# 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. Max Wiklundh – maxwi824*

*2. Magnus Kling – magkl572*

*Inlämningsdatum: 2023-02-01*

*Version (ifall ni behöver lämna retur): 1*

**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?

Svar: Då b endast är allokerad 1 byte är det inte möjligt att spara värden som inte är heltal i intervallet [0, 255]. b har inte tillräckligt mycket minne för att spara hela svaret (1.25) och avrundar till det godkända intervallet för att få ett ungefärligt svar.

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

b/12 = 0, a/12 = 0.4167. b är endast allokerad 1 byte vilket ger 256 olika värden, 0 till 255. Då a blir 0,4167 försöker operationen b/12 ge svaret som är närmast det resultatet, närmast utav de 256 tillåtna värdena. Då resultatet är mindre än 0.5 avrundas svaret ner till 0.

**Uppgift 1.2)**

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

k2 kräver 2 097 152 bytes och k kräver 262 144 bytes.

2 097 152/ 262 144 = 8 gånger mer minne.

Detta beror på hur k2 sparar varje värde i matrisen som en double vilket allokerar 8 bytes istället för 1 byte per ”pixel” i matrisen.

**Uppgift 1.3)**

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

Det beror på hur imshow tolkar datatypen. Om datatypen är uint8 tolkar imshow 0 som svart och 255 som vit men om datatypen är double tolkar imshow 0 som svart och 1 som vit. Problemet uppstår då konverteringen via k2 = double(k) då funktionen behåller alla värden som matrisen k har. För att få samma bild måste vi dela alla värden i matrisen med 255. Intervallet [0,255] ska kopplas till intervallet [0,1].

**Uppgift 1.4)**

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

Det beror på hur imshow tolkar k och k2. Då k är en uint8 så kopplas 0 till svart och 255 till vit. Om vi delar k med 255 kommer alla värden vara mellan 0 och 1 vilket på skalan för uint8 kommer resultera i enbart svart färg. För imshow av k2/255 tolkar imshow bilden som en double och tolkar då 0 som svart och 1 som vit. Genom att då skala ner alla värden i k2 till värden mellan 0 och 1 kop

## **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; 0.2 0.2 0.2; 1 0 1; 0.8 0.8 0.8])

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

colormap(map)

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

En bild som visar torg

Automatiskt genererad beskrivning

## **Matriser och punktvis operation**

**Uppgift 3.1)**

*>> N(1:3:end, 1)=*

ans =

1

10

*>> N(1:3:end, :)=*

ans =

1 2 3

10 11 12

**Uppgift 3.2)**

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

*s1= 1*

*s2= Arrays have incompatible sizes for this operation.*

*Det går inte att enhets multiplicera en 1x4 matris med en 1x5 matris då det finns 4 element i en vektor och 5 element i den andra vektorn*

## **4) Logiska operationer**

**Uppgift 4.1)**

*u1=*

*3×3 logical array*

*0 0 0*

*1 1 1*

*1 1 1*

*u2=*

*3×3 logical array*

*0 0 0*

*0 0 1*

*0 0 0*

*u3=*

*3×3 logical array*

*0 0 0*

*1 1 1*

*1 1 1*

*u4=*

*3×3 logical array*

*0 0 0*

*0 0 0*

*1 1 1*

## **5) Färgbilder**

**Uppgift 5.1)**

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

bild = imread('Butterfly.tif');

fargbild = double(bild) / 255;

mygray = (fargbild(:,:,1) + fargbild(:,:,2) + fargbild(:,:,3))/3;

imwrite(mygray, 'mygray.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 = mygrey(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):

En bild som visar växt, tvåskiktsdjur

Automatiskt genererad beskrivning

**Uppgift 6.3)**

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

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

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

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

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

En bild som visar växt, blomma

Automatiskt genererad beskrivning

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

En bild som visar växt, blomma

Automatiskt genererad beskrivning

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

En bild som visar växt, blomma

Automatiskt genererad beskrivning

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

**Uppgift 6.4)**

MATLAB kommandon här: (max två rader)

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

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

Infoga *b64*:



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

Bilden har tappat mycket detaljer då endast hälften av pixelvärdena behölls och ”smetades” ut för att fylla samma storlek.

**Uppgift 6.5)**

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

Infoga *b65* här:

En bild som visar växt, blomma

Automatiskt genererad beskrivning

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

b65 liknar mest originalet då endast den röda och blåa kanalerna blev komprimerade jämför med b64 där alla tre kanaler blev komprimerade.

**Uppgift 6.6)**

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

4000 x 2000 = 8 \* 10^6 pixlar

3 byte per pixel för färgbild i uint8-format

8 \* 10^6 \* 3 = 2,4 \* 10^7 byte = (ungefär) = 24 MB

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)

(r, g, b) = 1 + 1+ 1 = 3 bytes per pixel

(r/2, g, b/2) = ½ + 1 + ½ = 2 bytes per pixel

4000 x 2000 = 8 \* 10^6 pixlar

8 \* 10^6 \* 2 = 1.6 \* 10^7 byte = (ungefär) = 16 MB

**Uppgift 6.7)**

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

R3 = imresize(R, 0.5, 'nearest');

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

G3 = imresize(G, 0.5, 'nearest');

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

b67(:,:,1) = R3

b67(:,:,2) = G3

b67(:,:,3) = B

Infoga *b67* här:



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

Anledningen varför b65 ser bättre ut än b67 beror på andelen grön färg i bilden. Då stora delar av bilden är gröna eller består av färger med mycket grönt betyder det att mer detalj försvinner då den gröna kanalen är komprimerad än när den blå blir komprimerad.

**Uppgift 6.8)**

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

bild1 = R + G + B

bild2 = R - G;

bild3 = R + G - 2\*B;

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

bild2 = imresize(bild2, 2, 'nearest');

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

bild3 = imresize(bild3, 2, 'nearest');

b68(:,:,1) = bild1/3+bild2/2+bild3/6;

b68(:,:,2) = bild1/3-bild2/2+bild3/6;

b68(:,:,3) = bild1/3-bild3/3

Infoga *b68* här:

En bild som visar växt, blomma

Automatiskt genererad beskrivning

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

*b68 ser bättre ut än b65 och ser nästan ut som originalbilden då komprimeringen inte direkt skedde på färgkanalerna. Då bild1 inte komprimerades fanns delar av den ursprungliga bilden kvar vilket såg till att detaljnivån inte blev för låg.*

**Uppgift 6.9)**

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

4000 x 12000 = 48 000 000 pixlar

(R, G, B) = 3 byte per pixel

48 000 000 \* 3 = 144 000 000 byte = (ungefär) = 144 MB

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

(4000 \* 1/4) x (12000 \* ¼) = 48 000 000 \* 1/16 = 3 000 000 pixlar

(R, G, B) = 3 byte per pixel

3 000 000 \* 3= 9 000 000 bytes = (ungefär) = 36 MB

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

(4000 \* 1/4) x (12000 \* ¼) = 48 000 000 \* 1/16 = 3 000 000 pixlar

(R, G, B) = 3 byte per pixel

3 000 000 \* 3= 9 000 000 bytes = (ungefär) = 36 MB

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