LAB 3: Operationer på gråskalebilder

Maria Magnusson, uppdaterad av Michael Felsberg
Avdelningen för Datorseende, Institutionen för Systemteknik,
Linköpings Universitet

Februari 2016

1 Introduktion

En datorsymbol innebär att en fil ska skapas och skickas till läraren. Kopiera bilderna `baboon.tif, circle.tif, pirat.npy, pattern.npy` från `/site/edu/bb/SigInfBild/imageLab/` till ditt hembibliotek. Starta sedan en lämplig editor (t.ex. `emacs`) och `ipython`.

2 Grundläggande om bilder i PYTHON

För att kunna jobba med bilder i PYTHON , följande paket måste importeras:

```
import numpy as np
from scipy import signal, misc
from matplotlib import pyplot as plt
plt.rcParams['image.interpolation'] = 'nearest'
```

Det första paketet (numpy) är ett Python paket för att hantera matriser. De andra två paketen är Python paket för att tillhandhålla mer avancerade vetenskapliga operationer (scipy) och plottar (pyplot). Fjärde raden behövs bara för `matplotlib` med äldre version än 2. Alla PYTHON skript måste innehålla denna *preambeln*. Det rekommenderas att spara dessa rader in en fil `preambel.py` som sedan exekveras med `execfile('preambel.py')`.

2.1 Visa bilder

Det finns ett flertal kommandon för att visa bilder i PYTHON , men enklaste varianten (som dessutom liknar MATLAB) är `plt.imshow()`:

```
plt.imshow(Im,'gray')
plt.show()
```

I detta exempel innehåller `Im` själva (gråskala) bilden. Dessa kommandon automatiskt skalerar bilden så att pixlarna blir kvadratiska och att minsta bildvärdet blir svart och högsta blir vit i en linjär färgkarta med gråvärden. Det finns andra färgskalor, t.ex. `jet`, eller kan man skapa egna färgskalor.

Om man vill mappa värdena till ett annat intervall, kan man ange detta som

```
plt.imshow(Im,'gray',clim=(min,max))
```

så att bilden visas med en linjär färgkarta mellan värdena `min` och `max`. Med kommandot

```
plt.colorbar()
```

får man en legend bredvid bilden som anger vilka färger som motsvarar vilka pixelvärden. I PYTHON måste man avsluta alla plottkommandon med

```
plt.show()
```

för att skapa eller uppdatera aktuella plotten.

2.2 Egna övningar

Skapa en fil `VisaApan.py` med preambeln och nedanstående innehåll och exekvera den.

```
execfile('preambel.py')
Im = np.double(misc.imread('baboon.tif'))
plt.subplot(121)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()
plt.show()
```

Kommandot `plt.subplot(121)` delar in fönstret i 1 rad och 2 kolumner,

dvs 2 rutor och visar bilden i den första av dem. Bilden `Im` innehåller bara en färgkanal och det behövs en färgkarta+värdesområde för att kunna visa bilden med `plt.imshow()`. Om bilden innehåller tre kanaler, uppfattas detta som RGB-värden utan mappning med hjälp av färgkartan.

Mata in nedanstående kommandon. `Im` gör att bildmatrisen skrivs ut på skärmen. Då matrisen är mycket stor, kommer PYTHON automatiskt plocka bort största delen av den. Det ger ändå en bra känsla för att en bild faktiskt bara är en matris.

```
>> Im
>> np.min(Im)
>> np.max(Im)
```

FRÅGA 1: Vad är min och max-värdet på `baboon`?

I kontrast till MATLAB, behöver PYTHON bara ett enkelt `np.min()` anrop. Om man vill få ut minimalvärdet för varje kolumn, anger man att operation ska användas på dimension 0: `np.min(Im,0)`. Om man vill få minimalvärdet för varje kolumn och färgkanal, anger man en tupel med dimensionerna 0 och 2: `np.min(Im,(0,2))`.

Utvidga filen `VisaApan.py` så att apan visas till höger med högre kontrast, t ex mellan 50 och 200. Ge också bilden en lämplig titel.

FRÅGA 2: Lägg till följande rader innan `plt.show()`:

```
def on_press(event):
    print(Im[round(event.ydata),round(event.xdata)])

fig = plt.gcf()
fig.canvas.mpl_connect('button_press_event',on_press)
plt.show()
```

Obs: det är viktig med antal mellanslag innan `print` (4) och tomma raden därefter. Nu går det att klicka på bilden och få ut värdet på aktuella positionen.

Klicka mitt emellan apans ögon. Vilken koordinat (X,Y) och vilket gråskalevärde (Index) får du? Kontrollera också att du får samma värden i den vänstra och högra bilden.

---

3 Färgtabeller

Ge kommandona

```
graycmap = plt.get_cmap('gray',256)
gray_vals = graycmap(np.arange(256))
```

FRÅGA 3: set Titta på `gray_vals` och jämför med den vanliga gråskalefärgtabellen som visas i Föreläsningen. Principen är densamma, men det skiljer lite - vadå?

Ge kommandona

```
gray_vals[200:] = [1, 0, 0, 1]
plt.register_cmap('ngray',graycmap.from_list('ngray', gray_vals))
```

FRÅGA 4: Titta på `gray_vals` och ge kommandot `plt.imshow()` med `'ngray'`. Förklara hur denna färgtabell påverkar bilden till vänster.

Gör en egen färgtabell, baserad på gråskalefärgtabellen, men låt värden $\geq$ 200 visas blå och värden ≤ 50 visas gröna.

FRÅGA 5: Testa till sist färgtabellen `'jet'` på apan. Vilken färg får apans nos?

---

4 Faltning (English: Convolution)

4.1 Viktat medelvärdesbildande filter (Lågpass-filter)

Skapa en fil `FaltaApan.py` med nedanstående innehåll och exekvera den.

```
execfile('preambel.py')
Im = np.double(misc.imread('baboon.tif'))
plt.subplot(121)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()
plt.show()
```

I föreläsningen har vi byggt olika filterkärnor från dem två basfiltren `b =` $\boxed{1}\boxed{1}/2$ och `d =` $\boxed{1}\boxed{-1}$. Notera att dessa två filter har sin origo på kanten i mitten, d.v.s. förskjuta signalen med 1/2 sampel.

FRÅGA 6: Motivera varför `b` divideras med normaliseringsfaktorn 2. Anta att normaliseringsfaktorn har ett lägre värde. Vad händer med signalen?

Filterkärnan `aver =`

1	2	1
2	[4]	2
1	2	1

/16

kan skapas i PYTHON på följande sätt:

```
b = np.array([0.5,0.5])
b2 = np.convolve(b,b).reshape(1,-1)
aver = np.kron(b2,b2.T)
```

Kommandon `reshape` behövs för att skapa en 1×3 matris från faltningsresultatet, vilken behövs för Kroneckerprodukten `kron`.

Kör koden och verifiera att kärnan är korrekt. Titta även på mellanresultaten.

Applicera sedan kärnan på apan med koden:

```
Imaver = signal.convolve2d(Im,aver,'same')
```

Utvidga filen `FaltaApan.py` med denna kod och visa den filtrerade apan `Imaver` till höger om original-apan.

FRÅGA 7: Medelvärdesbildande filter används ofta för brusreduktion, men vad händer med fina detaljer, såsom kanter och linjer, i bilden?

FRÅGA 8: Vad innebär `same` i `signal.convolve2d()`-kommandot? Kontrollera genom att skriva `help(signal.convolve2d)`.

Testa sedan att falta fler gånger med `aver` för att få en kraftigare effekt. Visa den 3ggr faltade apan till höger. Var noga med att ha samma kontrastfönster på båda bilderna, så att det blir pedagogiskt att jämföra de båda bilderna.

FRÅGA 9: Vad händer när filtret `aver` appliceras upprepade gånger?

4.2 Derivering i x- och y-led, gradient

Filterkärnan `d` beräknar finita differensen. För att undvika förflyttningen med 1/2 sampel, används i många fall centrala differensen istället:
`cd =` $\boxed{1 \mid [0] \mid -1}/2$

FRÅGA 10: Använd faltningen av `b` och `d` för att beräknar `cd`.

Skriv ner motsvarande PYTHON kod och verifiera att resultatet blir `cd`. Obs: använd för värden `1.0` istället för `1`, så att filterkärnan får flyttalformat.

För beräkningen av partiella derivatorna av en bild måste man välja ett koordinatsystem. Ett vanligt val är att identifiera kolumnindexen med x-koordinaten och radindexen med y-koordinaten. Derivatan i x-led av en bild $f(x, y)$ kan sedan beräknas enligt

$$\frac{\partial f(x,y)}{\partial x} = \frac{\partial}{\partial x} * f(x,y) \approx \text{sobelx} * f(x,y).$$

På liknande sätt kan derivatan i y-led, $\frac{\partial f(x,y)}{\partial y}$, beräknas. Sobel-filtren visas nedan.

$$\text{sobelx} = \begin{array}{|c|c|c|}\hline 1 & 0 & \text{-1} \\ \hline 2 & [0] & \text{-2} \\ \hline 1 & 0 & \text{-1} \\ \hline \end{array}\ , \text{sobely} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & [0] & 0 \\ \hline \text{-1} & \text{-2} & \text{-1} \\ \hline \end{array}\ .$$

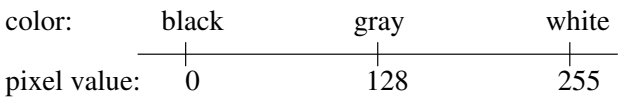
FRÅGA 11: Valet av koordinatsystemet resulterar i sobely = sobelx$^{\mathrm{T}}$. Vad händer med sobely om man placerar bildens origo längs ner till vänster?

Skapa en fil `DeriveraCirkeln.py` med nedanstående innehåll, komplettera och exekvera den.

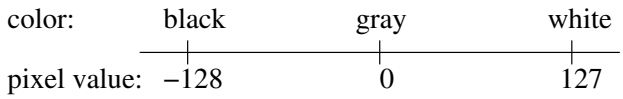
```
execfile('preambel.py')
Im = np.double(misc.imread('circle.tif'))
plt.subplot(221)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()

b = np.array([0.5,0.5])
b2 = np.convolve(b,b).reshape(1,-1)
d = ...
cd = ...
sobelx = ...

Imsobelx = signal.convolve2d(Im,sobelx,'same')
plt.subplot(223)
plt.imshow(Imsobelx,'gray',clim=(-128,127))
plt.title('sobel image')
plt.colorbar()

plt.show()
```

Lägg till kod i `DeriveraCirkeln.py` för att visa resultatet av sobely faltat med cirkeln. Visa resultatbilden nere till höger.

När en bild bara innehåller positiva värden från 0 till 255, fungerar **gray** färgtabellen så här:



De sobelfiltrerade bilderna innehåller negativa värden. När en bild bara innehåller värden i intervallet [-128 127], fungerar **gray** färgtabellen så här:



Följaktligen visas negativa värden mörka och positiva värden ljusa. Värden nära 0 visas grå.

FRÅGA 12: Titta på dina bilder och tala om varför kanten i originalbilden ibland blir mörk, ibland ljus, och ibland grå i de sobelfiltrerade bilderna.

Gradienten $\left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right)$ är en tvådimensionell vektor som pekar i den riktning där intensiteten i bilden $f(x, y)$ ökar snabbast.

FRÅGA 13: Skriv upp det matematiska uttrycket för magnituden av gradienten i bilden $f(x, y)$! (Alternativa benämningar på magnituden är längden eller absolutbeloppet.)

Lägg till kod i `DeriveraCirkeln.py` så att magnituden av gradienten på cirkeln visas uppe till höger.
Skriv också ett liknande skript för apan, `DeriveraApan.py`, alternativt ha filnamnet som argument till skriptet, och studera bilderna.

Ofta duger det bra att använda centrala differenser `cd` (i x- och y-led) istället för `sobel`, men om man vill vara noggrann är sobel bättre. För cirkeln gäller det ju att dess kant är lika stark runtom. Därmed bör också magnituden av gradienten vara lika stark runtom. Detta uppfylls bättre för `sobel` än `cd`. Visa detta med skriptet `MagnitudeComp.py`. För att se tydligt behöver man ändra kontrastintervallet. Ett tips är att börja med att sätta kontrasten mellan 0 och 160 enligt nedan och sedan justera 0 uppåt till ett värde L.

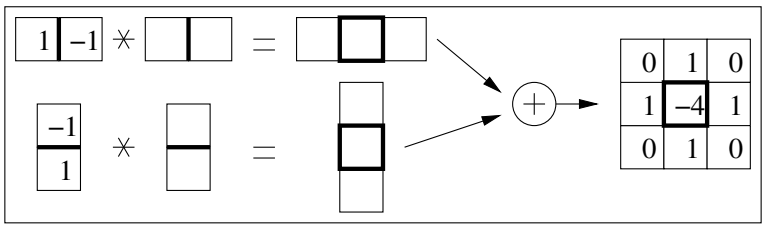
I figure(1) ska visas:	sobelMagnitude kontrast:[0 160]	centraldiffMagnitude kontrast:[0 160]
I figure(2) ska visas:	sobelMagnitude kontrast:[L 160]	centraldiffMagnitude kontrast:[L 160]

4.3 Laplace-filter (*negativt* Högpass-filter)

Laplace-operatorn definieras

$$\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) = \frac{\partial}{\partial x} * \frac{\partial}{\partial x} + \frac{\partial}{\partial y} * \frac{\partial}{\partial y}$$

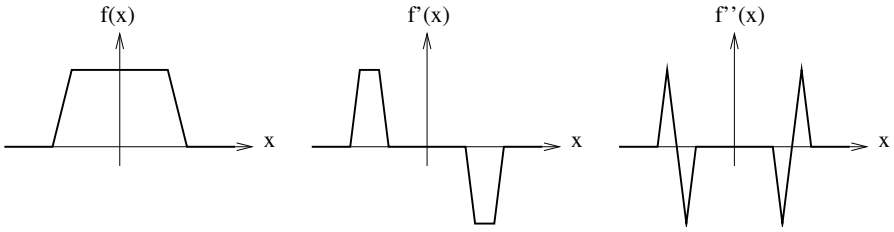
FRÅGA 14: Konstruera ett Laplace-filter genom att fylla i rutorna nedan. Som synes är Laplace-filtret givet. Dess centrum är markerat med en tjockare ram. För deriveringsoperatorerna är deras centrum faktiskt mittemellan pixlarna.



Skapa en fil `LaplaceCirkeln.py` med nedanstående innehåll, komplettera och exekvera den.

```
execfile('preambel.py')
Im = np.double(misc.imread('circle.tif'))
plt.subplot(121)
plt.imshow(Im,'gray',clim=(0,255))
plt.title('original image')
plt.colorbar()

d = ...
d2 = ...
laplace = ...

Imlaplace = signal.convolve2d(Im,laplace,'same')
plt.subplot(122)
plt.imshow(Imlaplace,'gray',clim=(-200,200))
plt.title('laplace image')
plt.colorbar()

plt.show()
```

Laplace-operatorn estimerar alltså en slags 2-D andraderivata. Nedan visas hur en 1-D andraderivata reagerar på en (approximativ) 1D rektangulär funktion $f(x)$.



FRÅGA 15: Stämmer bilden `Imlaplace` överens med skissen? Motivera ditt svar.

Skapa en fil `SharpenPirate.py` med liknande innehåll som `LaplaceCirkeln.py`. Raden med `misc.imread()` byts ut mot:

```
Im = np.load('pirat.npy')
```

FRÅGA 16: Du kan också behöva justera gränserna `(-200,200)`. Vad blir effekten av att ändra gränserna till `(-100,100)`?

FRÅGA 17: Studera bilden (`Imlaplace`) och jämför med originalbilden (`Im`). Hur stor är filtersvaret på jämna ytor (kinden t ex) jämfört med filtersvaret i regioner med snabba förändringar (fjädern t ex)? Hur påverkas kanter?

Laplace-filtret förstärker alltså snabba förändringar i bilden. Man talar om ett högpass-filter (HP). Egentligen ska detta ha omvänt tecken, bilda därför `ImHP = -Imlaplace` Komplettera sedan `SharpenPirate.py` med en bild som är summan av originalbilden och högpassbilden, dvs `Imsharp = Im + ImHP` Skapa också `ImSharp2 = Im + 2*ImHP`

FRÅGA 18: Kommentera resultatet!

---

5 Vikning (English: aliasing)

Skapa en fil `SamplaNed.py` med nedanstående innehåll och exekvera den.

```
execfile('preambel.py')
Im = np.load('pattern.npy')
plt.subplot(121)
plt.imshow(Im,'gray',clim=(-1,1))
plt.title('original pattern')
plt.show()
```

Sampla ned mönstret med faktor 2 i båda dimensioner med `Im[::2,::2]` och visa resultatet till höger.

FRÅGA 19: Hur skiljer sig bilderna? Hur kan man förklara det som händer med den övre delen av högra bilden?

Den observerade effekten är oönskad och beror på att bilden inte förbehandlades på rätt sätt.

FRÅGA 20: Hur ska bilden förbehandlas?

Upprepa behandlingen enligt fråga 9 innan nedsamplingen och visualisera resultatet. Variera antal upprepningar.

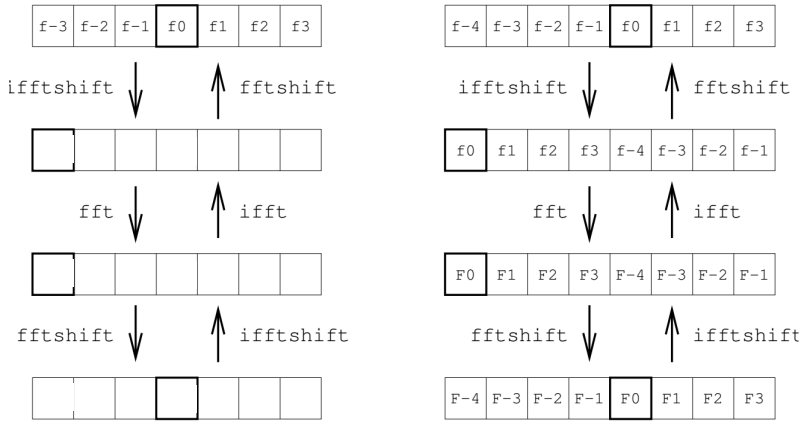
FRÅGA 21: Hur många upprepningar behövs ungefär för att undvika artefakterna? Vilken nakdel har metoden?

---

6 Visualisering av fourierspektrumet

I föreläsningen nämndes `np.fft.fftshift()` och `np.fft.ifftshift()` för visualisering. Båda funktioner ger samma resultat för ett jämn antal sampler, men olika för udda antal sampler.

FRÅGA 22: Komplettera följande bild genom att testa i PYTHON



Skapa en fil `FourierVis.py` och exekvera den:

```
execfile('preambel.py')
Im = np.load('pirat2.npy')
IM = np.fft.fftshift(np.fft.fft2(np.fft.ifftshift(Im)))
plt.subplot(221), plt.imshow(np.abs(IM),'gray'), plt.colorbar()
plt.subplot(222), plt.imshow(np.angle(IM),'gray'), plt.colorbar()
plt.subplot(223), plt.imshow(np.real(IM),'gray'), plt.colorbar()
plt.subplot(224), plt.imshow(np.imag(IM),'gray'), plt.colorbar()
plt.show()
```

FRÅGA 23: Varför visar bild 1 och 3 bara en vit punkt i mitten?

Ändra bild 1 till logaritmisk skala (OBS: undvik log(0) genom att addera 1) och visa bild 3 så att minsta värdet blir svart och 0 hamnar precis i mitten av färgskalan. Visa bild 2 bara där amplituden är större än dess medelvärde.

FRÅGA 24: Vad händer med bild 1 & 2 utan `ifftshift()`? Varför?
