

# LAB A: Operationer på gråskalebilder

Maria Magnusson  
Avdelningen för Datorseende, Institutionen för Systemteknik,  
Linköpings Universitet

January 2016

## 1 Introduktion



En datorsymbol innebär att en fil ska skapas och skickas till läraren. Kopiera bilderna `baboon.tif`, `circle.tif`, `pirat.mat` till ditt hembibliotek.

## 2 Visa bilder

### 2.1 Displaying images in MATLAB

There are two commands for displaying images in MATLAB, `imagesc` and `imshow`. The second one automatically scales the image so that the pixels become rectangular, removes the axes and sets the colormap to grayscale (with 256 different values). Consequently

```
imagesc(Im);                or                imshow(Im, []);  
axis image;                  colormap(gray(256));  
axis off;  
colormap(gray(256));
```

give the same result. In the commands above, `Im` contains the image and `[]` automatically finds the minimum and maximum of the image and displays it with a linear scale in between. Note that `[]` is not needed in `imagesc`. There exists also other colortables, i.e. `jet`, or it is possible to design your own colormap.

If another range of pixel values is desired, you can write

```
imagesc(Im, [min max]);      or                imshow(Im, [min max]);
```

giving that the image is displayed with a linear scale between the values `min` and `max`. The command

```
colorbar;
```

is useful in connection with displaying images. It shows how the colors in the image correspond to pixel values. The command

```
imshow(Im);
```

displays the image with a linear scale between 0 and 1. One exception to what have been said above is 8-bit images of class `uint8`, which may contain values between 0 and 255. For such images this command works fine:

```
imshow(Im);
```

## 2.2 Egna övningar

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

```
Im = double(imread('baboon.tif'));
figure(1)
colormap(gray(256))
subplot(1,2,1), imagesc(Im, [0 255])
axis image; title('original image')
colorbar('SouthOutside')
```

Kommandot `subplot(1,2,1)` delar in fönstret i 1 rad och 2 kolumner, dvs 2 rutor och visar bilden i den första av dem. Mata in nedanstående kommandon. Im utan semikolon gör att hela bildmatrisen skrivs ut på skärmen. Det ger en bra känsla för att en bild faktiskt bara är en matris.

```
>> Im
>> min(min(Im))
>> max(max(Im))
```

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

---

Orsaken till att man måste skriva `min(min( ))` är att först tas `min` individuellt på alla kolumner och därefter tas `min` på raden. Kontrollera gärna genom att göra `help min` eller `doc min`.



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

**FRÅGA 2:** Välj symbolen 'Data cursor' i figur-fönstret. (Den är gul med ett plustecken.) Klicka mitt emellan `apan`s 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

```
>> mycolormap0 = gray(256);  
>> mycolormapR = mycolormap0;  
>> mycolormapR(201:256, :, :) = ones(56,1)*[1 0 0];
```

**FRÅGA 3:** Titta på `mycolormap0` och jämför med den vanliga gråskalefärgtabellen som visas på sidan 6 i Föreläsning 3. Principen är densamma, men det skiljer lite - vadå?

---

**FRÅGA 4:** Titta på `mycolormapR` och ge kommandot `colormap(mycolormapR)`. Förklara hur denna färgtabell påverkar bilden till vänster.

---



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

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

---

## 4 Faltning (English: Convolution)

### 4.1 Viktat medelvärdesbildande filter (Lågpas-filter)

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

```
Im = double(imread('baboon.tif'));  
figure(1)  
colormap(gray(256))  
subplot(1,2,1), imagesc(Im, [0 255])  
axis image; title('original image')  
colorbar('SouthOutside')
```

Filterkärnan `aver` = 

1	2	1
2	4	2
1	2	1

 /16

kan implementeras i MATLAB och appliceras på `apan` med koden:

```
aver = [1 2 1; 2 4 2; 1 2 1] /16  
Imaver = conv2(Im,aver,'same');
```

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

**FRÅGA 6:** Medelvärdesbildande filter används ofta för brusreduktion, men vad händer med fina detaljer, såsom kanter och linjer, i bilden?

---

**FRÅGA 7:** Vad innebär `same` i `conv2`-kommandot? Kontrollera genom att göra `help conv2`.

---



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 8:** Vad händer när filtret `aver` appliceras upprepade gånger?

---

**FRÅGA 9:** Motivera varför `aver` divideras med normaliseringsfaktorn 16. Ändra sedan normaliseringsfaktorn till ett lägre värde. Vad händer?

---

**FRÅGA 10:** Parametrarna `valid` och `full` är alternativ till `same` i `conv2` kommandot. Vad innebär de?

---

**FRÅGA 11:** Utför faltningen nedan för hand. Verifiera sedan med `conv2` kommandot.

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 0 \\ \hline 1 & [3] & 0 \\ \hline 1 & 4 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 2 & 0 \\ \hline 0 & [2] & 0 \\ \hline 0 & 1 & 3 \\ \hline \end{array} = \begin{array}{|c|c|c|c|c|} \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline & & & & \\ \hline \end{array}$$

## 4.2 Derivering i x- och y-led, gradient

Derivatan i x-led av en bild  $f(x, y)$  kan beräknas enligt

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

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

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

Skapa en fil `DeriveraCirkeln.m` med nedanstående innehåll och exekvera den.

```

Im = double(imread('circle.tif'));
figure(1)
colormap(gray(256))
subplot(2,2,1), imagesc(Im, [0 255])
axis image; axis off; title('original image'); colorbar
sobelx = [1 0 -1; 2 0 -2; 1 0 -1] /8;
Im sobelx = conv2(Im,sobelx,'same');
subplot(2,2,3), imagesc(Im sobelx, [-127 127])
axis image; axis off; title('sobelx image'); colorbar

```



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

**FRÅGA 12:** Normaliseringsfaktorn kan inte väljas på samma sätt som för ett medelvärdesbildande filter. Varför?

---

**FRÅGA 13:** Vi valde normaliseringsfaktorn=8 i koden ovan. Hur motiverar man att normaliseringsfaktorn 8 är bra?

---

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

color:	black	gray	white
pixel value:	0	128	255

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

color:	black	gray	white
pixel value:	-128	0	127

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

**FRÅGA 14:** Titta på dina bilder och tala om varför kanten i originalbilden ibland blir mörk, ibland ljus, och bland 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 15:** 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.m` så att magnituden av gradienten på cirkeln visas uppe till höger. Skriv också ett liknande program för apan, `DeriveraApan.m` och studera bilderna.

Det finns många varianter på derivata-filter. Ett enklare filter-par är nedanstående som vi här kallar **simple** (det är dock inget etablerat namn).

$$\text{simplex} = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, \text{simpley} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}.$$

**FRÅGA 16:** Vad är en lämplig normaliseringsfaktor för **simplex** och **simpley**?

---



Ofta duger det bra att använda **simple** 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 **simple**. Visa detta med programmet `MagitudSimpleSobel.m`. För att se tydligt behöver man ändra kontrastintervallet. Ett tips är att börja med sätta kontrasten mellan 0 och 160 enligt nedan och sedan justera 0 uppåt till ett värde (?).

I figure(1) ska visas:	sobelmagngrad	simplemagngrad
	kontrast:[0 160]	kontrast:[0 160]
I figure(2) ska visas:	sobelmagngrad	simplemagngrad
	kontrast:[? 160]	kontrast:[? 160]

### 4.3 Laplacefilter (*negativt* Högpass-filter)

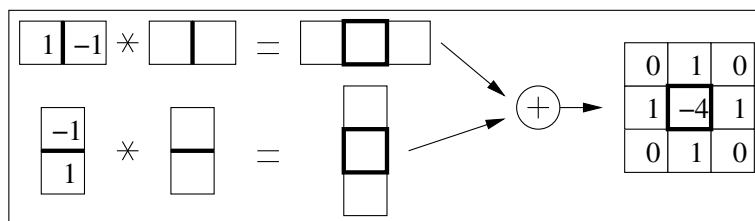
Laplace-operatorn definieras

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

Vi ska här använda ytterligare en approximation till deriveringsoperatorer i x- och y-led, nämligen

$$\frac{\partial}{\partial x} \approx \begin{bmatrix} 1 & -1 \end{bmatrix}, \quad \frac{\partial}{\partial y} \approx \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

**FRÅGA 17:** Konstruera ett Laplace-filter genom att fylla i rutorna nedan. Som synes är Laplacefiltret givet. Dess centrum är markerat med en tjockare ram. För deriveringsoperatorerna är deras centrum faktiskt mittemellan pixlarna.

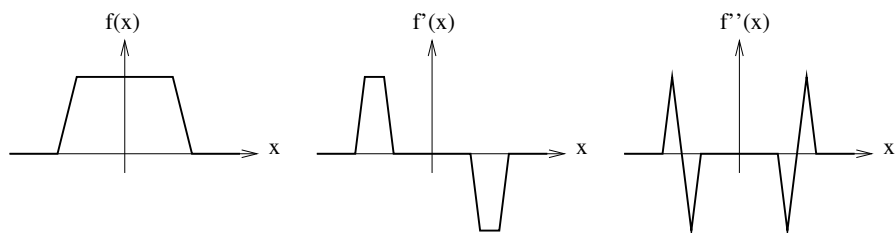


Skapa en fil `LaplaceCirkeln.m` med nedanstående innehåll och exekvera.

```
Im = double(imread('circle.tif'));
figure(1)
colormap(gray(256))
subplot(1,2,1), imagesc(Im, [0 255])
axis image; title('original image')
colorbar('SouthOutside')
laplace = [0 1 0; 1 -4 1; 0 1 0];
Imlaplace = conv2(Im, laplace, 'same');
subplot(1,2,2), imagesc(Imlaplace, [-200 200])
axis image; title('laplace image')
colorbar('SouthOutside')
```

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





**FRÅGA 18:** Stämmer bilden `Imlaplace` överens med skissen? Motivera ditt svar.

---

Skapa en fil `SharpenPirate.m` med liknande innehåll som `LaplaceCirkeln.m`. Den första raden byts ut mot:

```
load('pirat.mat')
Im = pirate;
```

**FRÅGA 19:** Du kan också behöva justera gränserna `[-200 200]`. Vad blir effekten av att ändra gränserna till `[-100 100]`?

---

**FRÅGA 20:** Studera bilden (`Imlaplace`) och jämför med originalbilden (`Im`). Hur påverkas jämna ytor (kinden t ex)? Hur påverkas kanter? Hur påverkas snabba förändringar (fjäders t ex)?

---



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

**FRÅGA 21:** Kommentera resultatet!

---