# Lab 1. Del 2- Laboration Svarsdokument

**Spara detta dokument som .pdf dokument innan ni lägger ut det på Lisam.**

*Studenternas namn och LiU-ID: (Max 2 studenter per grupp):*

*1. Johnny Elmér Jonel107*

*2. Elin Djurberg elidj214*

*Inlämningsdatum:*

*Version (ifall ni behöver lämna retur):*

**Observera att** ibland krävs det att ni ska skriva ett antal MATLAB kommandon som svar. Detta kan enkelt göras genom kopiera-klistra-in från MATLAB till svarsdokumentet.

**Observera att** ni kan skriva matriserna antingen genom att använda ”Equation i words” och skapa matriser där, eller skriva dem precis som man definierar en matris i MATLAB, eller på ett annat lämpligt sätt för att tydligt visa en matris.

## **Variabeltyper och bildvisning**

**Uppgift 1.1)**

Varför är *b/4* inte helt korrekt?

**Eftersom b har blivit en unsigned int, och ints avrundar till närmaste heltal.**

Vad får vi ut genom *b/12*? Och Varför?

**Vi får noll, pga. av 5/12 ≈ 0.417 och då b är en int avrundar den till närmaste heltal vilket i detta fall är 0.**

**Uppgift 1.2)**

Hur mycket mer minne behöver *k2* jämfört med *k* och varför? (skriv hur du räknar)

**Varje double tar upp 8 bytes av minne i MATLAB (i jämfört med uints 1 byte), alltså har vi 512 pixlar i höjd och 512 pixlar i bredd, som vardera tar upp 8 bytes.**

**512x512x8 = 2097152**

**K2 tar upp 1835008 bytes mer än k, alltså cirka 1.8MBytes mer.**

**Alltså k2 tar upp 8ggr mer minne än k.**

**Uppgift 1.3)** (men bodde inte det som är helt svart = 0 ändå visas som svart i den vita bilden)

Förklara varför medan bilden *k* visas som en korrekt bild, visas *k2* som en helvit bild.

Eftersom vår k2 innehåller värden mellan 0-255 men om bilden imshow ska visa en bild av typen double så antar imshow att 0 är svart och 1 är vitt. Eftersom de flesta värdena i bilden är över 1 (alltså värdena 1-255) så visas bilden mestadels/helt vit.

**Uppgift 1.4)** (255/2 ex 130 borde bli vit när vi delar k/255 ser vi att det blir lite 1:or och 0 or men hela bilden visas ändå svart varför??)

Förklara varför medan ***imshow(k/255)*** visas som en helt svart bild, visas ***imshow(k2/255)*** som en korrekt bild.

**Imshow(k/255) visar en hel svart bild då alla unsigned ints i k avrundas till noll eller 1:or då de delas på 255. , vilket gör att hela bilden blir svart.**

**Imshow(k2/255) blir dock ej avrundade till noll, utan istället till det närmaste motsvarande gråskalevärde. (då vi delar på 255 pga att det är hur många gråskalevärden vi kan ha.**

## **RGB färger och colormap**

**Uppgift 2.1)**

Er MATLAB kod (**tre rader**) här: (ni kan kopiera och klistra in direkt från MATLAB)

**map = ([1 1 0; 1 0 0; 0 1 1; 0.4 0.4 0.4; 1 0 1; 0.6 0.6 0.6 ])**

**image([1,2, 3;4,5,6])**

**colormap(map)**

(; gör en ny rad i image)(0 = svart så ju lägre värden ger mörkare grå)

Infoga bilden här: (**Ok att skala ner bilden i words**)

En bild som visar skärmbild

Automatiskt genererad beskrivning

## **Matriser och punktvis operation**

**Uppgift 3.1)**

*>> N(1:3:end, 1)=*  ***(var tredje rad av kolumn 1)***

*>> N(1:3:end, :)=*  ***(samma som ovan fast från alla kolonner)***

**Uppgift 3.2)**

Förklara anledningen om du får ett felmeddelande i någon av raderna.

*s1=* ***1***

*s2=* ***Matrix dimensions must agree.***

***För att utföra elementvis multiplikation krävs att båda matrisernas dimensioner är exakt likadana, men v1 är en 1x4 matris och v3 en 1x5 matris.***

## **4) Logiska operationer**

**Uppgift 4.1)**

*u1=*

*u2=*

*u3=*

*u4=*

## **5) Färgbilder**

**Uppgift 5.1)** (får vi använda rgb2gray?? För att göra den svart vit)

MATLAB kommandon ni har använt för att skapa *mygray* här under (5-6 rader kod):

bild = imread(’Butterfly.tif’):

bild = im2double(bild);

redChannel = bild(:, :, 1);

greenChannel = bild(:, :, 2);

blueChannel = bild(:, :, 3);

mygray = (redChannel+greenChannel+blueChannel)/3;

imwrite(mygray, 'grayButterfly.png');

Infoga *mygray* här (skala inte bilden efter att ni har infogat): 

## **6) Nedsampling och uppsampling**

**Uppgift 6.1)**

MATLAB kommandot här: (går med bara en rad)

b61 = mygray(1:2:end, 1:2:end);

Infoga *b61* här (**skala inte** bilden efter att ni har klistrat in i words):



**Uppgift 6.2)**

Infoga *b62* här (**skala inte** bilden efter att ni har klistrat in i words):

Del 3!!!

**Uppgift 6.3)**

MATLAB kommandon här (det krävs bara en rad per bild):

Infoga *b63\_nearest*: (**skala inte** bilden efter att ni har klistrat in i words)

b63\_nearest = imresize(b61, 2, 'nearest');



Infoga *b63\_linear*: (**skala inte** bilden efter att ni har klistrat in i words)

b63\_bilinear = imresize(b61, 2, 'bilinear');



Infoga *b63\_cubic*: (**skala inte** bilden efter att ni har klistrat in i words)

b63\_cubic = imresize(b61, 2, 'bicubic');



Diskutera vilken av dessa tre bilder och på vilket sätt ser bättre ut och liknar originalet mest:

Den som använde nearest är mest pixelerad. Cubic och linear tycker vi är relativt lika för just denna bild. Linear och cubic är mindre pixelerade och har mjukare övergångar mellan pixlarna. De är något suddigare än originalet men man ser ändå vilken bild den efterliknar.

**Uppgift 6.4)**

MATLAB kommandon här: (max två rader)

b64 = imresize(mycolorimages, 0.5, 'nearest');

b64 = imresize(b64, 2, 'nearest');

Infoga *b64*:



Beskriv de tydliga skillnaderna mellan *mycolorimage* och *b64*:

B64 är mer pixilerad än originalet.(När du samplar ner så försvinner information som ej kan återskapas när man samplar upp eftersom man slänger den infon.)

**Uppgift 6.5)** Har vi gjort rätt?? Ser ej pixlig ut??

MATLAB kommandon här: (max 10 rader men fullt möjligt med 6 rader)

R2 = imresize(mycolorimages(:,:,1), 0.5, 'nearest');

RR = imresize(R2, 2, 'nearest');

B2 = imresize(mycolorimages(:,:,3), 0.5, 'nearest');

BB = imresize(B2, 2, 'nearest');

G = mycolorimages(:,:,2);

b65 = cat(3,RR,G,BB);

Infoga *b65* här:



Vilken av bilderna *b64* och *b65* liknar originalet mest? Förklara varför!

b65 liknar originalet mest. Anledningen är att vi endast samplar ner 2 av färgkanalerna och alltså håll den gröna konstant. Detta betyder att vi inte slänger bort lika mycket information. Dessutom består bilden till största del av grönt och eftersom vi inte komprimerar den gröna färgen så ser de gröna bitarna normala ut.

**Uppgift 6.6)** stämmer detta??

Hur mycket minne utryckt i megabytes (MB) krävs för att spara en pixlar stor **färgbild** i uint8-format? (skriv hur du räknat)

Uint8 = 1byte/pixel

4000x2000x3(rgb)x1(uint)= 24 000 000

24 000 000/8 = 3MB

Svar: 3MB

Uint8 = 1byte/pixel

4000x2000x3(rgb)x1(uint byte)= 24 000 000

Svar: 24MB

Hur mycket minne krävs för den komprimerade bilden om vi samplar ner två av bildens färgkanaler till hälften så stor i varje led men behåller den tredje som den var? (skriv hur du räknat)

4000x2000x1(byte för grön)x0.5(byte för röd)x0.5(byte för blå)= 2 000 000

2 000 000/8= 0,25MB

Svar 0,25MB

4000x2000x1(byte för grön)x0.5(byte för röd)x0.5(byte för blå)= 2 000 000

Svar 2MB

Nu är väl den i bytes redan då ska vi väl ej della på 8?????

**Uppgift 6.7)**

MATLAB kommandon här: (max 7 rader men fullt möjligt med 3 rader)

R2 = imresize(mycolorimages(:,:,1), 0.5, 'nearest');

R3 = imresize(R2, 2, 'nearest');

G2 = imresize(mycolorimages(:,:,2), 0.5, 'nearest');

G3 = imresize(G2, 2, 'nearest');

B = mycolorimages(:,:,3);

b67 = cat(3,R3,G3,B);

imwrite(b67, 'b67.png')

Infoga *b67* här:



Förklara varför *b65* ser bättre ut än *b67* (är mer lik originalet).

Anledningen till att b65 ser bättre ut är att i b67 har vi samplat ner den gröna kanalen. Bilden består till mestadels av grönt samt att gult får man genom att ha maximala värdet på både grönt och rött. Så genom att samplan ner den gröna kanalen gör vi dels bakgrunden som är 50% av bilden pixlig samt vi gör de gula blommorna pixelerade (eftersom det består av grönt) och eftersom blommorna har så tunna blad/detaljrika syns det tydliga när det är pixelerat.

**Uppgift 6.8)** vi fick 14 rader??? Räknas cat

MATLAB kommandon här: (max 13 rader men fullt möjligt med 9 rader)

R = mycolorimages(:, :, 1);

G = mycolorimages(:, :, 2);

B = mycolorimages(:, :, 3);

bild1 = R + G + B;

bild2 = R - G;

bild3 = R + G - 2\*B;

bild2 = imresize(bild2, 0.5, 'nearest');

upbild2 = imresize(downbild2, 2, 'nearest');

downbild3 = imresize(bild3, 0.5, 'nearest');

upbild3 = imresize(downbild3, 2, 'nearest');

RR = (bild1/3)+(upbild2/2)+(upbild3/6);

GG = (bild1/3)-(upbild2/2)+(upbild3/6);

BB = (bild1/3)-(upbild3/3);

b68 = cat(3,RR,GG,BB);

Infoga *b68* här:



Ser ni att *b68* ser bättre ut än *b65* och är nästan identisk originalbilden?

Ja det är väldigt likt originalet.

**Uppgift 6.9)**

Hur mycket minne krävs för B1 (utryckt i MB)? (skriv hur du räknat)

4000x12000x3(rgb)x8(bitar)= 1 152 000 000bitar

1 152 000 000bitar/(8bitar/bytes)= 144 000 000bytes

Svar: 144MB

Hur mycket minne krävs för B2 (utryckt i MB)? (skriv hur du räknat)

Hur mycket minne krävs för B3 (utryckt i MB)? (skriv hur du räknat)

**Glöm inte att spara dokumentet som *.pdf* innan ni lägger ut det på Lisam.**