|  |
| --- |
| Fakultet for ingeniørvitenskap og teknologi  **Oppgave 2: Lossy Compression (DCT og Quantization)**  Gruppe «Gutta»  Jørgen Nordås, Isak Steinmo Hansen, Patrik Andreassen  DTE-2803-2, Høst 2023 |

Innholdsfortegnelse

[1 Om oppgaven 2](#_Toc148775506)

[2 Gjennomgang kode 2](#_Toc148775507)

[3 Resultater ved kjøring av kode 6](#_Toc148775508)

[4 Komplett kode 8](#_Toc148775509)

# Om oppgaven

I denne oppgaven skal vi implementere en «lossy» kompresjonspipeline ved hjelp av DCT (Discrete Cosinus Transform) og kvantisering. Vi skal bruke tre forskjellige kvantiseringsmatriser der vi viser minst 3 forskjellige scenarier.

1. "normal" kompresjon.. ved å sette høyfrekvente deler til 0
2. b) sette kun lavfrekvente deler til 0
3. c) sette høy- og lavfrekvente deler til 0 og beholde resten.

I dette dokumentet skal vi kort presentere kodens komponenter, samt presentere resultatene fra koden og diskuter forskjellene mellom disse.

# Gjennomgang kode

Her er en kort gjennomgang av koden vår og de forskjellige stegene som skjer fra start til slutt, samt noen viktige poeng å forstå. Hele koden kan du finne i siste kapittel.

Koden definerer fire hovedfunksjoner:

* **quantize(dct, quantize):** Vi kvantiserer DCT-koeffisienten ved a dele dem med de tilsvarende verdiene i kvantiseringsmatrisen. Dette trinnet innfører tap ved a forkaste noen av koeffisientene.

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

* **dequantize\_dct(dct\_matrix, quantization\_matrix):** Vi reverserer kvantiseringen ved a multiplisere DCT-koeffisienten med kvantiseringsmatrisen.

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

* **dtc\_quantiazation\_pipeline(image, B, quantization\_matrix\_a, quantization\_matrix\_b, quantization\_matrix\_c):** Denne funksjonen utfører DCT og kvantisering på bildet, deretter dekvantiserer og rekonstruerer bildet. Den gir også muligheten til å bruke tre forskjellige kvantiseringsmatriser for sammenligning slik som oppgaven vil.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

* **plot\_result(images):** Denne funksjonen brukes til å plotte de originale og resultatbildene etter komprimeringen.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

I main-funksjonen blir følgende gjort:

1. Originale bilder blir lastet inn og lagret i en liste som heter «original\_images».
2. Blokkstørrelsen B er satt til 8. Denne bestemmer størrelsen på blokkene som bildet deles opp i for komprimering. Denne blokkstørrelsen kan påvirke komprimeringskvaliteten og den visuelle kvaliteten på det endelige bildet. Ettersom kvantiseringsmatrisen var er en 8x8 matrise, sa må bildet som skal kvantiseres også være det. Bildene vi bruker er langt større enn det, sa derfor må vi dele opp bildet i flere 8x8 blokker som kvantiseres hver for seg før de blir satt sammen igjen.
3. Tre forskjellige kvantiseringsmatriser (quantization\_matrix\_a, quantization\_matrix\_b, og quantization\_matrix\_c) er definert. Disse matrisene brukes til å bestemme hvilke frekvenskomponenter som skal kvantiseres sterkest (settes til 0) og hvilke som skal bevares.
4. For hvert originalbilde i listen, blir dtc\_quantiazation\_pipeline-funksjonen kalt for å utføre DCT, kvantisering og rekonstruksjon av bildet med de tre forskjellige kvantiseringsstrategiene.
5. Alle de originale og komprimerte bildene blir lagret i en liste som heter «plotting\_images».
6. plot\_result-funksjonen blir kalt for å vise de originale og komprimerte bildene i et rutenett. Vi bruker biblioteket «matplotlib» til plottingen.

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font, dokument

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

Main-funksjonen blir kalt direkte (ved å kjøre programmet), så vil hele prosessen bli utført.

* 1. Originalbildene blir lastet inn.
  2. Bildene blir behandlet ved hjelp av DCT og kvantisering med de tre forskjellige kvantiseringsmatrisene.
  3. Resultatene blir vist i et rutenett med åtte bilder.

Dette programmet brukes til å demonstrere hvordan bildekomprimering kan utføres ved å sette visse DCT-koeffisienter til 0, og hvordan dette påvirker bildets kvalitet. Den visuelle sammenligningen gjøres ved å bruke tre ulike kvantiseringsstrategier.

# Resultater ved kjøring av kode

I dette kapittelet skal vi vise noe av resultatene fra kjøringen av koden med forskjellige kvantiseringsmatriser.

Oppgaven sa at vi skulle bruke tre forskjellige kvantiseringmatriser:

* en som setter bare høyfrekvente deler til 0 (a)
* en som setter bare lavfrekvente deler til 0 (b)
* en som setter både høyfrekvente og lavfrekvente deler til 0 (c).

Disse matrisene brukes vanligvis i DCT-basert kompresjon for å kontrollere hvor mye informasjon som bevares i forskjellige frekvensområder. Først tenkte vi å forklare litt hva de forskjellige matrisene beholder og fjerner av informasjon i bildet.

* **Kvantiseringstrategi a:** Dette er kvantiseringen som setter kun høyfrekvente deler av bildet til 0. Dette betyr at kvantiseringen beholder lavfrekvente detaljer i bildet, mens høyfrekvente detaljer blir fjernet.
* **Kvantiseringstrategi b:** Dette er kvantiseringen som setter bare lavfrekvente deler av bildet til 0. Denne strategien beholder høyfrekvente detaljer i bildet, men fjerner lavfrekvente komponenter.
* **Kvantiseringstrategi c:** Dette er kvantiseringen som setter både høy- og lavfrekvente deler av bildet til 0. Dette resulterer i en ekstrem komprimering, hvor både høy- og lavfrekvente detaljer går tapt.

Når du kjører koden får du følgende resultat:

Et bilde som inneholder idrettskonkurranse, sport, sportsutstyr, hockey

Automatisk generert beskrivelse

Som du ser så blir kvaliteten på bildene dårligere jo lengre til høyre du går, men fortsatt kan du se hva bildene viser.

* **Originalbilde:** Dette er det originale bildet før noen komprimering eller dekomprimering har skjedd.
* **"High-frequency deler to 0":** Dette bildet viser resultatet av kvantiseringstrategi a, hvor bare høyfrekvente deler er satt til 0. Det er en viss grad av komprimering, men det beholdt hovedsakelig lavfrekvente detaljer.
* **"Low-frequency deler to 0":** Dette bildet viser resultatet av kvantiseringstrategi b, hvor bare lavfrekvente deler er satt til 0. Her blir høyfrekvente detaljer bevart, men lavfrekvente detaljer går tapt.
* **"Både high and low-frequency deler til 0":** Dette bildet viser resultatet av kvantiseringstrategi c, hvor både høy- og lavfrekvente deler er satt til 0. Dette resulterer i ekstrem komprimering, og både høy- og lavfrekvente detaljer går tapt.

Ved å se på disse bilder kan du tydelig vurdere hvordan hver kvantiseringstrategi påvirker bildekvaliteten og komprimeringsgraden. Du kan også justere størrelsen på blokkene (B-verdien) for å eksperimentere med komprimeringsytelsen.

# Komplett kode

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse