Jämförelse och genomgång av inexakta bildkomprimeringformat med stöd i webbläsare

Josef Nylund
Kandidatavhandling i datavetenskap
Handledare: Sebastien Lafond
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
2022

Innehåll

Sammanfattning

1 Inledning

Bilder används överallt nuförtiden, inte minst på internet och på webbsidor. För att spara på lagringsutrymme och bandbredd och samtidigt få webbsidor att ladda snabbare komprimeras nästan alltid alla bilder på nätet. Utan någon komprimering alls skulle det vara opraktiskt och ibland helt omöjligt att använda bilder i så stor grad som det görs idag. Därför är det viktigt att det finns sätt att minska på bildernas storlek utan att sänka kvaliteten mer än nödvändigt.

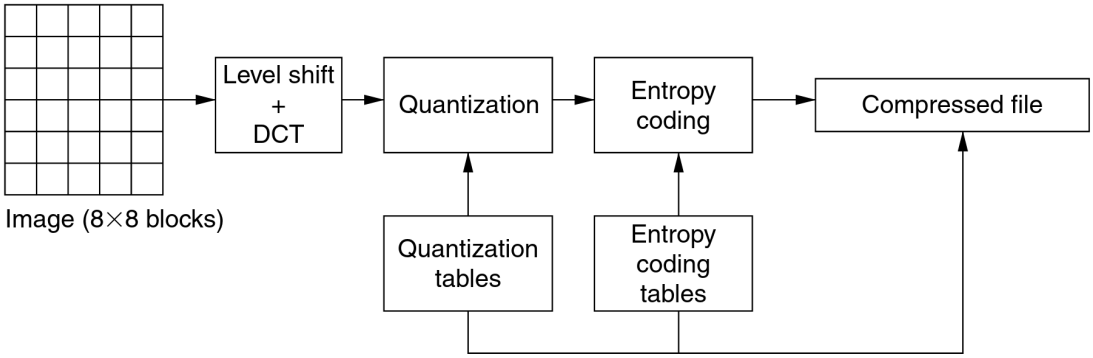
Bilder kan komprimeras med antingen förlustfria eller inexakta (eng. lossless och lossy) komprimeringsmetoder. Förlustfria komprimeringsalgoritmer behåller bildens utseende exakt under komprimeringen (alla bitar är exakt samma efter avkodningen som innan komprimeringen), medan inexakta komprimeringsalgoritmer förändrar utseendet delvis, till förmån för effektivare komprimering.

I denna avhandling kommer fyra bildkomprimeringsformat tas upp, beskrivas och jämföras. Alla bildkomprimeringsformat som tas upp i avhandlingen har stöd i åtminstone några av de nuvarande största webbläsarna. För att begränsa omfattningen kommer den här avhandlingen endast att fokusera på inexakta metoder för bildkomprimering.

2 Bakgrund

Målet med alla bildkomprimeringsformat är att beskriva en bild på ett annat sätt än att endast rada upp värdena för varje pixel en och en. Exakt hur det sker beror på komprimeringsformatet som används, men ofta delas bilden upp i mindre block som var och en beskrivs som en kombination av ett antal olika mönster. Den mest detaljrika informationen kan då enkelt tas bort genom att ta bort de mera komplicerade mönstren medan bildens helhet ändå till största del behålls lika. Detta möjliggör en kraftig minskning av bildens storlek.

Innan en bild kan komprimeras behöver färgrymden konverteras från RGB till YC_bC_r. Alla bildformat i denna avhandling använder inte YC_bC_r utan istället någon annan färgrymd och mera om det kommer tas upp då de formaten presenteras. I RGB-färgrymden har varje pixel ett R-, G- och B-värde som representerar hur röd, grön, respektive blå den pixeln är. I YC_bC_r-färgrymden har varje pixel ett Y-värde som står för luminansen (eller luma), dvs. ljusstyrkan, och två C-värden som står för krominans (eller chroma) och beskriver färgen. Det bör noteras att även om C_b och C_r står för blå-skillnad och röd-skillnad (eng. blue-difference och red-difference) är dessa värden inte är samma röda och blåa värden



Figur 1: Schemat för JPEG-komprimering. Schemat antar att bilden är svartvit. Om bilden inte är svartvit så kommer den först konverteras till YC_bC_r-färgrymd och de tre lagren kommer gå igenom schemat enskilt. *Källa:* [9]

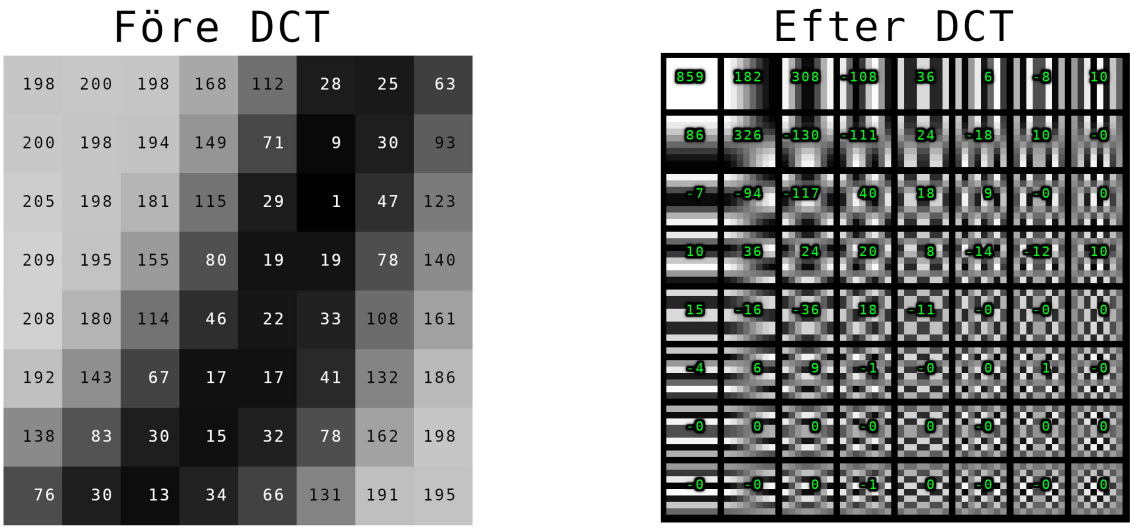
som i RGB-färgrymden. Y, C_b och C_r behandlas sedan helt skilt från varandra. C-värdena behandlas ofta också på ett lite annat sätt då människan inte är lika känslig för färg som för ljusstyrka. Det ger en möjlighet att komprimera dem ännu mera utan några större visuella brister.

De vanliga stegen i ett komprimeringsschema [9] är en dekorrelator (eng. decorrelator), en kvantiserare (eng. quantizer) och ett entropi-kodningssteg (eng. entropy coding stage). Allmänt kan man säga att en dekorrelator omformulerar beskrivningen av bilden från bara pixlar till något annat, t.ex. våglängder, en kvantiserare tar bort en del information för att möjliggöra komprimering och det sista steget, entropi-kodningssteget komprimerar det som återstår på ett effektivt sätt. Implementationen av de här stegen varierar mellan olika format, men alla format behöver inte nödvändigtvis följa denna paradigm heller. De nyare formaten har alla också fler steg och funktioner som gör komprimgeringsschemat mera komplicerat.

2.1 JPEG

JPEG-formatet är det äldsta av de bildkomprimeringsformat som kommer tas upp, men formatet är också det populäraste inexakta bildkomprimeringsformatet på internet. Ungefär 74 % av alla webbsidor använder bilder i jpg-format enligt w3techs.com [1]. Den första versionen av JPEG släpptes 1992 och den senaste versionen är från 1994 [4]. JPEG stöder fyra olika operationslägen men endast det vanligaste läget, sekventiellt, kommer tas i beaktande. De andra lägena är progressivt, hierarkiskt och förlustfritt.

Komprimeringsprocessen följer de steg som togs upp ovan och schemat kan ses i figur 1. Först delas bilden upp i luma och chroma och de tre bildlagren delas sedan upp i block på 8 x 8 pixlar. De resterande stegen i algoritmen körs sedan blockvis. Nästa



Figur 2: Ett 8x8 block pixlar konverteras till cosinusvågor med hjälp av DCT. I vänstra blocket beskriver värdena ljusstyrkan (eller krominansen) och i högra blocket beskriver värdena koefficienterna för cosinusvågorna.

steg är dekorrelationssteget och i JPEG-formatet består det av en DCT. DCT står för diskret cosinustransformation (eng. discrete cosine transformation). En DCT konverterar blocket så att det beskrivs av koefficienter av 64 olika cosinusvågor och i figur 2 syns denna process. Intressant nog är detta steg helt förlustfritt eftersom alla 8 x 8 block går att beskriva förlustfritt med 64 stycken cosinusvågor med olika frekvens och amplitud.

Det tredje steget är en kvantiserare. Kvantiseringssteget dividerar multipelvärdet för varje cosinusvåg med värde i en kvantiseringstabell. Vilken kvantiseringstabell som används påverkar kvalitetsgraden för bilden. Kvaliteten går mellan 1 % och 100 % och varje värde har en egen kvantiseringstabell. Kvantiseringstabellen har större tal nere och till höger i tabellen vilket gör att de mera detaljrika delarna av bilden minskar mer än de övriga. Då bilden dekomprimeras multipliceras varje värde med respektive värde i kvantiseringstabellen, men på grund av att värdena endast sparas som heltal försvinner en del information. Dessutom, om värdet blir noll efter kvantiseringen går det inte att få tillbaka något data alls. Det här är det enda steget i processen som sänker bildkvaliteten. Efter det skrivs tabellen om till en lång sträng enligt ett sicksackmönster. Sicksackmönstret börjar uppe i vänstra hörnet, går genom hela bilden och slutar sedan i nedre högra hörnet. Orsaken att ett sicksackmönster används är att man försöker få de lägsta talen att komma sist i talsträngen.

Det sista steget är Huffmankodning eller aritmetisk kodning. Huffmankodning är den

vanligare av de två, men har oftast lite sämre komprimeringsförmåga. Aritmetisk kodning var skyddad av patent då JPEG släpptes, men även om patentet numera gått ut har Huffmankodning fortsatt dominera i JPEG-komprimering. Tack vare kvantiseringen och sicksackordningen finns det nu långa strängar med endast 0:or i slutet av de flesta block. Alla blocksträngar radas sedan upp till en enda lång sträng och strängen kodas om på ett effektivare sätt. På så vis sjunker filstorleken avsevärt. Det som kommer ut på andra sidan komprimeringsprocessen är en färdig JPEG-fil.

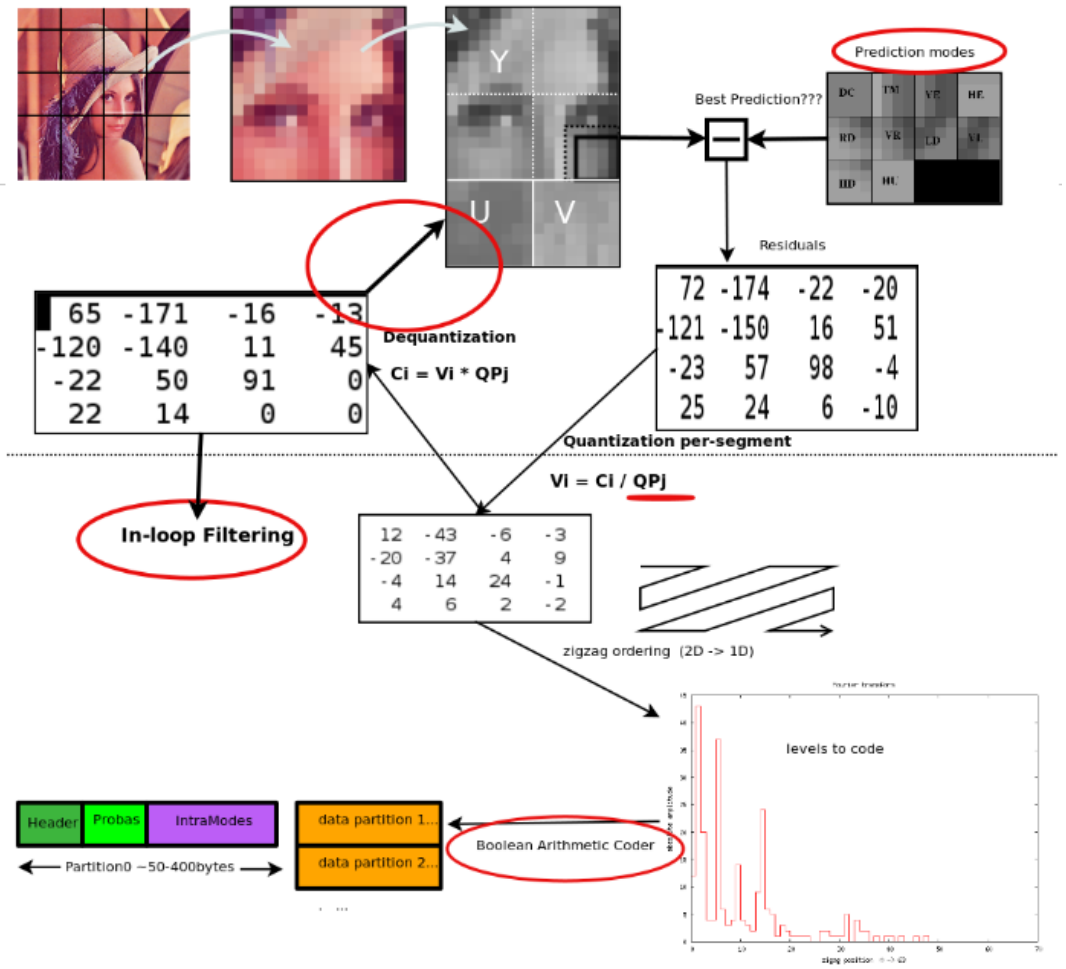
2.2 WebP

Bildformatet WebP [2] är i stor grad likt JPEG, men det har några tillägg som förbättrar komprimeringsförmågan. WebP är skapat av Google och första versionen släpptes 2010. Målet med WebP var att ersätta JPEG på webben genom att komprimera effektivare utan att försämra bildkvaliteten samt genom att stöda genomskinliga bilder, vilket är något som JPEG inte gör. Det kan noteras att WebP också stöder förlustfri komprimering som fungerar på ett helt annorlunda sätt.

Komprimeringsschemat för WebP kan ses i figur 3. Precis som JPEG konverteras bilden först till luma och chroma och sedan delas den upp i tre lager (ett luma-lager och två chroma-lager). Lagren delas sedan upp i block som har storleken 16x16 pixlar, men dessa delas sedan vidare in i mindre block på 4x4 pixlar. Upplösningen för krominansblocken halveras till 8x8 genom så kallad ”chroma-subsampling”. ”Chroma-subsampling” halverar mängden data som behövs för färginformationen till kostnad av sänkt färgkvalitet. Det märks däremot knappt eller inte alls tack vare att människans ögon inte är lika känsliga för färger som förändringar i ljusstyrka.

En skillnad mellan WebP och JPEG uppstår i nästa steg då pixlar utanför blocket används för att uppskatta hur blocket kommer se ut. De pixlar som finns på vänstra och övre kanten, samt övre vänsta hörnet av blocket, används för uppskattningen. Det finns ett antal metoder för att uppskatta innehållet i blocket och den metod som ger bästa resultat för den rutan används. I figur 4 syns denna uppskattning. Innehållet i uppskattningsblocket subtraheras från det verkliga innehållet i blocket och endast resten, samt information om vilken uppskattningsmetod som användes, går vidare i processen. Nästa steg är en DCT och steget är likadant som samma steg i JPEG-komprimeringen.

Efter det kommer nästa förändring, WebP stöder uppdelning av bilden i olika segment som har liknande visuella egenskaper. De olika segmenten kan således ha olika kvantiseringssteg och filterstyrka. Blocket kommer då divideras enligt den kvantiseringstabell som används för det segementet. Direkt efter det kommer blocket att multipliceras med samma kvantiseringstabell och dessa rekonstruerade pixelvärden kommer användas i uppskattningen för kommande block. (De ursprungliga pixlarna används inte för upp-

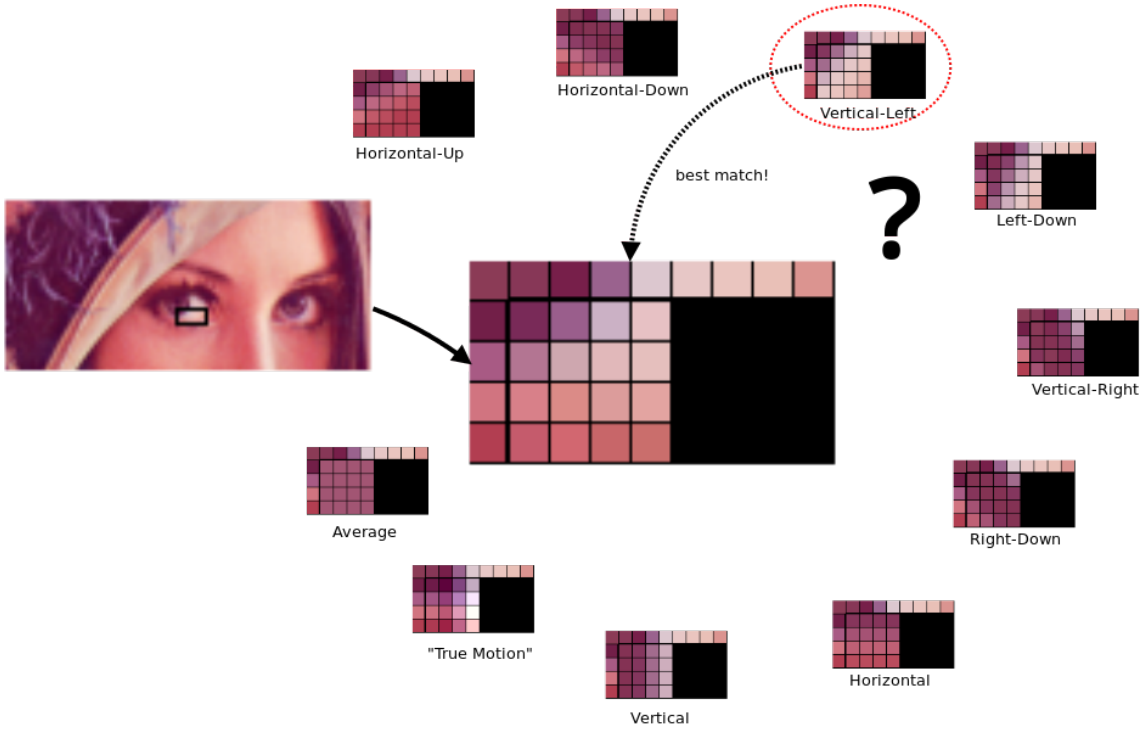


Figur 3: Komprimeringschemat för WebP. De delar som skiljer sig från JPEG är markerade i rött. *Källa:* [2]

skattningen för att de inte lagras i den slutliga filen och de behövs för rekonstrueringen av bilden då den avkodas.) Det kvantiserade blocket ordnas sedan enligt ett sicksackmönster enligt samma princip som i JPEG-formatet. Det sista steget är att komprimera alla block med aritmetisk kodning. Aritmetisk kodning komprimerar i regel lite bättre än Huffmankodning som JPEG oftast använder.

2.3 AVIF

AVIF kommer från videokomprimeringsformatet AV1. AVIF består av en HEIF-behållare som innehåller en AV1-video. Videon består oftast av en enda bild, men då formatet stöder animationer kan längre videor också förekomma. AVIF och AV1 är båda skapade av gruppen AOMedia [12] som står för ”Alliance for Open Media”. AV1-formatet är ett relativt nytt videokomprimeringsformat och 2019 släpptes första versionen av AV1 och AVIF. Målet med AV1 är att genom sin stora komprimeringsförmåga och öppna licens bli



Figur 4: Olika metoder för att uppskatta pixelvärdena i 4x4-blocket. I det här exemplet gav uppskattningen ”Vertical-Left” bäst uppskattning och därmed väljs den för detta block. *Källa:* [2]

den populäraste videokodeken som används. Denna komprimeringsförmåga kommer med nackdelen att komprimeringskedjan är mycket lång och komplicerad. Hela den processen används också i AVIF. Om AVIF-filen bara består av en bild och inte en animation så förenklas processen något, men den är fortfarande komplicerad. På grund av denna komplexitet kommer detta kapitel inte att gå in på detalj hur hela komprimeringsprocessen ser ut, utan endast ta upp de metoder som används för att förbättra komprimeringen.

TODO...

2.4 JPEG XL

JPEG XL [8] är ett nytt bildformat skapat av JPEG kommittén och formatet är, vid skrivande stund, fortfarande under utveckling. 2018 skedde ”Call for Proposals” för JPEG XL [5] och formen för filformatet standardiserades i slutet av 2021 [6]. Trots att formatet inte är helt färdigt ännu finns det redan stöd för JPEG XL-bilder bl.a. i webbläsarna Google Chrome och Mozilla Firefox (endast i ”Nightly”-versionen). Funktionaliteten är däremot avstängd som standard och behöver aktiveras på webbläsarnas configurationssidor.

Tanken bakom JPEG XL är, precis som för många andra format, att ersätta JPEG som det mest använda inexakta bildformatet på internet genom att erbjuda bättre komprimeringsförmåga för samma bildkvalitet jämfört med JPEG. Dessutom har JPEG XL

fler fördelar, bl.a. förlustfri konvertering till och från JPEG, animationer, snabb kodning och dekodning och ett fokus på att fungera som bäst för bilder med högre bittakt (eng. bit rate). Det sistnämnda är av stor betydelse. Forskarna bakom JPEG XL anser att tidigare bildformat har satsat på att göra bildkomprimeringen så effektiv som möjligt vid alldeles för låga bittakter och det har lett till att det mesta av effektiviteten förloras då det skalas upp till bilder med högre bittakt [8]. Det här stöder forskarna med empirisk data från Google Chrome som visar att medelvärdet för antalet bitar per pixel (bpp) ligger mellan 2 och 3 för bilder på internet. Deras mål blev således att kunna producera bilder med 1 bpp utan synliga artefakter istället för bilder med 0,5 bpp som innehåller ickestörande artefakter.

Komprimeringsprocessen är i stort sett lika som JPEG, men med uppgraderingar till de flesta steg. Det första steget är att konvertera färgrymden till en så kallad XYB-färgrymd. Denna färgrymd är speciellt skapad för JPEG XL. XYB använder sig av ”opponent-process”- och ”trichromatic”-färgteorierna och har färgrymden LMS som bas. LMS färgrymden försöker representera hur de tre olika konerna i en människas öga reagerar på ljus i olika våglängder. L står för långa våglängder (ung. rött ljus), M för medium våglängder (ung. grönt ljus) och S för korta våglängder (ung. blått ljus). ”Opponent-process”-teorin används för L- och M-värdena medan ”trichromatic”-teorin används för S-värdena . I praktiken betyder det här att Y-värdet i XYB-färgrymden består av summan av L och M, X-värdet består av differensen mellan dem, medan B har samma värde som S. Orsaken till det här att att då kan man separat kontrollera kvantiseringsnivån och dekorellationen för S receptorn och det ger möjlighet att komprimera B-värdet mera än X- och Y-värdena.

Nästa steg är DCT. JPEG XL använder sig av variabel-DCT och det gör att storleken på blocken kan variera. Sidlängderna kan fritt variera mellan 8, 16 och 32 pixlar. Sidorna behöver inte heller ha samma längd, d.v.s. blocken behöver inte vara kvadratiska. Genom att tillåta större block kan man effektivare komprimera större områden i bilden med liknande utseende. Nackdelen med större block är att de artefakter som kan uppstå blir större och synligare. Det finns också stöd för block med längerna 8 x 4, 4 x 8, 4 x 4 och 2 x 2. Med desssa block minskar utsträckningen för de arftefakter som bildas, men på grund av att blocken är så små möjliggör de inte lika stor komprimeringsförmåga som block med större storlek. Om områden på bilden med ett homogena utseenden får stora block och områden med mycket detaljer får mindre block kan bilden komprimeras effektivt samtidigt som detaljer bibehålls. En annan teknik JPEG XL har är en så kallad ”INDENTITY”-transformation. Om några pixlar i ett block har väldigt annorlunda värden är sina närliggande pixlar kan en sådan transformation köras istället för en DCT. Alla värden i blocket, förutom det på plats (1, 1), kommer att skrivas om som skillnaden (deltat) mellan det värdet och det på (1, 1). TODO... En tredje teknik som kan används

i DCT-skedet är en AFV-transformation. AFV-transformationen används om några pixlar i ett hörn av ett block har helt andra värden än resten av blocket. AFV-transformationen kör en DCT på blocket som vanligt men låter värdena på de tre pixlarna i ett av hörnen sparas skilt från resten av blocket.

Kvantiseringen är ganska lik den i JPEG. Endast en kvantiseringstabell väljs, precis som i JPEG, men i motsats till WebP som tillåter olika kvantiseringstabeller för olika områden i bilden. Däremot kan varje block välja att skala tabellen för att minska på artefakter i vissa områden. Kvantiseringen av DC-värdena (De värden som beskrivs av en cosinusvåg med frekvensen noll) beskrivs i högre detalj om de är nära 0. DC-värdena sparas inte direkt utan endast ett deltavärde från en annat blocks DC-värde sparas med hjälp av uppskattning. Ett DC-deltavärde nära noll betyder då att det egentliga värdet är mycket likt, men inte helt identiskt som ett block bredvid. Sådana här förändringar skapar ”banding” i JPEG-formatet, men det förhindras tack vare att DC-deltavärden nära noll kan beskrivas noggrannare i JPEG XL.

Till sist entropikodningen... TODO...

3 Hur kan man jämföra bildkvalitet objektivt?

Då bilder komprimeras med inexakta komprimeringsmetoder försvinner alltid en del information under komprimeringen. Mängden information som försvinner kan vara så pass liten att bildens utseende knappt eller inte alls förändras för ett mänskligt öga. Då komprimeringen blir grövre blir dessa artefakter mer märkbara och de kan vara tillräckligt synliga att de stör när man ser på bilden. Däremot är det inte tydligt när denna förändring sker för att det här handlar om människors subjektiva syner på bilder.

Ett sätt att försöka mäta kvalitet för komprimerade bilder är att man ber en tillräckligt stor grupp människor att poängsätta bilder av olika komprimeringsgrad. Medeltalet av poängsättningarna för varje bild borde då ge en ungefärlig bild av hur hög kvalitet bilden har. En styrka med den här metoden är att då bildkvalitet i grunden beror på hur en människa ser en bild så borde resultatet stämma bra överens med verkligheten. Den här metoden har tyvärr också nackdelar. Den första nackdelen är att man behöver fråga tillräckligt många människor för att resultatet ska vara pålitligt. Den andra nackdelen är att samma bild inte nödvändigtvis får samma poäng om den testas många gånger. Variationen i poängsättningen borde minska då antalet människor som poängsätter ökar. En sista nackdel är att metoden inte är snabb och inte går att automatisera. Därför behövs det ett sätt att objektivt mäta bildkvalitet, snabbt och enkelt.

Det finns ett flertal olika metoder för att mäta bildkvalitet objektivt och i här kommer två av dem att presenteras.

3.1 PSNR

PSNR står för peak signal-to-noise ratio. PSNR är en enkel metod för att beräkna bildkvalitet för en bild. Det enda processen gör är att jämföra skillnaden mellan pixelvärdena före och efter komprimeringen och beroende på hur mycket de skilijer, ge mer eller mindre poäng. Den här metoden är mycket enkel att implementera, men den tar inte i beaktande hur människor uppfattar bilder.

Först beräknas ett MSE-värde för bilden. MSE står för mean squared error (medelfelet i kvadrat). Vi låter a vara en komprimerad bild och b vara originalbilden, där a(x,y) och b(x,y) står för pixelvärdet för en pixel i a-, respektive b-bilden. MSE beräknas med formeln $MSE(a,b) = \frac{1}{MN}\sum_{m=0}^{M}\sum_{n=0}^{N}[a(m,n)-b(m,n)]^2$. M och N är bredden och höjden för bilderna. Om bilderna inte har samma bredd och höjd behöver den komprimerade bilden skalas om så att den matchar originalbildens storlek.

PSNR fås sedan från formeln $PSNR(a,b) = 10log_{10}\frac{peakval^2}{MSE(a,b)}$. peakval står för det största värdet en pixel kan ha, t.ex. 255 om bilden är en jpeg med 8-bitars färgdjup.

3.2 SSIM

SSIM står för structural similarity. SSIM är mera avancerat än PSNR och jämför de två bilder den tar in på tre sätt, luminans, kontrast och struktur.

Först jämförs medelluminansen mellan bilderna enligt formeln $l(x,y) = \frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1}$ där μ_x och μ_y är medelluminansen för bilderna x och y. Sedan subtraheras varje bilds medelluminans (μ_x och μ_y) från varje värde i båda bilderna och processen går vidare till nästa steg där kontrasten jämförs.

Kontrasten bestäms enligt formeln $c(x,y) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2}$ där $\sigma_x = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)^2}$ (dvs standardavvikelsen). Sedan divideras varje värde i bilderna med den bildens standarvavvikelse. Det gör att värdena blir standardfördelade, dvs väntevärdet är 0 och standardavvikelsen är 1.

Det tredje steget är att beräkna en strukturjämförelse och det görs med formeln $s(x,y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$. σ_{xy} beräknas på precis samma sätt som Pearsons korrelationskoefficient [10], dvs $\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu_x)(y_i-\mu_y)$.

Till sist kommer de tre värdena l, c och s att kombineras till ett värde enligt formeln $SSIM(x,y) = l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma$ och då fås SSIM-värdet. C_1, C_2 och C_3 är konstanter som används för att unvika instabiliteter då vissa beräkningar närmar sig 0. α, β och γ är parametrar (alltid > 0) som bestämmer den relativa vikten mellan de tre komponenterna. SSIM-värdet kommer alltid att variera mellan 0 och 1, där 1 betyder att båda bilderna är helt identiska och ju lägre talet är, desto mer skiljer sig bilderna från varandra.

4 Metod för jämförelse av bildkomprimeringsformat

I denna sektion kommer de bildformat som tagits upp testas för att jämföra vilka kompromisser det existerar mellan komprimeringsförmåga och bildkvalitet. För testningen har ett antal bilder med tillräckligt stor variation valts.

För JPEG-formatet valdes Mozillas JPEGkodare MozJPEG. Den här kodaren används för att den erbjuder bättre komprimeringsförmåga än många andra JPEGkodare och skulle därmed visa det bästa JPEG kan erbjuda. Guetzli [7] är en JPEGkodare från Google, men den har visat sig ge sämre resultat i SSIM-mätningar (citation needed). Det verkar vara oklart om MozJPEG eller Guetzli ger bättre resultat i allmänhet då de är optimerade för olika kvalitetsgranskningar. MozJPEG kan optimera bilden för PSNR och SSIM medan Guetzli anpassar sig enligt en ny metod, också skapad av Google, kallad Butteraugli. För att Mozjpeg ger bäst resultat för PSNR och SSIM valdes den för den här jämförelsen.

De andra kodarna har inte lika många alternativ. För WebP används cwebp av Google [2]. För AVIF används libavif (version 0.9.0) [13] och för JPEG XL används cjxl (version 0.6.1) från libjxl [3].

Bilderna har 8-bitars färgdjup och hämtades från imagecompression.info [11]. Samlingen bilder är skapad för just det här ändamålet, dvs jämföra bildkvalitet av komprimerade bilder. Bilderna är valda för att de ”stresstestar” komprimeringsalgoritmer på olika sätt. Denna bildsamling har sett användning i andra test, t.ex. Jyrki Alakuijala m. fl. använder bildsamlingen i artikeln ”JPEG XL next-generation image compression architecture and coding tools” [8].

5 Resultat

I figur 5 syns resultaten över jämförelerna för en av bilderna.

6 Diskussion

Källförteckning

[1] URL: https://w3techs.com/technologies/details/im-png (hämtad 2022-02-20).

[2] URL: https://developers.google.com/speed/webp/docs/compression (hämtad 2022-02-22).

[3] URL: https://github.com/libjxl/libjxl/ (hämtad 2022-03-24).

[4] The Joint Photographic Experts Group (JPEG). URL: https://jpeg.org/jpeg/ (hämtad 2022-02-20).

[5] The Joint Photographic Experts Group (JPEG). URL: https://jpeg.org/downloads/jpegxl/jpegxl-cfp.pdf (hämtad 2022-03-15).

[6] The Joint Photographic Experts Group (JPEG). URL: https://www.iso.org/standard/80617.html (hämtad 2022-03-15).

[7] Jyrki Alakuijala m. fl. ”Guetzli: Perceptually guided jpeg encoder”. I: *arXiv preprint arXiv:1703.04421* (2017).

[8] Jyrki Alakuijala m. fl. ”JPEG XL next-generation image compression architecture and coding tools”. I: *Applications of Digital Image Processing XLII*. Utg. av Andrew G. Tescher och Touradj Ebrahimi. Vol. 11137. International Society for Optics och Photonics. SPIE, 2019, s. 112–124. DOI: 10.1117/12.2529237. URL: https://doi.org/10.1117/12.2529237.

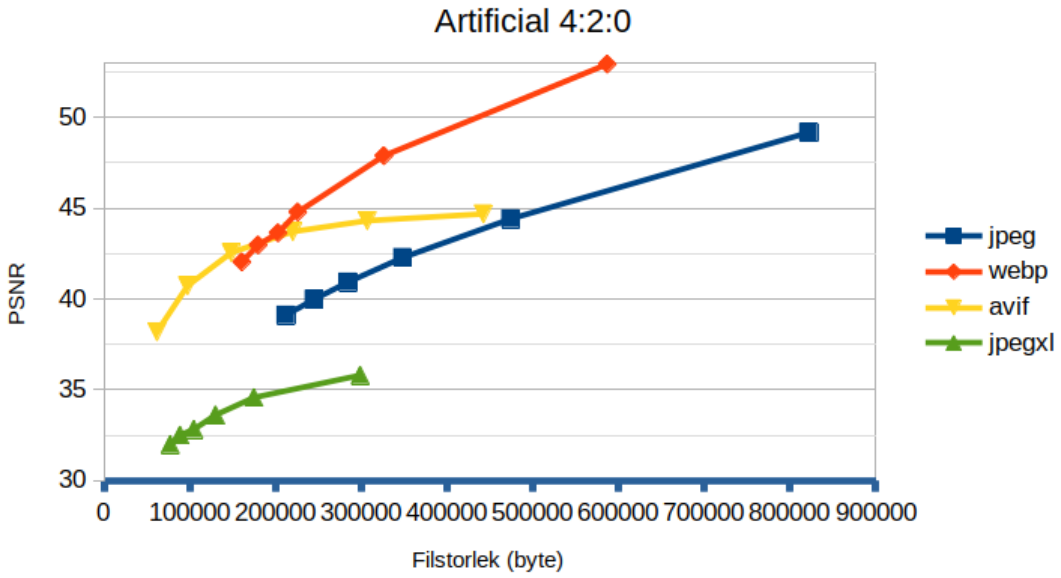
[9] M. (Ed.) Barni. *Document and Image Compression (1st ed.)* CRC Press, 2006, s. 4, 89–94. DOI: https://doi.org/10.1201/9781315221298.

[10] Jacob Benesty m. fl. ”Pearson correlation coefficient”. I: *Noise reduction in speech processing*. Springer, 2009, s. 1–4.

[11] Sachin Garg. *Image compression benchmark*. URL: imagecompression.info (hämtad 2022-03-24).

[12] The Alliance for Open Media. URL: https://aomedia.org/about/ (hämtad 2022-03-22).

[13] The Alliance for Open Media. URL: https://github.com/AOMediaCodec/libavif/ (hämtad 2022-03-24).



Figur 5: Figur över hur kvaliteten varierar mellan de olika bildformaten vid olika filstorlekar. OBS. bilden är troligen felaktig... TODO