Obligatorisk oppgave 2

IN2070 - høst 2023

Dette oppgavesettet er på 6 sider, og består av 2 bildebehandlingsoppgaver som skal besvares.

Besvarelsen av denne og neste 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 opp-til to studenter, men det er ikke noe i veien for å arbeide alene. Studenter i samme små— 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-en. Besvarelsen skal leveres via `https://devilry.ifi.uio.no`.

Legg merke til følgende:

- Alle filene må lastes opp hver gang man skal levere.
- PDF-en skal ha følgende navn: `in2070-oblig2-brukernavn.pdf`, der `brukernavn` byttes ut med ditt eget sådan.
- Oppgaven skal kunne kjøres fra Python- skript med navn: oppgave1.py (evt. oppgave1a.m osv.).
- Spørsmål angående innlevering: **mathime@ifi.uio.no**

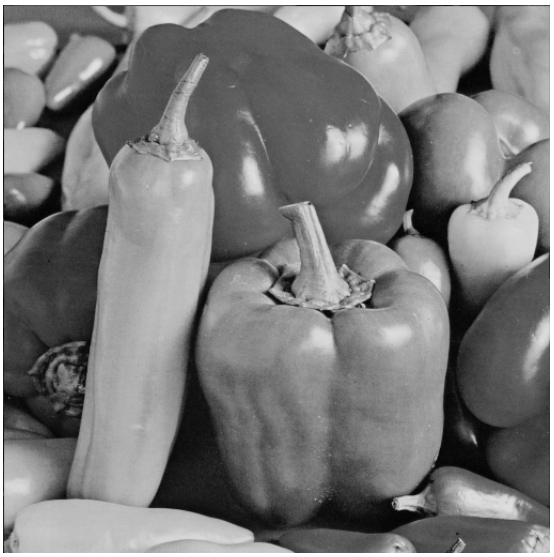
Bildene det refereres til vil være å finne under: `https://www.uio.no/studier/emner/matnat/ifi/IN2070/h23/undervisningsmateriale/bilder/`.

I implementeringsdelen av begge oppgavene kan du benytte deg av eksisterende funksjoner som er implementert i det språket du velger. Se **Appendiks A** på side 5 for hvordan noen funksjoner brukes i hhv. Matlab og Python. Når funksjonene blir referert til under benyttes navngivingen fra Python.

Vi påpeker også at dere må gjerne levere i Jupyter Notebook format hvis dere foretrekker, og dette er en fin måte å kombinere koding og rapportskriving.

Oppgaven utleveres torsdag 26. oktober. Innleveringsfrist er **onsdag 15. november**.

Lykke til!



Figur 1: Bildet `pepper_gray.png`.

1: Konvolusjonsfiltre i frekvensdomenet

I denne oppgaven skal vi konstruere enkle seperable filtre, evaluere konvolusjonsimplementasjoner, og implementere en enkel nedsamplingsmetode. Vi definerer først de to enkleste komponentene i høypass- og lavpassfiltre, som vi betegner ved

$$\ell = \frac{1}{2}[1, 1] \tag{1a}$$

$$h = \frac{1}{2}[-1, 1]. \tag{1b}$$

Fra forelesningene om naboskap og konvolusjon husker du sikkert at vi betegner et seperabelt todimensjonalt filter ved $k_1 * k_2 = k_1 k_2^\intercal$. Videre, definerer vi en familie med endimensjonale lavpassfiltre gitt ved

$$\ell_n = \frac{1}{n}\mathbf{1}_n = \frac{1}{n}[1, \ldots, 1], \tag{2}$$

og vi utvider dette til en familie med seperable todimensjonale middelverdifiltre gitt ved

$$\ell_{n,m} = \ell_n * \ell_m. \tag{3}$$

Oppgave 1.1

I den første oppgaven trenger du først laste inn bildet `pepper_gray.png`, som vist i Figur 1. I resten av oppgaven vil vi kalle dette bildet F. Videre skal du implementere en funksjon `seperable_2d` som tar to endimensjonale filtre og returnerer et seperabelt todimensjonalt filter. Med denne implementasjonen skal du konstruere filtrene

$$\ell\ell = \ell * \ell \tag{4a}$$

$$\ell h = \ell * h \tag{4b}$$

$$h\ell = h * \ell \tag{4c}$$

$$hh = h * h \tag{4d}$$

Filtrer bildet med hvert av disse filtrene og gjør en 2:1 nedsampling av bildet ved å kun plukke par- eller oddetallsindekser i høyde- og bredderetning. Til slutt skal du konstruere det "flislagte" (tiled på godt engelsk)

resultatbildet gitt ved

$$\Phi = \begin{bmatrix} \Phi_{\ell\ell} & \Phi_{\ell h} \\ \Phi_{h\ell} & \Phi_{hh} \end{bmatrix} = \begin{bmatrix} (F * \ell\ell)_{1::2,1::2} & (F * \ell h)_{1::2,0::2} \\ (F * h\ell)_{0::2,1::2} & (F * hh)_{0::2,0::2} \end{bmatrix}, \tag{5}$$

der F er det originale bildet, og vi tillater oss å stjele notasjon fra Python / Matlab ved å la $F_{1::2,0::2}$ eksempelvis indikere nedsampling med oddetallsindekser for **bredderetning** og partallsindekser i **høyderetning**.

I den endelige rapporten din, skal du inkludere:

(a) det "flislagte" resultatbildet,

(b) en kort forklaring på hva du ser i bildet,

(c) en fortolkning av hva hvert av de fire filtrene gjør med originalbildet.

Du skal også levere kildekode for implementasjonen din.

Oppgave 1.2

Benytt implementasjonen av `seperable_2d` for å konstruere filtrene $\ell_{3,3}, \ell_{7,7}$ og $\ell_{11,11}$. Filtrer så bildet F med hvert av disse filtrene ved både romlig konvolusjon, som vi betegner med notasjonen $*_{\text{rom}}$, samt spektral konvolusjon i frekvensdomenet, som betegnes med $*_{\text{frek}}$.

For hver konvolusjon skal du ta tiden over 64 kjøringer, og rapportere gjennomsnittstid og standardavvik for hver av implementasjonene. Regn også ut gjennomsnittlig absolutt avvik mellom de to implementasjonene for hvert filter, gitt ved

$$\text{MAE}(\ell_n *_{\text{frek}} F, \ell_n *_{\text{rom}} F) = \frac{1}{HW} \sum_{y=0}^{H} \sum_{x=1}^{W} \big| (\ell_n *_{\text{frek}} F)(x,y) - (\ell_n *_{\text{rom}} F)(x,y) \big|. \tag{6}$$

I den endelige rapporten skal du altså inkludere

(a) en tabell med gjennomsnittstid, standardavvik for hvert av filtrene og konvolusjonsimplementasjonene,

(b) en tabell med gjennomsnittlig absolutt avvik mellom de to implementasjonene,

(c) en analyse av forskjellen mellom resultene som skal inkludere:

(i) en diskusjon om hva some er likt og hva som er forskjellig i de kvantative resultatene i tabellene.

(ii) en kort forklaring på eventuelle forskjeller i de observerte resultatene.

Du skal også levere kommentert kildekode for implementasjonen din i en seperat fil.

2: Kompresjon og Koding

I denne oppgaven skal vi benytte det "flislagte" bildet Φ fra Oppgave 1.1 til å designe en enkel bildekompresjonsalgoritme i kombinasjon med kvantisering og løpelengdeenkoding. Ideen med algoritmen er å utnytte redudansen i residualregionene $\Phi_{\ell h}, \Phi_{h\ell}, \Phi_{hh}$ som er glisne (sparse på engelsk) representasjoner av orginalbildet.

Oppgave 2.1

La $\hat{F}$ være gitt ved

$$\hat{F}(2x, 2y) = \Phi_{\ell\ell}(x,y) + \Phi_{\ell h}(x,y) + \Phi_{h\ell}(x,y) + \Phi_{hh}(x,y) \tag{7a}$$

$$\hat{F}(2x, 2y+1) = \Phi_{\ell\ell}(x,y) + \Phi_{\ell h}(x,y) - \Phi_{h\ell}(x,y) - \Phi_{hh}(x,y) \tag{7b}$$

$$\hat{F}(2x+1, 2y) = \Phi_{\ell\ell}(x,y) - \Phi_{\ell h}(x,y) + \Phi_{h\ell}(x,y) - \Phi_{hh}(x,y) \tag{7c}$$

$$\hat{F}(2x+1, 2y+1) = \Phi_{\ell\ell}(x,y) - \Phi_{\ell h}(x,y) - \Phi_{h\ell}(x,y) + \Phi_{hh}(x,y). \tag{7d}$$

Implementer en funksjon som tar inn $\Phi_{\ell\ell}, \Phi_{\ell h}, \Phi_{h\ell}, \Phi_{hh}$ og returnerer $\hat{F}$. Regn også ut gjennomsnittlig absolutt avvik mellom $\hat{F}$ og orignalbildet F.

I den endelige rapporten skal du inkludere:

(a) bildene $\hat{F}$ og F, samt differansebildet $|F - \hat{F}|$,

(b) gjennomsnittlig absolutt avvik mellom $\hat{F}$ og originalbildet F.

(c) en kort diskusjon om rimelige antakelser ang. det observerte avviket,

(d) en forklaring, hypotese, eller formodning vedr. konstruksjonen av $\hat{F}$.

Du skal selvfølgelig også levere kildekode for oppgaven, helst med velvalgte kommentarer i pyntelig format.

Oppgave 2.2

I denne oppgaven skal du komprimere hver av de glisne residualbildene $\Phi_{\ell h}, \Phi_{h\ell}, \Phi_{hh}$ med hjelp av ikke-tapsfri kompresjon. Begynn med å bestemme en terskling τ for kvantiseringsprosessen. Bruk så denne til å sette alle elementene i $\Phi_{\ell h}, \Phi_{h\ell}, \Phi_{hh}$ til null dersom absoluttverdien er mindre enn tersklingen τ, slik at

$$\Phi'_{\ell h} = \begin{cases} \Phi_{\ell h}, & \text{if } |\Phi_{\ell h}| \geq \tau \\ 0, & \text{ellers} \end{cases}. \tag{8}$$

Gjør så det samme for $\Phi_{h\ell}$ og Φ_{hh}. Bestem et velvalgt tersklingsnivå basert på kvaliteten på rekonstruksjonen $\hat{F}$ fra forrige oppgave med den originale $\Phi_{\ell\ell}$ sammen med de kvantiserte residualene $\Phi'_{h\ell h}, \Phi'_{h\ell}, \Phi'_{hh}$. Bruk gjerne kvantitative mål (f.eks. MAE) for å bestemme tersklingsnivået.

I tillegg til kildekode, skal rapporten din inkludere:

(a) valg av tersklingsnivå τ med tilhørende rekonstruksjonsbilde $\hat{F}$,

(b) en kort begrunnelse for valget av τ.

Oppgave 2.3

Siste del av oppgaven går ut på å implementere en løpelengdeenkoder- og dekoder for residualene. Du bestemmer selv en egnet datastruktur for løpelengdeenkodingen, for eksempel et array med to rader eller en liste av tupler, der første rad / element er verdien og andre rad / element er lengden på løpet. En enkel metrikk for å evaluere kvaliteten på løpelengdeenkodingen er å vurdere den approksimerte kompresjonsraten ρ gitt ved

$$\rho = \frac{\#\text{ piksler i bildet}}{2 \times \#\text{ antall løp i enkodingen}}. \tag{9}$$

Til slutt, implementer hele enkodingsprosessen fra originalbilde til løpelengdeenkodingen, samt dekodingsprosessen. I denne sammenheng kan det være lurt å revurdere den endelige tersklingen τ i kontekst med metrikken ρ samt kvaliteten til det rekonstruerte bildet.

Fra denne oppgaven skal du inkludere

(a) den endelige rekonstruksjonen av originalbildet gitt dine valg og observasjoner,

(b) valg av τ, kvantitative mål for rekonstruksjon og kompresjonsrate ρ,

(c) en diskusjon og evaluering av kvalitet samt effektivitet på kompresjonsalgoritmen din,

(d) en diskusjon om potensielle grep du kunne tenke deg vil forbedre det endelige resultatet,

samt kildekode for implementasjonen din med funksjonene `pepper_encode` og `pepper_decode` som foretar hele enkoding og dekodingsprosessen for kompresjonsalgoritmen.

Appendiks A

Her kan du finne eksempler i Matlab og Python som utfører samme oppgave med noen nyttige funksjoner. Merk at at Matlab ikke har en dedikert `fftconvolve` funksjon, så denne må implementeres fra bunnen av. Husk også at det skal regnes ut gjennomsnitt og standardavvik for hver kjøring, koden under er kun et illustrerende eksempel.

Matlab

```
% Importer bilde
image = double(imread('pepper_gray.png'));

% Implementert funksjon
function result = separable_2d(arg1, arg2)
    % ...
end

# 1d filtre
l3 = % ...
l7 = % ...
l11 = % ...

% Start timer
tic;
for k = 1:64
    % Foreta test
    conv_result = conv2(image, separable_2d(l11, l11), 'same');
end
elapsedTimeConv = toc;
fprintf('convolve2d:\t %f\n', elapsedTimeConv);

% Start timer
tic;
for k = 1:64
    % Foreta test
    conv_result = fftconvolve(image, separable_2d(l11, l11), 'same');
end
elapsedTimeFFT = toc;
fprintf('fftconvolve:\t %f\n', elapsedTimeFFT);
```

Python

```
import numpy as np
import time
from PIL import Image
from scipy.signal import convolve2d, fftconvolve

# Importer bilde
image = np.array(Image.open('pepper_gray.png'), dtype=np.float64)

# Implementert funksjon
def seperable_2d(...):
    ...

# 1d filtre
l3, l7, l11 = ...

# Start timer
start_time = time.time()

# Foreta test
for _ in range(64):
    convolve2d(image, seperable_2d(l11,l11), mode='same')

# Beregn tidsforbruk
print('convolve2d:\t', time.time() - start_time)

# Start timer
start_time = time.time()

# Foreta test
for _ in range(64):
    fftconvolve(image, seperable_2d(l11,l11), mode='same', axes=(-2, -1))

# Beregn tidsforbruk
print('fftconvolve:\t', time.time() - start_time)
```