|  |
| --- |
| Fakultet for ingeniørvitenskap og teknologi  **Oppgave 1: Lossy Compression (FFT, Quantization og Huffman)**  Gruppe «Gutta»  Jørgen Nordås, Isak Steinmo Hansen, Patrik Andreassen  DTE-2803-2, Høst 2023 |

Innholdsfortegnelse

[1 Om oppgaven 2](#_Toc148775625)

[2 Gjennomgang kode 2](#_Toc148775626)

[3 Resultater ved kjøring av kode 5](#_Toc148775627)

[3.1 Resultat 1 – Bruk av standard JPEG-komprimeringsmatrise 5](#_Toc148775628)

[3.2 Resultat 2 – Fjerne høye og lave frekvenser 7](#_Toc148775629)

[3.3 Resultat 3 – Fjerne enda mer av høye og lave frekvenser 9](#_Toc148775630)

[4 Fordeler med FFT (Fast Fourier Transform) 11](#_Toc148775631)

[5 Komplett kode 12](#_Toc148775632)

# Om oppgaven

I denne oppgaven skal vi implementere en «lossy» kompresjonspipeline ved hjelp av FFT (Fast Fourier Transform), kvantisering og Huffman-koding (helt på slutten). I dette dokumentet skal vi kort presentere kodens komponenter, samt presentere resultatene fra koden og diskuter fordelene med FFT.

# 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.

1. **Lasting av bilder:** Vi laster bildene manuelt ved hjelp av modulen «OpenCV», som også gjør dem om til svart og hvit.

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

1. **Blokkstørrelse (B):** Vi definerer en blokkstørrelse (B), som 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 vår er en 8x8 matrise, så må bildet som skal kvantiseres også være det. Bildene vi bruker er langt større enn det, så derfor må vi dele opp bildet i flere 8x8 blokker som kvantiseres hver for seg før de blir satt sammen igjen.



Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

1. **Kvantiseringsmatrise:** Vi definerer en kvantiseringsmatrise. Denne matrisen spiller en betydelig rolle i komprimeringskvaliteten. Justering av verdiene i denne matrisen vil påvirke komprimeringsforholdet og bildekvaliteten. Høyere verdier vil resultere i høyere komprimering (mer tap) og lavere bildekvalitet, og vice versa. Vi har testet koden med forskjellige kvantiseringsmatriser, og resultatene av dette ser du i neste kapittel.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

1. **FFT-transformasjon:** Vi utfører FFT (Fast Fourier Transform) på blokker av bildet, noe som er en grunnleggende trinn i å transformere bildet fra det romlige domenet til frekvensdomenet. FFT hjelper med å konsentrere bildeinformasjonen i færre koeffisienter som er mer egnet for kvantisering. Vi har benyttet oss av biblioteket «scipy» som har en funksjon som heter «fft» som gjør operasjonen for oss.

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

1. **Kvantisering:** Vi kvantiserer FFT-koeffisientene ved å dele dem med de tilsvarende verdiene i kvantiseringsmatrisen. Dette trinnet innfører tap ved å forkaste noen av koeffisientene.

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

1. **Huffman-koding:** Etter kvantisering, flater vi den kvantiserte matrisen, bruker Huffman-koding på den resulterende 1D-dataen og lagrer den kodete dataen sammen med Huffman-kodeken for senere dekoding. Igjen har vi benyttet oss av et bibliotek som heter «dahuffman» for å gjennomføre huffman koderingen.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

1. **Huffman-dekoding:** Vi dekoder den Huffman-kodede dataen for å få kvantisert data tilbake.



1. **Dekvantisering:** Vi reverserer kvantiseringen ved å multiplisere den dekodede dataen med kvantiseringsmatrisen.

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

Et bilde som inneholder tekst, Font, skjermbilde, line

Automatisk generert beskrivelse

1. **Invers FFT-transformasjon (IFFT):** Vi bruker IFFT til å transformere dataen tilbake fra frekvensdomenet til romdomenet. Også her bruker vi biblioteket «scipy» som har funksjonen «ifft» som gjør denne operasjonen for oss.



1. **Komprimeringsforhold:** Vi beregner komprimeringsforholdet, som er forholdet mellom størrelsen på det opprinnelige bildet og størrelsen på den kodete dataen. Et høyere komprimeringsforhold indikerer mer komprimering (mer datatap), men kan også resultere i lavere bildekvalitet.



1. **Visning:** Vi viser både de opprinnelige og dekomprimerte bildene for både bildene, sammen med noen tekstanmerkninger om størrelser og komprimeringsforhold.

**Forbehold og mulige forbedringer:**

* Kvantiseringsmatrisen og blokkstørrelsen er viktige parametere som kan justeres for å kontrollere avveiningen mellom komprimering og bildekvalitet.
* Man bør være oppmerksom på at valget av disse parameterne kan variere for forskjellige typer bilder eller bruksområder.
* Koden håndterer ikke alle aspekter ved bildekomprimering, for eksempel håndtering av fargebilder eller tapsfri komprimering.

# Resultater ved kjøring av kode

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

## Resultat 1 – Bruk av standard JPEG-komprimeringsmatrise

Ved bruk av standard matrise for JPEG-komprimering ser vi at bildekvaliteten holder seg veldig, og komprimeringen blir bra også. Det må bemerkes at bildene som brukes allerede er «komprimert» ettersom de er av formatene PNG og JPEG.

**Matrise:**

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

**Resultat:**

Et bilde som inneholder tekst, sport, sko, idrettskonkurranse

Automatisk generert beskrivelse

## Resultat 2 – Fjerne høye og lave frekvenser

Ved å sette høye og lave frekvenser til 0 vil bildekvaliteten bli dårligere, men fortsatt er det lite merkbart selv om skyggene i bildet begynner å bli litt pikslerte. Komprimeringen blir ikke så mye bedre.

**Matrise:**

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

**Resultat:**

Et bilde som inneholder tekst, sport, idrettskonkurranse, fotball

Automatisk generert beskrivelse

## Resultat 3 – Fjerne enda mer av høye og lave frekvenser

Nå begynner bildekvaliteten å bli ganske dårlig, samtidig som at komprimeringen ikke blir så veldig mye bedre. Dette kan ha noe med at vi allerede bruker komprimerte bilder (PNG og JPG).

**Matrise:**

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

**Resultat:**

Et bilde som inneholder tekst, idrettskonkurranse, sport, sportsutstyr

Automatisk generert beskrivelse

# Fordeler med FFT (Fast Fourier Transform)

I dette kapittelet skal vi diskutere tre viktige fordeler med FFT, som er en matematisk algoritme og et kraftig verktøy som brukes på ulike felt, spesielt innen signalbehandling og bildebehandling som vi har brukt det til i denne oppgaven.

FFT er både effektiv og rask, noe som gjør den til en populær algoritme å bruke. Dette kommer av at algoritmen eliminerer redundante beregninger og bryter ned det opprinnelige problemet i mindre delproblemer, noe som gjør den til et førstevalg for sanntidsprogrammer. Hastighetsfordelen til FFT blir spesielt tydelig når man håndterer store datasett, hvor den tillater rask analyse og behandling av signaler og bilder.

FFT muligjør for mer effektiv lagring og overføring av data, noe som gjør den godt egnet for komprimering slik som vi har brukt den til i denne oppgaven. Ettersom FFT transformerer data fra tid (eller rom) til frekvensdomenet, så kan algoritmen konsentrerer de betydelige frekvensinformasjonene til et mindre sett med koeffisienter, som muliggjør den gode komprimeringen.

Til slutt er FFT en stabil og pålitelig algoritme som gir resultater som vi kan stole på. Denne stabiliteten er avgjørende for vitenskapelige og ingeniørapplikasjoner hvor nøyaktighet er viktig. FFT er så viktig at det ofte implementeres i maskinvare.

Oppsummert tilbyr FFT effektivitet, hastighet og allsidighet i transformasjon av data fra tid- eller romdomenet til frekvensdomenet. Den er et grunnleggende verktøy innen signalbehandling og spiller en avgjørende rolle i ulike applikasjoner, fra lydbehandling og bildeanalyse til vitenskapelig forskning og telekommunikasjon. Evnen til å håndtere store datasett effektivt og dens numeriske stabilitet gjør den til en uvurderlig ressurs innen mange felt.

# Komplett kode

Et bilde som inneholder tekst, skjermbilde, dokument, Font

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse