

# *Lineær algebra i billedbehandling og -kompression*

---



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:**

Billedekomprimerin

Synopsis:

Synopsis

**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: 10**

**Sidetal: 80**

**Appendiks: 3**

**Afsluttet 18-12-2014**

*Rapportens indhold er frit tilgængeligt, men offentliggørelse (med kildeangivelse) må kun ske efter aftale med forfatterne.*



# 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*.

Der arbejdes med projektkatalogets første projekt omhandlende *Billedekomprimering*, hvor der arbejdes med forskellige metoder/matematiske modeller til komprimering af et billede. Fokus ligges på komprimeringsmetoderne DCT og PCA, som bl.a. vidt brugt i JPEG-komprimering. De valgte metoder er derfor også centreret omkring udbredte jpeg-komprimeringsmetoder.

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

## Læsevejledning

Igennem rapporten vil der være kildehenvisninger, som afslutningsvist vil være samlet i en kildeliste bagerst i rapporten. Disse er opsat efter Harvardmetoden, hvorved kilder i teksten refereres med [Efternavn, År]. 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 fx første figur i kapitel 5 angives som *Figur 5.1*, den næste figur hedder derved *Figur 5.2* osv. Samme talnummering findes på tabeller og ligninger, men kun tabeller og figurer har en forklarende tekst under de respektive figurer og tabeller.

---

Frederik A. Vardinghus-Nielsen

Mike L. H. Nguyen

Thomas Kjems

---

Sebastian E. Schröder

Kenneth Borup



# Indholdsfortegnelse

---

<b>Kapitel 1 Indledning</b>	<b>1</b>
<b>Kapitel 2 Projektbeskrivelse</b>	<b>3</b>
2.1 Problemanalyse . . . . .	3
2.2 Problemformulering . . . . .	3
2.3 Metode . . . . .	4
<b>Kapitel 3 Billedkomprimering med diskret cosinustranformation</b>	<b>5</b>
3.1 Teori . . . . .	5
3.1.1 Digitale billeder . . . . .	5
3.1.2 Billedkomprimering . . . . .	5
3.1.3 JPEG . . . . .	6
3.1.4 Diskret Cosinustransformation . . . . .	7
3.1.5 Forbehandling af billede . . . . .	12
3.1.6 Kvantisering . . . . .	13
3.1.7 Entropikodning - Huffmankodning . . . . .	14
3.2 DCT anvendelse . . . . .	17
3.3 DCT udledning . . . . .	22
<b>Kapitel 4 Billedkomprimering med Principal Component Analysis</b>	<b>27</b>
4.1 Teori . . . . .	27
4.2 Anvendelse af PCA . . . . .	28
<b>Kapitel 5 Resultater</b>	<b>29</b>
<b>Kapitel 6 Vurdering</b>	<b>31</b>
6.1 Hastighed . . . . .	31
6.2 Kvalitet . . . . .	31
6.3 Størrelse . . . . .	31
<b>Kapitel 7 Konklusion</b>	<b>33</b>
<b>Litteratur</b>	<b>35</b>
<b>Appendiks A Huffmantræ</b>	<b>37</b>



# Indledning 1

---

Når man i dag skal sende et billede, eller lægge det over på sin computer og det fylder for meget, kan man bare tag det ind i et program som photoshop eller lignede og bede den om at komprimere billedet for dig, og man kan endda selv vælge hvilken kvalitet den skal komme ud i, så efter et par minutter kommer det samme billede ud men det fylder mindre. procenssen bag billed komprimeringen er i store omfang lineær algebra, altså operationer på matricer.



# Projektbeskrivelse 2

## 2.1 Problemanalyse

Teknologi udvikler sig i nutiden med rivende hast og specielt computere, smartphones, tablets m.m. har vundet utrolig frempas i det seneste årti. Dette har resulteret i flere og bedre kameraer med højere opløsning og bedre detaljegrads, hvilket ydermere betyder at den rå filstørrelse på mange fotografier er blevet større. Dette er et problem for den enkelte bruger, der oftere og ofte besidder et stort fotoalbum, der fylder deres digitale hukommelse meget hurtigt. Det kan derfor være interessant at komprimere billedet for at få plads til flere billeder, men uden at miste kvaliteten af disse billeder. Komprimeringen skal altså helst foregå uden tab af synlig kvalitet.

Problemet med store billeder strækker mere vidt end blot forbrugerens digitale enheder. Hjemmesider, og specielt billedetunge hjemmesider, bliver meget langsomme at hente, hvis ikke filstørrelsen på billederne bliver nedbragt, da store filer tager betydeligt længere tid at downloade end små filer. Billedkomprimering er specielt udbredt ved sider som Facebook, der dagligt håndterer et utal af billeder, der alle skal ligge på en server og skal deles med millioner af mennesker konstant. Her spiller størrelsen og kvaliteten af billederne en betydelig rolle.

Der findes mange forskellige komprimeringsmetoder til billeder, hvoraf JPEG formentlig er den mest udbredte metode. Joint Photographic Experts Group er bagmændende bag det udbredte filformat .jpeg/.jpg, der står for bagmændende med samme initialer. JPEG er en lossy komprimeringmetode, hvor der gives tab på data mod at opnå en mindre filstørrelse. Metoden der bruges i JPEG er tilnærmelsesvis en Diskret Cosinus Transformation (DCT), der arbejder ud fra tesen om at mennesker har sværere ved at se høje frekvenser, hvorved disse med fordeban fernes i billedet. Der vil i følgende rapport blive arbejdet med DCT, hvilke fordele og ulemper den har såvel som teste den på et billede og vurdere kvaliteten af komprimeringen. En anden metode til komprimering er Principal Component Analysis (PCA), som arbejder med egenværdier. PCA simplificerer billedet i forhold til hvilke data, der har størst betydning og fjerner dermed data af mindre betydning. Denne ekskludering af data nedbringer diversiteten i billedet og dermed nedbringes filstørrelsen af det komprimerede billede.

## 2.2 Problemformulering

I nærmeste rapport vil digitale billeder blive undersøgt og arbejdet med vha. lineær algebra. Der vil blive redegjort for JPEG-komprimeringens underliggende matematik (Diskret Cosinus Transformation) og ydermere bruges denne metode på et farvebillede.

Herefter redegøres der for matematikken bag Principal Component Analysis (PCA) og teorien anvendes på tidligere nævnte billede. Begge billede vil blive komprimeret via Huffman coding, der omdanner billeddataene til en fil. Resultaterne vurderes på baggrund af parametre som hastighed, komprimeringsgrad og kvalitet, der alle har betydning for komprimerings brugbarhed i virkeligheden og det konkluderes hvorvidt disse komprimeringsmetoder komprimerer billede tilstrækkeligt i forhold til tabet af synlig data.

### 2.3 Metode

Der bilver i rapporten ikke bliv  bejdet med alle billedekomprimeringsmetoder, men blot et udpluk af metoderne. Fokus  vil blive lagt på DCT og PCA, der begge arbejder med at fjerne data der for det menneskelige øje ikke er af stor betydning. Disse metoder vil blive brugt på billedet af Lena<sup>1</sup>, der er  $512 \times 512$  pixel stort og i farvespektret RGB. Dette billede vil blive komprimeret med begge metoder og resultaterne vil slutteligt blive sammenlignet i forhold til komprimeringsparametre; hastighed, komprimeringsgrad og billedkvalitet. Her spiller billeddtypen også en stor rolle, hvorved at resultaterne også sammenholdes med resultaterne af andre billede, for at fremhæve fordele og ulemper ved metoderne i forhold til de data de får som input



---

<sup>1</sup>FiXme Dødelige: kilde

# Billedkomprimering med diskret cosinustransformation

3



I dette afsnit udarbejdes en billedkomprimeringsalgoritme med udgangspunkt i den diskrete cosinustransformation. Algoritmen er inspireret af JPEG-algoritmen, som komprimerer billeder i .jpg-formatet.



## 3.1 Teori

### 3.1.1 Digitale billeder

Digitale billeder består af pixels. En pixel er den mindste bestanddel i et billede og er små ensfarvede områder. I gråtonebilleder beskrives en pixels farveintensitet med en værdi fra 0 til 255, hvor 0 = sort og 255 = 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 256 forskellige gråtoner.

Et farvebillede behøver flere værdier for at beskrive hver enkelt pixels farve. Der findes flere forskellige farvemodeller, til beskrivelse af farver, og i vores opgave vil vi beskæftige os med RGB-modellen (Red/Green/Blue), som beskriver en pixels farve med en tuple indeholdende 3 værdier; én for hver farves intensitet i den pågældende pixel, således: pixel =  $(R, G, B)$ . Til farvebilleder skal der altså bruges tre gange så mange værdier til at beskrive en pixels farve. Dette har den indlysende selvfølge at farvebilleder kræver mere information at beskrive.

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 (tuples for RGB). Et digitalt billede af dimensionerne  $m \times n$  kan altså beskrives ved en  $m \times n$  matrix. Et eksempel på dette ses på figur 3.2 på den følgende side.

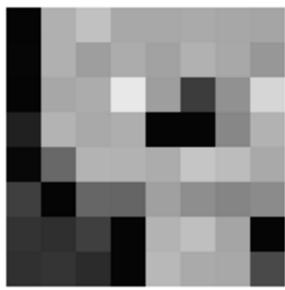


Når digitale billeder kan repræsenteres af matricer, betyder det også at de kan manipuleres ved brug af regneregler for matricer. Moderne billedbehandlingsteknikker manipulerer altså digitale billeder ved brug af regneregler for matricer - altså lineær algebra.

### 3.1.2 Billedkomprimering

Billedkomprimering er en billedbehandlingsteknik, som søger at komprimere en billedfils størrelse uden at forringe kvaliteten synligt. Når der arbejdes med digitale billeder handler dette som nævnt om matricer og manipulerer dem af disse.





**Figur 3.1.** 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 3.2.**  $8 \times 8$  matrix for billedet.

Billedkomprimeringsteknikker kan deles op i to forskellige slags: lossy og lossless.

Lossy betegner en komprimeringsteknik som under komprimeringen forårsager uigenkaldeligt tab af data fra det originale billede. Det komprimerede billede er altså kun en lignende kopi af det originale billede.

Lossless betegner en komprimeringsteknik, som under komprimeringen ikke forårsager noget tab af data fra det originale billede. Det originale billede kan derfor gendannes fuldstændigt ud fra de komprimerede data.

I vores rapport udarbejdes en lossy komprimeringsalgoritme på baggrund af det populære format .jpg, som gør brug af en diskret cosinus transformation i sin algoritme.



### 3.1.3 JPEG

Joint Photographic Experts Group udgav i 1992 den første JPEG-standard, som komprimerede og dekomprimerede billeder efter en bestemt algoritme. Et sådan program kaldes en codec. Standarden er open source og er siden 1992 blevet forbedret flere gange. Vi vil her beskæftige os med en simpel og tidlig version.

JPEG gør brug af fire skridt for at komprimere et billede fra dets fulde størrelse til en fil af mindre størrelse og de fire samme skridt i omvendt rækkefølge for at dekomprimere en fil til et billede [Society for Industrial and Applied Mathematics, a].

1. Forbehandling af billede
2. Diskret cosinus transformation
3. Kvantisering
4. Entropikodning

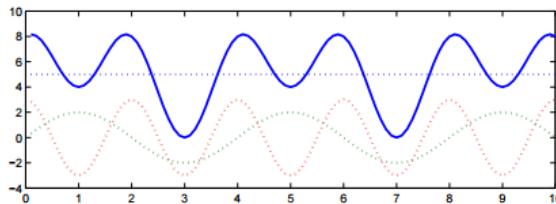
Algoritmen tager et billede  $A$ , transformerer det ved cosinustransformationen, smider mi  
dre betydelige data væk og inviterer cosinustransformationen for at danne det behandlede billede. Det originale billede kaldes  $A$  mens det behandlede billede kaldes  $A'$ . Hvis et billede behandles af algoritmen og derefter genskabes ved den inverse algoritme fås  $A'$ .



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

### 3.1.4 Diskret Cosinustransformation

En diskret cosinustransformation er en lineær transformation som afbilder en n-dimensionel vektor fra domænet  til kodomænet  $R^n$ . Basisvektorerne i transformationen udledes fra cosinusfunktioner med forskellige frekvenser. Som følge af dette bliver den  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 på figur 3.3. Der ses bort fra x- og y-aksen, som ikke nødvendigvis passer på vores beregninger. DCT'en bruges i udstrakt grad i signalbehandling på grund af dens høje energikomprimering. Energikomprimering er en transformations evne til at udtrykke et signals koefficienter ved få koefficienter i frekvensdomænet. DCT'en kan altså komprimere meget signalværdi ned til få værdier i frekvensdomænet [Ye Wang, Miikka Vilermo and Leonid Yaroslavski]. Transformationen er udledt  den diskrete Fouriertransformation (DFT), som fungerer på lignende vis. DCT adskiller sig fra DFT ved blandt andet kun at arbejde med reelle tal og ikke komplekse tal. Desuden er cosinustransformationen symmetrisk. Dette gør at den har bedre energikomprimering end DFT'en [Ye Wang, Miikka Vilermo and Leonid Yaroslavski]. Da DCT er en lineær transformation, kan den bruges i billedbehandling til



**Figur 3.3.** Funktion (blå) og bølgefunktioner (stippled) brugt til at udtrykke funktionen.

at energikomprimere digitale billeder, som netop er matricer. Den er imidlertid ikke den eneste eller bedste transformation til energikomprimering af billeder. En anden transformation er Karhunen-Loéve transformationen, som komponerer den optimale transformation ved at være specifikt lavet til det enkelte signal. Dette er dog omfattende og tidskrævende og dermed ikke smart i praksis. DCT giver lignende, og i nogle tilfælde præcis samme resultater som Karhunen-Loéve, men er meget hurtigere at arbejde med. Derfor er DCT at foretrække [Guillermo Sapiro af Duke university].



Der findes otte forskellige cosinustransformationer. I denne rapport benyttes DCT-II og DCT-III. DCT-II kaldes i daglig tale DCT, da det er den mest anvendte af cosinus transformationerne. DCT-III er den inverse DCT-II og kaldes også den inverse diskrete cosinus transformation (IDCT)

Guillermo Sapiro af Duke university viser at transformationen ser således ud 

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

Hvor

- $T_{(i,j)}^{II}$  = indgang  $(i, j)$  i  transformerede matrix ved DCT-II
- $f(x,y)$  = indgang  $(x, y)$  i  A
- n = matricens størrelse  $n \times n$

$$\bullet \alpha(i) = \alpha(j) = \begin{cases} \sqrt{\frac{1}{n}} & \text{hvis } i = 0 \\ \sqrt{\frac{2}{n}} & \text{hvis } i \neq 0 \end{cases}$$



Det er værd at bemærke at indgangene i matricen går fra  $(0, 0)$  til  $(n - 1, n - 1)$  i en  $n \times n$  matrix. Det ses at transformationen kan bruges på en hvilken som helst kvadratisk matrix af størrelsen  $n \times n$ . Det kan altså lade sig gøre at transformere et helt billede af gangen, såfremt billedet er kvadratisk. Dette gøres dog ikke i praksis, da det af flere forskellige grunde har vist sig at den mest effektive anvendelse af cosinustransformationen er ved transformation af matricer af størrelsen  $8 \times 8$ .



Blokke af  $2 \times 2$  pixels indeholder ikke nok data for DCT at arbejde med og når transformationen skal udtrykke matricen som funktioner af cosinusbølger, er der ikke nok cosinusbølger til at kunne udtrykke billedmatricen præcist - kun 4. Resultatet er at en masse data bliver smidt væk og at billedet mister for meget kvalitet - det kan ses med det menneskelige øje. Dette skyldes trasnformationens høje energikomprimering [Guillermo Sapiro af Duke university].

Blokke af  $4 \times 4$  lider under samme problem som  $2 \times 2$  dog i mindre grad.

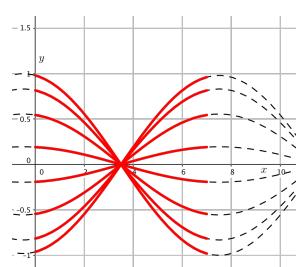
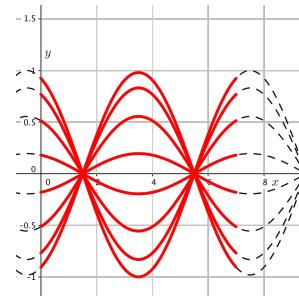
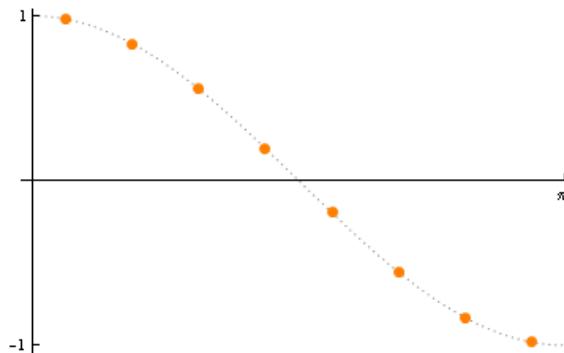
Blokke af  $8 \times 8$  pixels har en størrelse der gør cosinustransformationen effektiv. Der er nok indgange og dermed cosinusfunktioner til at kunne udtrykke matricen præcist. Desuden kræver energikomprimeringen at flere pixels kan repræsenteres af enkelte værdier og det er derfor vigtigt at de nærliggende pixels ligner hinanden. Hvis dette er tilfældet, kan blokken komprimeres. Altså er der nok cosinusfunktioner og nok sammenhæng mellem de enkelte pixels til at  $8 \times 8$  er en fornuftig størrelse [Guillermo Sapiro af Duke university]. Årsagerne til ikke at forsætte videre end  $8 \times 8$  til f.eks.  $16 \times 16$  er flere; det er tidsmæssigt langt mere effektivt at udføre komprimeringen på mindre blokke - en computer skal bruge færre beregninger, komprimeringsalgoritmen arbejder ud fra princippet om at nærliggende pixels ligner hinanden. Cosinustransformationens energikomprimering fungerer bedst, når der er sammenhæng mellem de pixels som transformeres - så kan flere pixels udtrykkes ved få værdier i frekvensdomænet. Problemet med at bruge større blokke end  $8 \times 8$  er at der ikke nødvendigvis er nogen sammenhæng mellem de enkelte pixels i en stor blok. Det bliver meget sandsynligt at billedet tydeligt skifter farve over et stort område og det giver derfor ikke mening at prøve at sammenligne disse pixels, da der sandsyndligvis ikke er nogen sammenhæng mellem dem [Guillermo Sapiro af Duke university].

Altså transformeres kun matricer af størrelse  $8 \times 8$  af gangen. Disse matricer, som er kvadratiske blokke af billedet, vil fremover blive kaldt A.

Det ses af ligning 3.1 på forrige side at hver transformerede 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 8 søjler hvor  $y$  går fra  $0 - 7$ . Dette giver koefficenter der alle ligger på en cosinuskurve. Hver gang en ny række  $i$  påbegyndes, multipliceres kurven med en ny koefficient og kurvens amplitude ændres. Når en ny transformerede indgang  $T(i, j)$  beregnes, ændres  $i$  og/eller  $j$  og frekvensen af cosinusfunktionerne i transformen bliver højere. Altså bliver hver transformerede indgang et unikt aftryk af forskelligt svingende cosinusfunktioner og værdierne i billedmatricen A.



Bølgerne for de 8 rækker for henholdsvis  $T(1, 1)$  og  $T(1, 2)$  ses på figurerne 3.4 og 3.5 på næste side.

**Figur 3.4.** Bølger for hver række for  $T(1, 1)$ .**Figur 3.5.** Bølger for hver række for  $T(1, 2)$ .**Figur 3.6.** Koefficienterne for  $T(1, 1)$  og  $f(0, y)$  afmærket som punkter på en cosinusbølge.

Det ses desuden på figur 3.6 at koefficienterne tilsammen har en sum = 0. Der er altså lige mange positive og negative koefficienter i alle bølgerne med untagelse af den første, som udelukkende består af koefficienten  $\frac{\pi}{2}$ .



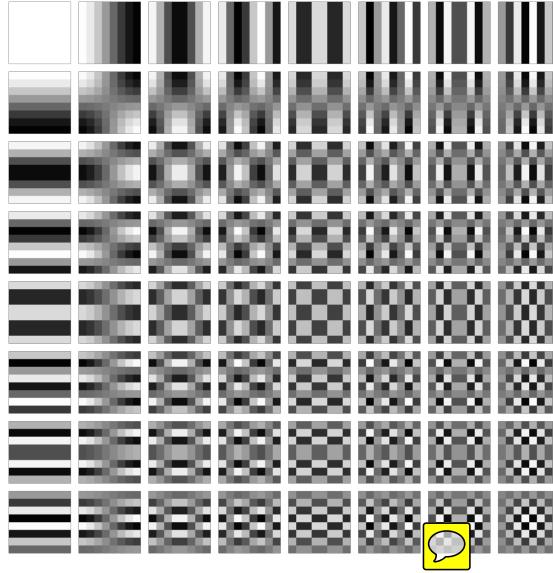
At hele den første række kun indeholder positive indgange bidrager til energikomprimeringen. Hvis A indeholder lignende værdier vil indgang  $(0, 0)$  have stor numerisk værdi. Netop fordi vi ser på en lille del af billedet regner vi med at de 64 pixels ligner hinanden. Derfor kan en stor del af informationerne komcentreres i den ene indgang. På samme måde vil resultatet af at prikke en konstant vektor med en hvilken som helst samling af koefficenter fra en cosinusbølge beregnet ved  $i = 1, \dots, 7$  blive lig 0. Resultatet af at prikke en hvilken som helst næsten konstant vektor med koefficienterne vil blive tæt på eller lig 0 [Society for Industrial and Applied Mathematics, b].

Indgang  $(0, 0)$  kaldes DC koefficienten mens de restante 63 kaldes AC koefficienter. De transformerede koefficienter er koefficienter for sort/hvid mønstre tegnet ud fra svingningerne i cosinusfunktionerne. Mønstrene for hver indgang ses på figur 3.7 på næste side, hvor bølgetoppene og -dalene (og dermed negative og positive koefficienter) er henholdsvis sort og hvid. Denne figur illustrerer hvordan transformationen behandler forskellige slags billedmønstre i billedet - mønstre som ligner billedet vil have høje koefficenter. Bemærk at DC koefficienten er koefficient for et ensfarvet mønster.



Indgangene i B er koefficienterne hvorved disse mønstre er repræsenteret i A. En lineær kombination af mønstrene og koefficienterne vil producere A - A kan illustreres ved brug af alle mønstrene ved forskellige intensiteter.

I vores opgave udføres beregninger ved brug af Python. Til dette er det beregningsmæssigt nemmere og hurtigere at regne med DCT'en på matrixform. Ligning 3.2 producer indgang



**Figur 3.7.** Repræsentation af de 64 forskellige frekvensmønstre i cosinus transformationen.

$(i, j)$  i en matrix af størrelse  $n \times n$  som kan bruges på samme måde som transformationen ved 3.1 på side 7. Matricerne indeholder de tidligere omtalte koefficienter som bl.a. ses på figur 3.6 på foregående side.

$$y_{(i,j)} = C(i) \sqrt{\frac{2}{n}} \cos \left( \frac{(2 \cdot j - 1) \cdot (i - 1)\pi}{2n} \right) \quad (3.2)$$

Hvor

- $C(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{hvis } i = 1 \\ 1 & \text{hvis } i \neq 1 \end{cases}$
- $y_{(i,j)}$  = indgang  $(i, j)$
- $n$  = matricens dimensioner  $n \times n$

Matricen beregnet ved formel 3.2 hvor  $n = 8$  ses på figur 3.3. I beregningerne bruges desuden den transponerede matrix, som ses på figur 3.4 på næste side.

$$U = \frac{1}{2} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \cos(\frac{\pi}{16}) & \cos(\frac{3\pi}{16}) & \cos(\frac{5\pi}{16}) & \cos(\frac{7\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{11\pi}{16}) & \cos(\frac{13\pi}{16}) & \cos(\frac{15\pi}{16}) \\ \cos(\frac{2\pi}{16}) & \cos(\frac{6\pi}{16}) & \cos(\frac{10\pi}{16}) & \cos(\frac{14\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{22\pi}{16}) & \cos(\frac{26\pi}{16}) & \cos(\frac{30\pi}{16}) \\ \cos(\frac{3\pi}{16}) & \cos(\frac{9\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{27\pi}{16}) & \cos(\frac{33\pi}{16}) & \cos(\frac{39\pi}{16}) & \cos(\frac{45\pi}{16}) \\ \cos(\frac{4\pi}{16}) & \cos(\frac{12\pi}{16}) & \cos(\frac{20\pi}{16}) & \cos(\frac{28\pi}{16}) & \cos(\frac{36\pi}{16}) & \cos(\frac{44\pi}{16}) & \cos(\frac{52\pi}{16}) & \cos(\frac{60\pi}{16}) \\ \cos(\frac{5\pi}{16}) & \cos(\frac{15\pi}{16}) & \cos(\frac{25\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{45\pi}{16}) & \cos(\frac{55\pi}{16}) & \cos(\frac{65\pi}{16}) & \cos(\frac{75\pi}{16}) \\ \cos(\frac{6\pi}{16}) & \cos(\frac{18\pi}{16}) & \cos(\frac{30\pi}{16}) & \cos(\frac{42\pi}{16}) & \cos(\frac{54\pi}{16}) & \cos(\frac{66\pi}{16}) & \cos(\frac{78\pi}{16}) & \cos(\frac{90\pi}{16}) \\ \cos(\frac{7\pi}{16}) & \cos(\frac{21\pi}{16}) & \cos(\frac{35\pi}{16}) & \cos(\frac{49\pi}{16}) & \cos(\frac{77\pi}{16}) & \cos(\frac{91\pi}{16}) & \cos(\frac{105\pi}{16}) & \end{bmatrix} \quad (3.3)$$

$$U^T = \frac{1}{2} \begin{bmatrix} \frac{2}{\sqrt{2}} & \cos(\frac{\pi}{16}) & \cos(\frac{2\cdot\pi}{16}) & \cos(\frac{3\cdot\pi}{16}) & \cos(\frac{4\cdot\pi}{16}) & \cos(\frac{5\cdot\pi}{16}) & \cos(\frac{6\cdot\pi}{16}) & \cos(\frac{7\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{3\cdot\pi}{16}) & \cos(\frac{6\cdot\pi}{16}) & \cos(\frac{9\cdot\pi}{16}) & \cos(\frac{12\cdot\pi}{16}) & \cos(\frac{15\cdot\pi}{16}) & \cos(\frac{18\cdot\pi}{16}) & \cos(\frac{21\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{5\cdot\pi}{16}) & \cos(\frac{10\cdot\pi}{16}) & \cos(\frac{15\cdot\pi}{16}) & \cos(\frac{20\cdot\pi}{16}) & \cos(\frac{25\cdot\pi}{16}) & \cos(\frac{30\cdot\pi}{16}) & \cos(\frac{35\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{7\cdot\pi}{16}) & \cos(\frac{14\cdot\pi}{16}) & \cos(\frac{21\cdot\pi}{16}) & \cos(\frac{28\cdot\pi}{16}) & \cos(\frac{35\cdot\pi}{16}) & \cos(\frac{42\cdot\pi}{16}) & \cos(\frac{49\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{9\cdot\pi}{16}) & \cos(\frac{18\cdot\pi}{16}) & \cos(\frac{27\cdot\pi}{16}) & \cos(\frac{36\cdot\pi}{16}) & \cos(\frac{45\cdot\pi}{16}) & \cos(\frac{54\cdot\pi}{16}) & \cos(\frac{63\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{11\cdot\pi}{16}) & \cos(\frac{22\cdot\pi}{16}) & \cos(\frac{33\cdot\pi}{16}) & \cos(\frac{44\cdot\pi}{16}) & \cos(\frac{55\cdot\pi}{16}) & \cos(\frac{66\cdot\pi}{16}) & \cos(\frac{77\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{13\cdot\pi}{16}) & \cos(\frac{26\cdot\pi}{16}) & \cos(\frac{39\cdot\pi}{16}) & \cos(\frac{52\cdot\pi}{16}) & \cos(\frac{65\cdot\pi}{16}) & \cos(\frac{78\cdot\pi}{16}) & \cos(\frac{91\cdot\pi}{16}) \\ \frac{2}{\sqrt{2}} & \cos(\frac{15\cdot\pi}{16}) & \cos(\frac{30\cdot\pi}{16}) & \cos(\frac{45\cdot\pi}{16}) & \cos(\frac{60\cdot\pi}{16}) & \cos(\frac{75\cdot\pi}{16}) & \cos(\frac{90\cdot\pi}{16}) & \cos(\frac{105\cdot\pi}{16}) \end{bmatrix} \quad (3.4)$$

Disse to matricer kan, som ligning 3.1 på side 7 også bruges til at opnå transformationen. Her prikkes  $U$  med  $A$  og derefter prikkes dette resultat med  $U^T$ . Transformationen ses i ligning 3.5.

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



Hvor

- $B$  = den transformerede matrix
- $U$  = DCT-matricen
- $A$  =  $8 \times 8$  matrix fra pixels
- $U^T$  = den transponerede  $U$



Denne formel vil have samme effekt på  $8 \times 8$ -blokken af pixels, som ligning 3.1 på side 7. Det er værd at bemærke at  $U$  er orthogonal og dermed gælder følgende lighed

$$U \cdot U^T = I_8 \quad (3.6)$$



altså

$$U^T = U^{-1} \quad (3.7)$$

Fremover vil vi kalde den transformerede matrix for  $B$ .

Den diskrete cosinus transformation er fuldstændig invertibel. Når en komprimeret fil skal dekomprimeres skal processen inverteres. Den inverse proces ses i ligning 3.8.

$$A = U^T \cdot B \cdot U \quad (3.8)$$

For at nå frem til denne ligning benyttes den associative lov om matrixmultiplikation qua Lawrence E. Spence, Arnold J. Insel og Stephen H. Friedberg [2008]

$$A \cdot (C \cdot P) = (A \cdot C) \cdot P \quad (3.9)$$

for matrixdimensioner

- $A = k \times m$
- $C = m \times n$
- $P = n \times p$

~~Da vores matricer er kvadratiske opstår der ingen problemer med matrixmultiplikationer i forbindelse med dimensioner af matricerne.~~ Dermed opskrives følgende ligning ved brug af ligning 3.5

$$A = U^T \cdot (U \cdot A \cdot U^T) \cdot U \quad (3.10)$$

som ved brug af den associative lov om matrixmultiplikation kan omskrives til

$$A = (U^T \cdot U) \cdot A \cdot (U^T \cdot U) \quad (3.11)$$

Da  $U$  er orthogonal omskrives der qua 3.6 på foregående side

$$A = I_8 \cdot A \cdot I_8 \quad (3.12)$$

$$= A \cdot I_8 \quad (3.13)$$

$$= A \quad (3.14)$$

Hermed er det bevist at ligning 3.8 på forrige side inverterer transformationen i 3.5 på foregående side.



### 3.1.5 Forbehandling af billede

Før billedet transformeres ved den diskrete cosinus transformation skal det forbehandles. Baggrunden for dette er forklaret i afsnit 3.1.4 på side 7. Dette indebærer altså at dele billedet op i "blokke" af  $8 \times 8$  pixels - kvadratiske matricer med 64 indgange.

Figur 3.1 på side 6 viser et eksempel på en  $8 \times 8$  pixelblok. Bemærk at denne blok kun består af pixels i gråtoner. Hele billedet deles op i tilsvarende blokke.

For billeder i gråtoner kan billedet deles op som i figur 3.1 på side 6. For billeder i RGB-farveformatet beskrives hver pixel ved en tuple med én værdi for hver farve - altså tre værdier. Her deles hver tuple op og der laves A-matricer for hver af farverne. Dette giver pixelblokke i gråtoner som har samme farveintensitet som respektive farve der blev brugt til at lave pixelblokken. Efter dette fortsættes som ved blokke i gråtoner.

Efter opdelingen af billedet i blokke, trækkes 128 fra samtlige indgange. Dette gøres for at ændre værdierne fra beliggende i intervallet  $[0; 255]$  til  $[-128; 127]$  og dermed centrere dem omkring 0. Dette er ønskværdigt for komprimeringen, da kvantiseringsskridtet dermed efter cosinustransformationen formår at skabe flere ens koefficienter, hvilket gør komprimeringen mere effektiv.

Såfremt 8 ikke går op i billedets dimensioner (både højde og bredde) fyldes de ufuldendte  $8 \times 8$  matricer ud med 0'er. Dette kaldes zero padding [Harvard Mudd College].

Zero padding er nødvendigt, da vi bruger DCT på matricer af størrelse  $8 \times 8$ . Ikke altid vil billedet have dimensioner, som passer perfekt til disse matricer og der fyldes ud med 0'er. Dette giver en unaturlig "kant" i indgangene i matricerne og effektivt ser det ud som om billedet i disse områder er gået, da der før zero padding er trukket 128 fra samtlige værdier. Dette vil have indflydelse på transformationen og dermed hele komprimeringsalgoritmen. Vi vil nøjes med at påpege problemet, men ikke lave en bedre løsning end zero padding. Det er i det hele taget et grundlæggende problem i JPEG-algoritmen at den ikke behandler store skift i farveintensitet godt.

Forbehandlingsskridtet inverteres simpelt at danne billedet ud fra de behandlede pixelblokke igen. For farvebilleder samles de tre gråtonerepræsentationer igen til én pixelblok i farver før det samlede billede dannes ud fra blokkene.

Efter forbehandlingen har man en samling af blokke bestående af 64 pixels hver og som tilsammen udgør det komplette billede. Dermed er billede klar til cosinustransformationen fra forrige section.

### 3.1.6 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 synlig indflydelse på billedkvaliteten. Da DCT'en har energikomprimeret A kan nogle af de små koefficienter smides væk. Desuden ser det menneskelige øje ikke meget hurtige ændringer i farveintensitet over små afstande - altså er koefficienterne i nedre højre hjørne af B ikke betydningsfulde for vores indtryk af billedkvalitet.

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 man har bestemt rent empirisk ved subjektive eksperimenter omhandlende det menneskelige syn. Tallene i matricen er tilpasset således at den synlige billedkvalitet er høj, mens billedet komprimeres mest muligt [Lok Ming Lui af The Chinese University of Hong Kong]. Kvantiseringsmatricen, som vi fremover vil kalde Q, ses i figur 3.15 på næste side. Dette er kvantiseringsmatricen for kvalitet 50. Matricen vi har valgt at bruge er den samme som bruges i JPEG-standarden.

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 50 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (3.15)$$

Matricen P kvantiseres ved indgangsvis division med Q.

$$C = \frac{B}{Q_{50}} \quad \text{ } \quad (3.16)$$

Det er værd at bemærke at matrixdivision ikke er hvad vi gør brug af her. Som sagt foregår der simpel indgangsvis division. Altså

$$C_{(i,j)} = \frac{B_{(i,j)}}{Q_{50(i,j)}} \quad (3.17)$$

Ingangene i B afrundes til nærmeste heltal og matricen er dermed kvantiseret. Vi kalder den kvantiserede matrix for C. Afrundingen er hvad der gør komprimeringen lossy, da det

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. Brugen af en lossy komprimering retfærdiggøres ved at de data, som smides væk, ikke har stor betydning for billedets generelle udseende - det menneskelige øje kan ikke opfatte de manglende værdier, når billedet igen pakkes ud. Desuden gør dette skridt den senere komprimering 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 afrundet til heltal tæt på eller lig 0 og at værdier i øverste venstre hjørne vil forblive høje i forhold til de øvrige værdier. De mange værdier tæt på eller lig 0 skyldes den tidligere centrering omkring 0, 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.

Matricen i figur 3.15 hedder  $Q_{50}$  fordi den er kvantiseringsmatricen for en komprimering med kvalitet 50. I JPEG kan komprimeringsgraden justeres og det er med denne matrix dette gøres. Matricen kan have en værdi 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  $Q_{50}$  ikke komprimerer 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 komprimering 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.

$$Q_n = \begin{cases} \frac{100-n}{50} \cdot Q_{50} & \text{hvis } n > 50 \\ \frac{50}{n} \cdot Q_{50} & \text{hvis } n < 50 \end{cases}$$



Det ses 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 0. 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 0. Filen vil i dette tilfælde ende med at fylde mere, men med højere billedkvalitet.

Når dette skridt skal inverteres, ganges C indgangsvis med Q-matricen, som blev brugt under komprimeringen. Det er nødvendigt at matricen er magen til, da inverteringen ellers ikke vil producere tal lignende de originale. Det er derfor vigtigt at oplysninger om hvilken grad billedet er komprimeret ved gemmes sammen med de komprimerede pixeldata.

### 3.1.7 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.

Huffmankodning er afart af entropikodning og en komprimeringsmetode, hvorpå en stor mængde data kan repræsenteres ved hjælp af de enkelte symbolers (i vores tilfælde tallene

0–255) sandsynlighed for at fremkomme. Komprimering med Huffmankodning tildeler alle symboler, der fremkommer i datastrengen, et kodeord 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 fx være

$$\text{"} A = 0, B = 10, C = 110, D = 111 \text{"} \quad (3.18)$$



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 3.18 på næste side 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 BA 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 via skabelsen af et Huffmantræ, som er et overblik over sandsynligheden for at de enkelte symboler fremkommer og hvilket kodeord de skal tildeles. 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 i datastrengen beregnes. Disse er agivet i tabel 3.1.

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>	$\approx 1$

**Tabel 3.1.** Sandsynligheder for de enkelte symboler

Ud fra tabel 3.1 opstilles der fire blade, med hvert deres symbol som indgang, se figur A.1 på side 37. Herefter opstilles de to blade med den mindste sandsynlighed i et undertræ, med bladene som indgange, se figur A.2 på side 37. 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 indgangene hhv. et 0– og 1–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 sandsynlighed er summen af bladenes sandsynlighed ( $0.27 + 0.17 = 0.44$ ), se figur A.3 på side 37. Samme fremgangsmåde gentages

for de sidste to undertræer/blade og derved skabes det totale og færdige Huffmantræ, se figur A.4 på side 38.

Qua træet på figur A.4 på side 38 kan symbolernes kodeord defineres som værende Det er

Symbol	Kodeord
a	0
b	100
c	101
d	11

*Tabel 3.2.* Symbolernes kodeord

her tydeligt at symbolerne med størst frekvens er tildelt de korteste binære repræsentationer, hvorved at de vil 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 vores C-matricer. I disse transformerede og kvantiserede matricer ligger energikomprimeerde informationer om det originale billede, således at mange indgange er lig 0. Altså er der mange fremkomster af 0 og disse kan udtrykkes ved en kort Huffmankode.

Når alle værdier er parret med en kodestreng, gemmes disse parringer sammen med en dictionary 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 algoritmen - de tidligere 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 afsnit 3.1.4, hvilket udføres på det henviste billede af Lena fra tidligere. Billedet har dimensionerne  $512 \times 512$  px, hvilket indledningsvist opdeles i 8 undermatricer. Endvidere er der  $64 \cdot 64$  matricer af  $8 \times 8$ , hvilket giver  $64 \cdot 64 \cdot 8 \cdot 8 = 262144$  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, opnår man en komprimering af det fulde farvebillede. Repetitionerne af beregningerne udelades, da dette blot er et eksempel. Derimod vises resultatet af hvert skridt for hele billedet, efter regneeksemplet med undermatricen er udført. For at skabe overblik ses komprimerings-algoritmen herunder:

### Algoritme: Komprimering

1. Input:
2. Output:

Billedet,  $A: m \times n$  pixels  
Dekomprimeret billede

#### Komprimering

3. Opdeling:
4. Centrering omkring 0:
5. DCT / DCT-II:
6. Kvantisering:
7. Entropi kodning:
8. Gentagelse:

Billedet opdeles i farvekanaler og  $8 \times 8$  matricer

$$A = A - 128$$

$$A \xrightarrow{\text{DCT}} B = U \cdot A \cdot V^T$$

$$[C] = \frac{B}{Q}$$

$C$  / Huffman → Komprimeret fil

Ovenstående gentages for samtlige  $8 \times 8$  matricer

#### Dekomprimering (invers proces)

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

Komprimeret fil → Huffman →  $C'$

$$B' = Q \cdot C'$$

$$A' = U^T \cdot B' \cdot U \rightarrow \text{Invers DCT} \rightarrow A'$$

$$A' = A' + 128$$

$8 \times 8$  matricer samles til ét billede

Den specifikke undermatrix i følgende eksempel ligger i øverste venstre hjørne, jævnfør Figur 3.10 på side 18. Første værdi for hver indgang, dvs. den røde kanal, findes via vores Python-program og følgende undermatrix,  $A_{1,1}$ , fremkommer.



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

**Figur 3.8.** Oprindelig - Visuel.

**Figur 3.9.** Oprindelig - Tal.



**Figur 3.10.** Lena delt op i 4096,  $8 \times 8$  undermatricer.

Herfra udføres DCT på undermatricen, og B findes via  $B = U \cdot A \cdot U^T$ , hvor  $U$  og  $U^T$  er givet i udtryk 3.3 på side 10.

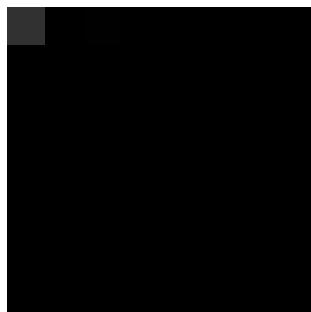


**Figur 3.11.** DCT - Visuel.

$$B_{1,1} = \begin{bmatrix} 791 & -7.1 & 8.12 & 4.23 & -1.8 & -3.6 & -1.2 & 2.49 \\ -1.5 & 0.43 & 0.01 & 1.90 & 0.78 & 1.25 & -3.4 & -0.9 \\ -0.7 & -0.8 & -0.7 & -0.9 & -0.1 & -0.2 & 0.93 & 1.02 \\ 2.66 & 1.02 & 1.38 & 0.21 & -0.5 & -0.5 & 0.95 & -1.3 \\ -3.3 & -0.9 & -1.5 & -0.2 & 0.75 & 0.18 & -1.0 & 1.55 \\ 2.40 & 0.53 & 1.12 & 0.85 & -0.6 & 0.75 & -0.4 & -1.7 \\ -1.1 & -0.1 & -0.6 & -1.3 & 0.32 & -1.4 & 1.75 & 1.50 \\ 0.21 & -0.1 & 0.18 & 0.98 & -0.1 & 1.15 & -1.6 & -0.9 \end{bmatrix}$$

**Figur 3.12.** DCT - Tal.

Det ses tydeligt på den visuelle figur 3.11, 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. I øvrigt har Python skaleret værdien 791 til 255 og negative værdier til 0. Videre i algoritmen udføres kvantiseringen, hvor der i dette eksempel tages udgangspunkt i  $Q_{50}$  jævnfør 3.15 på side 13. Den afrundede C findes via  $[C] = \frac{B}{Q}$ :

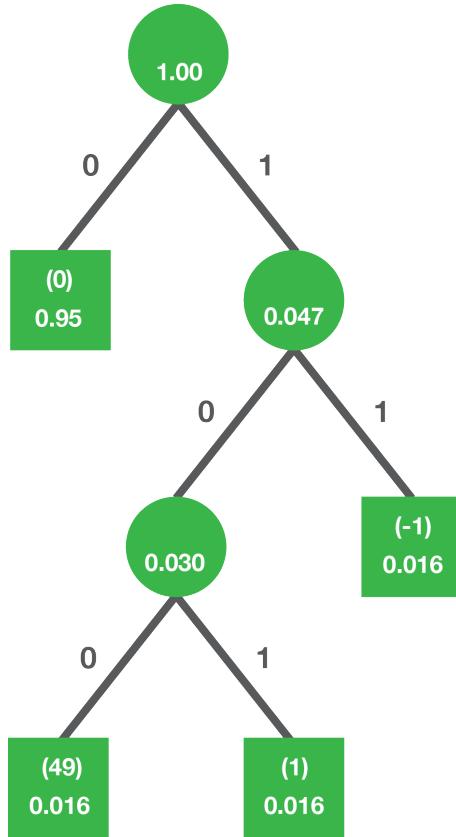


**Figur 3.13.** Kvantisering - Visuel.

$$C_{1,1} = \begin{bmatrix} 49 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figur 3.14.** Kvantisering - Tal.

Læg mærke til at divisionen er foretaget indgangsvis. Det ses, jævnfør tallene på figur 3.14, at matricen hovedsageligt udgøres af nuller. Dette er essentielt for huffman-entropikodningen, da den nu effektivt kan komprimeres. Huffman-træet for undermatricen kan ses herunder. Tallene uden parentes repræsenterer hyppigheden af tallene i parentes.



**Figur 3.15.** Huffman-træ for undermatricen.

Træet havde mildt sagt været markant større, hvis man f.eks. lavede huffman-træet ud fra  $A_{1,1}$  eller  $B_{1,1}$ . Her er det åbenlyst hvordan de to nævnte matricer kun er *forberedende* for en effektiv komprimering.

Man kan med fordel opstille en tabel med relevante observationer fra huffman-træet og dens resultater, se tabel 3.3. Den binære form blev fundet jævnfør 3.1.7 på side 15.

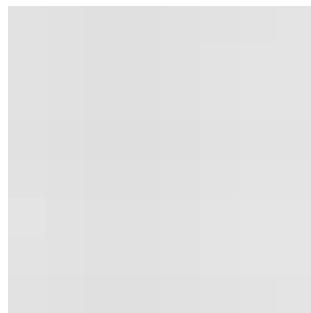
Indgangsværdi	Antal	Sandsynlighed	Binær form
49	1	$\frac{1}{64} \approx 0.016$	100
1	1	$\frac{1}{64} \approx 0.016$	101
-1	1	$\frac{1}{64} \approx 0.016$	11
0	61	$\frac{61}{64} \approx 0.95$	0

Tabel 3.3. Huffman-træet i tabelform

I binær form vil undermatricen nu hedde:

En komprimeret fil er opnået, og den lange streng af nulser gør, at filen fylder mindre end hvis huffman-entropikodningen var blevet gjort før kvantiseringen. Jævnfør algoritmen

på side 17 udføres *dekomprimeringen*, hvilket gøres med de modsatte regneoperationer i forhold til komprimeringen. Resultatet af algoritmen på undermatricen ender med følgende udseende og værdier:

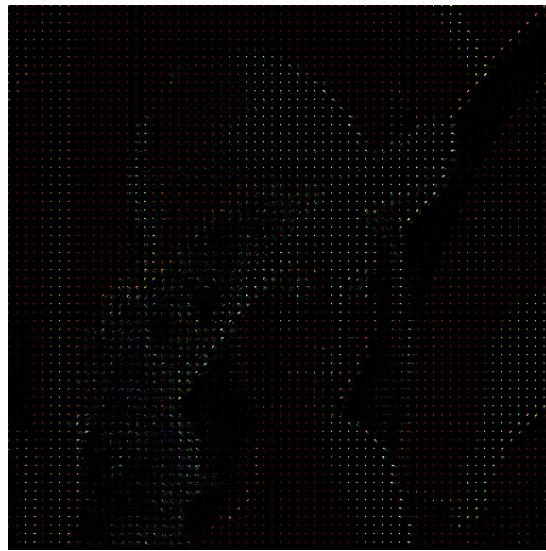


$$C_{1,1} = \begin{bmatrix} 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \\ 226 & 225 & 224 & 224 & 225 & 226 & 228 & 230 \end{bmatrix}$$

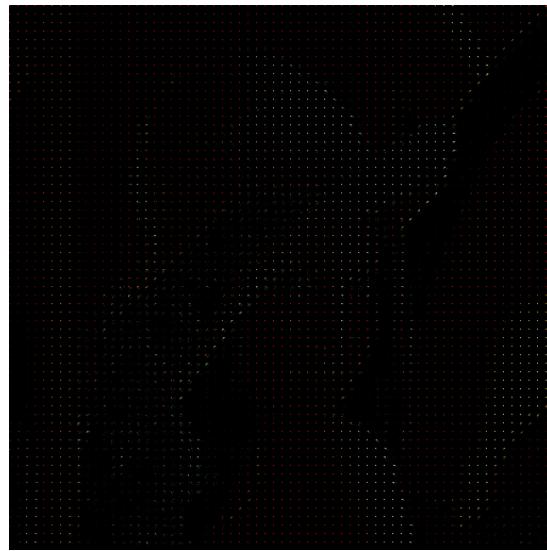
**Figur 3.16.** Dekomprimeret  
- Visuel.

**Figur 3.17.** Dekomprimeret - Tal.

Den umiddelbare største ændring af undermatricen, 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 kan man se algoritmen udført på hele Lena-billedet i de følgende figurer. Det undlades dog at vise den binære form.

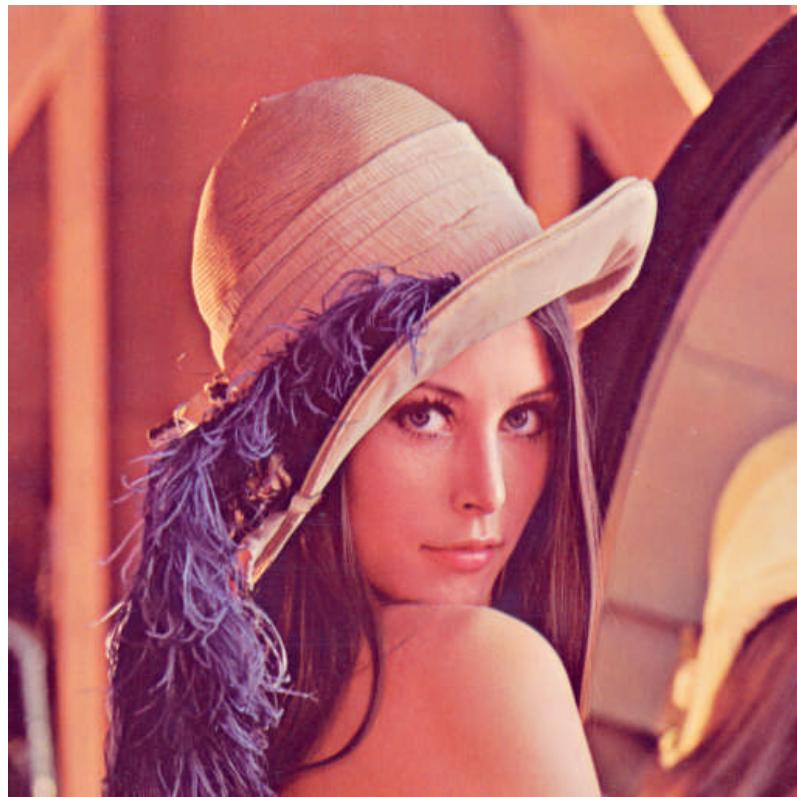


**Figur 3.18.** DCT - Lena.

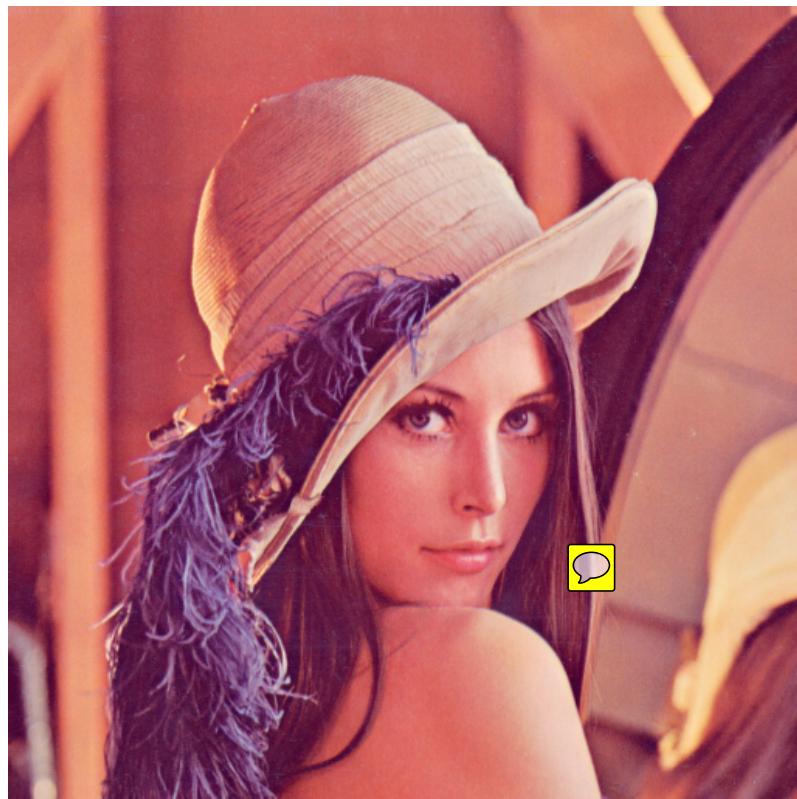


**Figur 3.19.** Kvantisering - Lena.

Jævnfør 3.20 på side 21 og 3.21 på modstående side er det enormt svært at skældne mellem billederne. Hensigten er opnået; billedet har ikke ændret sig betydeligt og kan samtidig komprimeres til en brøkdel af størrelsen end før algoritmen.



*Figur 3.20.* Dekomprimeret - Lena.



*Figur 3.21.* Oprindelig - Lena.

### 3.3 DCT udledning

I dette afsnit beskrives udledningen af DCT fra den diskrete fourier transformation med udgangspunkt i Charles K. Chui and Qingtang Jiang [2013].

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

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

$$\tilde{x}_l = \begin{cases} x_l, & \text{for } l = 0, \dots, n-1 \\ x_{2n-l-1}, & \text{for } l = n, \dots, 2n-1 \end{cases} \quad (3.20)$$

Qua Charles K. Chui and Qingtang Jiang [2013] er fourier transformationen for en vektor  $\mathbf{x} \in \mathbb{C}^n$  givet som

$$\hat{\mathbf{x}} = F_n \mathbf{x} \quad (3.21)$$

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

$$(\hat{\mathbf{x}})_j = \sum_{l=0}^{2n-1} \tilde{x}_l e^{-i \frac{2l\pi j}{2n}} \quad (3.22)$$

$\tilde{x}_l$  bliver delt op i to summationer i forhold til definitionsmængden for at arbejde med  $x_l$  i stedet for  $\tilde{x}_l$

$$(\hat{\mathbf{x}})_j = \sum_{l=0}^{n-1} x_l e^{-i \frac{l j \pi}{n}} + \sum_{l=n}^{2n-1} x_{2n-l-1} e^{-i \frac{l j \pi}{n}} \quad (3.23)$$

Derefter er det interessant at sikre sig at de to summationer har samme begrænsninger. Qua 3.20 variablen ændres ved ændring af summationstegnets begrænsninger<sup>1</sup>

$$\sum_{l=n}^{2n-1} x_{2n-l-1} e^{-i \frac{l j \pi}{n}} = \sum_{l=0}^{n-1} x_l e^{i \frac{(l+1) j \pi}{n}}$$

Nu skrives de to summationstegn i 3.23 sammen i henhold til ovenstående omskrivning og  $x_l$  sættes udenfor en parentes

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

Brøkerne i potensen forlænges med  $\frac{2}{2}$  og leddene ganges med  $e^{i \frac{j(1-1)\pi}{2n}} = 1$  for at isolere

<sup>1</sup>FiXme Dødelige: Hvordan ændrer potensen sig med  $-l - 1$ ? Og hvorfor bliver -i til i?

$e^{i\frac{j\pi}{2n}}$ , som derefter sættes udenfor en parentes

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

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

$$(\hat{\tilde{x}})_j = 2e^{i\frac{j\pi}{2n}} \sum_{l=0}^{n-1} x_l \cos\left(\frac{j(2l+1)\pi}{2n}\right) = e^{i\frac{j\pi}{2n}} \cdot d_j \quad (3.24)$$

↓

$$d_j = 2 \sum_{l=0}^{n-1} x_l \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \quad (3.25)$$

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

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

Dette kan qua 3.24 udtrykkes som

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

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

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

---

<sup>2</sup>Første Dødelige: Hvor kommer denne definition fra?

Så bliver  $(\hat{\tilde{x}})_{2n-j}$  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:

$$d_{2n-j} = -d_j \quad (3.26)$$

Og når  $j = n$  i 3.26, fremkommer

$$d_n = -d_n \quad (3.27)$$

$$d_n = 0 \quad (3.28)$$

Den inverse af DFT er givet som

$$\tilde{F}_n = \frac{1}{n} F_n^* \quad (3.29)$$

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

For at finde  $l$ 'te indgang af  $\tilde{x}$  bruges ligning 3.29 på  $j$ 'te af  $\hat{\tilde{x}}$  i den  $2n$  lange DFT

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

$(\hat{\tilde{x}})_j$  fra 3.24 indsættes

$$\begin{aligned} (\hat{\tilde{x}})_l &= \frac{1}{2n} \sum_{j=0}^{2n-1} d_j e^{\frac{ij\pi}{2n}} e^{\frac{ij2l\pi}{2n}} \\ &= \frac{1}{2n} \sum_{j=0}^{2n-1} d_j e^{\frac{ij(2l+1)\pi}{2n}} \\ &= \frac{1}{2n} \sum_{j=0}^{2n-1} d_j e^{\frac{ij(l+\frac{1}{2})\pi}{n}} \end{aligned} \quad (3.30)$$

Summen i 3.30 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 <sup>3</sup>

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

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 3.26 og ovenstående fremkommer

$$= \sum_{k=1}^{n-1} d_{2n-k} (-e^{\frac{-ik(l+\frac{1}{2})\pi}{n}}) \quad (3.31)$$

$$= \sum_{k=1}^{n-1} (-d_k) (-e^{\frac{-ik(l+\frac{1}{2})\pi}{n}}) \quad (3.32)$$

Summationstegnene sættes ind i 3.30, hvor begrænsningerne 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.

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

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:

$$= \frac{1}{n} \left( \frac{d_0}{2} \sum_{j=1}^{n-1} d_j \cos\left(\frac{j(l+\frac{1}{2})\pi}{n}\right) \right) \quad (3.33)$$

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

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

<sup>3</sup>Fixme Dødelige: Hvordan ændrer vi begrænsningerne?

<sup>4</sup>  $x_k$  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:

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

Denne ligning kan beskrives på vektorform for  $k, l = 0, \dots, n-1$  hvor  $n$  er vektor  $a_0, \dots, a_{n-1}$ .

$$a_k = \left[ \frac{1}{\sqrt{n}}, \sqrt{\frac{1}{n}} \cos\left(\frac{k+\frac{1}{2}\pi}{n}\right), \dots, \sqrt{\frac{2}{n}} \cos\left(\frac{(n-1)(k+\frac{1}{2}\pi)}{n}\right) \right]^T \quad (3.34)$$

Herefter kan den diskrete cosinustransformation defineres som

$$DCT = [\mathbf{a}_0, \dots, \mathbf{a}_{n-1}] \quad (3.35)$$

hvor  $n \geq 2$  og  $k = 0, \dots, n-1$ . Den diskrete cosinustransformation er altså en  $n \times n$ -matrix, som består af  $n$  søjlevекторer beskrevet ved 3.34 på side 26 hvor  $k = 0, \dots, n-1$ .

---

<sup>4</sup>Fixme Dødelige: hvor kommer  $\delta_{k-1}$  fra?

# **Billedkomprimering med Principal Component Analysis**

---

## **4.1 Teori**

## 4.2 Anvendelse af PCA

# **Resultater** 5

---

I kapitel ?? blev den energirenoverede bygnings forventede energibesparelse udregnet.



# Vurdering 6

---

Vurdér aritmetisk kodning i forhold til Huffman, samt de to kombineret.

## 6.1 Hastighed

## 6.2 Kvalitet

## 6.3 Størrelse

I kapitel ?? blev den energirenoverede bygnings forventede energibesparelse udregnet.



# Konklusion 7

---

I kapitel ?? blev den energirenoverede bygnings forventede energibesparelse udregnet.



# Litteratur

---

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

**Guillermo Sapiro af Duke university.** Guillermo Sapiro af Duke university. *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.

**Harvard Mudd College.** Harvard Mudd College. *Padding with Zeros*.  
<https://www.cs.hmc.edu/~kperdue/zeroPadding.html>. Besøgt: 4-11-2015.

**Lawrence E. Spence, Arnold J. Insel og Stephen H. Friedberg, 2008.** Lawrence E. Spence, Arnold J. Insel og Stephen H. Friedberg. *Elementary Linear Algebra 2015*. ISBN: 978-1-78448-372-2. Pearson Education, Inc., 2008.

**Lok Ming Lui af The Chinese Unisversity of Hong Kong.** Lok Ming Lui af The Chinese Unisversity of Hong Kong. *Image compression and the Discrete Cosine Trasnform*. <http://www.lokminglui.com/dct.pdf>. Besøgt: 4-11-2015.

**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.

**Tage Gutmann 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.

**Ye Wang, Miikka Vilermo and Leonid Yaroslavski.** Ye Wang, Miikka Vilermo and 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.

## Rettelser

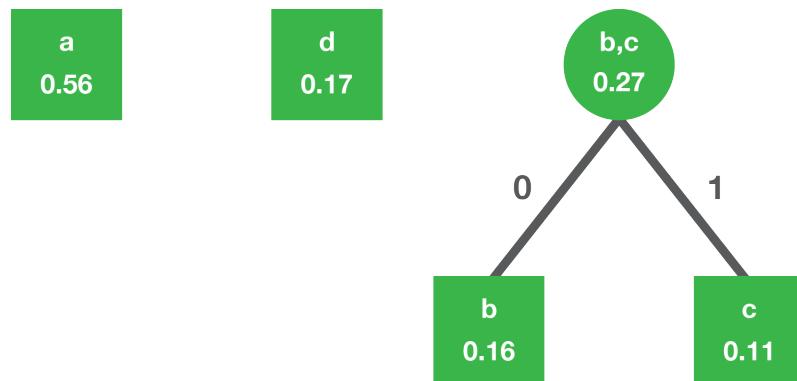
Dødelige: kilde . . . . .	4
Dødelige: Hvordan ændrer potensen sig med $-l - 1$ ? Og hvorfor bliver $-i$ til $i$ ? . . . . .	22
Dødelige: Hvor kommer denne definition fra? . . . . .	23
Dødelige: Hvordan ændrer vi begrænsningerne? . . . . .	25
Dødelige: hvor kommer $\delta_{k-1}$ fra? . . . . .	26

# Huffmantræ A

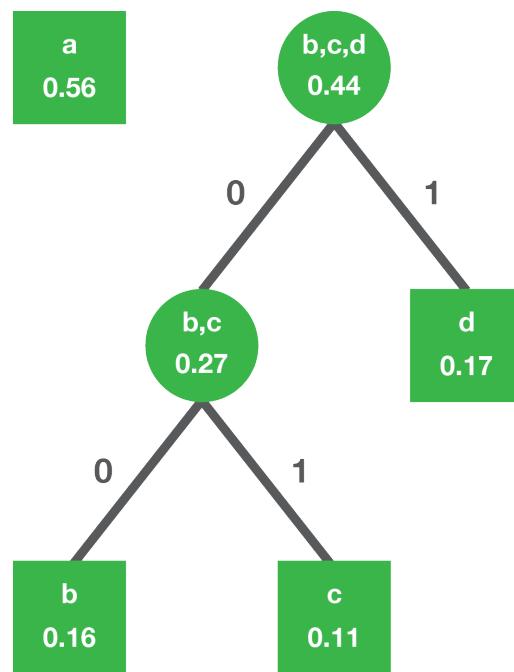
---



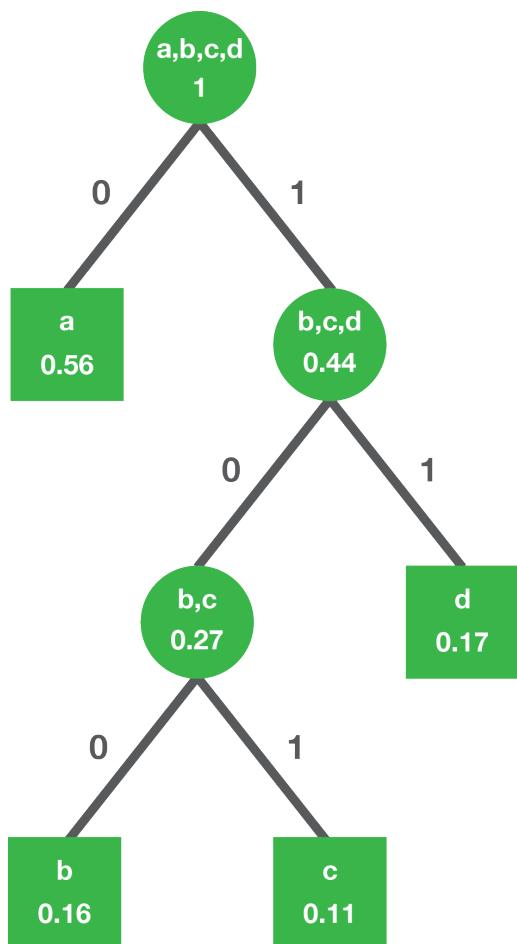
Figur A.1. Første trin i skabelsen af Huffmantræ



Figur A.2. Andet trin i skabelsen af Huffmantræ



Figur A.3. Tredje trin i skabelsen af Huffmantræ



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