Obligatorisk oppgave 2

INF2310
Vår 2016

Dette oppgavesettet er på fire sider og består av to oppgaver.

Besvarelsen av denne og forrige obligatoriske oppgave må være godkjent for at du skal få anledning til å gå opp til endelig skriftlig eksamen i kurset. Besvarelsene kan utarbeides i smågrupper på opptil to studenter, men det er ikke noe i veien for å arbeide alene. Studenter i samme smågruppe kan levere identisk besvarelse, men samarbeidet må framgå av navnene på forsiden av besvarelsen.

Av forsiden skal det fremgå hvem som har utarbeidet besvarelsen.

Det forventes at arbeidet er et resultat av egen innsats. Å utgi andres arbeid for sitt eget er uetisk og kan medføre sterke reaksjoner fra IFIs side. Se http://www.uio.no/studier/admin/obligatoriske-aktiviteter/mn-ifi-oblig.html. Den skriftlige rapporten leveres primært som en PDF-fil som inneholder hele besvarelsen, med figurer og bilder. Kode skal leveres i tillegg til PDF-filen. Besvarelsen skal leveres via http://devilry.ifi.uio.no. Følgende er viktig:

- Alle filene må lastes opp hver gang man skal levere.
- PDF-filen skal ha følgende navn: **inf2310-oblig2-brukernavn.pdf**.
- Oppgaven skal kunne kjøres fra MATLAB eller Python-scripts med navn: **oppgave1.m/.py** og **oppgave2.m/.py**.
- Spørsmål angående innlevering kan sendes til **olejohas@ifi.uio.no**.

Bildene det refereres til vil være å finne under:
http://www.uio.no/studier/emner/matnat/ifi/INF2310/v16/undervisningsmateriale/bilder/

Oppgaven utleveres **tirsdag 26. april 2015**.
Innleveringsfrist er **tirsdag 10. mai 2015**.

Oppgave 1: Filtrering av periodisk støy

Fjern den periodiske støyen i bildet **uio_noisy.png** ved bruk av et filter designet i Fourier-domenet som har Gaussiske overganger mellom 0 og 1. Ut-bildet skal ha samme størrelse som original-bildet. Filtreringen skal være akseptabel også nær bilderanden, spesielt skal den periodiske støyen være betydelig redusert også i dette området.

Hva skal leveres:

I Bilde av Fourier-spekteret til original-bildet.

II Drøfting av Fourier-spekteret til original-bildet, inkludert drøfting av hvor støyen er og hvorfor den er «utsmurt» slik den er.

III Bilde av Fourier-spekteret til filteret du designet.

IV Resultatet-bildet etter filtrering.

V Drøfting av resultat-bildet, inkludert drøfting av responsen nær bilderanden.

VI Kommentert programkode.

NB!

Du kan benytte ferdige MATLAB eller Python-funksjoner til å lese/skrive fra/til fil, og til å beregne Fourier-transformer og inverse Fourier-transformer. Filteret og filtreringen MÅ du implementere selv.

Oppgave 2: Ikke-tapsfri JPEG-kompresjon

I denne oppgaven skal du implementere de viktigste delene av ikke-tapsfri JPEG-kompresjon.

Du skal lage en funksjon som har to parametre:

1) et filnavn som spesifiserer plasseringen til bildefilen som skal benyttes, og

2) ett tall, q, som indirekte vil bestemme kompresjonsraten.

Funksjonen skal beregne omtrentlig hvor stor lagringsplass det angitte bildet vil bruke etter JPEG-komprimering, og finne hvilken kompresjonsrate dette tilsvarer. Vi vil anta at inputbildet er et gråtonebilde med heltallsintensiteter i intervallet [0, 255] og har både en bredde og en høyde som er multipler av 8. Bruk gjerne bildet **uio.png** for å teste implementasjonen din underveis.

Du skal bruke følgende algoritme/fremgangsmåte for å lage funksjonen:

Steg 1: Last inn bildet som den første parameteren spesifiserer.

Steg 2: Trekk 128 fra alle pikselintensiteter (dette gjør at den forventede gjennomsnittlige intensitetsverdien er omtrent 0).

Steg 3: Begynn i øverste, venstre hjørnet og del opp bildet i 8x8-blokker. Transformér hver blokk med den todimensjonale diskrete cosinus-transformen (2D DCT).

Hint 1: Siden hver blokk har størrelse 8x8, kan vi forenkle den generelle formelen for 2D DCT i forelesningsnotatene til:

$$F(u,v) = \frac{1}{4}c(u)c(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

der:

$$c(\zeta) = \begin{cases} \frac{1}{\sqrt{2}} & \text{hvis } \zeta = 0 \\ 1 & \text{ellers} \end{cases}$$

Hint 2: De transformerte 8x8-blokkene kan lagres som en matrise der hver 8x8-transformblokk er plassert på samme sted som den 8x8-bildeblokken den er beregnet fra.

Steg 4: Rekonstruér det opprinnelige bildet ved å invertere transformen du utførte i forrige steg og addere 128 til alle pikselintensiteter. Programmatisk verifiser at det rekonstruerte bildet er identisk som originalen.

Hint: Den forenklede formelen for den inverse todimensjonale diskrete cosinus-transformen (2D IDCT) er:

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} c(u)c(v)F(u,v)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

der funksjonen c er definert som over. Pga. upresis flyttallsaritmetikk vil det generelt være behov for å avrunde de resulterende verdiene til nærmeste heltall.

Steg 5: La Q være følgende kvantifiseringsmatrise:

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

Punktvis divider hver av de transformerte 8x8-blokkene fra steg 3 med qQ, dvs. produktet av tallparameteren q og kvantifiseringsmatrisen over. Avrund de resulterende verdiene til nærmeste heltall.

Steg 6: Dersom vi skulle fulgt (den sekvensielle modusen i) JPEG-algoritmen videre så skulle vi nå separert det øverste, venstre elementet av hver transformert og kvantifisert 8x8-blokk, kalt et kvantifisert DC-element, fra de resterende 63 elementene i hver blokk. DC-elementene skulle blitt differansetransformert (på tvers av blokkene), og de 63 elementene skulle blokk for blokk blitt sikk-sakk- skannet og deretter (0-basert) løpelengdetransformert. Til slutt skulle (antall biter i) differansene og «løpelengdeparene» blitt entropikodet.

I stedet for å følge denne prosedyren vil vi påstå at den vil grovt sett resultere i en dataforbruk som tilsvarer entropien til alle elementene i alle de transformerte og kvantifiserte 8x8-blokkene. Beregn denne entropien og bruk den til å estimere hvor stor lagringsplass det spesifiserte bildet vil bruke etter JPEG-komprimeringen og hvilken kompresjonsrate dette tilsvarer.

Steg 7: Bruk de transformerte og kvantifiserte 8x8-blokkene fra steg 5 til å rekonstruere en tilnærming av det opprinnelige bildet. Skriv dette bildet til fil.

Du skal teste funksjonen din ved å anvende den på bildet **uio.png** når du benytter hver av følgende verdier av tallparameteren q: 0.1, 0.2, 0.5, 1, 2, 4, 8, 16 og 32. Studer rekonstruksjonene, og bemerk når og hvor i bildet du først oppdager rekonstruksjons-feilene blokk-artefakter, glatting og ringing. Vurder også for hvilke(n) verdi(er) av tallparameteren q som du synes rekonstruksjonen er «god nok» for fremvisning av hele bildet på en vanlig dataskjerm.

Hva skal leveres:

VII De rekonstruerte bildene av **uio.png** med de ni forskjellige verdiene av tallparameteren q.

VIII Drøfting av komprimeringen av **uio.png**, inkludert

1) dine oppdagelser om rekonstruksjonsfeilene,
2) din vurdering av når rekonstruksjonen er «god nok» og
3) forklaring av hvorfor den estimerte kompresjonsraten øker med verdien av tallparameteren q.

IX Kommentert programkode.

NB!

Du kan benytte ferdige MATLAB-funksjoner til å lese/skrive fra/til fil. Blokkoppdelingen, transformasjonene og entropiberegningen MÅ du implementere selv.

Lykke til!