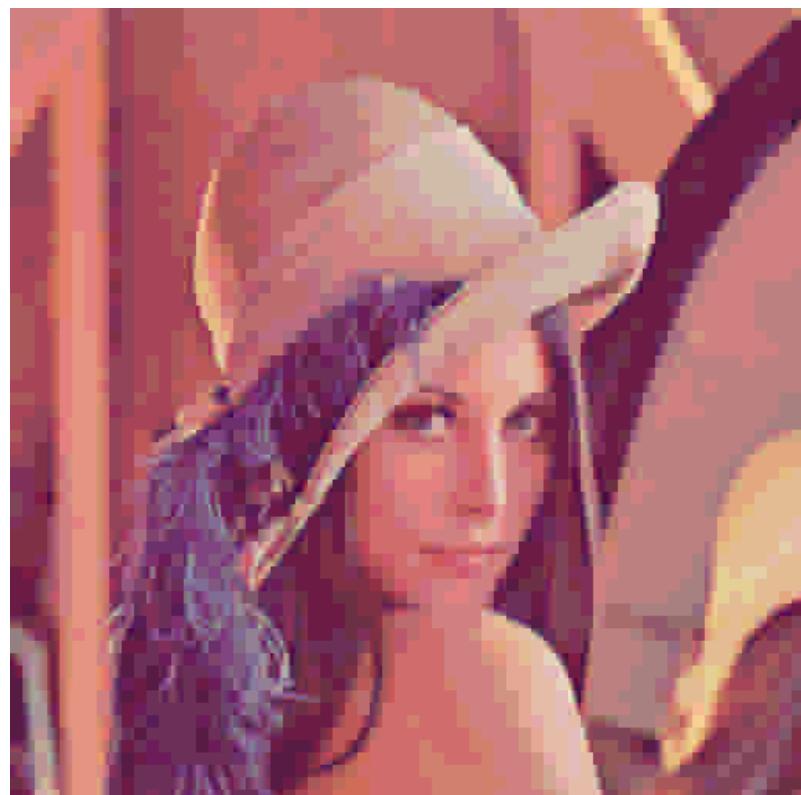


# *Lineær algebra i billedbehandling og -komprimering*

---



P1 PROJEKT  
GRUPPE B205  
MATEMATIK-TEKNOLOGI  
AALBORG UNIVERSITET  
DEN 18. DECEMBER 2015





**AALBORG UNIVERSITET**  
STUDENTERRAPPORT

Første Studieår v/ Det Teknisk-Naturvidenskabelige Fakultet  
Matematik-Teknologi  
Strandvejen 12-14  
9000 Aalborg  
<http://www.tnb.aau.dk>

**Titel:**

Lineær algebra i billedbehandling  
og -komprimering

**Projekt:**

P1-projekt

**Projektperiode:**

September 2015 - December 2015

**Projektgruppe:**

B205

**Deltagere:**

Frederik A. Vardinghus-Nielsen  
Mike L. H. Nguyen  
Thomas Kjems  
Sebastian E. Schrøder  
Kenneth Borup

**Vejledere:**

Christoffer Eg Sloth  
Mikkel Meyer Andersen

**Oplagstal: 7**

**Sidetal: 79**

**Appendiks: 4**

**Afsluttet 18-12-2015**

**Synopsis:**

This paper deals with the problem of ever growing resolutions, and therefore file sizes, of images taken with cameras on mobile phones. It examines two different signal processing techniques, which both detect correlation between values in an input signal, in the process of creating two different image compression algorithms. The two techniques are; a Discrete Cosine Transform and Principal Component Analysis. This paper will account for the mathematical properties of each of the techniques, incorporate them into image compression algorithms, and evaluate each algorithm by comparing quality and compression ratio of various test images for each algorithm. Through this comparison this paper determines which of the two is best suited as an image compression program for images taken with a mobile phone. This paper concludes, that the algorithm based on the Discrete Cosine Transform is more efficient at compressing images than the algorithm based on Principal Component Analysis, and it is therefore more suited as an image compression program for mobile phones.



## FORORD

Følgende rapport er udarbejdet af 1. semester studerende på uddannelsen for Matematik-teknologi ved Aalborg Universitet. Det overordnede tema for projektet er *Anvendt lineær algebra inden for teknologi*.

Rapporten omhandler projektkatalogets første projekt *Billedekomprimering*, hvor der undersøges forskellige metoder/matematiske modeller til komprimering af et billede. Fokus lægges på komprimeringsmetoderne DCT (brugt i bl.a. JPEG-komprimering) og PCA.

Forudsætningerne for at læse rapporten er et vist kendskab til lineær algebra og opbygningen af digitale billeder og tilhørende data, såvel som en indsigt i hvad en digital komprimeringsmetode er og gør.

### Læsevejledning

Igenmød rapporten er der kildehenvisninger, som afslutningsvist er samlet i en kildeliste bagerst i rapporten. Disse er opsat efter Harvardmetoden, hvorved kilder i teksten refereres med [Efternavn, [År], Side/Afsnit/Tidsinterval] (dog refereres der kun til årstal, såfremt dette kendes og kun til sidetal/afsnit, hvor dette giver mening). Kilden henviser til kildelisten, hvor boglige kilder er angivet med forfatter, titel, udgave og forlag, mens hjemmesider er angivet med forfatter, titel og dato for besøg. Figurer, tabeller og ligninger er nummereret i forhold til kapitel, hvorved eksempelvis første figur i kapitel 5 angives som *Figur 5.1*, den næste figur hedder derved *Figur 5.2* osv. Samme talnummerering findes på tabeller og ligninger, men kun tabeller og figurer har en forklarende tekst under de respektive figurer og tabeller. Talformatningen er dansk og decimalseparatot er "," og tusindseparatot er ".".

Der er i løbet af rapporten refereret til komprimeringsprogrammer såvel som billeder. Disse er at finde på nedenstående link, hvor billederne også kan ses i fuld størrelse:

Bilagslink: [www.goo.gl/qpC2Kz](http://www.goo.gl/qpC2Kz)

Indholdsfortegnelsen af dette link ses under appendiks D.

---

Frederik A. Vardinghus-Nielsen

Mike L. H. Nguyen

Thomas Kjems

---

Sebastian E. Schröder

Kenneth Borup



## INDHOLDSFORTEGNELSE

<b>Kapitel 1 Projektbeskrivelse</b>	<b>1</b>
1.1 Problemanalyse . . . . .	1
1.2 Problemformulering . . . . .	3
<b>Kapitel 2 Digitale billeder</b>	<b>5</b>
2.1 Billedkomprimering . . . . .	7
<b>Kapitel 3 Billedkomprimering med diskret cosinustransformation</b>	<b>9</b>
3.1 Diskret Fouriertransformation . . . . .	9
3.1.1 JPEG . . . . .	10
3.1.2 Diskret Cosinustransformation . . . . .	11
3.1.3 Eksempler på brug af den diskrete cosinustransformation . . . . .	17
3.1.4 Forbehandling af billede . . . . .	19
3.1.5 Kvantisering . . . . .	21
3.1.6 Entropikodning - Huffmankodning . . . . .	23
3.2 DCT anvendelse . . . . .	25
<b>Kapitel 4 Billedkomprimering med Principal Component Analysis</b>	<b>31</b>
4.1 Statistiske værktøjer . . . . .	32
4.1.1 Støj . . . . .	33
4.1.2 Redundans . . . . .	33
4.2 Kovariansmatrix . . . . .	33
4.2.1 Gennemsnit . . . . .	33
4.2.2 Varians og kovarians . . . . .	34
4.2.3 Diagonalisering af kovariansmatricen . . . . .	35
4.3 Udregn PCA . . . . .	36
4.4 Billedekomprimering med PCA . . . . .	38
<b>Kapitel 5 Resultater</b>	<b>45</b>
5.1 Komprimeringsgrad . . . . .	45
5.2 Kvalitet . . . . .	46
<b>Kapitel 6 Vurdering</b>	<b>51</b>
6.1 Sammenligning - Lena . . . . .	51
6.1.1 Skalering af Lena . . . . .	54
6.2 Vurdering af mobilbilleder . . . . .	55
6.3 Fejlkilder . . . . .	58
<b>Kapitel 7 Konklusion</b>	<b>61</b>
7.1 Perspektivering . . . . .	62

<b>Litteratur</b>	<b>63</b>
<b>Appendiks A DCT udledning</b>	<b>67</b>
<b>Appendiks B Huffmantræ</b>	<b>73</b>
<b>Appendiks C Bevis for positive egenværdier</b>	<b>75</b>
<b>Appendiks D Indhold i bilagsmappe</b>	<b>77</b>

## KAPITEL 1

---

### PROJEKTBESKRIVELSE

Følgende afsnit har til formål at skabe rammerne for rapportens retning igennem en problemanalyse, problemformulering og et metodeafsnit. Problemanalysen analyserer det initierende problem og undersøger præmisserne for problemet. Dette bliver efterfølgende brugt til klarlægning af problemformuleringen, der har hovedfokus på billedkomprimeringsmetoder vha. lineær algebra.

#### 1.1 PROBLEMANALYSE

Digitale billeder er en stor del af mange menneskers hverdag, både direkte og indirekte, og har ligeledes en afgørende betydning for mange store teknologivirksomheders eksistens. Qua den rivende hurtige teknologiudvikling det seneste årti, er kameraer i mobiltelefoner blevet udviklet til at have bl.a. højere opløsning. Højere opløsning betyder, at billedfilerne også fylder betydeligt mere digital hukommelse, hvorved en mobiltelefons lagerplads hurtigere end tidligere opfyldes. Det betyder ydermere, at download og upload til internettet bliver langsommere, da en større mængde data tager længere tid at downloade end en lille mængde data.

Den enkelte bruger, der typisk besidder et stort fotoalbum på sin mobil, påvirkes af den stigende størrelse på billedfilerne. Dette gør de ved, at deres mobiltelefons lagerplads hurtigere bliver opfyldt af billederne, og da de fleste ønsker at gemme så mange billeder som muligt, kan det være interessant at komprimere billederne for at få plads til flere billeder.

Hele essensen i komprimering af billeder er at gøre filerne mindre, hvorved de fylder mindre lagerplads, hvilket kan anses som en billedebehandling. Billedet, der behandles, kan anses som værende en  $m \times n$  matrix, hvor  $m$  og  $n$  er billedeets dimensioner i pixels. Indholdet i matricen er billedeets pixelværdier, og indeholder forskellige data afhængigt af billedeformatet. Billedbehandlinger, såsom komprimering, fremkommer ved operationer på matricen, og er af den matematiske gren lineær algebra, der netop omhandler matrixoperationer.

Der findes mange forskellige komprimeringsmetoder, og disse kan grundlæggende opdeles i to forskellige typer; tabsfri og ikke-tabsfri komprimering. Tabsfri komprimering er en komprimering af dataene uden at fjerne noget data, og ikke-tabsfri er det modsatte; en komprimering med tab af data.

JPEG-komprimering er den mest udbredte komprimeringsmetode, og har oprindelse i gruppen med samme initialer; Joint Photographic Experts Group. Metoden som bruges i JPEG er tilnærmelsesvist en *Diskret Cosinus Transformation* (fremover kaldet DCT), som fungerer på baggrund af tesen om at menneskets syn ikke særlig godt ser høje

frekvenser, hvorfor disse kan fjernes fra billedet uden betydelig synlig ændring. DCT er en ikke-tabsfrei komprimeringsmetode og ved dekodning af den komprimerede fil, vil det genskabte billede kun være en tilnærmelse af det originale, og ikke en eksakt kopi. Forskellen mellem det genskabte og originale billede er de høje frekvenser, og tabet ved komprimeringen burde derfor ikke være synlig for det blotte øje. DCT er dataafhængig og transformationerne fungerer derved på alle billede (dog med forskellig kvalitet). En anden metode til komprimering er *Principal Component Analysis* (fremover kaldet PCA), der ligesom DCT er en ikke-tabsfrei komprimeringsmetode. PCA er en statistisk metode, der i store mængder data identificerer signifikante (og dermed også insignifikante) data. PCA finder kovariansen mellem dataene og tildeler de enkelte dimensioner en egenvektor med dertilhørende egenværdi, der bruges til vurdering af signifikante og insignifikante data. Akserne med størst varians udtrykker mest af billedet, og akserne med mindst varians fjernes derfor i komprimeringen. Modsat DCT fjernes de høje frekvenser ikke med PCA, men derimod de data der har mindst betydning for billedets repræsentation. Det betyder dog også at billedet (ligesom DCT) blot skaber en tilnærmelse af det oprindelige billede ved dekodning. PCA er dataafhængig, og billedet har større betydning for komprimeringen med PCA end med DCT.

På trods af mange ligheder mellem PCA og DCT, fungerer de i praksis meget forskelligt, hvilket også betyder at de har hver deres fordele og ulemper. Det betyder også at hver metode fungerer bedre på nogle former for billede end andre, hvorved det kan være svært at finde den bedst egnede metode til et vilkårligt billede.

## Hastighed

I forbindelse med brug af et komprimeringsprogram i dagligdagen er det afgørende, at tiden det tager at udføre komprimeringen, er så kort som mulig, da det må antages at de fleste ønsker, at komprimeringen skal foregå hurtigst muligt. Da projektet ikke har sin kerne i programmering og kvaliteten af de programmer, der er udviklet til komprimeringen, er fokus ikke lagt på dette. Det betyder imidlertid, at hastigheden ikke kan måles i programmerne, der er udviklet til dette projekt. Til gengæld kunne en kompleksitetsanalyse laves, som giver et udtryk for hvor mange beregninger, der skal laves for at lave de to komprimeringsmetoder. Kompleksitetsanalyse ser bort fra den respektive enhed, der laver beregningerne, og giver derfor et udtryk for antal beregninger, der skal laves. Kompleksitetsanalyse er dog også udenfor denne rapports fokusområde og udføres derfor ikke. Bemærk at hastigheden ved komprimering har stor betydning for metodens relevans i virkeligheden men undersøges ikke i denne rapport.

## CASE - BILLEDER PÅ MOBILTELEFONER

Den følgende rapport har udgangspunkt i den stigende brug af mobiltelefoner som kamera i dagligdagssituationer, hvorved mængden af billede på den gængse mobiltelefon stiger. Dette giver dog problemer med hurtigt fyldt lagerplads på mobiltelefonerne, og deraf frustrerede brugere. Ved at komprimere billede opnås der mere fri hukommelse

og dermed plads til flere billeder. Der er dog begrænsninger i forhold til kvaliteten af komprimeringen, da den hurtige teknologiudvikling også resulterer i større krav til billederne, der tages. Det er for brugeren vigtigt, at komprimeringen kan være en dagligdagsopgave, og dermed ikke kræver lang tid at udføre. Det er også vigtigt, at billederne bibeholder en vis skarphed og sin farve, for at brugeroplevelsen bibeholdes.

## 1.2 PROBLEMFORMULERING

I kommende rapport bliver digitale billeder komprimeret vha. lineær algebra og undersøges ud fra problemformuleringen:

*Hvilken af komprimeringsmetoderne DCT og PCA komprimerer bedst billede Lena [Wakin], og hvordan fungerer disse komprimeringsmetoder rent matematisk?*

*Hvilken af metoderne bibeholder bedst kvaliteten af billede i forhold til komprimeringsgraden?*

*Hvilken metode komprimerer bedst rigtige mobilbilleder, og er der nogen sammenhæng mellem disse resultater?*

## METODE

På trods af at der findes et utal af komprimeringsmetoder, undersøges der i følgende rapport blot komprimeringsmetoderne DCT og PCA. Begge metoder fungerer ved at fjerne de visuelt mindst relevante data om billede. Komprimeringsmetodernes matematiske grundlag undersøges og bruges efterfølgende til udvikling af python-programmer til komprimering af et billede. Billedet der komprimeres er Lena [Wakin], som er  $512 \times 512$  pixels stort og i farveformatet RGB (Rød, Grøn, Blå). Resultaterne af komprimeringerne bliver sammenlignet i forhold til parametrene komprimeringsgrad og billedekvalitet. Komprimeringsgraden bliver sammenlignet på baggrund af antal bits i det oprindelige billede i forhold til det komprimerede billede. Billedkvaliteten vurderes på baggrund af forskellene i pixelværdierne mellem det oprindelige og komprimerede billede såvel som ved SNR-værdien for de forskellige komprimeringsgrader. Afslutningsvist bruges komprimeringsprogrammerne på flere forskellige mobilbilleder, for at afklare hvilke parametre, der har betydning for hvilken af metoderne, der fungerer bedst ved dagligdagsbilleder taget med en mobil.

På baggrund af de ovenstående afsnit og dertilhørende data bliver der i afsnit 6 vurderet på kvaliteten og komprimeringsgraden for de to metoder, og sætte dem i relation til casen for rapporten.



## KAPITEL 2

---

### DIGITALE BILLEDER

Digitale billeder består af pixels. En pixel er den mindste bestanddel i et digitalt billede og er små ensfarvede områder. I gråtonebilleder beskrives en pixels farveintensitet med en værdi fra nul til 255, hvor nul er sort, og 255 er hvid. Dette er de to ekstremer i gråtonerne - imellem de to er der yderligere 254 gråtoner, som udgør en glat overgang fra sort til hvid. Et gråtonebillede udgøres altså af pixels af op til 256 forskellige gråtoner.

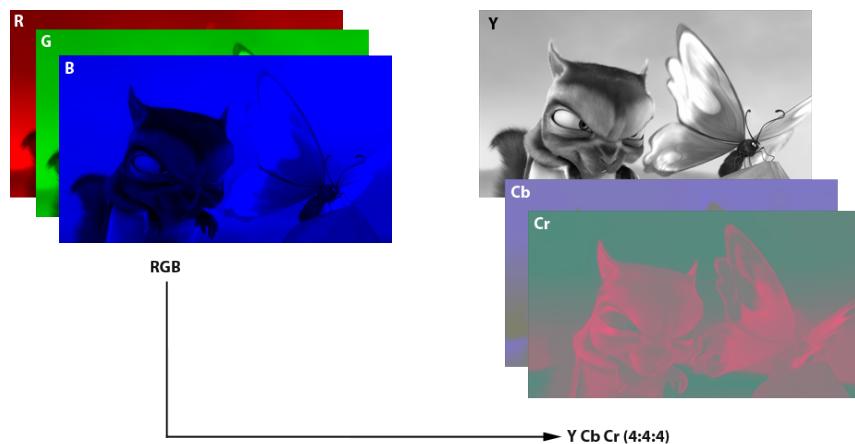
Et farvebillede behøver flere værdier for at beskrive hver enkelt pixels farve. Der findes flere forskellige farvemodeller til beskrivelse af farver. RGB-modellen (Rød, Grøn, Blå) er en udbredt additiv model, som beskriver en pixels farve med tre værdier; én for hver farves intensitet i den pågældende pixel, således:  $\text{Pixel} = (R, G, B)$ . RGB-modellen bruges digitalt til at vise farver på skærme. Modellen er en additiv model, da den adderer grundfarverne i modellen i skabelsen af nye farver. Adderes alle tre grundfarver med fuld intensitet fås hvid. Hvis sort ønskes bruges ingen farver. I skærme skabes oplevelsen af at se farver ved at flere grønne, røde eller blå pixels lyser, og da disse pixels er for små til at blive skelnet mellem af det menneskelige øje, opleves det som en kombination af disse - de adderes sammen til en ny farve [Newbold, afsnit 1-2]. På figur 2.2 på den følgende side ses hvordan flere forskelligfarvede pixels skaber et farveindtryk ud fra RGB-modellen. Digitale billeder er typisk defineret ud fra RGB, men i forbindelse med komprimering bruges farvemodellen YCbCR også ofte.

YCbCr står for  $Y$ : Lysintensitet (en repræsentation af lysintensiteten i billedet),  $Cb$  og  $Cr$ : De to farverepræsentationer (hhv. blå minus  $Y$  og rød minus  $Y$ ) [PC Mag Encyclopedia, afsnit 1]. Fordelen ved YCbCr er at menneskets øje ikke opfatter forskelle i farver ligeså tydeligt som i lysintensitet, hvorved de to farverepræsentationer i højere grad kan komprimeres end lysintensiteten. Dette betyder at et billede i YCbCr-farverummet kan komprimeres mere end et i RGB-farverummet. Det skal dog bemærkes, at de to farverum udtrykker de samme data, blot med to forskellige baser. [Marjanovic, afsnit 2].

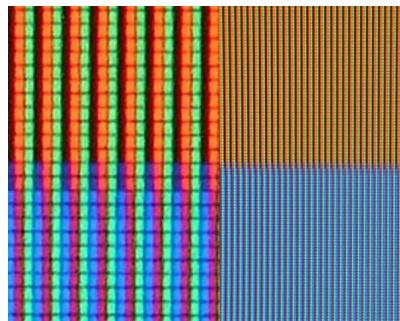
Der undersøges i følgende rapport med udgangspunkt i RGB-farverummet, da dette sparer transformationen til det nye farverum. Der ses bort fra, at brugen af YCbCr formentlig kunne have komprimeret billede yderligere, men da forskelle på farverum ikke er fokus i denne rapport, bibringes billede blot i RGB.

Når hver pixel kan beskrives ved én eller flere værdier, kan det digitale billede repræsenteres fuldstændigt som en matrix med indgange for hver pixels farveintensitet. Hvis billedet er et digitalt billede af dimensionerne  $m \times n$ , kan det altså beskrives ved en  $m \times n$  matrix. Undersøges farvebilleder, transformeres de enkelte farver hver for sig. Et eksempel på et gråtonebillede ses på figur 2.4 på næste side.

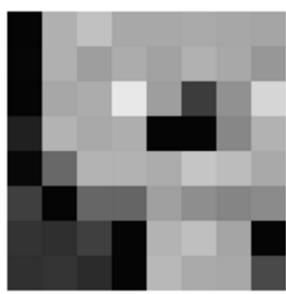
Når digitale billeder kan repræsenteres af matricer, betyder det også, at de kan behandles ved brug af regneregler for matricer. Moderne billedbehandlingsteknikker udnytter



**Figur 2.1.** Eksempel på farvetransformation fra RGB til YCbCr [Spears og Munsil, afsnit 4]



**Figur 2.2.** Nærbillede af pixels, i farverne RGB.



**Figur 2.3.** Billede af  $8 \times 8$  pixels.

5	176	193	168	168	170	167	165
6	176	158	172	162	177	168	151
5	167	172	232	158	61	145	214
33	179	169	174	5	5	135	178
8	104	180	178	172	197	188	169
63	5	102	101	160	142	133	139
51	47	63	5	180	191	165	5
49	53	43	5	184	170	168	74

**Figur 2.4.**  $8 \times 8$  matrix for billedet.

regneregler for matricer, altså lineær algebra til at behandle digitale billeder.

Ud over at være i stand til at blive behandlet vha. lineær algebra har billeder den egenskab, at nærliggende pixels i et billede ofte udviser stor korrelation - de ligner hinanden og pixels kan derfor ofte beskrives ud fra de nærliggende pixels [Lui [b], s. 4]. Dette udnyttes i den senere billedkomprimering.

## 2.1 BILLEDKOMPRIMERING

Billedkomprimering er en billedbehandlingsteknik, som søger at komprimere en billedfils størrelse. Billedkomprimeringsmetoder kan opdeles i to kategorier; *tabsfri* og *ikke-tabsfri* komprimering.

Mængden hvorved et billede kan komprimeres afhænger af mange faktorer, som bl.a. inkluderer billedets størrelse, om det er i gråtoner, sort/hvid eller farver, hvor mange forskellige farveintensiteter billedet indeholder, og hvilket format billedet er i. Som følge af disse faktorer er det ikke altid muligt at udvikle den ideelle metode, som kun fjerner ubetydelig information - nogle gange er det nødvendigt også at fjerne betydningsfuldt information.

\_tabsfri betegner en komprimeringsteknik, som under komprimeringen ikke forårsager noget tab af information fra det originale billede. Her udnyttes det, at inputtet af informationer ikke er homogen, og at der derfor kan laves en statistisk model over disse informationer. Modellen bruges til at lave en komprimering eksempelvis ved brug af entropikodning, som vil blive forklaret nærmere i afsnit 3.1.6 på side 23. Ved brug af tabsfri komprimering, er det dekomprimerede billede magen til det originale. Som følge af bevaringen af alle informationer af det originale billede, er det ikke altid muligt at lave effektive komprimeringer. Blandt tabsfri billedformater kan nævnes PNG og TIFF. Essensen i disse metoder er at de finder fordelagtige måder at udtrykke de samme data på.

Ikke-tabsfri komprimering betegner en komprimeringsteknik, som under komprimeringen forårsager uigenkaldeligt tab af informationer om det originale billede. Det dekomprimerede billede er en efterligning af det originale billede. Det er ikke helt magen til da informationerne, som er brugt til dekomprimering, ikke er identiske med informationerne om det originale billede. Komprimeringsteknikker med tab benytter det faktum, at nogle af informationerne om billedet er visuelt ligegyldige for det menneskelige øje. Som følge af dette kan et billede kvalitet byttes for højere komprimeringsgrad, hvorved en større del betydelige informationer om det originale billede tabes. Som følge af denne komprimeringstekniks større råderum i forhold til den tabsfri, kan denne også opnå langt større komprimeringsgrader end den tabsfri. Blandt billedformater med tab kan nævnes GIF og PNG [Matthews, afsnit 2 og 4].

Det udnyttes ofte i ikke-tabsfri komprimering, at nærliggende pixels i et billede har stor korrelation og derfor til en vis grad kan udtrykkes ved én samlet værdi i stedet for en værdi for hver enkelt pixel. Til dette bruges en transformation, som kan transformere et input

med stor korrelation til et output uden stor korrelation, men som udtrykker korrelationen i det originale input [Lui [b], s. 2 og 4]. I domænet med de ukorrellerede koefficienter, kan der derefter sorteres i koefficienterne for at bevare de, som er vigtigst for det samlede udtryk i det orginale input.

## KAPITEL 3

---

### BILLEDKOMPRIMERING MED DISKRET COSINUSTRANSFORMATION

I dette kapitel udarbejde en billedkomprimeringsalgoritme med tab med udgangspunkt i den en diskret cosinustransformation. Før denne forklares, er det nødvendigt at undersøge den diskrete Fouriertransformation, som den stammer fra.

#### 3.1 DISKRET FOURIERTRANSFORMATION

Inden for signalbehandling er den Diskrete Fouriertransformation (DFT) et kendt værktøj. Transformationen er en lineær transformation, som udtrykker et signal på bølgeform ved sinus- og cosinusfunktioner [Bevelacqua, afsnit 1]. De fleste signaler er på bølgeform og kan derfor beskrives ved sinus- og cosinusfunktioner gennem DFT.

Signalerne kan beskrives i to domæner

- *Tidsdomænet*  
Signalet beskrives ved funktionsværdier til tiden  $t$ ,  $f(t)$ .
- *Frekvensdomænet*  
Signalet beskrives ved amplitude og fase for en frekvensfunktion.

Regneoperationer i tidsdomænet har tilsvarende regneoperationer i frekvensdomænet, som ofte er beregningsmæssigt simplere [The Neurophysiological Biomarker Toolbox, afsnit 2]. Af denne grund bruges den diskrete Fouriertransformation til signalbehandling.

Transformationen har desuden den egenskab, at den energikomprimerer signalet, som behandles. Energikomprimering betegner en transformations evne til at udtrykke mange signalværdier med høje korrelationer i domænet som færre koefficienter med lav korrelation i kodomænet [Wang et al., s. 1]. Koefficienterne i frekvensdomænet fortæller i hvor høj grad de enkelte basisfunktioner er repræsenteret i tidsdomænet - høje koefficienter viser en høj optræden af den tilhørende basisfunktion, mens lave koefficienter viser en lav optræden af den tilhørende basisfunktion.

Når DFT udtrykker korrelerede signalværdier som ukorrelerede koefficienter i frekvensdomænet, betyder det også at transformationen ikke formår at lave energikomprimering i høj grad, hvis signalværdierne er ukorrelerede. Dette betyder at, hvis der ønskes høj energikomprimering, skal DFT kun bruges på signaler, som består af korrelerede værdier.

I billedkomprimering er det, som tidligere beskrevet, interessant at udtrykke et billede ved færre værdier, da der derfor er færre værdier at gemme. DFT kan være et værktøj til dette. DFT fungerer bedst på signaler, som består af korrelerede værdier, og det gør netop et billede. DFT kan altså bruges til billedkomprimering.

Det viser sig imidlertid, at der findes et bedre værktøj til transformation af korrelerede

signalværdier til ukorrelerede koefficienter. Transformationen kaldes den diskrete cosinustransformation (DCT), og er udledt fra DFT [Blinn [2014], s. 3]. Udledningen ses i appendiks A på side 67.

DCT udmærker sig inden for billedbehandling på flere områder i forhold til DFT:

1. *Energikomprimering*

DCT opnår højere energikomprimering end DFT [Wang et al., s. 3 og 5]. Den formår at udtrykke informationer om signalværdier i tidsdomænet som færre koefficienter i frekvensdomænet end DFT formår. Det er ønskværdigt at udtrykke mange informationer ved få koefficienter i billedkomprimering.

2. *Reelle tal*

DCT benytter sig udelukkende af reelle tal i de transformerede koefficienter, hvorimod DFT benytter sig af både reelle og komplekse tal som koefficienter. Det er ønskværdigt, at regne udelukkende med reelle tal, da dette forsimpler den videre behandling af signalet.

Der findes desuden en transformation, som bygger på sinusfunktioner - den diskrete sinustransformation (DST). Denne transformerer også signalværdier om til frekvenskoefficienter, men energikomprimerer ikke godt - den er ringere end både DFT og DCT [Wang et al., s. 5].

Som følge af ovenstående grunde er der valgt at bruge DCT i den mest udbredte open-source komprimeringsstandard; JPEG.

På baggrund af at DCT

- har bedre energikomprimering end DFT [Wang et al., s. 5], og dermed skal færre koefficienter end DFT bruges til at danne en god approksimation af et signal [Blinn [2014], s. 5]
- ofte bruges i billedbehandling [Kumar et al., s. 1]
- bruges i JPEG [Blinn [2014], s. 1]

tages der i denne opgave udgangspunkt i DCT til at udføre billedkomprimeringer.

### 3.1.1 JPEG

Joint Photographic Experts Group udgav i 1992 den første JPEG-standard, som komprimerede og dekomprimerede billeder efter en bestemt algoritme. Et program, som kan gøre dette, kaldes en codec. Standarden er open source, og er siden 1992 blevet forbedret flere gange. I denne rapport undersøges en simpel og tidlig version.

JPEG gør brug af fire skridt i sin algoritme for at komprimere et billede fra dets fulde størrelse til en fil af mindre størrelse. Den inverse algoritme bruges til at dekomprimere en JPEG-fil til et billede. Da JPEG er en ikke-tabsfri komprimeringsalgoritme, er det dekomprimerede billede kun en efterligning af det originale. [Society for Industrial and Applied Mathematics [a], afsnit 1-7].

De fire skridt, som forklares nærmere i et senere afsnit, ser ud som følgende

1. *Forbehandling af billede*

Forbehandlingen indebærer at dele billedet op i kvadratiske undermatricer af 64 pixels. Dette gøres for at gøre den videre behandling af billedet med den diskrete cosinustransformation mere effektiv. Derudover trækkes 128 fra hver indgang i hver undermatrix for at centrere undermatricerne omkring nul.

2. *Diskret cosinustransformation*

Den diskrete cosinustransformation energikomprimerer billedet ved at udtrykke korrelerede signalværdier som dekorrelerede koefficienter i frekvensdomænet.

3. *Kvantisering*

Kvantiseringen fjerner mindre vigtige informationer om billedet ved at fjerne de dekorrelerede koefficienter, som ikke er vigtige i forhold til de korrelerede signalværdier. Dette skridt gør algoritmen ikke-tabsfrei.

4. *Entropikodning*

Entropikodningen opstiller en statistisk model over de kvantiserede koefficienter og koder dem i strenge bestående af 0 og 1, som kan gemmes effektivt i en fil, som er mindre end den oprindelige billedfil.

For at forstå hvorfor og hvordan algoritmen virker er det nødvendigt først at forstå den diskrete cosinustransformation. Derfor bliver dette skridt forklaret først.

### 3.1.2 Diskret Cosinustransformation

En diskret cosinustransformation er en lineær transformation, som afbilder en  $n$ -dimensionel vektor fra domænet  $\mathbb{R}^n$  til kodomænet  $\mathbb{R}^n$  - altså har transformationen samme kodomæne som domæne. Basisvektorerne i transformationen udledes fra cosinusfunktioner med forskellige frekvenser. Som følge af dette bliver den  $n$ -dimensionelle vektor afbildet ind i frekvensdomænet - den udtrykkes som en sum af cosinusfunktioner med forskellige frekvenser. Et eksempel på en funktion udtrykt ved bølgefunktioner ses i figur 3.1 på den følgende side.

Som tidligere nævnt er den diskrete cosinustransformation udledt fra den diskrete Fouriertransformation. Udledningen ses i appendiks A på side 67.

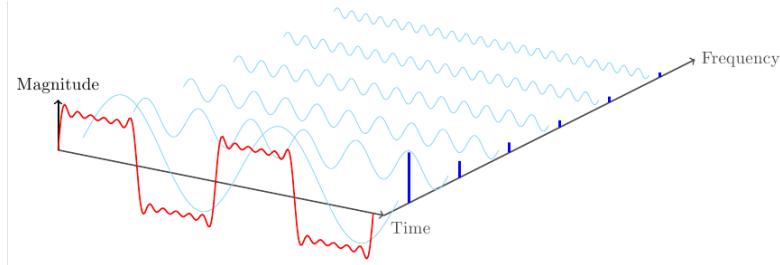
DCT bruges i udstrakt grad i signalbehandling på grund af dens høje energikomprimering. DCT kan altså komprimere meget signalværdi ned til få værdier i frekvensdomænet og dermed udtrykke signalværdier med stor korrelation som koefficienter med lav eller ingen korrelation [Lui [b], s. 1]. Transformationen fungerer dog bedst, når signalværdierne som transformeres, har en vis korrelation - hvis dette ikke er tilfældet, vil der ikke blive opnået gode resultater med transformationen.

DCT er imidlertid ikke den eneste transformation, som kan dekorrelere signalværdier. En anden kendt transformation er Karhunen-Loève transformationen. Denne transformation transformerer, som cosinustransformationen, også korrelerede koefficienter til dekorrelerede koefficienter. Karhunen-Loeve udmarkører sig ved at tilpasse sig hvordan signalet ser ud, for således at kunne energikomprimere meget effektivt - transformationen afhænger altså af signalet. Dette gør transformationen i stand til at lave den mest effektive

energikomprimering [Lui [b], s. 15]. Det er imidlertid ikke tidsmæssigt effektivt, at komponere en ny transformation hver gang et nyt signal behandles og derfor bruges den diskrete cosinustransformation i JPEG. Denne har en algoritme, som genererer en generel transformation, der kan bruges på mange datasæt [Wang, s. 2].

Da DCT er en lineær transformation, kan den bruges i billedbehandling til at energikomprimere digitale billeder, som er matricer, og som har stor korrelation mellem nærliggende pixels.

Den diskrete cosinustransformation, som kan udledes fra den diskrete Fouriertransfor-



**Figur 3.1.** Signal (rød) og cosinusbølger (blå kurver) samt amplituden (søjler)[Cao, Update 1].

tion, kan transformere i både én og to dimensioner. I denne opgave benyttes den todimensionelle. I ligning 3.1 ses den todimensionelle  $n$ -punkt diskrete cosinustransformation, hvor  $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$  transformerer til en matrix indeholdende dekorrelerede koefficenter. Matricen der transformeres, vil fremover refereres til som  $A$ .

$$T_{(i, j)} = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x, y) \cdot \alpha(i) \cdot \alpha(j) \cdot \cos\left(\frac{(2x+1) \cdot i \cdot \pi}{2n}\right) \cdot \cos\left(\frac{(2y+1) \cdot j \cdot \pi}{2n}\right) \quad (3.1)$$

[Sapiro, 7:12-11:10]

Hvor

- $T_{(i, j)}$  : indgang  $(i, j)$  i den transformerede matrix ved DCT
- $f(x, y)$  : indgang  $(x, y)$  i  $A$
- $n$  : matricens dimension
- $\alpha(i) = \alpha(j) = \begin{cases} \sqrt{\frac{1}{n}} & \text{hvis } i = 0 \\ \sqrt{\frac{2}{n}} & \text{hvis } i \neq 0 \end{cases}$

Det er værd at bemærke, at indgangene i matricen går fra  $(0, 0)$  til  $(n - 1, n - 1)$  i en  $n \times n$  matrix. Ydermere er  $\alpha(i)$  og  $\alpha(j)$  normaliseringsfaktorer [Rao og Yip [1990], s. 96].

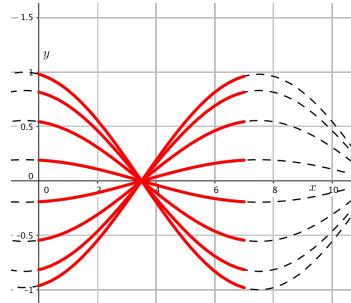
Ligning 3.1 bruges i denne opgave ikke til at beregne de transformerede koefficenter, da det beregningsmæssigt ikke er effektivt. Til dette bruges i stedet cosinustransformationen på matrixform, hvilket uddybes senere i dette afsnit. Ligning 3.1 er imidlertid god til at illustrere principippet bag transformationen, da det er tydeligt, hvordan cosinusfunktioner indgår i transformationen. Dette undersøges nu.

Det ses af ligning 3.1, at hver transformeret indgang er summen af produktet af de 64 indgange i  $A$  og en koefficient fra en cosinusfunktion. For hver række  $f(x, y)$  holdes  $x$  konstant gennem de otte søjler, hvor  $y$  går fra 0 – 7. Dette giver koefficenter, der alle

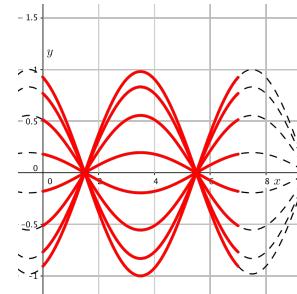
ligger på en cosinuskurve. Hver gang en ny række  $x$  påbegyndes, ændres amplituden af cosinusbølgen, som koefficienterne ligger på.

Når en ny transformert indgang  $T_{(i, j)}$  beregnes, ændres  $i$  og/eller  $j$ , og frekvensen af cosinusfunktionerne i transformen bliver højere. Altså bliver hver transformert indgang et unikt aftryk af forskelligt svingene cosinusfunktioner og værdierne i matricen til billedet  $A$ .

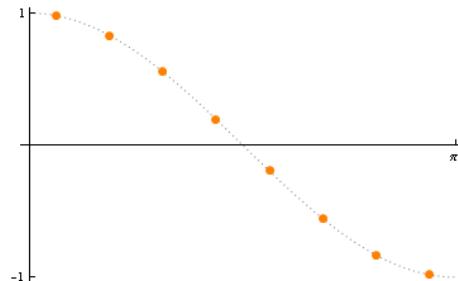
For at illustrere principippet bag de skiftende amplituder og frekvenser af cosinusbølgerne som koefficienterne ligger på, er et sæt af to grafer for cosinustransformationen illustreret i figurerne 3.2 og 3.3. Disse er tegnet ved brug af transformationen i ligning 3.1 på modstående side for henholdsvis  $T(1, 1)$  og  $T(1, 2)$  og for  $n = 8$ . For hver af funktionerne fremkommer otte bølger, og på hver af disse ligger de otte koefficienter ligeligt fordelt. På figur 3.4 ses de otte punkter på én af bølgerne for  $T(0, 0)$  [Society for Industrial and Applied Mathematics [b], afsnit 3].



**Figur 3.2.** Bølger for hver række for  $T(1, 1)$ .



**Figur 3.3.** Bølger for hver række for  $T(1, 2)$ .



**Figur 3.4.** Koefficienterne for  $T(1, 1)$  og  $f(0, y)$  afmærket som punkter på en cosinusbølge.

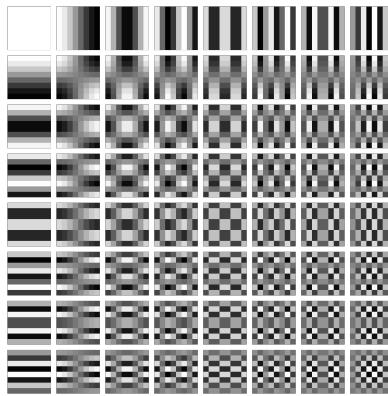
Det ses desuden på figur 3.4, at koefficienterne tilsammen har sum lig nul. Der er altså lige mange positive og negative koefficienter i alle bølgerne med undtagelse af den første, som udelukkende består af koefficienten  $\frac{\sqrt{2}}{2}$ .

At hele den første række kun indeholder positive indgange bidrager til energikomprimeringen. Hvis  $A$  indeholder lignende værdier, vil indgang  $(0, 0)$  have stor numerisk værdi efter transformationen. Derfor kan en stor del af informationerne koncentreres i denne ene indgang. På samme måde vil resultatet af at prikke en vektor med ens indgange,  $\vec{v} = [v_1 \ v_1 \ \dots \ v_1]$  med en hvilken som helst samling af koefficenter fra en cosinusbølge beregnet ved  $i = 0, \dots, 7$  blive lig nul. Resultatet af at prikke en vektor, med indgange der næsten er ens med koefficienterne, vil blive tæt på eller lig nul [Society for Industrial and Applied Mathematics [b], afsnit 4]

Koefficienten i indgang  $(0, 0)$  omtales som DC koefficienten eller Direct Current. De

resterende 63 koefficienter kaldes for AC koefficienter eller Alternating Current. Navnene stammer fra transformationens historiske brug i analyse af elektriske kredsløb og refererer til basisfunktionerne i transformationen, som for DC koefficienten er konstant som jævnstrøm, men de oscillerer som vekselstrøm for de resterende koefficienter [Lui [b], s. 5].

De transformerede koefficienter udtrykker bestemte mønstre i signalet. Mønstrene for hver koefficient ses i figur 3.5, hvor bølgetoppene og -dalene i cosinusfunktionerne (og dermed negative og positive koefficienter) er henholdsvis sorte og hvide. Denne figur illustrerer hvordan transformationen behandler forskellige slags mønstre i signalet - mønstre som ligner signalet vil have høje koefficienter. Bemærk at DC koefficienten er en koefficient for et ensfarvet mønster eller signal. Indgangene i den transformerede matrix er koefficienterne,



**Figur 3.5.** Repræsentation af de 64 forskellige basisfunktioner i cosinustransformationen af længde  $n = 8$ . Mønstrene hører til koefficienterne, som fås ved transformation af et signal. Koefficienternes størrelse fortæller hvor meget, det tilhørende mønster optræder i signalet.

hvorved disse mønstre er repræsenteret i signalet. Altså kan signalet genskabes ved en lineær kombination af disse mønstre.

I dette projekt udføres beregninger og dermed også transformationen i Python. Til dette er det beregningsmæssigt nemmere og hurtigere at regne med DCT på matrixform.

I udledningen af DCT i appendiks A på side 67 ses formlen for den todimensionelle DCT, hvor ligning A.18 på side 71 viser, hvordan hver enkelt søjle i en matrix for en n-punkts DCT beregnes. Da der i denne opgave regnes med matricer af størrelsen  $8 \times 8$ , vises der i ligning 3.2 en matrix beregnet ved  $n = 8$  - altså en  $8 \times 8$ -matrix.

$$U = \frac{1}{2} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \cos\left(\frac{\pi}{16}\right) & \cos\left(\frac{3\pi}{16}\right) & \cos\left(\frac{5\pi}{16}\right) & \cos\left(\frac{7\pi}{16}\right) & \cos\left(\frac{9\pi}{16}\right) & \cos\left(\frac{11\pi}{16}\right) & \cos\left(\frac{13\pi}{16}\right) & \cos\left(\frac{15\pi}{16}\right) \\ \cos\left(\frac{2\pi}{16}\right) & \cos\left(\frac{6\pi}{16}\right) & \cos\left(\frac{10\pi}{16}\right) & \cos\left(\frac{14\pi}{16}\right) & \cos\left(\frac{18\pi}{16}\right) & \cos\left(\frac{22\pi}{16}\right) & \cos\left(\frac{26\pi}{16}\right) & \cos\left(\frac{30\pi}{16}\right) \\ \cos\left(\frac{3\pi}{16}\right) & \cos\left(\frac{9\pi}{16}\right) & \cos\left(\frac{15\pi}{16}\right) & \cos\left(\frac{21\pi}{16}\right) & \cos\left(\frac{27\pi}{16}\right) & \cos\left(\frac{33\pi}{16}\right) & \cos\left(\frac{39\pi}{16}\right) & \cos\left(\frac{45\pi}{16}\right) \\ \cos\left(\frac{4\pi}{16}\right) & \cos\left(\frac{12\pi}{16}\right) & \cos\left(\frac{20\pi}{16}\right) & \cos\left(\frac{28\pi}{16}\right) & \cos\left(\frac{36\pi}{16}\right) & \cos\left(\frac{44\pi}{16}\right) & \cos\left(\frac{52\pi}{16}\right) & \cos\left(\frac{60\pi}{16}\right) \\ \cos\left(\frac{5\pi}{16}\right) & \cos\left(\frac{15\pi}{16}\right) & \cos\left(\frac{25\pi}{16}\right) & \cos\left(\frac{35\pi}{16}\right) & \cos\left(\frac{45\pi}{16}\right) & \cos\left(\frac{55\pi}{16}\right) & \cos\left(\frac{65\pi}{16}\right) & \cos\left(\frac{75\pi}{16}\right) \\ \cos\left(\frac{6\pi}{16}\right) & \cos\left(\frac{18\pi}{16}\right) & \cos\left(\frac{30\pi}{16}\right) & \cos\left(\frac{42\pi}{16}\right) & \cos\left(\frac{54\pi}{16}\right) & \cos\left(\frac{66\pi}{16}\right) & \cos\left(\frac{78\pi}{16}\right) & \cos\left(\frac{90\pi}{16}\right) \\ \cos\left(\frac{7\pi}{16}\right) & \cos\left(\frac{21\pi}{16}\right) & \cos\left(\frac{35\pi}{16}\right) & \cos\left(\frac{49\pi}{16}\right) & \cos\left(\frac{63\pi}{16}\right) & \cos\left(\frac{77\pi}{16}\right) & \cos\left(\frac{91\pi}{16}\right) & \cos\left(\frac{105\pi}{16}\right) \end{bmatrix} \quad (3.2)$$

Magen til ligning 3.1 på side 12 kan  $U$  bruges til at opnå DCT transformationen, blot hvor en stor del af udregningerne er udført på forhånd. På matrixform prikkes  $U$  med  $A$ , og

derefter prikkes dette resultat med  $U^T$ . Transformationen ses i ligning 3.3.

$$B = U \cdot A \cdot U^T \quad (3.3)$$

Hvor

- $B$ : den transformerede matrix
- $U$ : DCT-matricen
- $A$ :  $8 \times 8$  matrix af pixelværdier

Denne similærtransformation (similaritet forklares senere på side 16) har samme effekt på  $8 \times 8$ -matricen som ligning 3.1 på side 12. Ligning 3.3 kan skrives som en sammensat transformation bestående af to transformationer med henholdsvis  $U$  og  $U^T$  som standardmatricer. De to transformationer ses i ligning 3.4 og 3.5.

$$T(A) = U \cdot A \quad (3.4)$$

$$Q(A) = A \cdot U^T \quad (3.5)$$

Den sammensatte transformation ser derfor ud som i ligning 3.6.

$$Q(T(A)) = U \cdot A \cdot U^T = B \quad (3.6)$$

Fremover refereres der til den transformerede matrix som  $B$ .

## Ortonormalitet

$U$  er en ortonormal matrix, hvilket ses i ligning A.15 på side 71, hvilket er et specialtilfælde af en ortogonal matrix. For ortogonale matricer gælder, at

- alle ortogonale sæt af vektorer som ikke er nul er lineært uafhængige.
- vektorerne er ortogonale på hinanden.
- alle underrum i  $\mathbb{R}^n$  indeholder en ortogonal basis.
- $Z^{-1} = Z^T$  [Spence et al., s. 413].

En ortonormal matrix er en ortogonal matrix, hvor der gælder, at søjlerne er vektorer med længden 1 [Spence et al., s. 374].

## Bijektivitet og invertibilitet

Den diskrete cosinustransformation skal være invertibel, da det ellers ikke giver mening at transformere data til et kodomæne, hvorfra det ikke kan lade sig gøre at komme tilbage til domænet.

En lineær transformation  $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$  med standardmatrix  $A$  er invertibel, hvis og kun hvis  $A$  er invertibel. I så fald er den inverse transformation  $T^{-1}$  og dens standardmatrix er  $A^{-1}$  [Spence et al., s. 187].

En lineær transformation er kun invertibel, hvis den er bijektiv [Spence et al., s. 555].

En lineær transformation  $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$  siges at være bijektiv, hvis den er både surjektiv og injektiv og derved bijektiv [Spence et al., s. 555].

Altså ønskes det at vise, at den diskrete cosinustransformation er bijektiv.

### **Surjektivitet**

En lineær transformation  $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$  siges at være surjektiv, hvis transformationens billedmængde er lig kodomænet  $\mathbb{R}^m$  [Spence et al., s. 180].

For en surjektiv lineær transformation  $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$  med standardmatrix  $A$  gælder, at

$$\text{Rank}(A) = m \tag{3.7}$$

$$\text{Col}(A) = \mathbb{R}^m \tag{3.8}$$

[Spence et al., s. 181]

Da matricen  $U$ , som er standardmatrix i cosinustransformationen, opfylder de to ovenstående krav på baggrund af, at den er ortonormal og kvadratisk, kan der sluttes, at den lineære transformation med standardmatrix  $U$  er surjektiv.

### **Injektivitet**

En lineær transformation  $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$  siges at være injektiv, hvis hvert par af særskilte vektorer i domænet  $\mathbb{R}^n$  har særskilte billede i kodomænet  $\mathbb{R}^m$  [Spence et al., s. 182].

For en injektiv lineær transformation  $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$  med standardmatrix  $A$  gælder, at

$$\text{Null}(A) = 0 \tag{3.9}$$

$$\text{Rank}(A) = n \tag{3.10}$$

[Spence et al., s. 184]

Da matricen  $U$  opfylder de to ovenstående krav på baggrund af, at den er ortonormal og kvadratisk, kan der sluttes, at den lineære transformation med standardmatrix  $U$  er injektiv.

Da den lineære transformation med standardmatrix  $U$  er både injektiv og surjektiv, er den altså bijektiv. Disse egenskaber gør, at transformationen kan inverteres, som vist i ligning 3.13.

### **Similaritet**

Som tidligere nævnt er udtryk 3.3 en similærtransformation, og defintionen for similaritet lyder som følgende: To kvadratiske matricer,  $A$  og  $B$ , siges at være similære, hvis der findes en invertibel matrix,  $P$ , som opfylder

$$B = P^{-1} \cdot A \cdot P \tag{3.11}$$

Da der gælder at  $U^T = U^{-1}$  [Spence et al., s. 413] sluttes, at ligningerne 3.12 og 3.13 gælder.

$$B = U \cdot A \cdot U^{-1} = U \cdot A \cdot U^T \quad (3.12)$$

$$A = U^{-1} \cdot B \cdot U = U^T \cdot B \cdot U \quad (3.13)$$

Matricerne  $A$  og  $B$  er altså similære matricer. Transformationen er dermed en similartransformation [Weisstein]. Den diskrete cosinus transformation ses også som værende invertibel ved

$$B = U \cdot A \cdot U^T \quad (3.14)$$

Da  $A$  og  $B$  er similære, er transformationen et basisskift da følgende ligning gælder.

$$B = P^{-1} \cdot A \cdot P \quad (3.15)$$

Hvor

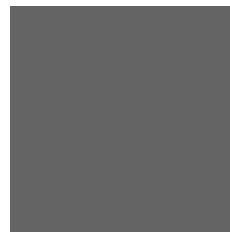
- $A$ :  $n \times n$ -matrix
- $B$ :  $n \times n$ -matrix
- $P$ :  $U^T$

$B$  er altså  $A$  udtrykt i basen  $U^T$ . Basisskiftet gør det nemmere at foretage den tidlige omtalte udvælgelse af vigtige informationer

Hermed er der redejort for DCT og dens egenskaber, og der bliver i næste sektion vist eksempler på transformation af diverse  $8 \times 8$ -matricer, før den videre algoritme forklares.

### 3.1.3 Eksempler på brug af den diskrete cosinustransformation

For at illustrere hvordan den diskrete cosinustransformation fungerer, vises her eksempler på  $8 \times 8$  matricer transformert ved den diskrete cosinustransformation. Da billeder er det, der undersøges i dette projekt, vises de respektive  $8 \times 8$  billeder tilhørende matricerne også. Der er trukket 128 fra alle indgange inden transformationen, hvilket bl.a. forsimpler processen med kvantisering - dette uddybes i afsnit 3.1.4. Alle koefficienter er desuden afrundet til nærmeste heltal efter transformationen, for at gøre matricerne letlæselige. Der gøres opmærksom på, at der kan være koefficienter, som bliver afrundet til nul. Alle koefficienter, som er nul, repræsenteres af ". ".



**Figur 3.6.**  $8 \times 8$  billede som består udelukkende af pixels med intensitet 100.

100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100

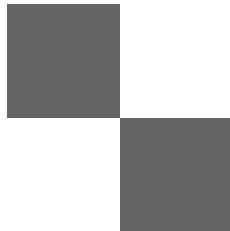
**Figur 3.7.** Pixelværdier for figur 3.6.

-224	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

**Figur 3.8.** Figur 3.7 transformert.

I figur 3.7 har alle indgange en værdi på 100, hvilket svarer til et billede bestående af 64 pixels i samme gråtone med farveintensitet på 100. I figur 3.6 ses matricen repræsenteret ved et billede, og i figur 3.8 ses den transformerede matrix. Det ses at alle koefficienter er lig nul med undtagelse af DC koefficienten, som er lig -224. Altså kan hele det behandlede signal udtrykkes ved indgang (0,0), da det kun er den første basisfunktion, som bruges til at repræsentere billedet.

I figur 3.10 udtrykkes matricen for et billede, som består af 64 pixels, hvoraf halvdelen er i gråtoner med farveintensitet = 100, og den anden halvdel er hvide med farveintensitet = 255. Det ses i 3.11, at koefficienterne er anderledes end i 3.8 og udtrykker signalet ved andre koefficienter. Denne gang benyttes flere AC koefficienter for at udtrykke signalet, da der er mere variation i signalet. Matricen i figur 3.11 viser, at når et billede har store og

**Figur 3.9.**  $8 \times 8$  billede bestående af grå og hvide pixels med farveintensiteter på henholdsvis 100 og 255.

100	100	100	100	255	255	255	255
100	100	100	100	255	255	255	255
100	100	100	100	255	255	255	255
100	100	100	100	255	255	255	255
255	255	255	255	100	100	100	100
255	255	255	255	100	100	100	100
255	255	255	255	100	100	100	100
255	255	255	255	100	100	100	100

**Figur 3.10.** Pixelværdier for figur 3.9.

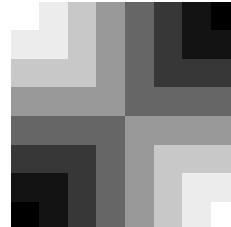
396	.	.	.	.	.	.	.
.	-509	.	179	.	119	.	101
.	.	.	.	.	.	.	.
.	179	.	-63	.	42	.	-36
.	.	.	.	.	.	.	.
.	-119	.	42	.	-28	.	24
.	.	.	.	.	.	.	.
.	101	.	-36	.	24	.	-20

**Figur 3.11.** Figur 3.10 transformert.

bratte ændringer mellem farveintensiteter, skal der bruges mange cosinusfunktioner til at udtrykke det bratte skift i farveintensitet. Dette er de tilfælde, som cosinusfunktionen ikke behandler godt - der opnås ikke god energikomprimering.

I figur 3.13 ses en matrix, som laver en glat overgang fra hvid til sort i samme mønster som den i figur 3.10. Denne glatte overgang skulle gerne være lettere for cosinusfunktionen at

energikomprimere, da der ikke sker bratte skift i farveintensitet. Det tilsvarende billede ses i figur 3.12 og den transformerede matrix ses i figur 3.14. Af den transformerede matrix ses det, at cosinustransformationen har haft letttere ved at energikomprimere denne matrix end den i figur 3.10. Det antages, at billedet primært har glatte overgange mellem farveintensiteter, som ligner den i figur 3.12. Den inverse transformation producerer de to



**Figur 3.12.**  $8 \times 8$  billede bestående af pixels med gradvis overgang fra hvid til sort.

$$\begin{bmatrix} 255 & 236 & 200 & 153 & 102 & 55 & 19 & 0 \\ 236 & 236 & 200 & 153 & 102 & 55 & 19 & 19 \\ 200 & 200 & 200 & 153 & 102 & 55 & 55 & 55 \\ 153 & 153 & 153 & 153 & 102 & 102 & 102 & 102 \\ 102 & 102 & 102 & 102 & 153 & 153 & 153 & 153 \\ 55 & 55 & 55 & 102 & 153 & 200 & 200 & 200 \\ 19 & 19 & 55 & 102 & 153 & 200 & 236 & 236 \\ 0 & 19 & 55 & 102 & 153 & 200 & 236 & 255 \end{bmatrix} \quad \begin{bmatrix} -4 & . & . & . & . & . & . & . \\ . & 568 & . & -40 & . & -4 & . & -1 \\ . & . & . & . & . & . & . & . \\ . & -40 & . & 54 & . & -11 & . & -1 \\ . & . & . & . & . & . & . & . \\ . & -4 & . & -11 & . & 24 & . & -5 \\ . & . & . & . & . & . & . & . \\ . & -1 & . & -1 & . & -5 & . & 22 \end{bmatrix}$$

**Figur 3.13.** Pixelværdier til figur 3.12.

**Figur 3.14.** Figur 3.13 transformert.

originale matricer uden afvigelser. Skridtet er inverterbart og uden tab af informationer.

Opsummeres ovenstående afsnit om DCT er de vigtigste pointer at

- ... DCT transformerer fra tidsdomænet til frekvensdomænet
- ... DCT på matrixform er udtrykt ved ligningen  $B = U \cdot A \cdot U^T$ , hvor  $U$  er udtrykt ved ligning 3.3 på side 15
- ... den inverse DCT er udtrykt som  $A = U^T \cdot B \cdot U$

Med DCT-transformationens egenskaber i mente, ses der nu på de resterende elementer i JPEG-komprimeringen.

### 3.1.4 Forbehandling af billede

Før billedet transformeres med den diskrete cosinustransformation, skal det forbehandles. Dette indebærer at dele billedet op i undermatricer af  $8 \times 8$  pixels - kvadratiske matricer med 64 indgange. Størrelsen af disse undermatricer af pixels er ikke tilfældig, og har udgangspunkt i egenskaberne ved den diskrete cosinustransformation. Som forklaret i afsnit 2 på side 5 antages det, at nærliggende pixels i et billede er korrelerede, og at ikke-nærliggende pixels ikke er korrelerede. Dermed er det formålsløst at forsøge at transformere et helt billede, i håb om at kunne energikomprimere dette. Derfor deles billedet op i mindre undermatricer, og disse transformeres enkeltvis. Den optimale størrelse af disse undermatricer er  $8 \times 8$  pixels, hvilket der er flere grunde til.

Matricer af  $2 \times 2$  pixels indeholder ikke nok data for DCT at bearbejde med, og når transformationen skal udtrykke matricen som funktioner af cosinusbølger, er der ikke nok cosinusbølger til at kunne udtrykke matricen til billedet præcist - der er nemlig kun fire. Resultatet er, at datarepræsentationen er upræcist, og at billedet mister for meget kvalitet - det kan ses med det menneskelige øje. Dette skyldes transformationens høje energikomprimering [Sapiro, 19:26-23:52]. Matricer af  $4 \times 4$  lader under tilsvarende problem som  $2 \times 2$  dog i mindre grad.

Matricer af  $8 \times 8$  pixels har en størrelse, der gør cosinustransformationen effektiv. Der er nok indgange og dermed cosinusfunktioner til at kunne udtrykke matricen præcist. Desuden kræver energikomprimeringen, at flere pixels kan repræsenteres af enkelte værdier, og det er derfor vigtigt, at de nærliggende pixels ligner hinanden. Hvis dette er tilfældet, hvilket det ofte er i  $8 \times 8$  matricer, kan matricen komprimeres. Altså er der nok cosinusfunktioner og nok korrelation mellem de enkelte pixels til, at  $8 \times 8$  er en fornuftig størrelse [Sapiro, 19:26-23:52].

Årsagerne til ikke at bruge større end  $8 \times 8$ , som eksempelvis  $16 \times 16$ , er flere; det er beregningsmæssigt langt mere effektivt at udføre komprimeringen på mindre matricer - en computer skal bruge færre beregninger, komprimeringsalgoritmen fungerer ud fra princippet om, at nærliggende pixels ligner hinanden, hvilket er mere usandsynligt ved  $16 \times 16$  end  $8 \times 8$ . Cosinustransformationens energikomprimering fungerer bedst, når der er korrelation mellem de pixels, som transformeres - så kan flere pixels udtrykkes ved få værdier i frekvensdomænet. Problemet med at bruge større matricer end  $8 \times 8$  er, at der ikke nødvendigvis er nogen korrelation mellem de enkelte pixels i en stor matrix. Det bliver meget mere sandsynligt, at billedet tydeligt skifter farve over et stort område, og det giver derfor ikke mening at prøve at sammenligne disse pixels, da der sandsynligvis ikke er nogen korrelation mellem dem [Sapiro, 21:16-21:52].

Altså transformeres kun matricer af størrelse  $8 \times 8$  af gangen. Disse kvadratiske matricer af billedet, kaldes som tidligere nævnt  $A$ . Figur 2.3 på side 6 viser et eksempel på en  $8 \times 8$  pixelmatrix. Bemærk at denne matrix kun består af pixels i gråtoner.

Billeder i gråtoner kan deles op som i figur 2.3 på side 6, og billeder i RGB-farveformatet beskrives med én værdi for hver farve i hver pixel - altså tre værdier. Billeder i farver deles op i de tre farverum, og de enkelte farverum bearbejdes på samme måde som ved gråtoner. Efterfølgende sammensættes de tre farverum igen for at danne billedet. Hvert farverum opdeles i  $A$ -matricer, og der subtraheres 128 fra samtlige indgange. Dette gøres for at ændre værdierne fra beliggende i intervallet  $[0; 255]$  til  $[-128; 127]$  og dermed centrere dem omkring nul. Dette er ønskværdigt for komprimeringen, da kvantiseringsskridtet dermed efter cosinustransformationen formår at skabe flere ens koefficienter, hvilket gør komprimeringen mere effektiv, men dette uddybes senere.

Såfremt billedets dimensioner (både højde og bredde) ikke går op i otte, fyldes de ufuldendte  $8 \times 8$  matricer ud med nuller. Dette hedder nulfyldning [Marcus, s. 1]. Nulfyldning er nødvendigt, da DCT i denne rapport bruges på matricer af størrelse  $8 \times 8$ , og billedet vil ikke altid have dimensioner, som 8 ikke går op i. Dette resulterer dog i, at der indsættes en grå kant på billedet, som kan give en kontrast til det originale billede, hvilket betyder, at der formentlig opstår større unøjagtigheder i kanten af billedet. Dette har indflydelse på transformationen og dermed hele komprimeringsalgoritmen. Der nøjes

dog med at påpege problemet, men ikke lave en bedre løsning end nulfyldning. Det er i det hele taget et grundlæggende problem ved DCT, at transformationen ikke behandler store skift i farveintensitet godt, da den forsøger at udtrykke glatte overgange. Hvis der ikke er glatte overgange, fungerer energikomprimeringen ringe.

Efter forbehandlingen fås en samling af undermatricer bestående af 64 pixels hver, og som tilsammen udgør det komplette billede. Dermed er billedet klar til cosinustransformationen fra forrige sektion.

Ønskes det at invertere forbehandlingsskridtet, skal der indgangsvist adderes med 128 på alle indgange, og det opsplittede billede skal samles fra  $8 \times 8$  matricerne til det samlede billede igen. Forbehandlingsskridtet er herved inverteret, hvilket gøres uden tab eller forandringer i dataene.

### 3.1.5 Kvantisering

Tredje skridt i komprimeringsalgoritmen omhandler kvantisering af de transformerede værdier. Kvantiseringen har til formål at smide de overflødige data væk og desuden gøre billedet klar til fjerde og sidste skridt. Med overflødig data menes der informationer om billedet, som ikke har nogen (eller lille) synlig indflydelse på billedkvaliteten. Da DCT-transformationen har energikomprimeret  $A$ , repræsenterer de mindste koefficienter de data, der fremkommer mindst i matricen. Da det menneskelige øje har svært ved at se meget hurtige ændringer i farveintensitet over små afstande, er høje frekvensændringer ikke tydelige for det menneskelige øje. Altså er koefficienterne, i nedre højre hjørne af  $B$ , ikke betydningsfulde for opfattelsen af billedet som helhed.

Kvantiseringen gør brug af indgangsvis division med en  $8 \times 8$  matrix (indgang  $B_{1,1}$  divideres med indgang  $Q_{1,1}$  osv.). Denne matrix består af heltal, som er bestemt empirisk ved eksperimenter omhandlende det menneskelige syn. Tallene i matricen er tilpasset således, at den synlige billedkvalitet er høj, mens billede komprimeres mest muligt [Lui [a], s. 4]. Der findes flere forskellige kvantiseringssmatricer, og den der bruges i dette projekt, er den som bruges i JPEG-standarden [Lui [a], s. 4]. Kvantiseringssmatricen, som fremover refereres til som  $Q$ , ses i ligning 3.16.

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 50 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (3.16)$$

Matricen i ligning 3.16 hedder  $Q_{50}$ , fordi den er kvantiseringssmatricen for en komprimering med kvalitet 50 (fremover benævnes Q-værdien blot som eksempelvis  $Q_{50}$  eller  $Q_{10}$  ved de respektive kvaliteter). I JPEG kan komprimeringsgraden justeres, og det er med denne matrix dette gøres. Matricen kan have en kvalitet mellem 1 og 100, hvor 100 resulterer

i laveste komprimering og højeste billedkvalitet, mens 1 omvendt resulterer i højeste komprimering og laveste billedkvalitet. Det er værd at bemærke at  $Q50$  ikke komprimerer et billede med 50% eller 50 gange - værdien er blot et udtryk for en position på en skala fra 1 til 100.

For at opnå en anden kvalitet end 50, ændres  $Q$  med en bestemt formel. Alt efter den ønskede kvalitet gælder to formler, som begge multiplicerer hver enkelt indgang med en konstant, der beregnes på baggrund af den ønskede kvalitet.

$$Q_n = \begin{cases} \frac{100-n}{50} \cdot Q50 & \text{hvis } n > 50 \\ \frac{50}{n} \cdot Q50 & \text{hvis } n < 50 \end{cases} \quad (3.17)$$

Herunder ses en kvantiseringsmatrix til en kvalitet på 25.

$$Q25 = \begin{bmatrix} 32 & 22 & 20 & 32 & 48 & 80 & 102 & 122 \\ 24 & 24 & 28 & 38 & 52 & 116 & 120 & 100 \\ 28 & 26 & 32 & 48 & 80 & 114 & 138 & 112 \\ 28 & 34 & 44 & 58 & 102 & 174 & 160 & 124 \\ 36 & 44 & 74 & 112 & 136 & 218 & 206 & 134 \\ 48 & 70 & 110 & 128 & 162 & 208 & 226 & 184 \\ 98 & 128 & 156 & 174 & 206 & 242 & 240 & 202 \\ 144 & 184 & 190 & 196 & 224 & 200 & 206 & 198 \end{bmatrix} \quad (3.18)$$

Matricen  $B$  kvantiseres ved indgangsvis division med  $Q$ .

$$C_{(i, j)} = \frac{B_{(i, j)}}{Q50_{(i, j)}} \quad (3.19)$$

Indgangene i  $B$  afrundes til nærmeste heltal, og matricen er dermed kvantiseret. Den kvantiserede matrix navngives  $C$ . Afrundingen gør at komprimeringen mister data, hvilket er et uigenkaldeligt tab af data og kan altså ikke gøres om. Når den komprimerede fil dekomprimeres til et billede, kan det originale billede ikke genskabes, da det inverterede kvantiseringsskridt vil resultere i andre værdier end det originale input, da afrundingen "sletter" dataene. Brugen af en ikke-tabsfrei komprimering retfærdiggøres ved at de data, som smides væk, ikke har stor betydning for billedeets generelle udseende - det menneskelige øje opfatter ikke tydeligt de manglende værdier, når billedet igen dekomprimeres. Desuden gør dette skridt den senere tabsfrei komprimering med Huffman væsentligt mere effektiv.

Det ses på  $Q$ , at indgangene i nederste højre hjørne er væsentligt højere end indgangene i øverste venstre hjørne. Dette resulterer i, at værdier i nederste højre hjørne af  $B$  bliver divideret med større tal end i øvre venstre hjørne. Dette betyder imidlertid, at der er større chance for, at tal i nedre højre hjørne bliver afrundet til heltal tæt på eller lig nul, og at værdier i øverste venstre hjørne vil forblive høje i forhold til de øvrige værdier. De mange værdier tæt på eller lig nul skyldes den tidligere centrering omkring nul, da der blev trukket 128 fra alle indgange i  $A$ . Altså vil  $C$  hovedsageligt bestå af nuller og nogle få indgange i øverste venstre hjørne, som ikke er nul.

Det ses også, at en højere komprimeringsgrad resulterer i højere heltal i  $Q$ . Dette gør, at  $C$  vil bestå af endnu lavere værdier, og flere vil blive afrundet til nul. Altså vil en endnu større mængde data gå tabt, og billedkvaliteten vil tilsvarende falde. Modsat vil en lavere

komprimering resultere i lavere heltal i  $Q$  og højere tal i  $C$  med færre værdier afrundet til nul. Filen vil i dette tilfælde ende med at fyldes mere, men med højere billedkvalitet.

Når dette skridt skal inverteres, ganges  $C$  indgangsvis med  $Q$ -matricen, som blev brugt under komprimeringen.

$$B_{(i, j)} = C_{(i, j)} \cdot Q_{(i, j)} \quad (3.20)$$

Det er nødvendigt at matricen  $Q$ , som bruges til inverteringen af skridtet, er magen til den som blev brugt under kvantiseringen. Hvis dette ikke gøres korrekt, vil det lede til værdier i  $B$ , som ikke ligner de originale, da der ganges med andre tal, end der oprindeligt blev divideret med.

I ligning 3.21 ses et eksempel på en kvantiseret matrix. Matricen som kvantiseres er den fra figur 3.7 på side 18, og den kvantiseres med  $Q50$ .

$$C = \begin{bmatrix} 2 & . & . & . & . & . & . & . \\ . & 4 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix} \quad (3.21)$$

Her ses det, at mange af koefficienterne er blevet afrundet til nul, og de som ikke er afrundet til nul, er blevet små tal, hvorved matricen er blevet kvantiseret.

### 3.1.6 Entropikodning - Huffmankodning

Entropikodning som emne vil ikke blive uddybet her, men blot anses som en metode til at definere sandsynligheden af et givent datasæts udkom og repræsentere dette på bedste vis. Der findes flere forskellige entropikodnings-metoder, bl.a. Zigzag- og Huffmankodning. I denne rapport undersøges Huffmankodning. Enkelte grundelementer i grafteori vil dog blive præsenteret, da disse er nødvendige for at forstå Huffmankodning.

Huffmankodning fungerer vha. grafteori, og de nødvendige begreber er [Rosenkilde [2014], s. 1-2]:

- En *knude* er en samling af kanter
- En *kant* er en forbindelse mellem knuder
- *Valens* er antallet af kanter, der støder op til en knude
- Et *blad* er et knude med valens én
- En *graf* består af knuder og kanter
- Et *træ* er en sammenhængende graf
- Et *undertræ* er et afsnit af træet
- *Sti* er en forbindelse mellem to knuder, hvor alle knuder på stien er forskellige

Huffmankodning er en slags entropikodning og er en tabsfri komprimeringsmetode, hvorpå en stor mængde data kan repræsenteres ved hjælp af de enkelte symbolers (i dette tilfælde tallene 0 – 255) sandsynlighed for at fremkomme. Komprimering med Huffmankodning tildeler alle symboler, der fremkommer i datastrengen, en bitrepræsentation af variabel længde, der afhænger af symbolets sandsynlighed. Dette betyder at et symbol, der fremkommer mange gange vil være tildelt et kortere kodeord end et symbol, der fremkommer få gange. Herved vil mængden af bits brugt til at repræsentere en datastreng blive nedbragt, da symboler der fremkommer mange gange fylder mindre, end dem der fremkommer få gange. Et kodeord er en binær repræsentation af symbolet og kunne eksempelvis være

$$\text{"A : 0, B : 10, C : 110, D : 111"} \quad (3.22)$$

Vigtigt at nævne er, at Huffmankodning er en præfix-fri kode, hvilket betyder, at symboler ikke kan beskrives som sammensætning af andre symboler. Havde eksempel 3.22 været ”A : 0, B : 1, C : 10, D : 11”, ville en streng af 0 og 1 ikke kunne afkodes uden, at kodeordene var synligt adskilt, da det ikke vil være muligt, at se om 10 betyder B, A eller C. Det er derfor vigtigt i Huffmankodning, at der ikke er nogle præfix for symbolerne, men at koden kan interpreteres entydigt.

## Huffmantræ

Huffmankodning foregår vha. skabelsen af et Huffmantræ, som er et overblik over sandsynligheden for, at de enkelte symboler fremkommer, og hvilket kodeord de skal tildeles. Kodeordet er stien ned til bladet. For at forklare skabelsen af et Huffmantræ laves et eksempel bestående af fire symboler, der viser principperne i skabelsen af træet. Lad os antage at datastrengen lyder

*abbacdababaccaddabbbacaadabaaddaaccaaaadaadaabaadacaadabaaadacaaadabaa*

så kan sandsynligheden for at de enkelte symboler fremkomst i datastrengen beregnes. Disse er angivet i tabel 3.15.

Ud fra tabel 3.15 på modstående side opstilles der fire blade med hvert deres symbol som indgang, se figur B.1 på side 73. Herefter opstilles de to blade med den mindste sandsynlighed i et undertræ med bladene som indgange, se figur B.2 på side 73. I dette eksempel vil det være bladene for b og c, hvor bladet med mindst sandsynlighed placeres yderst til højre. Hyppigheden af dette undertræ er summen af de to blades sandsynlighed og giver her  $0,11 + 0,16 = 0,27$ . Derudover tildeles kanterne hhv. et nul- og et-tal, som senere bruges til definering af symbolets kodeord. Herefter kigges der på de to blade/undertræer, der har den mindste sandsynlighed, hvilket er undertræet for b,c (samlet sandsynlighed: 0,27) og d (sandsynlighed: 0,17). Disse samles i et undertræ, hvor mindste sandsynlighed igen placeres yderst til højre og undertræets samlede sandsynlighed er summen af bladenes sandsynlighed ( $0,27 + 0,17 = 0,44$ ), se figur B.3 på side 73. Samme fremgangsmåde gentages for de sidste to undertræer/blade, og derved skabes det totale og færdige Huffmantræ, se figur B.4 på side 73.

På baggrund af træet på figur B.4 på side 73 kan symbolernes kodeord defineres som værende stien til de respektive blade, se tabel 3.16. Det er her tydeligt, at symbolerne med størst frekvens er tildelt de korteste binære repræsentationer, hvorved at de pr. symbol vil fylde mindre. Dette er ønskværdigt, når formålet er at bringe filstørrelsen ned.

Hele denne proces kan på samme vis benyttes til at komprimere de kvantiserede værdier i  $C$ -matricerne fra afsnit 3.1.5. I disse transformerede og kvantiserede matricer ligger energikomprimerede informationer om det originale billede, således at mange indgange er lig nul. Altså er der mange fremkomster af nul, og disse kan udtrykkes ved en kort binær repræsentation.

Når alle værdier er parret med en kodestreng, gemmes disse parringer sammen med en "ordbog" (hvilke symboler og kodeord hører sammen) og  $Q$ . Dermed kan en codec, ved åbning af den komprimerede fil, udpakke og dekomprimere filen til et dekomprimeret billede, som er en efterligning af det originale billede.

Altså er Huffmankodning det mest komprimerende skridt i DCT-algoritmen - de tidlige skridt gør det muligt ved at sørge for, at billedet beskrives med få værdier, og at de resterende værdier er ens. Dette gør Huffmankodningen langt mere effektiv.

## 3.2 DCT ANVENDELSE

I dette afsnit undersøges den praktiske anvendelse af DCT, jævnfør kapitel 3, hvilket udføres på billedet af Lena, se figur 3.19 på næste side. Billedet har dimensionerne  $512 \times 512$  pixel, hvilket indledningsvist opdeles i  $8 \times 8$  undermatricer. Endvidere er der  $64 \cdot 64$  matricer af  $8 \times 8$ , hvilket giver  $64 \cdot 64 \cdot 8 \cdot 8 = 512 \cdot 512 = 262.144$  pixels totalt.

De følgende regneoperationer bliver kun udført på den røde farvekanal og første  $8 \times 8$  undermatrix for at vise princippet ved brugen af metoden. Ved gentagelse med resten af undermatricerne og de resterende to farvekanaler opnås en komprimering af det fulde farvebillede. Repetitionerne af beregningerne udelades, da dette blot er et eksempel. Derimod vises resultatet af komprimeringen for hele billedet, efter regneeksemplet med undermatricen er udført. For at skabe overblik over komprimeringen ses komprimeringsalgoritmen herunder:

Den specifikke undermatrix i følgende eksempel ligger i øverste venstre hjørne, jævnfør figur 3.19 på den følgende side. Første værdi for hver indgang, dvs. den røde kanal, findes

Symbol	Antal	Sandsynlighed
a	39	$\approx 0,56$
b	11	$\approx 0,16$
c	8	$\approx 0,11$
d	12	$\approx 0,17$
<b>Total</b>	<b>70</b>	$\approx 1$

**Figur 3.15.** Sandsynligheder for de enkelte symboler

Symbol	Kodeord
a	0
b	100
c	101
d	11

**Figur 3.16.** Symbolernes kodeord

**Algoritme: Komprimering vha. DCT**

1. Input:	Billede, $A: m \times n$ pixels
2. Output:	Komprimeret fil
<i>Komprimering</i>	
3. Opdeling:	Billede opdeles i farvekanaler og $8 \times 8$ matricer
4. Centrering omkring 0:	$A = A - 128$ indgangsvist
5. DCT:	$B = U \cdot A \cdot U^T$
6. Kvantisering:	$C_{(i, j)} = \frac{B_{(i, j)}}{Q_{(i, j)}}$ afrundet
7. Entropikodning:	$C$ komprimeres vha. Huffman til en fil
8. Gentagelse:	Ovenstående gentages for samtlige $8 \times 8$ matricer

vha. vores Python-program og følgende undermatrix,  $A_{1,1}$ , fremkommer. Herfra udføres



**Figur 3.17.** Oprindelig - Visuel.

$$A_{1,1} = \begin{bmatrix} 226 & 226 & 223 & 223 & 226 & 226 & 228 & 227 \\ 226 & 226 & 223 & 223 & 226 & 226 & 228 & 227 \\ 226 & 226 & 223 & 223 & 226 & 226 & 228 & 227 \\ 226 & 226 & 223 & 223 & 226 & 226 & 228 & 227 \\ 226 & 226 & 223 & 223 & 226 & 226 & 228 & 227 \\ 227 & 227 & 227 & 222 & 226 & 228 & 226 & 230 \\ 228 & 228 & 225 & 224 & 225 & 229 & 229 & 229 \\ 223 & 223 & 226 & 221 & 227 & 225 & 226 & 228 \end{bmatrix}$$

**Figur 3.18.** Oprindelig - Tal.



**Figur 3.19.** Lena delt op i 4096,  $8 \times 8$  undermatricer.

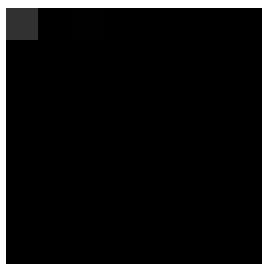
DCT på undermatricen, og  $B$  findes vha.  $B = U \cdot A \cdot U^T$ , hvor  $U$  og  $U^T$  er givet i udtryk 3.2 på side 14. Det ses tydeligt på den visuelle figur 3.20, hvordan alle overflødige høje frekvenser allerede er reduceret kraftigt. Dette er nøjagtigt det, som ønskes af DCT'en, da de lave frekvenser kan ses tydeligere af øjet end de høje. Den visuelle repræsentation er fremkommet ved at trunkere dataene til intervallet  $[0; 255]$ . Næste trin i algoritmen er kvantiseringen, hvor der i dette eksempel tages udgangspunkt i  $Q_{50}$  jævnfør 3.16 på side 21. Den afrundede  $C$  findes vha. 3.19 på side 22. Bemærk at der bruges indgangsvis division. Det ses, jævnfør tallene på figur 3.23, at matricen hovedsagligt udgøres af nuller. Dette er



**Figur 3.20.** DCT - Visuel.

$$B_{1,1} = \begin{bmatrix} 791 & -71 & 8,12 & 4,23 & -1,8 & -3,6 & -1,2 & 2,49 \\ -1,5 & 0,43 & 0,01 & 1,90 & 0,78 & 1,25 & -3,4 & -0,9 \\ -0,7 & -0,8 & -0,7 & -0,9 & -0,1 & -0,2 & 0,93 & 1,02 \\ 2,66 & 1,02 & 1,38 & 0,21 & -0,5 & -0,5 & 0,95 & -1,3 \\ -3,3 & -0,9 & -1,5 & -0,2 & 0,75 & 0,18 & -1,0 & 1,55 \\ 2,40 & 0,53 & 1,12 & 0,85 & -0,6 & 0,75 & -0,4 & -1,7 \\ -1,1 & -0,1 & -0,6 & -1,3 & 0,32 & -1,4 & 1,75 & 1,50 \\ 0,21 & -0,1 & 0,18 & 0,98 & -0,1 & 1,15 & -1,6 & -0,9 \end{bmatrix}$$

**Figur 3.21.** DCT - Tal.

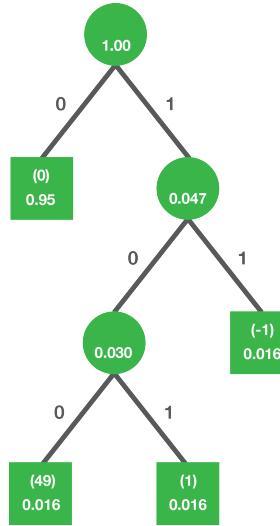


**Figur 3.22.** Kvantisering - Visuel.

$$C_{1,1} = \begin{bmatrix} 49 & -1 & 1 & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \end{bmatrix}$$

**Figur 3.23.** Kvantisering - Tal.

essentielt for Huffmankodningen, da den nu effektivt kan komprimere. Huffmantræet for undermatricen kan ses på figur 3.24. Tallene uden parentes repræsenterer hyppigheden af tallene i parentes.



**Figur 3.24.** Huffman-træ for undermatricen.

Træet havde mildt sagt været markant større, hvis at Huffmantræet var lavet ud fra eksempelvis  $A_{1,1}$  eller  $B_{1,1}$ . Her er det åbenlyst hvordan de to nævnte matricer, kun er *forberedende* for en effektiv komprimering.

Der kan med fordel opstilles en tabel med relevante observationer fra Huffmantræet og

dens resultater, se tabel 3.1. Den binære form blev fundet jævnfør 3.1.6 på side 24. I binær

Indgangsværdi	Antal	Sandsynlighed	Binær form
49	1	$\frac{1}{64} \approx 0,016$	100
1	1	$\frac{1}{64} \approx 0,016$	101
-1	1	$\frac{1}{64} \approx 0,016$	11
0	61	$\frac{61}{64} \approx 0,95$	0

**Tabel 3.1.** Huffmantræet i tabelform

form vil undermatricen nu hedde:

En komprimeret fil er opnået, og den lange streng af nuller resulterer i, at filen fylder mindre, end hvis Huffmankodningen var blevet gjort før kvantiseringen. Jævnfør algoritmen på side 28 udføres *dekomprimeringen*, hvilket gøres med de modsatte regneoperationer i forhold til komprimeringen. Resultatet af den inverse algoritme på  $A_{(1, 1)}$  resulterer i

## Algoritme: Dekomprimering vha. DCT

1. Input: Komprimeret fil
  2. Output: Dekomprimeret billede,  $A'$

### *Dekomprimering (invers proces)*

9. Entropidekodning:  
 10. Dekvantisering:  
 11. Invers DCT / DCT-III:  
 12. Decentrering omkring 0:  
 13. Samling:

Filen dekomprimeres vha. Huffman til  $C'$   
 $B' = Q_{(i, j)} \cdot C'_{(i, j)}$   
 $A' = U^T \cdot B' \cdot U$   
 $A' = A' + 128$  indgangsvist  
 $8 \times 8$  matricer samles til ét billede

værdierne illustreret i figur 3.25 og 3.26. Den umiddelbare største ændring af undermatricen

**Figur 3.25.** Dekomprimeret - Visuel.

**Figur 3.26.** Dekomprimeret - Tal.

er dens endnu mere ensartethed. Da der ikke forekommer drastiske spring i overgangene mellem pixelene ser den monoton ud. Det er svært for det menneskelige øje, at skældne mellem denne undermatrice og den oprindelige, hvilket var intentionen i første omgang. For referencens skyld udføres algoritmen på hele Lena i de følgende figurer. Det undlades dog at vise den binære form. Jævnfør 3.27 og 3.29 er det enormt svært at skelne mellem billederne. Dette forstærkes ydermere af fejlbilledet, der illustrerer forskellen mellem det



**Figur 3.27.** Dekomprimeret - Lena.



**Figur 3.28.** Fejlbillede - Lena.



**Figur 3.29.** Oprindelig - Lena.

nye og det oprindelige billede. Grå illustrerer ingen forskel, sort illustrerer en mørkere farve i det nye billede, og ligeså illustrerer en lysere farve, at det nye billede er blevet lysere. En farvenuineance illustrerer en fargeforskelse. Hensigten er opnået; billedet har ikke ændret sig betydeligt, og filstørrelsen er komprimeret i forhold til før algoritmen.



---

## BILLEDKOMPRIMERING MED PRINCIPAL COMPONENT ANALYSIS

Følgende afsnit vil undersøge den statistiske metode *Principal Component Analysis* (PCA), der bruger elementer fra lineær algebra. Den statistiske baggrund såvel som baggrunden i lineær algebra vil blive klarlagt og brugt til udviklingen af et Pythonprogram, der kan komprimere et billede.

For at give et indblik i PCA og brugen af denne, tages først udgangspunkt i et eksempel, som senere generaliseres til brug af PCA på et vilkårligt datasæt. Eksemplet til brug i de følgende undersøgelser er, at en given forsker forsøger at forstå et fænomen ved at undersøge forskellige variable i et stort datasæt; datasættets variable er målinger fra et eksperiment, men der fremstår ikke nogen tydelige sammenhænge i dataene. Dette er enten fordi dataene indeholder støj, eller fordi dimensionen af datamængden er så stor, at det er uoverskueligt at se sammenhængene.

Tag et legetøjseksperiment som eksempel: en bold hænger i en fjeder fra loftet og hopper op og ned. Dette forsøg måles vha. tre kameraer placeret omkring bolden og fjederen, som tilsammen kan måle boldens position i et tredimensionelt rum, hvor hvert kamera mäter to dimensioner. Uvidende om at bolden hopper lodret op og ned, måles alle tre dimensioner alligevel, men kameraerne er ikke nødvendigvis placeret i  $90^\circ$  i forhold til hinanden, hvorved det ikke er et standardkoordinatsystemet, der måles efter. Boldens position burde kunne repræsenteres vha. én variabel, men grundet det sløsede eksperimentsetup bliver boldens position repræsenteret ved mange variable frem for den enkelte variabel  $z$ . Spørgsmålet er altså; hvordan ændres et rodet datasæt (her menes: for mange variable og støjfyldt datasæt) til en simpel repræsentation vha. en variabel  $z$  [Shlens, s. 1-2]?

Formålet med PCA er at finde den "bedste" repræsentation (begrebet den bedste repræsentation vil blive uddybet i sektion 4.1) af et datasæt med støj og unødvendige dimensioner såvel som at afklare sammenhænge, der for det blotte øje, ikke er tydelige. Fordelene ved dette er at datasættet kan forsimples til de data, der reelt har betydning for forsøget, så eksempelvis to forskellige variable reduceres til én variabel i stedet.

Datasættet kan repræsenteres ved en  $m \times n$  matrix,  $X$ , bestående af  $n$  prøver (her: tidspunkterne kameraerne tager et billede) med  $m$  målinger (her:  $x(t)$ ,  $y(t)$  koordinaterne for de enkelte kameraer). Dermed kan *søjlerne* i  $X$  betegnes som vektorer,

$$\vec{z}_t = \begin{bmatrix} x_A & y_A & x_B & y_B & x_C & y_C \end{bmatrix}^T$$

hvor  $t$  er tidspunktet, og indgangene er koordinaterne målt af hhv. kamera A, B og C.

## Skift af basis

Lad  $V$  være et underrum af  $\mathbb{R}^n$ , så er en basis til  $V$  en lineært uafhængigt udspændende mængde bestående af vektorer i  $V$ , hvorved alle vektorer i  $V$  kan repræsenteres unikt som en lineær kombination af vektorerne i basen.

Standardbasen består af standardvektorerne, og det er denne basis, der normalt bruges til repræsentation af data. Der findes mange forskellige baser til et underrum, men det kan være fordelagtigt at ændre basen for et datasæt, så datasættets egenskaber fremstår tydeligere eller simplere [Spence et al., s. 241-242].

Det ønskes at finde en anden basis, som er en lineær kombination af den oprindelige basis, der udtrykker data på "bedste mulige måde". Det ønskes altså at lave en lineær transformation på  $m \times n$  matricen  $X$  til  $m \times n$  matricen  $Y$  for en  $m \times m$  matrix  $P$ , hvor  $Y$  er en repræsentationen af  $X$  i den nye basis  $P$ . Dette udtrykkes som

$$Y = PX \quad (4.1)$$

Undersøges dimensionerne af ligning 4.1 fremkommer det at

$$(m \times n) = (m \times m)(m \times n) \quad (4.2)$$

Det gælder ydermere at  $\vec{p}_i^T$  er *søjlerne* i  $P$ ,  $\vec{x}_j$  er *søjlerne* i  $X$  og  $\vec{y}_j$  er *søjlerne* i  $Y$ . Ligning 4.1 er et skift af basis, og det gælder at  $P$  er en matrix, der lineært transformerer  $X$  til  $Y$ . Det gælder ydermere, at rækkerne i  $P = [\vec{p}_1 \quad \vec{p}_2 \quad \dots \quad \vec{p}_m]^T$  er basisvektorerne i den nye basis, der udtrykker *søjlerne* i  $X$ . *Søjlerne*  $\vec{y}_j$  kan opskrives som

$$Y = \begin{bmatrix} \vec{p}_1 \cdot \vec{x}_1 & \cdots & \vec{p}_1 \cdot \vec{x}_n \\ \vdots & \ddots & \vdots \\ \vec{p}_m \cdot \vec{x}_1 & \cdots & \vec{p}_m \cdot \vec{x}_n \end{bmatrix}, \quad \vec{y}_j = \begin{bmatrix} \vec{p}_1 \cdot \vec{x}_j \\ \vdots \\ \vec{p}_m \cdot \vec{x}_j \end{bmatrix}$$

Endvidere er her tale om en ortogonal projktion af  $X$  over i  $P$ . Ved projekteringen udvælges en delmængde af  $P$  som nye basisvektorer som repræsenterer *søjlerne* af  $X$  [Richardson, s. 6]. Den j'te koefficient af  $\vec{y}_j$  er en projktion på den j'te række af  $P$ . Projektionen definerer reelt set et nyt ortogonalt koordinatsystem, som beskriver variansen, i dette tilfælde, et billede optimalt [Shlens, s. 3].

## 4.1 STATISTISKE VÆRKTØJER

Rækkevektorerne  $\vec{p}_1, \vec{p}_2, \dots, \vec{p}_m$  er principale komponenter til matricen  $X$ , hvorved de er ortogonalt lineært uafhængige. Principale komponenter er akser, hvorved det oprindelige datasæt har nye koordinater, der sikrer at enkelte akser har stor variation [Shlens, s. 5]. Som tidligere nævnt ønskes det at finde at den "bedste" repræsentation af  $X$  såvel som at vælge et passende  $P$ . Svarene på disse kommer af at bestemme, hvilke egenskaber  $Y$  skal have. Når egenskaberne til  $Y$  er afklarede så kan det afgøres, hvilke egenskaber  $P$  skal have for at danne den ønskede  $Y$ . Dette lægges dog lidt til side for nu, og der ses først på de overflødige data, der kan være i datasættet  $X$  (dette kan være hhv. støj og redundans), og efterfølgende hvordan det ønskes at repræsentere datasættet.

### 4.1.1 Støj

Støj i et datasæt kan fremkomme af mange forskellige årsager, men vigtigst er betydningen af støjen, uanset art. Et datasæt med støj vil, efter bearbejdning, give fejlbehæftede resultater, da der vil være fejlagtige målinger, der påvirker resultatet. Det ønskes altså at minimere eller helt fjerne støj i datasættet. En udbredt metode til at udregne støj i et datasæt er SNR (signal-to-noise-ratio), som er et forhold mellem varianserne for hhv. signalet og støj<sup>1</sup>. SNR er defineret som værende  $SNR = \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{støj}}^2}$ . Er SNR høj er det et udtryk for meget *lidt* støj i datasættet, men er SNR derimod lav er det et udtryk for meget støj i datasættet. I datasættet  $X$  kan stor støj fremkomme, hvis eksempelvis vind påvirker bolden, hvorved den ikke hopper lodret op og ned men derimod svinger lidt fra side til side en gang i mellem. Dette ville betyde at den grafiske repræsentation, der burde være en lige linje, nærmere ikke længere vil være så ensformig. Støj vil her være den vandrette varians [Shlens, s. 3-4].

### 4.1.2 Redundans

Redundans er et udtryk for, at det samme element er beskrevet ved flere forskellige variable, der egentlig siger det samme. Redundans kan i bold-forsøget fremkomme ved at to kameraer sidder tæt ved siden af hinanden, hvorved de begge to vil have variable, men variablene giver næsten de præcis samme målinger. Dette kan repræsenteres ved et plot af  $(x_A, x_B)$ , som er hhv. x-koordinaten målt i kamera A og kamera B, og hvor kamera A og B er meget tæt placeret. Dette plot ville være tæt på en ret linje med forskrift  $x = y$ . Der er her en stor kovarians<sup>2</sup>, hvilket vil sige at de er lineært afhængige, og det er dermed spild af ressourcer at lave denne måling. Det giver også unødvendige data i datasættet, da der ikke udtrykkes andet eller mere præcis data ved den ekstra variabel. Redundans ønskes minimeret, da det er overflødig data og dermed blot fremstår som fyld i datasættet [Shlens, s. 4].

## 4.2 KOVARIANSMATRIX

I sektion 4.1.1 blev SNR introduceret som værende et udtryk for mængden af støj i et datasæt, mens redundans i sektion 4.1.2 blev introduceret som værende kovariansen mellem to variable, dvs. relationen mellem de to variable. Disse vil blive uddybet i dette afsnit, men først vil ligningen for gennemsnittet blive introduceret.

### 4.2.1 Gennemsnit

Lad  $\vec{x}_i$  være en *rækkevektor* bestående af alle målinger,  $x_1, \dots, x_n$ , i variablen. Gennemsnittet af målingerne betegnes ved  $\mu_x$  og kan findes som værende

<sup>1</sup>Varians introduceres i afsnit 4.2.2, men kan her anses som spredningen i dataene.

<sup>2</sup>Begrebet kovarians præsenteres i afsnit 4.2

$$\mu_x = \frac{1}{n}(x_1 + \dots + x_n) \quad (4.3)$$

Vektoren centreres omkring nul fremfor gennemsnittet  $\mu$  ved at subtrahere gennemsnittet  $\mu_x$  fra vektoren  $\vec{x}_i$ . Denne vektor angives som  $\tilde{\vec{x}}_i$  og har gennemsnittet nul.

$$\tilde{\vec{x}}_i = [(x_1 - \mu_x) \ \dots \ (x_n - \mu_x)] \quad (4.4)$$

#### 4.2.2 Varians og kovarians

Varians er et udtryk for spredningen af data i en variabel. Spredningen, og dermed variansen i  $\tilde{\vec{x}}_i$ , kan findes som følgende prikprodukt

$$\sigma_{\tilde{\vec{x}}_i}^2 = \frac{1}{n-1} \tilde{\vec{x}}_i \cdot \tilde{\vec{x}}_i^T \quad (4.5)$$

En høj varians er udtryk for stor spredning af dataene i  $\tilde{\vec{x}}_i$ , mens en lille varians er et udtryk for en lille spredning i dataene og dermed meget ensformig data i denne vektor, og dermed i den række data vektoren repræsenterer. Det er ydermere interessant at se på, hvordan to rækkevektorer relaterer til hinanden, hvilket gøres ved beregning af kovariansen mellem de to vektorer. Kovariansen mellem vektorerne  $\tilde{\vec{x}}_a$  og  $\tilde{\vec{x}}_b$  kan beregnes som

$$\sigma_{(\tilde{\vec{x}}_a, \tilde{\vec{x}}_b)}^2 = \frac{1}{n-1} \tilde{\vec{x}}_a \cdot \tilde{\vec{x}}_b^T \quad (4.6)$$

hvor  $(\tilde{\vec{x}}_a) \cdot (\tilde{\vec{x}}_b)^T$  er prikproduktet mellem de to vektorer. Kovariansen mellem de to vektorer udtrykker den lineære relation mellem de to vektorer. En høj kovarians er udtryk for at dataene relaterer meget til hinanden, mens en lille kovarians er udtryk for, at dataene er lidt relaterede. Er kovariansen  $\sigma_{(\tilde{\vec{x}}_a, \tilde{\vec{x}}_b)}^2 = 0$ , er vektorerne fuldstændigt lineært urelaterede til hinanden. Det gælder at  $\sigma_{(\tilde{\vec{x}}_a, \tilde{\vec{x}}_b)}^2 = \sigma_{(\tilde{\vec{x}}_a)}^2$  hvis  $A = B$  [Shlens, s. 5].

Undersøges der mange variable, som i eksperimentet med bolden, er der flere variable end blot to, og det kan være interessant at se på variansen for alle variable, såvel som kovariansen mellem alle variable. Dette er dog en uoverskuelig process, hvis det skal gøres ved ovenstående metode for alle variable, specielt hvis datasættet består af mange variable. Der kan derfor være fordelagtigt at udregne matrixproduktet

$$C_{\tilde{X}} = \frac{1}{n-1} \tilde{X} \tilde{X}^T, \quad \text{hvor } \tilde{X} = [\tilde{\vec{x}}_1 \ \tilde{\vec{x}}_2 \ \dots \ \tilde{\vec{x}}_m]^T \quad (4.7)$$

$\tilde{X}$  består af rækkevektorerne  $\tilde{\vec{x}}_i = [\tilde{x}_1 \ \tilde{x}_2 \ \dots \ \tilde{x}_n]$ . Det smarte ved  $C_{\tilde{X}}$  er, at der ved hjælp af denne både beregnes variansen for alle vektorer såvel som kovariansen mellem alle vektorer. Dette ses tydeligt i den eksplisitte form af matrixproduktet

$$C_{\tilde{X}} = \frac{1}{n-1} \tilde{X} \tilde{X}^T = \frac{1}{n-1} \begin{bmatrix} \tilde{\vec{x}}_1 \cdot \tilde{\vec{x}}_1^T & \tilde{\vec{x}}_1 \cdot \tilde{\vec{x}}_2^T & \dots & \tilde{\vec{x}}_1 \cdot \tilde{\vec{x}}_m^T \\ \tilde{\vec{x}}_2 \cdot \tilde{\vec{x}}_1^T & \tilde{\vec{x}}_2 \cdot \tilde{\vec{x}}_2^T & \dots & \tilde{\vec{x}}_2 \cdot \tilde{\vec{x}}_m^T \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\vec{x}}_m \cdot \tilde{\vec{x}}_1^T & \tilde{\vec{x}}_m \cdot \tilde{\vec{x}}_2^T & \dots & \tilde{\vec{x}}_m \cdot \tilde{\vec{x}}_m^T \end{bmatrix} \quad (4.8)$$

Det gælder her at variansen, for de respektive vektorer, er at finde i diagonalen, mens kovariansen mellem de enkelte variable kan findes udenfor diagonalen. Der er altså ved en enkelt beregning udregnet alle varianser og kovarianser for datasættet. Bemærk ydermere at matricen  $C_{\tilde{X}}$  er en  $m \times m$  symmetrisk matrix navngivet kovariansmatricen.  $C_{\tilde{X}}$  er meget brugbar til hurtigt og nemt at se relationer i datasættet  $\tilde{X}$ , men det kan gøres endnu nemmere. Det ønskes at finde  $C_Y$ , som er en manipuleret kovariansmatrix, men før denne findes, skal egenskaberne for den defineres.

### 4.2.3 Diagonalisering af kovariansmatricen

Det er jævnfør afsnit 4.1.2 på side 33 ønskværdigt at nedbringe redundans mest muligt, og da redundans er et udtryk for kovariansen mellem to vektorer, må det gælde, at kovariansen skal nedbringes til nul, hvorved dataene ikke relaterer til hinanden længere. Det gælder altså at alle indgange i  $C_Y$ , der ikke er i diagonalen skal være nul, hvorved  $C_Y$  er en diagonalmatrix.

#### Diagonalisering

En diagonalisrerbar matrix er defineret ved følgende sætning:

En  $n \times n$  matrix  $A$  er diagonaliserbar, hvis og kun hvis  $A$  har  $n$  lineært uafhængige egenvektorer. I så fald gælder det at

$$A = PDP^{-1} \quad (4.9)$$

hvor  $P$  er en matrix med  $A$ 's egenvektorer som søjler, og  $D$  er en diagonalmatrix med  $A$ 's egenværdier som indgange. Ovenstående sætning bevises [Spence et al., s. 315-316]. Lad:

$$P = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \end{bmatrix}$$

$$D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$$

Det gælder dermed at:

$$AP = A \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \end{bmatrix} = \begin{bmatrix} A\vec{v}_1 & A\vec{v}_2 & \dots & A\vec{v}_n \end{bmatrix}$$

$$PD = P \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} = \begin{bmatrix} \lambda_1 \vec{v}_1 & \lambda_2 \vec{v}_2 & \dots & \lambda_n \vec{v}_n \end{bmatrix}$$

Det antages, at  $A$  er diagonaliserbar med ligning 4.9, der omskrives til:

$$AP = PD$$

$$A\vec{v}_1 = \lambda_1 \vec{v}_1, A\vec{v}_2 = \lambda_2 \vec{v}_2, \dots, A\vec{v}_n = \lambda_n \vec{v}_n$$

$P$  er invertibel, har lineært uafhængige søjler, og det gælder at  $\vec{v}_1 \neq 0, \dots, \vec{v}_n \neq 0$ . Dette betyder, at søjlerne i  $P$  består af de lineært uafhængige egenvektorer til  $A$  med dertilhørende egenværdier  $\lambda_1, \lambda_2, \dots, \lambda_n$ . Dette kan yderligere bevises ved at tage beviset fra den anden vej:

Givet er  $n$  lineært uafhængige egenvektorer  $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n$  med tilhørende egenværdier  $\lambda_1, \lambda_2, \dots, \lambda_n$ . Definerer  $P$  og  $D$  som ovenfor, og det følger dermed igen at  $AP = PD$ , da det gælder at:

$$A\vec{v}_1 = \lambda_1 \vec{v}_1, A\vec{v}_2 = \lambda_2 \vec{v}_2, \dots, A\vec{v}_n = \lambda_n \vec{v}_n$$

Da  $P$ 's søjler er lineært uafhængige, er  $P$  inverterbar, og det gælder derfor at

$$A = PDP^{-1} \quad \blacksquare$$

Da det ønskes at diagonalisere  $C_Y$ , er det er altså vigtigt i ligning 4.1 på side 32 at vælge en  $P$ , således at  $C_Y$  er en diagonalmatrix.

### 4.3 UDREGN PCA

I de foregående afsnit er baggrunden for PCA blevet undersøgt og begrundet. PCA vil i dette afsnit blive fundet algebraisk for en  $m \times n$  matrix  $X$ , der som i eksemplet med bolden har  $m$  variable og  $n$  målinger. Lad  $P$  være den ortonormale  $m \times m$  matrix  $P$ , hvor  $Y = PX$  således at  $C_Y = \frac{1}{n-1}YY^T$  er diagonaliseret, og rækkerne i  $P$  er principale komponenter til  $X$ .

Først omskrives  $C_Y$ , til at være et udtryk af  $P$  frem for  $Y$ . Dette gøres med henblik på at diagonalisere  $C_Y$ .

$$\begin{aligned} C_Y &= \frac{1}{n-1}YY^T \\ &= \frac{1}{n-1}(PX)(PX)^T \\ &= \frac{1}{n-1}PXX^TP^T \\ &= \frac{1}{n-1}P(XX^T)P^T \\ &= \frac{1}{n-1}PSP^T \quad \text{hvor} \quad S = C_X = XX^T \end{aligned} \tag{4.10}$$

Det gælder her, at  $S$  er en symmetrisk  $m \times m$  matrix, hvilket kan ses ved  $(XX^T)^T = (X^T)^TX^T = XX^T$ . Ifølge [Richardson, s. 8] og afsnit 4.2.3 er enhver symmetrisk matrix orthogonal diagonaliserbar og dermed også ortonormal diagonaliserbar (orthogonale med længden én). Dermed kan  $S$  udtrykkes ved

$$S = EDE^{-1}$$

hvor  $D$  er diagonalmatricen, hvor indgangene i diagonalen er egenværdierne til  $S$  ordnet efter  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m \geq 0$ <sup>3</sup>.  $E$  er en  $m \times m$  ortonormal matrix, hvis *søjler*

---

<sup>3</sup>Se bevis for at egenværdierne er positive i appendiks C

er normaliserede egenvektorer tilhørende egenværdierne og følger samme ordning som egenværdierne, hvorved egenvektoren, til den største egenværdi, står som første søjle i  $E$ . Da  $E$  er ortonormal, gælder det at  $E^{-1} = E^T$  [Spence et al., s. 413] og 4.12 kan udtrykkes som

$$S = EDE^T \quad (4.12)$$

Egenvektorer og egenværdier er defineret som følgende: Lad  $T$  være en lineær transformation hvor domænet og kodomænet er lig  $\mathbb{R}^n$ , hvilket er en lineær operator på  $\mathbb{R}^n$ . En vektor  $\vec{v}$  forskellig fra nul i  $\mathbb{R}^n$  benævnes som en egenvektor af  $T$ , hvis  $T(\vec{v})$  er et produkt af  $\vec{v}$ ; dvs.

$$T(\vec{v}) = \lambda \vec{v} \quad (4.13)$$

for skalarerne  $\lambda$ . Skalarerne  $\lambda$  benævnes som egenværdier af  $T$ , som tilhører  $\vec{v}$ . [Spence et al., s. 294]

Egenværdierne ved en kvadratisk matrix  $A$ , er værdierne af  $\lambda$  som tilfredsstiller

$$\det(A - I_m \cdot \lambda) = 0 \quad (4.14)$$

i forhold til  $\lambda$  og herefter indsættes disse i diagonalen i  $D$ . Udtryk 4.14 kaldes den karakteristiske polynomium af  $A$ , mens  $\det(A - I_m \cdot \lambda)$  kaldes det karakteristiske polynomium af  $A$  [Spence et al., s. 302].

Determinanten, som fremgår i fornævnte udtryk, har nyttige egenskaber i lineær algebra, som eksempelvis at finde egenværdier. Ved matricer hvor  $n \geq 3$ , hvilket gør sig gældende ved de billede, der undersøges i projektet, er determinanten defineret ved en  $n \times n$  matrix  $A$  ved første række som

$$\det A = a_{11} \cdot \det A_{11} - a_{12} \cdot \det A_{12} + \cdots + (-1)^{1+n} a_{1n} \cdot \det A_{1n}$$

Ved at lade  $c_{ij} = (-1)^{i+j} \cdot \det A_{ij}$  kan definitionen af determinanten af  $A$  skrives som

$$\det A = a_{11}c_{11} + a_{12}c_{12} + \cdots + a_{1n}c_{1n}$$

Dette udtryk er kofaktor ekspansionen af  $A$  hen ad første række. Et generelt eksempel på at finde egenværdier følger; lad  $S$  være en kendt matrix i  $\mathbb{R}^3$ . Det ønskes at finde løsningen til den ubekendte  $\lambda$ . Udtrykket  $\det(A - I_m \cdot \lambda)$  opstilles.

$$\det(A - I_m \cdot \lambda) = \det \begin{bmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{bmatrix}$$

Ved at benytte kofaktor ekspansion hen ad første række fås et ligningssystem, som ved de mulige løsninger udtrykker egenværdierne. De tilhørende egenvektorer findes som løsninger til

$$(A - I_m \cdot \lambda) \cdot \vec{v} = 0$$

for de respektive  $\lambda$ -værdier og indsættes som søjlevекторer i  $E$ . I praksis vil det sige at løse ligningssystemet til

$$\begin{bmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

hvilket blot gøres ved at gå til reduceret trappeform med  $A$  og opstille ligningssystemet på parametreret vektorform. Vektorerne danner en basis med de tilhørende egenvektorer for egenrummet af  $A$ . Dette er dog beregningsmæssigt så tungt (specielt med matricer større end  $3 \times 3$ , at det i praksis kun vil blive gjort vha. et digitalt beregningsværktøj. De egenvektorer, der fremkommer af ovenstående beregninger, er dog kun ortogonale og ikke ortonormale, som det ønskes i  $E$ . For at lave dem ortonormale normaliseres de til længden én ved den indgangsvise division med længden af sig selv

$$\hat{v}_i = \frac{v_i}{\|v_i\|} \quad (4.15)$$

Pr. definition er  $E$ 's søjler lineært uafhængige, ortogonale (nu ortonormale) og af dimension  $m$  [Spence et al., s. 315,426], da dette er egenskaberne for egenvektorer.

Transformationsmatricen  $P$  defineres, og det ses, at det er praktisk at sætte  $P = E^T$  og dermed  $P^T = E$ . Dette betyder, at *rækkerne* i  $P$  er søjlevекторerne i  $E$ , og disse er netop blevet defineret som værende egenvektorerne til  $S$ .  $E$  er ortonormal, og det gælder derfor at  $E^T E = I_m$ , hvor  $I_m$  er  $m \times m$  identitetsmatricen. Ligning 4.11 kan vha. ligning 4.12 omskrives til

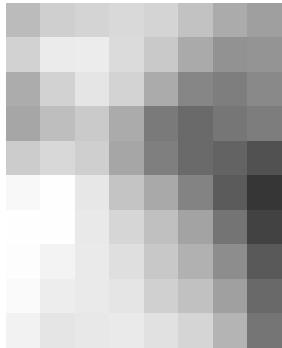
$$\begin{aligned} C_Y &= \frac{1}{n-1} P S P^T \\ &= \frac{1}{n-1} P (E D E^T) P^T \\ &= \frac{1}{n-1} E^T E D E^T E \\ &= \frac{1}{n-1} D \end{aligned} \quad (4.16)$$

Det ses her, at  $C_Y$  er blevet diagonaliseret når  $P = E^T$ , hvilket var målet med PCA. I de foregående afsnit er det fremkommet, at principal komponenterne til  $X$  er *rækkerne* i  $P$ . Det ses i ligning 4.16, at den  $i$ 'te diagonalindgang er variansen af  $X$  i den nye basis  $P$ , hvilket imidlertid betyder, at egenværdierne i  $D$  udtrykker variansen af de forskellige principal komponenter og dermed deres indflydelse på datasættet [Richardson, s. 8].

#### 4.4 BILLEDEKOMPRIMERING MED PCA

PCA-metoden kan anvendes på mange forskellige former for data med forskellige formål, men jævnfør rapportens formål undersøges PCA i forhold til komprimering af et billede. Billedet angives som tre  $m \times n$  matrix indeholdende pixelintensiteten i de forskellige indgange, der repræsenterer en pixel i billedet.

For overskuelighedens skyld undersøges her et gråtone-billede med  $10 \times 8$  pixels. Billedet betegnes ved matricen  $X$ . Udgangspunktet for de følgende regneoperationer er udtrykket  $Y = PX$ , hvor  $Y$  er den transformerede matrix, og  $P$  udgør en matrix med egenvektorerne til  $C_X$  som *rækker*, hvilket reelt set benævnes principale komponenter til  $X$  (fremover vil principale komponenter forkortes med PC). Det skal nævnes, at efter eksemplet udføres udregningerne på Lena, dog vil kun resultaterne af Lena fremgå. Dette gøres for at danne grundlag for sammenligning i forhold til DCT-billedkomprimeringen.



**Figur 4.1.** Oprindelig - Visuel.

$$X = \begin{bmatrix} 187 & 207 & 213 & 217 & 213 & 194 & 172 & 160 \\ 210 & 236 & 236 & 219 & 202 & 170 & 147 & 148 \\ 173 & 210 & 230 & 212 & 171 & 135 & 126 & 138 \\ 167 & 190 & 203 & 171 & 123 & 106 & 119 & 125 \\ 204 & 217 & 207 & 167 & 127 & 107 & 99 & 81 \\ 249 & 255 & 231 & 197 & 169 & 132 & 91 & 55 \\ 255 & 254 & 234 & 214 & 193 & 163 & 117 & 66 \\ 253 & 244 & 234 & 224 & 201 & 177 & 142 & 89 \\ 251 & 237 & 235 & 229 & 209 & 193 & 161 & 105 \\ 242 & 230 & 232 & 235 & 225 & 214 & 180 & 118 \end{bmatrix}$$

**Figur 4.2.** Oprindelig - Tal.

For at skabe et overblik over billedkomprimeringen med PCA ses algoritmen herunder med uddybbelse og anvendelse på  $X$  se figur 4.2.

---

#### Algoritme: Komprimering vha. PCA

---

1. Input:	Billede, $X: m \times n$ pixels
2. Output:	Komprimeret fil
<i>Komprimering</i>	
3. Gennemsnit:	$\mu_x = \frac{1}{n}(x_1 + \dots + x_n)$
4. Standardiserer:	$\tilde{x}_i = [(x_1 - \mu_x) \quad \dots \quad (x_n - \mu_x)]$
5. Kovarians matrix:	$C_{\tilde{X}} = \frac{1}{n-1}\tilde{X}\tilde{X}^T$ , hvor $\tilde{X} = [\tilde{x}_1 \quad \dots \quad \tilde{x}_m]^T$
6. Sortering:	Eigenvektorer, $P$ , sorteres vha. egenværdier
7. PC vælges:	Antal bibeholdte principale komponenter afgør kvalitet
8. Transformerer matrix	$Y = PX$
9. Entropikodning:	$Y$ , $P$ og $\mu_x$ gemmes i en fil vha. Huffman

Med udgangspunkt i  $X$  findes gennemsnittene for de enkelte rækkevektorer  $\tilde{x}_i$  som derefter trækkes fra de respektive rækkevektorer. Jævnfør vektoren 4.17 ses det, at gennemsnittet for første rækkevektor er  $\frac{1}{8}(187 + 207 + 213 + 217 + 213 + 194 + 172 + 160) = 195,38$ .

For at skabe indblik i følgende regnemetode stilles resultatet for samtlige gennemsnit op i en såkaldt gennemsnitsvektor, der indeholder alle gennemsnittene for de enkelte rækker

$$\mu_{\tilde{x}} = [195,38 \quad 196 \quad 174,375 \quad 150,5 \quad 151,125 \quad 172,375 \quad 187 \quad 195,5 \quad 202,5 \quad 209,5]^T \quad (4.17)$$

Gennemsnittet trækkes fra de respektive rækker i  $X$  - dvs.  $X$  standardiseres og resulterer i følgende matrix:

$$\tilde{X} = \begin{bmatrix} -8,375 & 11,625 & 17,625 & 21,625 & 17,625 & -1,375 & -23,375 & -35,375 \\ 14 & 40 & 40 & 23 & 6 & -26 & -49 & -48 \\ -1,375 & 35,625 & 55,625 & 37,625 & -3,375 & -39,375 & -48,375 & -36,375 \\ 16,5 & 39,5 & 52,5 & 20,5 & -27,5 & -44,5 & -31,5 & -25,5 \\ 52,875 & 65,875 & 55,875 & 15,875 & -24,125 & -44,125 & -52,125 & -70,125 \\ 76,625 & 82,625 & 58,625 & 24,625 & -3,375 & -40,375 & -81,375 & -117,375 \\ 68 & 67 & 47 & 27 & 6 & -24 & -70 & -121 \\ 57,5 & 48,5 & 38,5 & 28,5 & 5,5 & -18,5 & -53,5 & -106,5 \\ 48,5 & 34,5 & 32,5 & 26,5 & 6,5 & -9,5 & -41,5 & -97 \\ 32,5 & 20,5 & 22,5 & 25,5 & 15,5 & 4,5 & -29,5 & -91,5 \end{bmatrix} \quad (4.18)$$

Den nye matrix  $\tilde{X}$  består af rækkevektorerne  $\vec{x}_i$ , der nu er centreret omkring nul. Kovariansmatricen  $C_X$  findes vha. ligning 4.7 med dimensionerne  $m \times m$ , hvor  $m = 10$ , hvilket ses ved, at dimensionerne er  $(10 \times 8) \cdot (8 \times 10) = 10 \times 10$ .

$$C_X = \begin{bmatrix} 449 & 654 & 667 & 419 & 720 & 1156 & 1116 & 944 & 826 & 742 \\ 654 & 1336 & 1375 & 1164 & 1839 & 2563 & 2288 & 1881 & 1565 & 1226 \\ 667 & 1375 & 1568 & 1336 & 1832 & 2405 & 2091 & 1716 & 1410 & 1062 \\ 419 & 1164 & 1336 & 1339 & 1820 & 2214 & 1846 & 1499 & 1199 & 808 \\ 720 & 1839 & 1832 & 1820 & 2943 & 3916 & 3430 & 2817 & 2322 & 1728 \\ 1156 & 2563 & 2405 & 2214 & 3916 & 5533 & 4992 & 4132 & 3465 & 2728 \\ 1116 & 2288 & 2091 & 1846 & 3430 & 4992 & 4586 & 3824 & 3247 & 2637 \\ 944 & 1881 & 1716 & 1499 & 2817 & 4132 & 3824 & 3211 & 2744 & 2257 \\ 826 & 1565 & 1410 & 1199 & 2322 & 3465 & 3247 & 2744 & 2367 & 1985 \\ 742 & 1226 & 1062 & 808 & 1728 & 2728 & 2637 & 2257 & 198 & 1746 \end{bmatrix} \quad (4.19)$$

Da det gælder at  $S = \tilde{X}\tilde{X}^T$ , og endvidere som tidligere nævnt  $S = EDE^T$ , findes egenværdierne og egenvektorerne vha. det karakteristiske polynomium til kovariansmatricen hvor  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m \geq 0$ . Egenvektorerne sorteres ikke, men tilrettes samme ordning som egenværdierne, da egenværdierne og egenvektorerne hænger sammen i par. Metoden til at finde  $S$ ,  $E$  og  $D$  såvel som egenværdier og egenvektorer ses i afsnit 4.3. For forståelsens skyld kan egenværdierne opstilles som en diagonal matrix,  $D$ , med tilhørende egenvektorer som matricerne  $E$  og  $E^T$  og dermed udregne  $S$ , dette gøres dog ikke i denne rapport. Dette ville resultere i kovariansmatricen. Sorterede egenværdier findes, og egenvektorer opstilles som søger:

$$\begin{aligned} \lambda_1 &= 2,3 \cdot 10^4, & \lambda_2 &= 1,43 \cdot 10^3, & \lambda_3 &= 530, & \lambda_4 &= 10, & \lambda_5 &= 7,35, & \lambda_6 &= 2,22, \\ \lambda_7 &= 0,6, & \lambda_8 &= 5 \cdot 10^{-14}, & \lambda_9 &= -3,8 \cdot 10^{-13}, & \lambda_{10} &= -8,7 \cdot 10^{-14} \end{aligned}$$

$$E = \left\{ \begin{array}{c} \left[ \begin{array}{c} 0,11 \\ 0,07 \\ 0,55 \\ 0,16 \\ -0,52 \\ -0,36 \\ -0,23 \\ -0,30 \\ 0,28 \\ 0,06 \end{array} \right], \left[ \begin{array}{c} 0,23 \\ -0,19 \\ 0,30 \\ 0,40 \\ 0,03 \\ 0,25 \\ 0,73 \\ -0,02 \\ -0,03 \\ 0,12 \end{array} \right], \left[ \begin{array}{c} 0,22 \\ -0,45 \\ 0,54 \\ -0,06 \\ 0,34 \\ 0,15 \\ -0,37 \\ 0,006 \\ 0,03 \\ -0,33 \end{array} \right], \left[ \begin{array}{c} 0,20 \\ -0,54 \\ -0,06 \\ -0,49 \\ -0,07 \\ -0,30 \\ 0,17 \\ 0,36 \\ -0,35 \\ 0,50 \end{array} \right], \left[ \begin{array}{c} 0,34 \\ -0,32 \\ -0,39 \\ -0,15 \\ -0,45 \\ 0,16 \\ -0,07 \\ -0,46 \\ 0,41 \\ -0,44 \end{array} \right], \\ \left[ \begin{array}{c} 0,49 \\ -0,02 \\ -0,28 \\ 0,41 \\ 0,04 \\ 0,30 \\ -0,29 \\ 0,08 \\ -0,004 \\ 0,20 \end{array} \right], \left[ \begin{array}{c} 0,44 \\ 0,20 \\ -0,094 \\ 0,25 \\ -0,09 \\ -0,54 \\ 0,05 \\ 0,42 \\ -0,40 \\ -0,08 \end{array} \right], \left[ \begin{array}{c} 0,37 \\ 0,23 \\ -0,024 \\ -0,14 \\ 0,45 \\ -0,08 \\ -0,21 \\ -0,12 \\ -0,03 \\ 0,34 \end{array} \right], \left[ \begin{array}{c} 0,31 \\ 0,29 \\ 0,06 \\ -0,34 \\ 0,27 \\ -0,17 \\ 0,34 \\ -0,48 \\ 0,57 \\ -0,49 \end{array} \right], \left[ \begin{array}{c} 0,25 \\ 0,43 \\ 0,26 \\ -0,42 \\ -0,35 \\ -0,50 \\ 0,01 \\ 0,37 \\ -0,36 \\ 0,20 \end{array} \right] \end{array} \right\} \quad (4.20)$$

Ovenstående egenvektorer er ortonormale, da de er udregnet vha. Python, der automatisk omregner dem til ortonormale.

Som forklaret tidligere i afsnit 4.3 på side 36, kan  $P$  udtrykkes som  $P = E^T$ . Hermed kan  $Y$  udregnes. På dette tidspunkt i processen kan principale komponenter fjernes, hvor der typisk bibeholdes de komponenter, som udtrykker en stor andel af billedet - i praksis reducereres rækkerne af  $P$  fra  $m$  til  $k$ , hvorved dimensionen for  $P$  bliver  $k \times m$ . Det er dette trin, der komprimerer billedet, da der nu skal gemmes færre værdier for at gengive det tilnærmelsesvist samme billede. Der kan beregnes, hvor meget en enkelt principal komponent udgør af billedet ud fra den tilhørende egenværdi og summen af alle egenværdier. Eksempelvis udgør den første principal komponent i dette tilfælde  $\frac{1,61 \cdot 10^5}{1,76 \cdot 10^5} \approx 91,7\%$ , den anden  $\frac{10^4}{1,76 \cdot 10^5} \approx 5,7\%$  etc.

Ved dette regneeksempel vælges to principale komponenter til videre udregning. Typisk vælges antallet alt efter hvornår flere principale komponenter ikke ændrer billedet drastisk. Det er det essentielle ved metoden, da det ofte muliggør en fjernelse af  $n$ -dimensioner dvs. en høj data reduktion.

Videre i udregningen stilles de to valgte principale komponenter op som en  $2 \times 10$  matrix. Dimensionerne for transformationen kan opskrives som  $(k \times n) = (k \times m) \cdot (m \times n)$ . Den transformerede matrix findes vha. ligning 4.1,  $Y = P \cdot \tilde{X}$  med dimensionerne  $k \times n$ ,  $(2 \times 10) \cdot (10 \times 8) = 2 \times 8$ .

$$Y = \begin{bmatrix} 135.42 & 152.50 & 132.28 & 74.51 & 2.25 & -77.74 & -159.33 & -255.39 \\ 23.95 & -24.89 & -42.85 & -6.82 & 35.03 & 52.36 & 15.42 & -52.21 \end{bmatrix} \quad (4.21)$$

Jævnfør algoritmen på side 39, gemmes  $Y$ ,  $P$  og gennemsnitsvektoren i en komprimeret fil. Filen er komprimeret, da den mængde, der skal gemmes, er mindre efter transformationen end før. Fra 80 datapunkter,  $X : 10 \times 8$  - til 46 datapunkter,  $Y : 2 \times 8$ ,  $P : 2 \times 10$  og  $\mu_{\tilde{X}} : 10 \times 1$ . Grundet reduceringen af dimensionerne komprimeres billedet med tab. For at komprimere filen yderligere Huffmankodes den.

## Huffman

Det komprimerede billede, repræsenteret ved matricerne  $Y$  og  $P$  samt gennemsnitsvektoren  $\mu_{\tilde{X}}$  Huffmankodes, for at udtrykke det ved en binærstreg, som der blev gjort ved billedet

komprimeret med DCT i afsnit 3.1.6 på side 24.

Det bemærkes, at der ikke nødvendigvis opnås en yderligere komprimering af billedet ved Huffmankodning. Dette skyldes, at  $Y$ ,  $P$  og  $\mu_{\vec{x}}$  ikke nødvendigvis indeholder lignende værdier, som er tilfældet med billede behandlet med DCT. Det kan derimod lade sig gøre, at en Huffmankodet fil fylder mere, end filen der komprimeres uden Huffman. Der benyttes derfor udelukkende Huffmankodning til PCA for at kunne sammenligne de to datasæt fra DCT og PCA.

## Dekomprimering

Dekomprimeringen udføres vha. algoritmen på side 42.

---

### Algoritme: Dekomprimering vha. PCA

---

1. Input:	Komprimeret fil
2. Output:	Dekomprimeret billede, $\hat{X}$
<i>Dekomprimering</i>	
3. Dekodning:	Filen aflæses til $Y$ , $P$ og $\mu_{\vec{x}}$
4. Tilbage transformerer matrix	$\hat{X} = P^T Y$
5. Destandardiserer:	$\hat{X} = \hat{X} + \mu_{\vec{x}}$

Det ses at efter dekomprimeringen transformeres matricen vha.

$$\hat{X} = P^T Y \quad (4.22)$$

Dette kan lade sig gøre, grundet  $P$ s egenskaber som en højreinvertibel matrix.

En  $m \times n$  matrix  $A$ , hvor  $\text{Rank}(A) = m$ , er højreinvertibel og har en højreinvers matrix  $A^{-1}$  defineret ved [OpenCourseWare, s. 1]

$$A^{-1} = A^T (AA^T)^{-1} \quad (4.23)$$

$P$  har ortonormale rækker, og derfor gælder der, at

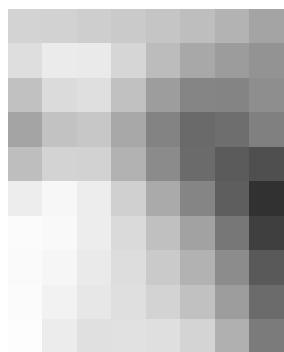
$$PP^T = I \quad (4.24)$$

Hvor  $I$  er identitetsmatricen. Derfor fremkommer, jævnfør ligning 4.23, at

$$P^T(PP^T)^{-1} = P^T \quad (4.25)$$

Det ses, at for den ikke-kvadratiske ortonormale matrix  $P$ , er den transponerede lig den inverse. Altså kan  $\hat{X}$  beregnes ved ligning 4.22. Efter dette skridt standardiseres den, dvs. den centreres tilbage omkring gennemsnittet, ved at addere alle søjler med  $\mu_{\vec{x}}$ . Resultatet af matricen ses herunder.

Regneeksemplet er lavet for at give et indblik i algoritmen for billedkomprimering med PCA. Ved metodens anvendelse på et farvebillede, gentages processen for hver farvekanal, hvilket er tilfældet ved Lena. Metoden anvendes nu på Lena, og repræsentationer af Lena med forskellige valg af principale komponenter kan ses i bilagsmappen.



$$\hat{X} = \begin{bmatrix} 212 & 210 & 207 & 203 & 197 & 190 & 179 & 164 \\ 223 & 236 & 235 & 214 & 189 & 168 & 156 & 147 \\ 193 & 219 & 223 & 194 & 158 & 133 & 132 & 142 \\ 164 & 194 & 200 & 169 & 131 & 107 & 111 & 128 \\ 190 & 211 & 210 & 179 & 139 & 108 & 91 & 80 \\ 238 & 247 & 238 & 209 & 170 & 133 & 94 & 49 \\ 252 & 250 & 237 & 219 & 193 & 163 & 119 & 63 \\ 251 & 246 & 235 & 221 & 203 & 179 & 140 & 89 \\ 252 & 243 & 231 & 224 & 212 & 193 & 157 & 108 \\ 253 & 237 & 224 & 225 & 224 & 212 & 176 & 124 \end{bmatrix}$$

**Figur 4.3.** Dekomprimeret  
- Visuel.

**Figur 4.4.** Dekomprimeret - Tal.



## KAPITEL 5

---

### RESULTATER

I det følgende afsnit, bliver resultaterne fra komprimering af Lena med hhv. DCT og PCA præsenteret. Resultaterne vil i afsnit 6 blive brugt til at sammenligne de to komprimeringsmetoder.

I forhold til den opstillede case er det interessant at kigge på forskellige parametre: komprimeringsgrad og billedekvalitet. Disse parametre er interessante, da det ønskes at nedbringe filstørrelsen, så en mobilbruger har plads til flere billeder på telefonen af gangen. Ydermere ønskes det, at brugeroplevelsen ikke skal påvirkes af komprimeringen, hvilket imidlertid betyder, at billedekvaliteten skal bibeholdes mest muligt.

Der er med hhv. DCT og PCA lavet test med forskellige grader af komprimering, hvor alle parametre er blevet målt med henblik på sammenligning af DCT og PCA som komprimeringsmetoder. Komprimeringer med DCT er lavet med følgende  $Q$ -værdier: 10, 25, 50, 75 og 90. PCA-komprimeringerne er målt med følgende antal principale komponenter: 1, 5, 10, 15, 20, 25, 50, 75, 100, 150, 200 og 512.

Der tages først udgangspunkt i komprimeringsgraden for de forskellige komprimeringer.

#### 5.1 KOMPRIMERINGSGRAD

Først ses der nærmere på DCTs og PCAs evne til at komprimere den originale fil til en mindre fil. Filstørrelsen for DCT måles i bitrepræsentationen fra en TXT-fil, der er Huffmankodet. Det er dermed ikke TXT-filens faktiske størrelse, da fokusset i dette projekt har været på matematikken bag komprimeringen og ikke datastrukturen i en binær fil. Ligeledes måles filstørrelsen af de PCA-komprimerede filer ved den totale længde af bitrepræsentationen i filen.

Den oprindelige filstørrelse betragtes som værende det totale antal pixels i alle farverum multipliceret med otte, da hvert tal repræsenteres med otte bit. Dette giver en filstørrelse på  $512 \cdot 512 \cdot 3 \cdot 8 = 6.291.456$  bit for det oprindelige billede. Komprimeringsgraden er beregnet som

$$\text{Komprimeringsgrad} = \frac{\text{Oprindelig filstørrelse}}{\text{Komprimeret filstørrelse}}$$

Tabel 5.1, 5.2 og 5.3 på den følgende side præsenterer komprimeringsgraden for de forskellige komprimeringer med de to metoder.

DCT	Q10	Q25	Q50	Q75	Q90
Bit	860.718	943.432	1.051.307	1.228.152	1.658.414
Kompressionsgrad	1:7,31	1:6,67	1:5,98	1:5,12	1:3,79

**Tabel 5.1.** Komprimeringsgrad med DCT ved forskellige  $Q$

<b>PCA</b>	<b>PC1</b>	<b>PC5</b>	<b>PC10</b>	<b>PC15</b>	<b>PC20</b>	<b>PC25</b>
<b>Bit</b>	64.385	207.719	371.527	532.286	532.286	845.598
<b>Kompressionsgrad</b>	1:97,72	1:30,29	1:16,93	1:11,82	1:9,12	1:7,44

*Tabel 5.2.* Komprimeringsgrad med PCA ved forskellige antal PC

<b>PCA</b>	<b>PC50</b>	<b>PC75</b>	<b>PC100</b>	<b>PC150</b>	<b>PC200</b>	<b>PC512</b>
<b>Bit</b>	1.606.858	2.346.277	3.066.789	4.475.223	5.836.314	13.473.311
<b>Kompressionsgrad</b>	1:3,92	1:3,39	1:2,05	1:1,41	1:1,08	1:0,47

*Tabel 5.3.* Komprimeringsgrad med PCA ved forskellige antal PC

Bemærk her er komprimeringsgraden under 1 ved 512 PC'er (fremover forkortes fornævnte som PC512). Dette skyldes at filstørrelsen tilnærmelsesvis (der ses her bort fra Huffmankodning, og at der kan kræves mere end otte bit pr. tal), kan findes ved  $((PC \cdot 512) + (PC \cdot 512) + (1 \cdot 512)) \cdot 3 \cdot 8$ , hvor det er tydeligt at ved PC256 gemmes flere data end i det oprindelige billede, hvorved at komprimeringen er ikke-eksisterende - altså forøges filstørrelsen, hvilket selvsagt ikke er ønskværdigt.

## 5.2 KVALITET

Billedekvaliteten af de komprimerede billeder skal vurderes for at sikre, at billederne ikke blot komprimeres, men at de også stadig repræsenterer det samme billede uden for mange forvrængninger. Dette kan gøres både som en subjektiv undersøgelse med en repræsentativ gruppe af mobilbrugere, og der kan objektivt undersøges parametre for kvaliteten af billedet i form af, hvor meget pixelændringerne har ændret sig, hvor mange pixels, der er ændret såvel som den maksimale og minimale ændring af disse pixelændringer. De sidstnævnte parametre giver et objektivt indblik i, hvordan billedet har ændret sig.

	<b>Ændrede pixels</b>
<b>Q10</b>	96,16 %
<b>Q25</b>	94,03 %
<b>Q50</b>	92,78 %
<b>Q75</b>	91,67 %
<b>Q90</b>	89,38 %

*Tabel 5.4.* Procentvise pixelændring, med DCT, i det dekomprimerede billede i forhold til det originale

	<b>Ændrede pixels</b>
<b>PC1</b>	98,93 %
<b>PC5</b>	98,08 %
<b>PC10</b>	97,11 %
<b>PC15</b>	96,61 %
<b>PC20</b>	96,12 %
<b>PC25</b>	95,53 %
<b>PC50</b>	94,02 %
<b>PC75</b>	92,65 %
<b>PC100</b>	91,66 %
<b>PC150</b>	88,83 %
<b>PC200</b>	86,13 %
<b>PC512</b>	80,36 %

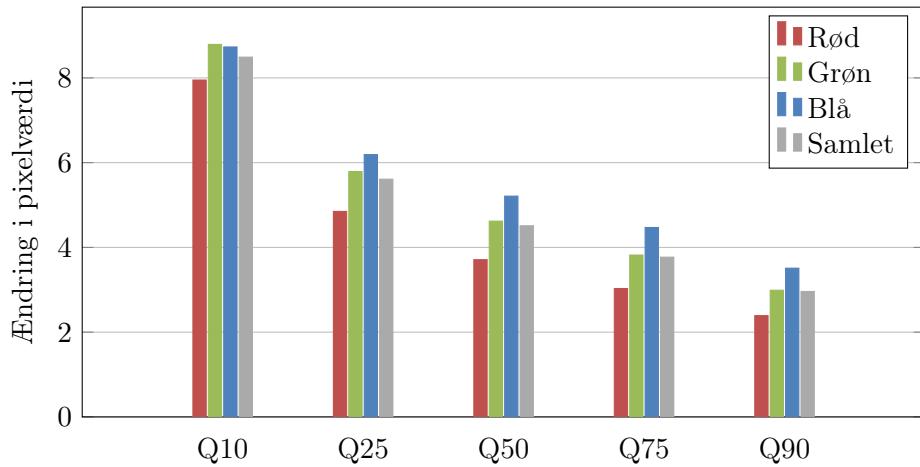
*Tabel 5.5.* Procentvise pixelændring, med PCA, i det dekomprimerede billede i forhold til det originale

I tabel 5.4 og 5.5 er der udtrykt, hvor mange af de oprindelige pixels, der er blevet ændret i en eller anden grad i det dekomprimerede billede. Tabellerne udtrykker ikke direkte noget

om graden af komprimering, eller hvor meget de enkelte pixels har ændret sig, men blot hvor mange pixels, der er blevet ændret. En procentvis ændring på ca. 90 % kan derfor være misvisende, da dette virker som en stor ændring. Parametret siger altså ikke noget om, hvor stor graden af ændring er, og det må forventes, at store dele af disse ændringer er af ubetydelig størrelse. Alternativt kunne man bestemme en tolerance, så eksempelvis kun ændringer over fem bliver noteret. Med denne modificering ville parametret kunne sige mere om ændringernes effekt.

I de pixels, hvor der er sket en ændring af intensiteten, er det interessant at kigge på ift. hvor stor en grad, de er blevet ændret. Dette har betydning for brugerens oplevelse af, hvor meget billedet er ændret efter komprimeringen. En stor ændring i pixels udtrykker stor ændring i billedet og større chance for at bemærke ændringerne end ved små pixelændringer. Typisk fremkommer lignende afvigelser i samme områder, da begge metoder forsøger at skabe glatte overgange mellem de enkelte pixels.

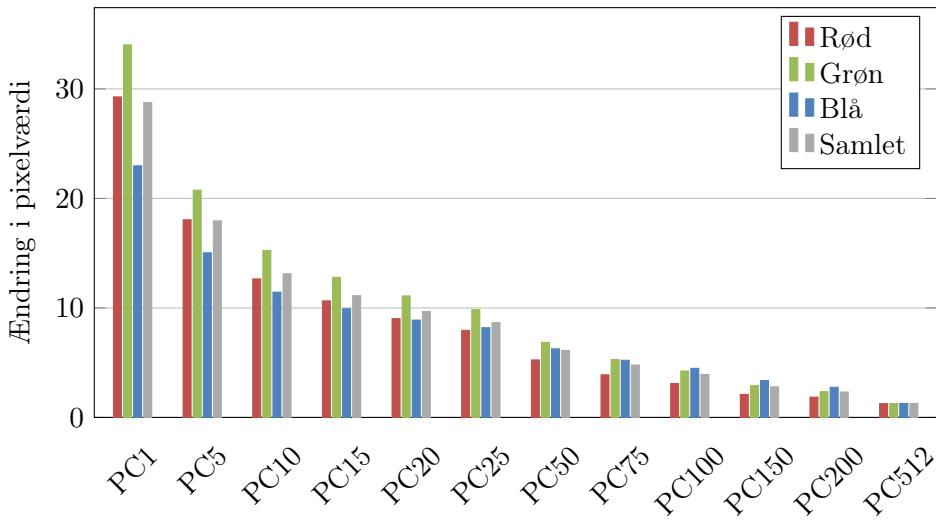
Ændring i pixelværdier er i alle komprimeringer målt i alle tre farverum. Den gennemsnitlige ændring over alle farverum er tilmed beregnet. Bemærk at det er den absolute pixelændring, der fremgår af figur 5.1 og 5.2 på næste side, hvorved der ikke differentieres mellem ændringer i positiv eller negativ retning. Ydermere er gennemsnittene beregnet ud fra antallet af pixels med ændringer, hvorved pixels uden ændring ikke medtages i disse beregninger. For overblikkets skyld repræsenteres den gennemsnitlige ændring i pixelværdi i figur 5.1 og 5.2.



**Figur 5.1.** Gennemsnitlig ændring i pixelværdi med DCT

Det kan også være interessant at se på, hvor stor den største ændring i pixelværdi i de respektive komprimeringer, er. Dette giver bl.a. et udtryk for størrelsen af den ændring, som den pixel med største ændring oplever. Enkelte ændringer, der er meget store, kan lige så vel, som en stor gennemsnitlig ændring, blive tydelig for brugeren og fremstå som fejlfarvede pixels. Det må dog forventes, at disse afvigelser typisk vil være i forbindelse med bratte overgange i billedet. Tabel 5.6 og 5.7 præsenterer den største ændring i de forskellige farverum i alle komprimeringerne. Bemærk, at disse ændringer udtrykker den *største* ændring, og dermed meget vel kun optræder meget få gange.

Begrebet SNR blev introduceret i afsnit 4.1.1 på side 33, og er et udtryk for mængden



**Figur 5.2.** Gennemsnitlig ændring i pixelværdi med PCA

	Rød	Grøn	Blå
<b>Q10</b>	110	124	110
<b>Q25</b>	69	74	104
<b>Q50</b>	48	62	87
<b>Q75</b>	33	55	70
<b>Q90</b>	22	39	43

**Tabel 5.6.** Største afvigelse for hver farve i DCT

	Rød	Grøn	blå
<b>PC1</b>	136	185	135
<b>PC5</b>	128	170	125
<b>PC10</b>	121	171	123
<b>PC15</b>	114	147	123
<b>PC20</b>	105	136	117
<b>PC25</b>	87	130	105
<b>PC50</b>	59	96	88
<b>PC75</b>	45	62	81
<b>PC100</b>	36	44	63
<b>PC150</b>	18	31	40
<b>PC200</b>	12	26	35
<b>PC512</b>	4	5	6

**Tabel 5.7.** Største afvigelse for hver farve i PCA

af støj (afvigelser) i det dekomprimerede billede i forhold til det oprindelige billede. SNR udregnes som værende  $SNR = \frac{\sigma_{\text{signal}}^2}{\sigma_{\text{støj}}^2}$ , hvor prikproduktet beregnes som:

$$\sigma_{\text{signal}}^2 = \frac{1}{mn-1} \tilde{x}_{\text{signal}} \cdot \tilde{x}_{\text{signal}}, \quad \text{hvor } \tilde{x}_{\text{signal}} = \vec{x}_{\text{signal}} - \mu_x \quad (5.1)$$

$$\sigma_{\text{støj}}^2 = \frac{1}{mn-1} (\hat{x} - \mu_{\hat{x}}) \cdot (\hat{x} - \mu_{\hat{x}}), \quad \text{hvor } \hat{x} = \vec{x}_{\text{støj}} - \vec{x}_{\text{signal}} \quad (5.2)$$

SNR kan bruges som parameter til at vurdere kvaliteten af billedet. En høj SNR indikerer, at der er lidt støj, og billedeet ligner dermed det oprindelige. Et SNR derimod lav, indikerer et stor mængde støj, og dermed også at billedeet er påvirket meget af støj, hvorved billedkvaliteten er dårlig. SNR udregnes for alle komprimeringerne i forhold til det oprindelige billede og er angivet i tabel 5.8 og 5.9.

Figur 5.3 viser, hvor mange procent hver egenværdi udgør for hele billedeet i forhold til hver enkelt principal komponent. Dette betyder også, at man får en god idé om billedeets

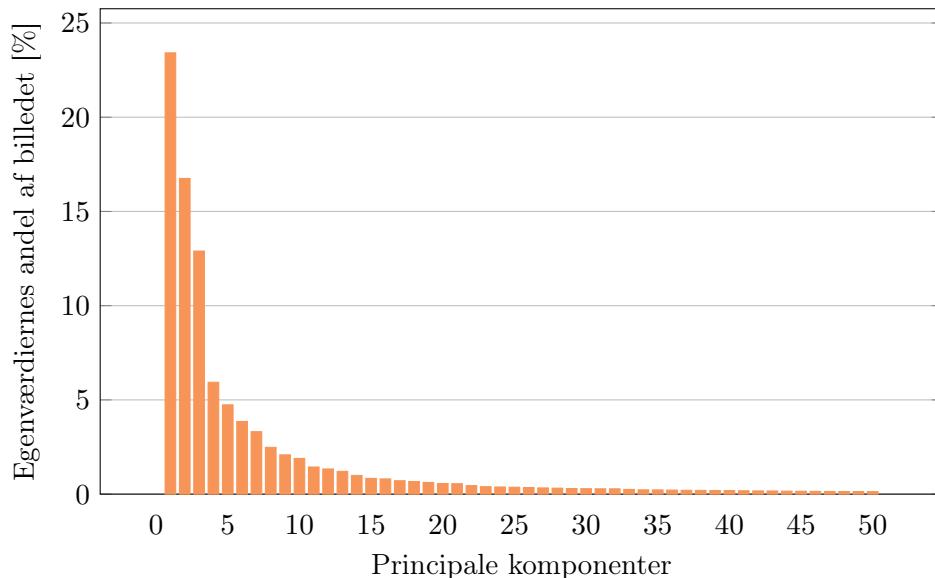
	<b>SNR</b>
<b>Q10</b>	29,77
<b>Q25</b>	66,51
<b>Q50</b>	106,58
<b>Q75</b>	161,64
<b>Q90</b>	292,41

**Tabel 5.8.** SNR for DCT-komprimeringer

	<b>SNR</b>
<b>PC1</b>	2,46
<b>PC5</b>	5,80
<b>PC10</b>	10,13
<b>PC15</b>	14,26
<b>PC20</b>	18,91
<b>PC25</b>	23,68
<b>PC50</b>	53,04
<b>PC75</b>	95,39
<b>PC100</b>	155,38
<b>PC150</b>	346,31
<b>PC200</b>	639,82
<b>PC512</b>	7440,81

**Tabel 5.9.** SNR for PCA-komprimeringer

indhold ved få principale komponenter. Diagrammet fortæller os til gengæld intet om den reelle billedkvalitet.

**Figur 5.3.** Procentudgørelse af billedet for hver egenværdi i forhold til hver principale komponent



## KAPITEL 6

---

### VURDERING

Følgende afsnit vil vurdere på resultaterne af de to komprimeringsmetoder ved komprimering af Lena og argumentere for, hvilken af de to metoder der fungerer bedst til komprimering af mobilbilleder. Parametrene, der vil blive sammenlignet, over er billedkvalitet og komprimeringsgrad. Der vil først blive opstillet tre par bestående af hhv. et DCT-komprimeret billede og et PCA-komprimeret billede. Parrene er sammensat ved enten at have samme komprimeringsgrad eller samme SNR-værdi. Konklusionerne heraf bruges til vurdering af seks testbilleder af forskellige motiver taget med en mobiltelefon. Det ønskes her at danne en mere generel konklusion af metodernes nytte.

Efterfølgende vil fordele og ulemper ved de to metoder blive fremhævet, hvilke fejlkilder de er behæftet, såvel som forslag til forbedring af metoderne brugt i rapporten.

#### 6.1 SAMMENLIGNING - LENA

Der vil i dette afsnit blive set på tre par af hhv. et DCT- og PCA-komprimeret billede. Det første par er DCT-komprimeret med Q90 og PCA-komprimeret med PC50. Dette par er sammensat på baggrund af nærliggende komprimeringsgrader, hvorved deres billedekvalitet vil blive sammenlignet. Næste par, bestående af Q50 og PC75, er sammensat grundet nærliggende SNR-værdier, der udtrykker en stor lighed i kvaliteten af billedet, hvorved komprimeringsgraden er interessant i disse par. Det sidste par har både samme komprimeringsgrad og SNR-værdi og består af Q10 og PC25.

Kvaliteten af billederne vurderes både objektivt og subjektivt for at påpege forskelle på den matematiske vurdering og den visuelle/subjektive vurdering, der unægtelig er nødvendig, når der undersøges billeder.

	<b>Q90</b>	<b>PC50</b>	<b>Q50</b>	<b>PC75</b>	<b>Q10</b>	<b>PC25</b>
<b>Komprimeringsgrad</b>	<b>1:3,79</b>	<b>1:3,92</b>	1:5,98	1:3,39	<b>1:7,31</b>	<b>1:7,44</b>
<b>% ændret pixel</b>	89,38	94,02	92,78	92,65	96,16	95,53
<b>Gns. ændring</b>	2,96	6,12	4,51	4,79	8,49	8,66
<b>Gns. ændring (R)</b>	2,39	5,25	3,71	3,89	7,95	7,95
<b>Gns. ændring (G)</b>	2,99	6,85	4,62	5,28	8,79	9,84
<b>Gns. ændring (B)</b>	3,51	6,26	5,21	4,79	8,73	8,19
<b>Max ændring (R)</b>	22	59	48	45	110	87
<b>Max ændring (G)</b>	39	96	62	62	124	130
<b>Max ændring (B)</b>	43	88	87	81	110	105
<b>SNR</b>	292,41	53,04	<b>106,58</b>	<b>95,39</b>	<b>29,77</b>	<b>23,68</b>

**Tabel 6.1.** Komprimeringsdata i sammenligningespar (fed indikerer sammenligningsparametre)

## Q90 og PC50 - komprimering

Komprimering med Q90 og PC50 har ca. samme komprimeringsgrad, hvorved de nedbringer det originale billede til ca. samme filstørrelse. Dette betyder, at kvaliteten af billederne kan sammenlignes og i dette specifikke komprimeringstilfælde konkluderes på, hvilken af metoderne, der giver bedste kvalitet. Ses der på SNR-værdierne i tabel 5.8 og 5.9 på side 49, bliver det tydeligt, at DCT leverer en betydelig bedre billedkvalitet end PCA. Dette fremkommer af en SNR, der er ca. seks gange så stor som for PCA. Dette understøttes yderligere af den gennemsnitlige ændring i pixelværdierne, hvor PCA har over dobbelt så stor en ændring. Store pixelændringer giver specielt store udslag i SNR, hvilket ses ved, at PCA-komprimeringen har betydeligt større maksimale ændringer end DCT. Der kan i ovenstående sammenligning konkluderes, at rent matematisk leverer DCT en bedre billedekvalitet end PCA ved samme komprimeringsgrad.

### Subjektiv vurdering Q90 og PC50

Rent subjektivt kan der ved sammenligning af Q90 og PC50, ses en stor forskel på billederne. Ved Q90 er det for det menneskelige øje ikke muligt at se en forskel fra det originale, og kvaliteten synes derfor uændret. Ved PC50 ses det til gengæld, at der er sket en markant ændring i billedet - billedet synes grumset. Dette skyldes ændringer i mange pixels i hele billedet. Fejlbillederne i figur 6.3 og 6.5 bekræfter dette - for Q90 ses næsten ingen fejl, da billedet er nær konstant gråt, mens der for PC50 ses tydelige ændringer i det grå billede.

De subjektive og objektive vurderinger stemmer altså overens, da Q90 producerer et billede, som både objektivt og subjektivt er af højere kvalitet end billedet produceret med PC50.

## Q50 og PC75 - SNR

Komprimering med Q50 og PC75 resulterer i en tilnærmedesvis lignende SNR-værdi ( $106,58 \approx 95,39$ ) efter komprimering.

Når SNR for de to billeder er relativt ens, kan komprimeringsgraden sammenlignes, og det fremkommer her, at Q50 komprimerer næsten dobbelt så meget som PC75. Det vurderes, at Q50 leverer samme billedekvalitet som PC75 men ved højere komprimeringsgrad.

### Subjektiv vurdering af Q50 og PC75

Ved Q50 kan det ses, at billedet er komprimeret - DCT har skabt blokke af pixels, som ikke er korrelerede, og dette kan ses i det færdige billede. Det fremgår ligeledes af fejlbilledet i figur 6.2, at der er fejl langs alle kanter, som består af bratte farveskift.

Ved billedet med PC75 er uskarpt og grumset. Det ses desuden der ved fejlbilledet, at der er lavet markante ændringer i hele billedet. Fejlbilledet ses i figur 6.6. Trods de lignende SNR'er synes billedet lavet ved Q50 at være af højere kvalitet end billedet lavet ved PC75, dog er forskellen ikke nær så stor som ved Q90 og PC50.

## Q10 og PC25 - SNR og komprimering

Q10 og PC25 er valgt på baggrund af den objektive kvalitet i form af deres SNR-værdier og samtidig en næsten ens komprimeringsgrad. Med Q10 ses det, at SNR ligger en anelse højere end PC25 på trods af at den gennemsnitlige afvigelse af pixels, som er lavere med Q10. Grunden til en større afvigelse på trods af en højere SNR-værdi ved Q10, er metoden ved beregning af SNR, hvor der fremkommer tal i anden potens. Dvs., at hvis eksemplvis blå optræder med enkelte meget høje tal ved Q10, og med PC25 lavere tal flere gange, vil de høje tal have enorm stor betydning. Derfor kan der opstå højere procentvis ændring af pixels og samtidig højere SNR-værdi ved samme metode.

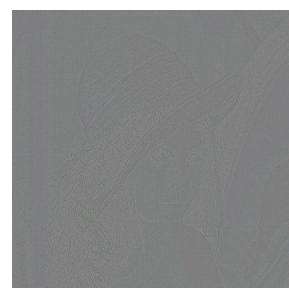
Det vurderes at ved høje komprimeringsgrader, laver metoderne billeder af tilnærmedesvis ens kvalitet.

### Subjektiv vurdering af Q10 og PC25

På begge billeder ses det tydeligt, at der er foregået en høj komprimering og dermed en forringelse af kvaliteten. DCT har lavet blokke af korrelerede pixels, men som ikke er korrelerede blokke imellem - dette ses som tydelige firkantede elementer i billedet. Billedet med 25 principale komponenter er meget utsynlig og grumset i en sådan grad, at Lenas ansigtsudtryk har ændret sig betragteligt. Fejlbilderne i figur 6.1 og 6.4 bakker det op ved meget tydelige ændringer for PC25, som kommer til udtryk som meget mørke og lyse områder i disse. Komprimeringen ved Q10 må siges at give et bedre billedkvalitetsmæssigt resultat ved den subjektive vurdering.



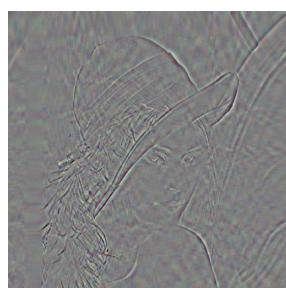
*Figur 6.1.* Støj ved Q10



*Figur 6.2.* Støj ved Q50



*Figur 6.3.* Støj ved Q90



*Figur 6.4.* Støj ved PC25



*Figur 6.5.* Støj ved PC50



*Figur 6.6.* Støj ved PC75

## Delkonklusion - Lena

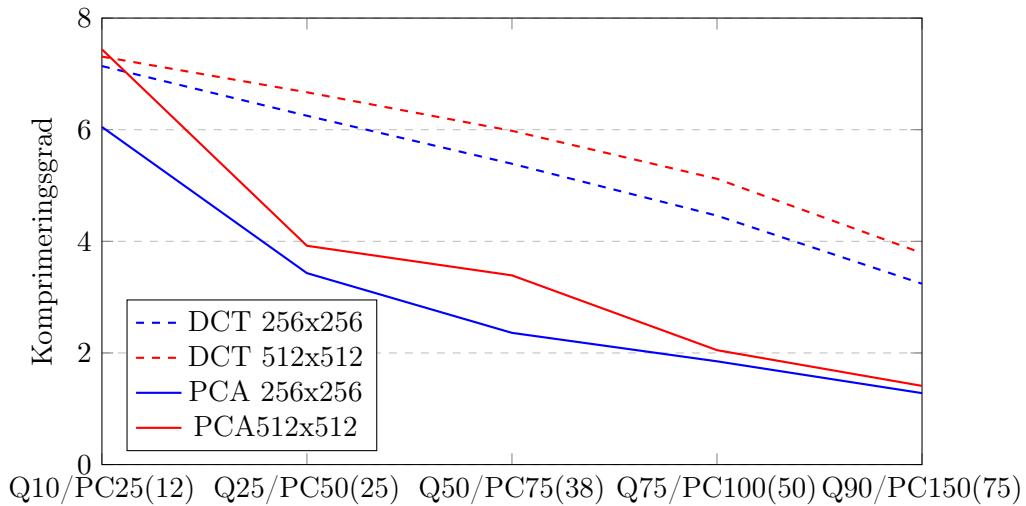
En konklusion på sammenligningen af DCT og PCA på Lena kan overvejende udtrykkes som, at begge metoder leverer omrent samme SNR-værdi, dvs. objektivt samme billedkvalitet, såvel som lignende komprimeringsgrader ved lave komprimeringskvaliteter. Dog er DCT bedre til at komprimere ved høje komprimeringskvaliteter, og samtidig holde en bedre billedkvalitet end PCA. På trods af at PCA er god til at komprimere billedet ved få principale komponenter, ses det tydeligt, at den har svært ved at komprimere særlig meget, når et vis antal principale komponenter gør sig gældende. Det er klart, da billedet ændrer sig mindre og mindre jo flere principale komponenter, der bruges. På trods af at DCT også komprimerer bedre ved lave kvaliteter, er der flere "komprimeringsgrader" ved PCA, da der ved Lena er alt fra PC1 til PC512 at benytte sig af, hvorimod DCT kun har Q1 til Q100. Altså er spændet af komprimeringsgrader for PCA større end for DCT. Hvordan dimensionerne af billedet påvirker resultatet i forhold til komprimeringsmetoden undersøges som det næste. Til sidst vil alle observationer fra Lena blive be- eller afkræftet ved en test af større billede taget med en mobiltelefon.

### 6.1.1 Skalering af Lena

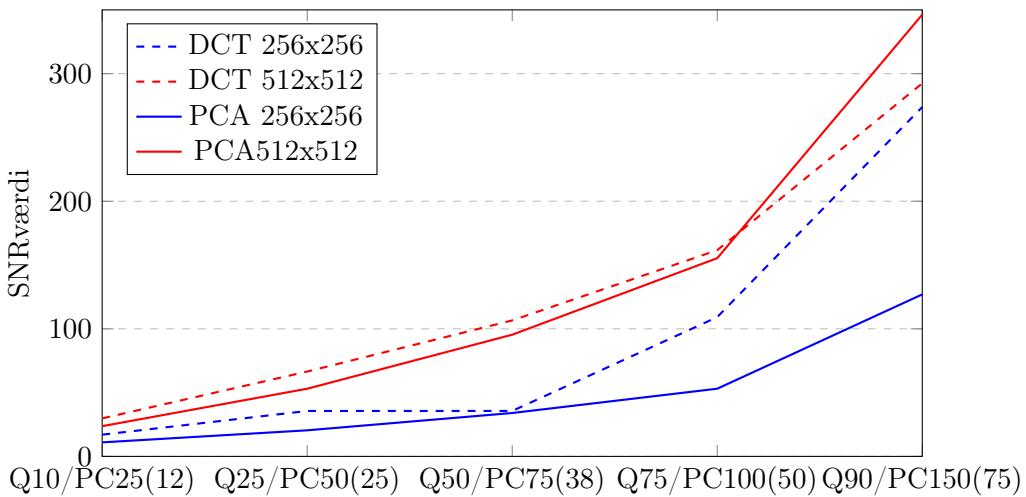
Det er interessant at kigge på, hvordan både DCT og PCA opfører sig ved forskellige størrelser af Lena. Ved at skalere billedet ned til halv størrelse, fra  $512 \times 512$  pixels til  $256 \times 256$  pixels, har billedet samme struktur men færre dimensioner. Dette kan udnyttes til at se på, om parametre som komprimeringsgraden og SNR-værdien ændrer sig drastisk ved størrelsesændring. For at dokumentere sammenhænge tilstrækkeligt, er fem forskellige kvantiseringsgrader og fem forskellige antal principale komponenter benyttet. Jævnfør figur 6.7, ses det at komprimeringen ved både DCT og PCA forholder sig nogenlunde ens. Udsvingene er grundet i, at det samme billede skal repræsenteres ved færre pixels. Dette betyder, at komprimeringen ved det mindre  $256 \times 256$  billede nødvendigvis må afvige lidt fra det større  $512 \times 512$  billede. Denne observation bekræfter os i, at størrelsen af billedet ikke har den store betydning hvad komprimeringsgraden angår - dvs. at effektiviteten forbliver den samme.

Jævnfør figur 6.8 ses igen nogle tendenser. På trods af at DCT overvejende har samme tendens, har det mindre billede væsentligt lavere SNR-værdier ved Q25, Q50 og Q75. Lignende ses det ved PCA, da det mindre billede har væsentligt lavere SNR-værdier ved alle komprimeringer uddover PC25 end det større. I begge tilfælde kan det tyde på, at antal pixels betyder en del for SNR-udregningen. Endvidere observeres en ensartethed ved både PCA og DCT ved det større billede, hvor de fungerer ens.

Tendensen for både komprimerings- og SNR-graferne er, at begge metoder fungerer respektivt ens ved mindre og større billede, dog med mindre udsving.



**Figur 6.7.** Komprimeringsgrad for DCT og PCA ved skaleret Lena



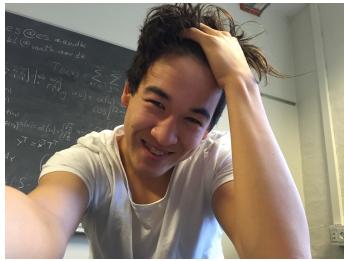
**Figur 6.8.** SNR-værdier for DCT og PCA ved skaleret Lena

## 6.2 VURDERING AF MOBILBILLEDER

Det forsøges, at generalisere konklusionen fra foregående afsnit, der i sin essens lød, at de to komprimeringsmetoder var lige gode ved lave komprimeringskvaliteter, mens DCT var betydeligt bedre ved højere kvaliteter. For at kunne generalisere denne konklusion vil seks forskellige billeder blive taget med en mobiltelefon (noteret som T1, T2, ..., T6), og vist i figur 6.9 til 6.14. Disse billeder vil blive komprimeret med de to metoder, og ved hhv. en lav, mellem og høj komprimeringskvalitet for at sammenligne resultaterne.

Billederne er taget i en opløsning på  $2448 \times 3264$  pixels, hvilket er væsentligt større end Lena med dimensionerne  $512 \times 512$  pixels. De specifikke komprimeringskvaliteter for DCT og PCA er valgt ud fra resultaterne med Lena.

Der er valgt tre par af komprimeringer, der udtrykker hhv. en lav, mellem og høj komprimeringskvalitet. Disse er bestemt til at være Q10, Q50 og Q90. PC25, PC75 og

**Figur 6.9.** Mobilbillede T1**Figur 6.10.** Mobilbillede T2**Figur 6.11.** Mobilbillede T3**Figur 6.12.** Mobilbillede T4**Figur 6.13.** Mobilbillede T5**Figur 6.14.** Mobilbillede T6

PC150 har tilnærmelsesvis samme SNR som disse ved komprimering af Lena, så disse vælges på baggrund af dette. Da mobilbillederne, som der undersøges med herunder, er langt større, skaleres antallet af principale komponenter op, så det passer til størrelsen af billederne. Dette betyder, at parrene kommer til se således ud

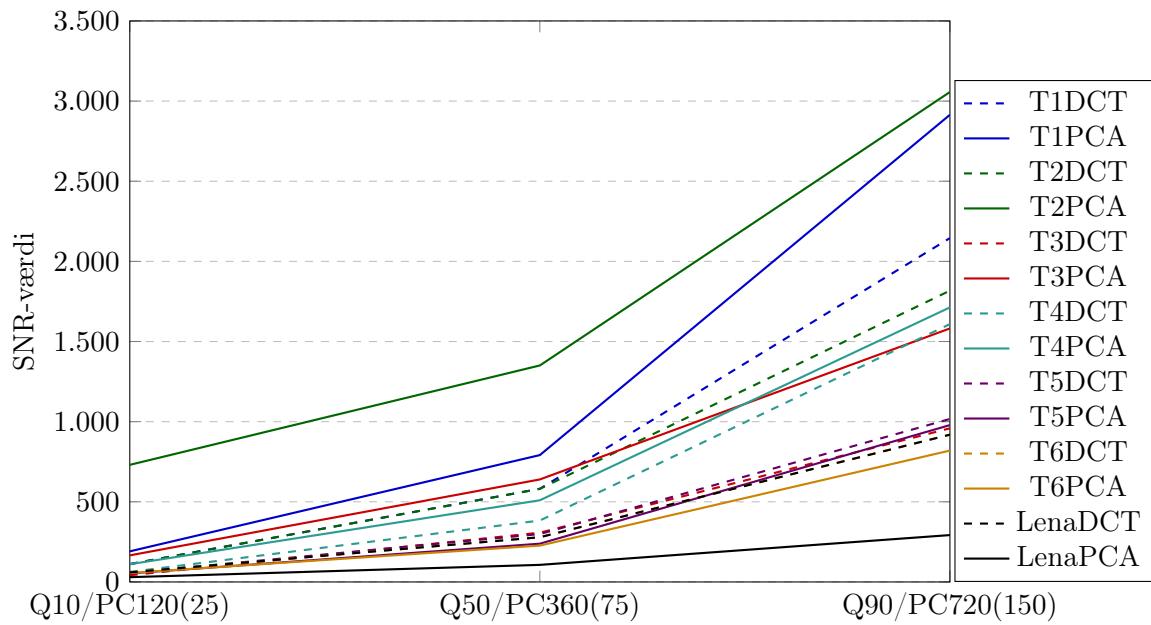
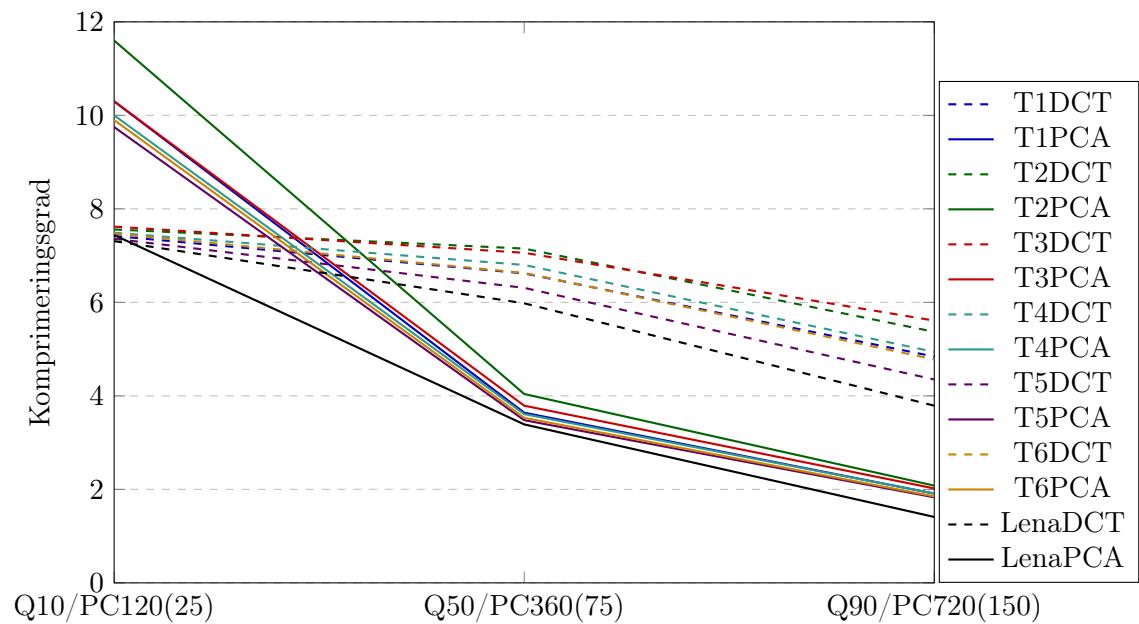
- Q10 og PC120
- Q50 og PC360
- Q90 og PC720

For hvert af parrene sammenlignes SNR og komprimeringsgrad. I figur 6.15 ses grafer for SNR for begge metoder med alle billederne. På samme måde illustrerer figur 6.16 grafer for komprimeringsgraderne. Desuden er der inkluderet en graf for komprimering af Lena udført med Q10, Q50, Q90 og PC25, PC75 og PC150 for sammenligning.

### **Q10 og PC120**

Det ses af grafen i figur 6.15, at PC120 opnår højere SNR for billede T1, T2 og T3, end DCT gør. Dette skyldes formentlig, at disse billeder indeholder få bratte ændringer - de består af mange glatte overgange, og PCA kan derfor let udtrykke meget af billedet ved få principale komponenter. DCT med Q10 opnår mindre støj i T5 og T6, hvilket er en følge af, at der er flere forskellige farver i disse billeder. For T4 opnås ca. samme SNR for de to metoder. Det ses desuden på grafen i figur 6.16, at PC120 tilmed opnår en højere komprimeringsgrad end DCT ved Q10.

På papiret er PC120 bedre end Q10, men i praksis bruges disse komprimeringsgrader sjeldent, da de laver billeder af en uønskværdig kvalitet.

**Figur 6.15.** SNR-værdier for mobilbilleder**Figur 6.16.** Komprimeringsgrad af mobilbilleder

### **Q50 og PC360**

Det ses af grafen i figur 6.15, at PCA ved PC360 stadig opnår bedre SNR end DCT med Q50 for billede T1, T2 og T3. Resultaterne for sammenligningen af SNR er tilmed magen til dem for Q10 og PC120 for de resterende billeder. Det ses dog på grafen i figur 6.16, at Q50 opnår væsentligt bedre komprimering end PC360 for alle seks billeder.

### **Q90 og PC720**

Sammenligningen af Q90 og PC720 har samme grundlæggende tendenser som den ovenstående - PCA giver for T1, T2 og T3 bedre billedkvalitet end DCT, mens DCT håndterer T5 og T6 bedre. De opnår lignende resultater for T4. DCT opnår dog imidlertid langt højere komprimeringsgrad end PCA gør og billede fylder dermed mindre, hvilket er ønskværdigt.

Bemærk, at DCT opnår både bedre SNR og komprimeringsgrad for alle par ved komprimering af Lena.

### **Delkonklusion - mobilbilleder**

Komprimering af mobilbillederne giver anderledes resultater, end komprimering af Lena gør. Ikke desto mindre opnår PCA stadig ringere komprimering end DCT ved højere kvaliteter. Dette ses dog i lyset at, at PCA opnår bedre SNR. PCA håndterer altså mobilbillederne bedre end den håndterer Lena.

At PCA er bedre ved lave kvaliteter er nærlugtlig, da det ikke er interessant at se på ringe kvaliteter - dette er derfor en negligerbar egenskab. DCT indgår et mere ønskværdigt kompromis med kvaliteten for højere komprimeringsgrad - mobilbilleder skal komprimeres meget og bibeholde kvaliteten. DCT må derfor siges at være bedre end PCA til komprimering af mobilbilleder.

## **6.3 FEJLKILDER**

I forbindelse med brug af hhv. DCT og PCA til komprimering af billeder er der nogle begrænsninger og fejlkilder, som påvirker resultatet. Det følgende afsnit vil præsentere disse fejlkilder, såvel som den indflydelse de har på resultaterne.

### **DCT fejlkilder**

Den diskrete cosinustransformation fungerer i billedkomprimeringsalgoritmer, fordi den energikomprimerer et signal effektivt. Dette er imidlertid kun tilfældet, når signalet er glat. Når der snakkes om billeder, vil det sige, at farveintensiteter i alle  $8 \times 8$  matricer er relativt ens, eller udtrykker en glat overgang. Hvis billedet ikke har denne egenskab, skal

transformationen bruge mange cosinusfunktioner til at udtrykke billedet, og der vil ikke blive opnået en god energikomprimering. Er der derfor mange bratte ændringer i billedet, vil det resultere i at DCT ikke vil være i stand til at komprimere så godt.

Netop fordi at DCT bygger på cosinustransformationer, der er svingninger, har denne metode også problemer med at udtrykke bratte ændringer korrekt. Med de 64 cosinusfunktioner (se ligning 3.1 på side 12, hvor  $n = 8$ ), der bruges i denne transformation (hvoraf dem med højeste frekvens ofte fjernes i kvantiseringen og afrundingen), er det ikke muligt at repræsentere bratte ændringer godt. Dette resulterer i at der omkring kanter, hvor der er stor ændring, typisk opstår større fejl og unøjagtigheder end i andre dele af billedet.

DCT fungerer på  $8 \times 8$  matricer, men såfremt et billede dimensioner ikke kan divideres med otte, vil den sidste matrix ikke indeholde tilstrækkelige data til at udfylde alle indgange. Der er to mulige løsninger til problemet: ikke at komprimere den yderste kant (op til syv pixels) af billede eller at nulfyld kanten, så billeddimensionerne går op i otte [Marcus, s. 1]. Komprimeres kanten ikke vil det give en dårligere komprimeringsgrad, da tallene ikke vil blive centreret omkring nul, og Huffmankodningen vil derfor ikke fungere optimalt i disse kanter. Nulfyldes kanten, hvilket svarer til at lave en grå kant, vil der i mange billeder opstå en brat ændring mellem billede og kanten, hvilket vil have betydning for komprimeringen af kanten, såvel som at det dekomprimerede billede vil være påvirket af den grå kant jævnfør tidlige fejlkilder.

### PCA fejlkilder

PCA bruges normalt til reducering af dimensioner på store datasæt og for at finde sammenhænge, der ikke nødvendigvis er tydelige, når der kigges på dataene. PCA reducerer herved dimensionerne til de mest betydende, og derved kan store dimensioner af data nedbringes til få betydende dimensioner. Det er ikke helt samme præmis, der er i spil, når PCA bliver brugt som billedekomprimeringsmetode. Her er der ikke nødvendigvis samme sammenhæng mellem de enkelte rækker af pixels, som der er mellem målinger i et datasæt, som PCA normalvis bruges på. Dette betyder, at PCA fungerer godt til at reducere dimensionerne meget, men dette resulterer ikke nødvendigvis i gode resultater visuelt. Det betyder også, at når PCA ikke reducerer dimensionerne meget, men bibeholder en betydelig mængde principale komponenter, så ændrer kvaliteten sig ikke meget, men komprimeringsgraden falder. PCA er altså ikke beregnet til billedekomprimering, men kan med acceptable resultater bruges til det.

Når det komprimerede billede gemmes til en fil, opstår der en fejlkilde. For at få sammenlignelige data bruges Huffmankodning også på PCA, men for at denne metode giver mening, skal tallene, der gemmes, være heltal. Dette er problematisk, da egenvektorerne i  $P$  er ortonormale, hvorved alle indgange er meget små for tilsammen at give en længde på én. Dette gør, at en afrunding fuldstændigt eliminerer nærmest samtlige værdier i  $P$ . Ved at multiplicere alle indgangene med 10.000 inden de afrundes og Huffmankodes, opnåes heltal på op til fire cifre. Tallene divideres blot med 10.000 i dekodningen, og afrundingen, der før ændrede alle indgange til nul, afrunder nu blot på det, der svarer til femte decimal.

Dette ændrer på dataene, der i dekomprimeringen ikke bliver de eksakt samme, men der kan ses bort fra fejlene fremkommet ved denne afrunding, da disse anses som ubetydeligt små.

### Generelle fejlkilder

Når de komprimerede billeder dekomprimeres, sker det, at de dekomprimerede pixelværdier afviger fra de tilladte værdier  $0 \leq \text{pixelværdi} \leq 255$ . Dette skyldes afrundingen af tallene i forbindelse med, at filerne bliver gemt med Huffmankodning. Dette betyder imidlertid, at alle tal under nul trunkeres til at være nul og alle tal over 255 til 255. Dette er dog i meget få tilfælde nødvendigt og er typisk tal, der i forvejen ville være endt på 255. Der vil altså ikke være en visuel tydelig forskel ved denne ændring.

Den subjektive vurdering af billedkvalitet er netop subjektiv, og det kan ikke regnes med, at denne siger noget endegyldigt om komprimeringsteknikkerne, med mindre det bakkes op af statistiske beviser i form af store datasæt fra store empiriske undersøgelser. I denne rapport er det kun gruppemedlemmerne, der har vurderet subjektivt på billederne, derfor kan der ikke siges at være statistisk belæg for konklusionerne, hvorved de blot bør anses som bemærkninger/suppleringer til den objektive vurdering.

I forbindelse med generaliseringen af delkonklusionen, der er udformet på baggrund af resultater med komprimering af Lena, sammensættes par af Q-værdier og antal principale komponenter, for at have ensformige data at vurdere på ved mobilbillederne. Disse par er dog sammensat på baggrund af metodernes effektivitet på Lena og ved andre billeder, kan disse par vise sig at være misvisende og ikke give udtryk for samme kvalitet, som de oprindeligt var udvalgt til. Denne pardannelse antages at give et nogenlunde grundlag for sammenligning.

## KAPITEL 7

---

### KONKLUSION

Følgende afsnit har til formål at konkludere på projektet og de brugte metoder i relation til den opstillede case.

De to billedkomprimeringsteknikker udarbejdet i rapporten bygger på to forskellige matematiske værktøjer - en diskret cosinustransformation og principalkomponentanalyse af et billede beregnet med brug af egenværdier. DCT udtrykker et signal som en sum af cosinusfunktioner og giver dermed et indblik i, hvordan signalet er bygget op. Når der ses på billeder, kan transformationen bruges til at identificere, hvilke informationer om billedet som er (næsten) redundante - disse kan dermed fjernes, og billedet kan komprimeres. PCA beregner egenvektorer for billedet og benytter de mest betydende af disse til at udtrykke billedet - disse hedder principale komponenter.

Det fremkom ved sammenligning af de behandlede Lena, at DCT, ved den samme komprimeringsgrad som PCA, gav et pænere billede, hvilket blev underbygget både objektivt af SNR og subjektivt. Der kan desuden konkluderes, at både DCT og PCA bliver påvirket af dimensionerne af et billede - mindre billeder giver ringere resultater for begge metoder, mens større billeder giver bedre resultater. Dette ændrer ikke på, hvordan de komprimerer i forhold til hinanden - DCT er stadig bedre end PCA.

PCA kan opnå langt flere komprimeringsgrader end DCT og er derfor mere fleksibel. Dette skyldes, at PCA oprindeligt er udviklet til at finde en sammenhæng i et datasæt og beskrive det med så få sammenhængende principale komponenter som muligt. Denne egenskab er dog ikke optimal, når et billede skal komprimeres, da hele billedet gerne skal beskrives så nøjagtigt som muligt. Dette gør, at der skal bruges et højere antal principale komponenter, end metoden var tiltænkt, for at billedet stadig kan beskrives, så det ikke er til gene for brugeren. Metoden kan med fordel bruges på billeder, som er meget ensartede, som det kan ses på resultaterne af T2. Men det er DCT, der ved subjektiv vurdering skaber det bedste billede set på resten af testbillederne. DCT er ikke i stand til at opnå samme komprimeringsgrad som PCA, men måden hvorpå den fjerner de høje frekvenser fra billedet, som til en vis grad er overflødige, betyder at den ofte danner de pæneste billeder, hvis komprimeringsgraden holdes ens med PCA.

Da formålet med rapporten er, at undersøge hvilken af de to komprimeringsalgoritmer, som bedst kunne bruges til komprimering af billeder taget med og lagret på en mobil, er det ønskværdigt at bevare billedkvaliteten i samspil med en høj komprimeringsgrad. Ud af de to metoder, PCA og DCT, viser DCT sig at være den bedste i tilfælde, hvor billedet ikke har mange glatte overgang over det hele - den er langt mere alsidig end PCA, der primært håndterer meget ensformige billeder godt.

Det kan derfor konkluderes, at komprimeringsalgoritmen, som bygger på en diskret cosinustransformation, er den bedste løsning til komprimering af billeder på en mobiltelefon

i forhold til PCA.

## 7.1 PERSPEKTIVERING

Dette projekt har været begrænset til billedkomprimering med en diskret cosinustransformation og principale komponenter i et billede. Der findes mange andre komprimeringsmetoder, tabsfri såvel som ikke-tabsfri, som ikke er undersøgt i denne rapport. I dette afsnit overvejes nogle af mulighederne, som kunne have været relevante for projektet, men som ikke er inkluderet.

### DCT

Der findes andre transformationer end en diskret cosinustransformation, som kan benyttes i en billedkomprimeringsalgoritme. Blandt disse kan nævnes

#### *Diskret Fouriertransformation (DFT)*

som allerede er blevet omtalt. Transformationen er DCT underlegen i forbindelse med billedkomprimering.

#### *Diskret Sinustransformation (DST)*

som udtrykker et singal ved sinusfunktioner, men som ikke opnår lige så høj energikomprimering af billeder som DCT.

#### *Diskret Wavelettransformation (DWT)*

som indgår i den nye og forbedrede JPEG-algoritme, JPEG2000. Transformationen er DCT overlegen og kan tilmed udføre tabsfri komprimering.

### PCA

Som tidligere bemærket kan PCA bruges til meget andet end billedkomprimering, hvor det er nyttigt at finde ligheder i store mængder data. En udbredt brug af PCA er inden for ansigtsgenkendelse, hvor ansigter udtrykkes som en sum af egenansigter komponeret af principale komponenter. Der kunne alternativt have været lavet egenansigter med PCA på DCT-komprimerede billeder, for at se hvordan komprimeringen påvirker PCAs funktionsdygtighed.

### Tabsfri

I projektet benyttes Huffmankodning, som er en tabsfri komprimeringsmetode. Denne bruges dog ikke alene til billedkomprimering, og i projektet er der ikke undersøgt en, udelukkende, tabsfri billedkomprimeringsmetode. Det kunne være interessant at undersøge eller lave en tabsfri algoritme, for at kunne sammenligne de to typer på billedkvalitet og komprimeringsgrad. Blandt tabsfrie billedformater kan nævnes PNG og TIFF.

### Farverum

I projektet undersøges farverbilleder efter RGB-farvemodellen, og komprimeringerne udføres ligeledes i denne opdeling af farverummene. Til billedkomprimering deles billeder ofte op i farverummene YCrCb (se afsnit 2 på side 5), da det med denne opdeling kan lade sig gøre at lave en mere effektiv komprimering grundet farverummernes egenskaber.

## LITTERATUR

**Bevelacqua.** Peter Bevelacqua. *The Fourier Transform .com.*  
<http://www.thefouriertransform.com/>. Besøgt 26-11-2015.

**Blinn, 2014.** James F. Blinn. *What's the Deal with the DCT?* Jim Blinn's Corner, California Institute of Technology, 2014.

**Cao.** Shuhao Cao. *Frequency Domain.*  
<http://tex.stackexchange.com/questions/127375/replicate-the-fourier-transform-time-frequency-domains-correspondence-illustrati>.  
Besøgt: 17-12-2015.

**Chui og Jiang, 2013.** Charles K. Chui og Qingtang Jiang. *Applied Mathematics: Data Compression, Spectral Methods, Fourier Analysis, Wavelets, and Applications.* ISBN: 978-94-6239-009-6. Atlantis Press, 2013.

**Jauregui.** Jeff Jauregui. *Principal component analysis with linear algebra.*  
<http://www.math.union.edu/~jauregуй/PCA.pdf>. Besøgt: 10-12-2015.

**Kumar et al.** D. Maruthi Kumar, Babu P. Prasanth og Leburu Rangaiah. *Implementation of Discrete Tchebichef Transform for Image Compression.*  
<http://jorstem.com/Proceedings/electronics/ICETSTEM201503006.pdf>. Besøgt 16-12-2015.

**Lui, a.** Lok Ming Lui. *Image compression and the Discrete Cosine Trasnform.*  
<http://www.lokminglui.com/dct.pdf>. Besøgt: 4-11-2015.

**Lui, b.** Lok Ming Lui. *The Discrete Cosine Transform (DCT): Theory and Application.*  
[http://www.lokminglui.com/DCT\\_TR802.pdf](http://www.lokminglui.com/DCT_TR802.pdf). Besøgt 11-11-2015.

**Madsen.** Tage Gutmann Madsen. *Konjugeret tal.*  
[http://www.denstoredanske.dk/It,\\_teknik\\_og\\_naturvidenskab/Matematik\\_og\\_statistik/Regning,\\_algebra\\_og\\_talteori/konjugeret\\_tal](http://www.denstoredanske.dk/It,_teknik_og_naturvidenskab/Matematik_og_statistik/Regning,_algebra_og_talteori/konjugeret_tal). Besøgt: 4-11-2015.

**Marcus.** Matt Marcus. *JPEG Image Compression.*  
<http://www.cs.dartmouth.edu/~mwm/Final%20Report.pdf>. Besøgt 12-12-2015.

**Marjanovic.** Matthew Marjanovic. *What is Y'CBCR?*  
<http://www.mir.com/DMG/ycbcr.html>. Besøgt 29-11-2015.

**Matthews.** Rick Matthews. *Digital Image File Types Explained.*  
<http://users.wfu.edu/matthews/misc/graphics/formats/formats.html>. Besøgt 11-11-2015.

**Newbold.** Curtis Newbold. *The Difference between CMYK and RGB color.*

<http://thevisualcommunicationguy.com/2013/05/14/>

the-difference-between-cmyk-and-rgb-color/. Besøgt 8-11-2015.

**OpenCourseWare.** Massachusetts Institute of Technology OpenCourseWare. *Left and right inverses; pseudoinverse.*

[http://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/positive-definite-matrices-and-applications/left-and-right-inverses-pseudoinverse/MIT18\\_06SCF11\\_Ses3.8sum.pdf](http://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/positive-definite-matrices-and-applications/left-and-right-inverses-pseudoinverse/MIT18_06SCF11_Ses3.8sum.pdf). Besøgt 17-12-2015.

**PC Mag Encyclopedia.** PC Mag Encyclopedia. *Definition of YCbCr.*

<http://www.pcmag.com/encyclopedia/term/55147/ycbcr>. Besøgt 29-11-2015.

**Rao og Yip, 1990.** K.R. Rao og P. Yip. *Discrete Cosine Transform, Algorithms, Advantages, Applications.* 0-12-580203-X. Academic Press, Inc., 1990.

**Richardson.** Mark Richardson. *Principal Component Analysis.*

<http://people.maths.ox.ac.uk/richardsonm/SignalProcPCA.pdf>. Besøgt 25-11-2015.

**Rosenkilde, 2014.** Kirsten Rosenkilde. *Grafteori,*

<http://www.georgmohr.dk/noter/grafteori2014.pdf>, 2014. Besøgt 27-11-2015.

**Sapiro.** Guillermo Sapiro. *Digital image processing: p010 - The Discrete Cosine Transform (DCT).* [https://www.youtube.com/watch?v=\\_bltj\\_7Ne2c](https://www.youtube.com/watch?v=_bltj_7Ne2c). Besøgt: 23-10-2015.

**Shlens.** Jonathon Shlens. *A Tutorial on Principal Component Analysis.*

<http://www.cs.cmu.edu/~elaw/papers/pca.pdf>.

**Society for Industrial and Applied Mathematics, a.** Society for Industrial and Applied Mathematics. *Image Compression: How Math led to the JPEG2000 Standard, Basic JPEG.* <http://www.whymath.org/node/wavlets/basicjpg.html>. Besøgt: 21-10-2015.

**Society for Industrial and Applied Mathematics, b.** Society for Industrial and Applied Mathematics. *Image Compression: How Math led to the JPEG2000 Standard, The Discrete Cosine Transformation.*

<http://www.whymath.org/node/wavlets/dct.html>. Besøgt: 21-10-2015.

**Spears og Munsil.** Spears og Munsil. *Choosing a Color Space.*

<http://handforgedvideo.com/portfolio/choosing-a-color-space/>. Besøgt 29-11-2015.

**Spence et al.** Lawrence E. Spence, Arnold J. Insel og Stephen H. Friedberg. *Elementary Linear Algebra 2015.*

**The Neurophysiological Biomarker Toolbox.** The Neurophysiological Biomarker Toolbox. *The Discrete Fourier Transformation (DFT): Definition and numerical examples.* [https://www.nbtwiki.net/doku.php?id=tutorial:the\\_discrete\\_fourier\\_transformation\\_dft](https://www.nbtwiki.net/doku.php?id=tutorial:the_discrete_fourier_transformation_dft). Besøgt 26-11-2015.

**Wakin.** Mike Wakin. *Lena.*

<http://www.ece.rice.edu/~wakin/images/lena512color.tiff>. Downloadet:  
20-10-2015.

**Wang.** Miaohui Wang. *Laboratory 1 Discrete Cosine Transform and Karhunen-Loeve Transform.* [http://www.ee.cuhk.edu.hk/~mhwang/website\\_files/eleg5431/report\\_sample.pdf](http://www.ee.cuhk.edu.hk/~mhwang/website_files/eleg5431/report_sample.pdf).

Besøgt 13-11-2015.

**Wang et al.** Ye Wang, Miikka Vilermo og Leonid Yaroslavski. *Energy Compaction Property of the MDCT in Comparison with Other Transforms.* [http://www.smcnus.org/papers/1.Audio\\_and\\_Music\\_Analysis\\_and\\_Retrieval/2001\\_Energy-Compaction\\_Property\\_of\\_the\\_MDCT\\_in\\_Comparison\\_with\\_Other\\_Transforms.pdf](http://www.smcnus.org/papers/1.Audio_and_Music_Analysis_and_Retrieval/2001_Energy-Compaction_Property_of_the_MDCT_in_Comparison_with_Other_Transforms.pdf).  
Besøgt: 30-10-2015.

**Weisstein.** Eric W. Weisstein. *Similarity Transformation.*

<http://mathworld.wolfram.com/SimilarityTransformation.html>. Besøgt  
8-12-2015.



## APPENDIKS A

---

### DCT UDLEDNING

I dette afsnit beskrives udledningen af DCT fra den diskrete Fouriertransformation med udgangspunkt i [Chui og Jiang [2013], s. 179-183].

Lad  $\vec{x} = [x_0, \dots, x_{n-1}]^T \in \mathbb{R}^n$  og  $\tilde{\vec{x}} = [\tilde{x}_0, \dots, \tilde{x}_{2n-1}]^T \in \mathbb{R}^{2n}$  således at  $\tilde{\vec{x}}$  er den forlængede  $\vec{x}$ , sådan at de er symmetrisk omkring  $n - \frac{1}{2}$ .

Definitionsmaengden for  $\tilde{x}_l$  er givet som

$$\tilde{x}_l = \begin{cases} x_l, & \text{for } l = 0, \dots, n-1 \\ x_{2n-l-1}, & \text{for } l = n, \dots, 2n-1 \end{cases} \quad (\text{A.1})$$

Qua [Chui og Jiang [2013], s. 174] er Fourier transformationen for en vektor  $\vec{x} \in \mathbb{C}^n$  givet som

$$\hat{\vec{x}} = F_n \vec{x} \quad (\text{A.2})$$

og den  $j$ 'te indgang af  $\hat{\vec{x}}$  er givet som

$$(\hat{\vec{x}})_j = \sum_{l=0}^{2n-1} \tilde{x}_l e^{-i \frac{2l\pi j}{2n}} \quad (\text{A.3})$$

$\tilde{x}_l$  bliver delt op i to summationer i forhold til definitionsmaengden for at undersøge  $x_l$  i stedet for  $\tilde{x}_l$

$$(\hat{\vec{x}})_j = \sum_{l=0}^{n-1} x_l e^{-i \frac{l j \pi}{n}} + \sum_{l=n}^{2n-1} x_{2n-l-1} e^{-i \frac{l j \pi}{n}} \quad (\text{A.4})$$

Derefter er det interessant at sikre sig, at de to summationer har samme begrænsninger. Qua A.1 ændres variablen ved ændring af summationstegnets begrænsninger. Da  $j$  er et heltal, kan Eulers identitet  $e^{-ix} = \cos(x) - i\sin(x)$  benyttes til at forkorte udtrykket

$$\begin{aligned} \sum_{l=n}^{2n-1} x_{2n-l-1} e^{-i \frac{l j \pi}{n}} &= \sum_{l=0}^{n-1} x_l e^{-i \frac{(2n-l-1)j\pi}{n}} \\ &= \sum_{l=0}^{n-1} x_l e^{-i \frac{2nj\pi}{n}} e^{-i \frac{(-l-1)j\pi}{n}} \\ &= \sum_{l=0}^{n-1} x_l \cdot 1 \cdot e^{-i \frac{(-l-1)j\pi}{n}} \\ &= \sum_{l=0}^{n-1} x_l e^{i \frac{(l+1)j\pi}{n}} \end{aligned}$$

Nu skrives de to summationstegn i A.4 sammen i henhold til ovenstående omskrivning, og  $x_l$  sættes udenfor en parentes

$$\left(\hat{\vec{x}}\right)_j = \sum_{l=0}^{n-1} x_l (e^{-i\frac{jl\pi}{n}} + e^{i\frac{(l+1)j\pi}{n}})$$

Brøkerne i potensen forlænges med  $\frac{2}{2}$ , og leddene ganges med  $e^{i\frac{j(1-1)\pi}{2n}} = 1$  for at isolere  $e^{i\frac{j\pi}{2n}}$ , som derefter sættes udenfor en parentes

$$\begin{aligned} \left(\hat{\vec{x}}\right)_j &= \sum_{l=0}^{n-1} x_l (e^{-i\frac{jl\pi}{n}} + e^{i\frac{(l+1)j\pi}{n}}) \\ &= \sum_{l=0}^{n-1} x_l (e^{-i\frac{j2l\pi}{2n}} + e^{i\frac{(2l+2)j\pi}{2n}}) \\ &= \sum_{l=0}^{n-1} x_l (e^{-i\frac{j2l\pi}{2n}} \cdot e^{i\frac{j(1-1)\pi}{2n}} + e^{i\frac{(2l+2)j\pi}{2n}} \cdot e^{i\frac{j(1-1)\pi}{2n}}) \\ &= \sum_{l=0}^{n-1} x_l (e^{i\frac{j(1-1-2l)\pi}{2n}} + e^{i\frac{(2l+2)j\pi}{2n}} \cdot 1) \\ &= \sum_{l=0}^{n-1} x_l (e^{i\frac{j(-1-2l)\pi}{2n}} \cdot e^{i\frac{j\pi}{2n}} + e^{i\frac{(2l+1)j\pi}{2n}} \cdot e^{i\frac{j\pi}{2n}}) \\ &= e^{i\frac{j\pi}{2n}} \sum_{l=0}^{n-1} x_l (e^{-i\frac{j(2l+1)\pi}{2n}} + e^{i\frac{j(2l+1)\pi}{2n}}) \end{aligned}$$

Ved hjælp af Eulers identitet, givet ved  $\cos(x) = \frac{e^{-ix} + e^{ix}}{2}$ , forsimples udtrykket til  $2\cos(x) = e^{-ix} + e^{ix}$ , hvorefter de variabler, der er uafhængige af de komplekse tal, skrives i en forskrift som navngives  $d_j$ .

$$\left(\hat{\vec{x}}\right)_j = 2e^{i\frac{j\pi}{2n}} \sum_{l=0}^{n-1} x_l \cos\left(\frac{j(2l+1)\pi}{2n}\right) = e^{i\frac{j\pi}{2n}} \cdot d_j \quad (\text{A.5})$$

hvor

$$d_j = 2 \sum_{l=0}^{n-1} x_l \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \quad (\text{A.6})$$

Da  $\hat{\vec{x}}$  er symmetrisk omkring  $n - \frac{1}{2}$ , kan den  $2n - j$ 'te indgang defineres som den kompleks konjugerede  $j$ 'te indgang [Madsen, afsnit 1].

$$\left(\hat{\vec{x}}\right)_{2n-j} = \overline{\left(\hat{\vec{x}}\right)_j}$$

Dette kan qua A.5 udtrykkes som

$$\left(\hat{\vec{x}}\right)_{2n-j} = d_j e^{-i\frac{j\pi}{2n}}$$

$j$  erstattes med  $2n - j$ , som indsættes i  $d_j$  for at finde  $d_{2n-j}$ :

$$d_{2n-j} = e^{-i\frac{(2n-j)\pi}{2n}} \left(\hat{\vec{x}}\right)_{2n-j}$$

Så bliver  $\left(\hat{\tilde{x}}\right)_{2n-j}$  erstattet med dens  $d_j$  værdi:

$$\begin{aligned} d_{2n-j} &= e^{-i(2n-j)\frac{\pi}{2n}} d_j e^{\frac{-ij\pi}{2n}} \\ &= e^{\frac{-i(2n-j)\pi - ij\pi}{2n}} \cdot d_j \\ &= e^{\frac{-i2n\pi + ij\pi - ij\pi}{2n}} \cdot d_j \\ &= e^{-i\pi} d_j \end{aligned}$$

På grund af Eulers formel  $e^{-ix} = \cos(x) - i\sin(x)$  fås følgende:

$$d_{2n-j} = -d_j \quad (\text{A.7})$$

Og når  $j = n$  i A.7 fremkommer

$$d_n = -d_n \quad (\text{A.8})$$

$$d_n = 0 \quad (\text{A.9})$$

Den inverse af DFT er givet som

$$\tilde{F}_n = \frac{1}{n} F_n^* \quad (\text{A.10})$$

hvor  $F_n^*$  er den transponerede og kompleks konjugerede  $F_n$ .

For at finde  $l$ 'te indgang af  $\tilde{x}$  bruges ligning A.10 på  $\hat{\tilde{x}}$  i den  $2n$  lange DFT

$$\left(\hat{\tilde{x}}\right)_l = \frac{1}{2n} \sum_{j=0}^{2n-1} \left(\hat{\tilde{x}}\right)_j e^{\frac{ij2l\pi}{2n}}$$

$\left(\hat{\tilde{x}}\right)_j$  fra A.5 indsættes

$$\begin{aligned} \left(\hat{\tilde{x}}\right)_l &= \frac{1}{2n} \sum_{j=0}^{2n-1} d_j e^{\frac{ij\pi}{2n}} e^{\frac{ij2l\pi}{2n}} \\ &= \frac{1}{2n} \sum_{j=0}^{2n-1} d_j e^{\frac{ij(2l+1)\pi}{2n}} \\ &= \frac{1}{2n} \sum_{j=0}^{2n-1} d_j e^{\frac{ij(l+\frac{1}{2})\pi}{n}} \end{aligned} \quad (\text{A.11})$$

Summen i A.11 opdeles i to summationer

$$\sum_{j=0}^{2n-1} = \sum_{j=0}^{n-1} + \sum_{j=n}^{2n-1}$$

Der ses nærmere på den sidste summation for at kunne omskrive denne til samme begrænsninger som første summation

$$\sum_{j=n}^{2n-1} = \sum_{j=n}^{2n-1} d_j e^{\frac{ij(l+\frac{1}{2})\pi}{n}}$$

Når  $d_n = 0$  og når  $j = n$  kan dette led udelades fra summationen og summationens begrænsninger omskrives til

$$= \sum_{j=n+1}^{2n-1} d_j e^{\frac{ij(l+\frac{1}{2})\pi}{n}}$$

For at samle summationstegnene skal begrænsningerne være de samme. Derfor sættes  $k = 2n - j$ , og benytter det til at udskifte alle  $j$ -variablerne. Dermed kommer begrænsningen af summationen til at være:

$$= \sum_{2n-k=n+1}^{2n-k=2n-1} = \sum_{k=n-1}^{k=1}$$

Nu tæller summationen oppe fra og ned. Det ændres ved at vende begrænsningerne.

$$= \sum_{k=1}^{n-1} d_{2n-k} e^{\frac{i(2n-k)(l+\frac{1}{2})\pi}{n}}$$

Der ses nu nærmere på potensen, som splittes i to led ved at gange ind i parentesen med  $(2n - k)$ . Da  $l = 0$  kan den første potens, beskrives som Eulers identitet ( $e^{ix} = \cos(x) + i \sin(x)$ ) i det sidste trin

$$e^{\frac{i(2n-k)(l+\frac{1}{2})\pi}{n}} = e^{i2(l+\frac{1}{2})\pi} e^{\frac{-ik(l+\frac{1}{2})\pi}{n}} = -e^{\frac{-ik(l+\frac{1}{2})\pi}{n}}$$

Qua A.7 og ovenstående fremkommer

$$= \sum_{k=1}^{n-1} d_{2n-k} (-e^{\frac{-ik(l+\frac{1}{2})\pi}{n}}) \quad (\text{A.12})$$

$$= \sum_{k=1}^{n-1} (-d_k) (-e^{\frac{-ik(l+\frac{1}{2})\pi}{n}}) \quad (\text{A.13})$$

Summationstegnene sættes ind i A.11, hvor begrænsningerne der går fra  $j = 1$  til  $n - 1$ .  $d_0$ , beskriver det punkt, som de to summationer ikke beskriver. Altså er det nødvendigt at have med.

$$(\tilde{\vec{x}})_l = \frac{1}{2n} d_0 + \frac{1}{2n} \sum_{j=1}^{n-1} d_j e^{\frac{ij(l+\frac{1}{2})\pi}{n}} + \frac{1}{2n} \sum_{k=1}^{n-1} (-d_k) (-e^{\frac{-ik(l+\frac{1}{2})\pi}{n}})$$

Herefter sættes  $\frac{1}{2n}$  uden for parentesen, og de to summationstegn lægges sammen ved hjælp af regnereglen  $\sum_{k=K}^n a_k = \sum_{j=K}^n a_j$ :

$$\frac{1}{2n} (d_0 + \sum_{j=1}^{n-1} d_j (e^{\frac{ij(l+\frac{1}{2})\pi}{n}} + e^{\frac{-ij(l+\frac{1}{2})\pi}{n}}))$$

Derefter ganges  $d_0$  med  $\frac{2}{2}$  og Eulers formel benyttes:

$$= \frac{1}{n} \left( \frac{d_0}{2} \sum_{j=1}^{n-1} d_j \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \right) \quad (\text{A.14})$$

Så sættes  $d_j$  og  $d_0$  definitionerne ind i ligningen igen:

$$\sum_{k=0}^{n-1} x_k \delta_{k-l} = \frac{1}{n} \left( \sum_{k=0}^{n-1} x_k + 2 \sum_{j=1}^{n-1} \left( \sum_{k=0}^{n-1} x_k \cos\left(\frac{j(k+\frac{1}{2})\pi}{n}\right) \right) \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \right)$$

$x_k$  ganges udenfor parentesen, og  $\sum_{k=0}^{n-1}$  sættes udenfor parentesen ved hjælp af regnereglen

$$\sum_{k=0}^n (a_k + b_k) = \sum_{k=0}^n a_k + \sum_{k=0}^n b_k :$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} \left( 1 + 2 \sum_{j=1}^{n-1} \cos\left(\frac{j(k+\frac{1}{2})\pi}{n}\right) \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \right) x_k$$

$\sum_{k=0}^{n-1} x_k$  isoleres, hvilket medfører at  $\delta_{k-l}$  kan beskrives som:

$$\delta_{k-l} = \frac{1}{n} \left( 1 + 2 \sum_{j=1}^{n-1} \cos\left(\frac{j(k+\frac{1}{2})\pi}{n}\right) \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \right)$$

Efter at have isoleret  $\delta_{k-l}$  defineres dens egneskaber som:

$$\delta_{k-l} = \begin{cases} 1 & \text{hvis } k - l = 0 \\ 0 & \text{ellers} \end{cases} \quad (\text{A.15})$$

Disse egneskaber er interessante fordi det er sikkert, når DCT'en skrives som en matrix, at den er ortonormal. Efter at have defineret  $\delta_{k-l}$  kan DCT'en omskrives til vektorform for  $k, l = 0, \dots, n-1$  hvor  $n$  er vektor  $a_0, \dots, a_{n-1}$ . Dette medfører at:

$$\delta_{k-l} = \vec{a}_k^T \cdot \vec{a}_l \quad (\text{A.16})$$

Hvor at  $\vec{a}_k$  og  $\vec{a}_l$  er defineret som

$$\vec{a}_k = \left[ \frac{1}{\sqrt{n}}, \sqrt{\frac{2}{n}} \cos\left(\frac{k+\frac{1}{2}\pi}{n}\right), \dots, \sqrt{\frac{2}{n}} \cos\left(\frac{(n-1)(k+\frac{1}{2}\pi)}{n}\right) \right]^T \quad (\text{A.17})$$

$$\vec{a}_l = \left[ \frac{1}{\sqrt{n}}, \sqrt{\frac{2}{n}} \cos\left(\frac{l+\frac{1}{2}\pi}{n}\right), \dots, \sqrt{\frac{2}{n}} \cos\left(\frac{(n-1)(l+\frac{1}{2}\pi)}{n}\right) \right]^T \quad (\text{A.18})$$

Herefter kan den diskrete cosinustransformation defineres som

$$DCT = [\mathbf{a}_0, \dots, \mathbf{a}_{n-1}] \quad (\text{A.19})$$

hvor  $n \geq 2$  og  $k = 0, \dots, n-1$ . Den diskrete cosinustransformation er altså en  $n \times n$ -matrix, som består af  $n$  søjlevektorer beskrevet ved A.18 hvor  $k = 0, \dots, n-1$ . Men da der bruges en  $8 \times 8$  matrix, bliver  $n = 8$ ,

$$U = \frac{1}{2} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\left(\frac{\pi}{16}\right) & \cos\left(\frac{3\pi}{16}\right) & \cos\left(\frac{5\pi}{16}\right) & \cos\left(\frac{7\pi}{16}\right) & \cos\left(\frac{9\pi}{16}\right) & \cos\left(\frac{11\pi}{16}\right) & \cos\left(\frac{13\pi}{16}\right) & \cos\left(\frac{15\pi}{16}\right) \\ \cos\left(\frac{2\pi}{16}\right) & \cos\left(\frac{6\pi}{16}\right) & \cos\left(\frac{10\pi}{16}\right) & \cos\left(\frac{14\pi}{16}\right) & \cos\left(\frac{18\pi}{16}\right) & \cos\left(\frac{22\pi}{16}\right) & \cos\left(\frac{26\pi}{16}\right) & \cos\left(\frac{30\pi}{16}\right) \\ \cos\left(\frac{3\pi}{16}\right) & \cos\left(\frac{9\pi}{16}\right) & \cos\left(\frac{15\pi}{16}\right) & \cos\left(\frac{21\pi}{16}\right) & \cos\left(\frac{27\pi}{16}\right) & \cos\left(\frac{33\pi}{16}\right) & \cos\left(\frac{39\pi}{16}\right) & \cos\left(\frac{45\pi}{16}\right) \\ \cos\left(\frac{4\pi}{16}\right) & \cos\left(\frac{12\pi}{16}\right) & \cos\left(\frac{20\pi}{16}\right) & \cos\left(\frac{28\pi}{16}\right) & \cos\left(\frac{36\pi}{16}\right) & \cos\left(\frac{44\pi}{16}\right) & \cos\left(\frac{52\pi}{16}\right) & \cos\left(\frac{60\pi}{16}\right) \\ \cos\left(\frac{5\pi}{16}\right) & \cos\left(\frac{15\pi}{16}\right) & \cos\left(\frac{25\pi}{16}\right) & \cos\left(\frac{35\pi}{16}\right) & \cos\left(\frac{45\pi}{16}\right) & \cos\left(\frac{55\pi}{16}\right) & \cos\left(\frac{65\pi}{16}\right) & \cos\left(\frac{75\pi}{16}\right) \\ \cos\left(\frac{6\pi}{16}\right) & \cos\left(\frac{18\pi}{16}\right) & \cos\left(\frac{30\pi}{16}\right) & \cos\left(\frac{42\pi}{16}\right) & \cos\left(\frac{54\pi}{16}\right) & \cos\left(\frac{66\pi}{16}\right) & \cos\left(\frac{78\pi}{16}\right) & \cos\left(\frac{90\pi}{16}\right) \\ \cos\left(\frac{7\pi}{16}\right) & \cos\left(\frac{21\pi}{16}\right) & \cos\left(\frac{35\pi}{16}\right) & \cos\left(\frac{49\pi}{16}\right) & \cos\left(\frac{63\pi}{16}\right) & \cos\left(\frac{77\pi}{16}\right) & \cos\left(\frac{91\pi}{16}\right) & \cos\left(\frac{105\pi}{16}\right) \end{bmatrix} \quad (\text{A.20})$$



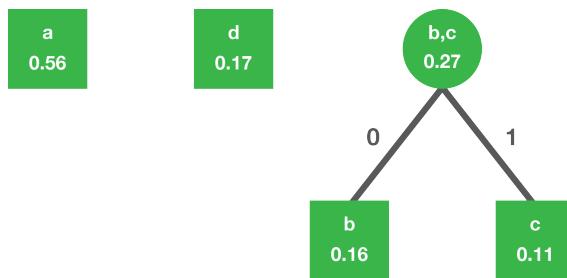
## APPENDIKS B

### HUFFMANTRÆ

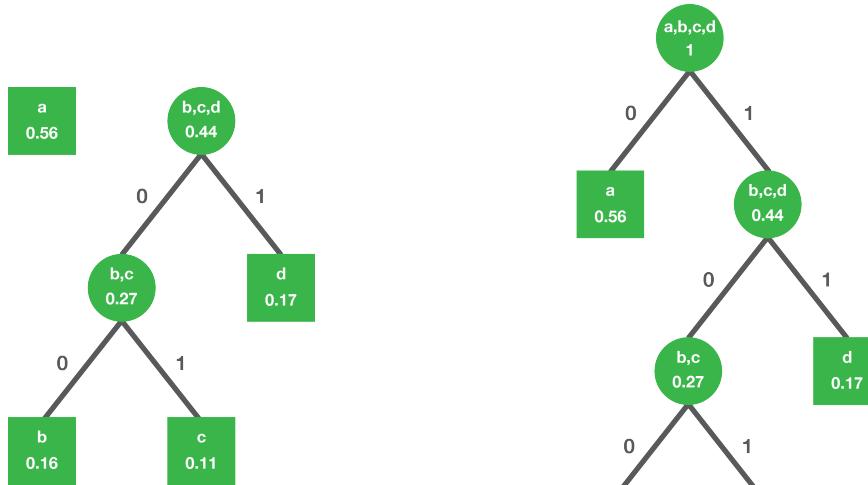
Følgende figurer beskriver trinnene i skabelsen af Huffmantræet i afsnit 3.1.6 på side 24.



**Figur B.1.** Første trin i skabelsen af Huffmantræ



**Figur B.2.** Andet trin i skabelsen af Huffmantræ



**Figur B.3.** Tredje trin i skabelsen af Huffmantræ

**Figur B.4.** Fjerde trin i skabelsen af Huffmantræ



---

## BEVIS FOR POSITIVE EGENVÆRDIER

Lad  $A$  være en  $m \times n$  matrix i  $\mathbb{R}^n$ . Det gælder, at  $A^T$  er den transponerede af  $A$ . Det ønskes i dette afsnit at vise, at matricerne  $AA^T$  og  $A^TA$  har samme positive eigenværdier. Matricerne undersøges først for at bevise, at de har samme ikke-nul eigenværdi.

### **Eigenværdierne til $AA^T$ er positive tal**

Det ønskes at bevise, at eigenværdierne til hhv.  $AA^T$  og  $A^TA$  er positive tal. Kvadratet af længden af en vektor  $\vec{w}$  er givet ved  $\|\vec{w}\|^2 = \vec{w} \cdot \vec{w} = \vec{w}^T \vec{w}$ . Lad  $\vec{v}$  være eigenvektor til  $A^TA$  med eigenværdien  $\lambda$ . Kvadratet af længden af  $A\vec{v}$  udregnes som værende

$$\|A^T\vec{v}\|^2 = (A^T\vec{v})^T(A^T\vec{v}) \quad (\text{C.1})$$

$$= \vec{v}^T(AA^T)\vec{v} \quad (\text{C.2})$$

$$= \lambda\vec{v}^T\vec{v} \quad (\text{C.3})$$

Da længder kun kan være positive, gælder der også ifølge ovenstående udledning, at  $\lambda$  er positiv. Altså er det hermed bevist, at eigenværdierne  $\lambda$  til  $AA^T$  er positive [Jauregui, s. 3].



---

## INDHOLD I BILAGSMAPPE

### Billeder

- Originalt Lenabilledede
- DCT-mappe
  - Fejlbilleder for DCT  
Fejlbilleder over forskelle mellem originalt og komprimeret billede ved Q10, Q25, Q50, Q75, Q90.
  - Lena med forskellige Q  
DCT-komprimerede billeder med Q-værdierne: Q10, Q25, Q50, Q75, Q90.
  - Skalering  
Originalt Lenabilledede skaleret ned til 256x256 pixels og DCT-komprimerede billeder ved hhv. Q10, Q25, Q50, Q75, Q90.
- Mobilbilleder
  - T1 (T2-T6 tilsvarende T1)
    - \* DCT  
T1 DCT-komprimeret ved Q10, Q50, Q90.
    - \* PCA  
T1 PCA-komprimeret ved PC120, PC360, PC720.
    - \* Originalt T1 - mobilbillede
- PCA
  - Fejlbilleder for PCA  
Fejlbilleder over forskelle mellem originalt og komprimeret billede ved PC25, PC50, PC75.
  - Lena med forskellige PC  
PCA-komprimerede billeder med antal PC: PC1, PC5, PC10, PC15, PC20, PC25, PC50, PC75, PC100, PC150, PC200, PC512.
  - Skalering  
Originalt Lenabilledede skaleret ned til 256x256 pixels og PCA-komprimerede billeder ved hhv. PC13, PC25, PC38, PC50, PC75.

### Program

- DCT (PCA har samme struktur som DCT)
  - README (indeholder oversigt over mappen)
  - Encode.py  
Input: *Billede*. Output: *Huffmankodet fil*.
  - Decode.py  
Input: *Huffmankodet fil*. Output: *Dekomprimeret billede*.
  - Variation for DCT.py  
Input: *Originalt billede*. Output: *Data om forskelle*.
- PCA