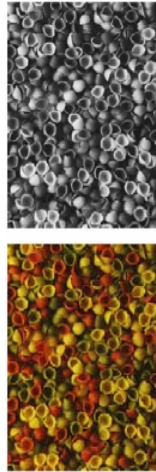


# INF 2310 – Digital bildebehandling

## FARGER OG FARGEROM

- Temaer i dag :
  1. Farge, fargesyn og deteksjon av farge
  2. Fargerom - fargemodeller
  3. Overganger mellom fargerom
  4. Fremvisning av fargebilder
  5. Fargetabeller
  6. Utskrift av fargebilder
  7. Pseudo-farger og falske farger
  8. Bildebehandling på fargebilder
- GW, Kapittel 6, unntatt 6.5 "Color Transformations" men med 6.5.5



# Motivasjon

- Vi kan skille mellom tusenvis av fargenyanser
- Farger gjør det lett å skille mellom objekter
  - Både visuelt
  - Og ved digital bildeanalyse
- Vi må
  - Vite hvilket fargerom vi skal bruke til forskjellige oppgaver
  - Kunne transformere fra ett fargerom til et annet
  - Kunne lagre fargebilder rasjonelt og kompakt
  - Kjenne teknikker for utskrift av fargebilder

# Fargen på lyset

- Lyset fra sola kan best beskrives ved strålingen fra et "svart legeme" med  $T \approx 5780$  K (Planck-kurve).

$$M(\lambda) = \frac{2\pi^5 hc^2}{15} \frac{1}{\lambda^5 \left( e^{\frac{hc}{\lambda kT}} - 1 \right)}$$

- Jorda ligger  $d \approx 1.5 \cdot 10^{11}$  m fra sola, og da er strålingen fra en kule med radius  $r \approx 6.96 \cdot 10^8$  m redusert til den irradiansen vi måler på toppen av jordatmosfæren:

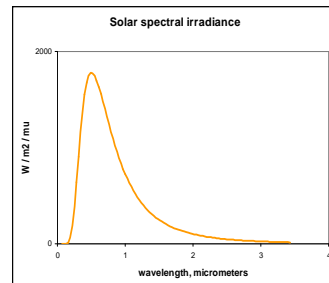
$$E_0(\lambda) = M(\lambda) \left( \frac{r}{d} \right)^2$$

- Synlig lys ligger mellom 0.4 og 0.7  $\mu\text{m}$

- Bølgelengden for maksimum i Planck-kurven er omvendt proporsjonal med T (Wien's lov)

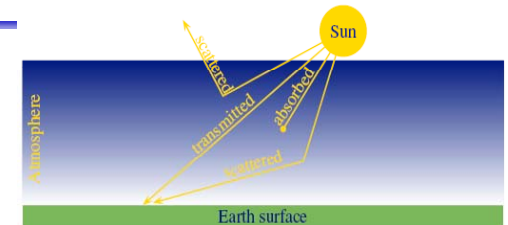
$$\lambda_{\text{max}} = \frac{2897}{T}$$

der  $\lambda_{\text{max}}$  er gitt i  $\mu\text{m}$



# Spredning, absorpsjon, transmisjon

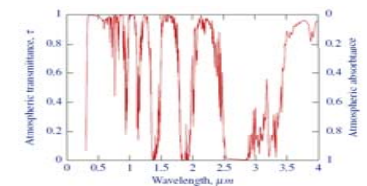
- I atmosfæren kan lyset
  - Spres
  - Absorberes
  - Transmitteres



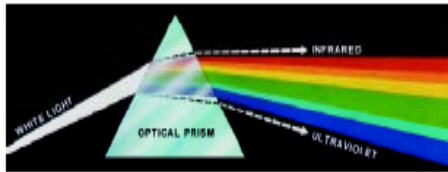
- Det absorberes mye stråling i UV og IR – det meste av enkle molekyler ( $\text{H}_2\text{O}$ ,  $\text{CO}_2, \dots$ )

- Vertikal-absorpsjonen pga gasser i normalatmosfæren er som vist til høyre.

- Absorpsjonene er proporsjonal med den luftmassen lyset må gå gjennom.



## Et prisme kan vise oss fargene i lyset



Rød	~ 625 - 740 nm
Oransje	~ 590 - 625 nm
Gul	~ 565 - 590 nm
Grønn	~ 500 - 565 nm
Cyan	~ 485 - 500 nm
Blå	~ 440 - 485 nm
Fiolett	~ 380 - 440 nm

F15 21.05.2013

INF 2310

5

## Fargen på et objekt

- Objektets farge bestemmes av
  - Det lyset som faller på objektet
  - Den andelen av dette lyset som *reflekteres*.
- Dermed er fargen avhengig av
  - Spektral-fordelingen til lyset som faller på objektet
  - Spektralfordelingen til refleksjonen
- Refleksjonsegenskapene til objektet bestemmes av
  - Kjemiske pigmenter
  - Fysiske overflate-strukturer
  - Dette bestemmer hvilke bølgelengder som reflekteres, absorberes eller transmitteres

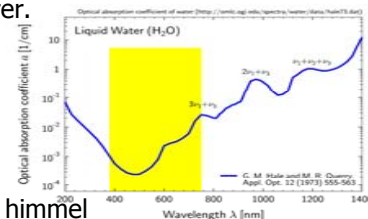
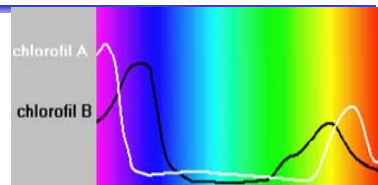
F15 21.05.2013

INF 2310

6

## Grønne blader, blått hav, blå himmel

- Klorofyll reflekterer grønt, men absorberer rødt og blått lys
  - Sommer: Klorofyll dominerer, og vegetasjonen er grønn
  - Høst: Mengden klorofyll minsker, xantofyll og beta-caroten dominerer.
- Absorpsjon av synlig lys i vann:
  - Vann ser derfor blått ut.
  - Alger gjør vannet blå-grønt.
- Rayleigh-spredning i luft  $\approx \lambda^{-4}$ :
  - Blått lys spres mer enn rødt => blå himmel
  - Samme årsak til rød solnedgang.



F15 21.05.2013

INF 2310

7

## Fargesyn

- Retina er følsom for lys mellom 350 og 760 nm.
- Ved sterk infrarød stråling kan vi oppfatte stråling helt opp til 1000 -1050 nm som lys, selv om dette er varmestråling.
- Simultane forskjeller ned til 1 nm i blå-grønt og gult kan sees, mens forskjellen må være minst 10 nm i dyp rødt og fiolett.
- Dette betyr at vi kan skille mellom ca 100 rene farger.

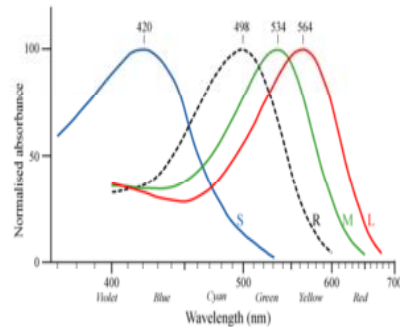
F15 21.05.2013

INF 2310

8

# Tre-farge syn

- Tre typer fargefølsomme tapper i retina:
  - S - rundt 420 nm, (2%). Dette er de mest sensitive tappene.
  - L - rundt 564 nm, (65%).
  - M - rundt 534 nm, (33%).



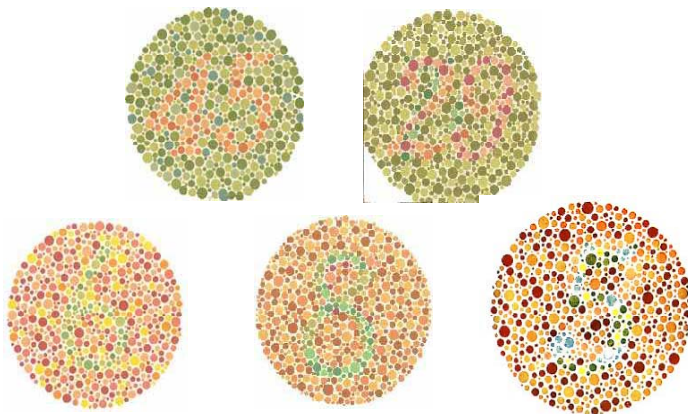
- Tappene analyserer lyset, og finner den dominerende bølglengden.
- Stavene (R) gir
  - gråtone-syn
  - Er ikke sensitive for rødt lys

# Tristimulus-verdier

- Fargen reduseres til tre verdier – **tristimulus-verdier**
- Mengden av alle slike mulige verdier utgjør vårt perseptuelle fargerom
- Det er noen kombinasjoner av stimuli som ikke er mulige
  - Vi kan ikke stimulere M-tappene uten å få noe respons fra S og L tappene samtidig
- En liten andel har nedsatt fargesyn eller er "fargeblinde"
  - Grønnblindhet mer utbredt enn rødblindhet
  - Oppfatter farger ved hjelp av to komponenter

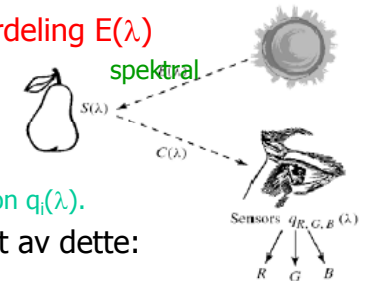
# Tester for fargeblindhet

- Med normalt syn ser du tallene 45, 29, 6, 8 og 5.
- Ved rød-grønn fargeblindhet ser du tallet 2 nederst til høyre.



# Tre integraler gir RGB

- **Lys fra en kilde med spektralfordeling  $E(\lambda)$** 
  - treffer et objekt med refleksjonsfunksjon  $S(\lambda)$ .
  - Reflektert lys detekteres av tre typer tapper med spektral lysfølsomhetsfunksjon  $q_i(\lambda)$ .
- Tre analoge signaler kommer ut av dette:



$$R = \int E(\lambda) S(\lambda) q_R(\lambda) d\lambda$$

$$G = \int E(\lambda) S(\lambda) q_G(\lambda) d\lambda$$

$$B = \int E(\lambda) S(\lambda) q_B(\lambda) d\lambda$$

# RGB primærfarger

- Commission Internationale de l'Éclairage, (CIE)  
(The International Commission of Illumination)

har definert primærfargene:

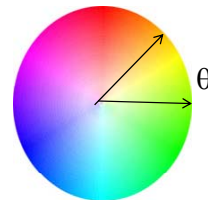
- Blå: 435.8 nm
- Grønn: 546.1 nm
- Rød: 700 nm

# Beskrivelse av farger

- En farge kan beskrives på forskjellige måter (kalles **fargerom**)
  - RGB
  - HSI (Hue, Saturation, Intensity)
  - CMY (Cyan, Magenta, Yellow)
  - pluss mange flere som vi snart skal se....
- HSI er viktig for hvordan vi beskriver og skiller farger.
  - I – Intensitet: hvor lys eller mørk er den
  - S – saturation/metning: hvor "sterk" er fargen
  - H – dominerende farge (bølgelengde)
  - H og S beskriver sammen fargen og kalles **kromatisitet**

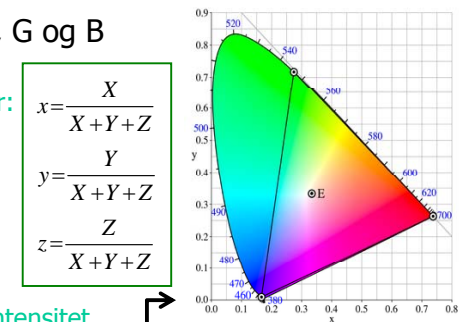
# Om kromatisitet

- Kromatisitet og intensitet (lyshet) beskriver en farge.
- Kromatisitet beskriver både dominerende bølgelengde og fargens metning.
- To forskjellige gråtoner har samme kromatisitet, men forskjellig intensitet.
- Tenk deg en sirkel der bølgelengden varierer med vinkelen  $\theta$ .
  - Full metning ytterst ved radius  $r=1$ .
  - Minker  $r$  langs samme  $\theta$ , så endres kun metningen.



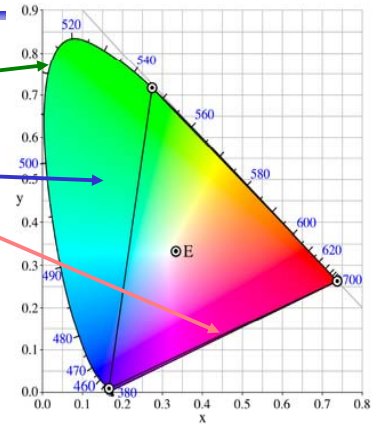
# Standardiserte trikromatiske koeffisienter (x,y,z-representasjon)

- X,Y,Z gir mengden av R, G og B
  - En farge spesifiseres med trikromatiske koeffisienter:
$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z}$$
$$z = \frac{Z}{X+Y+Z}$$
  - Ser at  $x+y+z=1$
- Den ene parameteren er ekvivalent med intensitet.
- De to andre gir fargen.
- Alle farger som har samme intensitet kan da gjengis i et 2-D kromatisitetsdiagram
- Merk: vi har isolert vekk intensitet for å få et 2D diagram



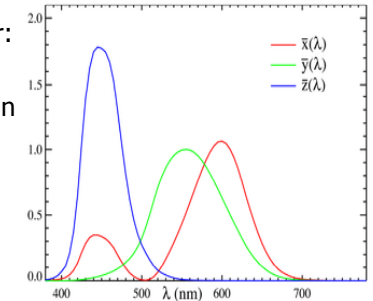
# CIE kromatisitetsdiagram

- Mettede farger langs "hestesko"
  - Mindre mettede inn mot midten.
  - Pastellfarger nede til høyre.
- Alle blandinger av N farger ligger innenfor N-kant med de N fargene som hjørner.
  - Alle mulige RGB-farger ligger innenfor markert trekant.



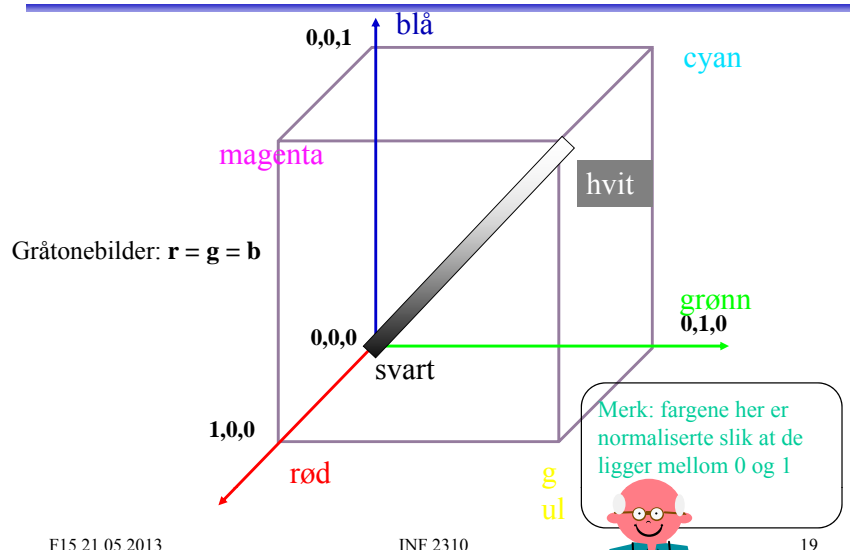
# Kameraets RGB detektorer

- Lysfølsomhet for RGB-detektorer:
- La spektralfordelingen til lyset inn i kamera være  $C(\lambda)$ .
- Tre tall bestemmer fargens posisjon i RGB-rommet:



$$c_i = \int C(\lambda) a_i(\lambda) d\lambda, \quad i=r, g, b$$

# RGB-kuben



# Eksempel RGB-bilde



Bånd 1: R



Bånd 2: G



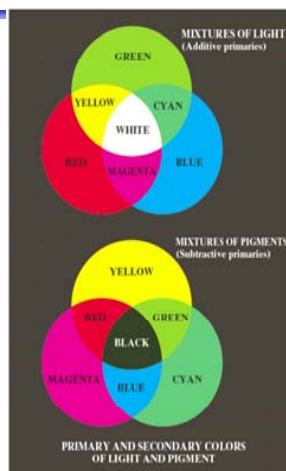
Bånd 3: B



RGB-bildet vist på skjerm

## Additive vs. subtraktive fargesystemer

- Lys:
  - Mikses additivt.
  - Primærfarger R, G, B.
  - Sekundærfarger: cyan, magenta, gul.
  - Øyet, kameraer, og monitorer/TV er additive.
- Maling/farge med pigment:
  - Kalles subtraktivt.
  - Primærfarger: yellow, cyan, magenta.
  - Primærfarger defineres her ved at de subtraherer en av lysets primærfarger og kun reflekterer de to andre.

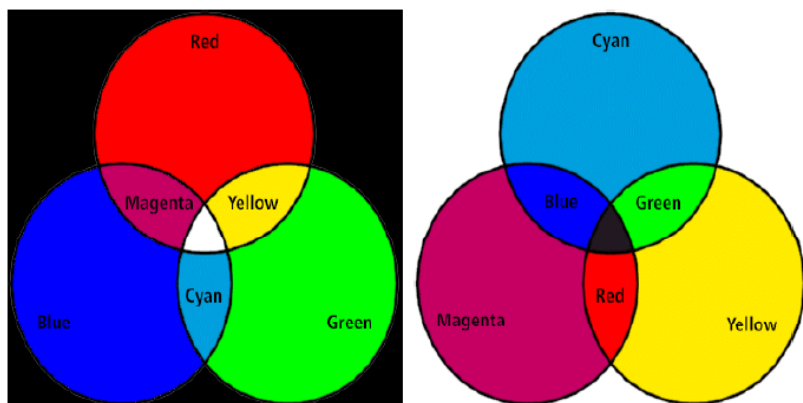


## CMYK-fargemodellen

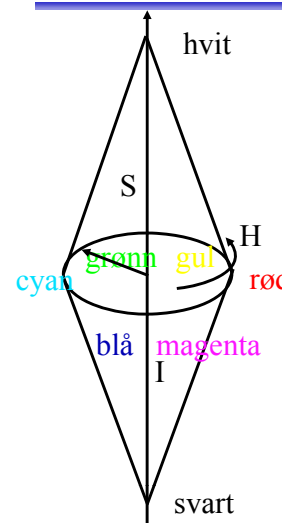
- CMYK- modellen er subtraktiv (start med hvitt, trekk fra farger).
- Alternativ til R,G,B; basisfarger er cyan, magenta, yellow (CMY-modeller).
  - $C = 1 - R$  eller  $255 - R$  hvis 8-bits ikke-normaliserte bilder
  - $M = 1 - G$   $255 - G$
  - $Y = 1 - B$   $255 - B$
- RGB er vanlig på display, men CMYK er vanlig på fargeprintere (K er ekstra komponent for svart).
  - Egen komponent for svart fordi full verdi av C, M og Y i praksis gir mørk brunt og ikke svart.
  - På ulike printere ser også de rene fargene ulike ut når de skrives ut, så fargebilder forvrenges ofte ved utskrift.

## RGB og CMY

- RGB og CMY er i prinsippet sekundærfarger for hverandre.



## Hue, Saturation, Intensity (HSI)



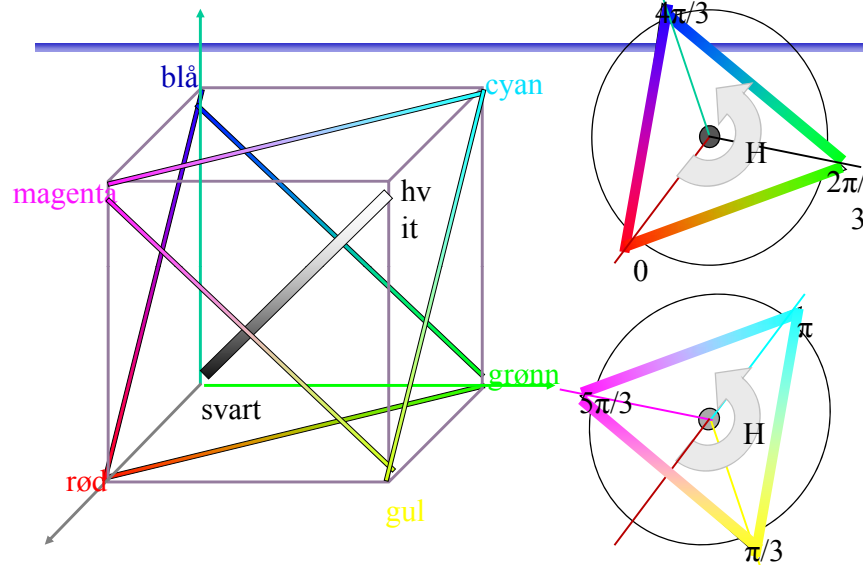
- Hue: ren farge - gir bølgelengden i det elektromagnetiske spektrum.



- H er vinkel og ligger mellom 0 og  $2\pi$ :  
 Rød:  $H=0$ , grønn:  $H=2\pi/3$ , blå:  $H=4\pi/3$ ,  
 gul:  $H=\pi/3$ , cyan:  $H=\pi$ , magenta:  $H=5\pi/3$
- Hvis vi skalerer H-verdiene til 8-bits verdier vil  
 Rød:  $H=0$ , grønn:  $H=85$ , blå:  $H=170$ ,  
 gul:  $H=42$ , cyan:  $H=127$ , magenta:  $H=213$ .



# RGB og IHS - primær og sekundær farger



# Mer om HSI

- Saturation: metning – hvor mye grått inneholder fargen
  - Hvis S=0, blir fargen grå uavhengig av hvilken verdi H har. (det vil si at vi ligger et sted på diagonalen i RGB-kuben)
- S ligger normalisert mellom 0 og 1, eller mellom 0 og 255 hvis 8-biters unsigned verdier pr. piksel.
- H og S tilsammen beskriver fargen og kalles kromatisitet
- I: intensitet, ligger mellom 0 og 1 eller 0 og 255.
- HSI-modellen egnet til å beskrive farge
- RGB-modellen egnet til å generere farger
- Konvertering fra HSI til RGB: formler finnes

# RGB og HSI

- La R,G,B-komponentene være normaliserte slik at de ligger mellom 0 og 1:

$$H = \begin{cases} \theta & B \leq G \\ 360 - \theta & B > G \end{cases} \quad \theta = \cos^{-1} \left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \right\} \quad S = 1 - \frac{3 \min(R,G,B)}{R+G+B} \quad I = \frac{R+G+B}{3}$$

Merk at H er udefinert når R=G=B, S er udefinert når I=0.

- Overgangen fra HSI til RGB kan enklest deles i tre tilfeller:
- Rød-grønn sektor:

$$0 < H \leq 120$$

$$R = I \left[ 1 + \frac{S \cos H}{\cos(60-H)} \right]$$

$$G = 3I - (R+B)$$

$$B = I(1-S)$$

- Grønn-blå sektor:

$$120 < H \leq 240$$

$$H = H - 120$$

$$R = I[1-S]$$

$$G = I \left[ 1 + \frac{S \cos H}{\cos(60-H)} \right]$$

$$B = 3I - (R+G)$$

- Blå-rød sektor:

$$240 < H \leq 360$$

$$H = H - 240$$

$$R = 3I - (G+B)$$

$$G = I[1-S]$$

$$B = I \left[ 1 + \frac{S \cos H}{\cos(60-H)} \right]$$

# HSI, HSV og HSL

- HSV og HSL er alternative sylindervektorrepresentasjoner til HSI.
- H er i praksis den samme i alle tre representasjonene.
- Intensity, Value og Lightness er forskjellige:

$$I = \frac{R+G+B}{3}, \quad V = M, \quad L = \frac{M+m}{2}; \quad M = \max(R,G,B), m = \min(R,G,B)$$

- Metningen (S) har ulike definisjoner i HSI, HSV og HSL, men vi har alltid S=0 når M=m=0.

- Ellers har vi de tre definisjonene for S:

$$S = 1 - \frac{m}{I}, \quad S = \frac{M-m}{1-|2L-1|} \quad \text{og} \quad S = \frac{M-m}{V}$$

- Merk også at HSI kan skrives som IHS, etc!

## Eksempler på RGB, CMYK, HSI

	RGB	CMYK	HSI
Rød	(255,0,0)	(0,255,255)	(0, 255, 85)
Gul	(255,255,0)	(0,0,255)	(42,255,170)
Grønn	(0,255,0)	(255,0,255)	(85,255,85)
Blå	(0,0,255)	(255,255,0)	(170,255,85)
Hvit	(255,255,255)	(0,0,0)	(0,0,255)
Grå	(192,192,192)	(63,63,63)	(0,0,192)
	(127,127,127)	(128,128,128)	(0,0,127)
Svart	(0,0,0)	(255,255,255)	(0,0,0)

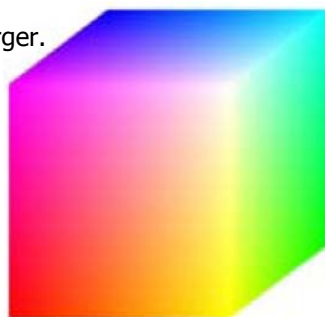


## Men bildet mitt ser ikke likt ut på to skjermer?

- RGB-farger på en skjerm avhenger av skjermens egenskaper, dvs. det samme bilde vist på to skjermer kan se ulikt ut.
- Samme bilde skrevet ut på to fargeprintere kan se HELT forskjellig ut, fargen avhenger av bl.a. skriveren, fargepatronene, papiret, etc.
- En skjerm kan vise flere farger en en CMYK-printer kan skrive ut (CMYK-skriver kan skrive noen farger en RGB-skjerm ikke kan vise).
- Vi sier at RGB og CMYK er utstys-avhengige fargerom.
- Det finnes internasjonale standarder for fargerom som er utstys-uavhengige. Et slikt system er CIEs XYZ-fargerom.
- Antall stabile, "gjenkjennbare farger" på en skjerm er ganske lite !

## Fargesyn

- Vi kan skille mellom ca. 100 rene farger (hue).
- Når fargene også varierer i intensitet, kan vi skille mellom ca. 6000 farger (hue+intensity).
- For hver av disse, kan vi skille mellom ca. 60 ulike metningsgrader (saturation).
- Vi kan altså skille totalt ca. 360 000 farger.
- Dette kan representeres med 19 biter. ( $2^{19} = 524\,288$ ).
- Lagrer R, G, B komponentene som byte-bilder.
  - totalt 24 biter per piksel.



## YIQ

- NTSC er standard for TV og video i USA. Bruker fargesystemet YIQ.
  - Y beskriver luminans, I og Q er krominanskomponentene.
  - samme signalet brukes både på farge- og gråtoneskjermer.
- Overgangen fra RGB til NTSC's YIQ :
  - Luminans-komponenten  $Y = 0.299*R + 0.587*G + 0.114*B$
  - Hue-komponenten  $I = 0.596*R - 0.274*G - 0.322*B$
  - Metnings-komponenten  $Q = 0.211*R - 0.523*B + 0.312*B$
- RGB svart (0,0,0) gir NTSC Y=0
- RGB hvit (1,1,1) gir NTSC Y=1.
- RGB grå (g,g,g) gir NTSC I=Q=0



## RGB og YIQ

- Transformasjonene kan uttrykkes ved matrisemultiplikasjon:

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

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.623 \\ 1 & -0.272 & -0.648 \\ 1 & -1.105 & 0.705 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

## YCbCr-modellen

- Dette er fargemodellen for digital TV og video!
  - Y er luminans (luma)
  - Cb er blå minus luma (B-Y)
  - Cr er rød minus luma (R-Y).
- YCbCr er digital, RGB kan være både analog og digital.
  - MPEG-kompresjon (i DVD'er, digital-TV og video CD'er) er kodet i YCbCr
  - digitale videokameraer (MiniDV, DV, Digital Betacam, osv.) gir et YCbCr signal over en digital link som FireWire eller SDI.
  - Den analoge "tvillingen" til YCbCr er YPbPr.

## YUV-modellen

- Brukes i analog TV (NTSC, PAL og SECAM).
  - Y representerer intensitet ("luma")
  - U og V er fargedifferansene B-Y og R-Y.
  - Et video-kamera konverterer RGB data som er registrert i fokalplanet til enten
    - "composite analog" (YUV)
    - analog YPbPr
    - digital YCbCr.
  - For framvisning på skjerm må alle disse tre fargerepresentasjonene konverteres tilbake til RGB.

## Fargebilder og fargetabeller

- RGB kan lagres med like mange biter for **r**, **g**, **b**, f.eks (8 + 8 + 8)
- Selv 3 + 3 + 3 = 9 biter gir oss 8 · 8 · 8 = 512 kombinasjoner, men bare 8 forskjellige nivåer av rødt, grønt og blått, og dermed også bare 8 forskjellige gråtoner.
- Et scene med mange nyanser av én farge vil da se ille ut !  
Hvorfor? Jo fordi denne fargen bare får 8 forskjellige nyanser !
- Det er ikke sikkert at alle de 512 fargene finnes i bildet.
- Alternativt kan man bruke 8 biter og [fargetabeller](#).
- Hver rad i tabellen beskriver en **r**, **g**, **b**-farge med 24 biter.
- **Tabellen inneholder de 256 fargene som best beskriver bildet.**
- I bilde-filen ligger pikselverdiene som tall mellom 0 og 255.
- Når vi skal vise bildet, slår vi bare opp i samme rad som pikselverdien, og finner [de tilsvarende r, g, b-verdiene](#).

## Fargetabell / oppslagstabell (LUT)

- Gråtone/fargeavbildningen utføres som tabell-oppslag
- LUT - Look Up Table
- Innholdet i bildefilen endres ikke, LUT-operasjonen utføres på datastrømmen mellom hukommelsen (databufferet) og skjermen

$$v_{out} = LUT(v_{in})$$

- Hvis vi ønsker endring i bildet:
  - Oppdatér bare G verdier i LUT (ikke n · m verdier i bildet)
- Q: Kan vi lage et negativt fra et positiv på denne måten ?

## Fargetabell

Pikselverdi	RGB-verdi
0	0,0,0
1	255,0,0
2	255,255,0
3	0,255,0
.	255,100,0
.	.
.	.
254	0,100,255
255	255,255,255

Disse verdiene ligger lagret på bildefilen

Disse verdiene vises på skjermen

- Kan vise 24 biters RGB-verdier på 8 biters skjerm
- Eller vise pseudofarger fra et gråtonebilde
- Pikselverdiene fra 0 til 255 tilordnes et RGB-triplet
- Ved framvisning leses pikselverdien
- Pikselverdien viser til et linjenummer i tabellen som inneholder RGB-fargene.

## "Median-cut" algoritmen

- En tilpasning til de farger som finnes i bildet:
  1. Finn den boksen i RGB-rommet som omslutter alle fargene i bildet.
  2. Sortér fargene i boksen langs den lengste RGB dimensjonen til boksen.
    - Dette gjøres enklest ved hjelp av et histogram.
  3. Del boksen i to ved medianen til den sorterte listen.
    - Dermed blir boksen delt i to nye bokser
    - omtrent like mange piksler tilhørende hver nye boks.
  4. Gjenta steg 2 og 3 for alle boksene som nettopp ble dannet.
    - Stopp når du har 256 bokser.
  5. For hver boks, la midtpunktets RGB-verdier representere boksen og lag en 256-linjers LUT som inneholder disse midtpunktene.
  6. Erstatt hver 3 · 8 biters pikselverdi med en 8 biters indeks som svarer til det boks-midtpunktet som ligger nærmest 3 · 8 biters pikselverdien i RGB-rommet.

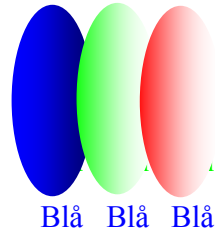
## Overganger mellom små og store "endianere"

- To piksler med 3·8 bit RGB "big endians": (R1G1B1)(R2G2B2) ex: (0,255,0),(0,255,0) avlest som "little endians" blir til (G1R1R2)(B1B2G2) (255,0,0),(0,0,255)
- La en LUT inneholde 256 farger
  - LUT'en – som inneholder 256 · 3 byte (RGB) vil bli utsatt for effekten ovenfor.
  - Samtidig vil to og to piksler i bildefilen bytte plass
- La en LUT inneholde  $2^{16} - 1 = 65535$  linjer (farger) a 16 biter.
  - Nå blir ikke lenger to og to piksler i bildefilen byttet om.
  - Men pikselverdien vil peke til feil sted i fargetabellen.
- Anta 16 biter = 2 byte RGB: 5 + 6 + 5 biter
  - 50% grå svarer til (16, 32, 16) i en (5 + 6 + 5) biters LUT, med bitmønster 1000010000010000
  - Bytter vi om på bytene får vi 0001000010000100 (2, 4, 4) dvs (0.0625, 0.0625, 0.125) på en skala fra 0 til 1. 50% grått er blitt til en ganske dunkel blåfarge.



# Alfa-kanal

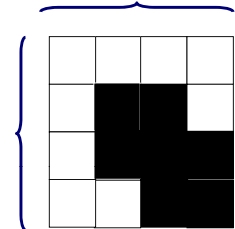
- $\alpha$  i (RGBA) eller (aRGB) spesifiserer om fargene (RGB) i bildet er helt eller delvis transparente.
- Verdier av  $\alpha$  fra 0 (helt transparent) til 255 (helt ugjennomsiktig).
- Hensikten med en "alfa-kanal" er at man kan la en bakgrunn synes gjennom et bilde.
- Bakgrunnen kan bestå av forskjellige grafiske elementer, eller av et annet bilde.
- Teknikken kalles "alpha blending", og kan både brukes til
  - å vise tekst og grafikk sammen med et bilde
  - "blending" av to bilder, to bildesekvenser, eller stillestående bakgrunn med en video-sekvens.
  - Finnes i Adobe Photoshop, Paint Shop Pro, GIMP ....
- Hvis vi legger et bilde oppå en bakgrunn, blir resultatet
 
$$(\text{bildefargen} \cdot \alpha + (\text{bakgrunnsfargen} \cdot (255 - \alpha)) / 255.$$
  - Resultat lik bakgrunn for  $\alpha = 0$
  - Resultat midt mellom for og bakgrunn for  $\alpha = 127$
  - Resultat lik forgrunn for  $\alpha = 255$ .



# Utskrift av gråtonebilder

- Problem: printere er binære, skriver svart eller ingenting
- Løsning: printeren jobber på et finere grid (bruker halvtoner)
- Virker fordi: øyet gjør en glatting av intensitetsverdier, slik at et gjennomsnitt vises
- Utfordring: hvordan lage mønstre av binære piksler som utgjør en gråtone
  - "Patterning" bruker  $n^2+1$  verdier fra  $n \times n$  rutenett
  - Ordnet "Dithering" terskler med en matrise
  - "feil-diffusjon" fordeler feilene ved terskling

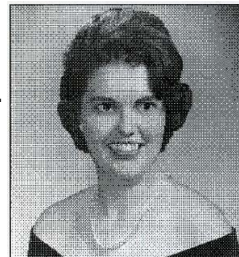
Et piksel



# "Dithering"

- Terskler gråtonebildet mot en "dither-matrise"
- Dither-matrisen  $D_n$ 
  - inneholder  $2^n \cdot 2^n$  elementer
  - deler gråtoneskalaen fra 0 til 255 inn i  $(2^n)^2$  ekvidistante trinn.
- Forstørr opp bildet med en faktor  $2^n$ .
- Matrisen legges som en maske over bildet
- Elementene i matrisen fungerer som terskler.
- Hvis pikselverdien > terskelen => hvit, ellers svart.
- Gir et tilsynelatende gråtonebilde som
  - Består av svarte og hvite punkter
  - Har samme størrelse som original-bildet
  - Har systematiske mønstre for hver gråtone.

$$D_2 = \begin{bmatrix} 0 & 128 & 32 & 160 \\ 192 & 64 & 224 & 96 \\ 48 & 176 & 16 & 144 \\ 240 & 112 & 208 & 80 \end{bmatrix}$$



# Feil-diffusjon

- Retter opp systematiske feil som innføres ved dither-terksling.
- En terskel = 128 vil avbilde en gråtoneverdi som 0 (svart) eller 255 (hvit)
  - OK hvis pikselverdi nær 0 eller 255
  - hvis pikselverdi nær terskelverdien blir feilen stor.
- Diffusjon sprer feilen over flere nabopiksler

$$\begin{bmatrix} .. & .. & .. \\ .. & P & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}$$

- Dette forbedrer det visuelle resultatet
  - Begrensninger:
    - Kan ikke spre feilen utenfor bildets grenser
    - Gråtoner kan ikke ende under 0 eller over 255.



## Utskrift av fargebilder

- CMYK-modell brukes
- Halvtonemønstre i bestemte vinkler (ulik for hver farge) må brukes til å lage fargemønstre
- Prinsipp: øyet kombinerer de fire fargene slik at ingen brå fargeoverganger ses
  - Hver farge skrives ut i et spesielt symmetrisk mønster



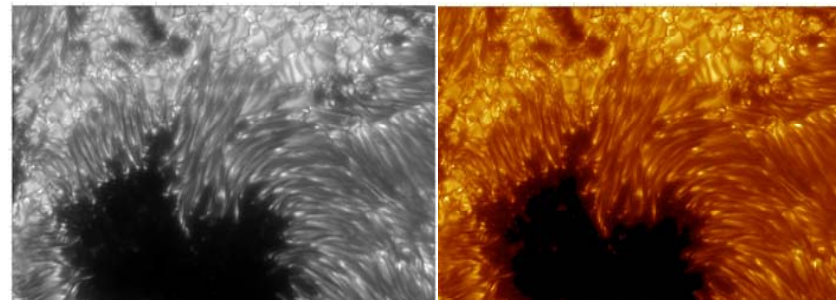
F15 21.05.2013

INF 2310

45

## Pseudo-farger

- *Pseudo-fargebilder* kan være gråtonebilder der man har tilordnet hver gråtone en RGB-farge ved hjelp av en oppslagstabell (LUT).
- Brukes i medisinsk avbildning for å framheve små gråtoneforskjeller.
- Brukes også i grafisk framstilling av data.
- Hvis farge-LUT'en gjengis i gråtoner, bør intensiteten bli riktig !



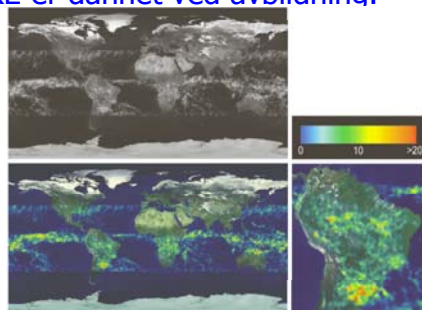
F15 21.05.2013

INF 2310

46

## Farge-grafikk - I

- Vi kan produsere raster-data basert på observasjoner, simuleringer, beregninger, etc.
- Et eksempel kan være nedbør-data i en kartprojeksjon.
- Bruk av en LUT gir da en grafisk framstilling **som IKKE er dannet ved avbildning.**



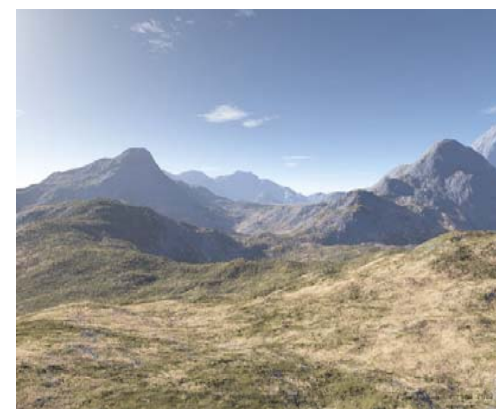
F15 21.05.2013

INF 2310

47

## Farge-grafikk - II

- Vi kan produsere vektor- eller rasterbilder vha fraktaler.
- Dette er heller ikke et resultat av avbildning.



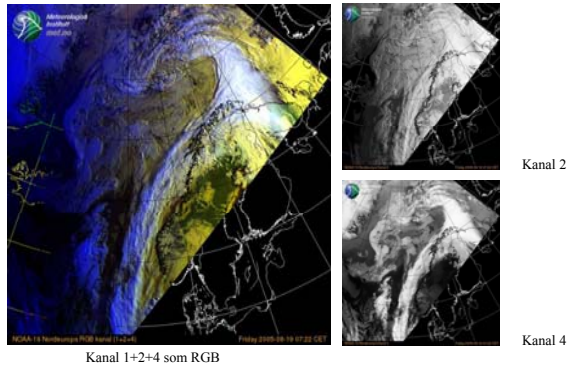
F15 21.05.2013

INF 2310

48

# Falske farger

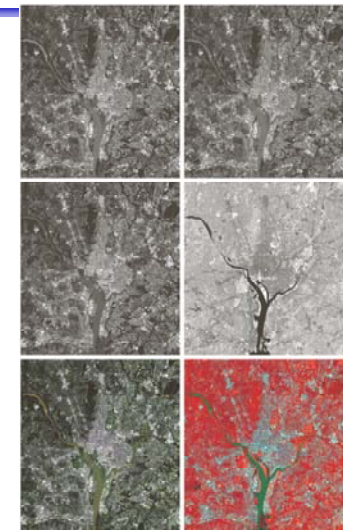
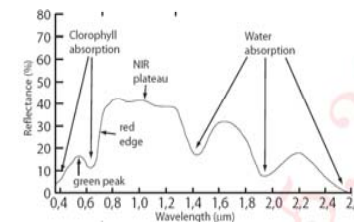
- NOAA AVHRR
- kanal 1:  
580-680 nm
- kanal 2:  
725-1000 nm
- kanal 4:  
1030 – 1130 nm



- vist som RGB-bilde (Meteorologisk Institutt)
  - kanalene er *ikke* RGB (700, 546.1, 435.8).
  - Altså *falske farger*.

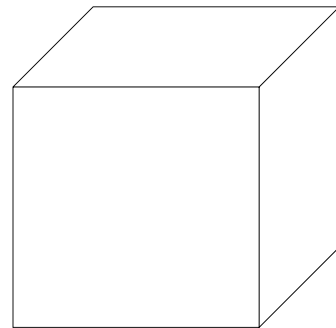
# Falske farger - II

- Gitt fire multispektrale kanaler.
- Kombinasjon av (0.45-0.52),(0.52-0.60),(0.63-0.69) gir et naturtro RGB-bilde.
- RGB=(0.45-0.52),(0.52-0.60),(0.76-0.90):  
**Biomasse synes som rødt**
- Dette skyldes "red edge" i reflektansen for klorofyll

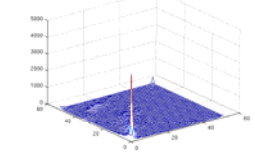
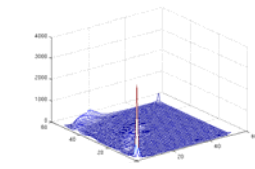
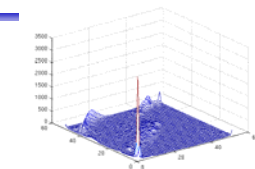
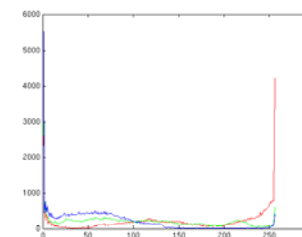


# Histogrammer av fargebilder - I

- Et bilde med tre bånd har egentlig en 3-dimensjonal kube som histogram
- Med 3 ganger 8 bit RGB får denne  $256 \times 256 \times 256 = 16\,777\,216$  "bins"
- Et bilde på  $1024 \times 1024$  piksler fyller maksimalt 1/16 av disse bins, dvs. 3D-kuben er for det meste tom.
- Man jobber vanligvis ikke på 3D-histogrammet, men på projeksjoner ned til 1D eller 2D
  - Projeksjon ned på R-, G- eller B-aksen
    - Et 1D-histogram for hver av fargene
  - Projeksjon på RG-, RB-, eller GB-planet
    - Et 2D-histogram for hvert farge-par.



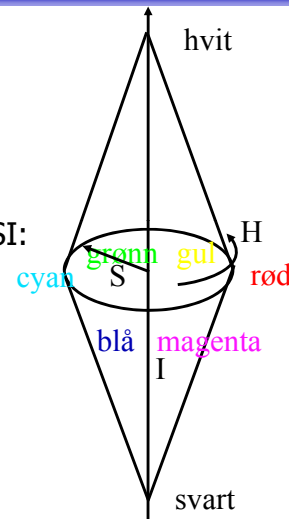
# Histogrammer fra fargebilder - II





## Histogramutjevning av RGB-bilder

- Histogramutjevning på hver komponent (R,G,B) uavhengig av hverandre
  - Ofte dårlig resultat
- Et bedre alternativ er å benytte HSI:
- Transformér bildet fra RGB til HSI
- Gjør histogramutjevning på I-komponenten
- Transformer HSI<sub>ny</sub> tilbake til RGB



## Eks: Histogramutjevning RGB vs HSI



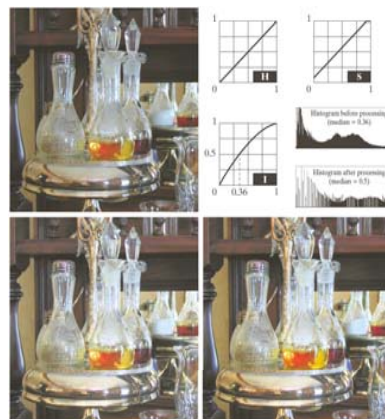
Originalbilde

Histogramutjevning på RGB

Histogramutjevning i intensitet i HSI

## Histogramutjevning i HSI

- Transformer fra RGB til HSI.
- Bruk kumulativt I-histogram histogramutjevning.
- Transformer tilbake til RGB.
- H og S er uforandret, men siden I er endret, kan farge-persepsjonen påvirkes.
- Juster eventuelt metningen S før transformen fra HSI til RGB.



## Lavpass-filtrering av fargebilder



- Et fargebilde kan representeres som R,G,B-komponenter (venstre) eller som HSI (nedenfor)

- RGB-filtrering gir blurring, endring av farge-kantene.
- Filtrering av I gir et mykere bilde uten endring i fargene.



# Laplace-filtrering av fargebilder

- Vi kan gjøre et gråtonebilde skarpere ved å addere skalert Laplace-bilde.  $g(x, y) = f(x, y) + c \left[ \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \right]$
- Vi kan addere Laplace til hver RGB-komponent.
  - Fargen i hvert piksel påvirkes av fargen til alle pikslene innenfor filteret
- Eller vi kan transformere til HSI, addere Laplace til I, konvertere tilbake.
  - Fargen er bevart, men intensiteten endres nær kanter og linjer i bildet.



F15 21.05.2013

INF 2310

57

# Terskling av fargebilder - I

- Anta at vi har observert samme scene på flere bølglengder.
- Vi kan da utføre terskling basert på
  - to-dimensjonale
  - tre-dimensjonale
  - eller multi-dimensjonale histogrammer
- Enkel metode:
  - 1: Bestem terskler uavhengig for hver kanal.
  - 2: Kombiner alle segmenterte kanaler til ett bilde.
- Dette svarer til at vi har delt opp f.eks. RGB-rommet i bokser.

F15 21.05.2013

INF 2310

58

# Terskling av fargebilder - II

- En mer kompleks metode:
- Velg et punkt i det multidimensjonale rommet som referanse, f.eks.  $(R_0, G_0, B_0)$
- Terskle basert på avstand fra dette referansepunktet.

$$d(x, y) = \sqrt{[f_R(x, y) - R_0]^2 + [f_G(x, y) - G_0]^2 + [f_B(x, y) - B_0]^2}$$

- Slik at  $g(x, y) = \begin{cases} 1 & \text{hvis } d(x, y) \leq d_{\max} \\ 0 & \text{hvis } d(x, y) > d_{\max} \end{cases}$
- Dette definerer en kule med radius  $d_{\max}$  omkring punktet  $(R_0, G_0, B_0)$ .
- Kan lett generaliseres til ellipsoide med forskjellige avstands-terskler i R, G, B

$$d(x, y) = \sqrt{\frac{[f_R(x, y) - R_0]^2}{d_R^2} + \frac{[f_G(x, y) - G_0]^2}{d_G^2} + \frac{[f_B(x, y) - B_0]^2}{d_B^2}}$$

- Merk at da er  $g(x, y) = \begin{cases} 1 & \text{hvis } d(x, y) \leq 1 \\ 0 & \text{hvis } d(x, y) > 1 \end{cases}$

F15 21.05.2013

INF 2310

59

# Terskling i HSI

- Transformer fra RGB til HSI.
- Anta at vi vil segmentere ut delene av bildet som
  - Har en gitt farge (H)
  - Er over en gitt metnings-terskel (S)
- Lag en maske ved å terskle S-bildet (velg en percentil)
- Multipliser H-bildet med masken.
- Velg et intervall i H som svarer til ønsket farge.
- Husk at H er sirkulær!

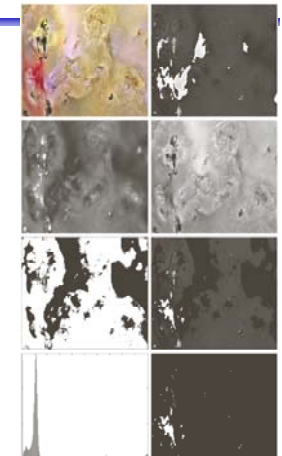


FIGURE 6.42 Image segmentation in HSI space. (a) Original. (b) Hue. (c) Saturation. (d) Intensity. (e) Binary saturation mask (black = 0). (f) Product of (b) and (e). (g) Histogram of (f). (h) Segmentation of red components in (a).

F15 21.05.2013

INF 2310

60

# Kant-deteksjon i fargebilder

- Gråtone gradient-estimatorene er ikke definert for vektorer.
- Vi kan finne gradient-magnitudo per RGB-komponent, summere og skalere.
- Vi kan finne gradient-magnitudo og retning vha prikk-produktene av x- og y-komponentene av gradienten i RGB-rommet:

$$F(x, y) = \sqrt{\frac{1}{2}[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})\cos 2\theta(x, y) + 2g_{xy}\sin 2\theta(x, y)]}$$

$$\theta(x, y) = \frac{1}{2} \tan^{-1} \left[ \frac{2g_{xy}}{g_{xx} - g_{yy}} \right]$$

$$g_{xx} = \left| \frac{\partial R}{\partial x} \right| + \left| \frac{\partial G}{\partial x} \right| + \left| \frac{\partial B}{\partial x} \right|^2$$

$$g_{yy} = \left| \frac{\partial R}{\partial y} \right| + \left| \frac{\partial G}{\partial y} \right| + \left| \frac{\partial B}{\partial y} \right|^2$$

$$g_{xy} = \frac{\partial R}{\partial x} \frac{\partial R}{\partial y} + \frac{\partial G}{\partial x} \frac{\partial G}{\partial y} + \frac{\partial B}{\partial x} \frac{\partial B}{\partial y}$$

- Differansen er litt komplisert å analysere!



# Støy i fargebilder



- Legger additiv Gaussisk støy til hver RGB-komponent ( $\mu=0, \sigma^2=800$ ).
- Støyen er mindre synlig i fargebildet.
- Konverterer det støyfylte bildet til HSI.

- H og S er veldig støyfylt
- I er mindre støyfylt enn RGB-kanalene



# Kompresjon av fargebilder

- Vi har sett på klassisk JPEG-kompresjon av fargebilder.

**JPEG-kompresjon av fargebilde**

- Vi skifter fargeom slik at vi separerer lysintensitet fra kromasi (perseptuell redundans, sparer plass).
- Bildet deles opp i blokker på 8x8 piksler, og hver blokk kodes separat.

- Hver blokk transformeres med DCT.
- Forskjellige vektmatriser for intensitet og kromatisitet.
  - ..... resten er som for gråtonebilde ...

F15 21.05.2013 INF 2310 63



- JPEG 2000 oppnår høyere kompresjon og bedre kvalitet.
- Eksempel: Kompresjonsrate = 230.