g_j = 1\}$ ja yhtälössä (*) on käytetty oletuksia

$$\begin{array}{llll} 1) & \alpha_{h,j} \perp \langle s_l, \{s_{h,i}\}_{i \neq j} \rangle & | & (s_{h,j=1,\alpha_l}), \quad \forall j. \\ 2) & s_{h,j} \perp \alpha_l & | & s_l, \quad \forall j. \end{array}$$

Superresoluutiomenetelmän toiminta kokonaisuudessaan on tiivistetty algoritmissa 1.

2.6.3 Parametrien opettaminen

Tässä osiossa lyhyesti selitetään, miten algoritmin 1 vaatimat parametrit opetetaan antamaan parhaan approksimaation suuriresoluutioiselle kuvalle pieniresoluutioisesta kuvasta käyttämällä N kappaleen pieni- ja suuriresoluutioista kuvaparia $\{p_l^k, p_h^k\}$ opetusjoukkona. Tavoitteena on opettaa parametrit

$$\Theta = \{D_l, D_h, C_{hl}, b_h, W_{hl}\}. \tag{2.10}$$

Algorithm 1 Superresoluutioalgoritmi

1: **Syöte:** Kuva z_l, skaalaus q sekä parametrit $D_l, D_h, C_{hl}, b_h, W_{hl}$
2: **Bicubic interpolaatio** skaalauksella q kuvalle z_l jolla saadaan kuva y_l
3: **Kuvan ositus** päällekkäisiin osakuviin $\{p_l^k\}$ joiden keskipisteet $k \in \Omega$ kuvasta y_l
4: **for** $k \in \Omega$ **do**
5: **Laske pieniresoluutioinen esitys** α_l^k osalle p_l^k kaavalla (2.7)
6: **Laske pieniresoluutioinen harvuuskaava** s_l^k esityksestä α_l^k kaavalla (2.8)
7: **Laske PNSE arvio esitykselle** $\hat{\alpha}_h^k$ kaavalla (2.9)
8: **Suuriresoluutioisen osakuvan rekonstruktio:** $\hat{p}_h^k = D_h \alpha_h^k$
9: **Laske** y_h **ja** y_l **kuvien erotus** $\hat{y}_{hl}$ keskiarvoistamalla osien $\{\hat{p}_h^k\}_{k\in\Omega}$ päällekäisyydet
10: **Tuloste: Suuriresoluutioisen kuvan rekonstruktio**: $\hat{y}_h = \hat{y}_{hl} + y_l$

Kuvan osat valitaan samoista keskipisteistä, eli joukosta Ω, molemmista kuvista y_l, y_h. Tätä varten pieniresoluutioiset kuvat interpoloidaan bicubic interpolaatiolla samankokoisiksi.

Parametrien Θ opettaminen voidaan esittää optimisaatio-ongelmana

$$\underset{\Theta}{\operatorname{argmin}} \sum_{k=1}^{N} \left\| D_h \left(\left[\Phi \left(b_h + W_{hl} s_l^k \right) \right] \circ \left[C_{hl} (D_l)^T p_l^k \right] \right) - p_h^k \right\|_2^2, \tag{2.11}$$

missä $\circ$ on Hadamard tulo ($A \circ B = AB$, missä $(AB)_{i,j} = A_{i,j} * B_{i,j}$).

Koska tämä viiden muuttujan optimisaatio (2.11) on todella vaikea ratkaista, niin annetaan aluksi alkuarvaukset kirjastoille D_l, D_h. Kirjaston D_l oletetaan olevan alimääritelty ja ortonormaali, jotta sen avulla laskeminen olisi nopeampaa kuin ylimääritellyn matriisin avulla ja se pienentää matriisien C_{hl} ja w_{hl} kokoa helpottaen niiden ratkaisemista. Kirjastoksi D_l valitaan ortogonaalinen muunnos, joka toimii hyvin luonnollisille kuville. Muunnos voi olla esimerkiksi DCT (diskreetti kosinimuunnos) tai PCA (Pääkomponenttianalyysi) [8].

Seuraavaksi opetetaan kirjasto D_h opetuskuvilla $\{y_h\}$ käyttäen K-SVD-algoritmia [6]. Tästä kirjastosta D_h lasketaan harvat esitykset $\{\alpha_h^k\}$ ja niitä vastaavat harvuuskaavat $\{s_h^k\}$ OMP-algoritmilla (orthogonal matching pursuit [9]). Näiden avulla voidaan määritellä alkuarvaus matriisille C_{hl}, joka kartoittaa parhaiten pieni- ja suuriresoluutioisten esitysten epänegatiiviset kertoimet,

$$\hat{C}_{hl} = \underset{C_{hl}}{\operatorname{argmin}} \sum_{k=1}^{N} \left\| \Lambda_h^k \left(C_{hl} \alpha_l^k - \alpha_h^k \right) \right\|_2^2,$$

missä λ_h^k on diagonaalimatriisi, jonka diagonaalille on sijoitettu vektori $\frac{1}{2}(s_h^k + 1)$. Tämä

on pienimmän neliösumman ongelma, johon on olemassa ratkaisu

$$\text{vek}\left(\hat{C}_{hl}\right) = \left[\sum_{k=1}^{N}\left(\alpha_l^k(\alpha_l^k)^T\right)\otimes\Lambda_h^k\right]^{-1}\text{vek}\left(\sum_{k=1}^{N}\Lambda_h^k\alpha_h^k(\alpha_h^k)^T\right), \tag{2.12}$$

missä vek(A) luo vektorin a siten, että siihen on pinottu matriisin A sarakkeet ja $\otimes$ on kroneckerin tulo

$$\mathbf{A}\otimes\mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B}\end{bmatrix}.$$

Seuraavaksi päivitetään D_h ja C_{hl} siten, että ne konstruoivat parhaimman arvion osakuville $\{p_h^k\}$,

$$\hat{D}_{h,j},\hat{C}_{hl,j} = \underset{D_h,C_{hl}}{\text{argmin}}\sum_{k=1}^{N}\left\|D_h\Lambda_h^kC_{hl}\alpha_l^k - p_h^k\right\|_2^2.$$

Tästä ratkaisuksi johdetaan

$$\begin{aligned}&\hat{d}_{h,j} = \text{matriisin } E_h^J(A_l^J)^\dagger A_l^J(E_h^J)^T \text{ suurin ominaisarvo,}\\ &\hat{c}_{hl,j}^T = \hat{d}_{h,j}^TE_h^J(A_l^J)^\dagger,\quad \forall j=1,\ldots,m_h,\end{aligned} \tag{2.13}$$

missä $X^\dagger = (X^*)^T$ on matriisin X hermiittinen konjugaatti, matriisin A_l sarakkeet on muodostettu vektoreista α_l^k, J on indeksijoukko niille indekseille $k=\{1,\ldots,N\}$, joilla $s_{h,j}^k = 1$ ja matriisin E_h sarakkeet ovat jäännösvektoreita

$$e_h^k = p_h^k - \sum_{i\neq j, s_{h,i}^k=1}\hat{d}_{h,i}\left(\hat{c}_{hl,i}^T\alpha_l^k\right).$$

Viimeisenä arvioidaan parametrit b_h ja W_{hl} joilla saadaan paras rekonstruktio suuriresoluutioisille osakuville pieniresoluutioisista kuvista. Koska yhtälön (2.9) PNSE-arvio termille $\alpha_{h,j}$ riippuu vain termistä $b_{h,j}$ ja W_{hl} sarakkeesta $w_{hl,j}$, niin voimme laskea parametrit b_h ja W_{hl} elementti kerrallaan ratkaisemalla

$$\hat{b}_{h,j},\hat{w}_{hl,j}^T = \underset{b_{h,j},w_{hl,j}^T}{\text{argmin}}\left\|d_{h,j}\left(\Theta(b_{h,j}+w_{hl,j}^TS_l)\right)\circ\left(c_{hl,j}^TA_l\right) - \tilde{E}_h\right\|_2^2,\quad \forall j=1,\ldots,m_h, \tag{2.14}$$

missä $\tilde{E}_h$ on muodostettu sarakkeittain jäännösvektoreista

$$\tilde{e}_h^k = p_h^k - \sum_{i\neq j}\hat{d}_{h,i}\Theta\left(b_{h,i}+w_{hl,i}^Ts_l^k\right)\left(\hat{c}_{hl,i}^T\alpha_l^k\right).$$

Koska optimisaatio-ongelma (2.14) on konveksi, niin approksimoimme sen ratkaisua konjugaatti-gradientti-menetelmän 20 iteraation tuloksella käyttäen alkuarvausta

$$\hat{b}_{h,j} = \frac{1}{2} \ln \left(\frac{|J|}{N - |J|} \right), \quad \hat{w}_{hl,j}^T = 0, \quad \forall j = 1, \ldots, m_h. \tag{2.15}$$

Lopuksi päivitetään lopullinen kirjasto D_h. Ensiksi päivitetyillä b_h, W_{hl} ja C_{hl} arvioilla määrittelemme tarkemmat $\{\alpha_l^k, s_l^k\}$ kaavalla (2.9). Sitten päivitämme kirjaston D_h siten, että se palauttaa parhaimmat arviot osakuville $\{p_h^k\}$ kaavalla

$$\hat{D}_h = \underset{D_h}{\operatorname{argmin}} \|D_h A_h - P_h\|_2^2 = P_h (\hat{A}_h)^\dagger, \tag{2.16}$$

missä matriisit $\hat{A}_h$ ja P_h ovat muodostettu sarakkeittain vektoreista $\{\hat{\alpha}_h^k\}$ ja $\{p_h^k\}$.

Parametrien opetusalgoritmi on tiivistetty algoritmiin 2.

Algorithm 2 Parametrien opetus

1: **Syöte:** Harjoitusjoukko $\{p_l^k, p_h^k\}$ pieni- ja suuriresoluutioisista osakuvista.
2: **Määritä pieniresoluutioinen kirjasto** D_l muunnoksella DCT tai PCA.
3: **Laske harvat esitykset ja harvuuskaavat** $\{\alpha_l^k, s_l^k\}$ osille $\{p_l^k\}$ kaavoilla (2.7-2.8).
4: **Asetetaan suuriresoluutioiseksi kirjastoksi** D_h alkuarvaus käyttämällä K-SVD algoritmia osille $\{p_h^k\}$.
5: **Laske suuriresoluutioiset harvat esitykset ja harvuuskaavat** $\{\alpha_h^k, s_h^k\}$ osista $\{p_h^k\}$ ja kirjastosta D_h käyttäen OMP algoritmia.
6: **Määritellään alkuarvaus matriisille** C_{hl} parametrien $\{\alpha_l^k, \alpha_h^k, s_h^k\}$ avulla kaavalla (2.12).
7: **Päivitetään kirjasto** D_h **sekä matriisi** C_{hl} parametrien $\{\alpha_l^k, s_h^k, p_h^k\}$ avulla kaavalla (2.13).
8: **Alustetaan alkuarvaukset** b_h **ja** W_{hl} **rajoitetulle Boltzmannin koneelle** kaavalla (2.15).
9: **Päivitetään parametrit** b_h **ja** W_{hl} parametrien $\{\alpha_l^k, s_l^k, p_h^k\}$ ja arvioiden D_h, C_{hl} avulla kaavasta (2.14) konjugaatti gradienttimenetelmällä.
10: **Laske PNSE arvio esityksille** $\{\hat{\alpha}_h^k\}$ parametreista $\{\alpha_l^k, s_l^k\}$ ja arvioista D_h, C_{hl}, b_h, W_{hl} kaavalla (2.9).
11: **Päivitä lopullinen kirjasto** D_h parametreista $\{\hat{\alpha}_h^k, p_h^k\}$ kaavalla (2.16).
12: **Tuloste:** Opetetut parametrit $D_l, D_h, C_{hl}, b_h, W_{hl}$.

2.7 SSIM

SSIM on algoritmi, jolla mitataan kahden kuvan samankaltaisuutta. Tätä algoritmia käytetään paljon kuvankäsittelyalgoritmien tuloksien vertailussa alkuperäiseen kuvaan. SSIM eli *Structural Similarity* indeksin on tarkoitus antaa parempia tuloksia kuin perinteisemmillä menetelmillä kuten *MSE*:llä (mean square error). SSIM pyrkii havaitsemaan rakenteellisia samankaltaisuuksia kuvien kesken, kuten ihmissilmä tekee. Tässä kappaleessa kerrotaan SSIM algoritmin toiminta, kuten se on esitelty artikkelissa [5].

SSIM vertaa useaa kuvan ominaisuutta samanaikaisesti. Se vertaa valoisuutta, kontrastia ja rakenteellisia eroja. Ensiksi kuvista verrataan valoisuutta. Verrataan kahta kuvaa eli signaalia x ja y. Oletetaan, että signaalit ovat diskreettejä, jolloin arvioidaan valoisuuden keski-intensiteetiksi

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{2.17}$$

Valoisuuden vertailufunktio $l(x, y)$ tehdään sitten valoisuuden keski-intensiteeteille μ_x ja μ_y.

Toiseksi poistetaan signaalista valoisuuden keski-intensiteetti laskemalla $x - \mu_x$. Käytämme keskihajontaa eli varianssin neliöjuurta arvionamme signaalien kontrastista. Keskihajonta on tällöin

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}. \tag{2.18}$$

Kontrastivertaus $c(x, y)$ on tällöin vertailu σ_x:stä ja σ_y:stä.

Kolmanneksi signaalit normalisoidaan jakamalla ne omilla keskihajonnoilla, jolloin molemmilla verrattavilla kuvilla on standardoitu normaalijakauma. Rakenteellinen vertailu $s(x, y)$ tehdään näille normalisoiduille signaaleille $(x - \mu_x)/\sigma_x$ ja $(y - \mu_y)/\sigma_y$.

Lopuksi nämä kolme komponenttia yhdistetään laskeaksemme samankaltaisuusmitan

$$S(x, y) = f(l(x, y), c(x, y), s(x, y)). \tag{2.19}$$

Jotta voidaan määritellä funktiot f, l, s ja c, niin meidän on huomioitava, että samankaltaisuusmitan on täytettävä seuraavat ehdot. Mitan on oltava symmetrinen: $S(x, y) = S(y, x)$, rajoitettu: $S(x, y) \leq 1$ ja maksimiarvon on oltava yksikäsitteinen: $S(x, y) = 1$ jos ja vain jos $x = y$.

Valoisuuden vertailufunktioksi määritellään

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.20}$$

missä C_1 on vakio, jonka avulla vältytään arvojen heilahtelulta, kun $\mu_x^2 + \mu_y^2$ on lähellä nollaa. Valitsemme $C_1 = (K_1 L)^2$, missä L on kuvan värisyvyys (255 8-bittisille mustavalkokuville) ja $k_1 << 1$. Yhtälö (2.20) selvästi toteuttaa vaaditut ehdot.

Kontrastin vertailufunktioksi määritellään

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \tag{2.21}$$

missä $C_2 = (K_2 L)^2$ ja $K_2 << 1$. Myöskin funktio c toteuttaa vaaditut ehdot.

Rakenteellinen vertailufunktio suoritetaan normalisoiduille standartoiduille signaaleille $(x - \mu_x)/\sigma_x$ ja $(y - \mu_y)/\sigma_y$. Määrittelemme funktioksi

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \tag{2.22}$$

missä C_3 on vakio ja

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y).$$

Lopuksi yhdistämme vertailufunktiot saadaksemme samankaltaisuusfunktioksi signaaleille x ja y olemaan

$$\mathrm{SSIM}(x,y) = l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma, \tag{2.23}$$

missä $\alpha > 0, \beta > 0$ ja $\gamma > 0$ määrittelevät kuinka paljon painotamme eri vertailukomponentteja. Yksinkertaistetaan funktiota määrittelemällä, että $\alpha = \beta = \gamma = 1$ ja valitsemalla $c_3 = c_2/2$. Tällöin funktio on muotoa

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}. \tag{2.24}$$

Luku 3

Tulokset

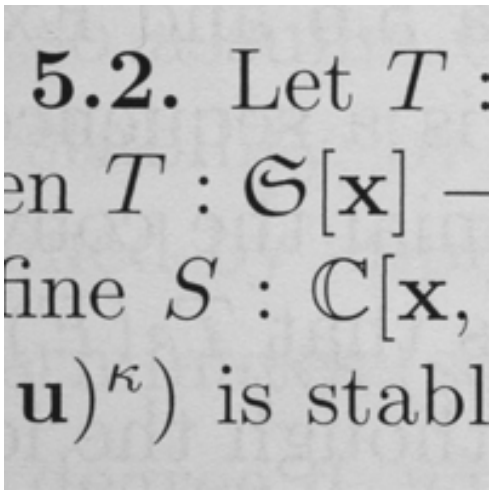
Luvussa 2.3 esitetyille kuville on suoritettu luvuissa 2.4-2.6 esitetyt superresoluutioalgoritmit. Aluksi osakuvat 2.3b ja 2.3a on alinäytteistetty puoleen kokoon bicubic-algoritmilla, eli esimerkiksi sammakkokuvan resoluutio on pienennnetty 800×800 pikelin resoluutioista 400×400 pikselin resoluutioon, joka näkyy kuvassa 3.1b. Tälle alinäytteistetylle kuvalle suoritettujen algoritmien tulokset näkyvät kuvassa 3.2. Kuvassa 3.2a on alkuperäisestä kuvasta 2.2 valittu 800×800 resoluution mustavalkoinen osakuva vertailua varten, kuvassa 3.2b on PESR-algoritmilla saatu suurennos, kuvassa 3.2c on bicubic-interpolaatiolla saatu suurennos ja kuvassa 3.2d on bilineaarisella-interpolaatiolla saatu suurennos.

Osakuva tekstistä 2.3a on myös alinäytteistetty puoleen kokoon eli 200×200 pikselin resoluutioon, joka näkyy kuvassa 3.1a. Alinäytteistykselle on suoritetttu luvuissa 2.4-2.6 esitetyt superresoluutioalgorimit. Tulokset on koottu kuvaan 3.2 alkuperäisen osakuvan kanssa. Kuvassa 3.3a on valittu 400×400 resoluution mustavalkoinen osakuva vertailua varten, kuvassa 3.3b on PESR-algoritmilla saatu suurennos, kuvassa 3.3b on bicubic-interpolaatiolla saatu suurennos ja kuvassa 3.3b on bilineaarisella-interpolaatiolla saatu suurennos.

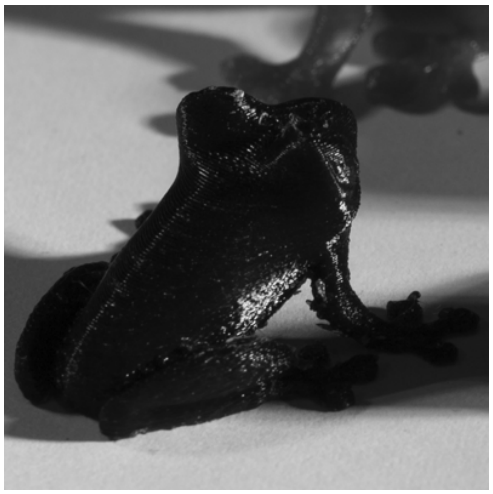
Jokaista eri superresoluutioalgoritmilla saatua kuvaa verrattiin alkuperäiseen kuvaan käyttämällä luvussa 2.7 esitettyä SSIM samankaltaisuusmittaa ja myös laskemalla kuvien erotuksen L^2-normi. Mitä lähempänä SSIM arvo on ykköstä sitä samankaltaisempia kuvat ovat keskenään. L^2-normi laskee pikselien erotusten neliöiden summasta neliöjuuren kaavalla

$$\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^{n} |x_i|^2},$$

missä x_i käy kuvien erotuksen kaikki pikselit läpi. Kuvien erotuksen L^2-normi kertoo kuvien välisen etäisyyden toisistaan, mitä lähempänä nollaa normi on sitä lähempänä kuvien pikselien arvot ovat keskenään.



(a)



(b)

Kuva 3.1: (a) Mustavalkoinen osakuva tekstistä alinäytteistettynä 200×200 pikselin kokoon (b) Mustavalkoinen osakuva 3D-tulostetusta sammakosta alinäytteistettynä 400×400 pikselin kokoon

SSIM ja L^2-normin tulokset kuvalle sammakosta on kirjattu taulukkoon 3.1 ja tulokset kuvalle tekstistä on kirjattu taulukkoon 3.2.

800×800-kokoinen kuva sammakosta puolen koon pienennöksestä

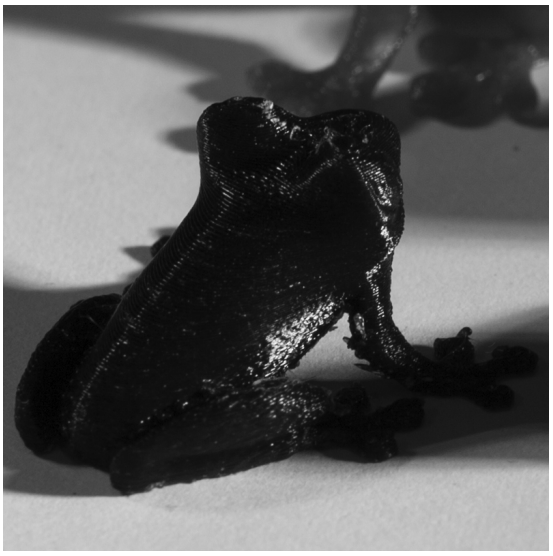
	SSIM	L^2-normi
PESR	0.9992	457.33
Bicubic-interpolaatio	0.9805	895.77
Bilineaarinen interpolaatio	0.9876	758.62

Taulukko 3.1: 800×800-kokoiselle kuvalle sammakosta suoritettu alinäytteistäminen koon puolittamiseksi ja tälle pienennökselle on suoritettu SR-algoritmeja, joita on verrattu alkuperäiseen 800×800 kuvaan.

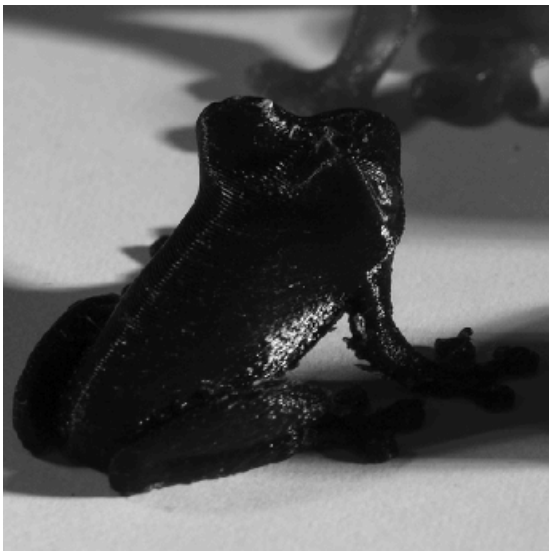
400×400-kokoinen kuva tekstistä puolen koon pienennöksestä

	SSIM	L^2-normi
PESR	0.9976	457.04
Bicubic-interpolaatio	0.9292	2357.34
Bilineaarinen interpolaatio	0.9621	1472.64

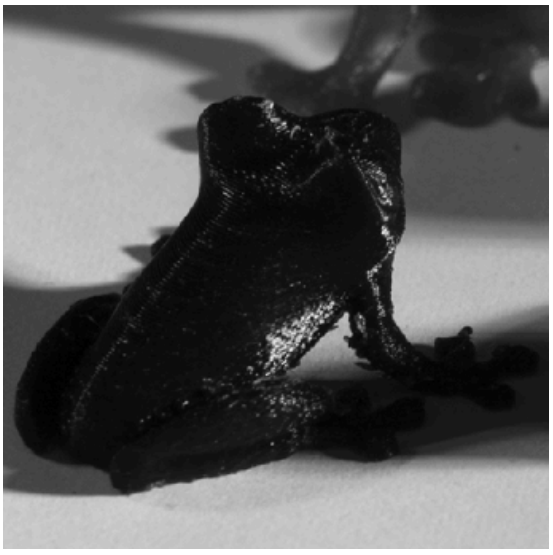
Taulukko 3.2: 400×400-kokoiselle kuvalle tekstistä suoritettu alinäytteistäminen koon puolittamiseksi ja tälle pienennökselle on suoritettu SR-algoritmeja, joita on verrattu alkuperäiseen 400×400 kuvaan.



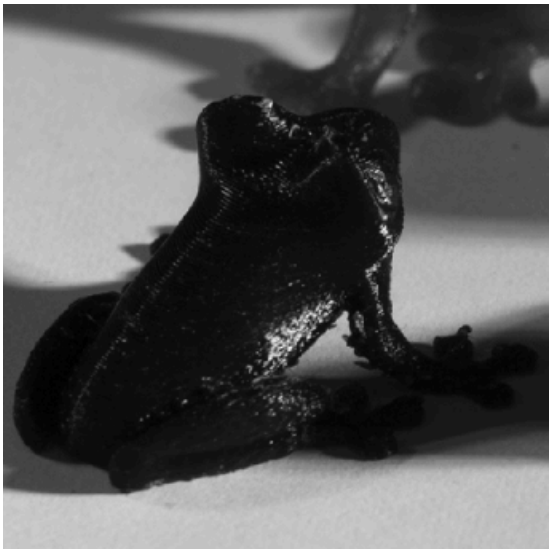
(a) 800×800 pikselin osakuva



(b) PESR-algoritmi

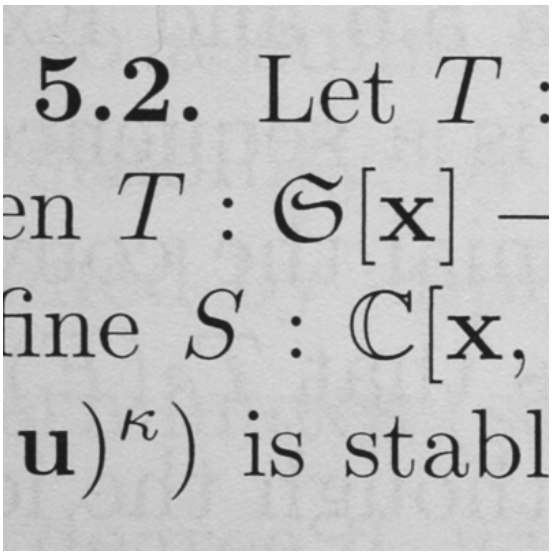


(c) Bicubic-interpolaatio

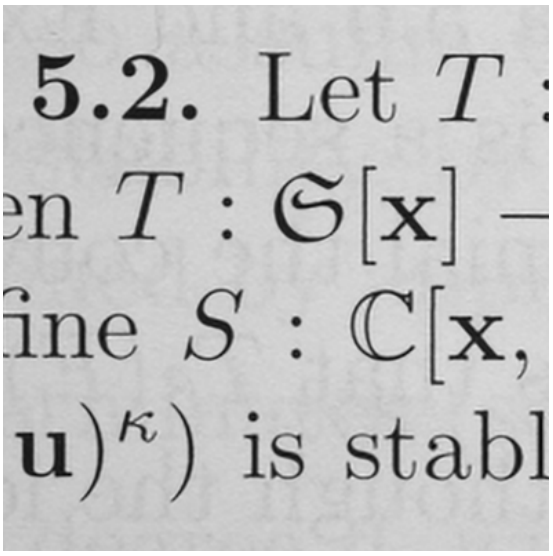


(d) Bilineaarinen interpolaatio

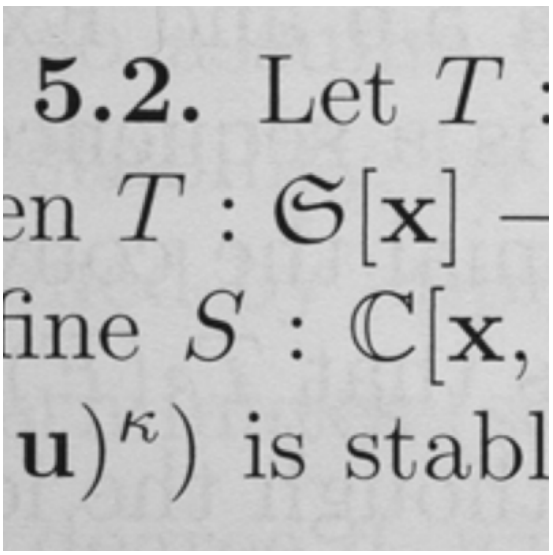
Kuva 3.2: Kaksinkertaiselle pienennökseelle suoritetut algoritmien tulokset sammakkokuvalle



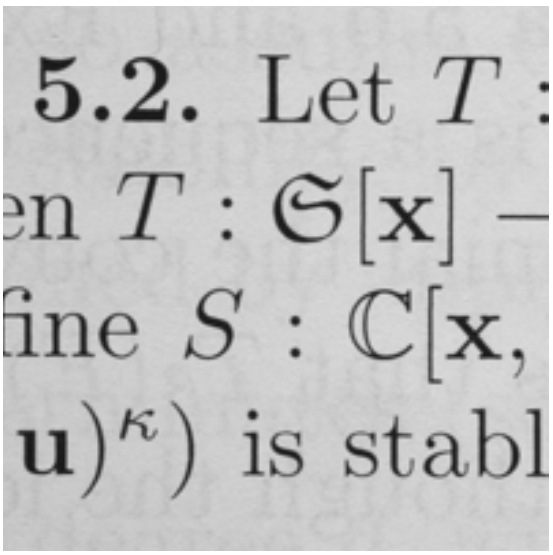
(a) 800×800 pikselin osakuva



(b) PESR-algortimi



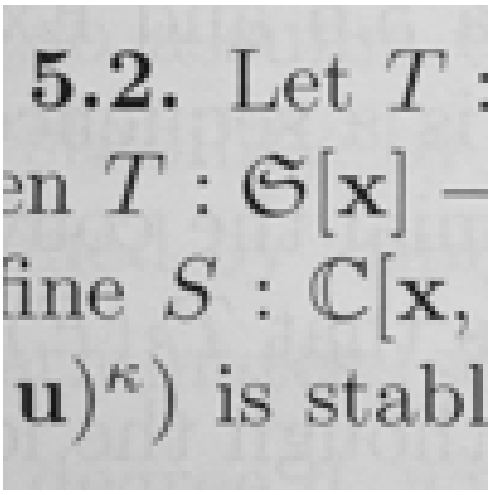
(c) Bicubic-interpolaatio



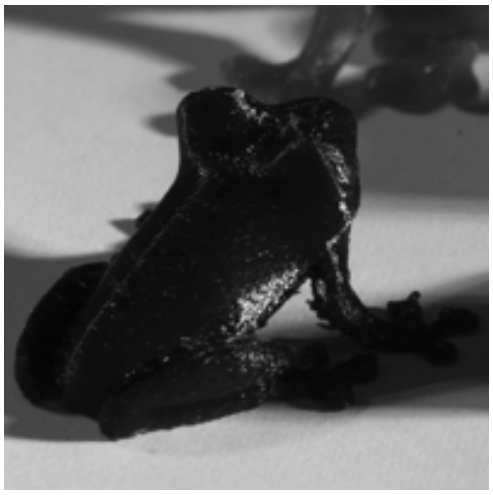
(d) Bilineaarinen interpolaatio

Kuva 3.3: Kaksinkertaiselle pienennökseelle suoritetut algoritmien tulokset tekstikuvalle

Jotta erot alkuperäisen kuvan ja pienennöksestä superresoluutioalgoritmeilla saatujen kuvien välillä näkyisivät selvemmin, on osakuvista alinäytteistämällä tehty neljäsosa kokoiset pienennökset. Neljäsosapienennös tekstistä on resoluutioltaan 100×100 pikseliä ja näkyy kuvassa 3.4a. Sammakosta tehty neljäsosapienennös näkyy kuvassa 3.4b ja on resoluutioltaan 200×200 pikseliä. Näistä pienennöksistä huomaa jo selvästi, miten osa yksityiskohdista on kadonnut ja yksittäiset pikselit alkavat näkyä. Kuvasta joka on otettu tekstistä huomaa, miten esimerkiksi kaksoispisteiden pisteet ovat menettäneet pyöreytensä ja muistuttavat neliöitä.



(a)



(b)

Kuva 3.4: (a) Mustavalkoinen osakuva tekstistä alinäytteistettynä 100x100 pikselin kokoon (b) Mustavalkoinen osakuva 3D-tulostetusta sammakosta alinäytteistettynä 200×200 pikselin kokoon

Kuvan 3.4 pienennöksille suoritettiin myös superresoluutioalgoritmit. Tulokset sammakkokuvalle on nähtävissä kuvassa 3.5 ja kuvassa 3.6 on tulokset kuvalle tekstistä. SSIM ja L^2-normin tulokset kuvalle sammakosta on kirjattu taulukkoon 3.3 ja tulokset kuvalle tekstistä on kirjattu taulukkoon 3.4.

800×800-kokoinen kuva sammakosta neljäsosa pienennöksestä

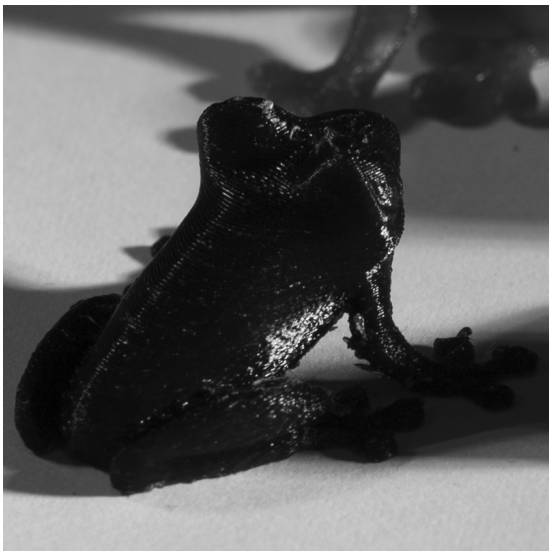
	SSIM	L^2-normi
PESR	0.9743	1534.09
Bicubic-interpolaatio	0.9530	2074.86
Bilineaarinen interpolaatio	0.9358	2072.64

Taulukko 3.3: 800×800-kokoiselle kuvalle sammakosta on suoritettu alinäytteistäminen neljäsosa kokoon ja tälle pienennökselle on suoritettu SR-algoritmeja, joita verrattu alkuperäiseen osakuvaan.

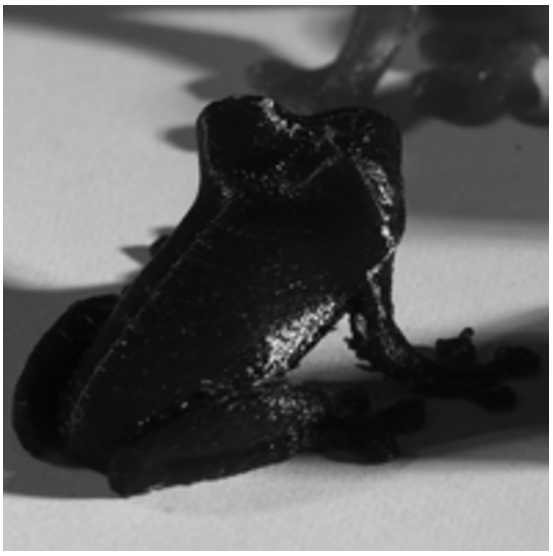
400×400-kokoinen kuva tekstistä neljäsosa pienennöksestä

	SSIM	L^2-normi
PESR	0.8772	2499.69
Bicubic-interpolaatio	0.8150	3368.63
Bilineaarinen interpolaatio	0.7809	3501.62

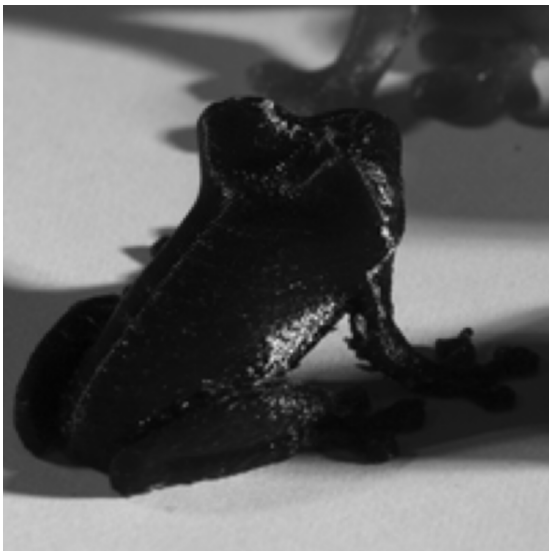
Taulukko 3.4: 400×400-kokoiselle kuvalle tekstistä on suoritettu alinäytteistäminen neljäsosa kokoon ja tälle pienennökselle on suoritettu SR-algoritmeja, joita on verrattu alkuperäiseen osakuvaan.



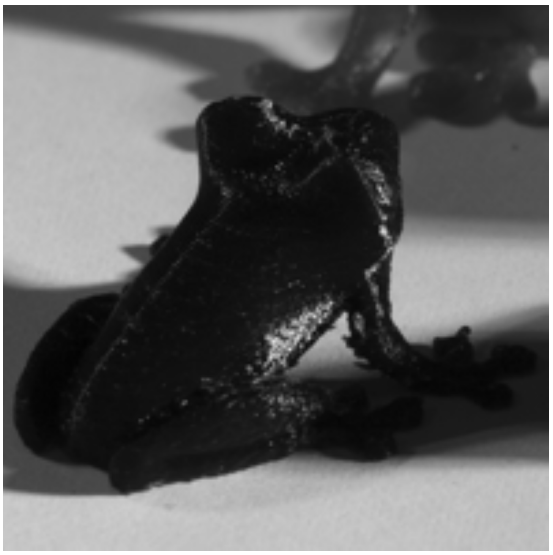
(a) 800×800 pikselin osakuva



(b) PESR-algoritmi

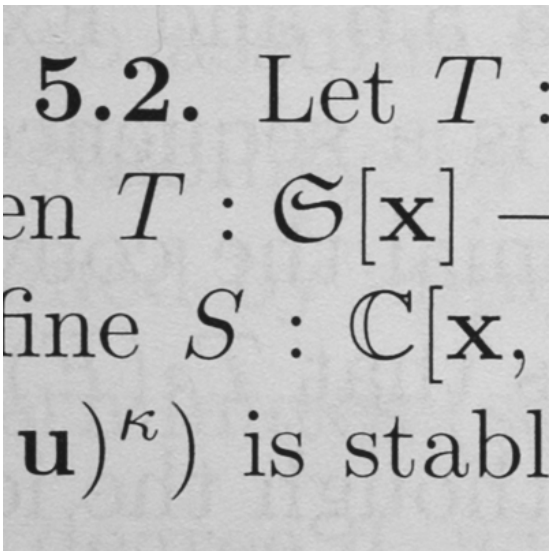


(c) Bicubic-interpolaatio

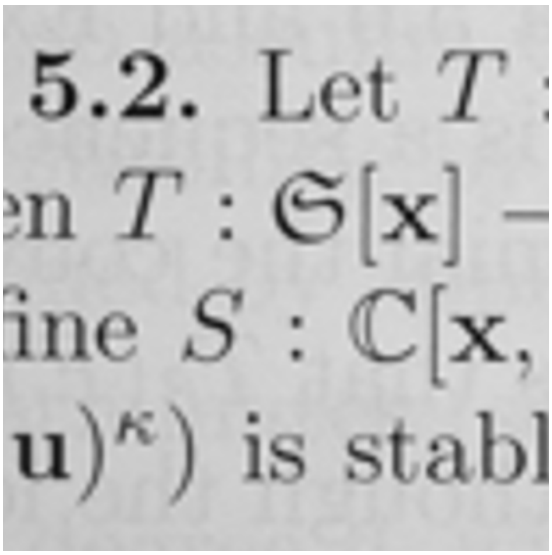


(d) Bilineaarinen interpolaatio

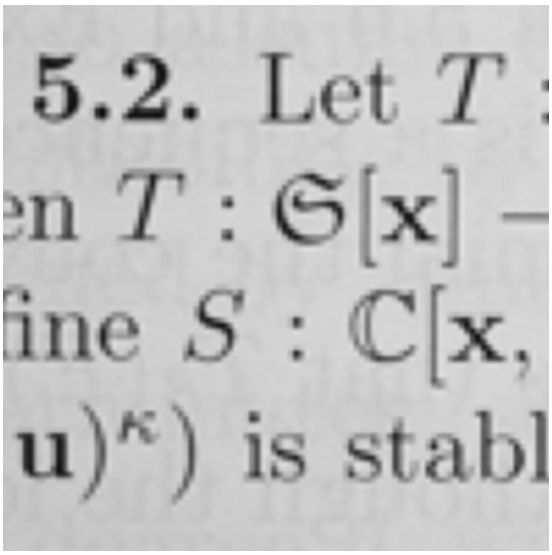
Kuva 3.5: Sammakkokuvan neljäsosapienennökselle suoritettujen algoritmien tulokset



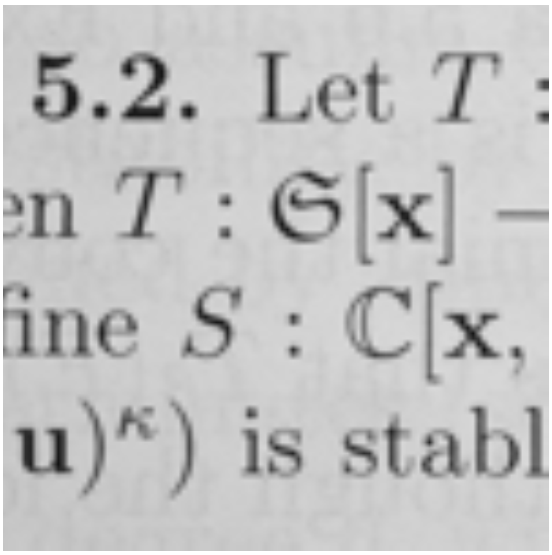
(a) 800×800 pikselin osakuva



(b) PESR-algoritmi



(c) Bicubic-interpolaatio



(d) Bilineaarinen interpolaatio

Kuva 3.6: Tekstikuvan neljäsosapienennökselle suoritettujen algoritmien tulokset

Luku 4

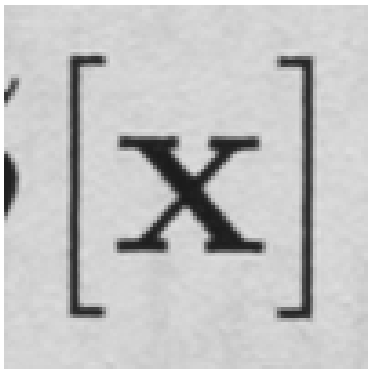
Johtopäätökset

Bilineaarinen interpolaatio, bicubic-interpolaatio ja PESR-algoritmi tuottivat hyviä tuloksia sekä pienemmälle 400×400 resoluution kuvalle tekstistä että isommalle 800×800 resoluution kuvalle sammakosta, kun algoritmit suoritettiin puolen koon alinäytteistykselle. Kaksinkertaista suurennosta tehdessä ei ole juuri merkitystä, mitä superresoluutio algoritmia käyttää.

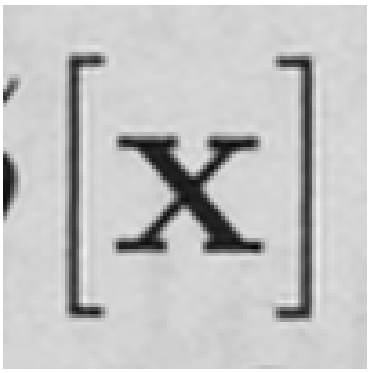
Kuvat 3.2 ja 3.3 sekä taulukot 3.1 ja 3.2 SSIM:n ja L^2-normin tuloksista kertovat, että kaksinkertaisella suurennoksella tässä työssä esitetyillä menetelmillä on päästy hyvin lähelle alkuperäistä kuvaa. Tekstikuvalla ollaan päästy yli 92% vastaavuuteen kaikilla algoritmeilla ja kuvalla sammakosta ollaan päästy jopa 98% vastaavuuteen huonoimmassa tapauksessa. Tarkastelemalla taulukoiden tuloksia huomataan, että PESR-algoritmilla ollaan saatu parhaimmat tulokset kummastakin kuvasta. Jopa 99,9% vastaavuus sammakkokuvalle ja 99,8% vastaavuus kuvalle tekstistä. Huonoiten menestyi bicubic-algoritmi, joka sai sammokkuvalle 98,8% vastaavuuden ja kuvalle tekstistä 92,9% vastaavuuden, joka itsessään on hyvä tulos, mutta 2 prosenttiyksikköä huonommin kuin bilineaarinen interpolointi. Bilineaarinen interpolointi, kuten edellä mainitut, antoi paremman tuloksen sammmakkokuvalle 98,8% vastaavuudella ja tekstikuvalle 96,2% vastaavuuden verrattuna alkuperäiseen kuvaan.

L^2-normin tuloksia tarkasteltaessa pitää huomata, että kuvan resoluutio vaikuttaa tuloksen suuruuteen. Esimerkiksi bilineaarisella interpolaatiolla saatu kuva sammakosta sai erotuksen L^2-normiksi 758.6, mutta tekstikuvan erotuksen L^2-normi sai noin kaksi kertaa suuremman arvon 1472.6, vaikka SSIM tulokset eroavat vain noin kahdella prosenttiyksiköllä. Tämä johtuu siitä, että sammakkokuvassa on 640 000 pikseliä, mutta tekstikuvassa on vain 160 000 pikseliä. Tämän takia, kun lasketaan neliöjuurta pikselien erotusten neliöiden summasta, niin suuremmalla kuvalla odotettavasti saamme suurempia lukuja. Tarkasteltaessa L^2-normin antamia tuloksia teemme samat johtopäätökset kuin SSIM tuloksia tarkasteltaessa. PESR-algortimi antaa selvästi bilineaarista interpolointia parem-

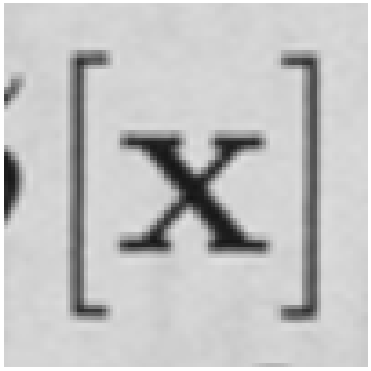
man tuloksen, mutta bilineaarisen ja bicubic-interpoloinnin välillä ei kovin suurta eroa ole sammakkokuvaa interpoloitaessa. Kuvaa tekstistä interpoloitaessa ero bilineaarisen ja bicubic-interpoloinnin välillä on jälleen suurempi.



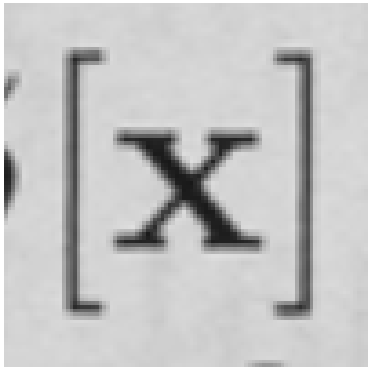
(a) Alkuperäinen



(b) PESR-algoritmi



(c) Bicubic



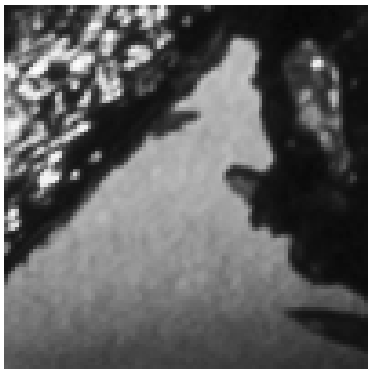
(d) Bilineaarinen

Kuva 4.1: Suurennoksia tekstikuvan puolikkaalle pienennökselle suoritettujen algoritmien tuloksista

Syy siihen, miksi kaikki algoritmit antavat huonompia tuloksia kuvalle tekstistä kuin sammakkokuvalle johtuu suurelta osin siitä, että ne toimivat heikoiten reunoilla. Tekstikuva sisältää käytännössä vaalean taustan lisäksi pelkästään reunoja, jos mielletään viivat reunoiksi. Tekstissä tulee suuri kontrastinvaihdos valkoisen taustan ja mustan kirjaimen vällillä. Jokainen algoritmi venyttää jokaista kahden erisuuruisen pikselin väliä kuvissa 3.3 kolmeksi pikseliksi, jolloin väliin interpoloitava harmaan sävy on täysin väärä. Sen tulisi olla mustempi tai valkoisempi, jotta se selvästi kuuluisi osaksi kirjainta tai taustaa. Tämä on nähtävissä kuvassa 4.1, jossa hakasulut ovat selvästi harmahtavampia interpoloiduissa kuvissa alkuperäiseen verrattuna. Kuvasta myös huomaa, että bicubic- ja bilineaarinen interpolaatio eivät tuota diagonaalien suuntaisia suoria kovin hyvin. Molem-

mat saavat selvästi näkyvää porrasmaista pikselöitymistä aikaan x-kirjaimessa. Bicubic-interpolaatiolla saadussa kuvassa tämä porrastuminen on kaikkein selvimmin huomattavissa. PESR-algoritmi sen sijaan suoriutuu todella hyvin myös diagonaaleilla suorilla, mikä on yksi syy, miksi sen saamat tulokset samankaltaisuusmitassa olivat parempia.

Myös sammakkokuvalle suoritettujen algoritmien tuloksista on valittu suurennos. Kuvassa 4.2a on alkuperäisestä kuvasta valittu 100×100 pikselin osakuva sammakon eturaajan yläosasta. Kuvissa 4.2b-4.2d on eri superresoluutiomenetelmillä saaduista tuloksista valittu samasta kohtaa 100×100 pikselin osakuva. Näistä suurennoksista huomaa, ettei yksikään metodi ole saanut aikaiseksi täysin samanlaista kuvaa alkuperäisen kanssa. Kaikki menetelmät tuottavat hieman sumennetun version alkuperäisestä. Jokaisessa kuvassa 4.2b-4.2d on tausta hyvin identtinen ja niissä sammakon varjon reuna on kuin tasainen siirtyminen mustasta vaaleanharmaaseen. Alkuperäisessä kuvassa varjon epätasaisuudesta voi jo päätellä, ettei sammakon pinta ole tasainen, mutta samaa päättelyä ei voi helposti tehdä superresoluutio-algoritmien tuloksista.



(a) Alkuperäinen

(b) PESR-algoritmi

(c) Bicubic

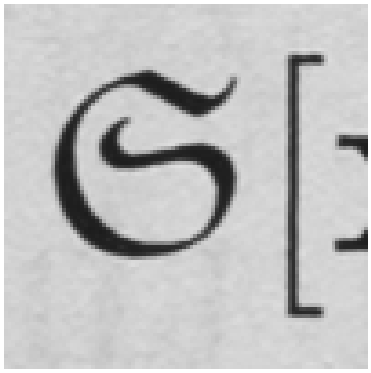
(d) Bilineaarinen

Kuva 4.2: Suurennoksia sammakkokuvan puolikkaalle pienennökselle suoritettujen algoritmien tuloksista

Toinen yksityiskohta, johon kiinnittää heti huomion on sammakon kyljen heijastama valkoinen valo. Alkuperäisessä kuvassa nämä valoläikät ovat hyvin teräviä ja kirkkaita, mutta jokaisessa superresoluutiomenetelmässä nämä laikut ovat haaleampia kuin alkuperäisessä kuvassa. Myöskin ne ovat enemmän ikään kuin sulautuneet osittain kiinni toisiinsa, eikä omia erillisiä valonheijastuksia.

Reunoja tarkasteltaessa on sama reunojen pehmentyminen huomattavissa sammakkokuvissa 4.2b-4.2d kuin huomasimme aiemmin kuvissa tekstistä 3.3b-3.3b. Sammakkokuvissa reunat ovat menettäneet hiukan terävyyttään, mutta ei niin paljon kuin kuvissa tekstistä.

Pienentämällä alkuperäisiä osakuvia 2.3 neljäsosakokoon ja suorittamalla algoritmit näille kuville, jotka näkyvät kuvassa 3.4, niin saamme aikaiseksi huomattavasti heikompia tuloksia. Algoritmien tuottamat kuvat tekstikuvalle näkyy kuvassa 3.6 ja näistä lasketut samankaltaisuusmitat sekä L^2-normin tulokset on tallennettu taulukkoon 3.4.



(a) Alkuperäinen

(b) PESR-algoritmi

(c) Bicubic

(d) Bilineaarinen

Kuva 4.3: Suurennoksia tekstikuvan nelinkertaiselle pienennökselle suoritettujen algoritmien tuloksista

Vertaamalla L^2-normin tuloksia tekstikuvalle kaksinkertaisella suurennoksella ja nelinkertaisella suurennoksella huomaamme, että bicubic-interpolaatio on heikentänyt tulostaan vähiten. Bicubic-interpolaation L^2-normi on kasvanut noin 42 prosentilla, kun bilineaarinen iinterpolaatio kasvatti normia noin 138 prosenttia ja PESR-algoritmin jopa 447 prosenttia.

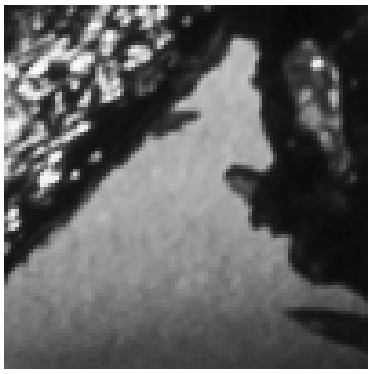
SSIM samankaltaisuusmitan tuloksia taulukosta 3.4 tarkastelemalla huomaamme samankaltaisen kasvun virheessä joka algoritmille. PESR-algoritmin tulos heikkeni noin 12 prosenttiyksiköllä, bilineaarisen interpolaation tulos heikkeni noin 18 prosenttiyksikköä ja bicubic-interpolaation tulos, joka heikentyi vähiten, heikentyi noin 11 prosenttiyksikköä. Näiden tulosten perusteella PESR-algoritmi toimii edelleen parhaiten, mutta bicubic-interpolaatio ei tuota heikoimpia tuloksia vaan nyt bilineaarinen interpolaatio antoi huonoimmat tulokset.

Tarkastelemalla kuvaa 4.3, jossa on suurennettuna $\mathfrak{S}$ kirjain, eri metodien tuloksista huomaa heti, miksi PESR-algoritmi on saanut selvästi paremman tuloksen samankaltaisuusmitassa kuin bilineaarinen tai bicubic-interpolaatio. Kirjain $\mathfrak{S}$ on selvästi tummempi PESR-algoritmilla saadussa kuvassa kuin muissa kuvissa, vaikka ero alkuperäiseen on jo aika suuri.

Yksikään esitetyistä superresoluutiomenetelmistä ei ole kunnolla pystynyt palauttamaan nelinkertaisesta pienennöksestä 3.4a kirjaimen $\mathfrak{S}$ kapeimpia kohtia. Tuloskuvia 3.6 katsomalla huomaa, kuinka $\mathfrak{S}$:n kapeimmat kohdat olisivat jopa katkonaisia, mutta alkuperäisestä kuvasta näkee, että näissä kohdissa pitäisi olla tummempi viiva. Tämä johtuu siitä, että pienennetyssä kuvassa nämä kapeat viivat ovat melkein hävinneet. Superresoluutioalgoritmit eivät osaa palauttaa sellaisia yksityiskohtia, joita ei ole kuvassa, jolle algoritmeja suoritetaan. Samoin on käynyt kirjaimen $\mathfrak{S}$, kuten muidenkin kirjainten, teräville kärjille. Kaikki terävät ominaisuudet näyttävät pyöristyneen paljon. Koska algoritmit interpoloivat valkoisen pisteen ja mustan pisteen välille harmaita sävyjä, niin kaikkein pienimmät yksityiskohdat, jotka sisältävät vähän pikseleitä vääristyvät paljon.

Sammakkokuvalle superresoluutioalgoritmit ovat tuottaneet parempia tuloksia nelinkertaisella suurennoksella. Tekstikuvan tulokset heikkenivät kaikilla superresoluutioalgoritmeilla yli 10 prosenttiyksikköä, kun verrattiin nelinkertaista suurennosta kaksinkertaiseen suurennokseen, mutta sammakkokuvan tilanteessa tulokset heikkenivät alle 6 prosenttiyksikköä.

Taulukon 3.3 mukaan PESR-algoritmi on edelleen paras menetelmistä, noin 97,4% vastaaavuudella alkuperäiseen kuvaan verrattaessa. Sammakkokuvallekin, kuten tekstikuvalle, bicubic-interpolaatio antaa paremman tuloksen kuin bilineaarinen interpolaatio nelinkertaisella suurennoksella. Bicubic-interpolaatio yltää noin 95,3% vastaavuuteen nelinkertaisella suurennoksella, joka ei ole kuin 3 prosenttiyksikköä huonompi vastaavuus kuin kaksinkertaisella suurennoksella. Heikoimman vastaavuuden tuotti bilineaarinen interpolaatio, joka pääsi vain noin 93,6% vastaavuuteen samankaltaisuusmitassa.



(a) Alkuperäinen

(b) PESR-algoritmi

(c) Bicubic

(d) Bilineaarinen

Kuva 4.4: Suurennoksia sammakkokuvan nelinkertaiselle pienennökselle suoritettujen algoritmien tuloksista

L^2-normissa tulokset ovat heikentyneet enemmän kuin samankaltaisuusmitassa. Katsomalla kuvaa 4.4 näkee, että suurennokset ovat entistä sumeampia versioita alkuperäisestä kuvasta, jonka takia yksittäisten pikselien erot kasvavat. Taulukon 3.3 tuloksia nelinkertaisille suurennoksille tarkasteltaessa huomaa PESR-algoritmin tuloksen kasvaneen kolminkertaiseksi kaksinkertaiseen suurennokseen verratessa. Bicubic- ja bilineaarisen interpolaatioiden tulokset kasvoivat kumpikin alle kolminkertaisiksi.

Nelinkertaisessa suurennoksessa yksikään superresoluutioalgoritmeista ei kykene palauttamaan pienimpiä epätasaisuuksia, jotka näkyvät hyvin alkuperäisen kuvan suurennoksessa 3.5a. Tämä oli oletettavissa kun katsoo kuvaa 3.4b, jossa on neljäsosakokoon alinäytteistetty kuva sammakosta. Tässä alinäytteistetyksessä on jo kadonnut pienimmät yksityiskohdat, joten niiden saaminen takaisin superresoluutioalgoritmeilla on mahdotonta ilman lisäoletuksia kuvasta.

Suurennoksia katsoessa kuvasta 3.5 näkee, että PESR-algoritmi toimii parhaiten tarkasteltaessa sammakon reunoja. Reunat pysyvät pyöreämpinä eikä samankaltaista por-

rasmaisuutta esiinny kuin kahdessa muussa algoritmissa. Bilineaarinen interpolaatio saa aikaan jo hyvin selvästi näkyvää porrasmaisuutta pikseleissä sammakon reunoilla ja halomaisuutta tasaisemmilla reunoilla. Bicubic-interpolaatio tuottaa vähemmän porrasmaisuutta sammakon reunalle, mutta bilineaarisen interpolaation tapaan sekin saa aikaan haloefektin sammakon reunoilla. PESR-algoritmi ei myöskään saa aikaan juuri ollenkaan haloefektiä, mutta nelinkertaisessa suurennoksessa reunat eivät ole myöskään aivan teräviä enää.

Molemmille kuville 2.1 ja 2.2 sekä niiden alinäytteistyksille käyttäen kaksinkertaista tai nelinkertaista suurennosta PESR-algoritmi antaa parhaimmat tulokset. Yhtä kuvaa suurentaessa kannattaa siis käyttää PESR-algoritmia eikä perinteisiä interpolaatiomenetelmiä, mutta PESR-algoritmin suoritus 800×800 resoluution kuvallekin kesti huomattavasti kauemmin kuin perinteriset bicubic- tai bilineaarinen interpolaatio. Bilineaarisen interpolaation suoritusaika 800×800 resoluution sammakkokuvalle oli vain muutama sekunti, joten sen käyttö olisi järkevää, jos haluaa suurentaa videokuvaa. Esimerkiksi videovalvonnassa käytettävissä kameroissa ei ole hyvä kuvanlaatu, joten näiden kuvaamien videoiden kuvanlaadun parantaminen olisi järkevää suorittaa bilineaarisella interpolaatiolla. Videokuvahan on kuvien esittämistä hyvin nopeassa tahdissa, usein 24 kuvaa sekunnissa. Algoritmia voisi suorittaa näille kuville yksitellen ja koota niistä uudestaan video. Bicubic-interpolaatio on taas hieman parempi kuin bilineaarinen interpolaatio kun käytetään nelinkertaista suurenosta. Ajallisesti bicubic-interpolaatio on huomattavasti nopeampi kuin PESR-algoritmi vaikka se ei olekaan yhtä nopea kuin bilineaarinen interpolaatio.

Kirjallisuutta

[1] Chris Solomon, Toby Breckon, *Fundamentals of Digital Image Processing*, Wiley-Blackwell, 2011.

[2] Paul Breeuwsma, *Cubic interpolation*, http://www.paulinternet.nl/?page=bicubic 2.12.2014

[3] Matthew Giassa, *Generalized Bicubic Interpolation*, http://www.giassa.net/?page_id=371 16.12.2014

[4] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, *Numerical Recipes in C - The Art of Scientific Computing*, Press Syndicate of the University of Cambridge, 1988.

[5] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli, *Image Quality Assessment: From Error Visibility to Structural Similarity*, IEEE Transactions on Image Processing, 2004.

[6] Michal Aharon, Michael Elad, Alfred Bruckstein, *K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation*, IEEE Transactions on Image Processing, VOL. 54, NO. 11 2006.

[7] Tomer Peleg, Michael Elad, *A Statistical Prediction Model Based on Sparse Representations for Single Image Super-Resolution*, IEEE Transactions on Image Processing, VOL. 23, NO. 6 2014.

[8] Guoshen Yu, Guillermo Sapiro, Stéphane Mallat, *Solving Inverse Problems with Piecewise Linear Estimators: From Gaussian Mixture Models to Structured Sparsity*, IEEE Transactions on Image Processing, VOL. 21, NO. 5 2011.

[9] Y. C. Pati, R. Rezaiifar, P. S. Krishnaprasad, *Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition* in Proc. 27th Asilomar Conf. Signals, Systems, and Computers, 1993