

Obligatorisk innlevering 1

Sivert M. Skarning

Mai 2019

1 Oppgave 1

Oppgavebeskrivelse Implementer konvolusjon på egen hånd. Implementasjonen skal være generell slik at den kan anvendes på alle bilder og filtere. Det er greit å begrense implementasjonen til å fungere på gråskalabilder, men det er ikke påkrevd. Implementasjonen skal støtte minst to ulike former for utvidelse/padding. Rapporten skal inneholde dokumentasjon for at implementasjonen fungerer.

1.1 Dokumentasjon

Vedlagt ligger et script kalt convolution.py. Koden utfører konvolusjon på bilde man legger ved som argument. Scriptet gir brukeren mulighet til å velge filterstørrelse og hvilke verdier filteret har. Brukeren har også mulighet til å velge mellom to forskjellige paddingmuligheter, zeropadding og reflectionpadding. Zeropadding er implementert med egenprogramert funksjon navngitt zero-pad. Reflectionpad blir implementert med pakken numpy og er navngitt np.pad. Begge funksjonene fungerer hvor zero-pad gir svart ramme rundt generert bild og reflectionpad gir utvidelse av bilde. Uten padding ville bruk av flere filtere etterhvert krympe bilde betydig. Selve implementasjonen er inneffektivt implementert og i praksis ville man brukt en annen algoritme for å gjøre en konvolusjon. O-notasjonen for min implementasjon vil være:

$$O(n^2 * m^2)$$

Listing 1: Convolve function

```
# Convolving through image with provided filter
def convolve(img, flt, pdng_func, pdng_rws, pdng_clms):
    padded_img = pdng_func
```

```

nmb_of_pxl_in_flt = flt .shape [0]* flt .shape [1]
for i in range(pdngrws , padded_img .shape [0]-
pdngrws) :
    for j in range(pdnrgclms , padded_img .shape [1]-
pdngrcls) :
        flt_i = -1
        flt_j = -1
        pxl_vlu = 0
        for k in range(i-pdngrws , i-pdngrws+flt .
shape [0]) :
            flt_i = flt_i+1
            for l in range(j-pdnrgclms , j-pdnrgclms +
flt .shape [1]) :
                flt_j = flt_j+1
                pxl_vlu = (flt [ flt_i , flt_j ] *
padded_img [k , l]) + pxl_vlu
            flt_j = -1
        padded_img [i , j] = pxl_vlu/
nmb_of_pxl_in_flt
io .imsave('lena_out.png' , padded_img)

```

1.2 Resultat

For å teste konvolveringsalgoritmen har jeg valgt å påføre tre typer filtere på lena.png.

- Laplacian sharpening
- Mean blure
- Gaussian blure

Gaussain blure Konvolverings algoritmen fungerte bra. Her med egen-implementert zeropadding. Hadde først problemer med dårlig kontrast på bildet. Reskalerte intensiteten før bilde ble lagret, og fikk dermed et godt resultat som sett i figur 3.



Figure 1: Bilde av Lena, et vanlig eksempel i bildebehandling

Mean blure Forskjellen mellom gaussian blure og mena blure for dette eksempelet er minimal. Ved større filtere ville man kunne se større forskjell. Man trenger vanligvis å kjøre gjennom et mean filter 4 ganger for å få den samme effekten som ved et gaussisk filter. Implementasjonen av konvolverings programmet er uheldig, da man heller skulle lagd det som et set med funksjoner man kunne kalle i steder for å lage det som et skript med bilde som argument. På denne måten kunne man ha testet større filtere lettere.

Laplacian sharpening Laplacian filteret fremhever raske endringer i et bilde. På denne måten blir bilde skarpere, fordi vi kan se kantene på bilde bedere. Man vil se dette på figur 7 i forhold til figur 1.

2 Opgave 2

Oppgavebeskrivelse Implementer algoritmen for histogram spesifikasjon gitt i fagboka/slides. Implementasjonen må kunne brukes med ulike histogramspesifikasjoner, så dere må kunne lese inn et histogram. Det kan være greit å bruke et bilde til å representere et histogram. Sørg i tilfelle for at korrekt antall bøtter (piksler) og at de summerer til 255. Det er også mulig å bruke en tekstfil med f.eks. en rad til hver bøtte. Rapporten skal inneholde dokumentasjon for at implementasjonen fungerer.

$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Figure 2: Gaussian filter

2.1 Dokumentasjon

Histogram spesifikasjon er implementert. Bruker Cumulative distribution function til mapping. Det som blir gjort i koden er å importere histogrammet fra et bilde inn i en fil for så å matche barbara.png histogrammet til histogrammet i filen.

Skriptet beregner histogrammet til begge bildene. Histogrammene viser forekomsten av intensitetsnivået til pikslene. Histogrammene blir så regnet om til å vise sannsynlighet for forekomsten av en gitt intensitetsverdi. Den kumulative sannsynlighetsfunksjonen blir så beregnet ved å regne ved å legge til forrige sannsynlighetsverdi i rekken. Vi går så igjennom alle verdiene i histogrammet for orginalbildet. For hver verdi i orginalhistogrammet går vi gjennom alle verdiene i det spesifiserte histogrammet. Hvis verdien for den spesifiserte histogrammet er større eller lik verdien for det orginale histogrammet setter merkerer vi oss denne indexen og lagrer den i en annen array. Når man har gått gjennom alle verdiene. Vil vi sitte igjen med en mapping array som viser hvilken intensitet en intensitet i bilde må for at histogrammet skal matche. Programmet går så igjennom hele bilde og setter riktig intensitetsverdi for hver piksel i bilde.

Gitt de kumulative distribusjonsfunksjonene til histogrammene $H_1()$, $H_2()$. Vi finner så for hver intensitet G_1 hvilke G_2 som gjør at $F_1(G_1) = F_2(G_2)$. Ved å gjøre dette får vi mapping funksjonen forklart ovenfor¹.

¹https://en.wikipedia.org/wiki/Histogram_matching



Figure 3: Lena konvolvert med filteret på Figur 2

2.2 Resultat

Skriptet fungerer fint og resultate stemmer med forventingen. Vi ser at figur 9 viser resultatet av histogram spesifikasjon med figur 11 som input og figur 10 som spesifisert histogram. Vi ser at figur 9 er dominert av høyere intensitet. Dette er fordi figur 10 også er dominert av høyere intensitet.

3 Oppgave 3

3.1 Oppgavebeskrivelse

Utfør utjevning og skjærping på følgende bilder. Ulke metoder for både utjevning og skjærping skal utprøves og resultatet fra filtreringen skal dokumenteres i rapporten. Det er lov (anbefalt av ytelseshensyn) å ta i bruk innebygde innebygde implementasjoner for å selve konvolusjonen.

3.2 Implementasjon

I denne oppgaven har jeg implementert funksjonalitet for fire spatiale filtere.

- Gaussian filter
- Laplacian sharpening
- Highboost filtering

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Figure 4: Mean filter

- Mean blure

Gaussian blure Gaussisk utjevning utjevner figur 11 bra. Et problem med gaussisk utjevning, som også blir observert her er at kantene på bilde ikke blir bevart spesielt bra. På figur 15 ser vi at bilde er preget av støy. Vi kan også se at gaussisk utjevning på figur 17 at støyen er redusert. Også her ser vi at kantene på bilde ikke blir bra bevart.

Mean blure Mean blure er en veldig grunnleggende utjevningsmetode. Den er raskere enn det gaussisk utjevning er, og den bevarer kantene bedre. Den er dog ikke like effektiv til å fjerne støy. Mean blure med et 5x5 filter med pixel verdier på 1/25 kan bli sett i figur 18

Laplacian sharpening En Lapllacian operator er et godt valg for å fremheve detaljer i bilde. Dette vil si at kantene blir fremhevret. På figur 19 ser vi at bilde har veldig lav kontrast. Usikker på hva som forårsaker dette resultatet. De kantene som er synlige er fremhevret so forventet.

Higboost filtering Highboost filtering er en teknikk hvor man forsterker kantene og fjerner delen av bilde som ikke er skarpt. I figur 20, ser vi at kantene er fremhevret. Deler av koden i denne implementasjonen er tatt fra ². Higboost filtering fungerer ved at man trekker i fra et utjevnet versjon av

²https://github.com/lavima/itd33517_examples/blob/master/lecture05_highboost.py



Figure 5: Lena konvolvert med filteret på Figur 4

bilde. Dette fører til at man beholder både de lav og de høye frekvensene i bilde samtidig som kantene blir fremhevet³

³<https://www.scribd.com/doc/119044470/High-Boost-Filtering>

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 6: Laplacian filter



Figure 7: Lena konvolvert med filteret på Figur 6



Figure 8: Barbara



Figure 9: Barbara etter histogram spesifikasjon

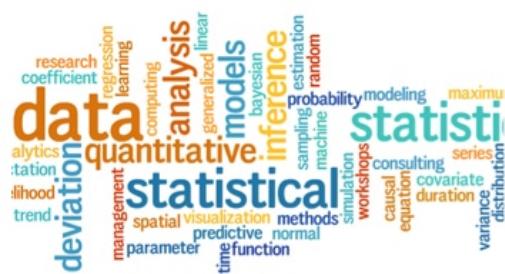


Figure 10: Det spesifiserte histogrammet er tatt fra dette bilde



Figure 11: Barbara



Figure 12: Barbara gaussian, standardavvik: 5

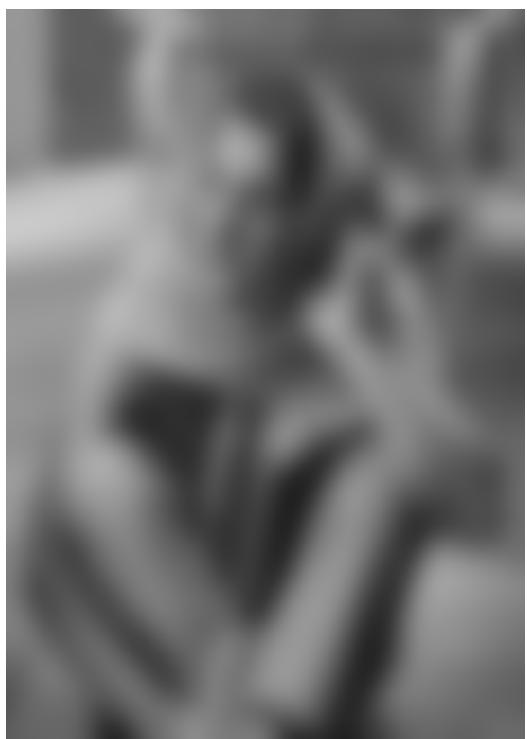


Figure 13: Barbara gaussian, standaravvik 20



Figure 14: Barbara med støy



Figure 15: Barbara med mer støy



Figure 16: Barbara med støy, utjevnet med standardavvik på 5



Figure 17: Barbara med mer støy, utjevnet med standardavvik på 5



Figure 18: Mean blure

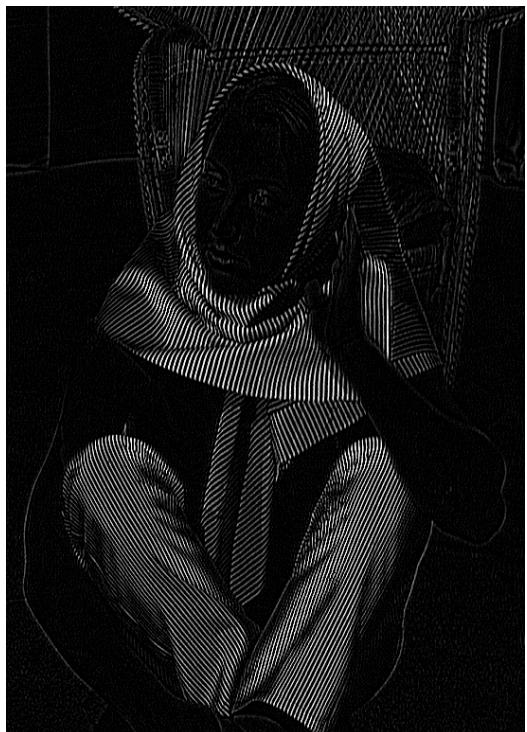


Figure 19: Laplacian sharpening



Figure 20: Barbara highboost filtering