Color Image Processing

Background

- Humans can perceive thousands of colors, and only about a couple of dozen gray shades (cones/rods)
- Divide into two major areas: full color and pseudo color processing
  - Full color – Image is acquired with a full-color sensor like TV camera or color scanner
  - Pseudo color – Assign a color to a range of monochrome intensities
  - The availability of inexpensive and powerful hardware has resulted in the proliferation of applications based on full color processing
- 8-bit color vs 24-bit color
  - Color quantization
- Some of the gray scale image processing methods are directly applicable to color processing but others will need reformation

Color fundamentals

- Color spectrum/prism
  - Figure 7.1
  - White light divided into different colors
  - Colors blend into each other smoothly (Figure 7.2)
- Color – Perceptual result of light in the visible region of spectrum as incident on the retina
  - 400 nm to 700 nm
  - Visible light is a narrow band of frequencies in the electromagnetic spectrum (Figure 6.2)
  - White light is result of reflected light balanced across all visible wavelengths
  - Reflectance from a body in limited range of visible spectrum is perceived as color
    - Green objects reflect light with wavelength in the 500-570nm range while absorbing other wavelengths
- Characterization of light
  - Achromatic (no color) or monochromatic light characterized by intensity
  - Gray level as a scalar measure of intensity from black to white
- Chromatic light
  - Spans the electromagnetic spectrum from approximately 400–700nm
  - Light source characterized by three quantities
    - Radiance Total amount of energy emitted by light source
      - Physical power of light energy, measured in watts
      - Directional quantity
      - Measures the quantity of radiation that passes through or emitted from a surface and falls within a given solid angle in a specified direction
      - Expressed in a spectral power distribution, often in 31 components, each representing a 10 nm band
      - Historically, also called intensity
    - Brightness Achromatic notion of intensity to describe color sensation
* Attribute of a visual sensation according to which an area appears to emit more or less light
* Subjective attribute of an object being observed
* Cannot be measured quantitatively

**Luminance** Measure of amount of energy as *perceived* by an observer, measured in lumens or candelas per square meter

* Light may contain a lot of energy in IR bands but that is not perceptible to the observer
* More tractable version of brightness, defined by CIE
* Radiant power weighted by a spectral sensitivity function that is characteristic of vision
* Luminous efficiency peaks at 555nm
* CIE luminance, denoted by $Y$, is the integral of spectral power distribution, using spectral sensitivity curve as a weighting function
* Magnitude of luminance is proportional to physical power, but spectral composition is related to brightness sensitivity of human vision
* Units of measurement for image processing
  - Normalized to 1 or 100 with respect to a standard white reference
  - $Y = 1$ is the white reference of a studio broadcast monitor whose luminance is 80 cd/m$^2$

Cones in the eye respond to three colors: red, green, blue

* 6 to 7 million cones in human eye
* 65% cones respond to red eye
* 33% cones respond to green light
* 2% cones respond to blue light, these being most sensitive
* Figure 7.3

* Red, green, and blue are known as primary colors
  - In 1931, CIE designated specific wavelengths for primary colors
    - Red – 700nm
    - Green – 546.1nm
    - Blue – 435.8nm
  - To generate all colors, we may have to vary the wavelengths of primary colors while mixing colors; so the three primary colors are neither fixed nor standard
  - The curves in Figure 7.3 indicate that a single color may be called red, green, or blue

Secondary colors

* Created by adding primary colors
  - Cyan = Green + Blue
  - Magenta = Red + Blue
  - Yellow = Red + Green
* Mixing all three primary colors produces white
* Figure 7.4

* The secondary colors are primary colors of pigments, which have red, green, and blue as secondary colors

How do we represent black? Absence of color.

* While printing, we need to print black on white
* Subtractive colors based on pigments
* Primary color of a pigment is defined as one that absorbs a primary color of light and transmits the other two
* Given by cyan, magenta, yellow (CMY)
* A secondary combined with its opposite primary produces black

Color TV reception

* Characterized by additive nature of colors
* Large array of triangular dot patterns of electron sensitive phosphor
* Intensity of individual phosphors modulated by electron gun, one corresponding to each primary color
* The same technology is used in the flat panel displays, using three subpixels to generate a color pixel
* LCDs use properties of polarized light to block/pass light through LCD screen
  · Active matrix display technology uses thin film transistors to provide proper signal to each pixel on screen

  → Color characterized by three quantities

  **Hue** Dominant color as perceived by an observer (red, orange, or yellow)

  **Saturation** Relative purity of color; pure spectrum colors are fully saturated
  * Saturation is inversely proportional to the amount of white light added

  **Brightness** Achromatic notion of intensity

  → Chromaticity
  * Combination of hue and saturation
  * Allows a color to be expressed as its brightness and chromaticity

  → Tristimulus values
  * Three types of cones in the eye require three components for each color, using appropriate spectral weighting functions
    · Based on standard curves/functions defined by CIE – Commission Internationale de L’Éclairage
    · Curves specify the transformation of spectral power distribution for each color into three numbers
  * Amount of red, green, and blue to express a color
  * Denoted by $X$, $Y$, and $Z$
  * Color specified by its tristimulus coefficients

\[
x = \frac{X}{X + Y + Z}
\]
\[
y = \frac{Y}{X + Y + Z}
\]
\[
z = \frac{Z}{X + Y + Z}
\]

* Note that $x + y + z = 1$
* Any wavelength of light in visible spectrum may be expressed by its tristimulus values from curves or tables compiled from experimental results

  → Chromaticity diagram
  * Figure 7.5
  * Color given as a function of $x$ and $y$
  * The corresponding value of $z$ is obtained by $1 - (x + y)$
  * Points on the boundary are fully saturated colors
  * Saturation at point of equal energy is 0
  * Mainly useful for color mixing
    · Any straight line joining two points defines all the color variations obtained by combining the two colors additively
    · Extension to three colors by using a triangle to connect three points
    · Supports the assertion that not all colors can be obtained with three single, fixed primaries as some of them are outside the triangle
  * Figure 7.6 – Color gamut
  Triangle encloses colors produced by RGB monitors
  Shaded region inside the gamut shows the colors printed by high quality printers
**Color models**

- Also called color space or color system
- Allow the specification of colors in some standard way
- Specification of a coordinate system and a subspace within that system
  - Each color represented by a single point
- Models oriented towards hardware (rendering and scanning) or software (reasoning and applications)
  - RGB for monitors/cameras
  - CMY and CMYK for printing
  - HSI for human-like reasoning and interpretation
- **RGB color model**
  - Figure 7.7
  - Unit cube
    - Based on Cartesian coordinate system
    - All color values are assumed to be normalized to the range [0,1]
    - Colors defined by vectors extending from origin; origin represents black
    - RGB primaries are at the corners that are neighbors to the origin; other corners (at distance 2 from origin) represent secondary colors (CMY)
    - Corner opposite to origin, given by point (1,1,1), represents white
    - Different shades of gray are distributed along the cube diagonal from black to white corners
  - Pixel depth – Number of bits used to represent each pixel in RGB space
    - Depth of 24-bits when each color represented by 8 bits in the triplet to represent pixel
    - Figure 6-08
  - Rendering an image
    - Images consist of three component images, one for each primary color
    - Figure 6-09
    - Fuse the three color components together
  - Acquiring an image
    - Figure 6-09, but in reverse
    - Acquire individual color planes and put them together
  - Does not make sense to use all the possible $2^{24}$ colors in 24-bit space
    - Safe colors
      - Can be reproduced on a variety of devices
      - Likely to be reproduced faithfully, reasonably independent of hardware capabilities
    - Safe RGB colors or safe browser colors
      - Number of colors that can be reproduced faithfully in any system – 256
      - 40 of these colors are known to be processed differently by different OSs
      - Number of colors common to most systems – 216
    - Safe RGB color values
      - Formed from 6 possible values of each component as follows

<table>
<thead>
<tr>
<th>Number System</th>
<th>Color Equivalents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex</td>
<td>00 33 66 99 CC FF</td>
</tr>
<tr>
<td>Decimal</td>
<td>00 51 102 153 204 255</td>
</tr>
</tbody>
</table>
• Each successive color is 51 (0x33) more than its predecessor
• Triplets give $6^3 = 216$ possible values
• Figure 6-10
• Not all possible 8-bit gray colors are included in the set of 216 colors
• RGB safe-color cube – Figure 6.11
• Color safe cube has valid colors only on the surface

• CMY and CMYK color models
  – Primary colors of pigments
  – Pigments subtract light rather than radiate light
    * Illuminating a surface coated with cyan pigment absorbs red component of light
  – Devices that deposit color pigments on paper perform an RGB to CMY conversion internally by a simple operation
    \[
    \begin{bmatrix}
    C \\
    M \\
    Y
    \end{bmatrix}
    =
    \begin{bmatrix}
    1 \\
    1 \\
    1
    \end{bmatrix}
    -
    \begin{bmatrix}
    R \\
    G \\
    B
    \end{bmatrix}
    \]
  – Equal contribution of cyan, magenta, and yellow should produce black but in practice, it produces muddy-looking black
    * This may be also due to the fact that the CMY inks may not be pure
    * A fourth color is added, yielding CMYK system
  – Conversion from CMY to CMYK
    \[K = \min(C, M, Y)\]
    * $K = 1$ implies no color contribution, or
    \[
    \begin{align*}
    C &= 0 \\
    M &= 0 \\
    Y &= 0
    \end{align*}
    \]
    * If $K \neq 1$
    \[
    \begin{align*}
    C &= (C - K)/(1 - K) \\
    M &= (M - K)/(1 - K) \\
    Y &= (Y - K)/(1 - K)
    \end{align*}
    \]
  – Conversion from CMYK back to CMY
    \[
    \begin{align*}
    C &= C \times (1 - K) + K \\
    M &= M \times (1 - K) + K \\
    Y &= Y \times (1 - K) + K
    \end{align*}
    \]

• Indexed or palette image
  – Uses a fixed number of colors within the color or gray component of an image
  – Image values are just indices in a table of color values

• HSI color model
  – Hue, saturation, intensity
  – RGB and CMY models
    * Ideally suited for hardware implementation
* RGB matches the human eye’s perception for primary colors
* RGB and CMY not suitable for describing colors for human interpretation
* Dark or light or pastel colors
* Humans do not think of color images as being composed of three primary images that form a single image

-- Human description of images/colors
* In terms of hue, saturation, and brightness

-- HSI model decouples intensity component from the color-carrying components (hue and saturation)
* Ideal tool for developing image processing algorithms
* Natural and intuitive to humans

-- Intensity
* Measure over some interval of the electromagnetic spectrum of the flow of power that is radiated from, or incident on, a surface
* Linear light measure, expressed in units such as watts per square meter
* Controlled on a CRT monitor by voltages presented, in a nonlinear manner for each color component
* CRT voltages are not proportional to intensity
* RGB color images can be viewed as three monochrome intensity images

-- Extracting intensity from RGB images
- Stand the RGB color cube on the black vertex, with white vertex directly above it (Figure 7.10a)
- Line joining the black and white vertices is now vertical
- Intensity of any color given by intersection of intensity axis and a plane perpendicular to it and intersecting with the color point in cube
- Saturation of color increases as a function of distance from intensity axis
- Saturation of points along intensity axis is zero (all points on intensity axis are gray)

-- Hue
* Color attribute that describes a pure color
* Consider the plane defined by black, white, and cyan (Figure 7.10b)
* Intensity axis is contained within this plane
* All points contained in plane segment given by these three points have the same hue – cyan
* Rotating the plane about the intensity axis gives us different hues
* HSI space is represented by a vertical intensity axis and the locus of color points that lie on planes perpendicular to the axis
- As planes move up and down on intensity axis, the boundaries of intersection of each plane with the faces of the cube have a triangular or hexagonal shape

-- Above discussion leads us to conclude that we can convert a color from the RGB values to HSI space by working out the geometrical formulas (Figure 7.11)
* Primary colors are separated by 120°
* Secondary colors are 60° from the primaries
* Hue of a point is determined by an angle from a reference point
- By convention, reference point is taken as angle from red axis
- Hue increases counterclockwise from red axis
* Saturation is the length of vector from origin to the point
- Origin is given by intensity axis

-- Figure 7.12 to describe HSI model

* Converting colors from RGB to HSI

-- Consider RGB values normalized to the range [0, 1]
Given an RGB value, $H$ is obtained as follows:

$$
H = \begin{cases} 
\theta & \text{if } B \leq G \\
360 - \theta & \text{if } B > G
\end{cases}
$$

* It should be normalized to the range $[0, 1]$ by dividing the quantity computed above by 360

$\theta$ is given by

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

* $\theta$ is measured with respect to red axis of HSI space

Saturation is given by

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

Intensity component is given by

$$
I = \frac{1}{3}(R + G + B)
$$

**Converting colors from HSI to RGB**

- Consider the values of HSI in the interval $[0, 1]$
- $H$ should be multiplied by 360 (or $2\pi$) to recover the angle; further computation is based on the value of $H$

**RG sector** – $0^\circ \leq H < 120^\circ$

$$
\begin{align*}
B &= I(1 - S) \\
R &= I \left[ 1 + \frac{S \cos H}{\cos(60^\circ - H)} \right] \\
G &= 3I - (R + B)
\end{align*}
$$

**GB sector** – $120^\circ \leq H < 240^\circ$

$$
\begin{align*}
H' &= H - 120^\circ \\
R &= I(1 - S) \\
G &= I \left[ 1 + \frac{S \cos H'}{\cos(60^\circ - H')} \right] \\
B &= 3I - (R + G)
\end{align*}
$$

**BR sector** – $0^\circ \leq H < 360^\circ$

$$
\begin{align*}
H' &= H - 240^\circ \\
G &= I(1 - S) \\
B &= I \left[ 1 + \frac{S \cos H'}{\cos(60^\circ - H')} \right] \\
R &= 3I - (G + B)
\end{align*}
$$

**Figure 7.13**

- HSI components of RGB cube, plotted separately
- Discontinuity along the $45^\circ$ line in the hue figure
  * See the reason by going around the middle in Figure 7.8
- Saturation image shows progressively darker values close to the white vertex of RGB cube
- Intensity is simply the average of RGB values at the corresponding pixel
• Manipulating HSI component images
  – Figure 7.14 – image composed of primary and secondary RGB colors and their HSI equivalents
    * In hue, red region maps to black as its angle is 0°
    * In b, c, and d parts of the image, the pixels are scaled to the range [0,1]
  – Individual colors changed by changing the hue image
  – Purity of colors changed by varying the saturation
  – Figure 7.15a – Change blue and green pixels in Figure 7.14a to 0 (compare with Figure 7.14b)
  – Figure 7.15b – Change saturation of cyan component in Figure 7.14c to half
  – Figure 7.15c – Reduce the intensity of central white region in Figure 7.14d by half
  – Figure 7.15d – Combine the three HSI components back into RGB image

HSV color space

• Projects the RGB color cube onto a non-linear chroma angle (H), a radial saturation percentage (S), and a luminance-inspired value (V)
• Similar to HSI color space
• Used to compare the hue channel in OpenCV

Pseudocolor image processing

• Also known as indexed color or false color
• Assign colors to gray values based on a fixed criteria
  – 216 index entries from 8-bit RGB color system as a 6 × 6 × 6 cube in a direct color system
  – Gives an integer in the range 0 to 5 for each component of RGB
  – Requires less data to encode an image
  – Some graphics file formats, such as GIF and TIFF add an index colormap to the image with gamma-corrected RGB entries
• Used as an aid to human visualization and interpretation of gray-scale events in an image or sequence of images, such as visualizing population density or temperature in different areas on a map

\[
g(x, y) = \begin{cases} 
0XFF0000 & \text{if } g(x, y) > 100 \\
0XFF00FF & \text{if } 90 < g(x, y) \leq 100 \\
0XFFFF00 & \text{if } 80 < g(x, y) \leq 90 \\
0X00FF00 & \text{if } 70 < g(x, y) \leq 80 \\
0X00FFFF & \text{if } 60 < g(x, y) \leq 70 \\
0X0000FF & \text{otherwise}
\end{cases}
\]

• May have nothing to do with processing of true color images
• Intensity slicing
  – Also called density slicing or color coding
  – Slicing planes parallel to horizontal plane in 3D space, with the intensity of image providing the third dimension on image plane
    * Figure 7.16
* Plane at $f(x, y) = l_i$ to slice the image function into two levels
* Assign different colors to area on different sides of the slicing plane
* Relative appearance of the resulting image manipulated by moving the slicing plane up and down the gray-level axis

- Technique summary
  * Gray scale representation – $[0, L - 1]$
  * Black represented by $l_0, [f(x, y) = 0]$
  * White represented by $[l_{L-1}], [f(x, y) = L - 1]$
  * Define $P$ planes perpendicular to intensity axis at levels $l_1, l_2, \ldots, l_P$
  * $0 < P < L - 1$
  * $P$ planes partition the gray scale into $P + 1$ intervals as $I_1, I_2, \ldots, I_{P+1}$
  * Make gray-level to color assignment as
    \[
    f(x, y) = c_k \text{ if } f(x, y) \in I_k
    \]
    where $c_k$ is the color associated with $k$th intensity interval $I_k$ defined by partitioning planes at $l = k - 1$ and $l = k$

- Alternative mapping function to intensity slicing planes
  * Figure 7.17
  * Staircase form of mapping with multiple levels
  - Figure 7.18 – Picker Thyroid Phantom (radiation test pattern)
    * Intensity slicing image into eight color regions
    * Idea is to make it easy to distinguish between shades without assigning any semantic interpretation to the color
    * Characteristics of intensity variations in grayscale image are more apparent by varying the number of colors and the span of intensity intervals
  - Figure 7.19
    * Full strength of X-rays passing through is assigned one color; everything else a different color
  - Figure 7.20 – Measurement of rainfall levels
    * Current temperature map of US

* Gray level to color transformations
  - Separate independent transformation of gray level inputs to three colors
  - Figure 7.21
  - Composite image with color content modulated by nature of transformation function
  - Piecewise linear functions of gray levels
  - Figure 7.22 – Luggage through X-ray scanning system
    * Image on right contains a block of simulated plastic explosives
    * Figure 7.23 – Transformation functions used
    * Emphasize ranges in gray scale by changing sinusoidal frequencies

* Combining several monochrome images into a single color composite
  - Figure 7.24
  - Used in multispectral image processing, with different sensors producing individual monochrome images in different spectral bands
  - Figure 7.25
    * Images of Washington, DC, and Potomac river in red, green, blue, and near IR bands
Image $f$ generated by replacing the red component of image $e$ by NIR image
- NIR strongly responsive to biomass component
- Image $f$ shows the difference between biomass (red) and man-made features such as concrete and asphalt (bluish green)

- Figure 7.26
  - Jupiter moon Io, using images in several spectral regions by the spacecraft Galileo
  - Bright red depicts material recently ejected from an active volcano while surrounding yellow shows older sulfur deposits

- Chroma key compositing or Chroma keying
  - Used in movies and TV broadcasting to separate foreground from background to blend multiple images
  - Make a color range in the foreground region transparent, and insert separately filmed background footage into the scene
  - Most common use is weather forecasting using green screen
    - Green (or blue) color used more often because they are distinct from human skin tones
    - The camera is most sensitive to the green light
    - Notice that the weatherman will never appear in a green tie or any other green-colored dress

  - Produce a mask by thresholding the amount of green color
    \[
    M(x, y) = \begin{cases} 
    1 & \text{if } I_G(x, y) > \tau \\
    0 & \text{otherwise} 
    \end{cases}
    \]

- Example of memes with Melania Trump during Republican National Convention

**Basics of full-color image processing**

- Two major categories of processing
  1. Process each component of image (RGB or HSI) individually and then form a composite processed color image
    - Each component can be processed using gray-scale processing techniques
  2. Work with color pixels directly, treating each pixel as a vector
    \[
    \text{Color vector } \mathbf{c} = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}
    \]
    - Since each pixel is a function of coordinates $(x, y)$, we have
    \[
    \mathbf{c}(x, y) = \begin{bmatrix} c_R(x, y) \\ c_G(x, y) \\ c_B(x, y) \end{bmatrix} = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix}
    \]
    - Each component of the vector is a spatial variable in $x$ and $y$
    - For an $M \times N$ image, there are $MN$ vectors $\mathbf{c}(x, y)$ for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$
    - As the data in a pixel contains two or more components, we need a third dimension to represent the pixel, leading to the term *voxel*, or volumetric pixel

- The two methods may or may not produce equivalent results
  - Scalar versus vector operations
    - The process used should be applicable to both scalars and vectors
* The operation on each component of the vector must be independent of the other components
  – Neighborhood processing may be an example where we get similar or different results (Figure 7.27)
  * Averaging the images separately in individual planes and averaging the vectors will give the same result

**Color transformations**

• Process the components of a color image within the context of a single color model, without converting components to different color space

• Think of an application that needs to brighten a picture
  – Can we achieve this by adding a constant quantity to each of the three RGB channels?
  – This will not only increase the intensity of each pixel but also hue and saturation
  – A better solution will be to manipulate the luminance I to recompute a valid RGB image with the same hue and saturation

• Formulation
  – Model color transformations using the expression
    \[ g_i(x, y) = T_i[f_i(x, y)] \quad i = 1, 2, \ldots, n \]
    
    \( n \) is the number of component images or channels, \( T_i \) is the set of transformation (or color mapping functions) over a spatial neighborhood of \((x, y)\)
  – For RGB images, each \( f(x, y) \) component is a triplet in the chosen color space (Figure 7.27)
  – The subscript \( i \) for \( T \) indicates that each component may have a different transform
  – Figure 7.28 – Various color components of an image
  * The transform \( T_i \) may be applied to any component of the image
  – Must consider the cost of converting from one color space to another when looking at the operations
  – Modifying intensity of an image in different color spaces, using the transform
    \[ g(x, y) = kf(x, y) \]

  * In HSI color space, converting a pixel \( h, s, i \) to \( h', s', i' \)
    \[ h' = h \]
    \[ s' = s \]
    \[ i' = ki \]

  * In RGB color space, converting a pixel \( r, g, b \) to \( r', g', b' \)
    \[
    \begin{bmatrix}
    r' \\
    g' \\
    b'
    \end{bmatrix}
    = k \cdot
    \begin{bmatrix}
    r \\
    g \\
    b
    \end{bmatrix}
    \]

  * In CMY color space
    \[ c' = kc + (1 - k) \]
    \[ m' = km + (1 - k) \]
    \[ y' = ky + (1 - k) \]

  * In CMYK space
    \[ g_i(x, y) = \begin{cases} f_i(x, y) & i = 1, 2, 3 \\ kf_i(x, y) + (1 - k) & i = 4 \end{cases} \]
– Simple operation in HSI but cost to convert to HSI may not be justifiable
  * Figure 7.29, using $k = 0.7$
– Application of transform to each component is independent of other components

• Color complements
– Hues directly opposite one another on the color circle
  * Figure 7.30
– Analogous to gray scale negatives
– Can be used to enhance details buried in dark regions of an image
  * Figure 7.31
  * May not have the same saturation in negative image in HSI
  * Figure shows saturation component unaltered

• Color slicing
– Used to highlight a specific range of colors in an image to separate objects from surroundings
– Display just the colors of interest, or use the regions defined by specified colors for further processing
– More complex than gray-level slicing, due to multiple dimensions for each pixel
  * Practical color-slicing approaches require each pixel’s transformed components to be a function of all components of the original pixel
– Dependent on the color space chosen; I prefer HSI
– Mapping unwanted colors (outside of specified range) to a neutral color
  * Use a cube (or hypercube) of width $W$ to enclose the reference color with components $(a_1, a_2, \ldots, a_n)$
  * Transformation is given by
    \[
    s_i = \begin{cases} 
    0.5 & \text{if } \sum_{j=1}^{n} (r_j - a_j)^2 > R_0^2 \\
    r_i & \text{otherwise}
    \end{cases} 
    \]
  * If the color of interest is specified by a sphere of radius $R_0$, the transformation is
    \[
    s_i = \begin{cases} 
    0.5 & \text{if } \sum_{j=1}^{n} (r_j - a_j)^2 > R_0^2 \\
    r_i & \text{otherwise}
    \end{cases} 
    \]
– Figure 7.32 – Separate the strawberries in Figure 7.29a
  * Select RGB color (0.6863, 0.1608, 0.1922)
  * $W = 0.2549$, $R = 0.1765$

• Color balancing
– Process to compensate for incandescent lighting
– You can perform color balancing by multiplying each channel with a different scale factor, or by mapping the pixels to XYZ color space, changing the nominal white point, and mapping back to RGB

Tone and color corrections

• Used for photo enhancement and color reproduction
– Tonal range of an image
  * Also called key-type
  * Gives general distribution of color intensities
* High key, middle key, and low key images
  * Distribute intensities equally between highlights and shadows

- Device independent color model from CIE relating the color gamuts
- Use a color profile to map each device to color model
- CIE $L^*a^*b^*$ system
  - Most common model for color management systems
  - Components given by the following equations

\[
\begin{align*}
L^* &= 116 \cdot h\left(\frac{Y}{Y_W}\right) - 16 \\
a^* &= 500 \cdot \left[h\left(\frac{X}{X_W}\right) - h\left(\frac{Y}{Y_W}\right)\right] \\
b^* &= 200 \cdot \left[h\left(\frac{Y}{Y_W}\right) - h\left(\frac{Z}{Z_W}\right)\right]
\end{align*}
\]

where
\[
h(q) = \begin{cases} 
q^{\frac{1}{3}} & \text{if } q > 0.008856 \\
7.787q + \frac{16}{116} & \text{otherwise}
\end{cases}
\]

- $X_W$, $Y_W$, and $Z_W$ are values for reference white, called $D_65$ which is defined by $x = 0.3127$ and $y = 0.3290$ in the CIE chromaticity diagram
- $X, Y, Z$ are computed from RGB values as

\[
\begin{bmatrix} 
X \\
Y \\
Z 
\end{bmatrix} = \begin{bmatrix} 
0.412453 & 0.357580 & 0.180423 \\
0.212671 & 0.715160 & 0.072169 \\
0.019334 & 0.119193 & 0.950227
\end{bmatrix} \begin{bmatrix} 
R_{709} \\
G_{709} \\
B_{709}
\end{bmatrix}
\]

- Rec. 709 RGB corresponds to $D_65$ white point
- $L^*a^*b^*$ is calorimetric (colors perceived as matching are encoded identically), perceptually uniform (color differences among various hues are perceived uniformly), and device independent
- Not directly displayable on any device but its gamut covers the entire visible spectrum
- $L^*a^*b^*$ decouples intensity from color, making it useful for image manipulation (hue and contrast editing) and image compression applications
  * $L^*$ represents lightness or intensity
  * $a^*$ gives red minus green
  * $b^*$ gives green minus blue
- Allows tonal and color imbalances to be corrected interactively and independently
  * Tonal range refers to general distribution of key intensities in an image
    - Adjust image brightness and contrast to provide maximum detail over a range of intensities
    - The colors themselves are not changed
  * Figure 7.33
  * Typical RGB transformations to correct three common tonal imbalances – flat, light, and dark images
  * Keys in the images are visually evident; they may be computed using histograms of color components

- Color balancing
  - Objectively performed using a color spectrometer
  - Can also be assessed visually using skin tones (Figure 7.34)
Adjusting color components

- Every action affects the overall color balance of the image
- Perception of a color is affected by surrounding colors
- Use color wheel (Figure 7.30) to increase the proportion of a color by decreasing the amount of complementary color
- May also increase the proportion of a color by raising the contribution of its adjacent colors

Histogram processing

- Provides an automated way to perform enhancement
- Histogram equalization
  - Adapt the grayscale technique to multiple components
  - Applying grayscale techniques to different colors independently yields erroneous colors
  - Spread the intensities uniformly leaving the hues unchanged
  - Figure 7.35 – histogram equalization followed by saturation adjustment in HSI color space

Compositing

- Used for special effects in movies; blend live action with graphics
- Method to blend two layers into one another
- Dissolving
  - Simplest effect
  - Given two images $I_1$ and $I_2$ of the same size
  - Their weighted combination is given by
    \[ I'(x, y) = w_1 I_1(x, y) + w_2 I_2(x, y) \]
  - $w_1$ and $w_2$ are scalar weights with the condition that $w_1 + w_2 = 1$
  - Changing $w_1$ slowly from 1 to 0, with $w_2$ computed as $1 - w_1$ slowly dissolves the first image into the second
- Compositing with binary masks
  - Let the two images be accompanied by binary masks $M_1$ and $M_2$ to define the support of pixels
    - $M(x, y) = 1$ if $I(x, y)$ is valid
    - $M(x, y) = 0$ if $I(x, y)$ is invalid
    - The main idea is to work just with the valid pixels
  - Binary masks allows for four cases for any given pixel in two images
    - $M_1(x, y) = M_2(x, y) = 0 \Rightarrow$ output pixel mask $M'(x, y) = 0$, or both inputs are invalid; value of input pixels is irrelevant
    - $M_1(x, y) = 0, M_2(x, y) = 1 \Rightarrow$ output pixel mask can be 0 (invalid) or 1 (valid); set output pixel to the only valid input $I_2(x, y)$
    - $M_1(x, y) = 1, M_2(x, y) = 0 \Rightarrow$ output pixel mask can be 0 (invalid) or 1 (valid); set output pixel to the only valid input $I_1(x, y)$
    - $M_1(x, y) = 1, M_2(x, y) = 1$ leads to three choices: output pixel can be selected from $I_1(x, y)$ or $I_2(x, y)$ or neither
The binary masks give us 1 choice for the first, 2 choices for the second and third, and three choices for the fourth case.

This leads to $1 \cdot 2 \cdot 2 \cdot 3 = 12$ possible ways to combine two images with binary masks.

The 12 compositing operators are known as Porter-Duff operators.

With $I_1$ and $I_2$ as the input images and $I'$ as the output image, the operators can be summarized as:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Output $I'(x,y)$</th>
<th>Mask $M'(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>copy $I_1$</td>
<td>$I_1(x,y)$</td>
<td>$M_1(x,y)$</td>
</tr>
<tr>
<td>copy $I_2$</td>
<td>$I_2(x,y)$</td>
<td>$M_2(x,y)$</td>
</tr>
<tr>
<td>$I_1$ over $I_2$</td>
<td>$(I_1 \land M_1) \lor (I_2 \land M_2 \land \neg M_1)$</td>
<td>$M_1 \lor M_2$</td>
</tr>
<tr>
<td>$I_1$ in $I_2$</td>
<td>$I_1$</td>
<td>$M_1 \land \neg M_2$</td>
</tr>
<tr>
<td>$I_1$ out $I_2$</td>
<td>$I_1$</td>
<td>$M_1 \land \neg M_2$</td>
</tr>
</tbody>
</table>

**Blend Modes**

**Smoothing and sharpening**

- **Color image smoothing**
  - Extend grayscale spatial filtering mask to color smoothing, dealing with component vectors.
  - Let $S_{xy}$ be the neighborhood centered at $(x,y)$ in an RGB color image.
  - Average of RGB components in the neighborhood is given by:
    \[
    \bar{c}(x,y) = \frac{1}{K} \sum_{(s,t) \in S_{xy}} c(s,t)
    \]
    which is the same as:
    \[
    \bar{c}(x,y) = \left[ \begin{array}{c} \frac{1}{K} \sum_{(s,t) \in S_{xy}} R(s,t) \\ \frac{1}{K} \sum_{(s,t) \in S_{xy}} G(s,t) \\ \frac{1}{K} \sum_{(s,t) \in S_{xy}} B(s,t) \end{array} \right]
    \]
  - Same effect as smoothing each channel separately.
  - Figure 7.36 (RGB components), Figure 7.37 (HSI components).
    - Figure 7.38a – Smooth each component of RGB image independently.
    - Figure 7.38b – Smooth only the intensity component of HSI.
    - Pixel colors do not change as they do with RGB smoothing.
    - Figure 7.38c – Difference between the two smoothed out images.
    - More efficient to smooth in HSI; difference from RGB becomes more pronounced with increase in kernel size.

- **Color image sharpening**
  - Use Laplacian:
    \[
    \nabla^2[c(x,y)] = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix}
    \]
  - Figure 7.39
    - a: Laplacian applied to RGB components independently.
    - b: Laplacian applied to just the intensity component of HSI.
    - c: Difference between the two sharpened images.

**Image segmentation based on color**
• Segmentation partitions images into regions

• Segmentation in HSI color space
  – Color is conveniently represented in hue component
  – Saturation is used as a masking image to isolate regions of interest in the hue component
  – Intensity image used less frequently as it has no color information
  – Example 7.14
    * Segment the reddish region in lower left of Figure 7.40a
    * Figure 7.40e: Binary mask by thresholding the saturation image with 10% of the maximum value in the image
    * Figure 7.40f: Product of hue and thresholded saturation
    * Figure 7.40g: Histogram of Figure 7.40f

• Segmentation in RGB vector space
  – Create an estimate of the average color to be segmented as vector \( \mathbf{a} \)
  – Let \( \mathbf{z} \) be an arbitrary point in the RGB color space
  – \( \mathbf{z} \) is similar to \( \mathbf{a} \) if the Euclidean distance between them is less than specified threshold \( D_0 \)
    \[
    D(\mathbf{z}, \mathbf{a}) = |\mathbf{z} - \mathbf{a}| = [(\mathbf{z} - \mathbf{a})^T(\mathbf{z} - \mathbf{a})]^{\frac{1}{2}} = [(z_R - a_R)^2 + (z_G - a_G)^2 + (z_B - a_B)^2]^{\frac{1}{2}}
    \]
  – Locus of points such that \( D(\mathbf{z}, \mathbf{a}) \leq D_0 \) is a solid sphere of radius \( D_0 \) (Figure 7.41a)
  – Simple generalization provided by
    \[
    D(\mathbf{z}, \mathbf{a}) = [(\mathbf{z} - \mathbf{a})^T C^{-1}(\mathbf{z} - \mathbf{a})]^{\frac{1}{2}}
    \]
    where \( C \) is the covariance matrix of samples chosen to represent color range to be segmented
    * Locus of points such that \( D(\mathbf{z}, \mathbf{a}) \leq D_0 \) is a solid ellipsoid with principal axis oriented in the direction of maximum data spread (Figure 7.41a)
    * When \( C = I \), the ellipsoid reduces to sphere
  – Distances are positive and monotonic
    * We can work with square of the distance to avoid computationally expensive square roots
    * Still, the computations are expensive for images of practical size
    * Compromise by using a bounding box (Figure 7.41c)
    * Box is centered on \( \mathbf{a} \), with dimensions along the color axes proportional to the standard deviation of the samples along each axis
    * A given color point is segmented based on whether it is inside the box
  – Example 7.15
    * Figure 7.42
    * Color to be segmented selected by rectangular region inside Figure 7.42a
    * Compute mean vector \( \mathbf{a} \) of the shades in the rectangle
    * Compute the standard deviation of the red, green, and blue components

Color Edge Detection

• Computing edges in individual component planes in the image
  – Edge detection by gradient operator for image sharpening
* May not work with vector data for pixels
* Working on separate color channels and combining the result may not work either
* Figure 7.43
  · Results may be acceptable but not accurate
  - Define the gradient (magnitude and direction) of the vector \( c \) at any point \((x, y)\)
  - For a scalar function \( f(x, y) \), gradient is a vector pointing in the direction of maximum rate of change of \( f \) at coordinates \((x, y)\)
  - Let \( r \), \( g \), and \( b \) be the unit vectors along the R, G, and B axes of RGB color space
  - Define the vectors
    \[
    \mathbf{u} = \frac{\partial R}{\partial x} \mathbf{r} + \frac{\partial G}{\partial x} \mathbf{g} + \frac{\partial B}{\partial x} \mathbf{b} \\
    \mathbf{v} = \frac{\partial R}{\partial y} \mathbf{r} + \frac{\partial G}{\partial y} \mathbf{g} + \frac{\partial B}{\partial y} \mathbf{b}
    \]
  - The gradients are defined as the dot product of these vectors as
    \[
    g_{xx} = \mathbf{u} \cdot \mathbf{u} = \mathbf{u}^T \mathbf{u} = \left| \frac{\partial R}{\partial x} \right|^2 + \left| \frac{\partial G}{\partial x} \right|^2 + \left| \frac{\partial B}{\partial x} \right|^2 \\
    g_{yy} = \mathbf{v} \cdot \mathbf{v} = \mathbf{v}^T \mathbf{v} = \left| \frac{\partial R}{\partial y} \right|^2 + \left| \frac{\partial G}{\partial y} \right|^2 + \left| \frac{\partial B}{\partial y} \right|^2 \\
    g_{xy} = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \frac{\partial R}{\partial x} \frac{\partial R}{\partial y} + \frac{\partial G}{\partial x} \frac{\partial G}{\partial y} + \frac{\partial B}{\partial x} \frac{\partial B}{\partial y}
    \]
  - Direction of maximum rate of change of \( c(x, y) \) is given by
    \[
    \theta(x, y) = \frac{1}{2} \tan^{-1} \left( \frac{2g_{xy}}{g_{xx} - g_{yy}} \right)
    \]
  - Value of the rate of change at \((x, y)\) in the direction of \( \theta(x, y) \) is given by
    \[
    F_\theta(x, y) = \left\{ \frac{1}{2} \left[ (g_{xx} + g_{yy}) + (g_{xx} - g_{yy}) \cos 2\theta(x, y) + 2g_{xy} \sin 2\theta(x, y) \right] \right\}^{1/2}
    \]
  - Since \( \tan(\alpha) = \tan(\alpha \pm \pi) \), if \( \theta_0 \) is a solution to the equation for \( \theta \) above, so is \( \theta_0 \pm \pi/2 \)
    · Additionally, \( F_\theta = F_{\theta+\pi} \)
    · The equation for \( \theta \) gives two values 90° apart, or a pair of orthogonal directions
    · \( F \) is maximum along one direction and minimum along the other
  - Example 7.16 – Edge detection in RGB vector space
  * Figure 7.44

**Image File Formats**

- Files used to store, archive, and exchange image data
  - Standardized file formats facilitate the exchange of images and allow different applications to read those images
- Criteria to select appropriate file format
  - Image type
    * Binary, grayscale, or color images
    * Document scans, floating point images
    * Maximum image size for satellite images
  - Storage size and compression
* Lossy or lossless compression
  – Compatibility
    * Exchange of image data with others and across applications
    * Long-term machine readability of data
  – Application domain
    * Print, web, film, graphics, medicine, astronomy

• Raster vs vector data
  – All images considered thus far have been raster images
  – Vector graphics represent geometric objects using continuous coordinates
    * The objects are rasterized when they need to be displayed on a physical device
  – Used to encode geodata for navigation systems

• Tagged Image File Format (TIFF)
  – Supports grayscale, indexed, and true color images
  – A single file may contain a number of images with different properties
  – Provides a range of different compression methods (LZW, ZIP, CCITT, and JPEG), and color spaces
  – You can create new image types and information blocks by defining new tags
    * Proprietary tags may not be always supported leading to “unsupported tag” error
    * Web browsers do not natively support TIFF

• Graphics Interchange Format (GIF)
  – Originally designed by CompuServe in 1986
  – Provided early support for indexed color at various bit depths
  – Provided LZW compression, interlaced image loading, and ability to encode simple animations by storing a number of images in a single file for sequential display
  – Does not support true color images
  – Allows pixels to be encoded using fewer bits
  – Uses lossless color quantization and lossless LZW compression

• Portable Network Graphics (PNG)
  – Developed as a replacement for GIF because of licensing issues
  – Supports three different types of images
    1. True color, with up to $3 \times 16$ bpp
    2. Grayscale, with up to 16 bpp
    3. Indexed, with up to 256 colors
  – Also may include an $\alpha$-channel for transparency with a maximum width of 16 bits
    * $\alpha$-channel of a GIF image is only 1 bit
  – Supports only one image per file, with maximum size as $2^{30} \times 2^{30}$ pixels
    * Cannot support animation like GIF
  – Supports lossless compression by a variation of PKZIP but no lossy compression

• Joint Photographic Experts Group (JPEG)
  – Goal to achieve average data reduction of 1:16
  – Supports images with up to 256 color components
Three steps in the core algorithm for RGB images

1. Color conversion and down sampling
   * Transform from RGB to YCbCr space; Y is brightness while the other two components are color
   * Human visual system is less sensitive to rapid color change; compress color components more to achieve significant data reduction without a perceptive change in image quality

2. Cosine transform and quantization in frequency space
   * Image is divided into a regular grid of 8 × 8 blocks
   * Compute frequency spectrum of each block using discrete cosine transform
   * The 64 spectral components of each block are quantized into a quantization table
   * Reduce high frequency components and recompute them during decompression

3. Lossless compression
   * Compress quantized spectral component data stream using arithmetic or Huffman encoding

Not a good choice for images such as line drawings