Color transformation

process the components of a color image within the context of a single color model

- Color can be described (RGB system), or by linear transformation as XYZ, CMY.
- image by some type of cylindrical-like coordinate system, it means by its hue, saturation and some value representing brightness. If the RGB coordinates are in the interval from 0 to 1, each color can be represented by the point in the cube in the RGB space.
- we convert an image in some image processing application into some brightness-hue-saturation model and we would like to work with individual components as with separate images. There is desirable regarding to the back conversion to have all combinations of the values. It means we need such model, where the range of values of saturation is identical for all hues.

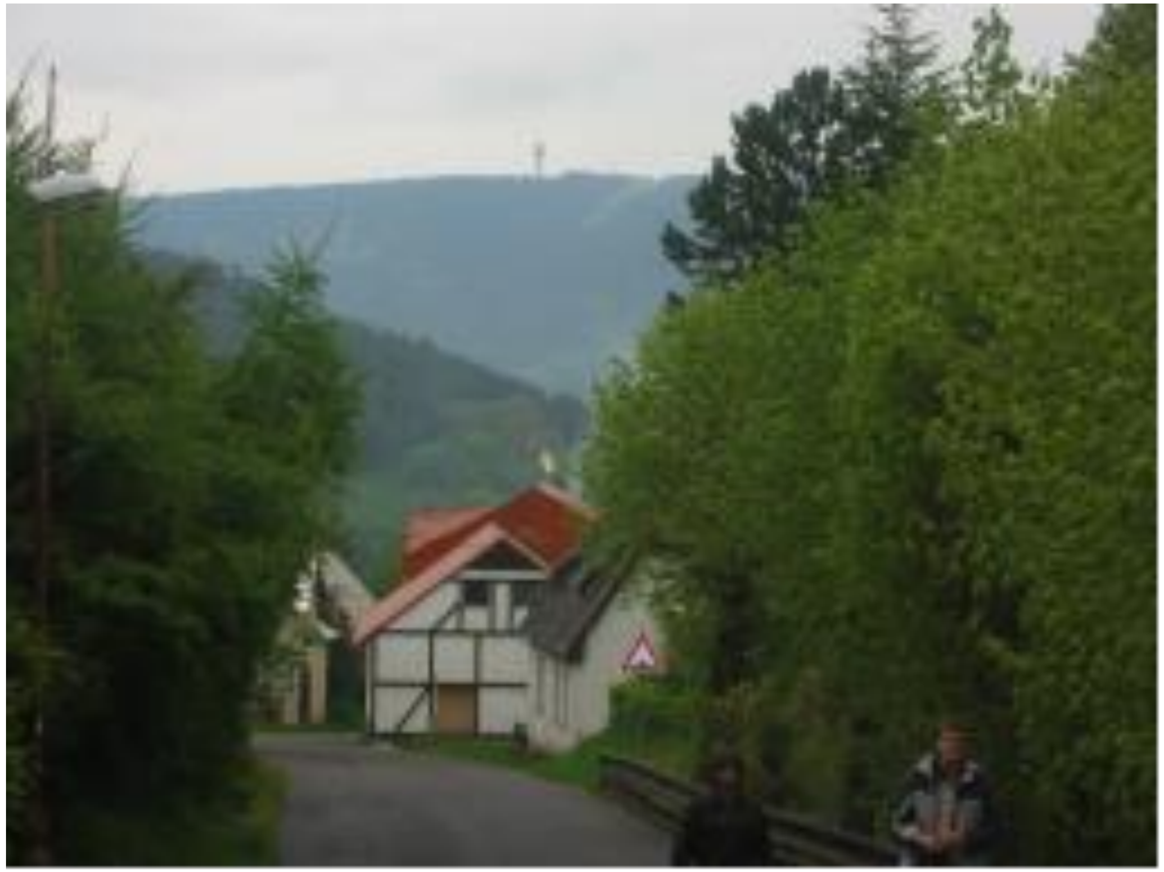
- From this point of view, the GLHS color is probably the best from the current ones, particularly for wmin = wmid = wmax = 1/3. The good model should satisfy some demands as:

  1-The brightness should be a linear combination of all three RGB components. At least, it must be continuous growing function of all of them

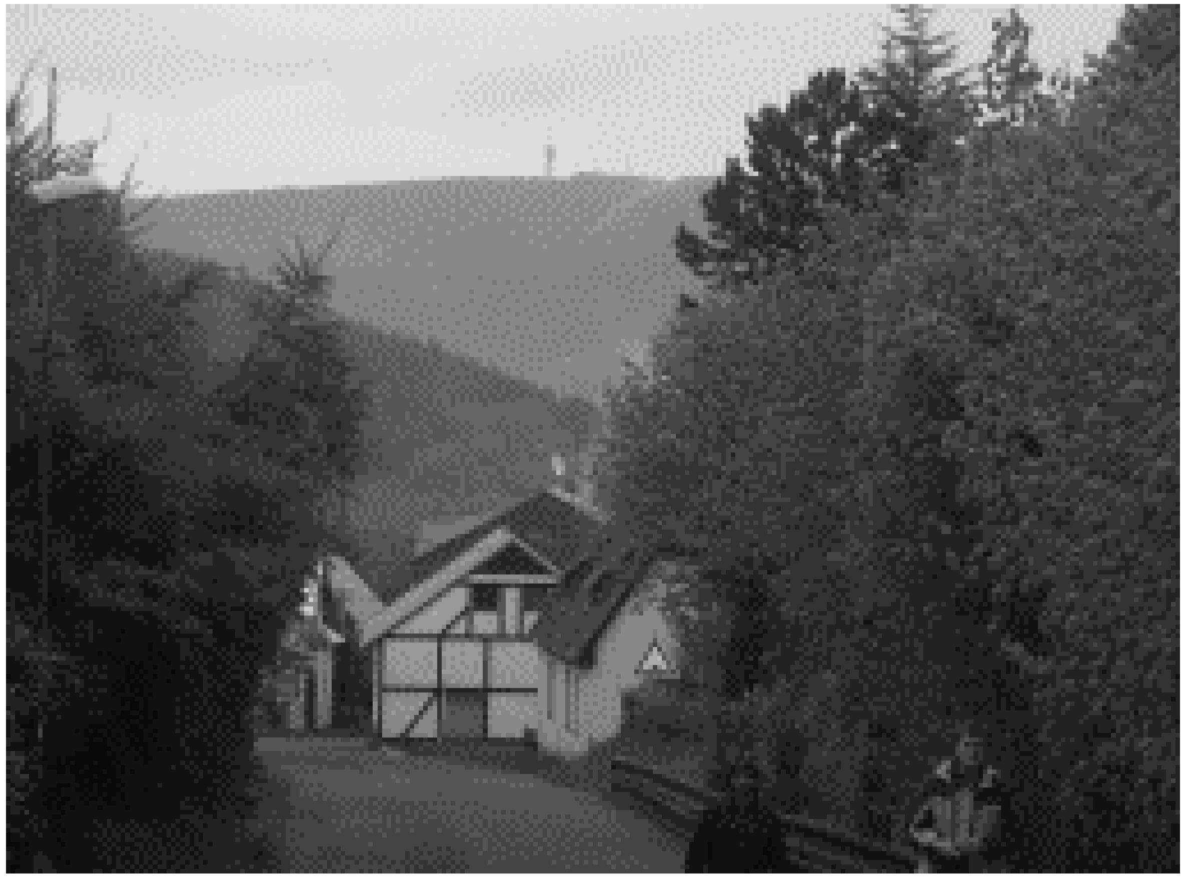
  2-The hue differences between the basic colors (R,G,B) should be 120◦ and similarly between the complement colors (yellow, purple and cyan). The hue difference between a basic color and an adjacent complement one (R ,Y) should be 60

- 3-The saturation should be 1 for the colors on the surface of the RGB color cube, it means in case of one of the RGB components is 0 or 1 except black and white vertices and it is 0 in case of R=G=B

Original Image

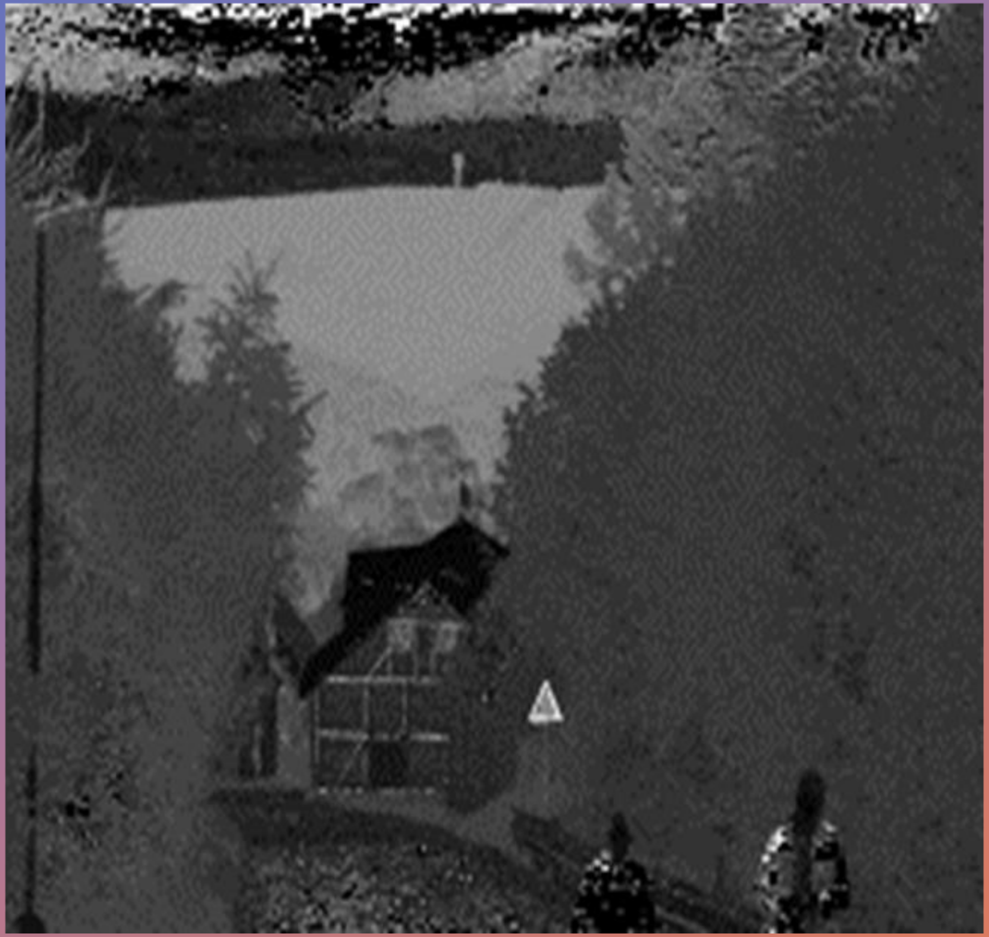


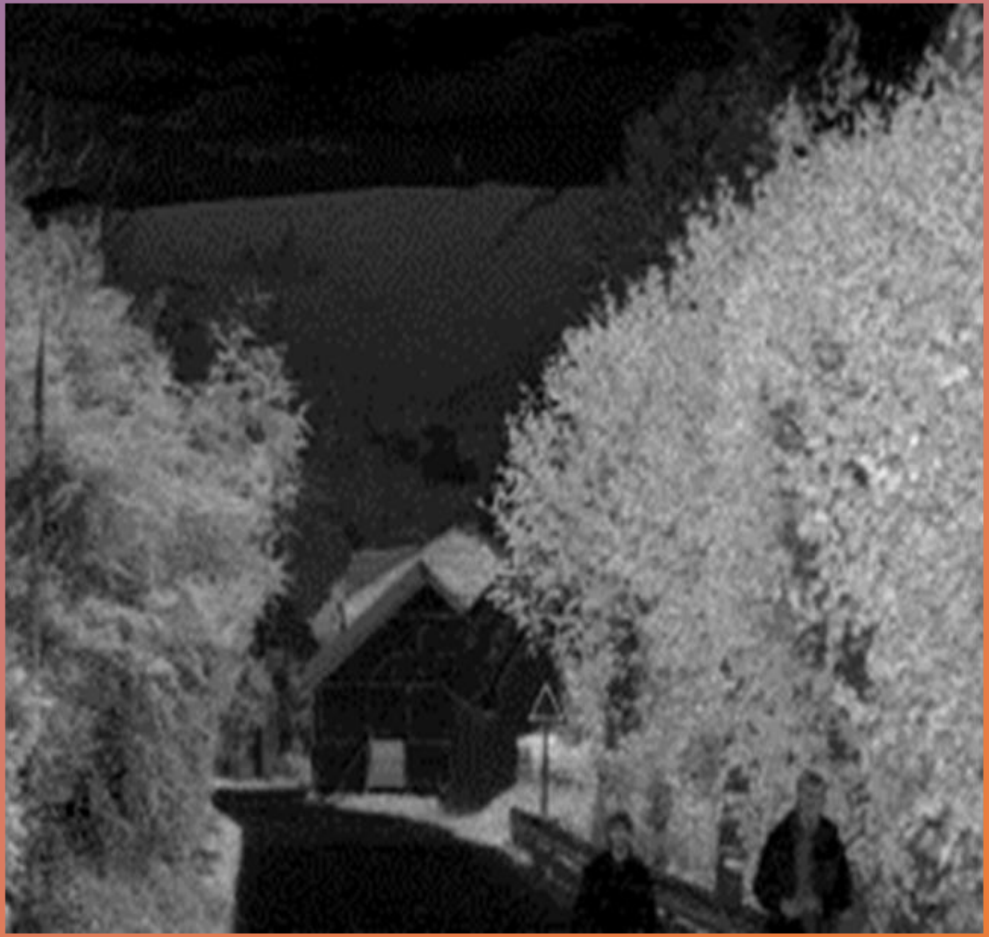
Brightness Y



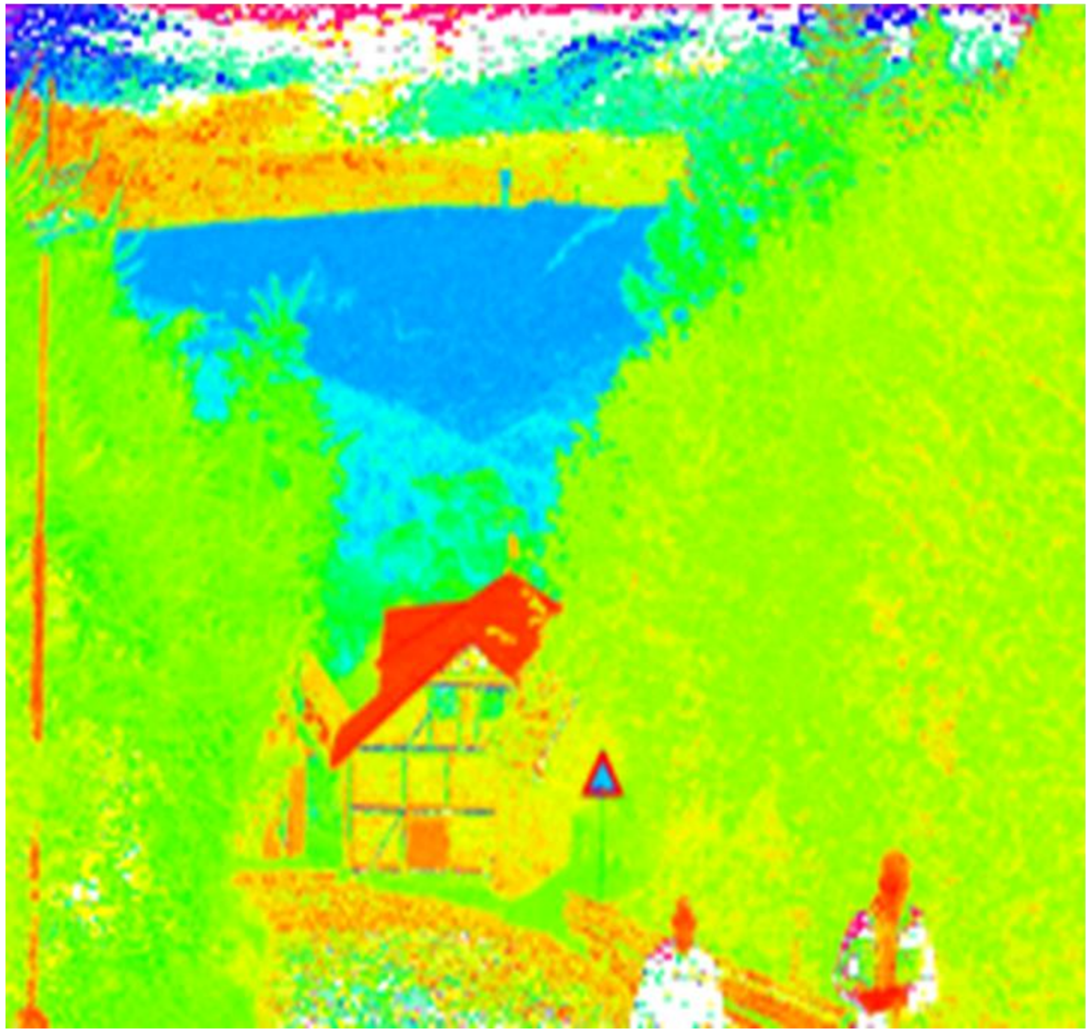
Hue H

Saturation S

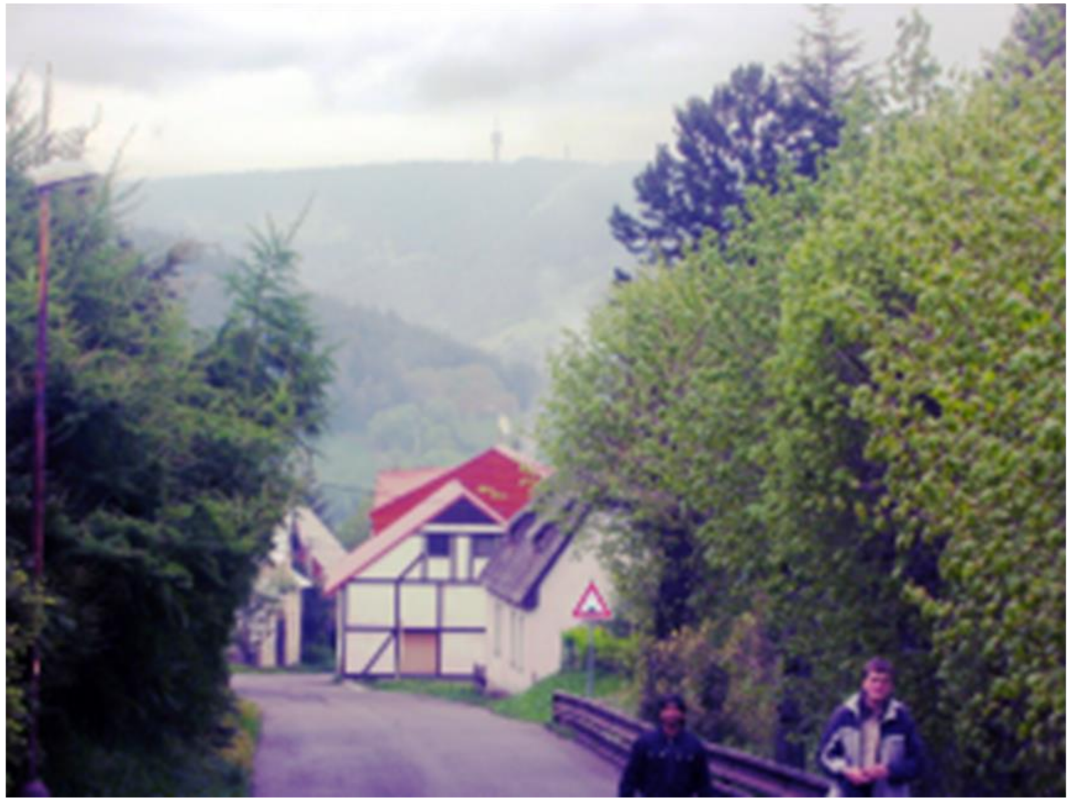




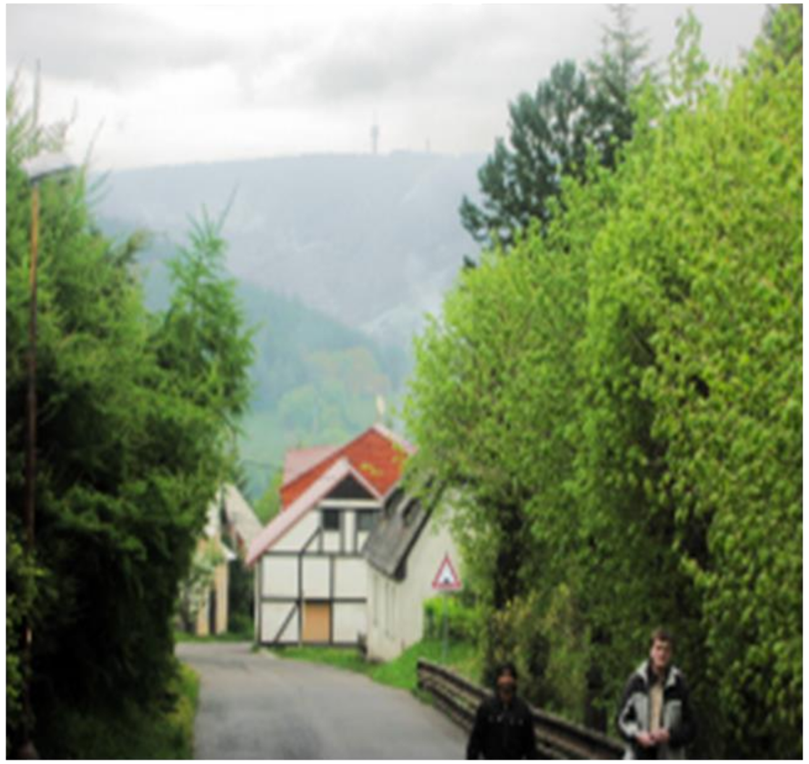
Hue with maximum saturation

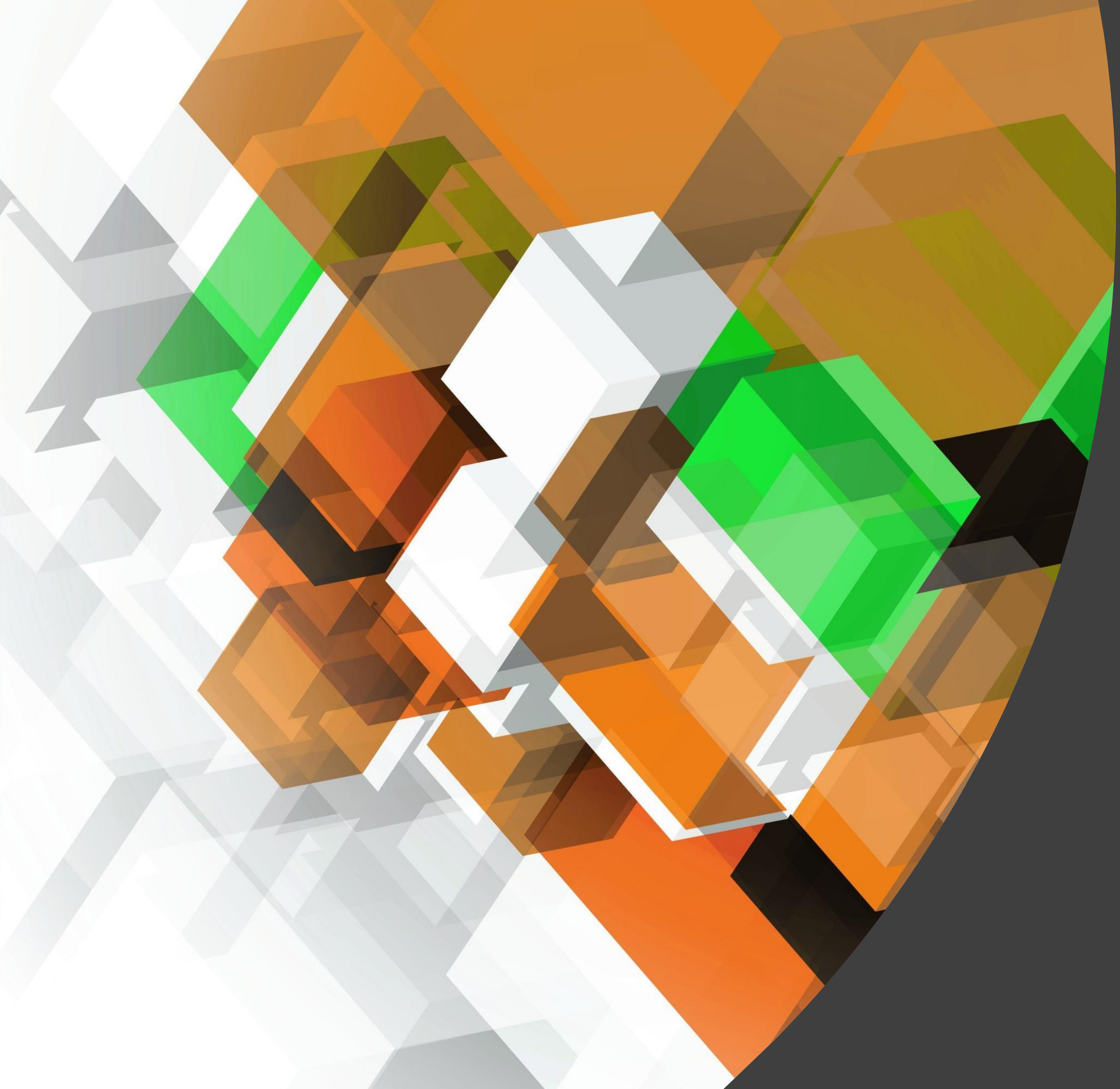


Optimized histogram in RGB

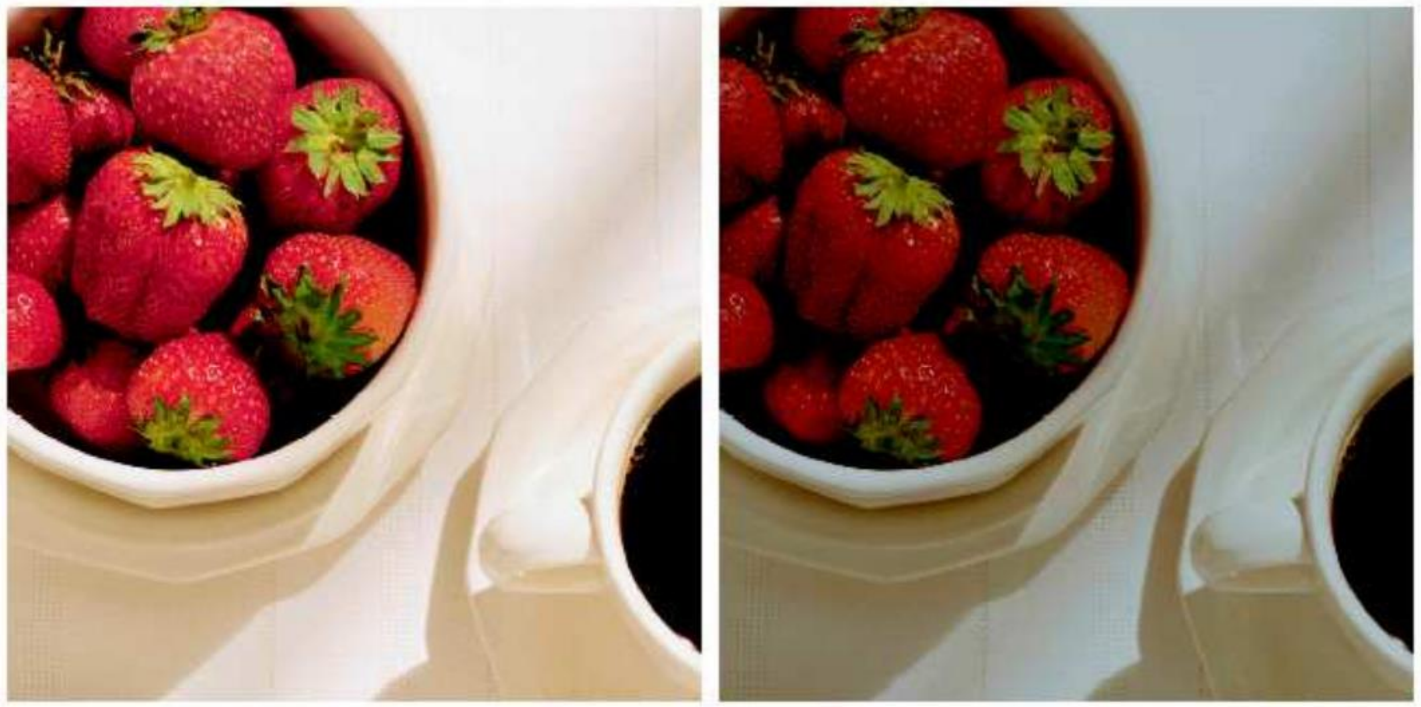


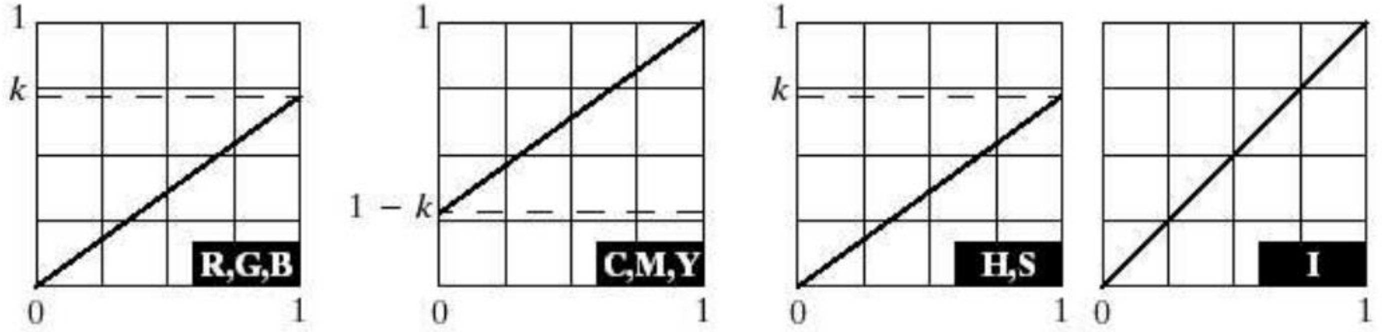
Optimized histogram in YHS





In our opinion, the best brightness, hue and saturation system consists of the brightness as linear combination of the RGB values, the hue as actual angle in the color cube and saturation as relative distance from the body diagonal to the surface of the color cube. Such a system, called YHS, is presented in [1]. It satisfies all three demands and makes easier some color manipulations

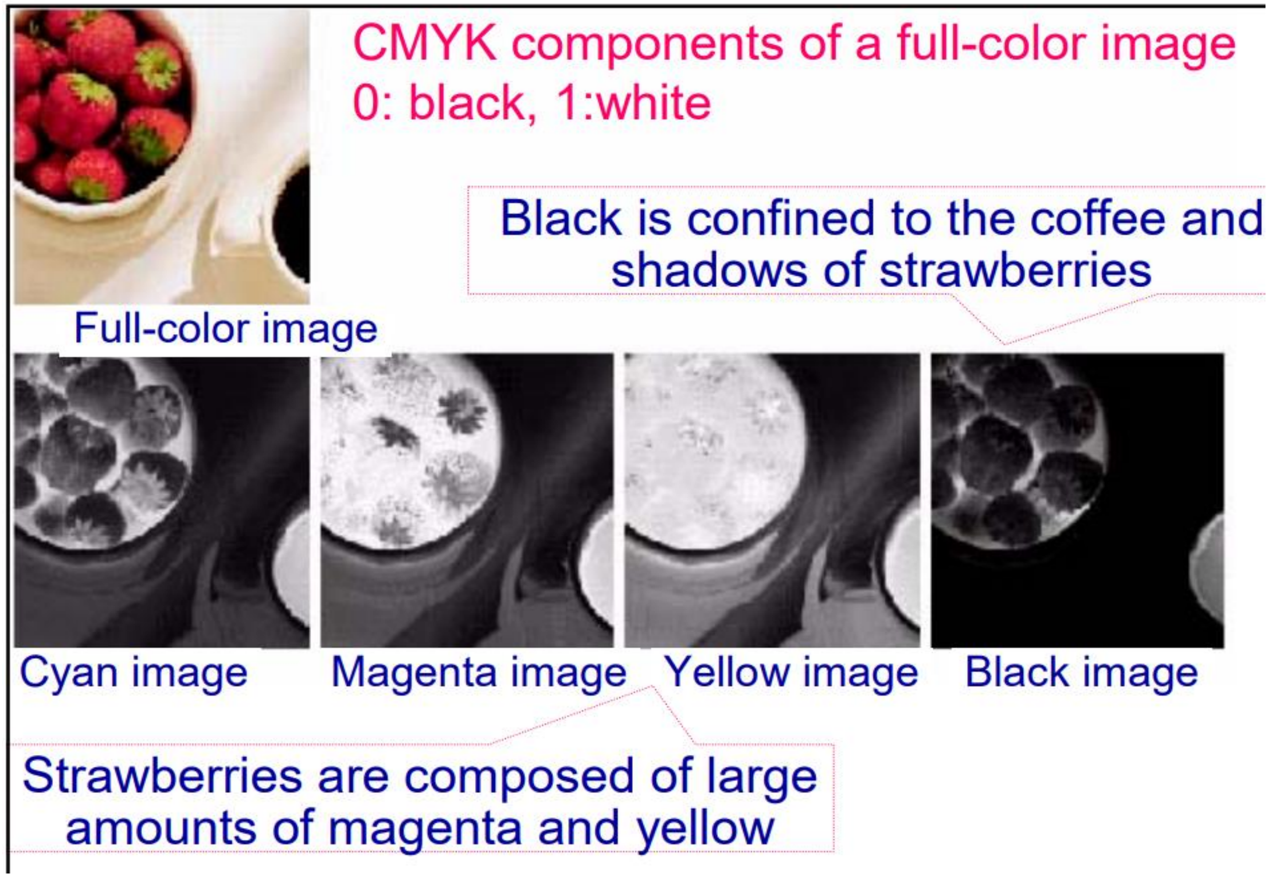




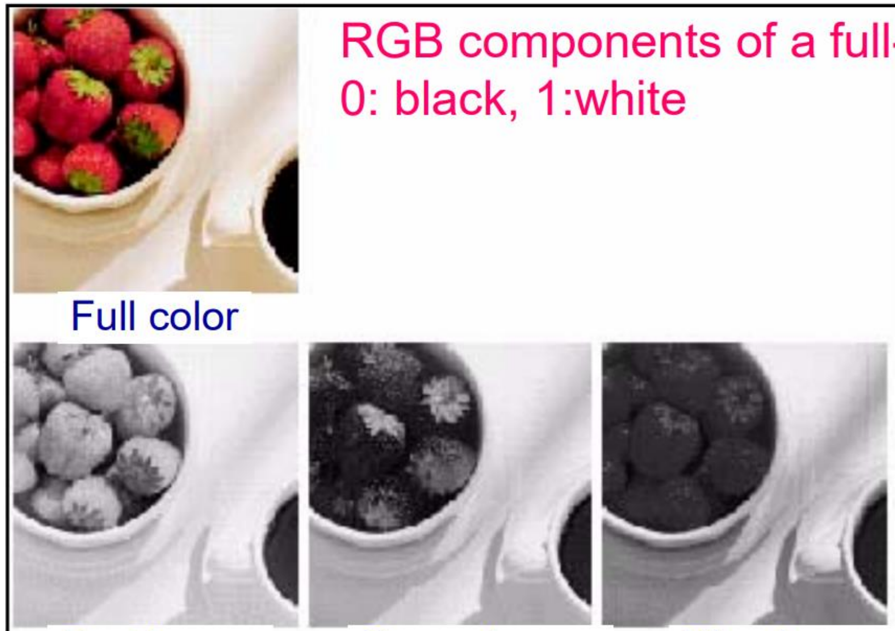
a b
c d e

FIGURE 6.31 Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$). (c)–(e) The required RGB, CMY, and HSI transformation functions. (Original image courtesy of MedData Interactive.)

- Example:
- Pixel values are triplets (RGB, CMY, HSI) or quartets (CMYK)

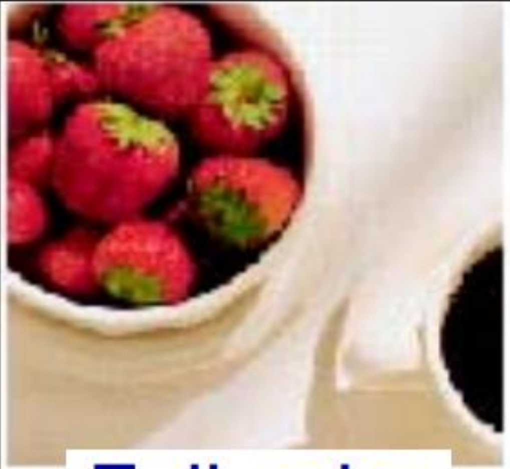

CMYK components of a full-color image
0: black, 1:white
Black is confined to the coffee and shadows of strawberries
Full-color image
Cyan image
Magenta image
Yellow image
Black image
Strawberries are composed of large amounts of magenta and yellow

RGB components of a full-color image
0: black, 1:white

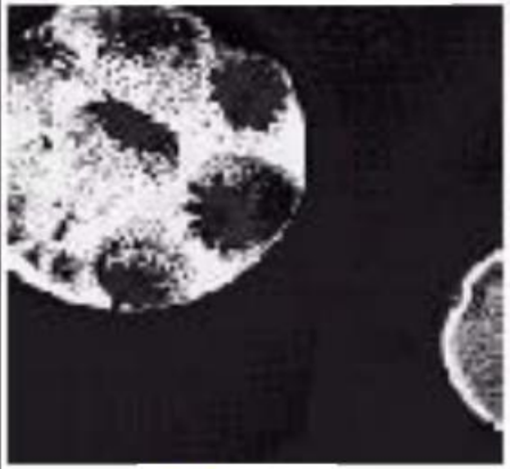

Full color
Red image
Green image
Blue image

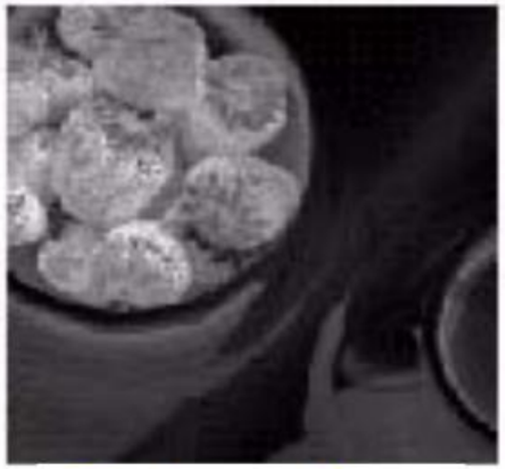
Strawberries contain a large amount of red and very little green and blue

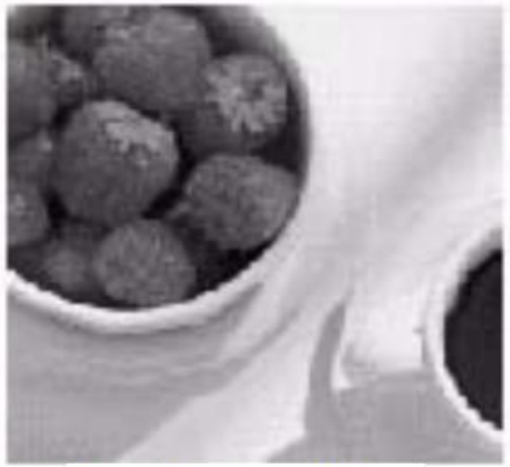
HSI components of a full-color image


Full color

Strawberries are relatively pure in color and possess the highest saturation


Hue


Saturation


Intensity

Intensity component is a monochrome rendition of the full-color image

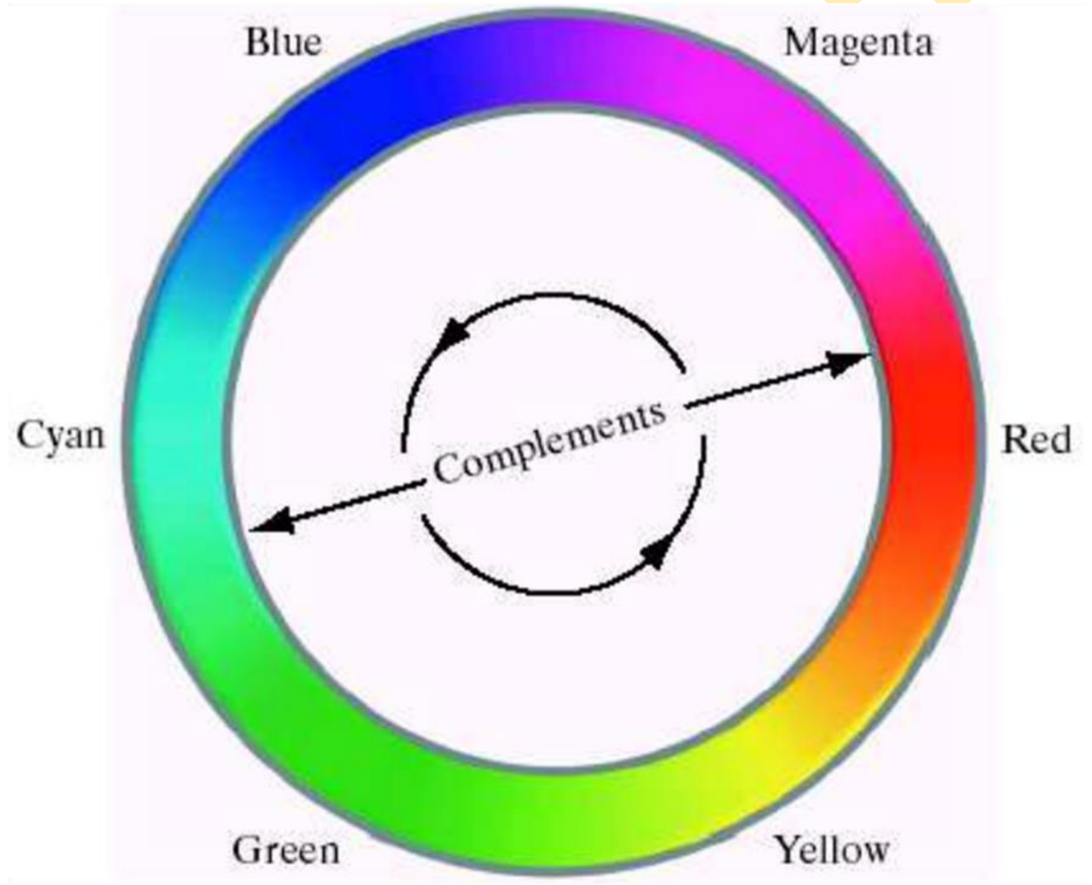
Color Complement:

Hues opposite one another in a color circle are called complements

- Click This is analogous to gray-scale negatives.

- As in the grayscale case, this transformation is useful in enhancing details embedded in dark portions of a color image.

- Complementation can be easily implemented in the RGB space. However, there is no simple equivalent of this in the HIS space. An approximation is possibleto add text



Eample1

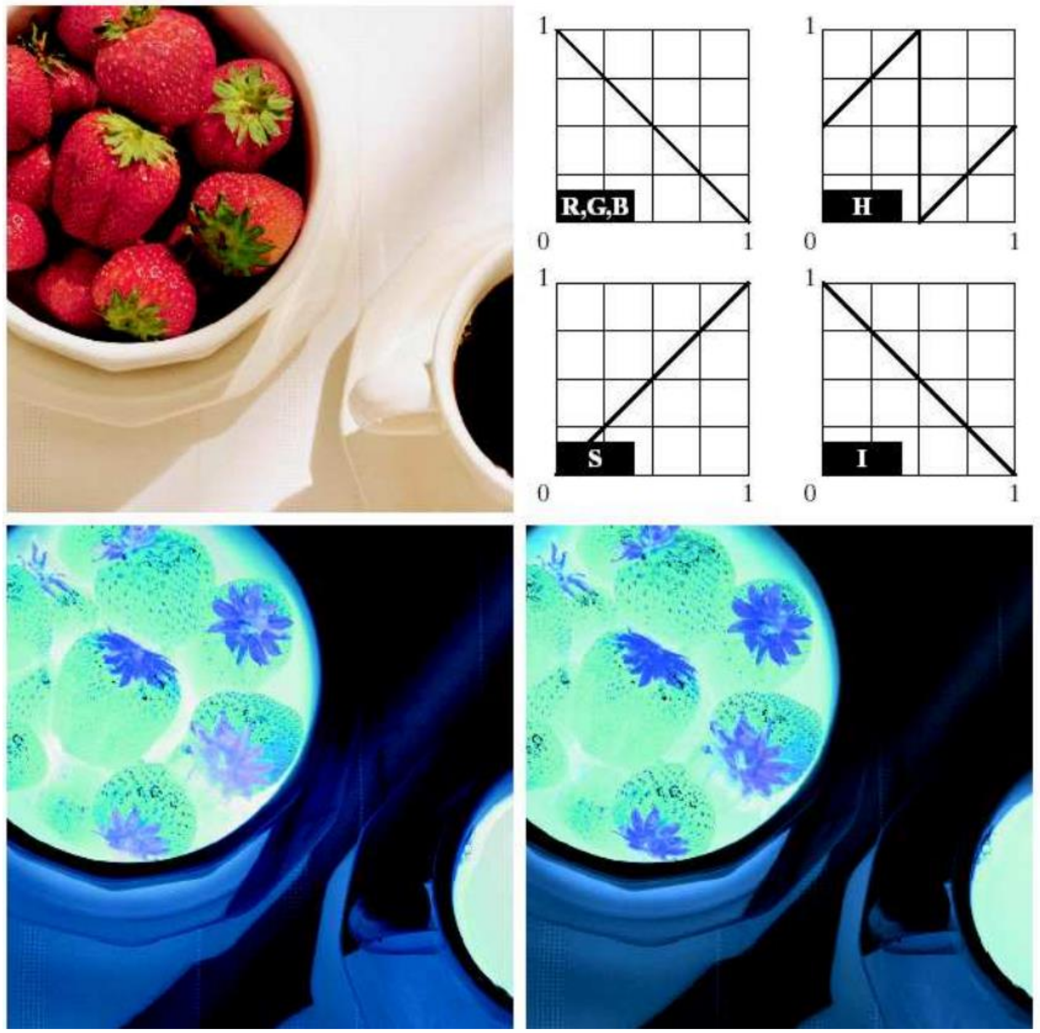
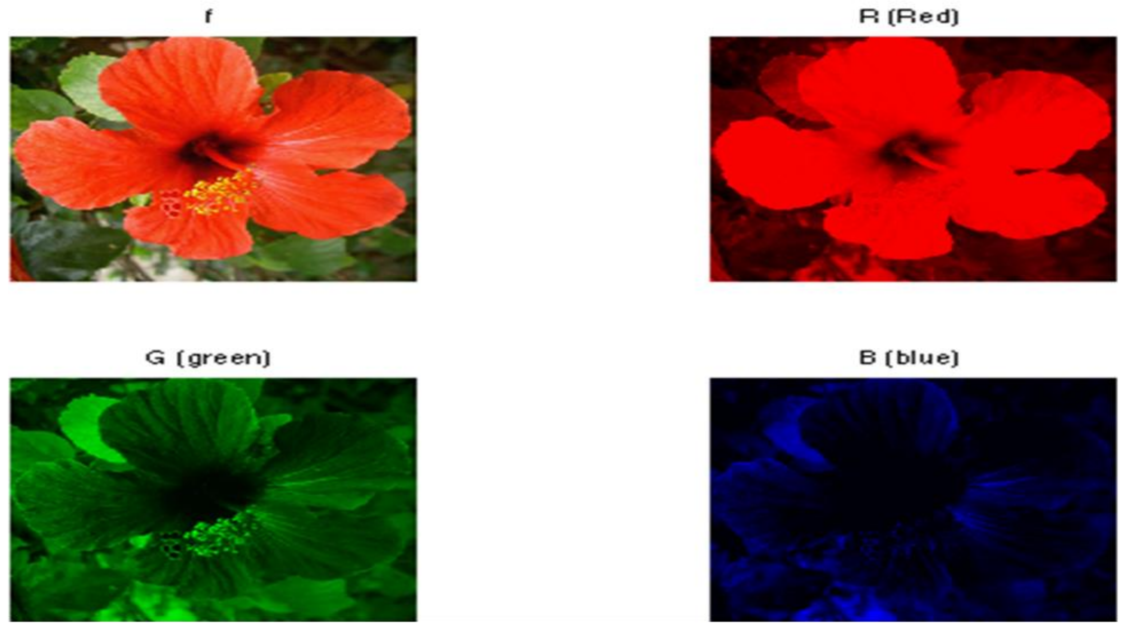


FIGURE 6.33 Color complement transformations. (a) Original image. (b) Complement transformation functions. (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

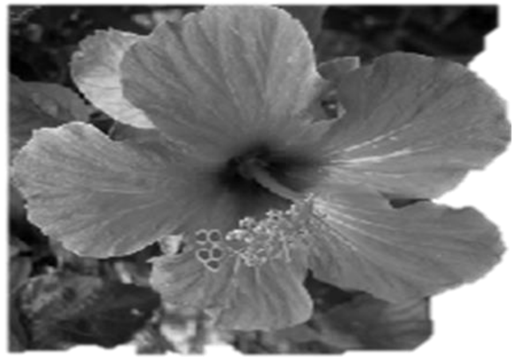
Matlab

- RGB color space
- A color image \(f \in \RR^{N \times 3}\) is made of three independent images, one for each channel red, green and blue (RGB color space).
- Size \(n= n \times n\) of the image.
- n = 256;
- N = n*n;
- Loading an image \(f \in \RR^{N \times 3}\).
- name= 'hibiscus';
- f = rescale(load_image(name,n));
- One can display on screen a color image in RGB space using the rule of additive color mixing.
- Display the image \(f\) and the three channels that compose the colors.
- R = cat(3, f(:,:,1), zeros(n), zeros(n));
- G = cat(3, zeros(n), f(:,:,2), zeros(n));
- B = cat(3, zeros(n), zeros(n), f(:,:,3));
- clf;
- Imageplot({f R G B}, ...
-     { 'f' 'R (red)' 'G (green)' 'B (blue)'}, 2, 2);
- It is possible to obtain a grayscale image from a color image by linear averaging of the channels, to obtain the luminance channel \[L = \frac{r+g+b}{3} \]
- clf;
- imageplot({f mean(f,3)}, {'f' 'L'});



F

L



CMY Color Space

Another popular representation for color images uses as basis colors the cyan, magenta and yellow (CMY color space). They are computed as \[C=1-R, \quad f=1-G, \quad Y=1-B. \]

One can display on screen a color image in CMY space using the rule of substractive color mixing.

Show the C, f, Y channels.

```
f1 = cat(3, f(:,:,1),     f(:,:,2)*0+1, f(:,:,3)*0+1);
f2 = cat(3, f(:,:,1)*0+1, f(:,:,2)    , f(:,:,3)*0+1);
f3 = cat(3, f(:,:,1)*0+1, f(:,:,2)*0+1, f(:,:,3));
clf;
imageplot({f f1 f2 f3}, ...
      { 'f' 'C' 'f' 'Y'}, 2, 2);
```

t

c

f

y

HSV Color Space

- A non-linear color space is obtained by a polar or conical parameterization of a linear color space. The angular coordinate in the plane orthogonal to the first linear axis (which is usually the luminance) is called the Hue, and the radial coordinates is called the saturation.
- Using a luminance which is the sum of the 3 coordinates, one obtain a color system that is quite close to the HSV color system (which has a more complicated definition, but leads to similar results).
- First we compute the value (luminance) coordinate, which is the orthogonal projection on \([1, 1, 1]\).

```
Value = @(f)sum(f, 3) / sqrt(3);
```

- The we compute the projection on the plane orthogonal to \([1, 1, 1]\), for instance using the projections \(A\) and \(B\) on the two orthognoal unit vectors \[[0, 1, -1]/\sqrt{2} \qandq [2, -1, -1]/\sqrt{6}. \]

```
A = @(f)( f(:,:,2)-f(:,:,3) )/sqrt(2);
B = @(f)( 2*f(:,:,1) - f(:,:,2) - f(:,:,3) )/sqrt(6);
```

- The \((V,A,B)\) components are obtained from RGB using a transformation with an orthogonal matrix \(T\).

```
T = [   1/sqrt(3) 1/sqrt(3) 1/sqrt(3); ...
        0 1/sqrt(2) -1/sqrt(2); ...
        2/sqrt(6) -1/sqrt(6) -1/sqrt(6)];
```

- The Hue/Saturation are the polor coordinates within this plane.

```
Saturation = @(f)sqrt( A(f).^2 + B(f).^2 );
Hue = @(f)atan2(B(f),A(f));
```

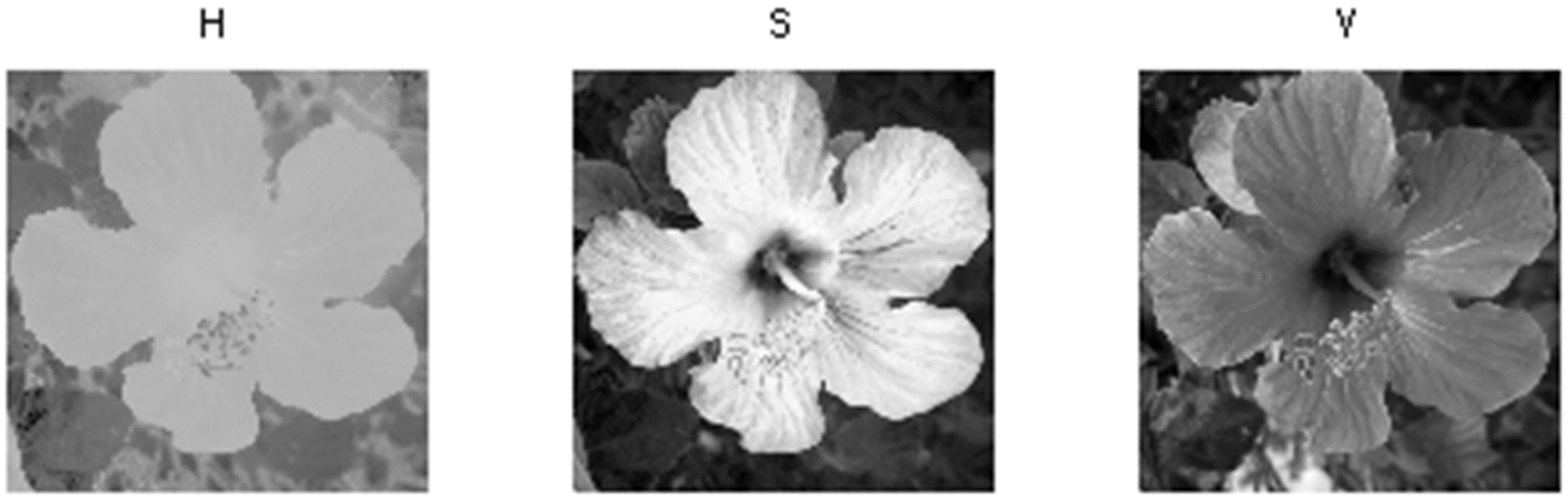
- Shortcut for HSV color transformation. We name it rgb2hsv1 because it is not exactly a mapping to the classical HSV space.

```
rgb2hsv1 = @(f)cat(3, Hue(f), Saturation(f), Value(f));
```

- Compute the transformation.

```
g = rgb2hsv1(f);
```

- Display.


H
S
V

Noise in color image

Noise in color images

- Is random variation of brightness or color information in images
- It can be produced by the image sensor and circuitry of a scanner or digital camera
- Image noise can also originate in film grain .
-
- noise affects all the three color components. Usually, across the three color channels, the noise is independent and its statistical characteristic are identical
- Noise filtering by means of a simple averaging can be accomplished by performing the operation independently on the R, G, and B channels and combining the results.

The problem with noise

Noise is a byproduct of irregular signal fluctuations that accompany a transmitted signal. What’s important to understand here is that these fluctuations are not a part of the signal and instead obscure the intended target.

Thus, one of the most crucial tasks in imaging is finding a solution to create a strong signal with a minimum amount of noise beside it. Unfortunately, finding a solution often proves to be a significant challenge in imaging, particularly in a low-light situation where the signal is already low. When dealing with image noise, the first step is to identify the type of noise you’re encountering.

Noise types

- Temporal noise: **completely random and results from variations in generating a digital value from a single pixel by converting incoming photons into electrons**

- Spatial noise: Variations in an individual pixel typically cause spatial noise and are therefore not random. This noise type is often referred to as "non-uniformities" because the term noise itself implies a random process. EMVA1288 uses the term "non-uniformity

- variations between the pixel can also be caused by temporal noise. The various forms of spatial noise are only observable when minimizing the temporal noise

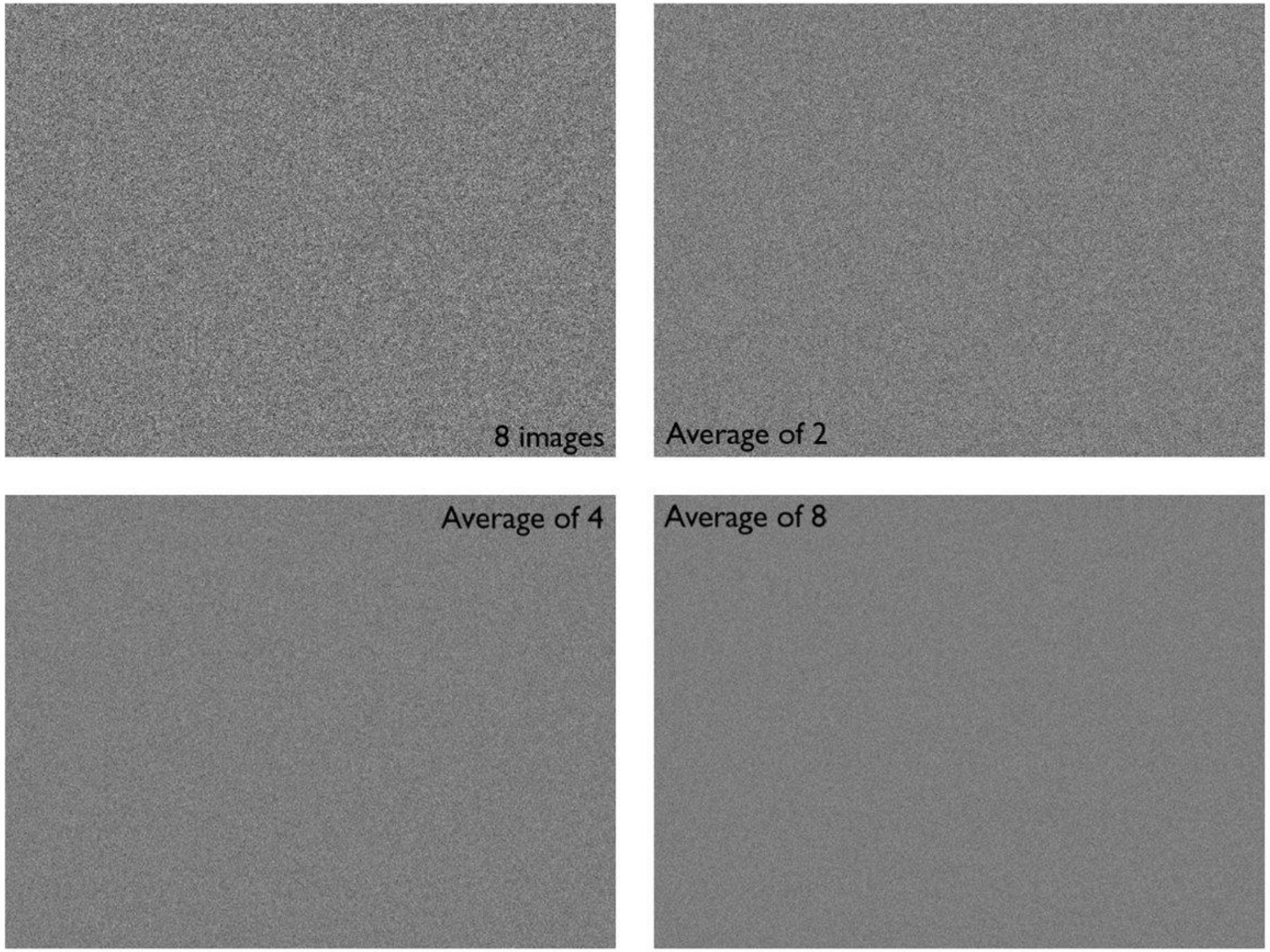


Image 3: Averaging images together to reduce the presence of noise.

Color noise vs. Intensity noise

Color noise is created and amplified during the generation of color information. Essentially, a single-pixel captures only color information for a specific band of the light spectrum (e.g., Red, Green, or Blue).
Color noise is attributed to a process known as demosaicing. Essentially, the missing color information is interpolated from a neighboring pixel to ensure Red, Green, and Blue are obtained in each pixel.

As a result, the noise of an individual pixel will affect the color information of a neighboring pixel. During the interpolation process, the noise in these pixels will distort.

The typical color noise scenario in imaging is strong noise in the blue channel and lower noise in the green and red channels. The intense noise in the blue channel will also affect the other channels due to demosaicing

.

It's important to note that the human observer is much more sensitive to intensity noise than color noise. Nevertheless, intense color noise can still be disturbing to the overall image quality.

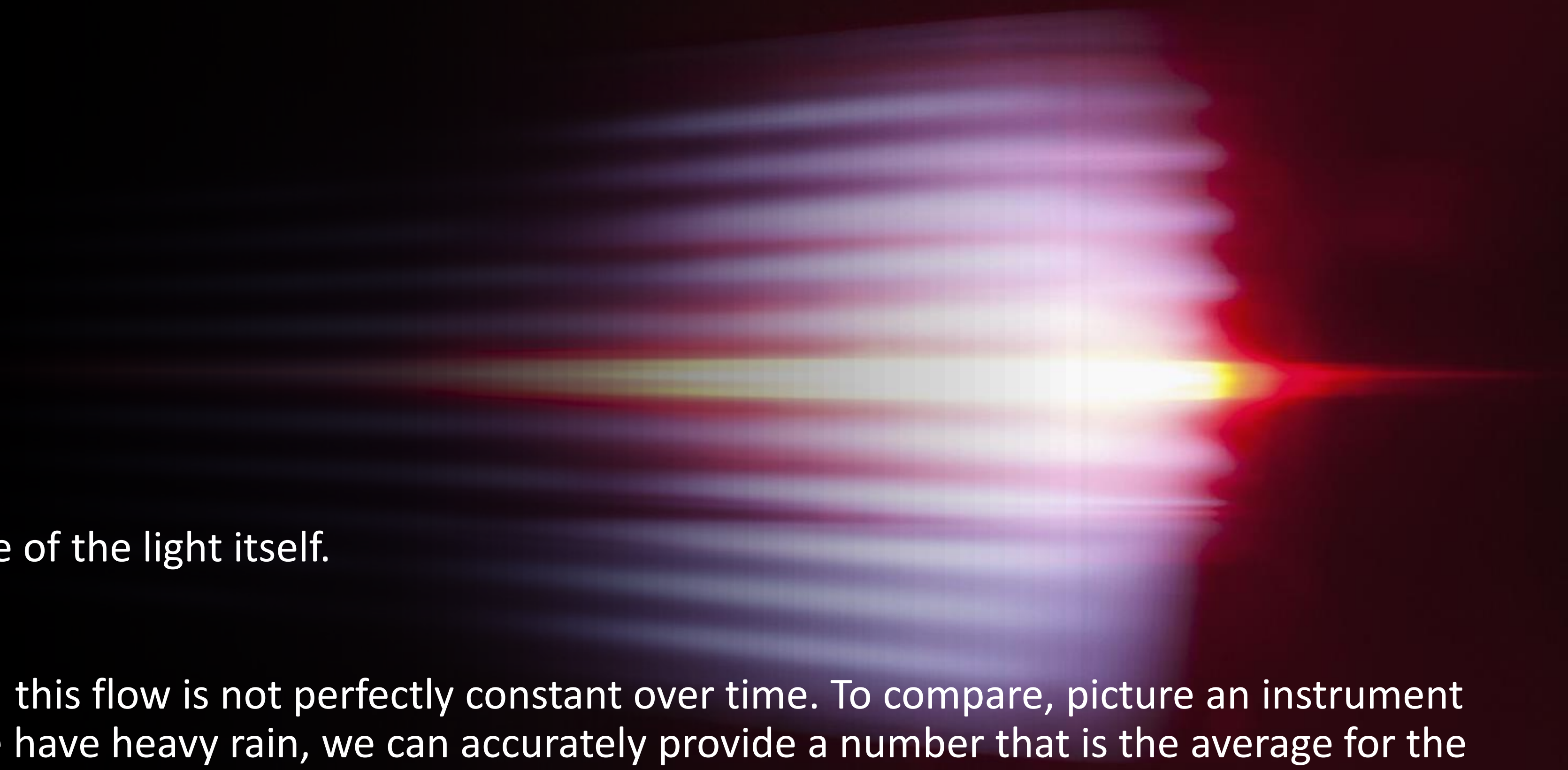
Noise sources

differentiate into two primary noise sources:

1- Photon-Shot noise

2-Read noise.

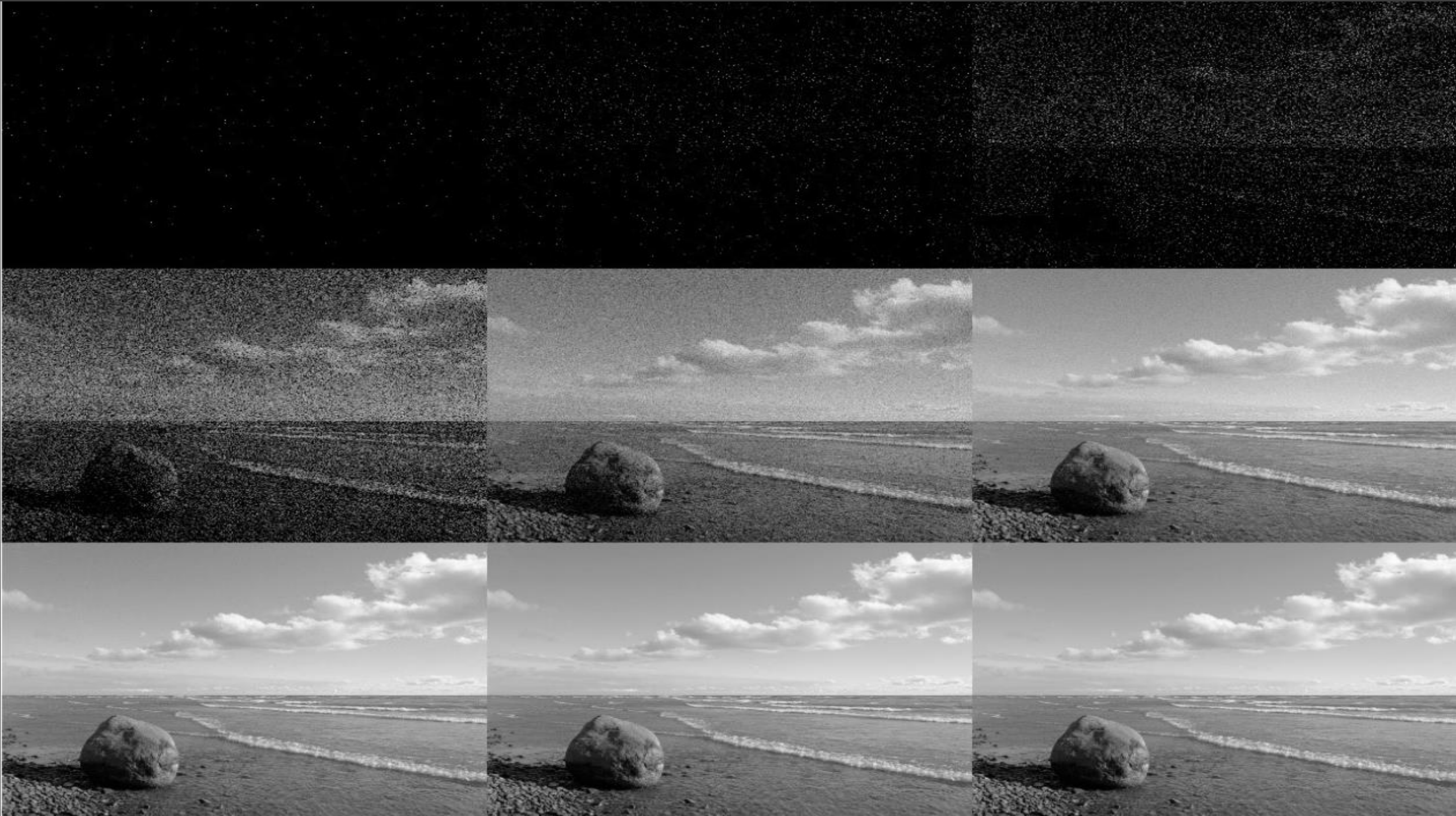
Photon-shot noise •



- The photon-shot noise refers to the noise of the light itself.

- If we imagine light as a flow of photons, this flow is not perfectly constant over time. To compare, picture an instrument that measures rain on a small surface. If we have heavy rain, we can accurately provide a number that is the average for the surface per time interval. However, very light rain will only show a few drops per time interval and change rapidly for the various measurements.

- The same idea from the rain example applies to photon-shot noise. The ratio of signal to noise (SNR) equals the signal's square root for photon-shot noise. Put simply, the more photons we have, the better the SNR and vice versa.



Read noise

Read noise is a summary of multiple types of noise sources within the reading process of the sensor .

In many cases, the noise is constant, so the lower the signal, the worse the SNR. Likewise, the lower the number of photons, the lower the SNR.

When we plot the SNR vs. the number of photons per pixel per exposure, we can differentiate the SNR into two regions:

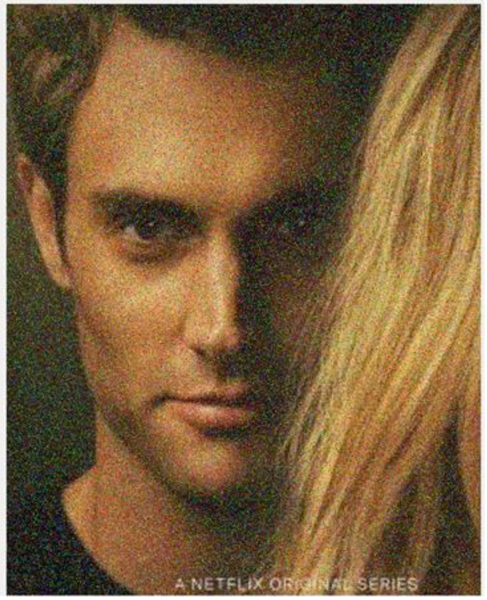
Read-noise limited: Occurs when the read noise is so intense that the SNR is significantly lower than the lowest SNR that appears from the photon shot noise.

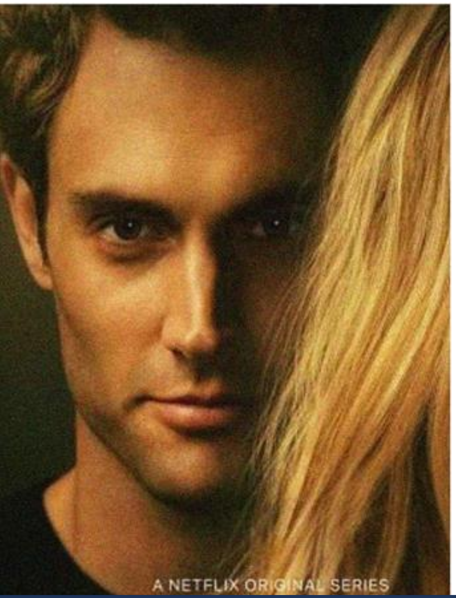
Photon shot noise limited: Occurs when the measured SNR is just slightly below the highest SNR that you can reach with the photon shot noise.

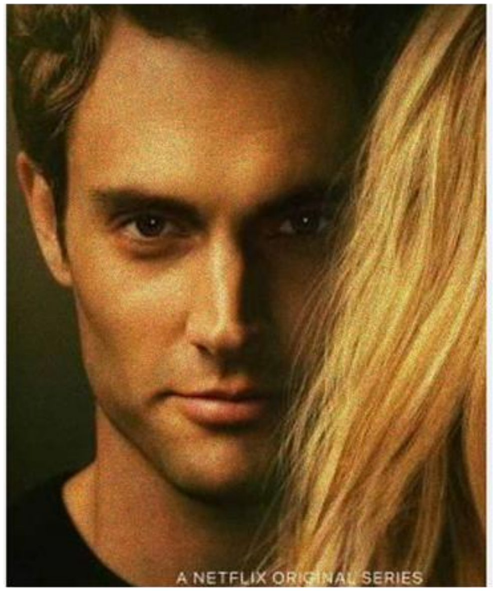
- **Read noise**
- Read noise is a summary of multiple types of noise sources within the reading
- process of the sensor.
- In many cases, the noise is constant, so the lower the signal, the worse the SNR. Likewise, the lower the number of photons, the lower the SNR.
- When we plot the SNR vs. the number of photons per pixel per exposure, we can differentiate the SNR into two regions:
- ***Read-noise limited:*** Occurs when the read noise is so intense that the SNR is significantly lower than the lowest SNR that appears from the photon shot noise.
- ***Photon shot noise limited:*** Occurs when the measured SNR is just slightly below the highest SNR that you can reach with the photon shot noise.

- matlab
- 1.a = imread('D:\ben.png');
- 2.figure,imshow(a)
- 3.%%%%%%%%%%%%%%%%%%%%%%%%%%%Noise%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
- 4.% 1. Salt and pepper noise
- 5.a1 = imnoise(a,'salt & pepper',0.2); %0.2 is the noise ratio
- 6.figure,imshow(a1)
- 7.%Then convert matrix from 3D to 2D
- 8.c = permute(a1, [1 2 3]);
- 9.c = reshape(c, [], size(a1,2), 1);
- 10.%To remove the noise using median filter   (The best filter)
- 11.a2 = medfilt2(c, [5 5]);
- 12.figure,imshow(a2)
- 13.%To remove the noiseLow pass filter (disadvantage: increase the brightness)
- 14.a3 = fspecial('average');
- 15.a4 = filter2(a3,c);
- 16.figure,imshow(a4)
- 17.%%%%%%%%%%%%%%%%%%%%%%%%%
- 18.%2. Gaussian noise
- 19.a1 = imnoise(a, 'gaussian');
- 20.figure,imshow(a1)
- 21.%Then convert matrix from 3D to 2D

```
c = permute(a1, [1 2 3]);
c = reshape(c, [], size(a1,2), 1);
%to remove the noise using median filter   (the best filter)
a2 = medfilt2(c, [5 5]);
figure,imshow(a2)
%to remove the noise using mean filter
a3 = mean(c,3);
figure,imshow(a3)
                    %%%%%%%%%%%%%%%%%%%%%%%
%3. speckle noise
a1 = imnoise(a, 'speckle');
imshow(a1)
%then convert matrix from 3d to 2d
c = permute(a1, [1 2 3]);
c = reshape(c, [], size(a1,2), 1);
%to remove the noise using median filter   (the best filter)
a2 = medfilt2(c, [5 5]);
figure,imshow(a2)
   %%%%%%%%%%%%%%%%%%%%%%%
```

A NETFLIX ORIGINAL SERIES

A NETFLIX ORIGINAL SERIES


A NETFLIX ORIGINAL SERIES


A NETFLIX ORIGINAL SERIES

Thank you