

In [34]:

```
import matplotlib.pyplot as plt
import numpy as np
from imageio import imread
from skimage.color import rgb2hsv, yiq2rgb, rgb2yiq
x = 255; y = 30; gråtoner = np.array([[int(i*255/x) for i in range(x)] for j in range(y)])
mona = imread("bdy.jpeg")
hsv_mona = rgb2hsv(mona)
yiq_mona = rgb2yiq(mona)
```

## Litt om forrige forelesning: Fargerbilder og fargerom

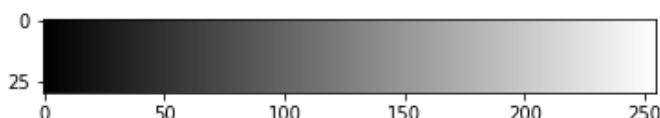
### Det som skal skje:

- Fargerom
  - RGB
  - HSI
  - YIQ
  - CMY(K)
  - Honorable mentions
- Ting dere har lært, men i farger
- Falske og pseudo farger

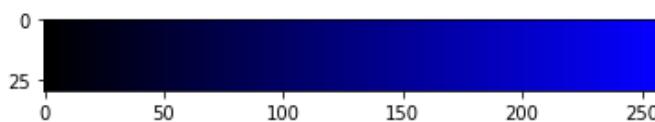
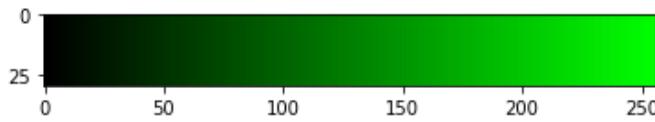
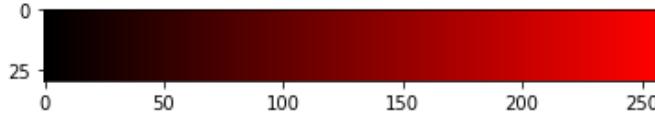
## Fargerom

En farge er en vektor: [255] eller [0,0,255]

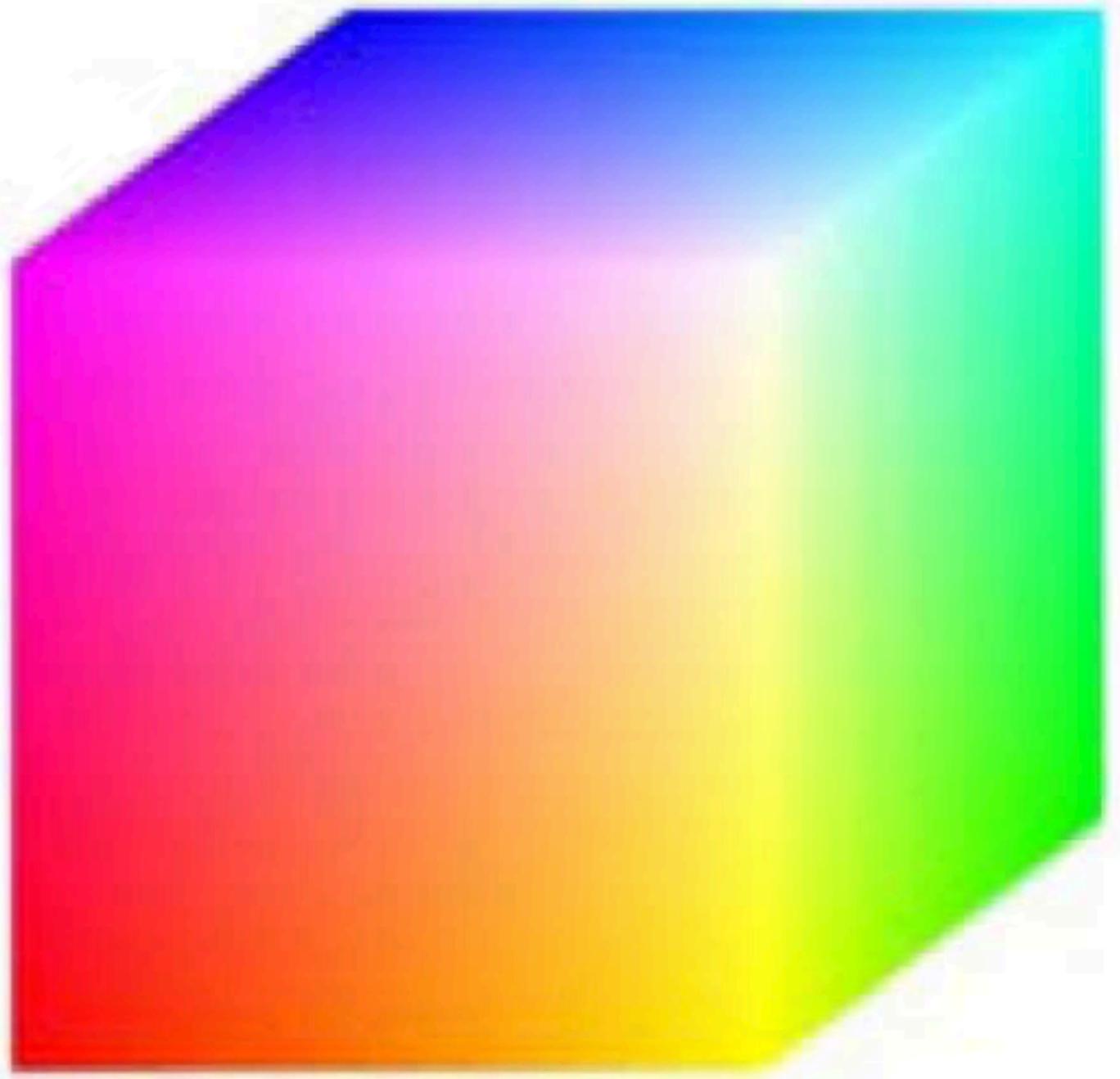
Opp til nå har vi hatt 1D fargerom:



Men vi går nå over i farger, som gir oss 3D fargerom:



Men akkurat som at det er vanskelig å visualisere en posisjon i 3D rom gitt som en verdi på en slider, er det vanskelig å visualisere det slik. Vi bruker derfor 3D figurer:

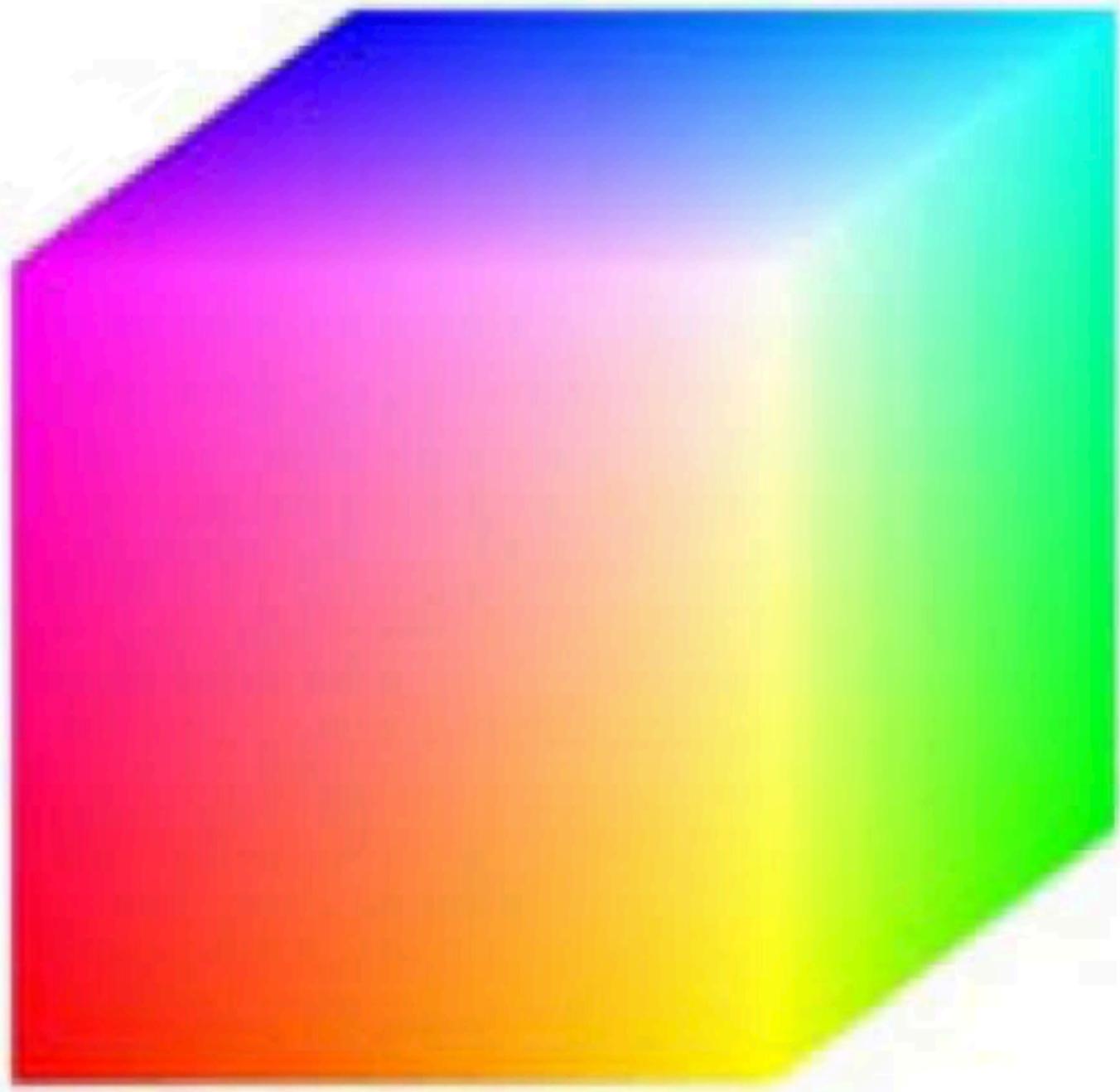


## Hvorfor forskjellige fargerom?

Akkurat som når vi bytter til frekvensdomenet er det ting som egner seg bedre, eller er mindre regnekrevende, i andre fargerom.

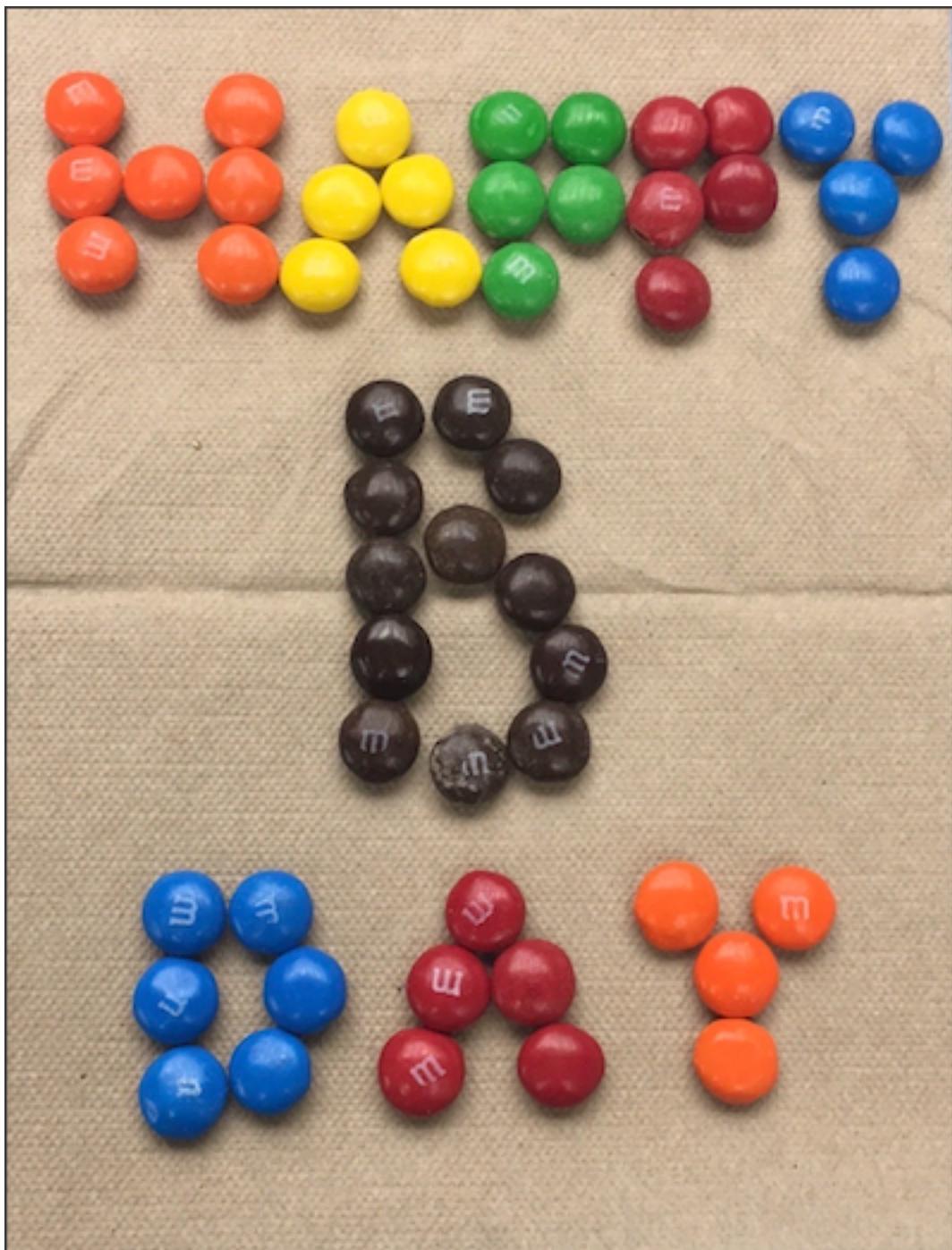
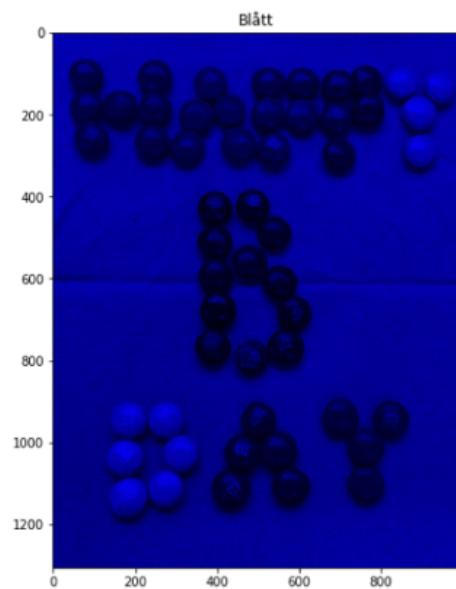
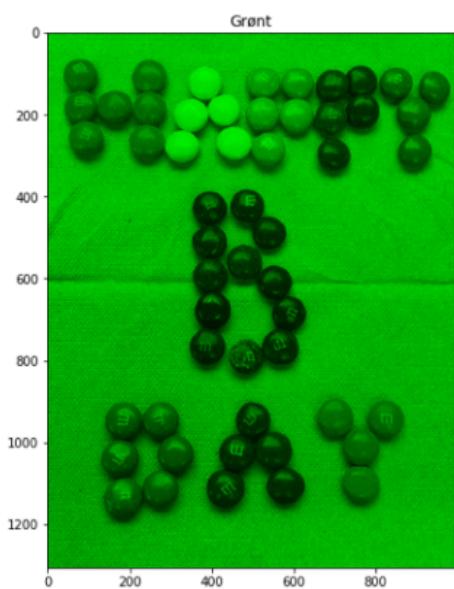
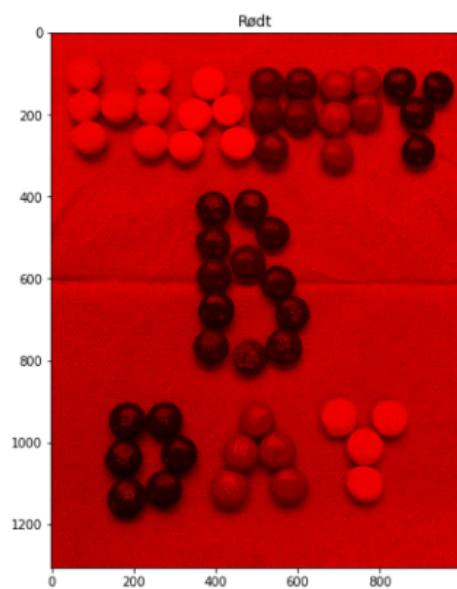
Feks: Kan du tenke deg å histogramutgjevne i RGB?

**RGB: Rødt Grønt Blått**



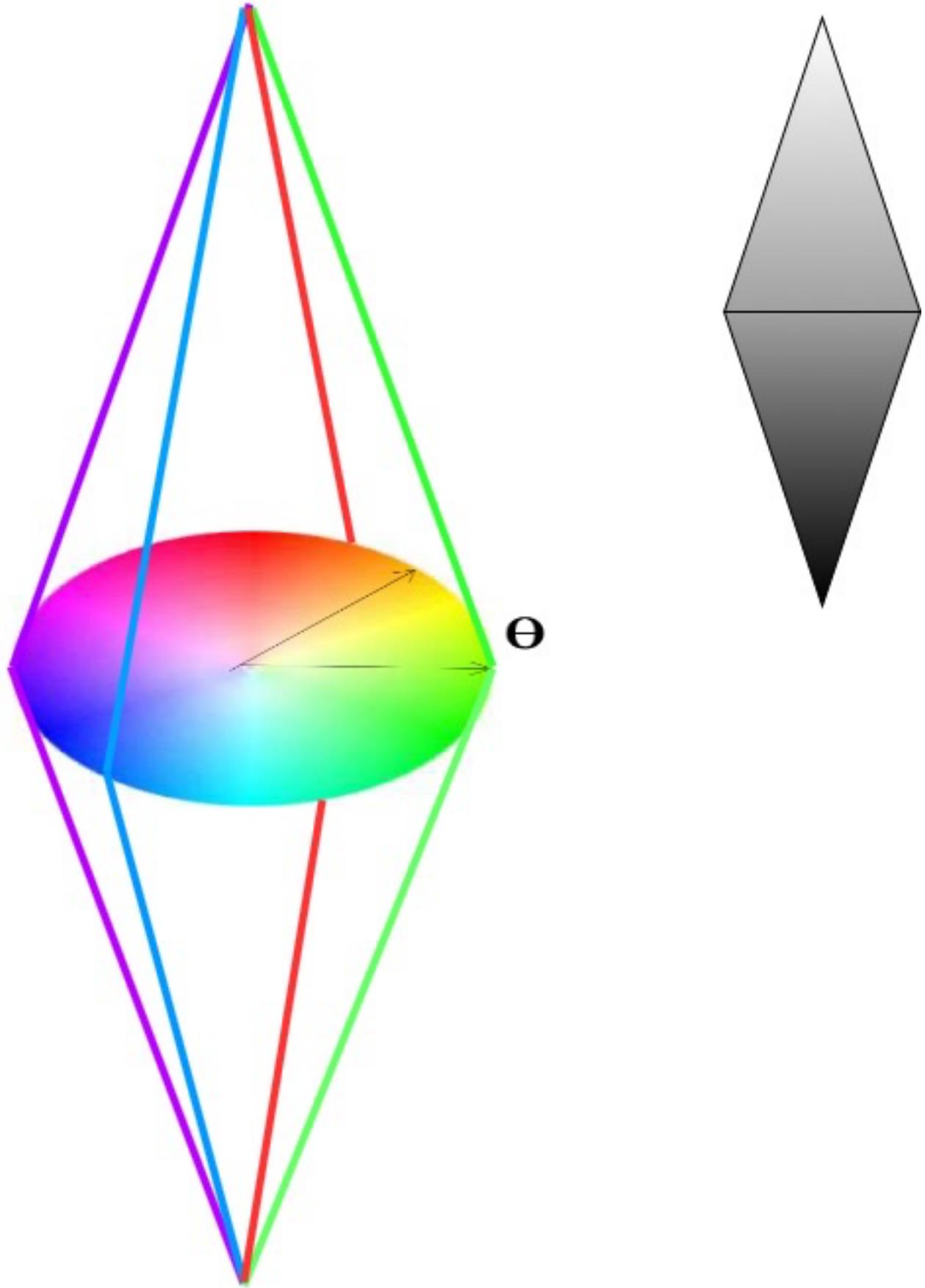
Default rom for de fleste informatikere. Brukes i kode fordi

- Brukt av skjermer
- Brukt av øynenes "sensorer"
- Brukt av sensorer for synlig lys



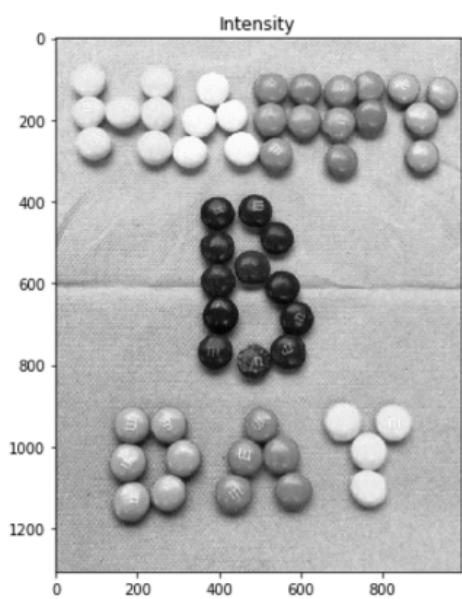
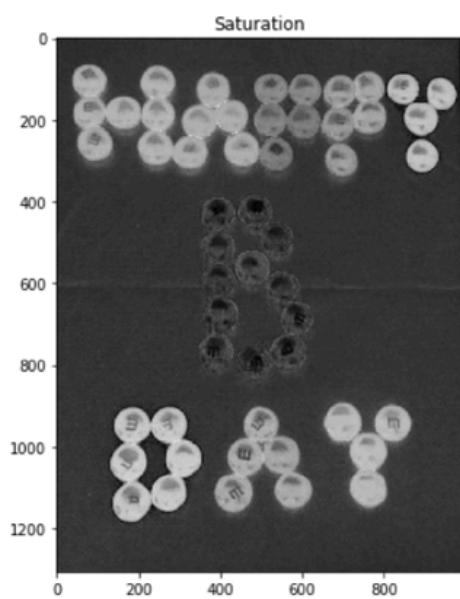
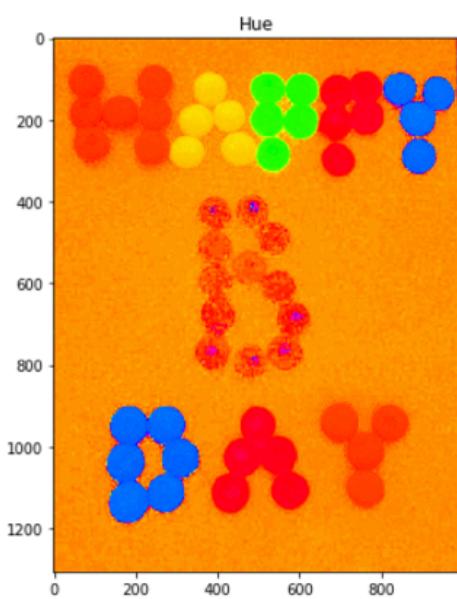
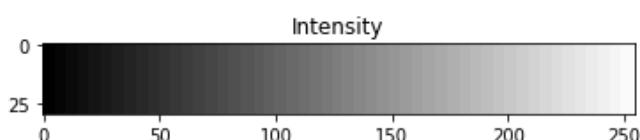
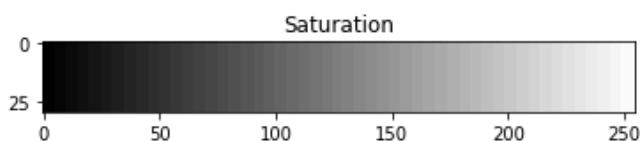
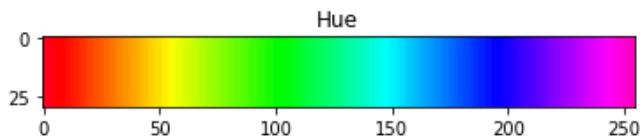
RGB er ikke veldig intuitivt for mennesker. Det viser seg også at mange metoder kan gjøres bedre i andre rom.

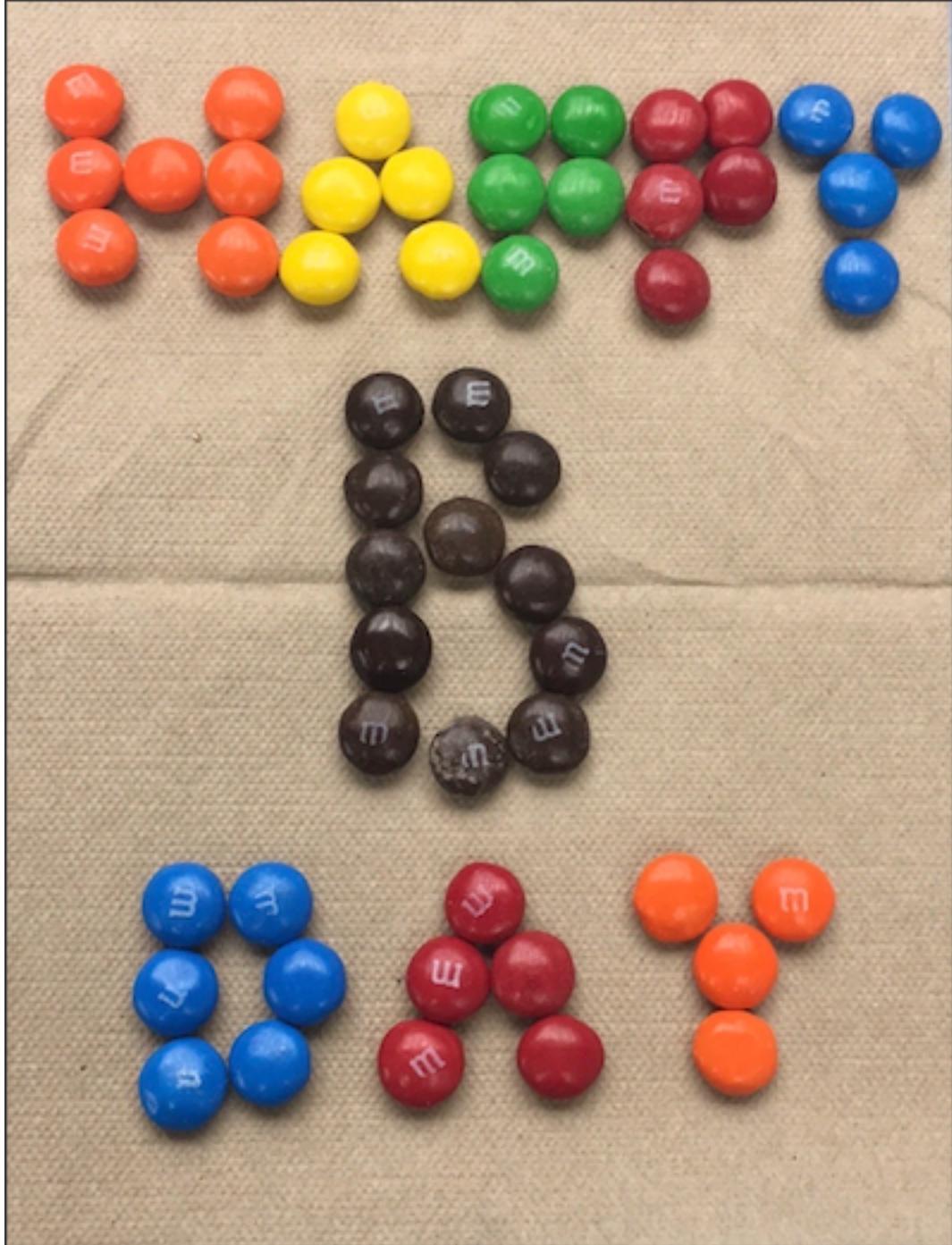
## **HSI: Hue Saturation Intensity**



For dere som liker matte: Dette er polarkoordinater!

Et intuitivt rom. Sliderne sier nå mer for mennesker enn ved RGB:





## Hvordan komme til dette rommet?

Formler:

$$H = \theta \text{ if } B \leq G \text{ else } 360 - \theta$$

$$S = 1 - \frac{3\min(RGB)}{R + G + B}$$

$$I = \frac{R + G + B}{3}$$

$$\theta = \cos^{-1} \left( \frac{\frac{1}{2}((R - G) + (R - B))}{\sqrt{(R - G)^2 + (R - B)(G - B)}} \right)$$

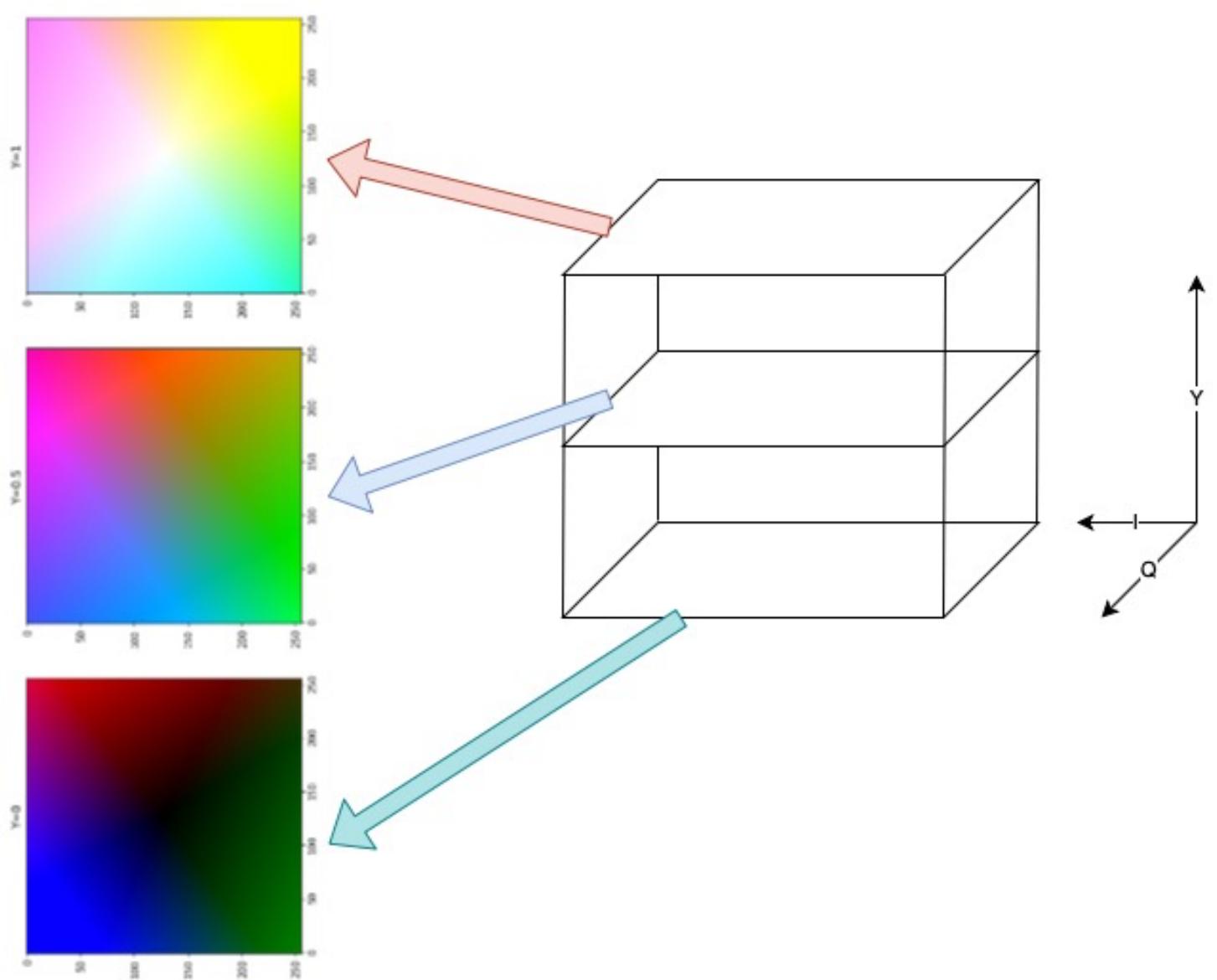
Og det finnes formler for å komme tilbake også.

s. 27 i foilene viser alt dette mer nøyaktig. Dere må bare vite at de finnes!

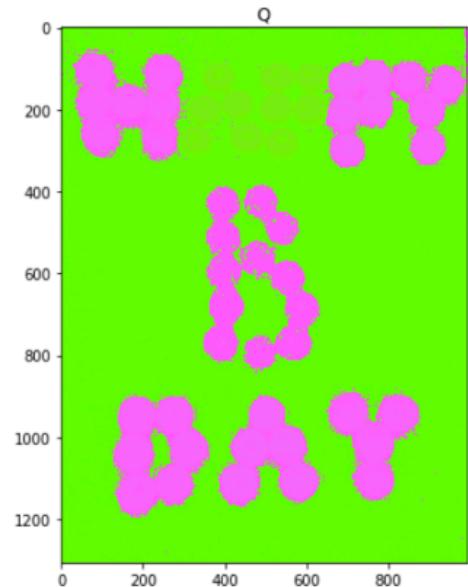
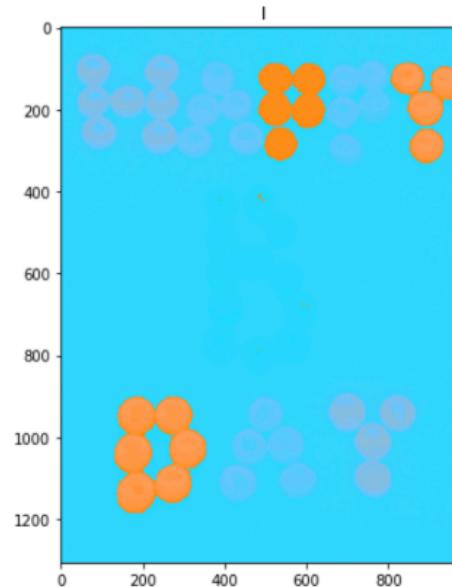
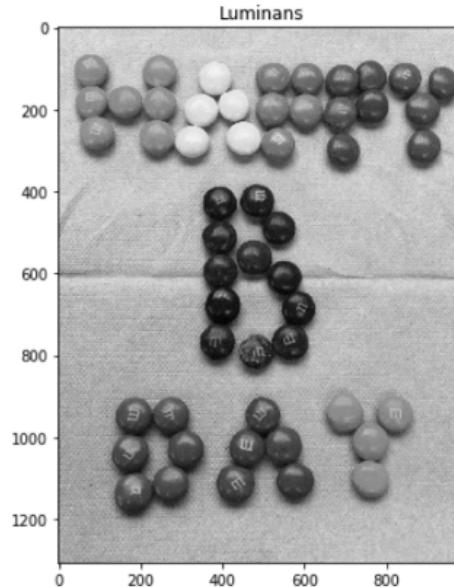
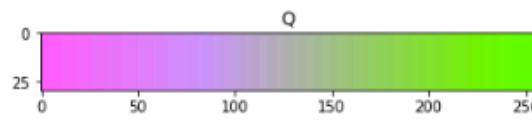
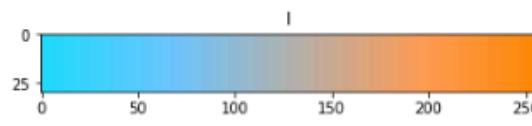
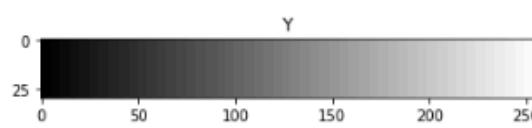
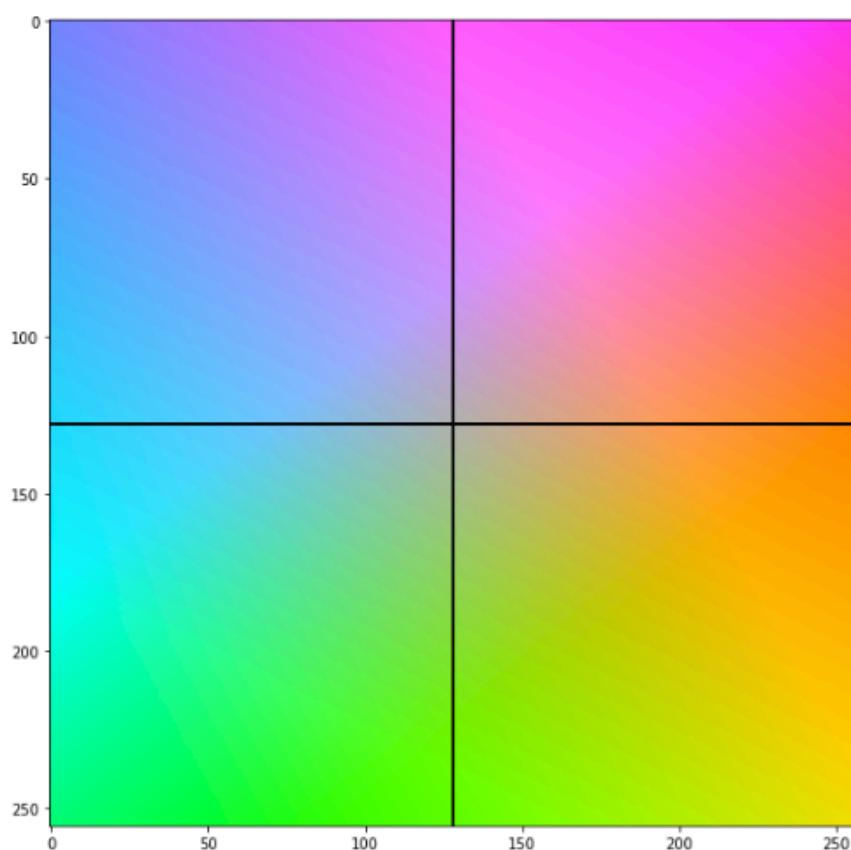
## **YIQ: Luminans Hue Metning**

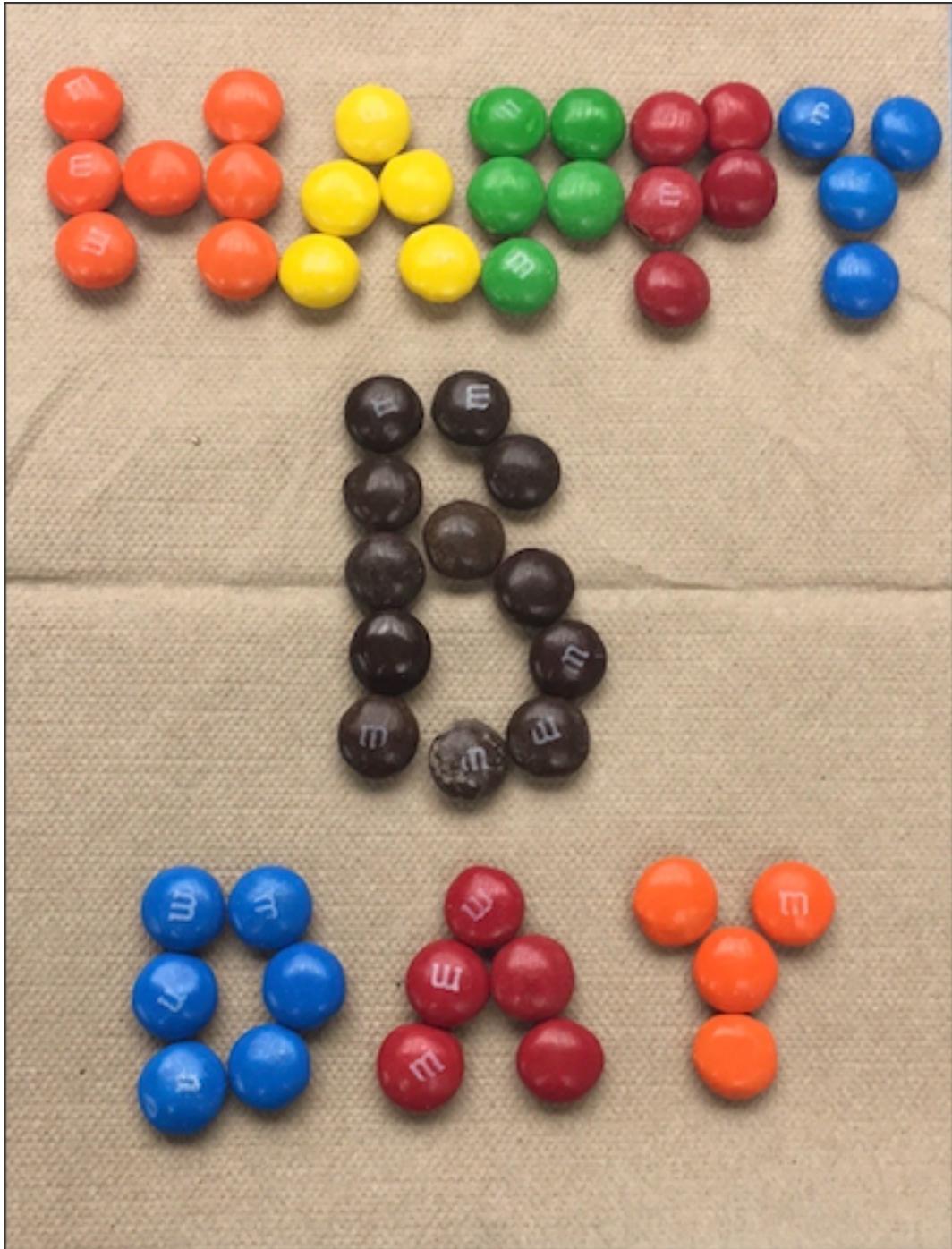
I og Q: Farge (kromasitet)

Y = Lyshet



Standard for TV og video i USA. Viktig for dere: Brukes i JPEG.





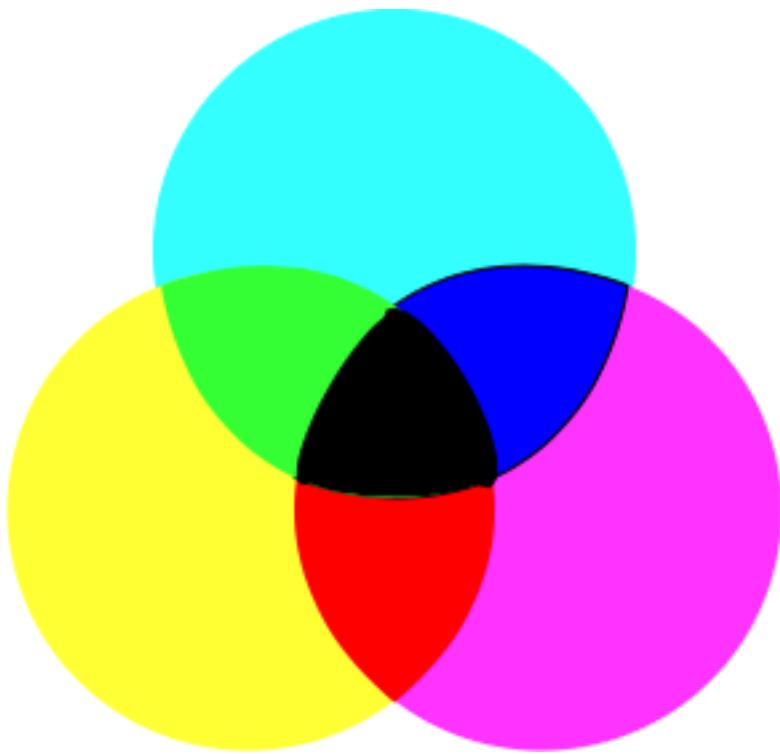
Luminansen kan vises alene, som på svart-hvitt TV. Gjerne scroll opp til HSI og se forskjellen mellom Intensity og Luminans!

## Hvordan komme til dette rommet?

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Tilsvarende finnes for å komme tilbake, s. 33 i foilene.

# **CMY(K): Cyan Magenta Yellow (Black)**



CMY blir noen ganger omtalt som CMYK hvis svart også er med. Brukes i printere, følger maling-logikk, og skal bli mørkere jo mer du legger til.

## **Hvordan komme hit?**

$$C = 1 - R$$

$$M = 1 - G$$

$$Y = 1 - B$$

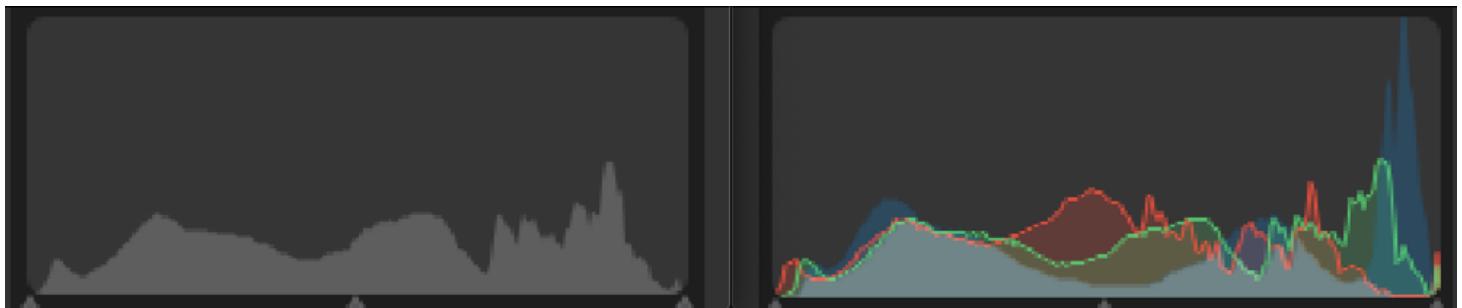
## Honorable mentions

- HSV og HSL er sylinder-versjonen av HSI-diamanten
- Alle de forskjellige stokkingene av HSI, feks IHS, er ofte bare HSI
- YCbCr
  - Digital
  - Brukt i MPEG-kompresjon
- YUV
  - Brukt i analog TV

**Ting dere har lært, men i farger**

**Histogram**

Et ekte RGB-histogram vil nok være en dimensjon per farge, som betyr at kurven er i 4D rom. Vi er i stedet oppfinnsomme: Legg alle RGB oppå hverandre, eller bare bruk Intensitet eller Luminans fra HSI eller YIQ:



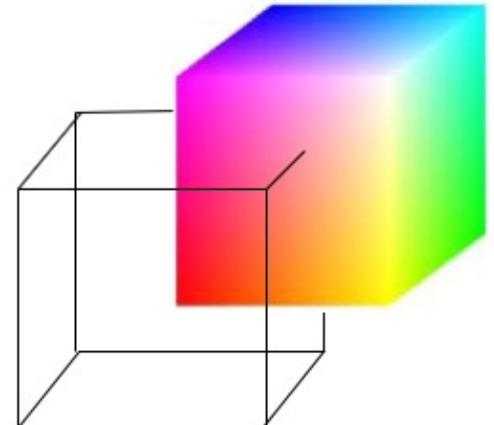
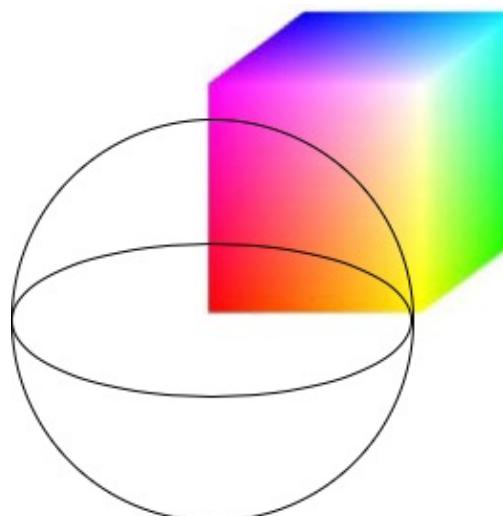
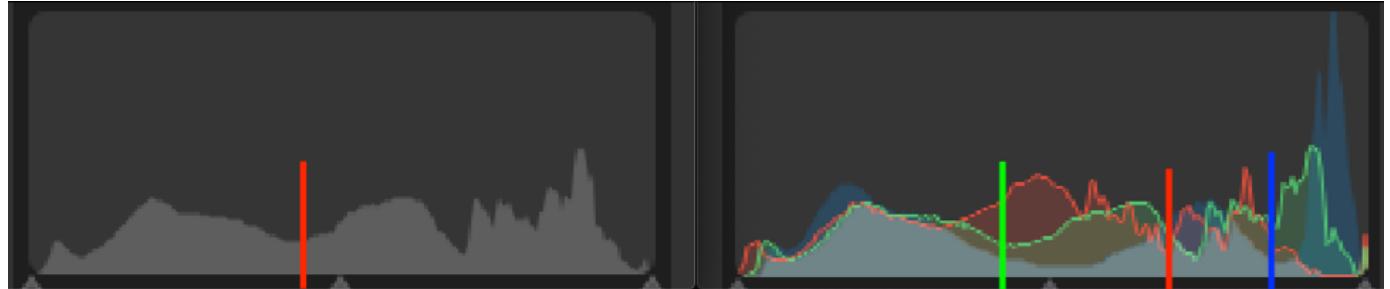
## Histogramutjevning og gråtonetransformer ( $a f(x,y) + b$ )

Bør nok ikke utføres i RGB, heller i HSI, HSV, YIQ, på intensitet / luminans. Fargene blir bevart, kontrasten blir forbedret. Men så klart, det spørs hva du vil!

## Terskling

Mange muligheter:

- en enkel terskel i I på HSI
- en terskel for hver R G og B ( så en boks i RGB-rommet)



- en elliposide i RGB-rommet

- andre figurer i andre fargerom

## Sharpening

Gjøre bilder skarpere ved å legge til Laplace. Under har jeg addert LoG til originalen for et skarpere bilde.

Forskjellig effekt, eller mindre regnekrevende, i forskjellig fargerom. Her: Samme effekt for en tredjedel av jobben.

In [27]:

```
# Kode kan ignoreres

from skimage.color import yiq2rgb, rgb2yiq
from scipy.signal import convolve2d;
img = imread("Juno2.jpeg")[200:2000,200:1500,:];
img_yiq = rgb2yiq(img);

h = [[- 2, -8, -14, -16, -14, -8, -2],
      [- 8, -24, -24, -16, -24, -24, -8],
      [-14, -24, 30, 80, 30, -24, -14],
      [-16, -16, 80, 160, 80, -16, -16],
      [-14, -24, 30, 80, 30, -24, -14],
      [- 8, -24, -24, -16, -24, -24, -8],
      [- 2, -8, -14, -16, -14, -8, -2]] # LOG filter
h = np.array(h)/255; # Interessant effekt om dette ikke gjøres

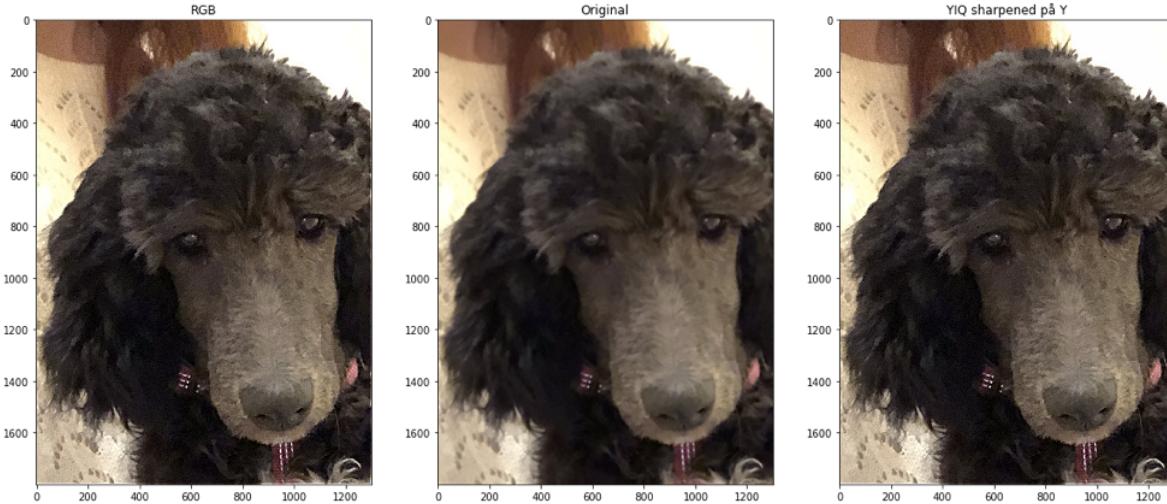
img_rgb_sharp = np.zeros(img.shape)
for i in range(3):
    img_rgb_sharp[:, :, i] = (img[:, :, i] + 2*convolve2d(img[:, :, i]
, h, mode="same"))/255 # Sharp alle kanaler

y_skarpere = convolve2d(img_yiq[:, :, 0], h, mode="same"); # Sharp 1
uminans
img_yiq[:, :, 0] = img_yiq[:, :, 0]+y_skarpere*2

fig = plt.figure(figsize=(20,20));
fig.add_subplot(1,3,1); plt.imshow(img_rgb_sharp); plt.title(
"RGB")
fig.add_subplot(1,3,2); plt.imshow(img); plt.title(
"Original")
fig.add_subplot(1,3,3); plt.imshow(yiq2rgb(img_yiq)); plt.title(
"YIQ sharpened på Y"); None
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

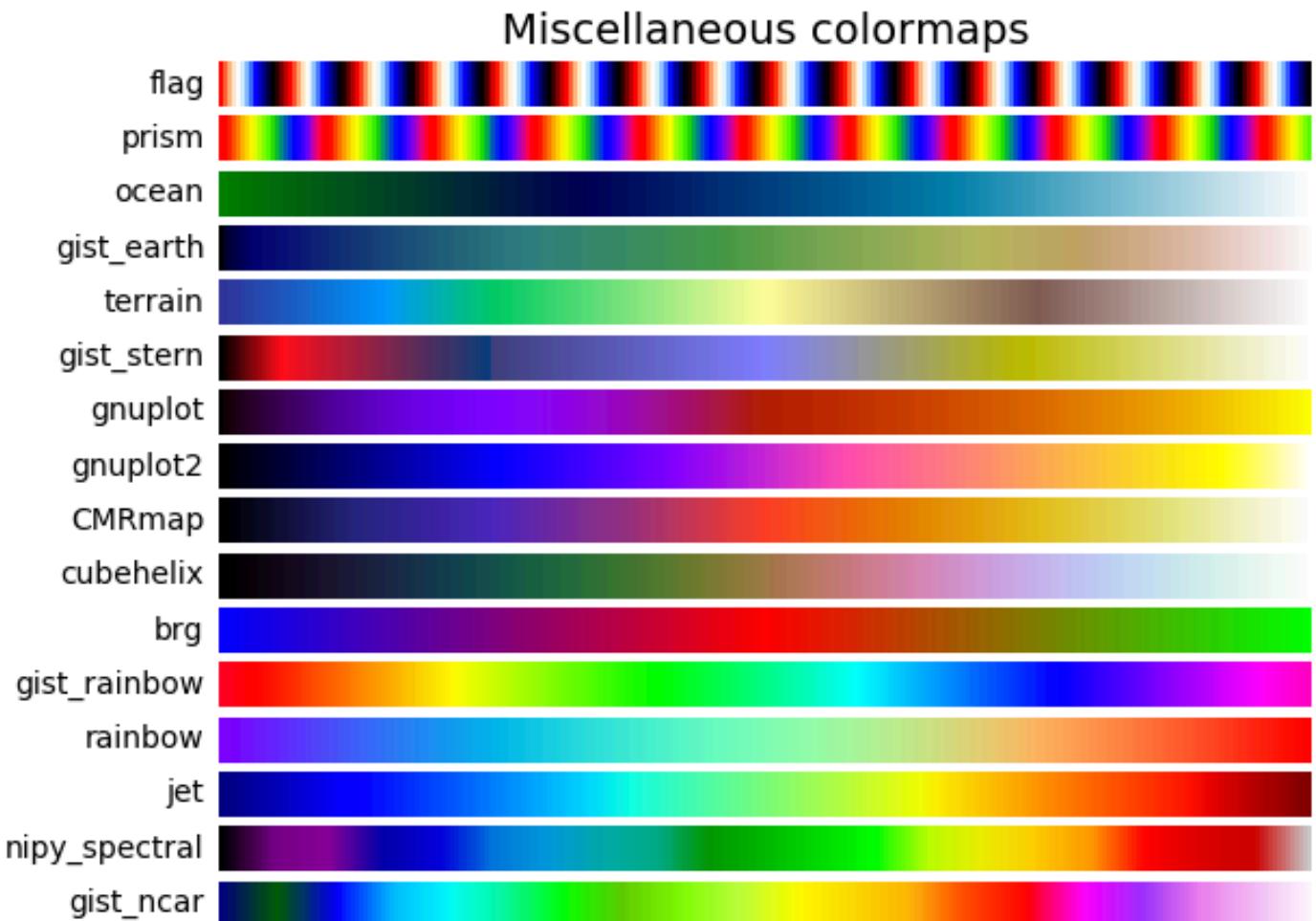


## Alt dere kan, kan ovesettes til farger

Men dere må passe på hvilket rom det gir mening å gjøre ting i!

# Falske- og pseudo farger

Hvis du har et gråtonebilde, bruker du `cmap="gray"`. Dette er et pseudofarge-bilde. Det finnes mange `cmap`, og felles for dem er at du bruker en LUT for å mappe en verdi til en farge:



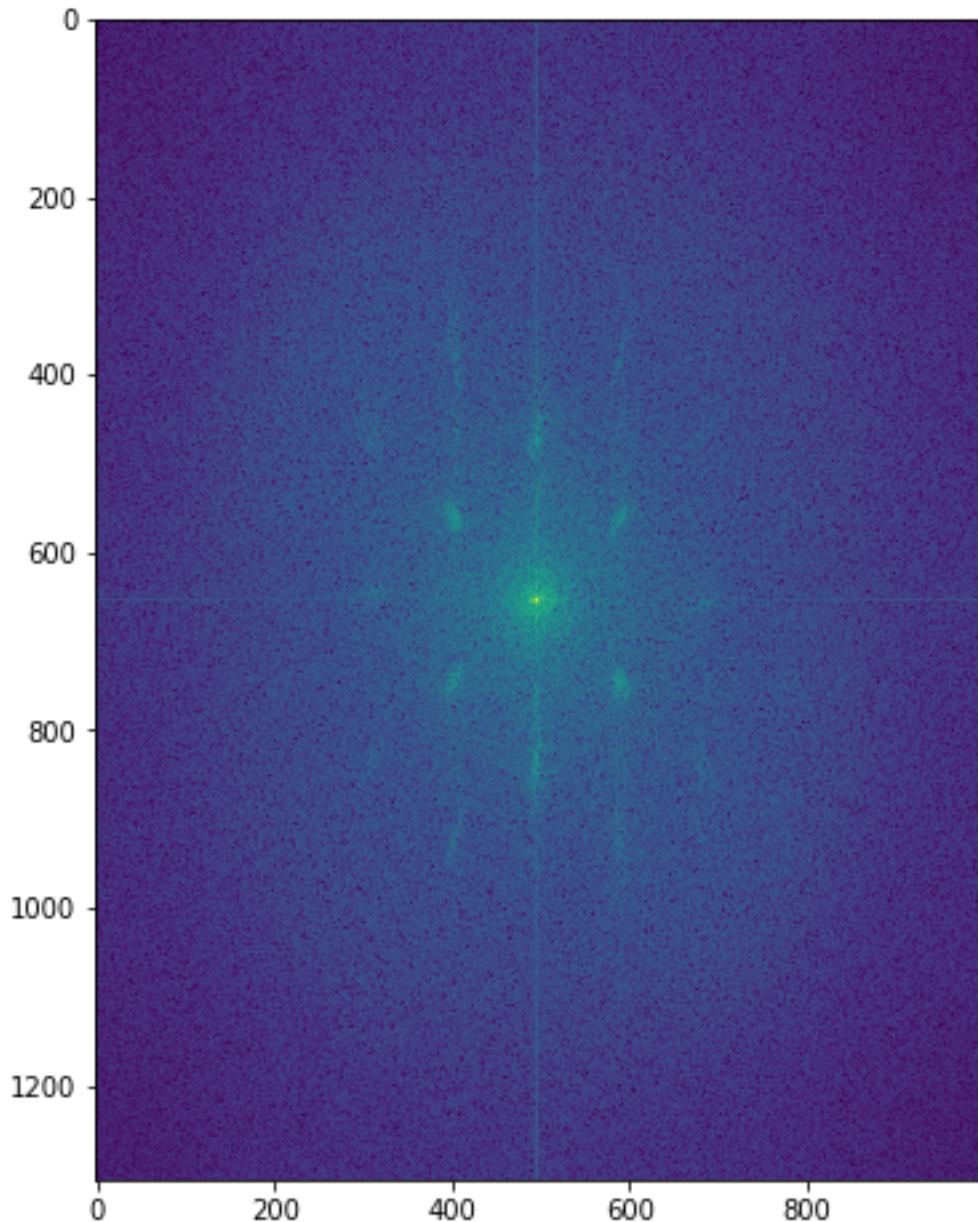
Feks når du viser spekter:

In [30]:

```
plt.figure(figsize=(8,8)); plt.imshow(np.log(1+abs(np.fft.fftshift(np.fft.fft2(yiq_mona[:, :, 0])))))
```

Out[30]:

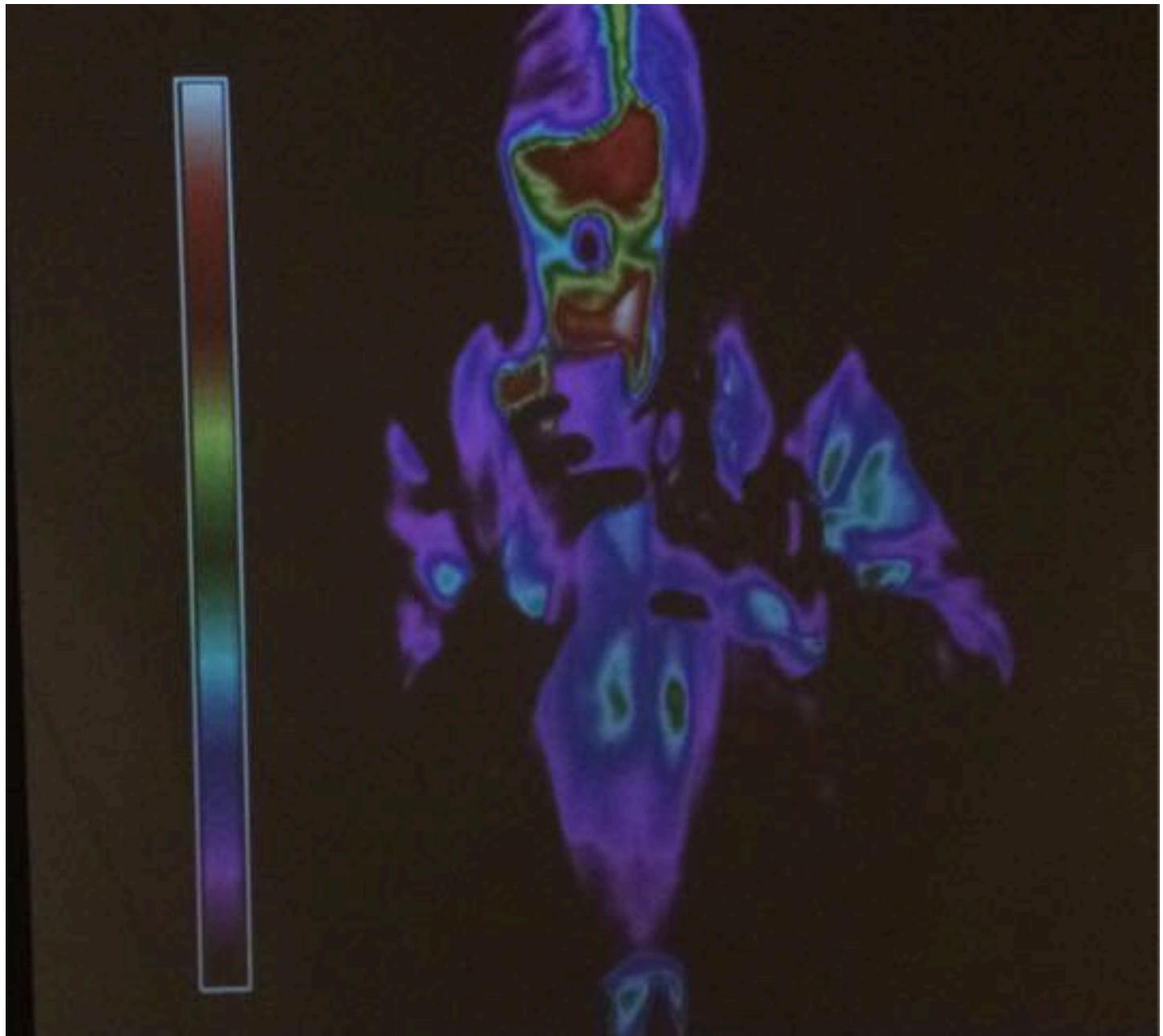
```
<matplotlib.image.AxesImage at 0x1c379d72d0>
```



Spesielt bra hvis du vil uttrykke små nyanser i gråtone, eller små endringer, som kanskje ikke ville vært synlig for menneskeøyet til vanlig.

Det behøver ikke å være gråtone-bilder heller, det kan være ikke-synlig lys, frekvenser, vær-data.

Her er et eksempel på falske farger på et varmekamera:



Verdien "varmt" blir mappet ved en LUT til rødt, og "kaldt" mappes til lilla.

**Og det var det! Lykke til på eksamen :)**