HO
GENT

Faculteit Bedrijf en Organisatie

Je kijkt er naar, maar ziet het niet: datacompressie principes — ontstaan en uitdagingen — implementaties — afbeeldingscompressie: PNG | JPEG | JPEG2000 | WebP | HEIF — videocompressie: H.264-AVC | H.264-SVC | H.265/HEVC | AV1

Bontinck Lennert

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Wim De Bruyn
Co-promotor:
Tom Paridaens

Instelling: —

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Je kijkt er naar, maar ziet het niet: datacompressie principes — ontstaan en uitdagingen — implementaties — afbeeldingscompressie: PNG | JPEG | JPEG2000 | WebP | HEIF — videocompressie: H.264-AVC | H.264-SVC | H.265/HEVC | AV1

Bontinck Lennert

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Wim De Bruyn
Co-promotor:
Tom Paridaens

Instelling: —

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

De zoektocht naar een leuk, leerrijk en bruikbaar bachelorproefonderwerp is niet makkelijk. Wanneer ik echter op zoek was naar een geschikt afbeeldingsformaat voor het bewaren van mijn steeds groeiende afbeeldingscollectie was ik stomverbaasd hoe weinig ik over dit onderwerp wist. Ook mijn vrienden, waarvan vele medestudenten Toegepaste Informatica te HoGent, konden mij geen antwoord geven op de vraag welk afbeeldingsformaat een goede keuze zou zijn en kenden buiten het feit dat PNG wel een transparante achtergrond kan hebben en JPEG niet, geen echte verschillen tussen deze twee bekendste afbeeldingsformaten.

Ook op mijn stageplaats, waar websites en webshops gemaakt en onderhouden worden, waren maar weinig mensen zich echt bewust van de voordelen en nadelen van de verschillende afbeeldingsformaten. Meestal exporteerden ze afbeeldingen vanuit Adobe Photoshop met een kwaliteitsinstelling zodanig het eindbestand kleiner was dan 100kb om de performance van een website hoog te houden. De keuze voor een afbeeldingsformaat anders dan JPEG of PNG wordt hier en in het algemeen te weinig overwogen ondanks dat hier een grote kwaliteitswinst gedaan kan worden met dezelfde of kleinere bestandsgrootte.

Deze onwetendheid deed mij beseffen dat een onderzoek naar de verschillende soorten datacompressie interessant kon worden. Zowel voor mede programmeurs, die hun CSS minifyen om enkele kilobytes te besparen maar niet stilstaan hoeveel data ze kunnen besparen door het kiezen van een gepast afbeeldingsformaat, als content creators en tal van andere personen in de IT-wereld.

De achterliggende wiskunde en boeiende uitdagingen zoals DNA compressie wisten mij ook te overtuigen dat dit onderwerp zeer verreikend zou zijn voor mij. Het vinden van een geweldige co-promotor, vakexpert Tom Paridaens, was dan ook de kers op de taart.

Graag bedank ik dan ook mijn co-promotor, Tom Paridaens, voor de nauwe samenwerking binnen deze bachelorproef. Ook bedank ik graag Wim De Bruyn voor de leerrijke lessen onderzoekstechnieken die ons de nodige kennis brachten voor het voeren van een gegrond onderzoek. Bovendien is Wim De Bruyn de promotor van deze bachelorproef waarvoor ik hem ook wil danken.

Ook wil ik Mayté Bogaert, van MaytéB fotografie, bedanken voor het aanleveren van meerdere RAW bestanden die ik heb gebruikt voor het uitvoeren van mijn onderzoek.

Bert Van Vreckem en collega's verdienen ook een dankwoord voor het opstellen van een LaTeX sjabloon voor deze bachelorproef. Tot slot bedank ik graag de deelnemers die aan de hand van mijn afbeeldingsevaluatietool hebben bijgedragen naar het onderzoek van een geschikt afbeeldingsformaat voor de use case besproken in deze bachelorproef.

Samenvatting

Datacompressie: een fundamenteel onderdeel van de IT-wereld waar weinig belanghebbenden een basiskennis van hebben. Waarom is dat? Schrikken de complexe en zeer uitgebreide papers reeds geschreven over dit onderwerp geïnteresseerden af? Is het een te complex onderwerp om te voorzien in meer IT-gerelateerde opleidingen? Staat datacompressie stil in de tijd dat standaarden als het afbeeldingsformaten JPEG al meer dan twintig jaar het bekendste afbeeldingsformaat is? Deze bachelorproef tracht een antwoord te geven op die vragen en de tal van andere onderzoeksvragen besproken in deel 1.2.

Dit document is gemaakt met een eenvoudig visie: de nodige basiskennis over het ontstaan van datacompressie, de werking van enkele compressie-algoritmen, tal van afbeeldingsformaten en video codecs en de manieren voor het evalueren van compressiekwaliteit toe te lichten. Na het lezen van deze bachelorproef zal de lezer meer stilstaan bij de keuze voor een geschikt compressie-algoritme.

Er is zowel een proof of concept datacompressietool als een uitgebreide afbeeldingsevaluatietool geschreven voor de bachelorproef die gratis zijn in gebruik en open source toegankelijk zijn op de GitHub repository van deze bachelorproef[1]. Er wordt dieper ingegaan op het ontstaan, de werking en de voordelen en nadelen van volgende afbeeldingsformaten: PNG | JPEG | JPEG2000 | WebP | HEIF. Hetzelfde wordt gedaan voor de volgende video codecs: H.264-AVC | H.264-SVC | H.265/HEVC | AV1.

Er wordt een subjectief onderzoek gevoerd aan de hand van de eerder benoemde afbeeldingsevaluatietool. Dit geeft samen met de theoretische kennis die zal verkregen worden door de bachelorproef een antwoord op de hoofdonderzoeksvraag van deze bachelorproef: Waarom moet er stilgestaan worden bij het gebruiken van compressie-algoritmen, hoe kies

[1] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

je een geschikt compressie-algoritme voor een bepaalde use case en hoe implementeer je dit het best?

Het volledige verloop van deze bachelorproef is beschreven in deel 1.4.

Inhoudsopgave

1 Inleiding 13

1.1 Probleemstelling 14

1.2 Onderzoeksvragen 14

1.3 Onderzoeksdoelstelling 15

1.4 Opzet van deze bachelorproef 15

1.4.1 Deel 1: situering en literatuurstudie 15

1.4.2 Deel 2: datacompressietool ontwikkelen 15

1.4.3 Deel 3: afbeelding- en videocompressie 16

1.4.4 Deel 4: onderzoek afbeeldingscompressie 16

1.4.5 Deel 5: uitdagingen en conclusie 16

2 Belangrijke termen in datacompressie 17

3 Methodologie 23

3.1 Aanpak van deze bachelorproef 23

3.1.1 Deel 1: situering en literatuurstudie 24

3.1.2 Deel 2: datacompressietool ontwikkelen 24

3.1.3 Deel 3: afbeelding- en videocompressie 25

3.1.4 Deel 4: onderzoek afbeeldingscompressie 26

3.1.5 Deel 5: uitdagingen en conclusie 27

4 Literatuurstudie 29

4.1 Bestandsgrootte en dataopslag 29

4.1.1 Voorvoegsels voor het uitdrukken van bestandsgrootte 29

4.2 Ontstaan datacompressie en primitieve technieken 31

4.2.1 Eerste vorm van datacompressie 31

4.2.2 Ontwikkeling datacompressie binnen IT 31

4.3 Primitieve technieken: een voorbeeld 32

4.3.1 Situering 32

4.3.2 ASCII encoding en decoding 33

4.3.3 RLE: run length encoding en decoding 33

4.3.4 Huffman coding 34

4.4 Lossless vs lossy datacompressie 37

5 Proof of concept datacompressietool 39

5.1 Gebruikte technologie 40

5.2 Run length encoding - lang 40

5.3 Run length encoding - kort 42

5.4 Run length decoding 42

5.5 Huffman encoding 43

5.6 Huffman decoding 45

5.7 Opslaan van de gecomprimeerde bestanden 46

5.8 Resultaten 46

5.9 Beperkingen met deze datacompressietool 47

6 Afbeeldingscompressie 49

6.1 Raster vs vector afbeeldingsformaten 50

6.2 Afbeeldingsformaten 51

6.2.1 RAW 51

6.2.2 PNG 52

6.2.3 JPEG 54

6.2.4 JPEG2000 57

6.2.5 WEBP 59

6.2.6 HEIF/HEIC 60

6.3 De juiste keuze 61

6.3.1 Functievereisten 61

6.3.2 Ondersteuning 62

6.4 Implementatiemogelijkheden voor nieuwe afbeeldingsformaten 62

6.4.1 Webomgeving 63

6.4.2 Manueel 63

6.4.3 Geautomatiseerd 63

6.4.4 On-premise omgeving 64

7 Videocompressie 65

7.1 Intra- vs inter-frame 66

7.2 Video codecs 66

7.2.1 H.264-AVC 67

7.2.2 H.265/HEVC 69

7.2.3 AV1 70

7.3 De juiste keuze 71

7.3.1 Functievereisten 71

7.3.2 Ondersteuning 71

7.3.3 Licenties 72

7.4 Implementatie 72

8 Kwaliteit beoordelen 73

8.1 Objectieve vs subjectieve onderzoeken naar kwaliteit 73

8.2 Software voor het objectief evalueren van afbeeldingen 74

8.2.1 RMSE 75

8.2.2 SSIM 75

8.2.3 PSNR 75

8.2.4 Het probleem met deze formules 75

8.3 Instellingen voor kwaliteitsevaluatie 76

8.3.1 DxOMark 76

8.3.2 Maximum Compression 76

8.3.3 Het probleem met deze instellingen 76

9 Onderzoek 77

9.1 Waarom een subjectief onderzoek? 77

9.2 Use case 78

9.3 Afbeeldingsevaluatietool 78

9.3.1 Opzetten van de afbeeldingsevaluatietool 79

9.3.2 Verloop van de afbeeldingsevaluatietool 83

9.3.3 Verloop van de afbeeldingsevaluatietool: Introductie 83

9.3.4 Verloop van de afbeeldingsevaluatietool: Informatie over deelnemer . 83

9.3.5 Verloop van de afbeeldingsevaluatietool: beoordeling 84

9.3.6 Exporteren van de verzamelde gegevens 85

9.4 Uitvoering 86

9.4.1 Geëvalueerde afbeeldingen 87

9.4.2 Deelnemers 88

9.5 Resultaten 88

9.6 Resultaten lossless afbeeldingsformaten 88

9.7 Resultaten lossy afbeeldingsformaten 88

9.8 Besluit 89

10 Huidige en toekomstige uitdagingen 91

10.1 DNA compressie 91

10.2 AI en compressie 92

11 Conclusie 93

11.0.1 Hoofdonderzoeksvraag 95

11.0.2 Mogelijke uitbreidingen 95

12 Bijlages 97

12.1 Figuren literatuurstudie 97

12.2 Screenshots datacompressietool 98

12.3 Screenshots afbeeldingsevaluatietool 99

12.4 Extra documenten onderzoek 101

A Onderzoeksvoorstel 107

A.1 Introductie 107

A.2 Stand van zaken 108

A.3 Methodologie 109

A.4 Verwachte resultaten 109

A.5 Verwachte conclusies 109

Bibliografie 111

1. Inleiding

Datacompressie, en compressie in het algemeen is niets nieuw. Integendeel, het is één van de oudste concepten binnen IT en tot op heden van fundamenteel belang voor zowat alle IT-toepassingen. Compressie-algoritmen zorgen ervoor dat er veel sneller en goedkoper gewerkt kan worden. Datacompressie en de altijd groeiende waaier aan compressie-algoritmen maken het ook mogelijk om data te verwerken waar voorheen geen beginnen aan was. Denk hierbij bijvoorbeeld aan recente doorbraken binnen DNA compressie die het mogelijk maken steeds meer onderzoeken met betrekking tot het menselijk genoom uit te voeren.

Er zijn reeds tal van uitgebreide en professionele papers geschreven rond zowel datacompressie als afbeeldingscompressie en videocompressie in het bijzonder. Deze onderwerpen zijn immers van fundamenteel belang binnen IT. De focus van deze paper ligt dan ook niet op het herschrijven van papers die reeds bestaan, maar aan het maken van een nuttig document voor iedereen binnen de IT-wereld.

Door de verworven kennis zal je als programmeur, content creator of eender welk andere belanghebber inzicht krijgen hoe datacompressie ontstaan is, hoe het werkt en waarom het zo belangrijk is. Verschillende mogelijkheden voor afbeeldingscompressie en videocompressie zullen besproken worden, samen met hun voordelen en nadelen. Dit zal inzicht geven tot waarom het zo belangrijk is om voor de juiste video codecs en afbeeldingsformaten te kiezen.

Er zal een onderzoek uitgevoerd worden welk afbeeldingsformaat het best is voor een bepaalde use case: portret foto's in de portfolio van een fotografe. De hiervoor ontwikkelde en gratis te gebruiken open source afbeeldingsevaluatietool zal de mogelijkheid bieden dit onderzoek te reproduceren of een gelijkaardig onderzoek met andere afbeeldingsformaten,

voor een andere use case of met andere ondervraagde kenmerken, uit te voeren.

1.1 Probleemstelling

Datacompressie, afbeeldingsformaten en video codecs spelen een belangrijke rol voor vele werknemers binnen de IT-sector. De kennis van deze onderwerpen is bij velen echter ondermaats.

Dit is deels te wijten aan vele opleidingsinstellingen die niet dieper ingaan op datacompressie binnen opleidingen als Toegepaste Informatica, Communicatie Management en Digital Media Manager.

Anderzijds is de beschikbare informatie veelal te complex uitgelegd zodat een doorsnee lezer al snel afhaakt. Ook focussen veel van deze documenten zich op één specifiek onderdeel binnen datacompressie, zoals het gedetailleerd uitleggen van één afbeeldingsformaat, waardoor een globaal beeld moeilijk te scheppen valt.

Het is juist hier waar deze paper beter wilt doen. Dit wordt bereikt door het gebruik van een brede waaier van besproken afbeeldingsformaten en video codecs, een woordenlijst (hoofdstuk 2), en de nodige theoretische en praktische uitleg gevalideerd door vakexpert Tom Paridaens.

Deze bachelorproef schept inzicht over datacompressie, het gebruik en de implementatie en voorziet de nodige basiskennis om eventueel verder onderzoek te voeren.

1.2 Onderzoeksvragen

Deze bachelorproef tracht een antwoord te geven op de volgende vragen:

- Hoe is datacompressie binnen IT ontstaan?
- Wat waren enkele van de eerste compressie-algoritmen?
- Waar zitten de verschillen tussen de afbeeldingsformaten en video codecs?
- Hoe kan datacompressie correct geïmplementeerd worden?
- Wat is het verschil tussen de afbeeldingsformaten: PNG, JPEG, JPEG2000, WebP en HEIF
- Wat is het verschil tussen de video codecs: H.264-AVC, H.264-SVC, H.265/HEVC en AV1?
- Wat is DNA compressie en wat zijn andere uitdagingen binnen datacompressie?

Hierdoor zou de hoofdonderzoeksvraag moeten kunnen beantwoord worden, zijnde:

- Waarom moet er stilgestaan worden bij het gebruiken van compressie-algoritmen, hoe kies je een geschikt compressie-algoritme voor een bepaalde use case en hoe implementeer je dit best.

1.3 Onderzoeksdoelstelling

De doelstelling van deze bachelorproef is het vormen van een nuttig document voor het brede scala belanghebbenden naar de onderzoeksvragen (zie 1.2). Enerzijds is dit document dus een vergelijkende studie tussen verschillende video codecs en afbeeldingsformaten waardoor het ook een verslag vormt met tal van aanbevelingen. Anderzijds zorgt de technische uitleg en basic proof-of-concept datacompressietool samen met de open source afbeeldingsevaluatietool voor de nodige kennis en middelen om verder onderzoek te verrichten.

1.4 Opzet van deze bachelorproef

1.4.1 Deel 1: situering en literatuurstudie

Deze paper zal zich in het eerste deel focussen op het toelichten van de belangrijkste termen binnen datacompressie. In hoofdstuk 2 is een lijst met belangrijke termen te vinden die binnen datacompressie en deze paper vaak voorkomen. Doorheen deze paper zullen tal van referenties naar deze termen gemaakt worden.

Hoofdstuk 3 licht de gebruikte methodologie voor deze paper toe. Hieruit wordt duidelijk dat deze paper zo objectief mogelijk is opgesteld met een focus op duidelijkheid en reproduceerbaarheid.

Hoofdstuk 4 zal de nodige basis voor het begrijpen van deze bachelorproef toelichten. Ook enkele van de eerste compressie-algoritmen zullen besproken worden alsook de theoretische werking ervan in deel 4.3. De besproken compressie-algoritme in dit deel zijn dit run length encoding en Huffman coding, deze worden ook gebruikt voor het maken van de datacompressietool uit hoofdstuk 5.

1.4.2 Deel 2: datacompressietool ontwikkelen

In het tweede deel zal een basis datacompressietool programmatisch geïmplementeerd worden om de theorie uit het eerste deel in praktijk te brengen.

Hoofdstuk 5 is hierdoor gericht aan zowel technische lezers als programmeurs. Er is echter telkens voldoende randinformatie gegeven zodat ook de minder technische lezers een blik achter de schermen kunnen verkrijgen.

Deze datacompressietool en de achterliggende compressie-algoritmen zijn geschreven in PHP. Er is een grafische interface voorzien die gebruik maakt van HTML, CSS, JavaScript en Bootstrap. De geïmplementeerde compressie-algoritmen zijn twee varianten van run length encoding en een uitwerking van Huffman coding.

1.4.3 Deel 3: afbeelding- en videocompressie

In het derde deel worden twee subdomeinen van datacompressie verder toegelicht: afbeeldingscompressie en videocompressie.

In hoofdstuk 6 zal er dieper ingegaan worden op afbeeldingscompressie en volgende afbeeldingsformaten: PNG, JPEG, JPEG2000, WebP en HEIF. De ondersteuning van de verschillende afbeeldingsformaten zal toegelicht worden. De mogelijkheden voor implementatie van nog niet overal ondersteunde afbeeldingsformaten zal in deel 6.4 besproken worden.

In hoofdstuk 7 zal er verder ingegaan worden op de gekende videocompressie standaarden: H.264-AVC en H.264-SVC. Ook de opvolger H.265/HEVC en het open source alternatief AV1 zullen besproken worden. De ondersteuning van de verschillende video codecs zal toegelicht worden alsook welke video codecs enkele gekende bedrijven gebruiken.

1.4.4 Deel 4: onderzoek afbeeldingscompressie

In het vierde deel wordt besproken hoe compressiemethoden voor video's en afbeeldingen geëvalueerd worden. Hoofdstuk 8 zal enkele veel gebruikte tools en methoden voor objectieve en subjectieve vergelijkingen toelichten. Ook instellingen die zich bezig houden met het evalueren van afbeeldingskwaliteit en videokwaliteit zullen besproken worden en hoe de resultaten soms met een korrel zout genomen moeten worden.

In hoofdstuk 9 wordt een subjectieve onderzoek voor het evalueren van afbeeldingskwaliteit besproken. Dit onderzoek focust zich op portretfoto's die weergegeven zullen worden op een online portfolio van een fotografe. Hierbij zullen enkele van de besproken afbeeldingsformaten uit hoofdstuk 6 tegen elkaar concurreren. De gebruikte afbeeldingsevaluatietool is voor deze paper opgesteld en is gratis open source toegankelijk gesteld, wat het eenvoudig mogelijk maakt om een gelijkaardig onderzoek uit te voeren. Dit moet de lezer in staat stellen om samen met de verworven kennis van het voorgaande deel een gegronde keuze te kunnen maken tussen de verschillende compressie-algoritmen voor een bepaalde use case.

1.4.5 Deel 5: uitdagingen en conclusie

In het vijfde deel zullen de huidige uitdagingen van datacompressie kort toegelicht worden. Zo zal hoofdstuk 10 een beeld geven van de taken die mensen als Tom Paridaens, co-promoter voor deze paper, krijgen.

In hoofdstuk 11 worden de onderzoeksvragen herhaalt en nagegaan op welke manier deze paper een antwoord heeft geboden op die onderzoeksvragen. In deel 11.0.2 wordt kritisch teruggeblikt op deze bachelorproef en vermeld waar uitbreidingen voor de bachelorproef mogelijk zijn. Dit moet de lezer stimuleren om verder opzoekingswerk omtrent de interessante wereld van datacompressie te verrichten.

2. Belangrijke termen in datacompressie

Om deze paper vlot te kunnen lezen, zijn er enkele termen die de lezer moet kennen. Termen die vaak voorkomen in datacompressie en doorheen deze paper worden hier opgesomd. Er zal een referentie voorzien zijn naar deze lijst bij het gebruik van een term uit de lijst.

Adobe Illustrator Computerssoftware voor het bewerken van vector afbeeldingen.

Adobe Photoshop Computersoftware voor het aanmaken en bewerken van rasterafbeeldingen.

afbeeldingscompressie Afbeeldingscompressie bestaat eruit een afbeelding in zo weinig mogelijk aantal bits op te slaan terwijl een aanvaardbare kwaliteit behouden blijft. Dit kan zowel via lossless als lossy algoritmes. Afbeeldingscompressie wordt uitgebreid besproken in hoofdstuk 6.

afbeeldingsevaluatietool Een applicatie gemaakt voor het voeren van een objectief of subjectief onderzoek naar afbeeldingskwaliteit. De mogelijke soorten tools worden verder besproken in hoofdstuk 8. Een subjectieve afbeeldingsevaluatietool werd gebouwd voor deze paper en wordt verder besproken in hoofdstuk 9.

afbeeldingsformaat Een afbeeldingsformaat bevat alle gegevens voor het digitaal opslaan van een afbeelding. De vier bekendste categorieën van afbeeldingsformaten zijn: raster, vector, compound en stereo formaten.

array Een term binnen programmeren die slaat op een verzameling van variabelen.

artefact Een kunstmatig verschijnsel. Binnen datacompressie zijn artefacten zichtbare of hoorbare fouten.

artificiële intelligentie Wanneer een apparaat kan reageren op binnenkomende data en op basis daarvan een eigen beslissing kan maken zonder dat dit expliciet geprogram-

meerd is, spreekt men van artificiële intelligentie.

ASCII Een tekenset die voorgesteld wordt door acht bits.

AV1 AV1 is een videocodec ontwikkelt als open standaard. AV1 is een afkorting voor AOMedia Video. AV1 is vooral interessant omdat het royalty free is en dus geen licentiekosten heeft. AV1 wordt uitgebreid besproken in deel 7.2.3.

AVI Een container voor onder andere videocodecs. Videocodecs worden verder toegelicht in hoofdstuk 7.

bandbreedte Een term waarmee de maximale beschikbare hoeveelheid data dat over een netwerkverbinding verstuurd kan worden bedoeld wordt.

bestandsextensie Een toevoeging op het einde van een bestandsnaam om weer te geven wat voor soort bestand het is.

Binaire voorstelling De binaire voorstelling voor bestandsgroottes maakt gebruik van 1024 als basis, wat overeen komt met 2^{10}. Dit wordt verder besproken in deel 4.1.1.2.

bit Zoals de naam bit, kort voor binary digit, suggereert kan een bit beschouwd worden als een binair signaal. Een bit wordt beschouwd als de kleinste eenheid voor dataopslag. Meer informatie rond bestandsgrootte en dataopslag wordt besproken in hoofdstuk 4, deel 4.1.

bitplane Een term voor een groepering van bits.

Bootstrap Een toolkit voor het eenvoudig programmeren met HTML, CSS, JS en jQuery.

byte Een term voor acht bits.

cluster Het kleinste deel van een oplsagmedium waar data in voorzien kan worden. Dit is verder besproken in deel 4.1.1.3.

clustergrootte De grootte van een cluster. Dit is verder besproken in deel 4.1.1.3.

CMS Een content management system is een systeem dat het voor een eindgebruiker eenvoudig moet maken om de inhoud binnen een IT-applicatie eenvoudig te beheren.

CMYK Cyan, Magenta, Yellow, Key is een kleurcodering systeem voor het voorstellen van kleuren aan de hand van deze drie basiskleuren.

codec De coder-decoder. Binnen datacompressie betekent codec de gebruikte techniek om een bestand te comprimeren. De codec is dus de technologie verantwoordelijk voor het encoden en decoden van een bestand volgens een bapaald compressiealgoritme. Bv: H.265/HEVC.

compressie-algoritme Een set instructies die het mogelijk maken om bepaalde informatie met minder resources voor te stellen. Binnen deze bachelorproef wilt dit code voorstellen die een bestand zijn bestandsgrootte kan verkleinen. Dit kan zowel lossless als lossy.

compressieratio De verhouding waarmee een bestand gecomprimeerd is ten opzichte van het originele bestand.

container Binnen datacompressie kan een container vrijwel letterlijk vertaald worden. Het is een verpakking voor alle data die men opslaat en metadata. Onder andere de codec plaatst data in deze container. Wanneer er gesproken wordt over bestandsextensies wordt vaak de container bedoeld. Bv: MP4.

CRC Cyclic redundancy check is een foutdetectie code om te kunnen nagaan of een bestand goed is toegekomen.

CSS Cascading Style Sheets. Een programmeertaal die verantwoordelijk is voor het opmaken van een webpagina.

datacompressie Datacompressie bestaat uit het digitaal opslaan van een bestand met zo weinig mogelijk bits. Enkele belangrijke factoren voor het bepalen van de juiste datacompressie zijn gewenste kwaliteit, bestandsgrootte en snelheid.

datacompressietool Een kleine applicatie voor het comprimeren van data. In hoofdstuk 5 wordt een proof of concept compressietool geïmplementeerd.

DCT Discrete cosine transform is een wiskundige formule die binnen datacompressie voornamelijk gebruikt wordt om een pixel te kunnen voorstellen als een makkelijk comprimeerbaar getal.

decoder Een softwareapplicatie die een gecomprimeerd bestand terug omzet naar zijn originele vorm, al dan niet met kwaliteitsverlies.

decoding Het process dat een decoder uitvoert.

deflate Een datacompressie techniek die voornamelijk binnen videcompressie gebruikt wordt.

DNA compressie Het menselijke DNA kan digitaal voorgesteld worden door een lange lijst van 5 verschillende karakters, gekend als basen. Deze digitale voorstelling bestaat uit meer dan 3 miljard van deze basen (UGent, 2011). DNA compressie bestaat eruit deze reeks van basen zo efficiënt mogelijk op te slaan zodat performante bewerkingen mogelijk zijn met een zo klein mogelijke bestandsgrootte.

DNG Digital Negative Specification is een RAW formaat dat Adobe heeft uitgevonden met de bedoelding dat dit wereldwijd geadopteerd zou worden als standaard voor RAW afbeeldingsformaten.

Drift Een JavaScript library die het eenvoudig maakt om in te zoomen op afbeeldingen zonder manipulaties.

DxOMark Een erkende instelling die de kwaliteit van, voornamelijk smartphone, camera's beoordeeld. DxOMark wordt verder toegelicht in deel 8.3.1.

encoder Een softwareapplicatie dat een bestand omzet naar een gecomprimeerde vorm, al dan niet met kwaliteitsverlies.

encoding Het proces dat een encoder uitvoert.

frame Een frame is één stilstaande afbeelding binnen een video.

frame rate De snelheid waarmee frames elkaar opvolgen, vaak uitgedruk in frames per seconden (FPS).

GIF Graphics interchange format is een afbeeldingsformaat dat op heden voornamelijk gebruikt wordt om bewegende delen voor te stellen.

GitHub Een online platform voor versiebeheer van code.

H.264-AVC H.264-AVC is één van de bekendstes videocodecs die grootschalig gebruikt wordt. AVC is een afkorting van Advanced Video Coding. H.264-AVC wordt uitgebreid besproken in deel 7.2.1.

H.264-SVC H.264-SVC is een videocodec ontwikkelt als extensie op H264-AVC. SVC is een afkorting voor Scalable Video Coding. De nadruk bij deze extensie ligt zoals de naam suggereert op schaalbaarheid. H.264-SVC wordt uitgebreid besproken in deel 7.2.1.3.

H.265/HEVC H.265/HEVC is een videocodec ontwikkelt als opvolger van H.264/AVC. H.265/HEVC wordt uitgebreid besproken in deel 7.2.2.

HDR High Dynamic Range beelden zijn beelden die een hoog dynamisch bereik hebben.

HEIC Een afbeeldingsformaat dat uitgebreid besproken wordt in 6.2.6.
HEIF Een codec dat uitgebreid besproken wordt in deel 6.2.6.
HLG Hybrid Log-Gamma is een standaard voor het voorzien van HDR content.
hosting Het delen van een bepaalde webpagina met anderen.
HTML Een programmeertaal voor het opbouwen van webpagina's.
Huffman coding Een compressie-algoritme gebasseerd op prefix code. Huffman coding wordt verder toegelicht aan de hand van een voorbeeld in deel 4.3.4. Een implementatie wordt voorzien in hoofdstuk 5.

IEEE Institute of Electrical and Electronics Engineers. Het is een non-profit instituut dat zich inzet voor technologische vooruitgang.
inter-frame Inter-frame compressie is een techniek van afbeeldingen comprimeren binnen videocompressie die uitgebreid besproken wordt in deel 7.1.
interlace Een techniek waarbij twee frames tegelijk getoond worden waardoor de kijker een vals beeld krijgt dat de frame rate sneller is dan hij werkelijk is.
intra-frame Intra-frame compressie is een techniek van afbeeldingen comprimeren binnen videocompressie die uitgebreid besproken wordt in deel 7.1.
ISO De International Organization for Standardization is een organisatie die internationale standaarden oplegt voor bijvoorbeeld compressie-algoritmen.

JavaScript Een programmeertaal waarmee onder andere de inhoud van een webpagina gemanipuleerd kan worden.
JPEG Ook gekend als JPG. JPEG is een afkorting voor Joint Photographic Experts Group. JPEG is een bestandsformaat voor het opslaan van digitale afbeeldingen via lossy compressie. Afbeeldingscompressie en JPEG worden uitgebreid besproken in hoofdstuk 6.
JPEG/Exif Een bestandsformaat verder toegelicht in deel 6.2.3.
JPEG/JFIF Een bestandsformaat verder toegelicht in deel 6.2.3.
JPEG2000 Ook gekend als JPEG2K. JPEG2000 is een bestandsformaat voor het opslaan van digitale afbeeldingen gemaakt als opvolger van JPEG. Net zoals JPEG maakt het gebruik van lossy compressie. Afbeeldingscompressie en JPEG2000 worden uitgebreid besproken in hoofdstuk 6.
JPF JPF is de bestandsexentsie van JPEG2000. JPEG2000 wordt uigebreid besproken in deel 6.2.4.
JQuery Een uitbreiding op JavaScript.
json Een techniek voor het opslaan van variabelen in een string.

lbhuffman Een bestandsextensie die gebruikt wordt binnen de datacompressietool. Wordt verder besproken in deel 5.7.
lbrle Een bestandsextensie die gebruikt wordt binnen de datacompressietool. Wordt verder besproken in deel 5.7.
lbrlea Een bestandsextensie die gebruikt wordt binnen de datacompressietool. Wordt verder besproken in deel 5.7.
leestijd De tijd die verstrijkt tussen het aanvragen van een bestand op een opslagmedium tot het effectief verkrijgen van dat bestand.
library Een verzameling codes dat door een programma gebruikt kunnen worden.
lookup table Een tabel waar waardes in opgezocht kunnen worden. Bij prefix code is dit de tabel waar de lange waarde staat die de korte prefix code voorstelt.

lossless Binnen datacompressie slaat lossless compressie op het comprimeren van een bestand zonder kwaliteitsverlies. In het geval van video's en afbeeldingen wilt dit zeggen dat een bestand gecomprimeerd met een lossless algoritme identiek is aan het origineel.

lossy Binnen datacompressie slaat lossy compressie op het comprimeren van een bestand met kwaliteitsverlies voor het besparen van data. In het geval van video's en afbeeldingen wilt dit zeggen dat een bestand gecomprimeerd met een lossy algoritme een significante hoeveelheid aan data en kwaliteit kan verliezen in vergelijking met het originele bestand.

Maximum Compression Een erkende instelling die de compressieratio van lossles compressiealgoritmes beoordeelt. Maximum Compression wordt verder toegelicht in deel 8.3.2.

metadata Data over data.

minifyen Een proces waarbij code zodanig herschreven wordt dat dit de minste hoveelheid ruimte in beslag neemt.

MKV Een container voor onder andere videocodecs. Videocodecs worden verder toegelicht in hoofdstuk 7.

MP4 Een container voor onder andere videocodecs. Videocodecs worden verder toegelicht in hoofdstuk 7.

MPEG-4 Een videocompressie algoritme dat de voorganger was van H.264/AVC, toegelicht in deel 7.2.1.

NEF Het RAW afbeeldingsformaat van Nikon.

on premise Software is on-premise wanneer deze lokaal geïnstalleerd en gedraaid wordt op het toestel van de gebruiker.

open source Als een programmeer project open source is, wilt dit zeggen dat de broncode raadpleegbaar is. Dit wil echter niet gegarandeerd zeggen dat het software programma gratis is in gebruik of de code zomaar aangepast mag worden. Dit hangt af van de licentie.

Pandas Een uitbreiding voor Python.

PHP Een programmeertaal die voornamelijk de logica bij websites voorziet.

pixel Één punt binnen een rasterafbeelding.

pixel prediction Een datacompressie techniek die de pixel van een afbeelding gaat voorspellen aan de hand van een andere pixel.

plug-in Een downloadbare collectie code die bepaalde functionaliteit voorziet.

PNG PNG is een afkorting voor Portable Network Graphics. PNG is een bestandsformaat voor het opslaan van digitale afbeeldingen. PNG maakt gebruik van lossless compressie. Afbeeldingscompressie en PNG worden uitgebreid besproken in hoofdstuk 6.

prefix code Een korte waarde die een langere waarde voorstelt binnen prefix coding. Dit is verder beschreven in deel 4.2.2.

prefix coding Een vorm van datacompressie waarbij een waarde wordt toegekend aan iets dat verwijst naar een andere, langere waarde. Dit is verder beschreven in deel 4.2.2.

Python Een programmeertaal die voornamelijk gebruikt wordt voor het verwerken van data in de vorm van grote datascheets.

raster Een categorie voor afbeeldingsformaten die verder toegelicht wordt in deel 6.1.

RAW Een benaming voor afbeeldingsformaten die in geen enkele vorm gecomprimeerd worden. RAW wordt uitgebreid besproken in deel 6.2.1.

recursieve functie een recursieve functie is een functie die zichelf (al dan niet meermaals) oproept.

regex Regular Expressions zijn uitdrukkingen waarmee op zoek gegaan kan worden naar een tekst met een bepaalde structuur.

render Het genereren van een digitale afbeelding of video met behulp van computersoftware.

RGB Red Green Blue is een kleurcodering systeem voor het voorstellen van kleuren aan de hand van deze drie basiskleuren.

RLE Kort voor run length encoding.

rmse Root mean square error, besproken in deel 8.2.1.

run length encoding Een compressietechniek die verder besproken wordt in deel 4.3.3 en geïmplementeerd wordt in deel 5.

SI voorstelling De SI voorstelling voor bestandsgroottes maakt gebruik van 1000 als basis wat overeen komt met 10^3. Dit wordt verder besproken in deel 4.1.1.1.

SQL Een standaard programmertaal voor communicatie met een relationele databank.

SSIM Structural similarity index, besproken in deel 8.2.2.

string Een term binnen programmeren die slaat op een tekstfragment zonder opmaak.

SVG Scalable Vector Graphics zijn een schaalbare vorm van vector afbeeldingen. Vector wordt verder toegelicht in deel 6.1.

use case De beschrijving van het gewenst gedrag van een systeem of applicatie.

VDP Visual difference predictor is een term voor functies die visuele verschillen tussen afbeeldingen probeert uit te drukken.

vector Een categorie voor afbeeldingsformaten die verder toegelicht wordt in deel 6.1.

videocompressie Videocompressie bestaat eruit een videobestand in zo weinig mogelijk aantal bits op te slaan terwijl een accepteerbare kwaliteit behouden blijft. Dit kan zowel via lossless als lossy algoritmes. Videcompressie wordt uitgebreid besproken in hoofdstuk 7.

VP8 Een video codec gemaakt door Google. WebP, besproken in deel 6.2.5, is afkomstig van VP8. AV1, besproken in 7.2.3, is een verre opvolger van VP8.

W3C World Wide Web Consortium is een organisatie die standaarden voor het web voorziet.

wavelet Een golfvormige soort data.

WebP Een afbeeldingsformaat dat uitgebreid besproken wordt in deel 6.2.5.

WordPress Een CMS voor web development.

WTCQ Wavelet/Trellis Coded Quantization is een compressie algoritme dat de start vormde voor JPEG2000. JPEG2000 wordt uigebreid besproken in deel 6.2.4.

XAMPP Een software pakket waarmee eenvoudig een hosting omgeving kan opgezet worden.

YouTube Een online website waar gebruikers gratis video's kunnen uploaden. YouTube is een dochterbedrijf van Google en is één van de grootste streaming diensten.

3. Methodologie

Bij het schrijven van een wetenschappelijk document is een gegronde methodologie vereist. Dit hoofdstuk licht de gekozen methodologie voor deze bachelorproef toe en bespreekt die keuze per deel van dit document. Ook de gebruikte LaTeX packages worden kort toegelicht.

3.1 Aanpak van deze bachelorproef

Deze bachelorproef streeft naar een correcte en verantwoordelijke manier van werken om de betrouwbaarheid van dit document te bewaren. Hiervoor is dit document meermaals gevalideerd door en veranderd naar input van vakexpert en co-promotor Tom Paridaens.

Om dit te garanderen is er ook uitsluitend gebruik gemaakt van primaire of secundaire bronnen en geen tertiaire bronnen. De kennis verworven uit primaire bronnen is steeds gevalideerd met secundaire bronnen. De volledige bronnenlijst is ter beschikking gesteld op het einde van dit document. Wanneer data uit een bron overgenomen is naar dit document is steeds een verwijzing naar de oorspronkelijke bron voorzien.

De kennis uit primaire bronnen komt voornamelijk uit de vijf jaar informatica gerelateerde studies en twee jaar fotografie gerelateerde studies die door de auteur van deze bachelorproef gevolgd zijn.

Dit document is geschreven in LaTeX en is voorzien van een BibTeX bibliografische databank. Er is onder andere gebruik gemaakt van volgende packages:

- Glossary voor het voorzien van een woordenlijst. Woorden die voorkomen in de woordenlijst uit hoofdstuk 2 bevatten een verwijzing naar dit overzicht wanneer er

op geklikt wordt.
- Listings en colors voor het voorzien van code met de gepaste highlighting.
- Xcolor voor het voorzien van achtergrondkleuren in de cellen van een tabel.
- Placeins voor meer controle over de plaatsing van tabellen en andere figuren door LaTeX.

In de volgende secties zal voor de vijf verschillende delen van dit document kort toegelicht worden wat de gekozen methodologie is en waarom.

3.1.1 Deel 1: situering en literatuurstudie

In hoofdstuk 2 is er voor gekozen een lijst van belangrijke termen te voorzien. Deze lijst helpt de lezer de bachelorproef vlot te lezen. Er is gebruik gemaakt van de Glossary package omdat deze de mogelijkheid voor referenties en een alfabetisch gerangschikte woordenlijst voorziet.

De literatuurstudie (hoofdstuk 4) is kort gehouden, maar volstaat samen met de woordenlijst uit hoofdstuk 2 voor het begrijpen van de overige besproken zaken uit dit document. Deze literatuurstudie licht ook enkele primitieve compressie-algoritmen toe en maakt daarvoor gebruik van de originele documenten omtrent de uitgave van deze compressie-algoritmen.

In dit deel worden de volgende onderzoekvragen (deels) beantwoord:

- Hoe is datacompressie binnen IT ontstaan?
- Wat waren enkele van de eerste compressie-algoritmen?
- Waar zitten de verschillen tussen de afbeeldingsformaten en video codecs?
- Hoe kan datacompressie correct geïmplementeerd worden?

3.1.2 Deel 2: datacompressietool ontwikkelen

Aan de hand van de verworven kennis uit de hoofdstuk 4 is een proof of concept datacompressietool geschreven in PHP met een grafische interface in HTML, CSS en Bootstrap. Er is gekozen om de datacompressietool in deze talen te schrijven aangezien deze eenvoudig online te hosten zijn of lokaal te runnen. De tool is dan ook online ter beschikking gesteld op de website van Lennert Bontinck[1]. Dit maakt het voor de lezer eenvoudig om de datacompressietool zelf te testen.

De code van de datacompressietool is publiek ter beschikking gesteld op de GitHub repository van deze bachelorproef[2]. Deze is vrijgegeven onder de GNU GPLv3 licentie en mag dus gratis aangepast en gebruikt worden voor alle doeleinden. Dit maakt het eenvoudig voor de lezer om de broncode te raadplegen en, aan de hand van de uitleg in hoofdstuk 5, aan te passen.

[1] POC compressietool - BAP – http://bit.ly/2Qx0gX9.
[2] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

Er is bewust gekozen om de code achter deze tool simpel te houden en te werken met Nederlandse variabelen zodat de code, mits de voorziene uitleg in hoofdstuk 5, ook voor minder technische lezers verstaanbaar is. Hiervoor zijn ook tal van links naar de geziene theorie uit deel 4.3 voorzien.

De verworven bestanden na compressie worden met het origineel vergeleken op basis van het aantal karakters nodig om de tekst op te slaan voor de run length encoding gebaseerde compressie-algoritmen en het aantal bits nodig om de tekst op te slaan voor het Huffman coding gebaseerde compressie-algoritme. Dit geeft een duidelijker beeld van de prestatie van het compressie-algoritme dan zuiver de bestandsgrootte.

Aangezien het om een proof of concept datacompressietool gaat zijn er enkele beperkingen, deze worden dan ook toegelicht in deel 5.9. Dit doet de gebruiker stilstaan over mogelijke (ongewenste) beperkingen die kunnen voorkomen bij het implementeren van een compressie-algoritme en nadenken over mogelijke oplossingen.

In dit deel worden de volgende onderzoekvragen (deels) beantwoord:

- Wat waren enkele van de eerste compressie-algoritmen?
- Hoe kan datacompressie correct geïmplementeerd worden?

3.1.3 Deel 3: afbeelding- en videocompressie

In dit deel is er voor gekozen om de volgende afbeeldingsformaten toe te lichten: PNG | JPEG | JPEG2000 | WebP | HEIF. PNG en JPEG zijn namelijk de bekendste en op het web meest gebruikte afbeeldingsformaten. JPEG2000, WebP en HEIF zijn dan weer enkele van de bekendste nieuwe generatie afbeeldingsformaten. Dit zorgt er voor dat de besproken afbeeldingsformaten diegene zijn die het meeste potentieel hebben om een goede keuze te zijn voor het doelpubliek van deze bachelorproef.

Voor elk afbeeldingsformaat is kort het ontstaan toegelicht en aan de hand van de bijhorende ISO de werking (voor PNG en JPEG diepgaander dan de anderen) uitgelegd. Buiten de ISO is ook veel informatie over de werking van enkele van de besproken afbeeldingsformaten uit het uitgebreide boek rond datacompressie: *Data Compression: The Complete Reference* (Salomon, 2006) gehaald. Deze ISO is door de maker mee opgesteld en garandeert dus dat de werking van het afbeeldingsformaat juist beschreven is. De belangrijkste voordelen en nadelen zijn ook steeds toegelicht zodat de lezer zelf een beeld kan scheppen welk afbeeldingsformaat geschikt is voor zijn use case. De overzichtstabellen in deel 6.3.1 en 6.3.2 omtrent functievereisten en ondersteuning helpen de lezer hier ook bij.

Ook de betekenis van RAW afbeeldingen wordt hier toegelicht en waarom ze belangrijk zijn om op een objectieve manier een onderzoek te voeren naar de prestatie van een afbeeldingsformaat.

Om te vermijden dat lezers schrik hebben om nieuwe afbeeldingsformaten te gebruiken, is in deel 6.4 besproken hoe je deze nieuwe afbeeldingsformaten eenvoudig kan implementeren. Ook oplossingen voor situaties waar de gebruiker geen ondersteuning heeft

voor deze nieuwe generatie afbeeldingsformaten, wordt besproken. Er zijn ook enkele geautomatiseerde oplossingen voor het voorzien van deze nieuwe afbeeldingsformaten besproken in deel 6.4.3. Dit helpt de lezer inzicht te krijgen in hoe de implementatie zal verlopen voor zijn use case.

In hoofdstuk 7 zijn H.264-AVC, H.264-SVC, H.265/HEVC en AV1 besproken. Dit omdat H.264-AVC, H.264-SVC en H.265/HEVC de bekendste video codecs zijn en AV1 een gekend open source alternatief is voor vrij gebruik. Ook hier is er gekozen om kort het ontstaan toe te lichten alsook de voordelen en de nadelen. Ook de werking wordt hier kort aangehaald met ondersteuning van de bijhorende ISO. De belangrijkste voordelen en nadelen worden voor elke video codec toegelicht en helpen de lezer samen met de punten uit deel 7.3 een juiste keuze te maken.

In dit deel worden de volgende onderzoekvragen (deels) beantwoord:

- Waar zitten de verschillen tussen de afbeeldingsformaten en video codecs?
- Hoe kan datacompressie correct geïmplementeerd worden?
- Wat is het verschil tussen de afbeeldingsformaten: PNG, JPEG, JPEG2000, WebP en HEIF
- Wat is het verschil tussen de video codecs: H.264-AVC, H.264-SVC, H.265/HEVC en AV1?
- Hoe kan datacompressie correct geïmplementeerd worden?

3.1.4 Deel 4: onderzoek afbeeldingscompressie

In hoofdstuk 9 wordt een subjectief onderzoek naar de afbeeldingskwaliteit van PNG, JPEG, JPEG2000 en WebP gevoerd. Er is gekozen voor een subjectief onderzoek omdat dit voor de use case de aangeraden manier van werken is. Voor dit subjectieve onderzoek is een afbeeldingsevaluatietool geschreven in PHP, SQL, HTML, CSS, JavaScript en Drift. Net zoals de datacompressietool is deze publiek ter beschikking gesteld op de GitHub repository van deze bachelorproef[3]. Deze is vrijgegeven onder de GNU GPLv3 licentie en mag dus gratis aangepast en gebruikt worden voor alle doeleinden. Dit maakt het eenvoudig voor de lezer om zelf, aan de hand van de uitleg in deel 9.3, een subjectief onderzoek te voeren.

De resultaten van het onderzoek worden met Python en Pandas in overzichtelijke grafieken en tabellen gezet. Deze scripts zijn samen met de resultaten beschikbaar op de GitHub van deze bachelorproef. Er is bewust gekomen om de lezer zelf de kans te geven een besluit te trekken aan de hand van de geziene theorie en de resultaten.

In dit deel worden de volgende onderzoekvragen (deels) beantwoord:

- Waar zitten de verschillen tussen de afbeeldingsformaten en video codecs?
- Wat is het verschil tussen de afbeeldingsformaten: PNG, JPEG, JPEG2000, WebP en HEIF

[3] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

3.1.5 Deel 5: uitdagingen en conclusie

Tot slot is er gekozen om nog enkele van de uitdagingen binnen datacompressie toe te lichten aan de lezer. Dit is deels als ode aan vakexpert en co-promotor Tom Paridaens, maar ook om de lezer warm te maken zich verder te verdiepen in deze uitdagingen.

In de conclusie (hoofdstuk 11) wordt op een objectieve manier teruggeblikt naar de antwoorden die deze bachelorproef biedt, alsook naar interessante vragen die ze opwekt. Het grote doel is om de lezer te overtuigen zich verder te verdiepen in datacompressie en meer stil te staan bij de keuze van een compressie-algoritme.

In dit deel wordt het antwoord op de volgende onderzoekvragen gegeven: 'Wat is DNA compressie en wat zijn andere uitdagingen binnen datacompressie?'. In de conclusie (hoofdstuk 11) wordt nog eens teruggeblikt op alle delen waardoor ook het antwoord op de hoofdonderzoeksvraag in dit deel gegeven wordt.

4. Literatuurstudie

Deze literatuurstudie zal samen met de lijst van termen uit hoofdstuk 2 de nodige achtergrondinformatie bieden om de volgende delen van de paper te begrijpen. Er zal verwezen worden naar meerdere papers van andere instellingen zodat u zich verder kan inlezen waar gewenst.

4.1 Bestandsgrootte en dataopslag

Zoals in de definitie van datacompressie besproken is, bestaat datacompressie uit het digitaal opslaan van een bestand met zo weinig mogelijk bits.

Bit staat kort voor binary digit. Een bit wordt beschouwd als de kleinste data eenheid voor dataopslag. Een bit kan twee waarden aannemen, deze worden voorgesteld door 1 of 0 (binair talstelsel), maar kunnen ook geïnterpreteerd worden als aan of uit, ja of nee...

4.1.1 Voorvoegsels voor het uitdrukken van bestandsgrootte

Bestandsgroottes worden meestal uitgedrukt in bytes (8 bits), al dan niet met voorvoegsel dat een veelvoud voorstelt. Deze voorvoegsels en het door elkaar gebruik van de SI voorstelling en Binaire voorstelling kan voor enige verwarring zorgen. Denk hierbij aan het fenomeen dat harde schijven die geadverteerd zijn als 1TB (SI voorstelling) overeen komen met 931GiB (Binaire voorstelling) op de meeste besturingssystemen. Doorheen deze paper zal de SI voorstelling gebruikt worden.

4.1.1.1 SI voorstelling

De SI voorstelling gebruikt als basis 1000 wat overeen komt met 10^3. SI staat voor International System of Units en beschrijft de standaardeenheden voor het meten van bepaalde data. Het wordt beschouwd als een moderne vorm op het metrisch stelsel. SI beschrijft onder IEC 60027 het gebruik van bepaalde voorvoegsels voor het uitdrukken van machten op 10. (IEC, 1999) Een conversietabel is hieronder raadpleegbaar.

Voorvoegsel	symbool	waarde
kilo	Ki	$10^3 = 1000^1$ = 1 000
mega	Mi	$10^6 = 1000^2$ = 1 000 000
giga	Gi	$10^9 = 1000^3$ = 1 000 000 000

4.1.1.2 Binaire voorstelling

De Binaire voorstelling gebruikt als basis 1024, wat overeen komt met 2^{10}. Deze voorstelling is een standaardisatie opgelegd door IEEE 1541-2002 („IEEE Standard for Prefixes for Binary Multiples”, 2009). Een conversietabel is hieronder raadpleegbaar.

Voorvoegsel	symbool	waarde
kibi	Ki	$2^{10} = 1024^1$ = 1 024
mebi	Mi	$2^{20} = 1024^2$ = 1048 576
gibi	Gi	$2^{30} = 1024^3$ = 1 073 741 824

4.1.1.3 Clustergrootte

Een andere belangrijke term bij dataopslag is de clustergrootte. Data moet namelijk bijgehouden worden in één of meerdere clusters op een opslagmedium zodat naar deze clusters kan verwezen worden voor het lezen van de data.

Aangezien alle data steeds minstens in één cluster staat en twee verschillende databestanden nooit een cluster kunnen delen, kan dit voor opslagruimteverlies zorgen.

Neem bijvoorbeeld een clustergrootte van 4096 bytes, een vaak voorkomende clustergrootte. Als in deze situatie een bestand van 2000 bytes groot zou opgeslagen worden, zijn de overige 2096 bytes aan opslagcapaciteit op die schijf verloren. Een bestand van 4097 bytes zou twee clusters in beslag nemen waardoor 4095 bytes verloren gaan.

Dit speelt vooral een rol wanneer datacompressie gebruikt wordt voor het besparen van opslagruimte op een opslagmedium. Een gecomprimeerd bestand met een kleinere bestandsgrootte dat dezelfde hoeveelheid clustergroottes nodig heeft op het medium zal dus niet voor plaatsbesparing zorgen op dat opslagmedium.

Theoretisch gezien zal er wel een verbetering te zien zijn in leestijden en de gebruikte bandbreedte bij een bestandsoverdracht omdat de effectieve bestandsgrootte kleiner is.

Er zijn tal van reden waarom een andere clustergrootte aangeraden is, een recente discussie is terug te vinden in een blogpost van Microsoft[1]

4.2 Ontstaan datacompressie en primitieve technieken

4.2.1 Eerste vorm van datacompressie

Vele onderzoekers zijn het erover eens dat datacompressie dateert van voor de uitvinding van de computer. Zo kan morsecode gezien worden als een vorm van datacompressie. Morsecode is uitgevonden voor het computertijdperk, in 1832, door Samuel F.B. Morse. Het kan gezien worden als een vorm van datacompressie doordat veel voorkomende letters een kortere audiotoon kregen dan minder gebruikte letters. (Pop, 2015)

4.2.2 Ontwikkeling datacompressie binnen IT

Bij de prille opkomst van mainframes eind de jaren 40 en begin de jaren 50 zijn twee belangrijke doorbraken binnen datacompressie gemaakt. Beiden maken gebruik van prefix code. De originele uitvinder van dit soort compressie was Shannon Claude die Shannon coding uitvond, een proof of concept voor zijn artikel „A Mathematical Theory of Communication" (Shannon, 1948). In diezelfde periode werd ook Shannon-Fano coding voorgesteld, een project samen met Robert Fano dat verschillende use cases had. Geen van beide technieken waren echter optimaal aangezien de compressie-algoritmen niet gegarandeerd de kortst mogelijke prefix codes gaven.

Huffman coding, voorgesteld in „A Method for the Construction of Minimum-Redundancy Codes" (Huffman, 1952), was een optimale variant op deze techniek. Dit is een compressie-algoritme door David Huffman als examen vervangende opdracht. Zijn lector had vermeld dat er eerste vormen van prefix code compressie-algoritme bestonden maar geen enkel optimaal was. Een student die een optimaal algoritme wist te vinden kreeg een vrijstelling voor het desbetreffende examen. Bijna 70 jaar na publicatie legt dit nog steeds de basis voor vele lossless compressie-algoritmen. Deze soort compressie-algoritmen worden frequency-based compressie-algoritmen genoemd. Het exacte verschil tussen Huffman coding en Shannon-Fano coding en meer informatie over deze compressie-algoritmen zijn beschreven in „Data Compression" (Lelewer en Hirschberg, 1987). In deel 4.3.4 wordt een praktisch voorbeeld van Huffman coding uitgewerkt. In hoofdstuk 5 zal onder andere Huffman coding gebruikt worden voor het maken van de compressietool.

Zoals in „Results of a prototype television bandwidth compression scheme" (Robinson en Cherry, 1967) beschreven staat, werd run length encoding gebruikt in de jaren 60 en 70 voor het uitzenden van televisiesignalen. Het is echter moeilijk te zeggen dat deze implementatie van RLE ook het eerste voorkomen van RLE was. Het concept achter run length encoding is één van de eenvoudigste compressie-algoritmen tot op heden waardoor

[1] Cluster size recommendations for ReFS and NTFS – http://bit.ly/2YvtFDS.

het ontstaan moeilijk te achterhalen is. In deel 4.3.3 wordt een praktisch voorbeeld met run length encoding uitgewerkt.

De jaren 70 en 80 zorgden voor tal van belangrijke doorbraken binnen datacompressie. Dit kwam door de opkomst van het internet en de steeds groter wordende bestanden. Ook werd hardwarematige compressie (zoals prefix code met vaste lookup table voor tekstbestanden) steeds meer vervangen door dynamische compressie (codegewijs).

Deze eerste softwareoplossingen waren veelal implementatie van Huffman coding, eventueel met kleine aanpassingen. Eind jaren 70 werden de eerste Lempel-Ziv compressie-algoritmen uitgevonden: LZ77 en LZ78. Dit zijn de grondleggers van dictionary coding. Een veelgebruikte variant van LZ78 is LZW (1984). Net zoals Huffman coding de basis legde voor vele van de eerste softwareoplossingen, zorgden de grondleggers van dictionary coding voor vele nieuwe softwareoplossingen. De doorgroei van deze compressie-algoritmen is zichtbaar in figuur 12.1.

Het grootste verschil tussen prefix coding en dictionary coding zit in de naam zelf. Bij prefix coding wordt elk karakter vervangen door een prefix code terwijl bij dictionary coding een reeks van karakters vervangen kunnen worden door één enkele prefix code.

Eind jaren 80 en begin de jaren 90, door de digitalisering van afbeeldingen en muziek, begonnen lossy compressie-algoritmen steeds meer op te komen. Het verschil tussen lossless en lossy compressie-algoritmen wordt in deel 4.4 verder besproken.

4.3 Primitieve technieken: een voorbeeld

Aan de hand van een eenvoudig voorbeeld zal de basis werking van ASCII, run length encoding en Huffman coding uitgelegd worden. De innerlijke werking van deze compressie-algoritmen begrijpen is niet nodig om de kern van deze paper te begrijpen, maar dient als verduidelijking voor mogelijke vragen en als inleiding op hoofdstuk 5. Een basiskennis van binaire bomen is vereist.

4.3.1 Situering

Huffman coding en run length encoding zijn lossless compressie-algoritmen. Huffman coding pronkt in het opslaan van tekst gebaseerde bestanden, maar is bruikbaar voor tal van toepassingen. Run length encoding in zijn basisvorm is heel use case gebonden en gaat gemiddeld gezien niet zo goed presteren in tekstbestanden. De reden hiervoor zal duidelijk worden door het voorbeeld 4.3.3 en probleemstelling 4.3.2.1. Een blik op de prestatie van onder andere run length encoding en Huffman coding is te lezen in „Comparison of Lossless Data Compression Algorithms for Text Data” (Kodituwakku en U, 2010).

In dit voorbeeld wordt er van uitgegaan dat een gebruiker de zin “lennert eet veel” wenst op te slaan. Aan de hand van dit voorbeeld zal duidelijk worden hoe ASCII, Huffman coding en run length encoding werken en wat mogelijke problemen zijn.

4.3.2 ASCII encoding en decoding

Een makkelijke manier om tekst om te zetten naar een binaire reeks is door gebruik te maken van ASCII. ASCII is een tekenset waarbij elke teken een getal toegekend heeft gekregen. Dat getal kan dan weer eenvoudig binair voorgesteld worden. Er zijn meerdere conversielijsten online raadpleegbaar[2]

De zin "lennert eet veel"zou met ASCII binair voorgesteld worden als: “01101100 01100101 01101110 01101110 01100101 01110010 01110100 00100000 01100101 01100101 01110100 00100000 01110110 01100101 01100101 01101100”

Er zijn dus 16 karakters (inclusief spaties) die elk voorgesteld worden met acht bits. In totaal heeft deze tekst dus 128 bits nodig om te worden opgeslagen via ASCII.

Decoding kan met een conversietabel eenvoudig in de omgekeerde richting gedaan worden.

4.3.2.1 ASCII Probleemstelling 1: 8 bits per karakter

Een ASCII teken wordt steeds door acht bits, één byte voorgesteld. Dit wilt zeggen dat er theoretisch gezien maximaal 256 verschillende tekens kunnen gerepresenteerd worden. De standaard ASCII tekenset gebruikt echter maar zeven van die acht bits en is dus gelimiteerd tot 128 verschillende karakters. Dit maak conversie van speciale characters zoals chinese tekens onmogelijk.

4.3.3 RLE: run length encoding en decoding

Zoals besproken in deel 4.2.2 is de basis implementatie van RLE zeer eenvoudig. Run length encoding werkt door run en run value paren op te slaan in plaats van individuele karakters. Dit werkt door te kijken naar het eerste karakter in de reeks en te tellen hoeveel opeenvolgende karakters hetzelfde karakter zijn. Vervolgens wordt het aantal (run) opgeslagen gevolgd door het karakter (run value). Hierna wordt recursief gestart vanaf het eerstvolgende verschillende karakter.

Van RLE zijn enorm veel varianten gemaakt en veel compressie-algoritmen gebruiken RLE als onderdeel. Een interessante paper die zich dieper toespitst op RLE en een proof of concept variant is „Mespotine-RLE-basic v0.9 - An overhead-reduced and improved Run-Length-Encoding Method” (Mespotine, 2015).

In zijn basisvorm zou RLE de zin ’lennert eet veel’ als volgt comprimeren: ’1l1e2n1e1r1t1 2e1t1 1v2e1l’. Deze string kan op zijn beurt door ASCII voorgesteld worden.

[2] ASCII Conversion Chart – http://bit.ly/2w2IpxV.

4.3.3.1 RLE Probleemstelling 1: gecomprimeerd bestand groter dan bron

Uit het voorbeeld 4.3.3 blijkt dat RLE voor de zin "lennert eet veel"voor een output zorgt van meer dan 150% van de originele lengte. Run length encoding is voor deze situatie dus aanzienlijk groter dan het origineel niet gecomprimeerde bestand.

Dit zal het geval zijn voor veel gebruiker gegenereerde tekstbestanden en met RLE kan in het slechtste geval een bestandsgrootte van 200% bereken worden. Denk hierbij aan het comprimeren van het alfabet. RLE is zeer use case gebonden en wordt vaak gevarieerd gebruikt, maar kan in die bepaalde use cases voor een grote databesparing zorgen.

Dit probleem kan omzeild worden door geen run voor de run value te zetten wanneer deze één is.

4.3.4 Huffman coding

Het bericht kan ook voorgesteld worden door Huffman coding. Huffman coding valt onder de categorie prefix coding. Hierbij zal elk karakter voorgesteld worden door een prefix code dat op zijn beurt verwijst naar het karakter in bijvoorbeeld ASCII.

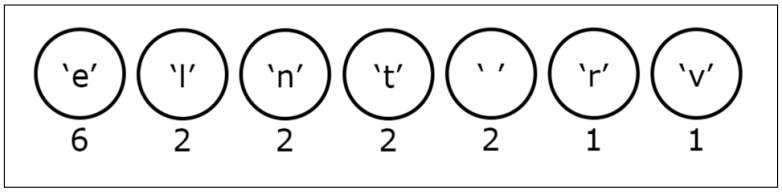
Doordat Huffman coding optimaal is, zal het meest voorkomende karakter steeds de kleinste prefix code toegewezen krijgen.

4.3.4.1 Huffman encoding stap 1: meerdere bomen

Als eerste stap dient een boom gemaakt te worden voor elk karakter met een toegevoegde eigenschap zijnde de frequentie van voorkomen. Deze bomen bestaan dus telkens uit 1 knoop (het hoofd).

Deze bomen moeten aflopend gesorteerd worden op deze eigenschap.

Een voorbeeld van hoe het resultaat van deze stap er voor de zin “lennert eet veel” kan uitzien is te vinden op figuur 4.1.

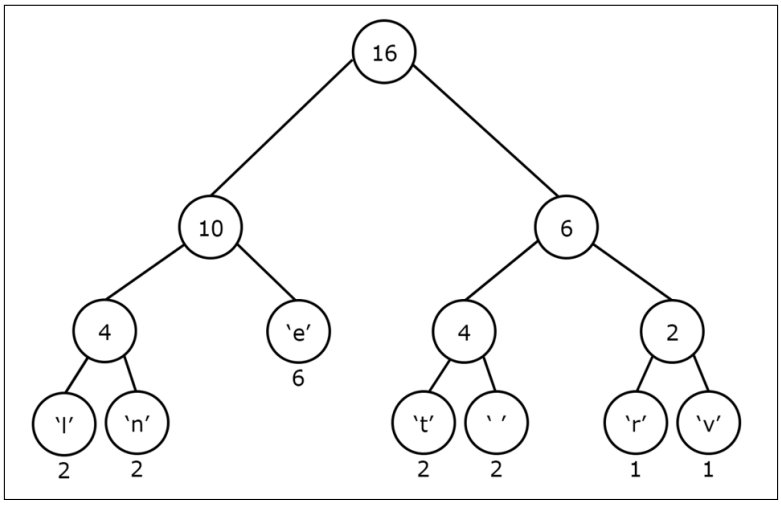


Figuur 4.1: Een voorbeeld van hoe het resultaat na stap 1 (4.3.4.1) van Huffman coding er voor de zin “lennert eet veel” kan uitzien. De volgorde van bomen met eenzelfde frequentie mogen van plaats gewisseld worden.

4.3.4.2 Huffman encoding stap 2: bomen samenvoegen

Als tweede stap moeten alle bomen samengevoegd worden tot één boom waarbij voorrang gegeven wordt aan diegene met het kleinste totaal van frequenties. Indien er evenveel frequenties zijn, heeft diegene met de kleinste diepte voorrang.

Een voorbeeld van hoe het resultaat van deze stap er voor de zin “lennert eet veel” kan uitzien is te vinden op figuur 4.2.



Figuur 4.2: Een voorbeeld van hoe het resultaat na stap 2 (4.3.4.2) van Huffman coding er voor de zin “lennert eet veel” kan uitzien. De knopen van een karakter met eenzelfde frequentie kunnen in sommige gevallen van plaats gewisseld worden.

4.3.4.3 Huffman encoding stap 3: prefix tabel (optioneel)

Als derde, optionele, stap kan voor elk karakter de prefix code gemaakt worden en bijgehouden worden in een lookup table. Dit wordt gedaan door het pad naar het blad van het karakter bij te houden startend uit het hoofd waarbij een stap naar links als 0 genoteerd wordt en een stap naar rechts als 1.

Het is ook mogelijk deze berekening van het pad steeds opnieuw te doen en dus geen gebruik te maken vaan een lookup table.

De bekomen binaire boom van stap 4.3.4.2 levert onderstaande lookup table op.

karakter	prefix
’e’	01
’l’	000
’n’	001
’t’	100
’ ’	101
’r’	110
’v’	111

4.3.4.4 Huffman encoding stap 4: encoding

Als vierde en laatste stap kan nu eenvoudigweg elk karakter vervangen worden door zijn bijhorende prefix code. De zin “lennert eet veel” wordt door de bovenstaande lookup table voorgesteld als: "000 01 001 001 01 110 100 101 01 01 100 101 111 01 01 000"

Het valt direct op dat deze reeks van bits (42 bits) veel kleiner is als in het ASCII voorbeeld (128 bits). Dit is de reden dat Huffman coding tot op heden de grondlegger is voor veel compressie-algoritmen. Hoewel Huffman coding lossless is en dus voornamelijk in andere lossless compressie-algoritmen gebruikt wordt, is Huffman coding ook te vinden als onderdeel van tal van lossy compressie-algoritmen.

4.3.4.5 Huffman decoding

De decoding is visueel het eenvoudigst met behulp van de binaire boom door links en rechts te gaan afhankelijk van de bit. Bij het vinden van een blad is een karakter gevonden en kan recursief terug van het hoofd begonnen worden. Binnen het compressie-algoritme zal een soort van lookup table gebruikt worden dat samen met het gecomprimeerd bestand dient opgeslagen te worden.

4.3.4.6 Huffman coding Probleemstelling 1: binaire boom niet opgeslagen

In het voorgaande deel (4.3.4.5) is bewezen dat via de Huffman coding reeks en de oorspronkelijke binaire boom het oorspronkelijk bericht volledig terug kan opgehaald worden (of via de lookup table). Er is echter enkel rekening gehouden met de Huffman coding reeks voor het bepalen van de bestandsgrootte. Volgens die logica zou de originele binaire boom (of lookup table) dus niet opgeslagen worden en bijgevolg zou de originele tekst niet meer te reconstrueren zijn. Een soort lookup table zal dus ook moeten opgeslagen worden als metadata.

4.3.4.7 Huffman coding Probleemstelling 2: gecomprimeerd bestand groter dan bron

Zoals in probleemstelling 4.3.4.6 besproken is moet een soort lookup table bijgehouden worden als metadata voor het gecomprimeerd bestand. Dit veroorzaakt echter een volgend potentieel probleem .

Wanneer zowel de Huffman coding reeks als de binaire zoekboom moeten opgeslagen worden bestaat de kans dat het gecomprimeerde bestand een grotere bestandsgrootte heeft dan het oorspronkelijk (door ASCII encoded) bestand. De mogelijkheid dat een gecomprimeerd bestand groter is dan een niet gecomprimeerd bestand is wederkerend bij datacompressie en lossless compressie in het bijzonder.

4.3.4.8 Huffman coding Probleemstelling 3: overlappende prefix codes

Door het gebruik van een binaire boom waarbij elk karakter voorgesteld wordt door een blad is het onmogelijk om een prefix code te bekomen die het begin is van een andere prefix code. In dit geval zou namelijk een knoop zijn aangeduid en geen blad (karakter).

Vooraleer er gebruik gemaakt werd van een binaire boom voor het bepalen van de prefix codes en het maken van de lookup table was dit echter wel een probleem. Zo konden prefix codes als 00, 001 en 00100 gelijktijdig voorkomen wat voor een gecomprimeerd bestand

zorgt dat nooit met 100% zekerheid gedecomprimeerd kan worden.

4.4 Lossless vs lossy datacompressie

Zoals in hoofdstuk 2 gedefinieerd slaat lossless en lossy op een categorie voor compressie-algoritmen. Bij lossless datacompressie is er geen verlies van 'kwaliteit' terwijl dit bij lossy datacompressie geen garantie is.

Het besproken Huffman coding principe valt onder lossless datacompressie. Immers, de originele boodschap (en eender welke vorm van input) is bit per bit reconstrueerbaar door het compressie-algoritme, er gaat dus geen data verloren. Een bit per bit reconstrueerbare clone is echter geen garantie voor lossless compressie-algoritmen aangezien metadata en andere zaken die geen impact hebben op "kwaliteit" wel verloren kunnen gaan.

Een typische use case voor lossless datacompressie zijn tekstdocumenten, de eindgebruiker wilt namelijk niet dat na het comprimeren letters verloren gaan in het document.

Zoals reeds besproken in deel 4.2.2 was de doorbraak van lossy datacompressie de opkomst van digitale afbeeldingen en muziek. Zo behoudt MP3 (lossy codec voor audio) bepaalde (combinaties van) audiofrequenties niet omdat die door het menselijke oor niet waarneembaar zijn. Maar ook waarneembare frequenties kunnen door de MP3 codec verloren gaan voor het besparen van data. JPEG is één van de oudste en bekendste vormen van lossy compressie-algoritme voor afbeeldingen.

Zoals besproken in „A Review of Image Compression Techniques" (Kaur en Pooja, 2016) kunnen audiobestanden door het gebruik van lossy datacompressie tot 90% kleiner worden in bestandsgrootte zonder een storende vermindering aan kwaliteit. Bij video kan een een nog grotere databesparing voorkomen, tot meer dan 99% zonder dat daar veel visueel waarneembaar verschil mee gepaard gaat. Immers, opeenvolgende beelden lijken meestal erg op elkaar. Bij stilstaande afbeeldingen is het kwaliteitsverlies door lossy datacompressie vaak wel zeer zichtbaar wanneer 90% of meer aan bestandsgrootte bespaard wordt.

De keuze voor lossless of lossy is use case gebonden. Bij het kiezen voor lossless datacompressie is een objectief onderzoek naar een goede balans tussen snelheid en databesparing voor de use case aangeraden. Bij lossy datacompressie zijn zowel objectieve als subjectieve onderzoeken mogelijk. In hoofdstuk 8 wordt dieper ingegaan op de verschillende manieren om de prestatie van een compressie-algoritme voor afbeeldingen te evalueren. In hoofdstuk 9 wordt een subjectief onderzoek gevoerd voor het bepalen van het geschikte afbeeldingsformaat voor een bepaalde use case.

5. Proof of concept datacompressietool

In deel 4.3.3 en 4.3.4 wordt uitgelegd hoe run length encoding en Huffman coding werken. Dit hoofdstuk focust zich op de implementatie van deze compressie-algoritme. Er is een proof of concept datacompressietool gebouwd in PHP en de werking hiervan zal toegelicht worden.

Deze datacompressietool is in staat om input bestand met de bestandsextensie 'txt' bestaande uit karakters die door ASCII voorgesteld kunnen worden (exclusief nummers) om te zetten naar een:

- Lbrle bestand dat gebruik maakt van een run length encoding gebaseerd compressie-algoritme waar altijd een frequentie voorzien wordt.
- Lbrle bestand dat gebruik maakt van een run length encoding gebaseerd compressie-algoritme waar een frequentie voorzien wordt indien deze groter is dan één.
- Lbrle bestand dat gebruik maakt van een Huffman coding gebaseerd compressie-algoritme.

Enkele screenshots van de afbeeldingsevaluatietool zijn raadpleegbaar in deel 12.2. De datacompressietool is online raadpleegbaar op de website van Lennert Bontinck[1] of downloadbaar via de GitHub repository[2].

[1] POC compressietool - BAP – http://bit.ly/2Qx0gX9.
[2] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

5.1 Gebruikte technologie

Deze datacompressietool en de achterliggende compressie-algoritmen zijn geschreven in PHP. Er is een grafische interface voorzien die gebruik maakt van HTML, CSS, JavaScript en Bootstrap. Dit maakt het mogelijk de tool eenvoudig lokaal te runnen door het gebruik van een webserver omgeving als XAMPP of hem online te zetten op een hosting platform.

Er is geen gebruik gemaakt van Libraries voor de compressie-algoritmen te voorzien, deze zijn volledig zelf geïmplementeerd aan de hand van de besproken theorie in hoofdstuk 4 en standaard PHP functies.

Er zit een minimale vorm van validatie in de datacompressietool om na te gaan of de gekozen bestanden voldoen aan de gestelde eisen.

5.2 Run length encoding - lang

De gebruiker voorziet een tekstbestand, dat enkel waarden bevat die voorgesteld kunnen worden door ASCII met uitzondering van de cijfers uit ons talstelsel, en klikt op de knop 'bestand encoderen' weergegeven op figuur 12.2. De gebruiker kan nu een door run length encoding gecomprimeerde lbrlea versie van de tekst in dat bestand downloaden via het scherm weergeven in figuur 12.3. Deze versie van run length encoding in de datacompressietool voorziet voor elk karakter de opeenvolgende frequentie links van het karakter. De theoretische werking van run length encoding is besproken in deel 4.3.3.

Achterliggend wordt hiervoor één PHP functie aangesproken: 'rle_encode_all' voorzien in 'algoritmes/rle.php'. De code van deze functie is hieronder weergegeven.

```
1  function rle_encode_all($input) {
2    $minimum_aantal_matches = 0;
3
4    if (!$input) {
5      return return ['', 0, 0];
6    }
7
8    $output = '';
9    $vorige = $letter = null;
10   $frequentie = 1;
11
12   foreach (str_split($input) as $letter) {
13       if ($letter === $vorige) {
14         $frequentie++;
15       } else {
16         if ($frequentie > $minimum_aantal_matches && $vorige !=
               null) {
17           $output .= $frequentie;
18         }
19       $output .= $vorige;
20       $frequentie = 1;
21     }
22
```

```
23      $vorige = $letter;
24    }
25
26    if ($frequentie > $minimum_aantal_matches) {
27      $output .= $frequentie;
28    }
29
30    $output .= $letter;
31
32  return [$output, strlen($input), strlen($output)];
33  }
```

Op regel twee in deze functie is de variabele voor het minimum aantal matches (de minimum run) in te stellen. Deze is standaard ingesteld op nul wat wilt zeggen dat elk karakter (run value) voorafgegaan wordt door de run. Dit gaat de bestandsgrootte te min, doordat het probleem beschreven in deel 4.3.3.1 kan voorkomen, maar is hoe de originele uitwerking van run length encoding is geschreven. Deze kan op één gezet worden om geen run te voorzien links van de run value indien deze gelijk is aan nul. Indien deze variabele op nul staat is het aantal karakters na de encoding maximaal dubbel zoveel als voor de encoding. Indien deze op nul staat, is het maximum aantal karakters na de encoding evenveel als voor de encoding.

Op regel vier wordt de controle uitgevoerd of de input niet leeg is. Indien dit het geval is wordt op regel vijf een lege string als antwoord gestuurd. Regel acht tot elf voorzien de nodige variabelen.

Regel 13 tot 23 worden voor elk karakter van de input string uitgevoerd. In deze herhaling wordt op regel 13 nagegaan of de vorige letter gelijk is aan de huidige. Indien dit het geval is, wordt op regel 14 de huidige frequentie (run) voor het huidige karakter (run value) met één verhoogd.

Indien de vorige letter niet gelijk is aan de huidige letter wordt op regel 16 gecontroleerd of de voorgaande letter reeds geïnitialiseerd was en of de run hoger is dan het ingestelde minimum. Indien dit het geval is, wordt op regel 17 de run toegevoegd aan de output string, vervolgens wordt op regel 19 de run value toegevoegd aan de output string. De run wordt dan ook teruggezet naar één en de run value wordt ingesteld naar het nieuwe karakter.

Na de herhaling wordt de huidige run (indien hoger dan de ingestelde minimum) en run value nog aan de output string toegevoegd. Op regel 32 wordt deze input string samen met de lengte van de originele string en de lengte van de gecomprimeerde string als antwoord verstuurd.

De logica in deze code maakt geen gebruik van PHP exclusieve functies en zou dus in de meeste programmeertalen na te maken moeten zijn.

5.3 Run length encoding - kort

De gebruiker voorziet een tekstbestand, dat enkel waarden bevat die voorgesteld kunnen worden door ASCII met uitzondering van de cijfers uit ons talstelsel, en klikt op de knop 'bestand encoderen' weergegeven op figuur 12.2. De gebruiker kan nu een door run length encoding gecomprimeerde lbrle versie van de tekst in dat bestand downloaden via het scherm weergeven in figuur 12.3. Deze versie van run length encoding in de datacompressietool voorziet links van elke run value de run indien deze groter is dan nul. De theoretische werking van run length encoding is besproken in deel 4.3.3.

Achterliggend wordt hiervoor één PHP functie aangesproken die veel korter is dan de optie besproken in deel 5.2: 'rle_encode' voorzien in 'algoritmes/rle.php'. De code van deze functie is hieronder weergegeven en komt overeen met de output van de functie uit deel 5.2 wanneer de minimum run value zou ingesteld zijn op één.

```
1 function rle_encode($input){
2   $output = preg_replace_callback('/(.)\1+/', function ($
        overeenkomst) {
3     return strlen($overeenkomst[0]) . $overeenkomst[1];
4   }, $input);
5
6   return [$output, strlen($input), strlen($output)];
7 }
```

Deze functie maakt gebruik van een PHP exclusieve functie die in staat is een bepaalde regex match te vervangen door iets. Op regel twee wordt gezocht naar alle instanties waar meer dan één opeenvolgende karakters hetzelfde zijn en vervangt deze met regel drie door de run en run value. Op regel drie wordt de door de functie verkregen gecomprimeerde string samen met de lengte van de originele string en de lengte van de gecomprimeerde string als antwoord verstuurd.

5.4 Run length decoding

De gebruiker voorziet een lbrlea of lbrle bestand, dat voordien door de datacompressietool gemaakt is, en klikt op de knop 'RLE decoderen' weergegeven op figuur 12.2. De gebruiker kan nu de inhoud van het originele tekstbestand downloaden via het scherm weergeven in figuur 12.4.

Achterliggend wordt hiervoor één PHP functie aangesproken: 'rle_decode' voorzien in 'algoritmes/rle.php'. De code van deze functie is hieronder weergegeven.

```
1 function rle_decode($input) {
2   return preg_replace_callback('/(\d+)(\D)/', function ($
        overeenkomst) {
3     return str_repeat($overeenkomst[2], $overeenkomst[1]);
4   }, $input);
5 }
```

Deze functie maakt gebruik van dezelfde PHP exclusieve functie uit deel 5.3. Deze gaat op zoek naar een run gevolgd door de run value (regel twee) en vervangt de match door de nodige hoeveelheid karakters (regel vier).

Indien de gebruikte preg_replace_callback of een gelijkaardige functie niet beschikbaar is in gewenste programmeertaal kan de string karakter per karakter overlopen worden. Wanneer een numerieke waarde ontdekt wordt, is er sprake van de run en is het volgende karakter de run value. Met een simpele lus kan voor run keer run value toegevoegd worden aan de output string in plaats van de run en run value.

5.5 Huffman encoding

De gebruiker voorziet een tekstbestand, dat enkel waarden bevat die voorgesteld kunnen worden door ASCII met uitzondering van de cijfers uit ons talstelsel, en klikt op de knop 'bestand encoderen' weergegeven op figuur 12.2. De gebruiker kan nu een door Huffman coding gecomprimeerde lbhuffman versie van de tekst in dat bestand downloaden via het scherm weergeven in figuur 12.3.

Achterliggend wordt hiervoor één PHP functie aangesproken: 'huffman_encode' voorzien in 'algoritmes/huffman.php'. Deze functie roept een andere functie op: 'recursive_huffman_lookup_table_generator' voorzien in 'algoritmes/huffman.php'. De code van deze functies is hieronder weergegeven.

```
1  function huffman_encode($input) {
2    $originele_input = $input;
3    $input_voor_karakters_bepaling = $input;
4    $karakters_met_frequentie = array();
5
6    while (isset($input_voor_karakters_bepaling[0])) {
7      $karakters_met_frequentie[] = array(substr_count($
           input_voor_karakters_bepaling, $
           input_voor_karakters_bepaling[0]), $
           input_voor_karakters_bepaling[0]);
8      $input_voor_karakters_bepaling = str_replace($
           input_voor_karakters_bepaling[0], '', $
           input_voor_karakters_bepaling);
9    }
10
11   $huffman_bomen = $karakters_met_frequentie;
12   sort($huffman_bomen);
13
14   while (count($huffman_bomen) > 1) {
15     $row1 = array_shift($huffman_bomen);
16     $row2 = array_shift($huffman_bomen);
17     $huffman_bomen[] = array($row1[0] + $row2[0], array($row1, $row
           2));
18     sort($huffman_bomen);
19   }
20
21   $lookup_table = [];
```

```
22    recursive_huffman_lookup_table_generator($lookup_table, is_array(
          $huffman_bomen[0][1]) ? $huffman_bomen[0][1] : $huffman_bomen)
          ;
23
24    $output = '';
25    for ($i = 0; $i < strlen($originele_input); $i++) {
26      $output .= $lookup_table[$originele_input[$i]];
27    }
28
29    return [$output, $lookup_table, strlen($input) * 8, strlen($
          output)];
30  }
```

Van regel één tot en met drie worden de nodige variabelen geïnitialiseerd. De lus van regel zes tot negen wordt gebruikt voor het bepalen van alle karakters in de input string en hun frequentie. Deze stellen de bomen voor met hun frequentie zoals in deel 4.3.4.1 voorgesteld en worden op regel twaalf gesorteerd op hun frequentie (klein naar groot).

Vervolgens wordt van regel 14 tot 19 een lus gemaakt zolang er meerdere bomen aanwezig zijn waarin de twee kleinste bomen samen genomen worden en één nieuwe boom maken zoals besproken in deel 4.3.4.2.

De lookup table wordt aangemaakt door een recursieve functie die aangeroepen wordt op regel 22 en verder in dit document besproken wordt. De lookup table wordt als variabele meegeven in deze functie op een manier zodat de originele variabele gebruikt kan worden en doorheen de iteraties éénzelfde lookup table aangepast wordt.

Uiteindelijk wordt aan de hand van de lookup table elk karakter vervangen door zijn prefix code in de lus van regel 25 tot 27. De string die deze lus genereert wordt dan teruggestuurd als antwoord samen met de lookup table, de bestandsgrootte in bit van de originele string en de bestandsgrootte in bit van de gecomprimeerde output string.

```
1  function recursive_huffman_lookup_table_generator(&$lookup_table, $
       karakter, $pad_naar_karakter = '') {
2    if (!is_array($karakter[0][1])) {
3      $lookup_table[$karakter[0][1]] = $pad_naar_karakter . '0';
4    } else {
5      recursive_huffman_lookup_table_generator($lookup_table, $
           karakter[0][1], $pad_naar_karakter . '0');
6    }
7    if (isset($karakter[1])) {
8      if (!is_array($karakter[1][1])) {
9        $lookup_table[$karakter[1][1]] = $pad_naar_karakter . '1';
10     } else {
11       recursive_huffman_lookup_table_generator($lookup_table, $
             karakter[1][1], $pad_naar_karakter . '1');
12     }
13   }
14 }
```

Deze recursieve functie wordt gebruikt voor het maken van de lookup table. De array wordt aanzien als boom en zolang een knoop kinderen heeft (array in array - if op regel

twee en zeven) moet er dieper gegaan worden in de boom tot een blad gevonden wordt. Een stap naar links wordt daarbij als nul opgeslagen en een stap naar rechts als één.

Van zodra een blad gevonden is, wordt het pad naar dat blad en het karakter van dat blad bijgehouden in de lookup table. Dit gebeurt in de else van regel vier tot zes en regel tien tot twaalf.

Er moet geen waarde teruggegeven worden aangezien de lookup table die meegeven wordt als parameter een verwijzing is naar de originele lookup table waardoor deze gelijk blijven tussen de functie en de oproepende functie.

5.6 Huffman decoding

De gebruiker voorziet een lbhuffman bestand, dat voordien door de datacompressietool gemaakt is, en klikt op de knop 'Huffman decoderen' weergegeven op figuur 12.2. De gebruiker kan nu de inhoud van het originele tekstbestand downloaden via het scherm weergeven in figuur 12.4.

Achterliggend wordt hiervoor één PHP functie aangesproken: 'huffman_decode' voorzien in 'algoritmes/huffman.php'. De code van deze functie is hieronder weergegeven.

```
1  function huffman_decode($input, $lookup_table) {
2    $lookup_table = (array) $lookup_table;
3    $input_array = str_split($input);
4    $huidige_bit_stream = "";
5    $output = "";
6
7    foreach ($input_array as $bit) {
8      $huidige_bit_stream .= $bit;
9      foreach ($lookup_table as $lookup_table_karakter=>$
           lookup_table_prefix_code) {
10       if ($lookup_table_prefix_code === $huidige_bit_stream) {
11         $output .= $lookup_table_karakter;
12         $huidige_bit_stream = "";
13         break;
14       }
15     }
16   }
17
18   return $output;
19 }
```

Van regel twee tot vijf worden de variabelen klaargemaakt voor gebruik.

Van regel zeven tot zestien worden alle bits van de input string overlopen. Op regel acht wordt eerst de huidige bit toegevoegd aan de huidige volgorde van bits sinds een match in de lookup table. Dan wordt aan de hand van een tweede lus van regel negen tot vijftien gekeken of de huidige volgorde van bits zonder match overeenkomt met een prefix code uit de lookup table. Indien dit het geval is, wordt het bijhorend karakter aan de output string

toegevoegd en de huidige volgorde van bits zonder match gereset.

Deze output string wordt uiteindelijk teruggestuurd op regel 18.

5.7 Opslaan van de gecomprimeerde bestanden

De besproken functies in de voorgaande delen slaan geen bestanden op, maar sturen steeds het gecomprimeerde bericht als variabele terug. Het opslaan van deze variabelen gebeurt in 'index.php' waar deze functies ook aangesproken worden.

Voor de functies die gebruik maken van run length encoding, besproken in deel 5.2 en 5.3, kan dit eenvoudig door de string op te slaan als een tekstbestand. De extensie hiervan wordt dan ingesteld als respectievelijk lbrlea of lbrle. De winst reflecteert zich hierbij dan ook direct op de bestandsgrootte van het gecomprimeerde bestand.

Voor de functie die gebruik maakt van Huffman coding besproken in deel 5.6 is dit moeilijker. De bekomen gecomprimeerde string is geen tekst maar een volgorde van bits. PHP bied geen voor de hand liggende manier om effectieve bytes op te slaan en aangezien deze implementatie ook enkel als proof of concept dient, is ervoor gekozen om deze op te slaan als json samen met de lookup table. Dit maakt het bekomen bestand met lbhuffman extensie veel groter dan het in een effectieve implementatie zou zijn.

5.8 Resultaten

Er zijn twee test tekstbestanden voorzien onder de map 'input' of downloadbaar via de website, namelijk:

- 'lennert_eet_veel.txt': een tekstbestand met de zin 'lennert eet veel' in. Deze zin werd ook gebruikt in de theoretische uitleg van hoofdstuk 4.
- 'dna_met_1_miljoen_bases.txt': een tekstbestand met één lange string van miljoen bases. Dit is het soort bestanden dat DNA compressie probeert te comprimeren besproken in deel 10.1.

Voor run length encoding is de winst het duidelijkst wanneer de tekst wordt weergegeven als aantal karakters. De resultaten hiervoor zijn te vinden in tabel 5.1. In deze tabel is duidelijk te zien dat run length encoding een heel use case gebonden compressie-algoritme is. Zo is het bestand na compressie in het beste geval even groot als dat het origineel zou zijn bij de zin 'lennert eet veel'. Dit is reeds besproken in deel 4.3.3.1. Het is met deze zin ook duidelijk dat door het gebruik van run length encoding in de variant waar de run enkel voorzien wordt indien deze groter is dan één, er nooit een groter bestand dan origineel kan verkregen worden. Voor het bestand met de miljoen bases is het voordeel van run length encoding wel zichtbaar. Het bestand bekomen door de tweede variatie heeft een verkleining van meer dan 6%.

Voor Huffman coding is de winst het duidelijkst wanneer de tekst wordt weergegeven als aantal bits. De resultaten hiervoor zijn te vinden in tabel 5.2. De afgebeelde grootte in bit is echter wel zonder lookup table wat in werkelijkheid ook steeds met het bestand mee moet opgeslagen worden.

De datacompressietool kan met eigen tekstbestanden getest worden op de website van Lennert Bontinck[3]. Een uitbreiding, zoals het gebruiken van zowel Huffman coding als run length encoding, kan eenvoudig voorzien worden door het downloaden van de code via de GitHub repository[4], maar wordt niet verder in deze bachelorproef besproken.

	origineel	**lbrle**	**lbrlea**
lennert eet veel	16	26	16
miljoen bases	1000000	1499216	937151

Tabel 5.1: Het aantal karakters nodig om het gekozen tekstbestand voor te stellen origineel en na de twee variaties run length encoding.

	origineel	**lbhuffman**
lennert eet veel	128	42
miljoen bases	8000000	2000000

Tabel 5.2: Het aantal bits nodig om het gekozen tekstbestand voor te stellen origineel en na Huffman coding (zonder lookup table).

5.9 Beperkingen met deze datacompressietool

Deze datacompressietool dient enkel als proof of concept en bevat daardoor enkele beperkingen.

De run en run value waardes bij run length encoding worden bij de encoding opgeslagen als ASCII waarden. Doordat het input bestand geen numerieke waardes mag bevatten, weet de decoder dat bij het vinden van eender welke numerieke waarde een run is gevonden. Als bij een implementatie wel numerieke waarden in het input bestand mogen voorkomen werkt deze decoder dus niet.

Een mogelijke oplossing is dat elk gecomprimeerd karakter verplicht een run en run value moet hebben. Op die manier is de eerste waarde in het bestand steeds de run en de daaropvolgende waarde de run value, waardoor onderscheid tussen een run en een numerieke waarde in het originele input bestand gemaakt kan worden. Dit veroorzaakt echter een bijkomend probleem: wat als de run hoger is dan negen en dus met twee karakters uitgedrukt moet worden? Het tweede karakter van de run wordt dan aanzien als run value wat de decoder laat falen. Ook hier zijn oplossingen voor, bijvoorbeeld door het opsplitsen van ’13a’ in ’9a4a’. De bachelorproef gaat niet dieper in op deze implementatie.

Bij de Huffman coding implementatie is in deel 5.7 reeds besproken dat de bestandsgrootte de besparing door dit compressie-algoritme niet representeert. De door Huffman coding

[3]Bachelorproef onderzoek: afbeeldingskwaliteit – http://bit.ly/2JIIBeE.
[4]Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

bekomen bits worden echter opgeslagen als ASCII waarden waardoor één bit voorgesteld wordt door acht bits.

6. Afbeeldingscompressie

Afbeeldingscompressie is een subdomein van datacompressie. Afbeeldingscompressie bestaat er uit een afbeelding in zo weinig mogelijk aantal bits op te slaan terwijl een aanvaardbare kwaliteit behouden blijft. Dit kan zowel via lossless als lossy compressie-algoritmen.

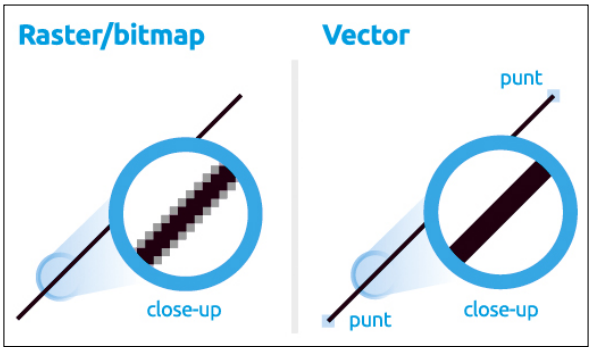
Afbeeldingscompressie is zeer belangrijk voor de snelheid en schaalbaarheid van IT-projecten alsook voor de gebruikerservaring. Bijna alle IT-projecten bevatten afbeeldingen, dit is zeker bij websites het geval.

In een officiële blog post van Google, „You and site performance, sitting in a tree...” (Google, 2010), wordt besproken dat een efficiënte webpagina steeds onder de twee seconden zou moeten geladen kunnen worden. In diezelfde post wordt zelfs aangehaald dat binnen een halve seconde de gebruiker reeds inhoud van de webpagina zou moeten zien.

Uit een nog recenter onderzoek door Akamai, „Akamai Online Retail Performance Report: Milliseconds Are Critical” (Akamai, 2017), blijkt dat bij een laadtijd van meer dan drie seconden op een mobiele webpagina meer dan de helft van de bezoekers de webpagina verlaat.

Afbeeldingen zijn meestal de grootste bestanden die bij het laden van een webpagina gedownload moeten worden. Het is dan ook met een juiste keuze aan afbeeldingscompressie dat de meeste tijd voor de bezoeker kan gespaard worden.

Dit hoofdstuk licht toe welke soorten afbeeldingscompressie er bestaan. Ook enkele afbeeldingsformaten zullen besproken worden samen met hun voordelen en nadelen. De mogelijke problemen en oplossingen bij de implementatie van nieuwe generatie afbeel-



Figuur 6.1: Illustratie dat het verschil tussen raster en vector weergeeft (Shoppen, 2017)

dingsformaten zullen ook besproken worden.

Hoofdstuk 8 bespreekt hoe de kwaliteit van een afbeeldingsformaat objectief en subjectief beoordeeld kan worden. In hoofdstuk 9 wordt voor een bepaalde use case een geschikt afbeeldingsformaat gezocht aan de hand van een subjectief onderzoek met een voor deze bachelorproef geschreven afbeeldingsevaluatietool.

6.1 Raster vs vector afbeeldingsformaten

Binnen afbeeldingsformaten kan men een onderscheid maken tussen twee soorten afbeeldingsformaten: raster en vector. Net zoals bij de keuze tussen lossless en lossy is er geen eenduidig antwoord welke beter is. De keuze is use case gebonden. Het grote verschil tussen de twee is de manier waarop ze de data bijhouden voor het weergeven van de afbeelding.

Raster afbeeldingen bestaan uit een grid van kleine punten, meestal vierkantjes, die pixels genoemd worden. Voor elke pixel wordt een bepaalde kleur bijgehouden. Bij bepaalde afbeeldingsformaten kan ook de doorzichtigheid van een pixel bijgehouden worden. Het is door het naast elkaar weergeven van al deze pixels dat we een afbeelding verkrijgen. Hoe dichter deze pixels bij elkaar staan hoe scherper de afbeelding oogt. Wanneer je voldoende inzoomt op een afbeeldingen met een raster afbeeldingsformaat kan je deze pixels visueel zien.

Vector afbeeldingen werken niet met pixels, maar kunnen gezien worden als een soort tekening. Een afbeelding bestaat dan uit allerlei vormen, denk hierbij onder andere aan cirkels, rechthoeken en (gebogen) lijnen. Voor elk van deze vormen wordt dan de kleur bijgehouden, de start en eindpunten, de boog in graden, ... Het is door het samenvoegen van al deze vormen dat we de finale afbeelding verkrijgen. Het grote voordeel van vector afbeeldingsformaten is dat je kan blijven inzoomen zonder dat je daarbij kwaliteit verliest. Er is namelijk geen sprake van een gelimiteerd aantal pixels maar van bepaalde vormen die zo groot of klein als gewenst getekend kunnen worden. Figuur 6.1 geeft dit duidelijk weer.

Raster afbeeldingsformaten zijn ideaal voor het opslaan van foto's aangezien de beeldsensor van een camera ook bestaat uit meerdere punten (pixels). Een foto met natuurlijke objecten

is veelal ook (veel) te complex om op een efficiënte en realistische manier te kunnen voorstellen met de vormen die raster afbeeldingsformaten ondersteunen. Grafisch werk zoals logo's en iconen kunnen vaak wel opgeslagen worden door deze vormen waardoor een vector afbeeldingsformaat aangeraden is.

Gekende software voor het maken en bewerken van raster afbeeldingen is Adobe Photoshop, voor het maken en bewerken van vector afbeeldingen is dit Adobe Illustrator. De afbeeldingsformaten die in dit hoofdstuk besproken zullen worden zijn allen voorbeelden van raster afbeeldingsformaten. Een gekend voorbeeld van vector afbeeldingsformaten is het SVG afbeeldingsformaat.

6.2 Afbeeldingsformaten

Het nemen van een foto met een digitale camera komt overeen met het openstellen van de beeldsensor aan licht voor een bepaalde duur (sluitertijd). De gegevens die gedurende die tijd waargenomen worden, kunnen direct verwerkt worden en gecomprimeerd opgeslagen worden, bijvoorbeeld als een JPEG. Dit is hoe de meeste smartphone camera's te werk gaan. Bij vele, voornamelijk professionele, toestellen kan ingesteld worden dat er geen gecomprimeerd afbeeldingsformaat gebruikt moet worden, maar maar een RAW afbeeldingsformaat.

Hieronder worden enkele afbeeldingsformaten verder toegelicht. Het is belangrijk om te weten dat dit niet de enige afbeeldingsformaten zijn die er bestaan. De lijst van afbeeldingsformaten blijft groeien en bestaande afbeeldingsformaten kunnen extensies krijgen om gekende problemen als bepaalde artefacten tegen te gaan.

Een mogelijke manier om deze artefacten tegen te gaan is het gebruik van een andere wavelet zoals besproken in „A New Algorithm for Removing Blocking Artifacts in JPEG Compressed Images" (Nallaperumal, Ranjani, Varghese en Annam, 2006). Recente doorbraken binnen artificiële intelligentie maken het zelfs mogelijk op een nog meer dynamische manier artefacten in afbeeldingsformaten tegen te gaan zoals besproken in „S-Net: A Scalable Convolutional Neural Network for JPEG Compression Artifact Reduction" (Zheng, Sun, Tian en Chen, 2018).

6.2.1 RAW

Een RAW afbeeldingsformaat bevat alle ruwe, onbewerkte en ongecomprimeerde gegevens die de beeldsensor heeft vastgelegd. In een RAW afbeeldingsformaat wordt ook tal van metadata bijgehouden zoals de gebruikte camera en lens, hun instellingen… De bestandsgrootte van een RAW bestand is hierdoor aanzienlijk.

RAW is geen afkorting noch een echt afbeeldingsformaat zoals JPEG of PNG maar een benaming voor een groep van afbeeldingsformaten die voldoen aan de benoemde eigenschappen. Het effectieve afbeeldingsformaat kan verschillen van merk tot merk en zelfs van toestel tot toestel. Zo zijn de RAW bestanden gebruikt voor het onderzoek in hoofdstuk

9 afkomstig van een Nikon toestel en zijn ze opgeslagen in het NEF afbeeldingsformaat.

Hoewel er reeds voorstellen zijn gedaan voor een open RAW standaard om bewerkingen makkelijk te maken, zoals DNG van Adobe, is er tot op heden een grote diversiteit aan RAW afbeeldingsformaten te vinden. Dit vormt binnen afbeeldingscompressie en de evaluatie ervan enkele nadelen. Doordat er zo veel verschillende RAW afbeeldingsformaten zijn en dus veel uiteenlopende licenties en rechten, is het een uitdaging een compressie-algoritme voor een afbeeldingsformaat te maken dat alle RAW afbeeldingsformaten ondersteunt als input.

Starten van een RAW afbeeldingsformaat voor het evalueren van afbeeldingscompressie is echter wel aangeraden aangezien zelfs het gebruik van een lossless afbeeldingsformaat voor verlies van metadata kan zorgen zoals eerder besproken. Het weglaten van deze metadata kan onderdeel zijn van het gekozen afbeeldingsformaat en bijhorend compressie-algoritme. Als deze metadata niet inbegrepen is in het inputbestand wordt het weglaten ervan niet gerepresenteerd in de eindscore, wat voor een vals beeld kan zorgen.

6.2.2 PNG

Portable Network Graphics is een lossless afbeeldingsformaat. Het is ontwikkeld door de Portable Network Graphics Development Group met een eerste beta in 1995, een draft versie voor W3C eind 1995, een officiële W3C voorstelling op 1 juli 1996 en goedgekeurd als W3C aanbeveling op 1 oktober 1996. Datums uit „History of the Portable Network Graphics (PNG) Format” (Roelofs, 2009).

PNG was gemaakt als vervanger van het toen veelgebruikte GIF afbeeldingsformaat dat net zoals vele andere compressie-algoritmen een nachtmerrie van licenties en patenten aan het worden was.

PNG had als doel een afbeeldingsformaat te worden dat zeer flexibel is, makkelijk te gebruiken is op het internet en allerlei soorten afbeeldingen ondersteund. Meer dan 20 jaar later slaagt het daar nog altijd in.

6.2.2.1 PNG: werking

De werking van PNG wordt niet diepgaand uitgelegd omdat hierover volledige papers geschreven kunnen worden. De informatie over de werking van PNG is uit het uitstekende boek over datacompressie: *Data Compression: The Complete Reference* (Salomon, 2006) gehaald waar nog dieper op de werking van PNG wordt ingegaan.

Een PNG bestand is opgebouwd uit verschillende delen dat ’chunks’ genoemd worden. Een chunk bestaat uit:

- Grootte van het dataveld in deze chunk.
- Naam van deze chunk. Vier letters lang.
- Dataveld.

- Een CRC voor het valideren van de chunk. Deze is steeds 32 bits groot.

Chunks kunnen verplicht (critical chunks) of optioneel (ancillary chunks) zijn. Optionele chunks kunnen door een decoder genegeerd worden en kunnen zaken als metadata bevatten. Critical chunks moeten door de decoder gelezen kunnen worden en de CRC controle moet kloppen anders wordt er een error weergegeven. Een chunk is verplicht wanneer de eerste letter een hoofdletter is. Indien de tweede letter een hoofdletter is duid het op een standaard door PNG voorziene chunk, anders is het een uitbreiding dat door een extensie op PNG is toegevoegd. De derde letter is steeds een hoofdletter en de vierde letter is een hoofdletter wanneer de chunk niet gekopieerd mag worden.

Een PNG bestand kan één van de vijf volgende kleursoorten hebben:

- RGB met acht of zestien bitplanes.
- RGB met acht of zestien bitplanes en een alpha kanaal voor transparantie.
- Palette met één, twee, vier of acht bitplanes.
- Grijsschaal met één, twee, vier, acht of zestien bitplanes.
- Grijsschaal met acht of zestien bitplanes en een alpha kanaal voor transparantie.

De decoder kan een PNG progressief inladen wanneer er gebruik gemaakt wordt van de Adam zeven interlacing functionaliteit. Deze interlacing verdeelt de afbeelding in gelijke blokken van 64 pixels (8x8) die in zeven stappen ingeladen wordt. Eerst wordt de pixel links vanboven ingeladen en weergegeven over alle 64 pixels in die blok. Wanneer dit voor alle blokken gedaan is worden in totaal twee pixels ingeladen per blok die terug gekopieerd worden over de gehele blok. Het aantal in te laden pixels blijft verdubbelen tot alle 64 pixels van dat blok ingeladen zijn en dus het gehele PNG bestand ingeladen is.

De effectieve compressie van PNG gebeurt door pixel prediction en deflate. Eerst wordt een waarde voor een pixel berekend door pixel prediction, de 'predicted value', waarna het verschil tussen de pixel en de predicted value opgeslaan wordt door het te encoden met deflate. De pixel omzetten naar het verschil van de predicted value is niet hetgeen dat voor de datacompressie zorgt. Dit zorgt ervoor dat de pixel gerepresenteerd wordt door een volgorde van bits dat met een hogere compressieratio kan gecomprimeerd worden door het lossless compressie-algoritme deflate.

6.2.2.2 PNG: voordelen

Het PNG afbeeldingsformaat biedt tal van voordelen ten opzichte van zijn voorgangers en lossy tegenstanders. Enkele van deze voordelen zijn:

- Beschikt over een alpha kanaal waardoor doorzichtigheid meegegeven kan worden als een getal tussen 0 (volledig doorzichtig) en 100 (geen doorzichtigheid).
- Wordt gezien als één van de standaarden voor lossless afbeeldingsformaten waardoor het een goede support heeft overheen verschillende hardware en software.
- Lossless afbeeldingsformaat waardoor er geen kwaliteit verloren gaat.
- De decoder kan progressief de afbeelding inladen, startend met een weergave tegen lage resolutie tot deze uiteindelijk volledig is ingeladen.

- Goede uitbreidbaarheid waardoor metadata en andere randvariabelen aan een PNG bestand kunnen toegevoegd worden terwijl het bestand compatibel blijft met oudere versies.
- Een keuze uit meer dan 16 miljoen kleuren dankzij het RGB kleurenprofiel met alpha kanaal. Een groot contrast tegenover GIF dat maar 256 kleuren ondersteund.

6.2.2.3 PNG: nadelen

PNG heeft echter ook enkele minpunten, voornamelijk te wijten aan het feit dat PNG ontwikkeld is om te gebruiken op het internet.

- Geen standaard ondersteuning voor geanimeerde beelden.
- Grote bestandsgrootte door zijn lossless eigenschap.

6.2.3 JPEG

Joint Photographic Experts Group is technisch gezien geen afbeeldingsformaat maar een codec. Het is een compressie-algoritme dat oorspronkelijk op een lossless of lossy manier te werk kan gaan. De lossless variant wordt echter niet grootschalig gebruikt voor afbeeldingsformaten en wordt daarom niet verder toegelicht in deze bachelorproef. Wanneer gesproken wordt over het JPEG als afbeeldingsformaat verwijst dit meestal naar JPEG/JFIF of JPEG/Exif welke wel een afbeeldingsformaat zijn.

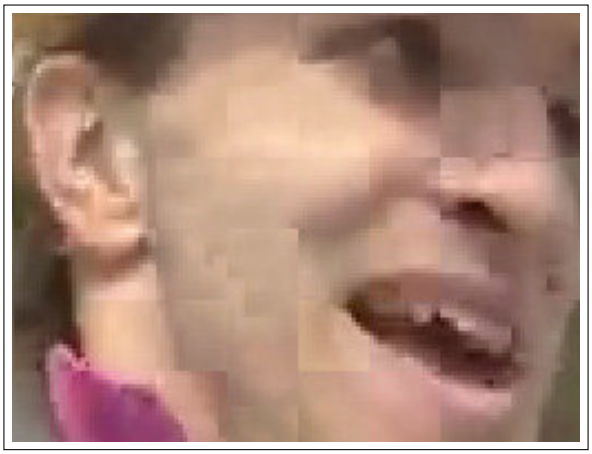
JPEG wordt doorgaans als lossy compressie-algoritme gebruikt voor het opslaan van afbeeldingen waarbij een controle over bestandsgrootte en kwaliteit gewenst is. Dit is mogelijk doordat JPEG verschillende parameters ondersteund om de werking van het compressie-algoritme te beïnvloeden en dus ook de kwaliteit en bestandsgrootte van het uiteindelijk bestand.

De ontwikkeling van het JPEG compressie-algoritme is begonnen in 1986 en de JPEG standaard is gemaakt in 1992. Deze bestaat uit 7 delen met de laatste officiële revisie in 1994. Uiteraard zijn er tal van uitbreidingen (of extensies zoals deel 3 van de ISO/IEC 10918 standaard ze benoemt) gemaakt tot op heden. Datums overgenomen van de officiële JPEG website (WG1, 2014).

JPEG is ook gekend onder de kortere vorm JPG omdat dit de extensie is die het meest gebruikt wordt voor de JPEG codec. Dit omdat in oudere versies van het Windows besturingssysteem, zoals bijvoorbeeld Windows 98, een bestandsextensie maximaal drie karakters lang mocht zijn. In de huidige versies van Windows is deze beperking echter niet meer actief waardoor de JPG en JPEG bestandsextensies door elkaar gebruikt kunnen worden.

6.2.3.1 JPEG: werking

De werking van JPEG en JPEG/JFIF wordt niet diepgaand uitgelegd omdat hierover volledige papers geschreven kunnen worden. De informatie over de werking van JPEG



Figuur 6.2: Zwaar gecomprimeerde afbeelding met blokartefacten. Afbeelding overgenomen uit „Temporal Error Concealment with Block Boundary Smoothing" (Il Choi en Jeon, 2004)

en JPEG/JFIF is uit het uitstekende boek over datacompressie: *Data Compression: The Complete Reference* (Salomon, 2006) gehaald waar nog dieper op de werking van JPEG en JPEG/JFIF wordt ingegaan.

JPEG is een compressie-algoritme en geen volledig afbeeldingsformaat. Daarom zijn belangrijke zaken als de aspect ratio en het kleurprofiel niet opgenomen in het JPEG compressie-algoritme zelf, maar in de bijhorende afbeeldingsformaten zoals JPEG/JFIF en JPEG/Exif.

Kenmerkend aan JPEG is dat de lossy variant enorm veel inputmogelijkheden bevat die controle geven over wat juist verloren mag gaan door het compressie-algoritme en in welke mate. Deze input variabelen aanpassen past dus het compressieratio van JPEG aan.

JPEG is een symmetrisch compressie-algoritme wat wilt zeggen dat de decoder dezelfde stappen van de encoder uitvoert in omgekeerde volgorde. Deze stappen zijn:

1. Indien de te encoderen afbeelding in kleur is, wordt het kleurprofiel aangepast naar een luminantie/ chrominantie kleurprofiel. Dit kleurprofiel past beter bij de perceptie van het menselijke oog. Deze ziet kleine variaties in luminantie echter zeer sterk terwijl variaties in chrominantie veel minder worden waargenomen. JPEG maakt hier gebruik van door veel chrominantie informatie te verkleinen en aan te passen zodat het beter gecomprimeerd kan worden. Deze stap is optioneel, maar aangeraden om het beste compressieratio met JPEG te bereiken.
2. Indien de te encoderen afbeelding in kleur is en in het luminantie/ chrominantie kleurprofiel wordt de chrominantie waarde over meerdere aaneensluitende pixels gedeeld. Overheen hoeveel pixels de chrominantie gedeeld moet worden is een instelbare input variabele.
3. De afbeelding wordt ingedeeld over blokken van 64 pixels (8x8) die elk apart gecomprimeerd zullen worden. Dit worden data units genoemd. Het is door deze groepering en onafhankelijke datacompressie dat bekende artefacten als blokartefacten voorkomen bij JPEG. Een voorbeeld waar deze blokartefacten goed zichtbaar zijn is weergegeven in figuur 6.2.

4. Door het gebruik van DCT wordt een map van 64 (8x8) frequentie componenten gemaakt. Dit zijn de pixels nu voorgesteld als een getal. In deze stap is reeds enige informatie verloren gegaan door het gebruik van DCT, een wiskundig algoritme, maar dit verlies aan informatie is niet hetgeen dat voor het grootte compressieratio van JPEG zorgt.
5. De 64 frequentie componenten per data unit worden nu gedeeld door 64 te voorziene input variabele: de kwantisatiecoëfficiënten (QCs). Het bekomen resultaat wordt afgerond naar een natuurlijk getal. Het is deze stap waar het lossy compressie-algoritme data gaat snoeien, hoe groter de QCs hoe meer kwaliteitsverlies.
6. De bekomen resultaten worden door een combinatie van run length encoding en Huffman coding gecomprimeerd bijgehouden. Er kan ook gekozen worden om QM coder te gebruiken in plaats van Huffman coding.
7. De laatste stap voorziet de door run length encoding en Huffman coding gecomprimeerde data van de nodige metadata en geeft dit finaal pakket terug als output.

Het effectieve afbeeldingsformaat, bijvoorbeeld JPEG/JFIF, zorgt dan voor de nodige metadata met informatie over de afbeelding zelf. Het is ook dat afbeeldingsformaat dat de mogelijkheid voor progressief decoding voorziet.

6.2.3.2 JPEG: voordelen

JPEG is één van de meest gebruikte lossy compressie-algoritmen voor afbeeldingen en heeft onder andere daarom enkele belangrijke voordelen zoals:

- Uitstekende support overheen verschillende hardware en software.
- De decoder kan progressief de afbeelding inladen, startend met een weergave tegen lage resolutie tot deze uiteindelijk volledig is ingeladen.
- Door de mogelijkheid om als lossy compressie-algoritme te werken kan JPEG een enorm kleine bestandsgrootte aannemen afhankelijk van de instellingen.
- De kleinere bestandsgrootte creëert de mogelijkheid om meer foto's per second te verwerken. Op deze manier kan een digitale camera meer opnames maken in burst wanneer er voor JPEG/Exif is gekozen als afbeeldingsformaat in plaats van een RAW afbeeldingsformaat.

6.2.3.3 JPEG: nadelen

JPEG heeft uiteraard ook enkele nadelen. De voornaamste zijn:

- Door de lossy eigenschap aan de hand van clustering kunnen allerlei vormen van artefacten voorkomen.
- Geen mogelijkheid voor doorzichtigheid.
- Onnatuurlijke afbeeldingen zoals logo's zijn zeer gevoelig aan het verlies van scherpe lijnen en het ontstaan van artefacten door scherp contrast.

6.2.4 JPEG2000

JPEG2000 is, zoals de naam doet vermoeden, een opvolger van JPEG gemaakt door de Joint Photographic Experts Group. Deze waren van mening dat door de opkomst van het internet een betere variant van JPEG nodig was.

In maart 1997 kondigde de Joint Photographic Experts Group aan dat ze een nieuwe standaard voor afbeeldingscompressie wouden ontwikkelen en open stonden voor bijdrages. Dit wekte de interesse van vele scholen en bedrijven. Zo had enkele maanden na de aankondiging de Universiteit van Arizona samen met SAIC reeds een proof of concept voorgesteld op basis van een WTCQ compressie-algoritme in plaats van het DCT compressie-algoritme gebruikt bij JPEG.

In augustus 2000 besloot het Joint Photographic Experts Group dat de toen huidige draft versie klaar was om voor te stellen als nieuwe standaard aan de ISO. In December van 2000 werd dit voorstel goedgekeurd en sinds heden is JPEG2000 te vinden onder ISO/IEC 15444.

Die ISO standaard bestaat op het moment van schrijven uit 14 delen, het laatste deel is gepubliceerd in 2013. De meest gebruikte afbeeldingsformaten voor JPEG2000 zijn die beschreven in deel één en twee. De gebruikte variant van JPEG2000 voor het onderzoek van hoofdstuk 9, JPF, is beschreven in het tweede deel 2 (ISO/IEC 15444-2).

Hoewel JPEG2000 veel voorkomend is in videocompressie en afbeeldingscompressie daar waar kwaliteit belangrijk is zoals cinema en medische afbeeldingen, is het nooit de opvolger geworden van JPEG die de Joint Photographic Experts Group voor ogen had.

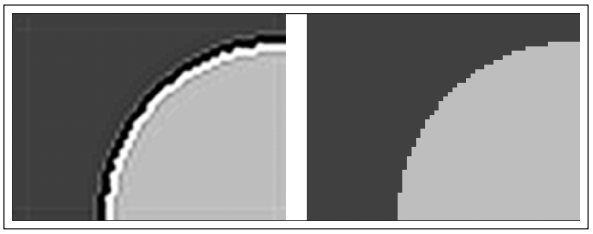
6.2.4.1 JPEG2000: werking

Doordat JPEG2000 meerdere revisies kent, elk met hun eigen encoders en decoders, wordt de effectieve werking van het compressie-algoritme niet uitgelegd in deze bachelorproef. JPEG2000 is een zeer uitgebreid en complex compressie-algoritme dat zowel lossless als lossy te werk kan gaan. De standaard is gedefinieerd in ISO/IEC 15444.

JPEG2000 wordt een schaalbaar compressie-algoritme genoemd doordat de decoding op verschillende manieren kan gebeuren. Zo kunnen de bits op een andere manier gerangschikt en deels weggelaten worden om een lagere resolutie variant te maken.

Een belangrijk verschil met JPEG is dat het geen gebruik meer maakt van de besproken 8x8 data units. Dit zorgt ervoor dat de storende blokartefacten niet meer voorkomen bij JPEG2000. JPEG2000 kan wel ringartefacten bevatten, deze zijn weergegeven in figuur 6.3.

JPEG2000 maakt gebruik van wavelets en het quantization compressie-algoritme.



Figuur 6.3: Links: gecomprimeerde afbeelding met ringartefacten. Rechts: oorspronkelijke afbeelding. Afbeeldingen van Barth, 2010

6.2.4.2 JPEG2000: voordelen

Aangezien JPEG2000 door de Joint Photographic Experts Group zelf als opvolger van JPEG is benoemd, zijn sommige van de voordelen van JPEG2000 dezelfde als die van JPEG.

De voornaamste voordelen van JPEG2000 zijn:

- In vergelijking met JPEG kan JPEG2000 zowel als lossless en lossy compressie-algoritme gebruikt worden.
- JPEG2000 heeft voor de meeste use cases betere kwaliteit dan JPEG voor een afbeelding met dezelfde bestandsgrootte. Naarmate de compressieratio stijgt, wordt dit voordeel groter. (Zoals aangetoond in studies als „JPEG vs. JPEG2000: An objective comparison of image encoding quality" - Ebrahimi, Chamik en Winkler, 2004)
- Kent meerdere varianten wat voor zeer veel flexibiliteit zorgt.
- Minder artefacten dan JPEG.
- De decoder kan progressief de afbeelding inladen, startend met een weergave tegen lage resolutie tot deze uiteindelijk volledig is ingeladen.
- Ondersteuning voor transparantie. In latere versies niet besproken in deze bachelorproef is ook support voor animatie aanwezig.
- Schaalbaar en tal van andere voordelen bij het gebruik van JPEG2000 als intra-frame datacompressie schema in videocompressie. Deze paper gaat niet verder in op het gebruik van JPEG2000 binnen videocompressie.

6.2.4.3 JPEG2000: nadelen

Hoewel het plan van JPEG2000 om de nieuwe standaard te worden enigszins gelukt is binnen videocompressie, is dit in afbeeldingscompressie niet het geval. Dit en nog enkele limitaties van JPEG2000 zorgen onder andere voor meer volgende nadelen:

- Slechte internetbrowser support: momenteel enkel ondersteund op Safari voor macOS en iOS.
- Doordat de verschillende ISO revisies steeds andere bestandsextensies toelichten is er geen achterwaartse compatibiliteit met oude decoders.
- Encoding duurt met de standaard encoders doorgaans langer dan bij JPEG. Het encoding en decoding proces vereist ook meer systeemresources dan JPEG.

6.2.5 WEBP

WebP is een afbeeldingsformaat dat door Google is uitgebracht in 2010 en tot op heden uitgebreid wordt.

WebP is een veelbelovend afbeeldingsformaat voor het internet. Het ondersteunt een zeer grote variatie van use cases. Het kan lossless en lossy gebruikt worden. Het presteert in zijn lossless variant gemiddeld gezien beter dan PNG. De lossy variant presteert gemiddeld gezien dan weer beter dan JPEG.

De prestatie van het WebP afbeeldingsformaat is reeds objectief en subjectief getest door zowel Google zelf als door onafhankelijke partijen. De resultaten zijn daar gelijklopend en steeds in het voordeel WebP. De objectieve evaluatie tussen JPEG en WebP (KeyCDN, 2019) alsook dat tussen PNG en WebP (KeyCDN, 2018) van KeyCDN preekt zowel op vlak van kwaliteit als snelheid in het voordeel van WebP.

Dit alles is niet alleen te danken aan het budget dat Google heeft om verdere ontwikkeling voor dit afbeeldingsformaat te voorzien, maar ook door de macht die het bedrijf heeft. Ze laten hun eigen projecten zoals Android, ChromeOS, YouTube, Gmail, de Google Play Store en meer zo veel mogelijk WebP gebruiken. Door dat dit een groot deel van de markt is, worden concurrenten onrechtstreeks verplicht dit nieuw afbeeldingsformaat ook te gebruiken.

6.2.5.1 WEBP: werking

De werking van de WebP wordt in dit deel beknopt uitgelegd aan de hand van de officiële Google documentatie[1].

De lossy variant van WebP is een afbeeldingsformaat dat zoals vele nieuwe generatie afbeeldingsformaten ontstaan is uit een intra-frame compressie-algoritme voor videocompressie, in dit geval VP8, een compressie-algoritme dat in 2008 door Google ontwikkeld is. Het is gebaseerd op het block prediction compressie-algoritme.

Het compressie-algoritme voor de lossy variant deelt de afbeelding dus op in verschillende segmenten die macroblocks genoemd worden. Een macroblock wordt bijgehouden door de verschillen met de voorgaande macroblock bij te houden, zoals PNG dat doet voor pixels. Dit wordt predictive coding genoemd. De resulterende data is dan net zoals JPEG omgezet via DCT wat resulteert in een resultaat met veel nul waardes. Tot deze stap werkt het compressie-algoritme volledig lossless. Na deze stap wordt het resultaat lossy comprimeerd aan de hand van Quantization. Dat resultaat wordt vervolgens entropy-coded.

De lossless WebP variant gebruikt een compressie-algoritme dat volledig door Google geschreven is. Ze gebruiken hier onder andere een variatie van Huffman coding voor.

[1] WebP Compression Techniques – http://bit.ly/2I58kdN.

6.2.5.2 WEBP: voordelen

Het gebruik van WebP bied vele voordelen, enkele van de voornaamste zijn:

- In vergelijking met andere 'nieuwe generatie' afbeeldingsformaten heeft WebP een goede internetbrowser support. Volgens caniuse.com is WebP ondersteund in elke grote internetbrowser buiten Safari.
- Zowel de lossless als lossy variant presteren gemiddeld gezien beter dan respectievelijk PNG en JPEG. (Zie 6.2.5)
- Ondersteunt transparantie en animatie.

6.2.5.3 WEBP: nadelen

WebP heeft vooral ondersteuningsgerelateerde problemen. De voornaamste zijn:

- Geen support in Safari en enkele kleinere of gedateerde internetbrowsers: Internet Explorer, KaiOS en Blackbarry Browser.
- Geen progressieve decoding mogelijk.
- Geen standaard support in Adobe Photoshop, maar gratis plug-ins beschikbaar. In het onderzoek uit hoofdstuk 9 wordt de voor het onderzoek gebruikte plug-in toegelicht.

6.2.6 HEIF/HEIC

High Efficiency Image Format is in een codec ontwikkeld door de Moving Picture Experts Group dat het zelf benoemt als afbeeldingsformaat. Het is herkend als standaard in het twaalfde deel van ISO 23008.

Het voornaamste gebruik van HEIF als afbeeldingsformaat is bij intra-frames bij H.265/HEVC videocompressie.

Je kan het aanzien als een heel geavanceerde vorm van JPEG. Er is ondersteuning voor blokken van 64 x 64 pixels voor compressie in plaats van de 8 x 8 bij JPEG. Het maakt gebruikt van arithmetische codering (CABAC) en geavanceerde voorspellingen... Door de complexiteit van HEIF kunnen alleenstaande papers geschreven worden omtrent de werking. Deze werking beschrijven valt dus niet in de scope van deze bachelorproef.

Met de opkomst van iOS 11 in 2017 was Apple het eerste bedrijf dat HEIF grootschalig gebruikte voor opslag van afbeeldingen. Alle videobestanden waren sindsdien intern opgeslagen met de H.265/HEVC codec en alle afbeeldingen als HEIF met de de HEIC bestandsextensie.

De overstap van JPEG naar het nieuwe generatie afbeeldingsformaat HEIF was voor Apple interessant op verschillende vlakken. De voornaamste voordelen waren kwaliteits- en snelheidswinst. Dit resulteerde dan ook in een kleinere bestandsgrootte waardoor er meer afbeeldingen opgeslagen kunnen worden op een iOS toestel.

Ook de ondersteuning voor burst foto's en live foto's kwam deze keuze ten voordele, het is

namelijk mogelijk meerdere afbeeldingen op te slaan onder één HEIF bestand.

6.2.6.1 HEIF: voordelen

HEIF en het door Apple gebruikte HEIC biedt als nieuwe generatie afbeeldingsformaat tal van voordelen. De voornaamste zijn:

- Zowel lossless als lossy operatie mogelijk.
- Photoshop CC ondersteuning sinds het einde van 2018. Deze bestanden worden aanzien als RAW.
- Een uitgebreid assortiment aan functionaliteit zoals het opslaan van meerdere afbeeldingen in één bestand. Deze functionaliteiten zijn interessant voor burst foto's, HDR foto's...
- Ondersteuning voor transparantie en animatie.
- HEIF biedt ook tal van voordelen bij het gebruik in videocompressie die niet verder in deze bachelorproef besproken worden.

6.2.6.2 HEIF: nadelen

Net zoals WebP heeft HEIF één groot nadeel: ondersteuning. Apple lost dit op door bij het delen van een foto de afbeelding te converteren naar JPEG, maar daarmee gaan ook alle voordelen van HEIF verloren...

Desondanks Apple gebruiker is van HEIF/HEIC is er nog geen ondersteuning voor in de Safari internetbrowser. Ook andere gekende internetbrowsers bieden geen ondersteuning voor dit afbeeldingsformaat.

Ook is er geen progressieve decoding mogelijk.

6.3 De juiste keuze

De juiste keuze van afbeeldingsformaat maken is geen eenvoudige taak en zeer use case gebonden. Hoewel nieuwe afbeeldingsformaten tal van interessante voordelen bieden, is vooral compatibiliteit een wederkerend probleem. Hier bestaan relatief eenvoudige oplossingen voor in webomgevingen, waarvan enkele besproken zijn in deel 6.4.4. De keuze voor een nieuw afbeeldingsformaat bij on premise applicaties is iets lastiger om te implementeren en wordt kort besproken in deel 6.4.4. Indien de use case professioneel drukwerk omvat is een keuze voor een raster afbeeldingsformaat ten sterkste aangeraden. Deze afbeeldingsformaten zijn niet verder besproken in deze bachelorproef.

6.3.1 Functievereisten

Het is belangrijk om voor elke use case grondig na te denken welke afbeeldingsformaten de beste keuzes zijn. De selectie verfijnen kan je reeds eenvoudig doen door naar enkele

	PNG	JPEG	JPEG200	WebP	HEIF
Lossless	Ja	Deels*	Ja	Ja	Ja
lossy	Nee	Ja	Ja	Ja	Ja
Transparantie	Ja	Nee	Ja	Ja	Ja
Animatie	Nee	Nee	Deels*	Ja	Ja
Progressief decoden	Ja	Ja	Ja	Deels*	Nee

Tabel 6.1: Overzichtstabel van enkele kernfunctionaliteiten per afbeeldingsformaat. Dit wordt verder besproken in deel 6.3.1.

functievereisten als ondersteuning voor transparantie te kijken.

Een korte overzichtstabel van enkele kernfunctionaliteiten per afbeeldingsformaat is te vinden in figuur 6.1. Hier zijn enkele opmerkingen bij:

- JPEG2000 ondersteunt pas sinds ISO 15444 deel drie animatie onder de vorm van Motion JPEG2000 (.mj2). In deze bachelorproef is echter de meest voorkomende versie .jpf uit deel twee besproken. Deze ondersteunt geen animatie.
- WebP ondersteunt geen progressieve decoding, maar wel incrementele decoding. Dit zorgt ervoor dat het wel mogelijk is reeds 'iets' weer te geven terwijl decoding (en zelfs download) nog gaande is.

6.3.2 Ondersteuning

De algemene ondersteuning voor JPEG en PNG is zeer uitgebreid. Alle recente internetbrowsers en besturingssystemen kunnen er perfect met om. Dit is één van de grootste redenen waarom deze oudere afbeeldingsformaat nog zo dominant aanwezig zijn. Nieuwe afbeeldingsformaten falen namelijk vaak doordat er een slechte ondersteuning is en niet door een slecht compressie-algoritme.

Zo heeft HEIC geen browserondersteuning tot op heden. JPEG2000 heeft een hele slechte internetbrowser ondersteuning met enkel ondersteuning in Safari onder de gekende browsers. WebP heeft een aanvaardbare internetbrowser ondersteuning met momenteel enkel geen ondersteuning in Safari onder de gekende internetbrowser.

Een compleet overzicht is beschikbaar in figuur 6.2

6.4 Implementatiemogelijkheden voor nieuwe afbeeldingsformaten

Het gebruik van nieuwe afbeeldingsformaten kan afgeschrikt worden wanneer je leest dat de ondersteuning nog niet optimaal is. Er zijn echter tal van manieren om toch de juiste afbeelding weer te geven wanneer er geen ondersteuning beschikbaar is. Dit wordt in onderstaande delen kort besproken voor zowel implementatie in webomgevingen als on premise omgevingen.

	PNG	JPEG	JPEG200	WebP	HEIF
Chrome	Ja	Ja	Nee	Ja	Nee
Edge	Ja	Ja	Nee	Ja	Nee
Firefox	Ja	Ja	Nee	Ja	Nee
Safari	Ja	Ja	Ja	Nee	Nee
Internet explorer	Ja	Ja	Nee	Nee	Nee
Opera	Ja	Ja	Nee	Ja	Nee
Blackberry browser	Ja	Ja	Nee	Nee	Nee
KaiOS browser	Ja	Ja	Nee	Nee	Nee

Tabel 6.2: Overzichtstabel van internetbrowser ondersteuning per afbeeldingsformaat. Data verkregen van caniuse.com in mei 2019.

6.4.1 Webomgeving

Binnen webomgevingen zijn tal van manieren om een alternatieve afbeelding op te geven als terugval afbeelding, mocht het laden van een afbeelding mislukken. Dit zorgt ervoor dat een internetbrowser die het gekozen afbeeldingsformaat niet ondersteunt de terugval afbeelding weergeeft. Deze is doorgaans dan een PNG of JPEG. Dit kan op verschillende manieren bereikt worden.

6.4.2 Manueel

Een eenvoudige en universeel werkende manier om een terugval afbeelding op te geven, is door gebruik te maken van een picture tag in HTML. Deze ziet er als volgt uit:

```
1 <picture>
2   <source srcset="pad/naar/afbeelding.webp" type="image/webp">
3   <source srcset="pad/naar/afbeelding.heic" type="image/heic">
4   <source srcset="pad/naar/afbeelding.jpf" type="image/jpx">
5   <source srcset="pad/naar/afbeelding.png" type="image/png">
6   <source srcset="pad/naar/afbeelding.jpg" type="image/jpeg">
7   <img src="pad/naar/afbeelding.jpg">
8 </picture>
```

De volgorde is hier enorm van belang. Internetbrowsers die de picture tag niet ondersteunen negeren de source tags ook en herkennen enkel de img tag en geven de afbeelding binnen die tag ingesteld weer. Indien een internetbrowser de picture tag wel ondersteunt, zal hij het type uit de eerste sourceset die hij ondersteunt nemen als bron voor de afbeelding. De internetbrowser werkt hier van boven naar onder en stopt bij een match.

6.4.3 Geautomatiseerd

Er zijn tal van mogelijkheden om op een geautomatiseerde manier gebruik te maken van nieuwe afbeeldingsformaten. Vele caching diensten, zoals Cloudflare, bieden de mogelijkheid automatisch elke afbeelding te converteren naar verschillende afbeeldingsformaten en diegene weer te geven met de kleinst mogelijke bestandsgrootte.

Voor WordPress en tal van andere CMS'en zijn er ook plug-ins voorzien die op eenzelfde manier te werk gaan. Voor WordPress is er bijvoorbeeld de WebP Express[2] plug-in.

Sommige automatisaties kiezen ervoor de afbeelding direct op te slaan in alle afbeeldingsformaten wat de opslagruimte te min gaat maar response time te goeie doet. Anderen houden enkel het bronbestand bij en genereren het gevraagde formaat bij aanvraag. Dit gaat dan weer te min van de response time en het cpu gebruik van de hosting server. Dit is uiteraard ook manueel te implementeren.

Voor de meeste afbeeldingsformaten zijn ook JavaScript decoders beschikbaar. Dit is JavaScript code dat afbeeldingen van een bepaald afbeeldingsformaat omzet naar een ander afbeeldingsformaat. Deze conversie gebeurt op het toestel van de bezoeker en vereist dus geen systeemresources van de hosting server. De conversie gebeurt meestal naar een lossless afbeeldingsformaat, zoals PNG, omdat op die manier geen kwaliteit verloren gaat en PNG een goede internetbrowser ondersteuning heeft. Deze conversie kan altijd gebeuren of wanneer de bezoeker zijn internetbrowser het normale afbeeldingsformaat niet ondersteunt. Een voorbeeld van een JavaScript decoder om het WebP afbeeldingsformaat naar PNG om te zetten is WebPJS van Dominik Homberger[3].

6.4.4 On-premise omgeving

Bij een on premise omgeving zijn er ook verschillende mogelijkheden om een terugval afbeelding in te stellen. In een situatie als die van Apple (besproken in 6.2.6.2) kan gekozen worden om een converter in te bouwen die het afbeeldingsformaat omzet naar een wel ondersteund formaat bij het bekijken of delen van een afbeelding. Dit kan echter heel intensief voor de CPU worden.

Een andere oplossing is een encoder en/of decoder te voorzien in de installatie die de ondersteuning voor het afbeeldingsformaat levert. Dit bestaat in het geval van WebP voor zowel macOS, Windows als bepaalde Linux distributies.

Er kan uiteraard ook gekozen worden om de terugval afbeelding effectief mee op te slaan op het toestel. Er kan dan gebruik gemaakt worden van een simpele universeel aanspreekbare controle die het juiste afbeeldingsformaat selecteert. Deze controle kan ook tijdens de installatie gedaan worden en zo enkel de afbeeldingen in het nodige afbeeldingsformaat over te zetten naar het toestel. Dit kan echter voor compatibiliteitsproblemen zorgen wanneer het systeem geüpdatet wordt en bepaalde ondersteuning wijzigt.

[2] WebP Express – http://bit.ly/2JHjUix.

[3] WebPJS - Google's new image format WebP for not supported browsers (with alpha-channel) – http://bit.ly/2JKiKD1.

7. Videocompressie

Videocompressie is een subdomein van datacompressie. Videocompressie bestaat uit een video (verzameling van opeenvolgende afbeeldingen die frames genoemd worden) in zo weinig mogelijk aantal bits op te slaan terwijl een aanvaardbare kwaliteit behouden blijft. Dit kan zowel via lossless als lossy compressie-algoritmen. Lossless videocompressie wordt in de praktijk echter zelden gebruikt voor web doeleinden door de grote hoeveelheid data die het in beslag neemt. Lossless videocompressie wordt vooral voor medische doeleinden gebruikt.

De output van een lossless video codec kan oplopen tot een bestand dat vijf tot wel twintig keer groter is dan een lossy variant met een visueel zeer gelijkaardig resultaat. Een lossy variant kan met enig verlies van kwaliteit meer dan 100 keer kleiner zijn dan een niet gecomprimeerde variant (Bhaskaran en Konstantinides, 1995). Het is door deze hoge compressieratio dat de keuze voor een goede video codec zo belangrijk is en dat het (live) streamen van video's mogelijk is. Zonder lossy compressie zouden de meeste toestellen ook niet genoeg systeemresources hebben om de video vloeiend af te spelen.

De meest gebruikte lossy video codecs zijn H.265/HEVC en motion JPEG2000. Deze hebben beiden ook een lossy modus. De lossy modus van H.265/HEVC wordt in dit hoofdstuk besproken. Er wordt in deze bachelorproef niet verder ingegaan op lossy video codecs.

Dit hoofdstuk licht enkele van de bekendste video codecs toe die grootschalig gebruikt worden door websites als YouTube en Netflix.

7.1 Intra- vs inter-frame

Inter-frame en intra-frame datacompressie zijn technieken voor videocompressie die gezamenlijk of individueel gebruikt kunnen worden door een video codec. Dit kan volledig lossless of lossy gebeuren.

Intra-frame datacompressie kan het best vergeleken worden met de compressie die binnen een afbeeldingsformaat gebeurt. Het is een compressie-algoritme dat één alleenstaand frame comprimeert met enkel de data die het heeft van dat frame. Dit kan bijvoorbeeld door pixel prediction (zoals PNG besproken in deel 6.2.2.1). Het is niet ongewoon dat alleenstaande afbeeldingsformaten gebruikt worden voor intra-frame compressie. JPEG2000 wordt vaak gebruikt voor videobestanden waar kwaliteit een grote rol speelt (in zijn lossless of lossy variant). Tegenwoordig is echter de omgekeerde verschijning meer voorkomend, dat compressie-algoritmen, oorspronkelijk gebruikt voor intra-frame videocompressie, overgezet worden naar alleenstaande afbeeldingsformaten. Denk hierbij aan WebP dat van VP8, een voorganger van AV1, afkomstig is. HEIC komt oorspronkelijk ook uit intra-frame datacompressie bij H.265/HEVC.

Inter-frame datacompressie maakt voor het comprimeren niet alleen gebruik van het huidige frame maar ook voorgaande of volgende frames. Zo is het mogelijk om bijvoorbeeld verwijzingen te behouden naar éénzelfde achtergrond binnen een videofragment overheen de frames. Het aantal frames waarnaar verwezen kan worden is afhankelijk van de gebruikte codec. Aangezien videofragmenten vaak terugkerende elementen hebben (zoals de achtergrond) voor een langdurige periode kan deze vorm van datacompressie voor enorme besparingen zorgen.

Een probleem dat bij inter-frame datacompressie opduikt, is wanneer een bestand dat gebruik maakt van dit compressie-algoritme achteraf bijgesneden wordt. De verwezen frames (of delen van het frame bij croppen) kunnen hierdoor verloren zijn waardoor het decoden faalt. Hoewel sommige video bewerkingsoftware hier rekening mee houden door de verwijzingen naar frames die verwijderd gaan worden eerst te vervangen door de effectieve data, is dit geen garantie.

7.2 Video codecs

In die hoofdstuk worden video codecs besproken en geen containers. Een container is, zoals de naam doet vermoeden, een verpakking voor alle data die opslagen moet worden en eventuele metadata. Deze containers bepalen de bestandsextensie, enkele gekende video containers zijn: MP4, AVI en MKV. Bij video's slaat de codec dus data op in een passende container. Ook de ondertiteling en dergelijke zijn opgenomen in de container.

Een codec, de coder-decoder, is de technologie verantwoordelijk voor de encoding en decoding van een bestand dat gebruik kan maken van één of meerdere compressie-algoritmen. In het geval van een video codec is dit de technologie die het videofragment omzet in een zo klein mogelijke collectie van bits.

7.2.1 H.264-AVC

H.264-AVC, waarbij AVC voor 'advanced video coding' staat, is een onderdeel van de MPEG-4 ISO standaard dat in deel tien van deze standaard voor het eerst toegelicht wordt. Één van de eerste edities van ISO/IEC 14496-10 is online raadpleegbaar[1] en geeft een inzicht in de basiswerking van deze video codec.

H.264-AVC is aangekondigd in mei 2003 en is daarmee de oudst besproken video codec in deze bachelorproef. Desondanks zijn leeftijd is het nog altijd één van de meest gebruikte video codecs op het web en in het algemeen. Dit is te danken aan de jarenlange verdere ontwikkeling van de standaard. De huidige versie van de ISO standaard is momenteel versie 25 uit april 2017. Deze voegt documentatie over het gebruik van HLG met H.264-AVC toe aan de ISO. Een veel gebruikte HDR standaard.

H.264-AVC is een uitbreiding op vorige standaarden van de MPEG-4 standaard dat onder andere de volgende belangrijke extra functionaliteit met zich meebrengt:

- Multi-picture inter-picture prediction. Deze techniek biedt tal van voordelen op voorgaande technieken. Het voornaamste is de mogelijkheid om tot wel 16 verschillende referentie frames te kunnen bijhouden voor het comprimeren van één frame. In vorige (bekende) standaarden waren hooguit twee referentie frames mogelijk. Deze uitbreiding kan voor (veel) kwaliteitswinst en een (veel) hoger compressieratio zorgen in tal van scenario's. Bijvoorbeeld bij een frame waar een achtergrond bestaat uit een combinatie van de achtergronden van meerdere frames.
- Flexibelere interlace opties die het mogelijk maken de kijker het gevoel te geven dat de frame rate twee keer hoger is dan ze werkelijk is door twee verschillende frames met elkaar te combineren.
- Tal van technieken die de decoder geschikt maken om te werken met livestreams waar enkele frames soms verloren gaan.

7.2.1.1 H.264-AVC: voordelen

H.264-AVC is een standaard die reeds jarenlang wereldwijde implementaties kent. Dit is te danken aan ondere andere volgende voordelen:

- Een compressieratio tot meer dan het dubbel dan dat van de voorgangers zoals MPEG-2 (Nemcic, Vranjes en Rimac, 2007) voor dezelfde perceptuele kwaliteit. Dit resulteert in een kleinere bestandsgrootte.
- Een grote ondersteuning zowel op vlak van input frame afbeeldingsformaten voor de encoder als in internetbrowsers. Enkel ondersteuning op de Opera Mini browser ontbreekt volgens de gegevens van caniuse[2].
- Ondersteuning voor livestreams met tal van functies om artefacten zoveel mogelijk te bestrijden.
- Achterwaartse compatibiliteit voor codecs gebaseerd op H.263 en H.261.

[1] ISO/IEC 14496-10 – http://bit.ly/2ws11rd.
[2] Can I use MPEG-4/H.264 – http://bit.ly/30WgcXS.

7.2.1.2 H.264-AVC: nadelen

H.264-AVC biedt ook enkele nadelen, de voornaamste zijn:

- Heeft meer systeemresources nodig dan voorgaande codecs voor zowel het encoding als decoding proces. Dit kan tragere chipset moeilijkheden geven om een video tegen zijn volle frame rate af te spelen. Ook het streamen van meerdere H.264-AVC encoded video's kan zijn tol eisen.
- H.264-AVC is een zeer complex compressie-algoritme wat het moeilijk maakt om als derde partij extensies te voorzien.
- H.264-AVC is ontwikkeld in een tijdperk waar full HD (1920 x 1080 pixels) een zeer hoge resolutie was. De maximum ondersteunde resolutie van H.264-AVC is dan ook maximaal 4096x2304 pixels. 4K (3840 x 2160 pixels) is dus nog net ondersteund maar hogere resoluties niet meer. De bestandsgrootte van een H.264-AVC groeit ook aanzienlijk eens de resolutie boven full HD gaat, wat het voor 4K streaming minder interessant maakt.
- H.264-AVC vereist complexe licenties, die bij een herziening in 2015 nog complexer zijn geworden, voor zowel encoding als decoding. Deze licenties moeten zowel betaald worden door de bedrijven die encoders en decoders voorzien (internetbrowsers, apps, ...) als diegene die video's publiek beschikbaar maken (DVD's, YouTube, ...). Dit is gebaseerd op de verkochte eenheden (bv: individuele software en video's) of op het aantal abonnees (bv: streaming diensten). Er is een minimum van 100 duizend gebruikers vooraleer licenties nodig zijn, daarna kan een licentie tot wel 20 cent per extra gebruiker zijn.

7.2.1.3 H.264-SVC

H.264-SVC, waarbij SVC voor 'scalable video coding' staat, is een extensie op de in deel 7.2.1 besproken H.264-AVC videocompressie standaard. Zoals de naam doet vermoeden verbetert H.264-SVC de schaalbaarheid van H.264-AVC.

H.264-AVC kan slecht in één resolutie geëncodeerd worden. Om de optie te voorzien een gebruiker te laten kiezen in welke resolutie deze de video wil bekijken was dus het maken van verschillende video bestanden nodig. Dit nam niet alleen meer tijd en bestandsruimte in dan één groot bestand hebben, maar kwam ook nadelig uit wanneer er dynamisch tussen verschillende resoluties geswitcht wou worden bij bijvoorbeeld een variërende bandbreedte tijdens het streamen.

H.264-SVC biedt een oplossing door één bestand te kunnen gebruiken voor het weergeven van meerdere resoluties. Dit maakt het eerder besproken dynamisch switchen tussen resolutie afhankelijk van de bandbreedte mogelijk. Iets cruciaal bij (live) streamen waar het niet altijd mogelijk is een cache te maken en dus bij een tekort aan bandbreedte geswitcht kan worden naar een lagere resolutie om te alle tijden 'iets' aan de eindgebruiker te kunnen laten zien. Hiervoor was zowel binnen hardware als software extra functionaliteit nodig op zowel netwerk als codec laag. Een uitgebreid artikel over H.264-SVC is „Overview of the Scalable Video Coding Extension of the H.264/AVC Standard" (Schwarz, Marpe en Wiegand, 2007).

7.2.2 H.265/HEVC

H.265/HEVC is de officiële opvolger van H.264-AVC dat het eerst is voorgesteld in april 2013. Het zorgt voor een bestandsgrootte die gemiddeld 25 tot 50% kleiner is dan H.264-AVC met visueel gelijkaardige kwaliteit („Coding Performance of H.262, H.264, and H.265", 2015). De gebruikte inter-frame datacompressie is de basis voor HEIF, besproken in deel 6.2.6.

Een ander belangrijk voordeel van deze video codec is dat het resolutie tot 8192×4320 pixels ondersteunt, ongeveer het dubbele van zijn voorganger H.264-AVC. H.265/HEVC ondersteunt dus nog net 8K UHD waardoor de eerste adoptie voornamelijk in cinema's plaatsvond.

De werking van H.265/HEVC is meer een uitbreiding op H.264-AVC dan een compleet nieuwe uitwerking. De voornaamste aanpassingen waardoor extra efficiëntie kan bereikt worden zijn:

- Een verbeterd inter-frame compressie-algoritme dat de basis legt voor HEIF.
- Ondersteuning voor een groter vlak (64x64 pixels in plaats van 16x16 pixels) voor patroon compressie en difference-coding.
- Een beter compressie-algoritme voor het gokken van een pixel (pixel prediction) door een betere kennis van de camera beweging en hoe een pixel zich verplaatst tegenover het voorgaande en volgende frames.

Hoewel H.265/HEVC nog complexer is dan zijn voorganger H.264-AVC, heeft dit voornamelijk impact op de encoding tijd en niet zo zeer op de decoding tijd. In 2014 zijn in een tweede revisie extensies toegevoegd voor schaalbaarheid.

7.2.2.1 H.265/HEVC: voordelen

De voornaamste voordelen bij het gebruik van H.265/HEVC over H.264-AVC zijn:

- Ondersteuning voor resoluties tot 8192×4320 pixels, waar 8K UHD net bijhoort.
- Gemiddelde besparing van 25 tot 50% ten opzichte van H.264-AVC met een visueel gelijkaardige kwaliteit („Coding Performance of H.262, H.264, and H.265", 2015).

7.2.2.2 H.265/HEVC: nadelen

De voornaamste nadelen bij het gebruik van H.265/HEVC zijn:

- Complexe licenties zoals zijn voorganger H.264-AVC die sterk kunnen oplopen, tot meer dan twintig keer zo duur als H.264-AVC.
- Relatief slechte browser support enkel volledige ondersteuning op recente versies van iOS Safari en gedeeltelijke ondersteuning op Safari voor macOS, Internet Explorer en Edge.
- Een (veel) langere encoding en iets langere decoding tijd dan H.264-AVC door de toegevoegde complexiteit.

7.2.3 AV1

AV1 is open source en royalty-free video codec dat komaf maakt met de complexe en dure licenties verbonden aan andere video codecs zoals H.264-AVC en H.265/HEVC. Het is ontwikkeld door AOMedia (Alliance for Open Media). Na een aankondiging in september 2015, samen met de opstart van AOMedia, is de eerste versie vrijgegeven in maart 2018, waarbij het ook het eerste project van AOMedia was.

AOMedia is een bedrijf van Google, Amazon, Netflix, Microsoft, Mozilla, Intel en Cisco. Grote bedrijven die een oplossing wouden bieden voor de problemen met de licenties van de voorgaande besproken video codecs. Hoewel AOMedia een non-profit is, hebben deze bedrijven er financieel baat bij dat AV1 doorgroeit naar de standaard voor videocompressie. Op die manier doen ze een immense besparing op licenties.

AV1 is gebaseerd op VP9, wat op zijn beurt gebaseerd is op VP8 en wiens inter-frame compressie-algoritme de basis legde voor het WebP afbeeldingsformaat. De codec begon zjin ontwikkeling als VP10, maar Google besloot het project over te plaatsen naar een afzonderlijk non-profit bedrijf om de redenen die hierboven reeds beschreven zijn.

Het vecht momenteel tegen H.265/HEVC om de nieuwe standaard video codec te worden.

7.2.3.1 AV1: voordelen

AV1 is een veelbelovende video codec en bied tal van voordelen. De voornaamste zijn:

- Gratis in gebruik en makkelijk in implementatie door zijn open source en royalty-free eigenschap.
- Een vergelijkbaar of beter compressieratio dan H.265/HEVC.
- Een betere internetbrowser ondersteuning dan H.265/HEVC door de relaties met Google en Mozilla. Een volledig overzicht wordt gegeven in deel 7.3.2.
- Meerdere 'levels' die het mogelijk maken om ondersteuning voor een hogere resolutie of frame rate toe te voegen. De huidige maximale resolutie is 7680 x 4320 pixels met een frame rate van 120 frames per seconde.

7.2.3.2 AV1: nadelen

Ten opzichte van H.265/HEVC heeft AV1 maar één noemenswaardig nadeel buiten de ondersteuning die nog niet optimaal is doordat de codec nog niet zo lang uit is. Het ondersteunt nog geen realtime encoding waardoor livestreaming met AV1 op het moment van schrijven nog niet mogelijk is.

AV1 heeft echter wel reeds een betere ondersteuning dan H.265/HEVC (op vlak van internetbrowsers), heeft geen complexe licenties en heeft een gelijkaardige encoding en decoding tijd ten opzichte van H.265/HEVC.

	H.264/AVC	H.264/SVC	H265/HEVC	AV1
Compressieratio	+	+	++	++
Betalend	Ja	Ja	Ja	Nee
Browsersupport	+	+	–	-
Realtime encoding	Ja	Ja	Ja	Nee
Schaalbaar	Nee	Ja	Ja	Ja
Maximum resolutie	4K	4K	8K	Uitbreidbaar

Tabel 7.1: Overzichtstabel van kernfunctionaliteit per video codec.

	H.264/AVC	H.264/SVC	H265/HEVC	AV1
Chrome	Ja	Ja	Nee	Ja
Edge	Ja	Ja	Ja	Gepland
Firefox	Ja	Ja	Nee	Ja
Safari	Ja	Ja	Ja	Nee
Internet explorer	Ja	Ja	Ja	Nee
Blackberry browser	Ja	Ja	Nee	Nee
Opera	Ja	Ja	Nee	Ja
Opera Mini	Nee	Nee	Nee	Nee

Tabel 7.2: Overzichtstabel van internetbrowser ondersteuning per video codec. Data verkregen van caniuse.com in mei 2019.

7.3 De juiste keuze

De keuze voor een juiste video codec is zeer use case gebonden en de juiste beslissing gaat gepaard met een goede keuze van de gebruikte audio codec en container. Verder onderzoek is hier dus aangeraden. De overzichten voor functionaliteit in deel 7.3.1, ondersteuning bij internetbrowsers in deel 7.3.2 en met betrekking tot licenties in deel 7.3.3 kunnen als startreferentie gebruikt worden.

7.3.1 Functievereisten

Het is belangrijk om voor elke use case grondig na te denken welke video codec de beste keuze is. De selectie verfijnen kan je reeds eenvoudig doen door naar enkele functievereisten te kijken. Een korte overzichtstabel van enkele kernfunctionaliteiten per besproken video codec is te vinden in figuur 7.1.

7.3.2 Ondersteuning

Een overzicht van de ondersteuning van de besproken video codecs in de gekende internetbrowsers is beschikbaar in figuur 7.2.

7.3.3 Licenties

Op vlak van licenties is er maar één duidelijke winnaar: AV1. Deze codec is open source en royalty-free waardoor de achterliggende code aangepast kan worden en gebruikt kan worden zonder een licentie te hoeven betalen. Een volledig overzicht van de licentievoorwaarden is te vinden op de website van AOMedia[3].

Licenties voor H.264-AVC, H.264-SVC en H.265/HEVC zijn betalend en op aanvraag. Hoewel allen gratis zijn in gebruik onder een bepaalde grens is het aangeraden extra opzoekingswerk te verrichten voor de use case waarin deze codecs zouden gebruikt worden.

7.4 Implementatie

De implementatie van video codecs kan op een gelijkaardige manier als die van afbeeldingsformaten. Er zijn veelal plug-ins beschikbaar om ondersteuning toe te voegen binnen videobewerkingssoftware dat de besproken codecs standaard niet ondersteunen. Voor het implementeren op een webomgeving bestaan er net zoals bij afbeeldingsformaten besproken in deel 6.4 automatische en manuele oplossingen aan de hand van terugval video's of JavaScript decoders. Deze implementaties zijn mede afhankelijk van de gekozen container en audio codec en worden daarom niet verder besproken in deze bachelorproef.

[3] AOMedia: Legal Stuff – http://bit.ly/2I713ua.

8. Kwaliteit beoordelen

In de twee voorgaande hoofdstukken is dieper ingegaan op een aantal afbeeldingsformaten en video codecs. Om de prestatie van achterliggende compressie-algoritmen te beoordelen, wordt meestal gesproken van een gelijkaardige, betere of slechtere kwaliteit met dezelfde bestandsgrootte. Maar hoe kan er nagegaan worden of die kwaliteit wel degelijk gelijkaardig, beter of slechter is? Dit hoofdstuk neemt een kijk in de mogelijke manieren om de kwaliteit van afbeeldingen en dus de achterliggende compressie-algoritmen te evalueren. Zowel objectieve als subjectieve manieren zullen besproken worden. In hoofdstuk 9 zal een subjectief onderzoek gevoerd worden om de kwaliteit van enkele afbeeldingsformaten te testen voor een bepaalde use case.

Kwaliteit van een afbeelding beoordelen is in theorie alleen interessant voor lossy compressie-algoritme aangezien lossless compressie-algoritme per definitie steeds dezelfde kwaliteit als de originele afbeelding dienen te hebben. De prestatie van lossless compressie-algoritme testen kan dus uitsluitend objectief gedaan worden door de bestandsgrootte samen met de encoding en decoding tijd en complexiteit te evalueren.

Hoewel deze bachelorproef niet dieper ingaat op mogelijke manieren voor kwaliteitsevaluatie van video codecs of andere compressie-algoritmen zoals bijvoorbeeld audio codecs, werken de onderzoeken voor deze domeinen vaak op een gelijkaardige manier.

8.1 Objectieve vs subjectieve onderzoeken naar kwaliteit

Er kan onderscheid gemaakt worden tussen twee grote soorten onderzoeken naar kwaliteit: objectieve en subjectieve onderzoeken. In een objectief onderzoek wordt veelal gebruik gemaakt van software toepassingen om een afbeelding te evalueren. Enkele van deze

toepassingen worden besproken in deel 8.2.

Er kan ook gekozen worden om een subjectief onderzoek te voeren. Deze gaan bijvoorbeeld te werk door de afbeeldingen aan deelnemers van een onderzoek te tonen. De deelnemers moeten dan scores toekennen voor één of meerdere kenmerken van de afbeelding. Deze scores worden gebruikt om de kwaliteit van een afbeelding finaal te beoordelen.

Een objectief onderzoek is veelal sneller en makkelijker om uit te voeren aan de hand van de vele goede beschikbare software. Een subjectief onderzoek is ook eenvoudig zelf uit te voeren maar eist meer tijd. Voor het voeren van een subjectief onderzoek kan gebruik gebruik gemaakt worden van de voor deze bachelorproef geschreven afbeeldingsevaluatietool verder besproken in deel 9.3. Er kan ook gekozen worden om de beoordeling van afbeeldingskwaliteit over te laten aan een instelling. Enkele van deze instellingen worden besproken in deel 8.3. Deze kunnen zowel objectieve als subjectieve onderzoeken voeren.

De keuze voor een subjectief onderzoek klinkt wetenschappelijk veelal minder verantwoord als een objectief onderzoek, maar is voor bepaalde use cases de aangeraden manier van werken. Vooral voor lossy compressie-algoritmen is het moeilijk een objectief onderzoek uit te voeren dat een duidelijk beeld geeft van de prestatie van het compressie-algoritme. Objectieve software tools kunnen bij deze compressie-algoritmen vele afwijkingen en artefacten opvangen wat resulteert in een slechte score. De eindgebruiker zal zich hier in werkelijkheid echter misschien niet aan storen. Ook omgekeerd kan een voor software kleine imperfectie de gebruikerservaring juist enorm te min gaan. Dit is de reden date subjectieve onderzoeken eigenlijk als betrouwbaarder beschouwd worden dan objectieve metrieken en wordt dan ook steevast gebruikt bij het vergelijken van nieuwe datacompressiestandaarden.

8.2 Software voor het objectief evalueren van afbeeldingen

Er bestaat enerzijds software om de kwaliteit van afbeeldingen te beoordelen op bijvoorbeeld scherpte en contrast. Bij dit soort software worden afbeeldingen dus individueel beoordeeld en niet vergeleken met een andere afbeelding. Dit soort software is vooral interessant voor het testen van de effectieve camera (hardware) en niet zo zeer voor het testen van compressie-algoritmen. Dit is het soort software dat makers van lenzen en camera's gebruiken om nieuwe hardware te testen. Een gekend bedrijf dat deze software samen met geautomatiseerde machines voor testen produceert is 'imatest'[1]. Dit soort software is echter niet zo interessant voor het evalueren van compressie-algoritmen en wordt daarom niet verder toegelicht in deze bachelorproef.

Anderzijds bestaat er software die twee afbeeldingen met elkaar vergelijkt. Dit is interessant voor het beoordelen van compressie-algoritme door een lossless gecomprimeerde variant te vergelijken met het te testen lossy compressie-algoritme. De software kan hiervoor gebruik maken van enkele variabelen die eenvoudig te berekenen zijn tussen twee afbeeldingen. Enkele van deze variabelen worden hieronder besproken. Aangezien dit

[1] ImaTest – http://bit.ly/2VXWXsY.

wiskundige berekeningen zijn kunnen deze berekend worden door software van bedrijven als MathWorks[2]. Er is ook veel, al dan niet betalende, software ter beschikking die het mogelijk maakt twee afbeeldingen te selecteren waarna de software alle berekeningen geautomatiseerd doet.

8.2.1 RMSE

De Root Mean Square Error (de wortel van de gemiddelde kwadratische afwijking) stelt de standaarddeviatie van het verschil tussen de gemeten waarde en de voorspelde waarde voor. De voorspelde waarde is in dit geval één pixel van de lossless variant en de gemeten waarde de overeenkomende pixel van het te evalueren afbeeldingsformaat. Hoe lager de rmse van een pixel hoe meer deze op elkaar lijken. Een bepaalde uitkomst voor een situatie zeer goed zijn en voor een andere zeer slecht doordat er geen vaste grenzen zijn en de uitkomst zonder context dus niets zegt. Dit wordt voor alle pixels herhaald.

Deze maatstaf werkt zeer goed vanuit een theoretisch standpunt, maar is niet representatief naar de perceptie van het menselijk oog. Het menselijke oog neemt de afbeelding echter als één geheel waar en niet pixel per pixel.

8.2.2 SSIM

De structural similarity index probeert het probleem met rmse op te lossen door een groepering van pixels te vergelijken in plaats van elke pixel individueel. Het vergelijkt zowel de belichtingssterkte, het contrast als de structuur van de groep pixels. Dit leunt meer aan naar hoe het menselijke oog twee afbeeldingen zou vergelijken.

In vergelijking met rmse, waar de uitkomst niet tussen twee vaste grenzen ligt, is de SSIM altijd uitgedrukt tussen -1 en 1. Dit maakt het makkelijker om te bepalen of het resultaat goed of slecht is zonder context. Een resultaat van SSIM dat bij -1 aanleunt duid op een groot visueel verschil, een resultaat dat naar 1 aanleunt duid op een bijna identieke match.

8.2.3 PSNR

Peak signal-to-noise ratio is een techniek dat, net zoals de vorige beschreven technieken, via een wiskundige berekening twee afbeeldingen vergelijkt om na te gaan hoeveel de twee afbeeldingen op elkaar lijken.

8.2.4 Het probleem met deze formules

Hoewel SSIM beter aanleunt naar de perceptie van een afbeelding zoals het menselijke oog, is er geen enkele VDP dat het menselijke oog perfect representeert. Dit is ook de

[2]MathWorks – http://bit.ly/2Widbm8.

reden dat subjectieve testen nog steeds zo belangrijk zijn binnen afbeeldingscompressie en andere datacompressie domeinen.

8.3 Instellingen voor kwaliteitsevaluatie

Er kan beroep gedaan worden op tal van instellingen die zich professioneel bezig houden met het evalueren van afbeeldingskwaliteit. Door het werken met een erkende derde partij vermijd je, althans in theorie, dat kwaliteitsbeoordelingen doelbewust beter zijn om je product ten voordele te komen.

8.3.1 DxOMark

DxOMark is een bedrijf dat voornamelijk gekend is voor het beoordelen van de camera's in smartphones. De duurste toestellen van producenten als Samsung en OnePlus gebruiken deze DxOMark score vaak als verkooppunt om te aan te tonen hoe goed de camera's van hun toestellen zijn. DxOMark evalueert zowel afbeeldingen als video's.

8.3.2 Maximum Compression

Maximum Compression is een instelling die voor verschillende bestandsformaten, waaronder het afbeeldingsformaat JPEG, nagaat wat de maximaal te bereiken compressieratio is. Het maakt hier uitsluitend gebruik van lossless compressie-algoritme. De gebruikte data en software is publiek beschikbaar wat Maximum Compression één van de meest objectieve instellingen op de markt maakt.

8.3.3 Het probleem met deze instellingen

Hoewel deze instellingen wel de indruk geven dat ze compleet onafhankelijk zijn dient dit met een korrel zout genomen te worden. Zo bied DxOMark betalende diensten om producenten te helpen een zo goed mogelijke DxOMark score te halen. Hoewel de werking van hun score systeem uitgebreid uitgelegd is op hun website[3] is de effectieve code van de achterliggende algoritmes niet raadpleegbaar.

[3] DxOMark – http://bit.ly/2YVnQQw.

9. Onderzoek

Het kiezen voor een bepaald compressie-algoritme is zeer use case gebonden. Zeker binnen videocompressie en afbeeldingscompressie is dit het geval. Ligt de focus op een zo klein mogelijke bestandsgrootte of juist op het behouden van zo veel mogelijk kwaliteit binnen een bepaalde omgeving? Wordt er voor een lossless of lossy compressie-algoritme gekozen?

Zoals besproken in hoofdstuk 8 zijn er zowel objectieve als subjectieve onderzoeken die kunnen gevoerd worden om een gepast compressie-algoritme te bepalen. In dit hoofdstuk wordt een subjectief onderzoek uitgevoerd voor het bepalen van een geschikt afbeeldingsformaat binnen een bepaalde use case. De gebruikte afbeeldingsevaluatietool is ontwikkeld voor deze bachelorproef en is open source en gratis in gebruik. De code is te vinden op de GitHub repository van deze bachelorproef[1]. Dit maakt het heel eenvoudig dit onderzoek te reproduceren of een gelijkaardig onderzoek uit te voeren met andere afbeeldingsformaten en/of voor een andere use case.

De gebruikte afbeeldingen zijn op aanvraag beschikbaar: info@lennertbontinck.com.

9.1 Waarom een subjectief onderzoek?

Zoals besproken in hoofdstuk 8 is een subjectief onderzoek aangeraden binnen tal van use cases. Zeker binnen visuele datacompressie kan dit de aangeraden manier van werken zijn, aangezien de kwaliteit van de afbeeldingen een grote invloed heeft op de gebruikerservaring.

[1] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

Een objectief onderzoek, dat bijvoorbeeld werkt aan de hand van het vergelijken van een afbeelding voor en na compressie, kan een vals positief of negatief beeld geven over een bepaald afbeeldingsformaat.

9.2 Use case

De RAW bestanden gebruikt binnen dit onderzoek zijn aangeleverd door Mayté Bogaert van MaytéB fotografie. Aangezien zij ook aan het plannen is een online portfolio te bouwen, vroeg ze zich af in welk afbeeldingsformaat en met welke compressie instellingen ze de afbeelding het best online zet. Deze vraag is dan ook de use case van dit onderzoek.

Voor deze use case speelt de gebruikerservaring een zeer belangrijke rol. Als fotografe zijn afbeeldingen je product en moeten potentiële klanten deze dus positief ervaren wanneer ze op je portfolio terecht komen.

Langs de andere kant gaat het om een online website en moet er dus rekening gehouden worden met hosting kosten, wachttijden en bandbreedte gebruik. Als de website te lang duurt om te laden of alle mobiele data van een potentiële klant opgebruikt, is dat geen goede reclame.

Er wordt dus een afbeeldingsformaat gezocht met (zeer) goede beoordelingen uit het onderzoek dat de kleinste bestandsgrootte heeft.

9.3 Afbeeldingsevaluatietool

De gebruikte afbeeldingsevaluatietool voor het voeren van dit onderzoek is een voor deze bachelorproef ontwikkelde afbeeldingsevaluatietool. De code is te vinden op de GitHub repository van deze bachelorproef[2]. De afbeeldingsevaluatietool is ook raadpleegbaar via de website van Lennert Bontinck[3].

Deze tool is geschreven in PHP met een achterliggende SQL databank. Er is een grafische interface voorzien die gebruik maakt van HTML, CSS, JavaScript, JQuery, Drift en Bootstrap. Dit maakt het mogelijk de tool eenvoudig lokaal te runnen door het gebruik van webserver omgevingen als XAMPP of online te plaatsen op de meeste hosting platformen.

De afbeeldingsevaluatietool maakt gebruik van een 50% grijze achtergrond doorheen de ondervraging. Dit is de aangeraden kleur om geen impact te hebben op de perceptie van de kleuren binnen de afbeelding. Het inzoomen gebeurt aan de hand van pure JavaScript waardoor geen manipulaties aan de afbeelding gebeuren en deze worden weergegeven zoals ze opgeslagen zijn.

[2]Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.
[3]Bachelorproef onderzoek: afbeeldingskwaliteit – http://bit.ly/2JIIBeE.

9.3.1 Opzetten van de afbeeldingsevaluatietool

Een identiek onderzoek maar met andere afbeeldingen kan gevoerd worden zonder code aanpassingen te moeten uitvoeren, wat de reproduceerbaarheid van dit onderzoek hoog houdt. De gebruikte afbeeldingen kunnen ook op aanvraag geleverd worden, neem hiervoor contact via info@lennertbontinck.com.

Om de afbeeldingsevaluatietool op te zetten, clone je de GitHub repository. Alle nodige bestanden voor deze afbeeldingsevaluatietool zijn te vinden onder de map 'evaluatietool'. Plaats alle bronbestanden uit deze map op de webserver en voorzie een database via localhost genaamd 'bachelorproef'. De gebruiker root met wachtwoord root moet toegang hebben tot deze database.

De te evalueren afbeeldingen dienen voorzien te worden onder de map 'evaluatiereeks' en/of 'testreeks', te vinden in de map 'evaluatie_afbeeldingen'.

Surf naar '/setup.php' en wacht tot er 'done' op het scherm verschijnt. De afbeeldingen zijn nu in de databank opgeslagen en het onderzoek is nu klaar om van start te gaan. Surf hiervoor naar '/index.php', je krijgt het welkomstscherm te zien zoals op figuur 12.6 weergegeven.

Aangezien bij dit onderzoek twee nog niet volledig ondersteunde afbeeldingsformaten worden getest, JPEG2000 en WebP, is een Apple computer met de Safari (voor JPEG2000) en Google Chrome internetbrowsers (voor WebP) nodig. Het onderzoek begint in de Google Chrome internetbrowser en halverwege het onderzoek krijgt de deelnemer een scherm te zien dat deze moet overschakelen naar de Safari internetbrowser.

Voor het opzetten van de afbeeldingsevaluatietool met extra functionaliteit, zoals meer ondervraagde kenmerken of andere afbeeldingsformaten, zijn aanpassingen van de code vereist. De code is in het volgende deel kort toegelicht en beschikt over verklarende functienamen en commentaar in de code zelf.

9.3.1.1 Afbeeldingsevaluatietool instellingen en uitbreidingen

9.3.1.1.1 Db/db_actions.php

De database instellingen zijn bovenaan het bestand 'db/db_actions.php' voorzien. De standaard waarden zijn in de code snippet hieronder weergegeven.

```
1 $servername = "localhost";
2 $username = "root";
3 $password = "root";
4 $dbname = "bachelorproef";
```

In de functie 'create_tables()' worden alle tabellen voorzien, deze wordt aangeroepen vanuit 'setup.php'. De volgende tabellen en velden worden reeds bijgehouden. In deze functie kunnen extra velden toegevoegd worden.

- images

  - image_id → auto increment integer (primaire sleutel)
  - filename → string
  - extension → string
  - path → string
  - practice_data → boolean
  - chrome_not_safari → boolean
  - filesize → integer
- participants
  - participant_id → auto increment integer (primaire sleutel)
  - gender → string
  - age → integer
  - expertise → boolean
  - colorblind → boolean
  - bad_vision → boolean
- ratings
  - participant_id → auto increment integer (samengestelde primaire sleutel)
  - image_id → auto increment integer (samengestelde primaire sleutel)
  - sharpness → integer
  - color_contrast → integer
  - general → integer

Ook de overige database bewerkingen zijn te vinden in dit bestand. Dit zijn onder andere de functies voor het opslaan van de gegevens van de deelnemer. Alle functies en variabelen hebben een verklarende benaming.

9.3.1.1.2 Js/*

In de folder 'js' is Drift voorzien onder het bestand 'drift.min.js'. In het bestand 'scripts.js' wordt een Drift instantie aangemaakt met de nodige instellingen. De standaard instellingen voor drift zijn hieronder zichtbaar. De zoom factor aanpassen kan door de waarde van 'zoomFactor' op de vijfde regel aan te passen.

```
1  if ($('.img_evaluation').length) {
2    new Drift(document.querySelector('.img_evaluation'), {
3      paneContainer: document.querySelector('.img_evaluation_zoomed')
         ,
4      inlinePane: 900,
5      zoomFactor: 5,
6      inlineOffsetY: -85,
7      containInline: true,
8      hoverBoundingBox: true,
9      touchBoundingBox: true
10   });
11 }
```

9.3.1.1.3 Layout/*

In de folder 'layout' is de header en footer dat gedeeld wordt overheen de pagina's voorzien. De titel van de webpagina aanpassen of extra tags in de header voorzien, kan in het bestand 'header.php'. De copyright in de footer veranderen of extra tags toevoegen op het einde van de webpagina kan in het bestand 'footer.php'.

9.3.1.1.4 Index.php

Het bestand 'index.php' verzorgt het welkomstscherm weergegeven in figuur 12.6. In dit bestand kan u de welkomsttekst wijzigen. De knop onderaan de pagina bevat een verwijzing naar 'introductie.php'.

9.3.1.1.5 Introductie.php

Het bestand 'introductie.php' verzorgt de webpagina met de introductievideo weergegeven in figuur 12.7. Het filmpje dat ingeladen wordt, kan aangepast worden door een andere YouTube URL in te geven onder het src attribuut van regel twee of het pad naar een lokaal voorziene video.

```
1 <div class="embed-responsive embed-responsive-16by9">
2   <iframe class="embed-responsive-item" src="https://
      www.youtube.com/embed/tqx-hmz04kQ" allowfullscreen></iframe>
3 </div>
```

De knop onderaan deze webpagina verwijst naar 'onderzoek.php'

9.3.1.1.6 Onderzoek.php

Het bestand 'onderzoek.php' verzorgt de webpagina waar de deelnemer zijn informatie ingeeft en de afbeeldingen kan beoordelen weergegeven in figuur 12.8 en 12.9.

In dit bestand worden volgende functies opgenomen met een verklarende naam:

- show_info_about_you()
- save_info_about_you_and_show_chrome()
- show_start_chrome_sequence($participant_id)
- show_next_iterative_photo_rating_chrome()
- show_photo_rating($image_id, $path)
- save_post_rating_image()
- show_start_safari_sequence()
- show_next_iterative_photo_rating_safari()
- show_thanks_screen()

9.3.1.1.7 Setup.php

Het bestand 'setup.php' maakt de afbeeldingsevaluatietool klaar voor gebruik. Hier worden de tabellen eerst verwijderd indien ze bestaan, waarna ze terug aangemaakt worden. Vervolgens wordt elke afbeelding in de tabel 'images' gezet door de code hieronder voorzien.

```
1 foreach (glob('evaluatie_afbeeldingen/testreeks/*.*') as $path) {
2   $extension = pathinfo($path, PATHINFO_EXTENSION);
3   $filename = pathinfo($path, PATHINFO_FILENAME);
4   $chrome_not_safari = ($extension == "jpg" || $extension == "webp"
        ) ? 1 : 0;
5   $practice_data = 1;
6   $filesize = filesize($path);
7   create_image_record($filename, $extension, $path, $practice_data,
        $chrome_not_safari, $filesize);
8 }
```

Op regel vier wordt bepaald of de afbeelding in Google Chrome moet weergegeven worden. Dit is het geval wanneer de bestandsextensie die voor JPEG of WebP is. Een gelijkaardige lus wordt gebruikt voor de evaluatiereeks.

9.3.1.1.8 Export.php

Het bestand 'export.php' verzorgt de webpagina waar de organisator de geëxporteerde gegevens kan downloaden, weergegeven in figuur 12.10.

Voor het maken van de CSV bestanden wordt de volgende functie gebruikt.

```
1  function make_csv_files() {
2    //participants
3    $file = fopen("temp/participants.csv", "w");
4    fputcsv($file, array('participant_id', 'gender', 'age', '
         expertise', 'colorblind', 'bad_vision'));
5    $records = get_all_participants();
6    while ($row = mysqli_fetch_assoc($records)) {
7      fputcsv($file, $row);
8    }
9    fclose($file);
10
11   //images
12   ...
13
14   //ratings
15   ...
16 }
```

De volledige SQL tabel met de bijhorende velden wordt dus gedumpt naar het CSV bestand met als heading de veldnamen. De code voor images en ratings is weggelaten omdat deze identiek is aan participants, maar met andere variabelen namen.

9.3.2 Verloop van de afbeeldingsevaluatietool

De afbeeldingsevaluatietool is alomvattend voor het verzamelen van de gegevens omtrent dit onderzoek. Dit wil zeggen dat in de afbeeldingsevaluatietool zelf beschreven wordt hoe deze dient gebruikt te worden en welke gegevens van de deelnemer bewaard worden. De afbeeldingsevaluatietool slaat ook alle input van de gebruiker automatisch op in de achterliggende SQL databank waardoor er geen manueel werk meer nodig is.

Enkele screenshots van de afbeeldingsevaluatietool zijn raadpleegbaar in deel 12.3.

9.3.3 Verloop van de afbeeldingsevaluatietool: Introductie

De deelnemer van het onderzoek wordt begroet met een scherm dat meedeelt waarover het onderzoek gaat, welke gegevens van de deelnemer gevraagd en bewaard zullen blijven en de geschatte duur van het onderzoek, zijnde een half uur tot drie kwartier. Een screenshot is voorzien in figuur 12.6.

Als de deelnemer akkoord gaat met de gegevensverwerking wordt deze doorverwezen naar een volgende pagina met een introductievideo[4] van 4 minuten op. Hierin worden de geëvalueerde onderdelen uitgelegd aan de hand van enkele extreme voorbeelden. Ook de use case en gegevensverwerking worden nogmaals uitgelegd. Een screenshot is voorzien in figuur 12.7.

9.3.4 Verloop van de afbeeldingsevaluatietool: Informatie over deelnemer

Er worden enkele gegevens van de deelnemer gevraagd en opgeslagen zijnde:

- Geslacht (man, vrouw, overige)
- Leeftijd
- Of de deelnemer expertise heeft binnen het onderzoeksdomein
- Of de deelnemer kleurenblind is
- Of de deelnemer slechtziend is

Het geslacht wordt bijgehouden om na te gaan of het mikpunt van 50/50 verhouding tussen man en vrouw bereikt is.

Een deelnemer wordt beschouwd als expertise hebbende wanneer deze een gegronde kennis heeft van afbeeldingscompressie en afbeeldingsformaten, het verschil kent tussen lossless en lossy compressie-algoritmen alsook artifacten zoals blokartifcaten kan herkennen in afbeeldingen.

Kleurenblindheid is een veelvoorkomend probleem bij mannen. Gemiddeld één op twaalf mannen heeft kleurenblindheid terwijl bij vrouwen dat gemiddelde onder de één op 250 ligt volgens „Color Blindness” (Porcella, 2008). Deze variabele wordt samen met

[4] Introductievideo bachelorproef onderzoek afbeeldingskwaliteit – http://bit.ly/2HqKN8k.

slechtziendheid bijgehouden om na te gaan of dit een impact heeft op de perceptie van de afbeeldingen.

Een screenshot van dit scherm is voorzien in figuur 12.8.

9.3.5 Verloop van de afbeeldingsevaluatietool: beoordeling

Uiteindelijk krijgt de deelnemer voor elke afbeelding een evaluatiescherm te zien. Een screenshot van dit scherm is voorzien in figuur 12.9.

De volgorde van de afbeeldingen is voor elke deelnemer random bepaald. Indien er een testreeks voorzien is, worden eerst de afbeeldingen uit deze testreeks aan de deelnemer getoond.

Op dit evaluatiescherm is aan de hand van Drift de mogelijkheid voorzien op de rechterhelft van het scherm een ingezoomde variant van de afbeelding op de linkerhelft te bekijken. Deze zoom is 5x en wordt bekomen door een pure JavaScript oplossing. Dit wilt zeggen dat het originele bestand getoond wordt en er dus geen kwaliteitsverlies mogelijk is door deze plug-in.

De deelnemer dient de afbeelding te beoordelen op de volgende drie kenmerken met een getal van één tot vijf:

- Scherpte
- Kleuren en contrast
- Algemene indruk

9.3.5.1 Kenmerk: scherpte

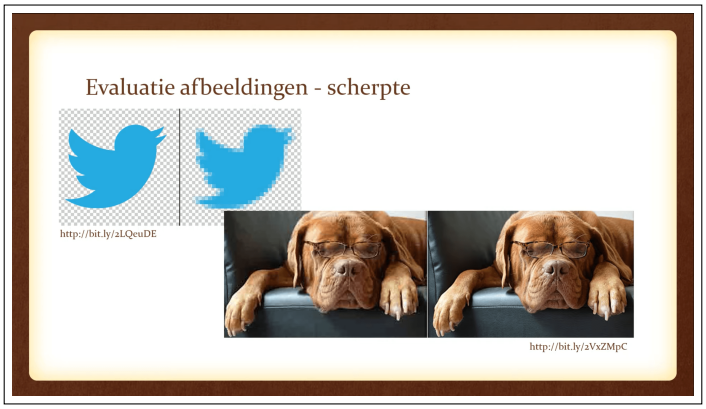
Een afbeelding wordt beschouwd als scherp wanneer de lijnen vloeiend worden weergegeven en er veel detail aanwezig is. Een fragment uit de introductievideo van de afbeeldingsevaluatietool waarin het kenmerk scherpte wordt uitgelegd is terug te vinden in figuur 9.1.

Hier wordt scherpte uitgelegd aan de hand van de randen in het Twitter logo links bovenaan waarbij de linkse variant zeer goed zou scoren (dus richting de vijf) en de rechter variant eerder slecht (richting de één).

Bij de hond is de linkerfoto de slechtere variant doordat veel detail rond de snuit verloren is gegaan. De hond heeft ook last van een slecht contrast op zijn poot waardoor bij de linker variant het verschil tussen de nagel en de poot soms slecht zichtbaar is.

9.3.5.2 Kenmerk: kleuren en contrast

Zoals hierboven besproken is de poot van de hond in figuur 9.1 een goed voorbeeld van een slecht contrast. Op figuur 9.2 is een fragment uit de introductievideo van de



Figuur 9.1: Een fragment uit de introductievideo van de afbeeldingsevaluatietool waarin het kenmerk scherpte (9.3.5.1) uitgelegd wordt. (Bontinck, 2019c)



Figuur 9.2: Een fragment uit de introductievideo van de afbeeldingsevaluatietool waarin het kenmerk kleuren en contrast (9.3.5.1) uitgelegd wordt. (Bontinck, 2019c)

afbeeldingsevaluatietool, waarin het kenmerk kleur wordt uitgelegd, zichtbaar.

Hier worden vooral veelvoorkomende soorten artefacten met betrekking op kleur weergeven. Bijvoorbeeld de artefacten zichtbaar bij de ster door clustering in bijvoorbeeld het JPEG afbeeldingsformaat.

9.3.5.3 Kenmerk: algemene indruk

Tot slot wordt ook de algemene indruk van de deelnemer gevraagd omtrent deze afbeelding. Hier is het belangrijk dat de score gegeven wordt vanuit het standpunt dat de deelnemer de foto is tegengekomen op het portfolio van een fotografe zoals meegeven in de use case (deel 9.2).

9.3.6 Exporteren van de verzamelde gegevens

Na het uitvoeren van het onderzoek kunnen de gegevens eenvoudig geëxporteerd worden naar drie verschillende CSV bestanden:

- images.csv: een dump van de SQL tabel 'images' met als header de veldnamen.

- participants.csv: een dump van de SQL tabel 'participants' met als header de veldnamen.
- ratings.csv: een dump van de SQL tabel 'ratings' met als header de veldnamen.

Om deze CSV bestanden te genereren moet naar '/export.php' gesurft worden. Het export scherm, zichtbaar op figuur 12.10, wordt weergegeven. De resultaten van dit onderzoek zijn beschikbaar op de GitHub repository van deze bachelorproef[5] onder de map 'resultaten'.

Het kan interessant zijn om de verschillende CSV bestanden samen te voegen naar één CSV bestand. Dit kan eenvoudig in Python door gebruik te maken van Pandas. Het commando ziet er als volgt uit:

```
1 import pandas as pd
2 images=pd.read_csv("/Users/lennertbontinck/Documents/github/
    bachelorproef-compressie/resultaten/csv/images.csv")
3 participants=pd.read_csv("/Users/lennertbontinck/Documents/github/
    bachelorproef-compressie/resultaten/csv/participants.csv")
4 ratings=pd.read_csv("/Users/lennertbontinck/Documents/github/
    bachelorproef-compressie/resultaten/csv/ratings.csv")
5 merged=images.merge(ratings,on="image_id")
6 merged=merged.merge(participants,on="participant_id")
7 merged.to_csv("/Users/lennertbontinck/Documents/github/
    bachelorproef-compressie/resultaten/csv/merged.csv", index=False
    )
```

De notebook voor het samenvoegen van de CSV bestanden is terug te vinden onder 'resultaten/python/merge/merge-csv.ipynb'.

Onder 'resultaten/python/*' zijn alle gebuikte notebooks voor het maken van de grafieken uit deel 9.5 opgenomen.

9.4 Uitvoering

Het onderzoek is gevoerd over twee maanden tijd waarbij in totaal 43 mensen hebben deelgenomen. Er wordt meer over de deelnemers gesproken in deel 9.4.2. Het verloop van het onderzoek is te lezen in deel 9.3.2.

Voor de effectieve uitvoering van het onderzoek is met éénzelfde laptop gewerkt: een Apple MacBook Pro Mid 2014 15 inch met retina display. Dit toestel is gekozen voor zijn hoge resolutie scherm (2880x1800 pixels) en, een goede representatie van het sRGB color gamut. Bij elke deelnemer werd er voor gezorgd dat de volgende zaken overeen kwamen:

- Het testtoestel heeft 100% batterij en is aangesloten op het netsnoer.
- Het testtoestel zijn helderheid staat op 100% met alle vormen van kleuraanpassingen uitgeschakeld (night mode,...).
- Het testtoestel zijn toetsenbordverlichting staat uit zodat dit niet voor reflectie in het scherm kan zorgen.

[5] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.

- Het testtoestel zijn scherm is volledig stof- en vlek vrij gemaakt door het gebruik van een microvezeldoek.
- De gebruikte browsers staan klaar met leeggemaakte cache en in incognito modus zonder dat enige vorm van databesparing aan staat.
- De deelnemer zit in een door led lamp verlichte ruimte waarbij geen directe inval van zonlicht op het scherm is.

Op deze manier zijn zoveel mogelijk randvariabelen geëlimineerd bij het uitvoeren van het onderzoek.

De auteur van deze bachelorproef was ten alle tijden aanwezig tijdens het onderzoek mochten er vragen zijn of er zich problemen voor doen.

9.4.1 Geëvalueerde afbeeldingen

Voor dit onderzoek zijn 15 afbeeldingen geëvalueerd in de volgende vier afbeeldingsformaten:

- PNG
- JPEG
- JPEG2000
- WebP

Er is gebruik gemaakt van een selectie RAW afbeeldingen aangeleverd door Mayté Bogaert van MaytéB fotografie met volgende kenmerken:

- Getrokken met een 'Nikon D750' toestel en 'TAMRON SP AF 28-75mm F2.8 XR Di LD Aspherical IF Macro A09NII' lens.
- Portretfoto's waarbij het onderwerp in focus is en de achtergrond wazig door het gebruik van een kleine diafragma waarde (grote opening).
- Zowel afbeeldingen die met en zonder flash genomen zijn.
- Zowel afbeeldingen die in een zeer goed verlichte omgeving als donkere omgeving genomen zijn.
- Zowel afbeeldingen met een groot contrast als een klein contrast.

De afbeeldingen zijn allemaal geëxporteerd vanuit Adobe Photoshop. Hiervoor is steeds het RAW bestand als startbestand gebruikt waarbij de canvasgrootte omgezet is naar 1000 x 1498 pixels (portret) of 1498 x 1000 pixels (landschap). Het exporteren naar PNG, JPEG, en JPEG2000 zit standaard ingebouwd in deze versie van Adobe Photoshop (CC 20.0.0). Voor het exporteren naar WebP is gebruik gemaakt van de gratis open source plug-in 'AdobeWebM' door fnordware[6].

De exacte instellingen gebruikt voor het opslaan van de afbeeldingen zijn terug te vinden in figuur 12.11.

[6] AdobeWebM – http://bit.ly/2Mmdho2.

9.4.2 Deelnemers

In totaal hebben 43 mensen deelgenomen aan het onderzoek waarvan 15 vrouwen en 28 mannen. Zes personen hebben opgegeven dat ze expertise binnen het domein hebben, drie mensen dat ze slechtziend zijn op het moment dat ze het onderzoek afleggen en twee personen hebben aangegeven dat ze kleurenblind zijn. Dit waren beide mannen. Deze gegevens komen overeen met de verwachting uit deel 9.3.4. Hoewel er geen perfecte 50/50 verhouding is tussen mannen en vrouwen is deze verhouding aanvaardbaar.

Alle informatie die in dit deel benoemd is, is te bepalen via de Python scripts te vinden onder 'resultaten/python/statistieken/participants/*'. De gebruikte CSV is te vinden onder 'resultaten/csv/participants.csv'.

9.5 Resultaten

De resultaten zijn na afloop van het onderzoek geëxporteerd naar drie CSV bestanden en samengevoegd naar één CSV bestand zoals uitgelegd in deel 9.3.6. Alle informatie die in dit deel toegelicht zal worden, is te bepalen via de Python scripts te vinden onder 'resultaten/python/statistieken/*'. De gebruikte CSV bestanden zjin te vinden onder 'resultaten/csv/*'.

9.6 Resultaten lossless afbeeldingsformaten

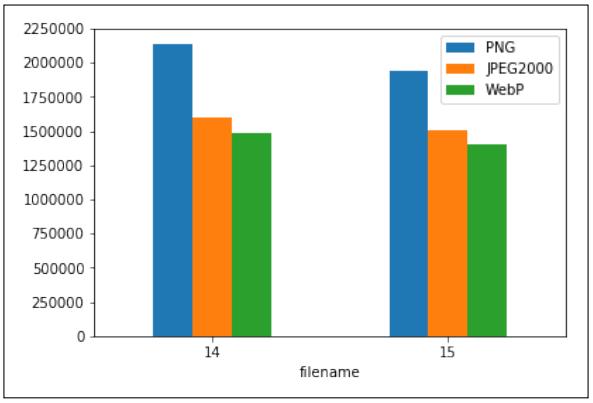
Zoals verwacht scoort PNG en de lossless variant van WebP en JPEG2000 zeer goed. In figuur 12.12 is duidelijk te zien dat PNG voor het kenmerk 'algemene indruk' steeds een mediaan heeft van vijf en het gemiddelde ook dicht bij vijf ligt, de maximumscore. Dit is ook voor kenmerk 'scherpte' en 'kleuren en contrast' het geval. Een bijna identiek resultaat is zichtbaar voor de lossless variant van WebP (figuur 12.13) en de lossless variant van JPEG2000 (figuur 12.14).

Wanneer echter naar de bestandsgrootte wordt gekeken is wel een groot verschil te zien tussen de verschillende afbeeldingsformaten. Dit wordt weergegeven in figuur 9.3. WebP scoort in beide gevallen iets beter dan JPEG2000 welke, met behoorlijke marge, beter presteert dan PNG.

9.7 Resultaten lossy afbeeldingsformaten

Afbeelding één tot en met dertien zijn door JPEG, JPEG2000 en WebP lossy gecomprimeerd. De gebruikte instellingen zijn terug te vinden in figuur 12.11.

Wat direct opvalt, is dat de bestandsgroottes immens verschillen desondanks percentuele instellingen voor kwaliteitsbehoud identiek waren. Dit is weergegeven in figuur 12.15.



Figuur 9.3: De bestandsgrootte in bits voor de afbeeldingen die lossless gecomprimeerd zijn door zowel PNG, JPEG2000 en WebP.

Dit maakt het moeilijk om de resultaten voor de lossy afbeeldingsformaten te vergelijken. Het verschil in compressieratio verschilt namelijk te sterk om per afbeelding de afbeeldingsformaten te vergelijken.

Er kan echter wel een interessant overzicht gemaakt worden door een verhouding tussen de bestandsgrootte en toegekende scores op te stellen. Deze verhoudingen zijn weergegeven in figuur 12.16, 12.17 en 12.18. Als we deze figuren vergelijken met de figuren van de bestandsgrootte, en de verhoudingen vergelijken daar waar de bestandsgroottes gelijkaardig zijn, is WebP de duidelijke winnaar.

9.8 Besluit

Hoewel de gegevens niet ideaal zijn door de onverwachts grote variatie in compressieratio tussen de afbeeldingsformaten, is er wel een duidelijke trend met de beschikbare gegevens. WebP behaalt voor eenzelfde bestandsgrootte een aanzienlijk grotere score dan JPEG2000 wat op zijn beurt beter presteert dan JPEG. Ook valt op dat de scores voor de drie kenmerken bij WebP zeer gelijkaardig zijn terwijl bij JPEG en JPEG2000 de kleur gemiddeld gezien bij bijna alle afbeeldingen het slechtst scoort, gevolgd door de scherpte. De gegeven algemene score ligt gemiddeld gezien ook hoger dan zowel de score op scherpte als op kleuren en contrast voor de drie afbeeldingsformaten.

Zoals verwacht krijgen de lossless afbeeldingsformaten zeer gelijkaardige scores. Dit is dan ook het kenmerk van lossless, dat ze een afbeelding produceren zonder kwaliteitsverlies en dus identiek zijn ongeacht het afbeeldingsformaat. Hier kan een winnaar dus puur objectief gekozen worden op basis van bestandsgrootte en ook daar wint WebP. JPEG2000 is bij de geteste lossless gecomprimeerde afbeeldingen tussen de vijf en tien percent groter dan WebP. PNG is aanzienlijk groter dan lossless JPEG2000 en WebP met een bestandsgrootte van ongeveer 30% meer dan de andere afbeeldingsformaten.

Voor deze use case kan dus besloten worden gebruik te maken van een manuele implementatie waarbij het picture element gebruikt wordt dat besproken is in deel 6.4.1. De volgorde voor ondersteuning is dan WebP, JPEG2000 en tot slot JPEG2000. Deze afbeeldingen

zullen manueel voorzien worden aangezien het gebruik van een JavaScript decoder een dubbele lossy datacompressie zou veroorzaken wat ongewenste kwaliteitsgevolgen kan hebben. Wanneer een bezoeker op een afbeelding klikt zal een lossless WebP afbeelding geladen en vergroot getoond worden. Indien deze niet ondersteund wordt, zal gebruik gemaakt worden van de JavaScript decoder om hem om te zetten naar PNG zoals besproken in deel 6.4.3. Dit beperkt de overhead voor het creëren van de afbeelding in alle afbeeldingsformaten enigszins. Het kiezen voor de lossy WebP afbeeldingen om te zetten naar een PNG afbeelding is ook een geldige keuze aangezien hier naar een lossless afbeeldingsformaat gegaan wordt waardoor er geen kwaliteit meer verloren gaat.

Op Google I/O 2019 is echter ook een nieuwe feature van Google Chrome aangekondigd die eenvoudig geïmplementeerd kan worden. Door het voorzien van de data tag loading=”lazy” zal Google Chrome automatisch de afbeeldingen lazy loaden waardoor de gebruiker de webpagina reeds kan zien zonder dat de afbeeldingen ingeladen zijn. Dit bevordert de response time enorm.

10. Huidige en toekomstige uitdagingen

In dit korte hoofdstuk zullen enkele huidige uitdagingen van datacompressie toegelicht worden. Hiermee zal worden aangetoond dat datacompressie een enorm groot begrip is dat veel verder gaat dan alleen afbeeldingscompressie en videocompressie.

10.1 DNA compressie

Het menselijke DNA kan digitaal voorgesteld worden door een lange lijst van 5 verschillende karakters, gekend als basen. Deze digitale voorstelling bestaat uit meer dan 3 miljard van deze basen (UGent, 2011). DNA compressie bestaat eruit deze reeks van basen zo efficiënt mogelijk op te slaan, zodat performante bewerkingen mogelijk zijn met een zo klein mogelijke bestandsgrootte.

Hoewel met de huidige technologie en kennis reeds tal van informatie uit DNA gehaald kan worden, zal verdere uitwerkingen van DNA compressie technieken het opslaan van een gigantische hoeveelheid DNA enkel bevorderen. Dit kan op zijn beurt voor tal van andere doorbraken zorgen. Zo is een AI, die voorzien is van voldoende trainingsdata, mogelijks in staat relaties te vinden tussen DNA en bepaalde aandoeningen die huidige wetenschappers momenteel niet vinden.

Vakexpert en co-promotor van deze bachelorproef, Tom Paridaens, houdt zich op professionele basis bezig met DNA compressie.

10.2 AI en compressie

Zoals besproken in deel 10.1 is het door verdere databesparing mogelijk meer data te verzamelen en op te slaan. Deze grotere hoeveelheid aan data kan tot een stijging van beschikbare trainingsdata voor AI's zorgen. Dit is ten voordele van de werking van de AI.

Maar ook de bestanden die een AI genereert, kunnen verdere datacompressie gebruiken. Zo is in Google I/O 2019 besproken dat het model voor natuurlijke stemherkenning verkleind is van ongeveer honderd gigabyte naar nog geen één gigabyte. Dit maakt het mogelijk de modellen te voorzien op toestellen zelf in plaats van in de cloud. Dit wil niet alleen zeggen dat de diensten offline gebruikt kunnen worden, maar ook dat de latency aanzienlijk kleiner wordt.

Meer details over hoe deze verkleining bereikt is, is op het moment van schrijven nog niet beschikbaar, maar er zal vermoedelijk gebruik gemaakt zijn van het selectief verwijderen van data aan de hand van een zeer complex lossy compressie-algoritme. Achterliggend zullen ook tal van lossless compressie-algoritmen gebruikt zijn om verdere datacompressie te bereiken.

11. Conclusie

Datacompressie, en compressie in het algemeen is niets nieuw. Integendeel, het is één van de oudste concepten binnen IT en tot op heden van fundamenteel belang voor zowat alle IT-toepassingen. Een basiskennis over datacompressie en de belangrijkste afbeeldingsformaten en video codec is dan ook geen luxe binnen de IT-wereld. Veel vakgerelateerde opleidingen, zoals de opleiding Toegepaste Informatica te HoGent, voorzien echter geen lessen rond datacompressie, waardoor deze basiskennis voor velen onbestaande is.

Deze bachelorproef biedt een oplossing voor dat probleem. Het vormt een gegronde basiskennis over datacompressie zonder onnodig complex te zijn, wat het geschikt maakt voor de grote variatie van belanghebbenden. Vanaf het voorstel waren de doelstellingen van deze bachelorproef, een antwoord bieden op zeven onderzoeksvragen en één hoofdonderzoeksvraag. Deze onderzoeksvragen worden hieronder nog eens kort aangegaan met een terugblik naar de kennis verworven in deze bachelorproef.

Hoe is datacompressie binnen IT ontstaan?

Vele onderzoekers zijn het erover eens dat datacompressie dateert van voor de uitvinding van de computer. Zo kan morsecode gezien worden als een vorm van datacompressie. Morsecode is uitgevonden voor het computertijdperk, in 1832, door Samuel F.B. Morse. Het kan gezien worden als een vorm van datacompressie doordat veelvoorkomende letters een kortere audiotoon kregen dan minder gebruikte letters (Pop, 2015).

Wat waren enkele van de eerste compressie-algoritmen?

Enkele van de eerste compressie-algoritmen komen aan bod in deel 4.2.2 van deze bachelorproef. Twee belangrijke compressie-algoritmen die al meer dan vijftig jaar bestaan, maar tot heden de basis vormen voor vele toepassingen binnen datacompressie zijn run length encoding en Huffman coding. De werking van deze compressie-algoritmen is dan ook uitgebreid aan bod gekomen in deze bachelorproef. Een theoretische uitleg met een eenvoudig voorbeeld is voorzien in deel 4.3. In de proof of concept datacompressietool gemaakt voor deze bachelorproef zijn het ook deze twee compressie-algoritmen die gebruikt worden. Deze datacompressietool is verder toegelicht in hoofdstuk 5.

Waar zitten de verschillen tussen de afbeeldingsformaten en video codecs?

Afbeeldingsformaten en video codecs hebben meer gemeen dan oorspronkelijk gedacht zou worden. Vele afbeeldingsformaten vormen de basis voor een goed presterende video codecs en de besproken afbeeldingsformaten WebP en HEIC vinden juist hun ontstaan in videocompressie. De onderlinge verschillen tussen de verschillende besproken afbeeldingsformaten en video codec is af te leiden uit de voordelen en nadelen te vinden in hoofdstuk 6 en 7. De delen over het maken van een juiste keuze van afbeeldingsformaat (deel 6.3) en video codec (deel 7.3) bieden aan de hand van enkele overzichten ook een duidelijk antwoord op deze vraag.

Hoe kan datacompressie correct geïmplementeerd worden?

Hoe datacompressie correct geïmplementeerd kan worden, is terug te vinden in verschillende delen van deze bachelorproef. De datacompressietool en achterliggende code wordt uitgebreid besproken in hoofdstuk 5. Deze is open source ter beschikking gesteld op GitHub en kan zonder enige licenties gebruikt en aangepast worden. Er worden ook enkele beperkingen met deze datacompressietool toegelicht en mogelijke oplossingen wat een geïnteresseerde lezer kan aanzetten deze beperkingen zelf weg te werken. Er wordt ook toegelicht hoe nieuwe generatie afbeeldingsformaten geïmplementeerd kunnen worden met ondersteuning voor alle internetbrowsers in gedachten. Dit is verder toegelicht in deel 6.4.

Wat is het verschil tussen de afbeeldingsformaten: PNG, JPEG, JPEG2000, WebP en HEIF?

Elk afbeeldingsformaat wordt toegelicht in hoofdstuk 6. Hier wordt zowel het ontstaan, de werking en voordelen en nadelen van de verschillende afbeeldingsformaten uitgelegd. Dit biedt samen met de resultaten van het onderzoek besproken in deel 9.5 en 9.8 een uitgebreid inzicht in de verschillen tussen deze afbeeldingsformaten.

Wat is het verschil tussen de video codecs: H.264/AVC, H.264/SVC, H.265/HEVC en AV1?

Elke video codecs wordt toegelicht in hoofdstuk 7. Hier wordt zowel het ontstaan als de voordelen en nadelen van de verschillende video codecs aangekaart. Zoals in de overzichten van deel 7.3 duidelijk is weergegeven is er binnen videocompressie een enorm probleem van complexe licenties. Het is ook daarom dat Google samenwerkt met tal van andere grote bedrijven als Mozilla en Microsoft en zo AV1 op de markt heeft gebracht. Deze veelbelovende video codec wordt ook in hoofdstuk 7 uitgebreid besproken.

Wat is DNA compressie en wat zijn andere uitdagingen binnen datacompressie?

Als afsluitend hoofdstuk (10) is een korte vermelding van enkele huidige uitdagingen binnen datacompressie toegelicht. Dit is bewust zeer beknopt gehouden zodat de lezer warm gemaakt wordt verder opzoekingswerk naar de interessante wereld van datacompressie te verrichten!

11.0.1 Hoofdonderzoeksvraag

Door het beantwoorden van alle sub onderzoeksvragen kan de hoofdonderzoeksvraag, 'waarom moet er stilgestaan worden bij het gebruiken van compressie-algoritmen, hoe kies je een geschikt compressie-algoritme voor een bepaalde use case en hoe implementeer je dit best', door de lezer zelf beantwoord worden. Deze bachelorproef bevat namelijk alle nodige informatie om op een gegronde manier op zoek te gaan naar een compressie-algoritme voor een bepaalde use case.

11.0.2 Mogelijke uitbreidingen

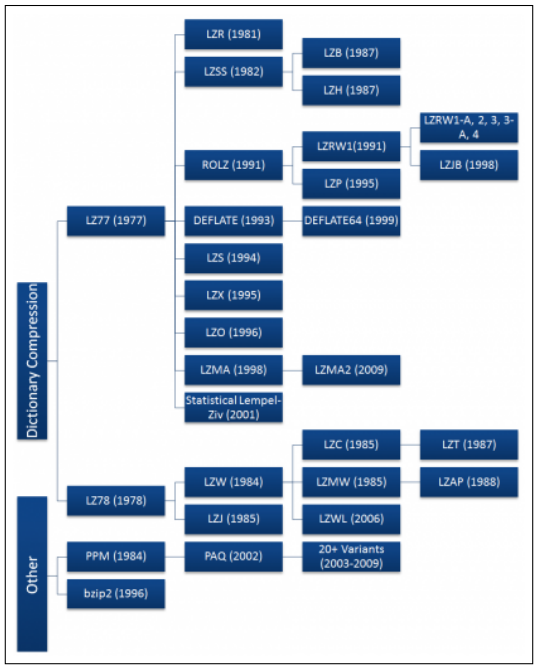
Desondanks deze bachelorproef reeds uit meer dan honderd pagina's bestaat, is er nog altijd ruimte voor uitbreidingen. Zo kan het onderzoek herwerkt worden zodat de verschillende afbeeldingen een gelijke bestandsgrootte hebben, wat een conclusie trekken makkelijker zal maken. Deze uitbreiding is relatief eenvoudig te voorzien doordat afbeeldingsevaluatietool, die gemaakt is voor deze bachelorproef, open source is en gratis gebruikt mag worden.

Maar ook uitbreidingen op de gemaakte proof of concept datacompressietool zijn mogelijk. Denk hierbij aan het combineren van run length encoding en Huffman coding of het implementeren van een compleet nieuw compressie-algoritme.

12. Bijlages

Om de leesbaarheid van de bachelorproef te behouden zijn sommige niet essentiële afbeeldingen, tabellen en overige documenten in dit hoofdstuk opgenomen.

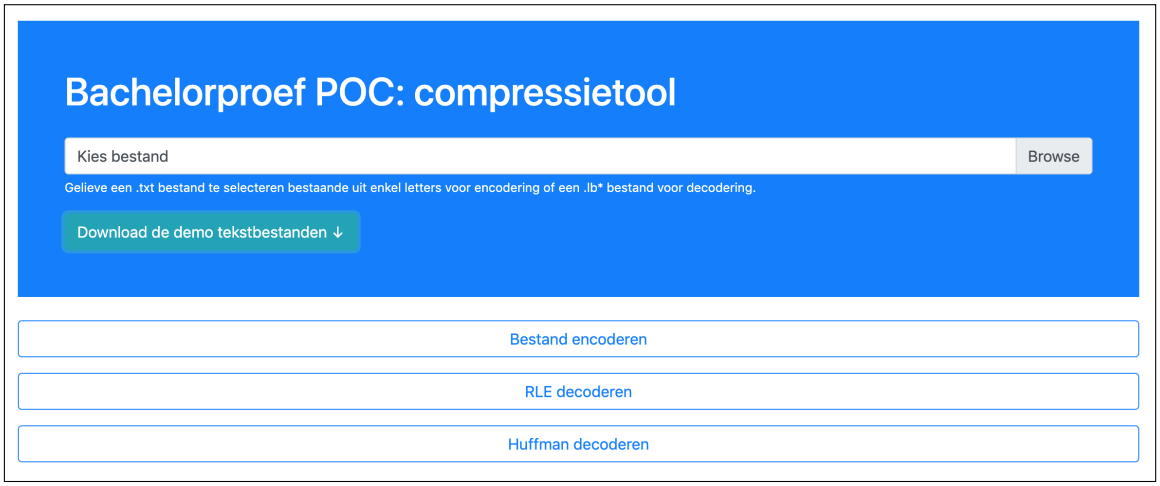
12.1 Figuren literatuurstudie



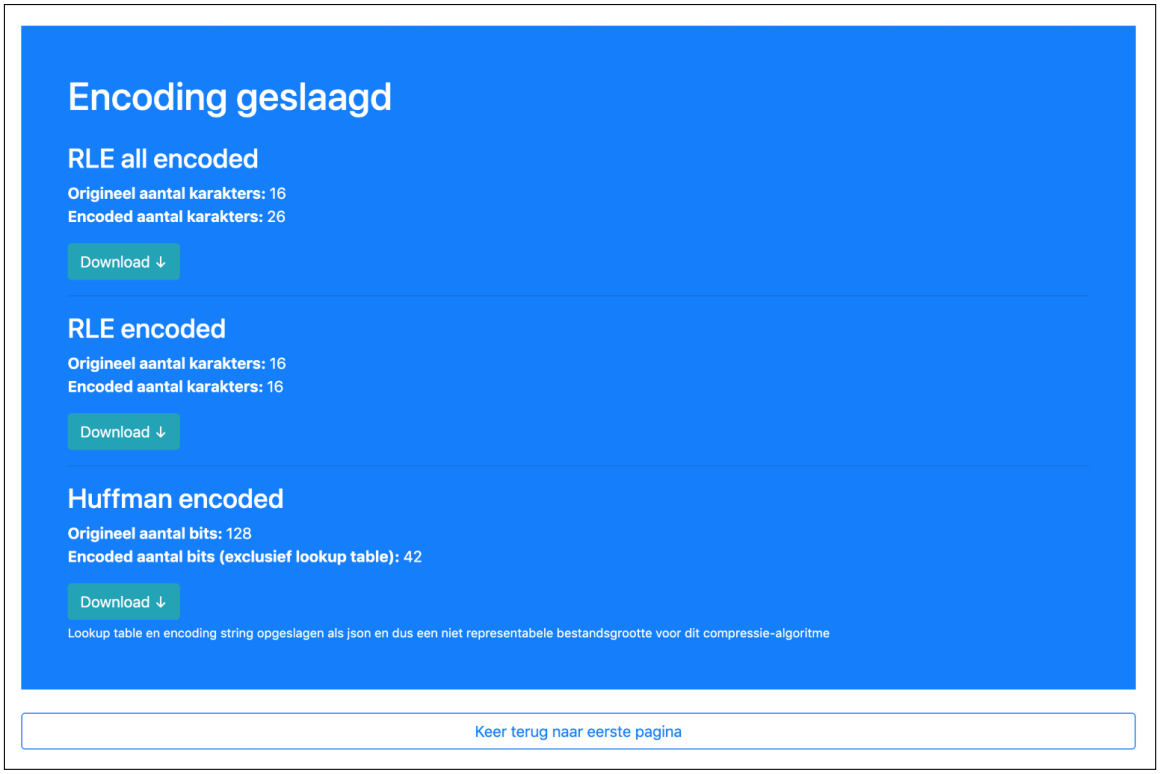
Figuur 12.1: Lossless datacompressie overzicht (Rihamichael, 2011)

12.2 Screenshots datacompressietool

De datacompressietool is raadpleegbaar via de website van Lennert Bontinck[1]. De code is te vinden op de GitHub repository van deze bachelorproef[2].



Figuur 12.2: Verwelkomingsscherm van de afbeeldingsevaluatietool.



Figuur 12.3: Scherm verkregen door een bestand te encoderen afbeeldingsevaluatietool.

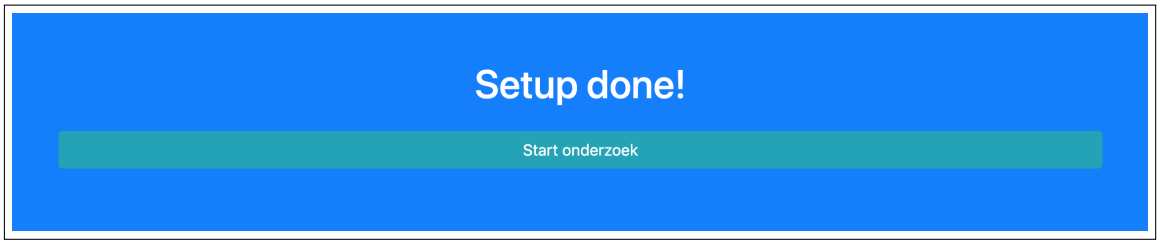
[1] POC compressietool - BAP – http://bit.ly/2Qx0gX9.
[2] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.



Figuur 12.4: Scherm verkregen door een bestand te decoderen afbeeldingsevaluatietool.

12.3 Screenshots afbeeldingsevaluatietool

De afbeeldingsevaluatietool is raadpleegbaar via de website van Lennert Bontinck[3]. De code is te vinden op de GitHub repository van deze bachelorproef[4].



Figuur 12.5: Setup scherm van de afbeeldingsevaluatietool.

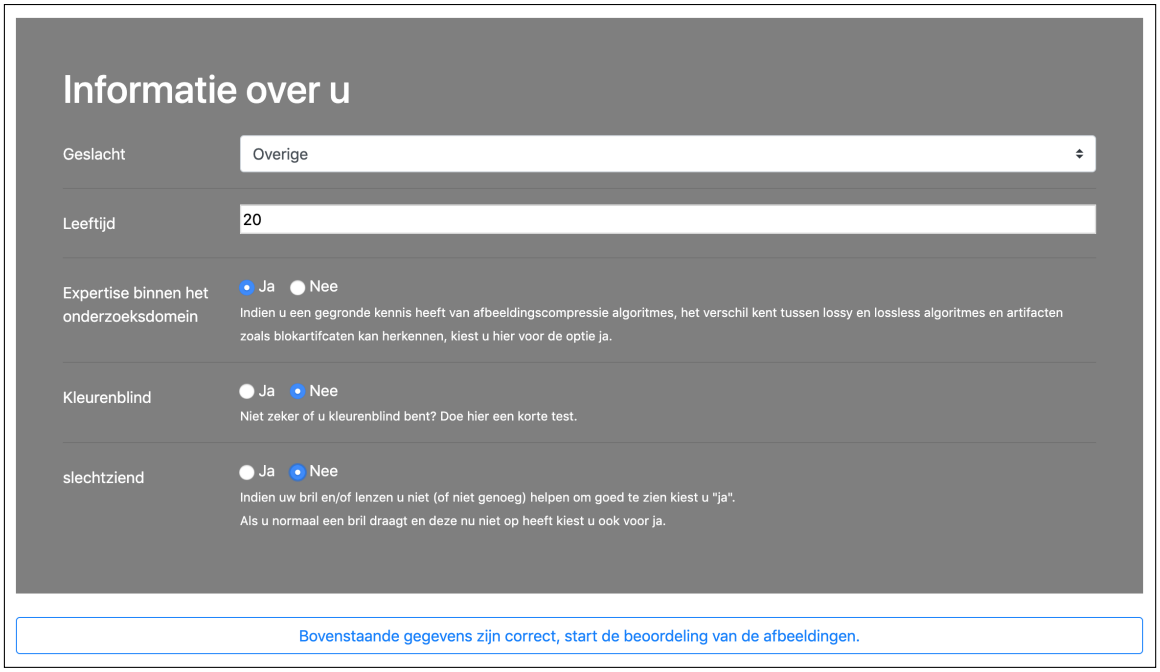


Figuur 12.6: Verwelkomingsscherm van de afbeeldingsevaluatietool.

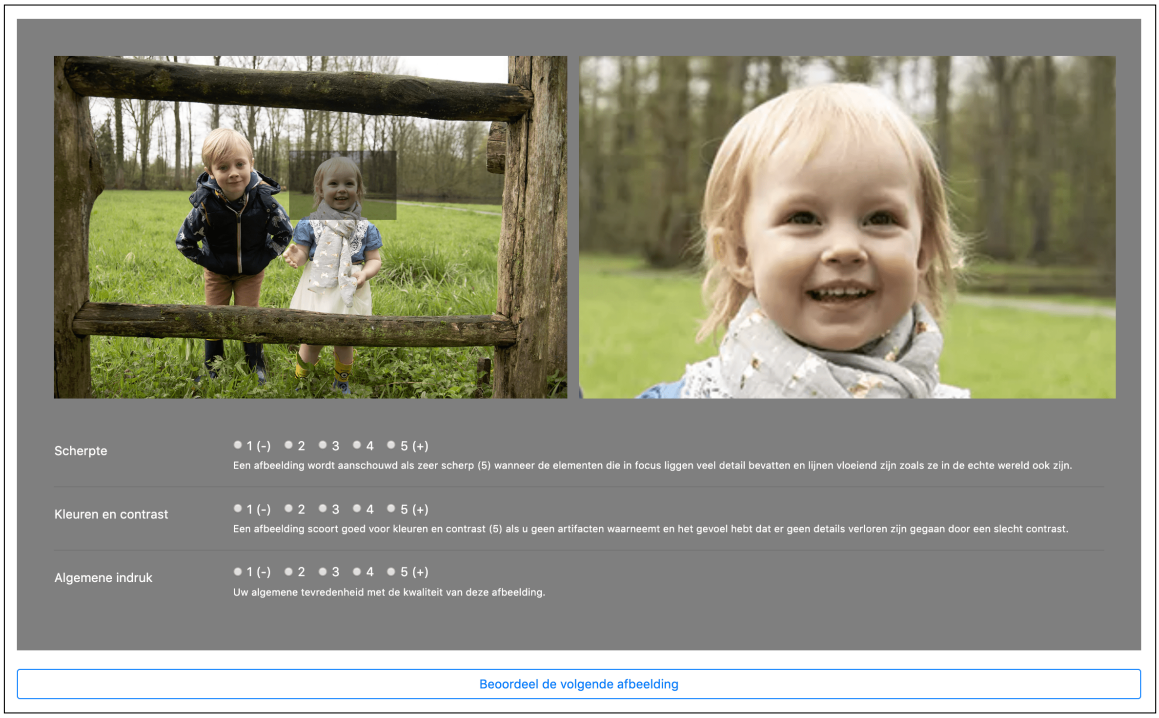
[3] Bachelorproef onderzoek: afbeeldingskwaliteit – http://bit.ly/2JIIBeE.
[4] Github repository bachelorproef datacompressie – http://bit.ly/2W98hqz.



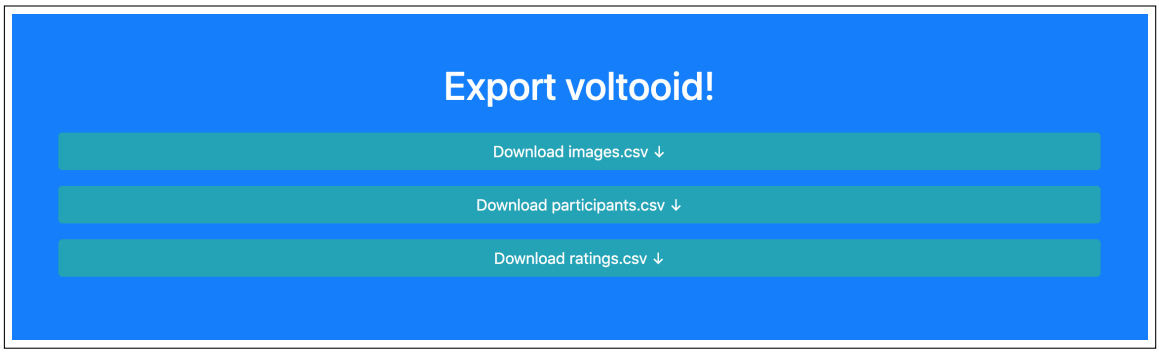
Figuur 12.7: Introductievideo met uitleg van de afbeeldingsevaluatietool.



Figuur 12.8: Informatie over de deelnemer in de afbeeldingsevaluatietool.



Figuur 12.9: Afbeelding beoordelen in de afbeeldingsevaluatietool met 5x zoom mogelijkheid.

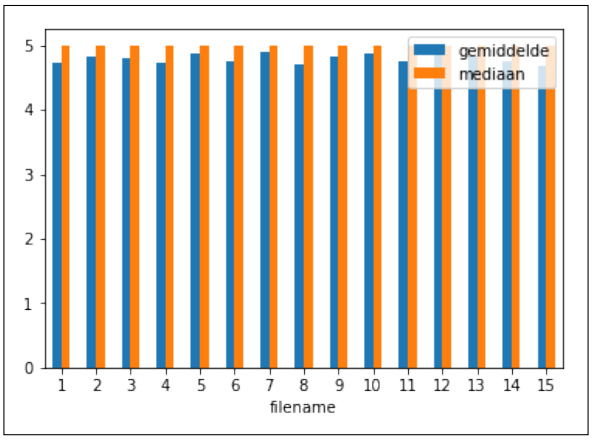


Figuur 12.10: Het scherm waar de resultaat gedownload kunnen worden.

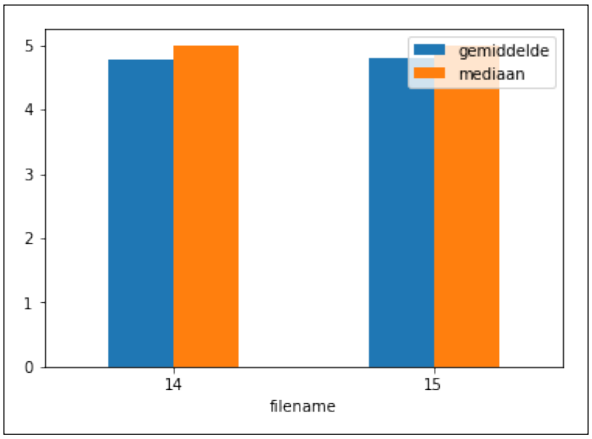
12.4 Extra documenten onderzoek

Afbeelding	render PNG	render JPEG	render JPEG2000	render WEBP
1	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 8 - high	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 60	embed color profile quality 60
2	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 8 - high	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 60	embed color profile quality 60
3	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 8 - high	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 60	embed color profile quality 60
4	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 8 - high	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 60	embed color profile quality 60
5	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 8 - high	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 60	embed color profile quality 60
6	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 10 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 80	embed color profile quality 80
7	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 10 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 80	embed color profile quality 80
8	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 10 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 80	embed color profile quality 80
9	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 10 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 80	embed color profile quality 80
10	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 10 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 80	embed color profile quality 80
11	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 12 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 100	embed color profile quality 100
12	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 12 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 100	embed color profile quality 100
13	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 12 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality 100	embed color profile quality 100
14	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 12 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality lossless	embed color profile lossless save meta NO
15	embed color profile smallest file size (slowest saving)	embed color profile baseline standard (browser support) quality: 12 - maximum	embed color profile NIET fast mode NIET include metadeta NIET JP2 compatible Wavelet float Tile size 1024x1024 growing thumbnail quality lossless	embed color profile lossless save meta NO

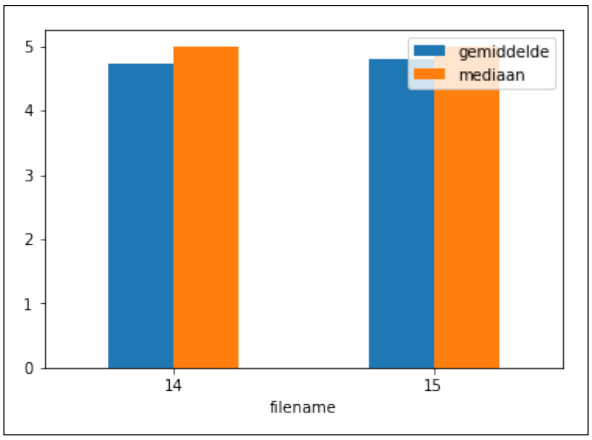
Figuur 12.11: De renderopties ingesteld in Adobe Photoshop voor alle vijftien afbeeldingen en vier afbeeldingsformaten.



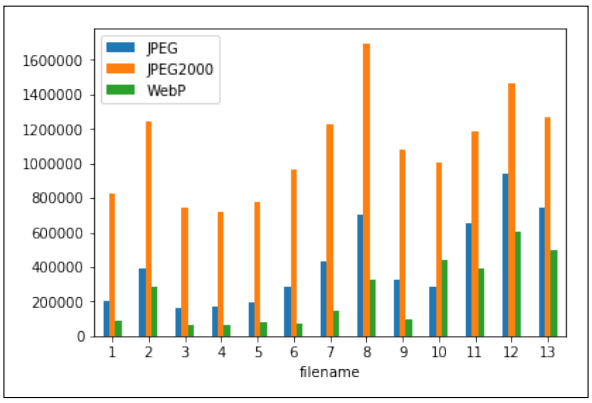
Figuur 12.12: Het gemiddelde en de mediaan voor het kenmerk 'algemene indruk' van afbeeldingen met het lossless PNG afbeeldingsformaat.



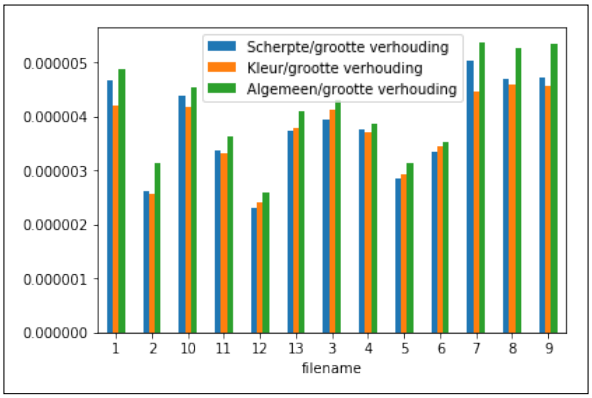
Figuur 12.13: Het gemiddelde en de mediaan voor het kenmerk 'algemene indruk' van afbeeldingen met het lossless WebP afbeeldingsformaat.



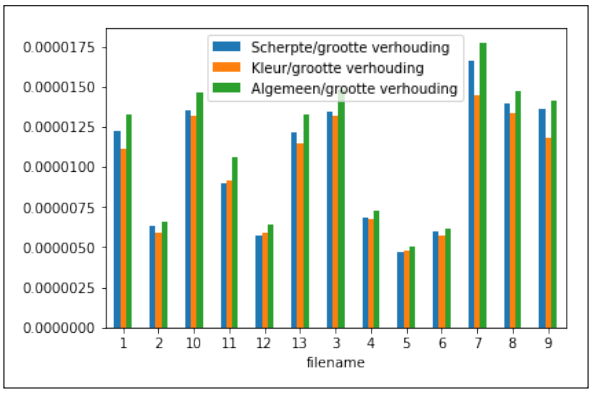
Figuur 12.14: Het gemiddelde en de mediaan voor het kenmerk 'algemene indruk' van afbeeldingen met het lossless JPEG2000 afbeeldingsformaat.



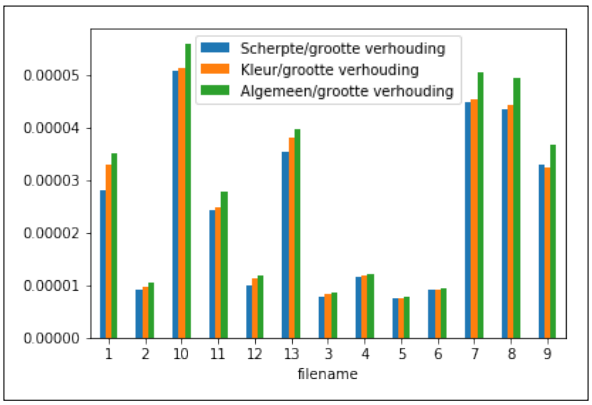
Figuur 12.15: De bestandsgrootte in bits voor de afbeeldingen die lossy gecomprimeerd zijn door zowel JPEG, JPEG2000 en WebP.



Figuur 12.16: Verhouding van de gemiddelde score ten opzichte van de bestandsgrootte voor JPF.



Figuur 12.17: Verhouding van de gemiddelde score ten opzichte van de bestandsgrootte voor JPEG.



Figuur 12.18: Verhouding van de gemiddelde score ten opzichte van de bestandsgrootte voor WebP.

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Datacompressie is overal, van vakantiefoto's op Instagram tot DNA compressie voor medisch onderzoek. Een wereld zonder datacompressie is ondenkbaar, er zouden enorme veelvouden van de huidige data opslag, bandbreedte en hardware capaciteit nodig moeten zijn om dezelfde data van vandaag te kunnen verwerken.

Bij DNA compressie is reeds een verkleining van meer dan 99 % behaald in sommige use cases (Afify, Islam & Wahed, 2011). Bij afbeelding- en videocompressie kan een andere codec, die een visueel gelijkaardig resultaat geeft, een bestandsgrootte van factor tien hebben. Dit wilt zeggen dat compressie één van de belangrijkste factoren is, zeker vanuit het perspectief van de eindgebruiker, voor het optimaliseren van snelheid en kostprijs bij applicatieontwikkeling en meer.

Bij een kleine bevraging van een tiental studenten toegepaste informatica te HoGent, één digital content team, twee mobile app developers en drie web developers bleek echter dat niemand van hen intensief bezig was met het bepalen van welke codec ze zullen gebruiken voor de afbeeldingen en video's binnen hun project. Vrijwel iedereen wist wel dat het belangrijk was afbeeldingen en video's te uploaden tegen een lagere resolutie, maar de gebruikte codec verdedigen ging voor velen niet verder dan "het is voorgesteld door dit tooltje" of "bij JPEG heb je geen doorzichtige achtergrond".

Deze vaststelling was de doorslaggevende factor voor het opstellen van deze bachelorproef. Door de grote diversiteit binnen de doelgroep voor wie dit onderzoek nuttig is, zal extra belang gehecht worden aan het eenvoudig uitleggen van complexe materie.

Deze bachelorproef en de bijhorende onderzoeken zullen trachten een antwoord te geven op volgende vragen:

- Hoe is datacompressie binnen IT ontstaan en wat waren enkele van de primitieve algoritmes?
- Waarom is er een groot verschil tussen de diverse afbeeldingcodecs en videocodecs?
- Wat is het verschil tussen JPEG en PNG?
- Wat is het verschil tussen H.264/AVC en H.264/SVC?
- Wat is DNA compressie en waarom is het de volgende uitdaging binnen datacompressie?

Hierdoor zou de hoofdonderzoeksvraag moeten kunnen beantwoorden worden, zijnde:

- Waarom moet stilgestaan worden bij het kiezen van een afbeelding- en/of videocodec?

A.2 Stand van zaken

Datacompressie bestaat al veel langer dan computers. Zo hebben bij morsecode, ontstaan in 1838, veelgebruikte letters een kortere code. Ook bij computers bestaat datacompressie al enige tijd, zo zijn LZ77 en opvolgers afkomstig uit 1977 en later. (Riha, 2011)

Dit wil ook zeggen dat er reeds een overweldigende hoeveelheid informatie te vinden is omtrent datacompressie en specifieke vormen van afbeelding- en videocompressie. Een heel goed boek over datacompressie is: 'Data compression, the complete reference' door Salomon (2006). Dit boek vereist, net zoals vele andere boeken omtrent datacompressie, een grondige kennis van algoritmes en wiskunde om de volledige 1017 pagina's te begrijpen.

Overigens zijn er al enkele interessante thesissen geschreven omtrent afbeeldingscompressie (bijvoorbeeld over JPEG optimalisatie door Wahlstrom (2015) en videocompressie (bijvoorbeeld over de artefacten die H.264 compressie met zich meebrengt door Anil (2013)). . Ook over het nog vrij recente topic, DNA compressie, zijn reeds tal van uitgebreide documenten beschikbaar. Enkele interessante artikels zijn die van Afify e.a. (2011) en Kuruppu (2012).

Bij deze documenten zijn echter enkele terugkerende problemen. Zelden of nooit wordt uitgelegd hoe de vergeleken bestanden verkregen zijn. Dit maakt het onmogelijk de experimenten te reproduceren of een soortgelijk onderzoek uit te voeren. Ook zijn de artikels vaak zeer complex (maar mathematisch correct) uitgelegd, wat het moeilijk maakt voor de doorsnee lezer alles te begrijpen. Of juist te simplistisch waardoor de verworven informatie niet volledig correct is. Ook ontbreekt er vaak een besluit om aan te tonen wat

moet onthouden worden en waarom al dan niet moet gekozen worden voor een bepaald datacompressie algoritme.

A.3 Methodologie

Deze bachelorproef zal bestaan uit zowel theoretische als praktische onderzoeken. De theoretische onderzoeken zullen bestaan uit enkele interviews met app en web development bedrijven alsook een uitgebreide literatuurstudie.

Het praktisch gedeelte zal bestaan uit het uitleggen van de onderliggende werking van JPEG en PNG en ook H.264/AVC en H.264/SVC. Er zal ook een vergelijkende studie gedaan worden door het comprimeren van dezelfde bestanden met de verschillende codecs. Een datacompressietool zal ook geschreven worden om de werking van primitieve compressietechnieken te verduidelijken.

A.4 Verwachte resultaten

De theoretische onderzoeken zullen een beeld geven van de kennis van datacompressie bij developers. Uit dit deel zal ook het ontstaan en de toekomst van datacompressie duidelijk worden. Dit gedeelte zal ook aantonen dat compressie meer is dan incrementele verbeteringen van oude technieken a.d.h.v. een bespreking van DNA compressie.

De praktische onderzoeken zullen inzicht proberen geven in de impact die het gebruik van een andere codec kan hebben op bestandsgrootte en kwaliteit. Dit zal gebeuren aan de hand van duidelijke grafieken waarop de bestandsgrootte in kb is af te lezen alsook de laadtijd in ms.

A.5 Verwachte conclusies

Het doel van deze bachelorproef is de lezer een beeld te geven hoe belangrijk het is stil te staan bij het kiezen van een gepaste codec voor de afbeeldingen en video's binnen een bepaald project. Dit omdat ook voor de eindgebruiker er een enorme tijdswinst en bandbreedte-/opslagbesparing tegenover kan staan. Ook de user experience is zeer belangrijk binnen deze bachelorproef: hoe bepaal je het middelpunt tussen kwaliteit en bestandsgrootte.

Deze bachelorproef zal de lezer in staat stellen een geschikte keuze te maken tussen JPEG en PNG afbeeldingcompressie alsook H.264/AVC en H.264/SVC videocompressie. Het zal de lezer ook een universele kennis geven van datacompressie en de huidige uitdagingen om hem aan te zetten tot verder onderzoek naar het voordeel van andere compressietechnieken.

Bibliografie

Adobe. (2018). Digital Negative (DNG). Verkregen van https://adobe.ly/2JQOdma

Afify, H., Islam, M. & Wahed, M. A. (2011). DNA Lossless Differential Compression Algorithm based on Similarity of Genomic Sequence Database. *CoRR*. Verkregen van https://goo.gl/BtWniW

Akamai. (2017). Akamai Online Retail Performance Report: Milliseconds Are Critical. Verkregen van http://bit.ly/2EfzfTl

Anil, C. R. (2013). *H.264 Video coding artifacts measurement and reduction of flickering* (masterscriptie, Blekinge Institute of Technology). Verkregen van https://goo.gl/iTrxTm

AOMedia. (2019). AOMedia: Legal Stuff. Verkregen van http://bit.ly/2I713ua

Barth, N. R. (2010). Ringing artifacts. Verkregen van http://bit.ly/2MrwAMW

Bhaskaran, V. & Konstantinides, K. (1995). Fundamentals of Lossy Video Compression. (pp. 87–128). doi:10.1007/978-1-4757-2358-8_4

Bontinck, L. (2019a). Bachelorproef onderzoek: afbeeldingskwaliteit. Verkregen van http://bit.ly/2JIIBeE

Bontinck, L. (2019b). Github repository bachelorproef datacompressie. Verkregen van http://bit.ly/2W98hqz

Bontinck, L. (2019c). Introductievideo bachelorproef onderzoek afbeeldingskwaliteit. Verkregen van http://bit.ly/2HqKN8k

Bontinck, L. (2019d). POC compressietool - BAP. Verkregen van http://bit.ly/2Qx0gX9

Cadik, M. (2004). Evaluation of Two Principal Approaches to Objective Image Quality Assessment. In *8th International Conference on Information Visualisation, IV 2004, 14-16 July 2004, London, UK* (pp. 513–518). doi:10.1109/IV.2004.1320193

Chen, Y., Mukherjee, D. & Han. (2018). An Overview of Core Coding Tools in the AV1 Video Codec. (pp. 41–45). doi:10.1109/PCS.2018.8456249

Coding Performance of H.262, H.264, and H.265. (2015). In *Next-Generation Video Coding and Streaming* (Hfdstk. 6, pp. 183–220). doi:10.1002/9781119133346.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119133346.ch6

DxOMark. (2019). DxOMark. Verkregen van http://bit.ly/2YVnQQw

Ebrahimi, F., Chamik, M. & Winkler, S. (2004). JPEG vs. JPEG2000: An objective comparison of image encoding quality. (Deel 5558). doi:10.1117/12.564835

fnordware. (2017). AdobeWebM. Verkregen van http://bit.ly/2Mmdho2

for Standardization, I. O. (2004). ISO/IEC 14496-10. Verkregen van http://bit.ly/2ws11rd

Google. (2010). You and site performance, sitting in a tree... Verkregen van http://bit.ly/2WNrkE1

Google. (2019). WebP Compression Techniques. Verkregen van http://bit.ly/2I58kdN

Grois, D., Nguyen, T. & Marpe, D. (2017). Performance comparison of AV1, JEM, VP9, and HEVC encoders (Conference Presentation). (p. 120). doi:10.1117/12.2283428

Homberger, D. (2018). WebPJS - Google's new image format WebP for not supported browsers (with alpha-channel). Verkregen van http://bit.ly/2JKiKD1

Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the Institute of Radio Engineers*, *40*(9), 1098–1101. Verkregen van http://bit.ly/2JInrfK

IEC. (1999). IEC 60027-2 Amendment 2: Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics. *IEC 60027-2 Amendment 2*.

IEEE Standard for Prefixes for Binary Multiples. (2009). *IEEE Std 1541-2002 (R2008)*, c1–4. doi:10.1109/IEEESTD.2009.5254933

Il Choi, W. & Jeon, B. (2004). Temporal Error Concealment with Block Boundary Smoothing. (Deel 3333, pp. 224–231). doi:10.1007/978-3-540-30543-9_29

imatest. (2019). ImaTest. Verkregen van http://bit.ly/2VXWXsY

Kaur, R. & Pooja. (2016). A Review of Image Compression Techniques. *International Journal of Computer Applications*, *142*, 8–11. doi:10.5120/ijca2016909658

KeyCDN. (2018). PNG to WebP - Comparing Compression Sizes. Verkregen van http://bit.ly/2X8h9d7

KeyCDN. (2019). JPG to WebP - Comparing Compression Sizes. Verkregen van http://bit.ly/2JGn1Hv

Kodituwakku, S. & U, S. (2010). Comparison of Lossless Data Compression Algorithms for Text Data. *Indian Journal of Computer Science and Engineering*, *1*.

Kuruppu, S. (2012). Compression of large DNA databases. Verkregen van https://goo.gl/5PsmZb

Lelewer, D. A. & Hirschberg, D. S. (1987). Data Compression. *ACM Computing Surveys*, *19*, 261–296.

M. Hannuksela, M., Lainema, J. & Malamal Vadakital, V. (2015). The High Efficiency Image File Format Standard [Standards in a Nutshell]. *Signal Processing Magazine, IEEE*, *32*, 150–156. doi:10.1109/MSP.2015.2419292

mathworks. (2019). MathWorks. Verkregen van http://bit.ly/2Widbm8

Mespotine, M. (2015). Mespotine-RLE-basic v0.9 - An overhead-reduced and improved Run-Length-Encoding Method.

Microsoft, G. W. (2019). Cluster size recommendations for ReFS and NTFS. Verkregen van http://bit.ly/2YvtFDS

Nallaperumal, K., Ranjani, J. J., Varghese, J. & Annam, S. (2006). A New Algorithm for Removing Blocking Artifacts in JPEG Compressed Images. doi:10.1109/ICIEA.2006.257369

Nemcic, O., Vranjes, M. & Rimac, S. (2007). Comparison of H.264/AVC and MPEG-4 part 2 coded video. (pp. 41–44). doi:10.1109/ELMAR.2007.4418796

Ng, W. K., Choi, S. & Ravishankar, C. (1998). Lossless and Lossy Data Compression. doi:10.1007/978-3-662-03423-1_10

Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., ... Wedi, T. (2004). Video coding with H.264/AVC: tools, performance, and complexity. *Circuits and Systems Magazine, IEEE*, *4*, 7–28. doi:10.1109/MCAS.2004.1286980

Pintus, M., Ginesu, G., Atzori, L. & Giusto, D. (2011). Objective Evaluation of WebP Image Compression Efficiency. (pp. 252–265). doi:10.1007/978-3-642-30419-4_22

Pop, M.-I. (2015). A Morse alphabet. Verkregen van http://bit.ly/2LQHPOt

Porcella, J. E. (2008). Color Blindness. *Encyclopedia of Special Education*. doi:10.1002/9780470373699.speced0461

Riha, M. (2011). History of Lossless Data Compression Algorithms. Verkregen 8 december 2018, van https://goo.gl/tUumPN

Rihamichael, E. (2011). History of Lossless Data Compression Algorithms. Verkregen van http://bit.ly/2VH7Bct

Robinson, A. & Cherry, C. (1967). Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE*, *55*(3), 356–364. doi:10.1109/proc.1967.5493

Roelofs, G. (2009). History of the Portable Network Graphics (PNG) Format. Verkregen van http://bit.ly/2HCbp4R

Rosell, B. (2019). WebP Express. Verkregen van http://bit.ly/2JHjUix

Salomon, D. (2006). *Data Compression: The Complete Reference* (4de ed.). Springer. Verkregen van https://goo.gl/yMoiqL

Schelkens, P., Bruylants, T., Temmermans, F., Barbarien, J., Dooms, A. & Munteanu, A. (2009). The JPEG 2000 family of standards. *Proceedings of SPIE - The International Society for Optical Engineering*. doi:10.1117/12.811847

Schwarz, H., Marpe, D. & Wiegand, T. (2007). Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Trans. Cir. and Sys. for Video Technol. 17*(9), 1103–1120. doi:10.1109/TCSVT.2007.905532

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, *27*(3), 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x

Shoppen, R. (2017). Wat is het verschil tussen raster- en vector afbeeldingen? Verkregen van http://bit.ly/2Qyh2VU

Skodras, A. & Ebrahimi, T. (2006). JPEG2000 image coding system theory and applications. (4 pp. - 3869). doi:10.1109/ISCAS.2006.1693472

Terras, M. M. (2008). *Digital Images for the Information Professional*. Brookfield, VT, USA: Ashgate Publishing Company.

tricks, C. .-.-. (2019). Using WebP Images. Verkregen van http://bit.ly/2K6IVTP

UGent, D. (2011). DNA compressie. Verkregen van http://bit.ly/2vrpKvt

Use, C. I. (2019). Can I use MPEG-4/H.264. Verkregen van http://bit.ly/30WgcXS

Wahlstrom, S. (2015). *Optimising JPEG image compression by identifying image characteristics* (masterscriptie, VU University Amsterdam en Mälardalens University). Verkregen van https://goo.gl/hEQ2LN

Weiman, D. (2010). ASCII Conversion Chart. Verkregen van http://bit.ly/2w2IpxV

WG1, I. (2014). Overview of JPEG. Verkregen van http://bit.ly/2wmNKjG

Zheng, B., Sun, R., Tian, X. & Chen, Y. (2018). S-Net: A Scalable Convolutional Neural Network for JPEG Compression Artifact Reduction. *Journal of Electronic Imaging*, *27*. doi:10.1117/1.JEI.27.4.043037