Random Experiments on Color Image Processing and Enhancement

Trần Minh Hiếu—BI9-101
Nguyễn Gia Phong—BI9-184
Nguyễn Văn Tùng—BI9-229
Nguyễn An Thiết—BI8-174
Nguyễn Thành Vinh—BI8-187

University of Science and Technology of Hà Nội

July 8, 2020

Contents

Humans and Colors

- Humans see the world in color
- Humans *love* colors!
- Color paintings, color pictures, color movies

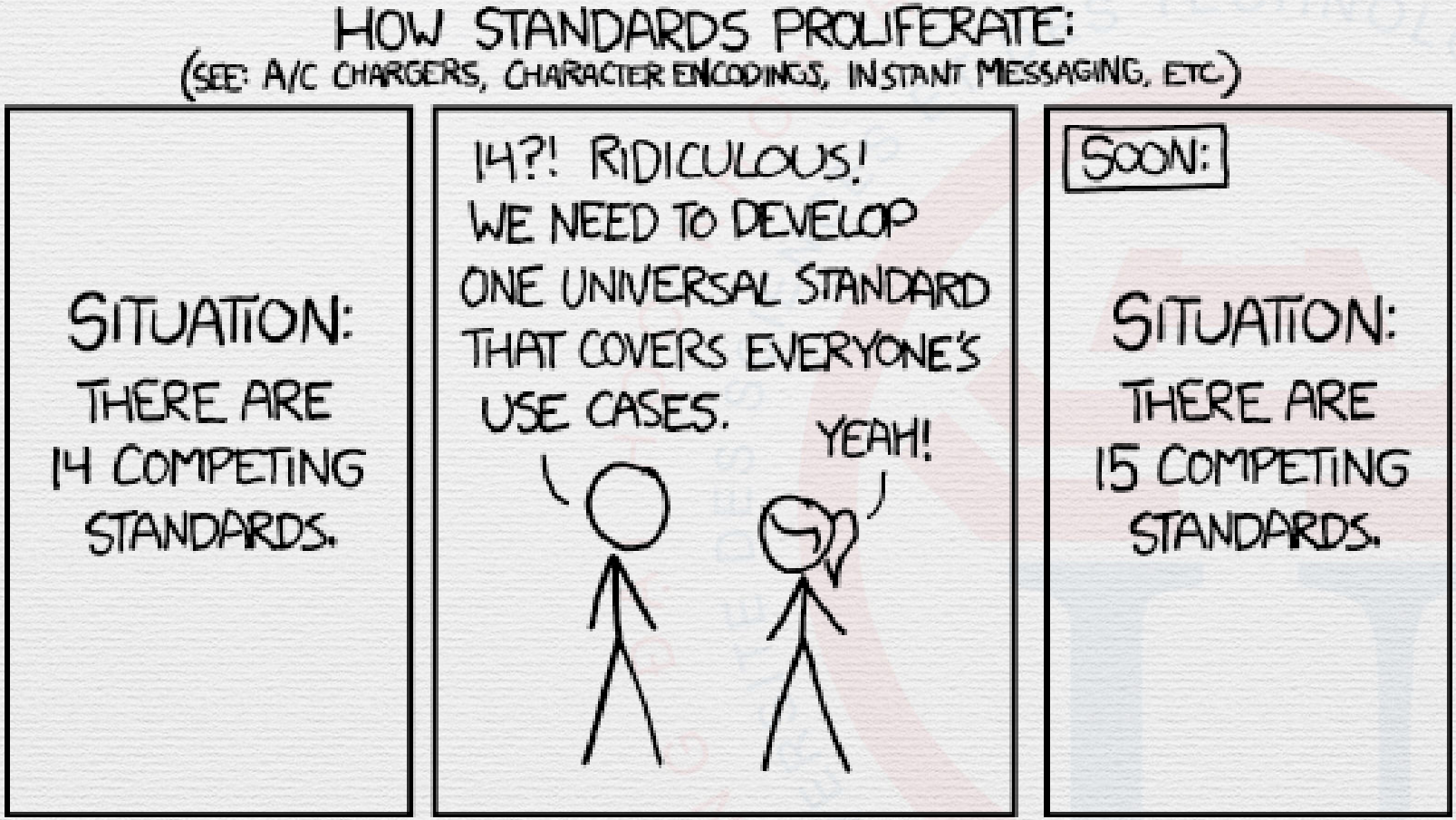
Colors Image Processing

- Most techniques similar to that of monochrome images
- There exists color-image-specific stuff too!
- And how computers represent colors?

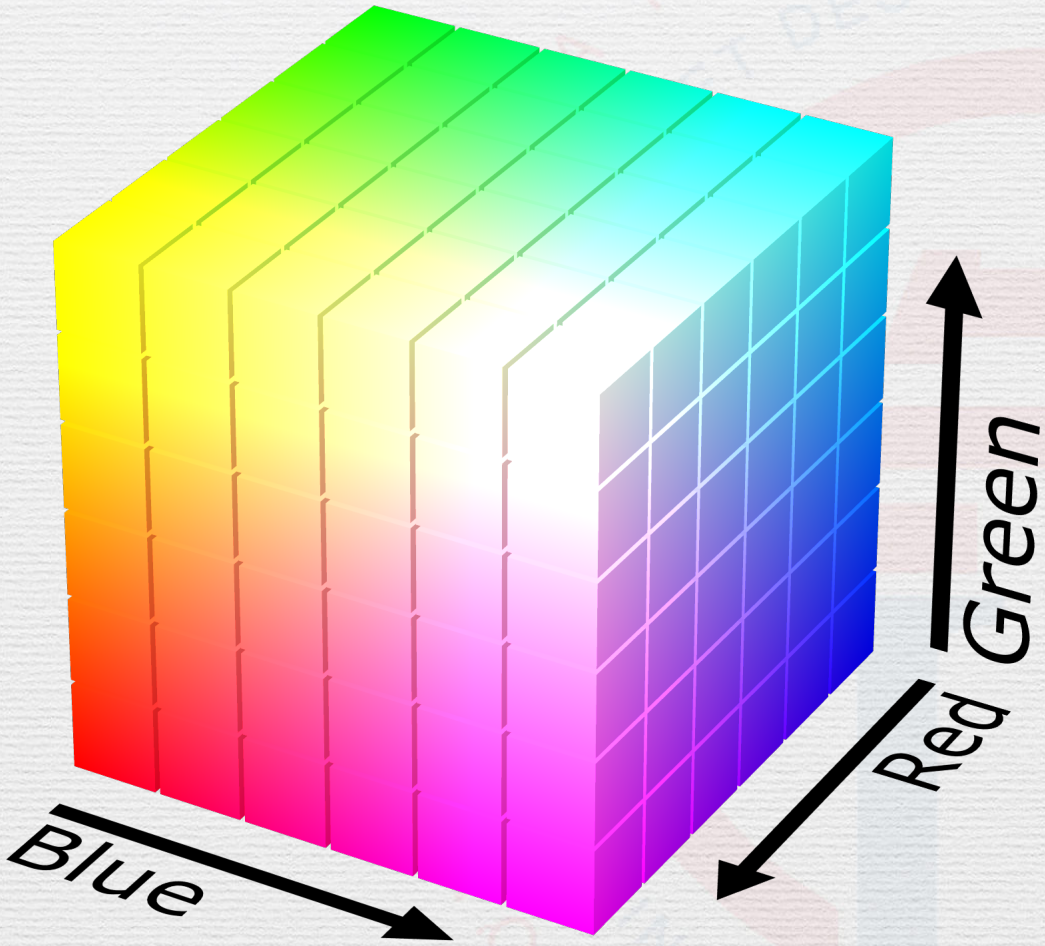
Why So Many?

- Each usage is needs a different color space
- Examples: RGB for computer display, CMYK for printing
- People try to standardize color specifications
- And people often don't agree, e.g. HSV vs HSB vs HSL

Relevant xkcd

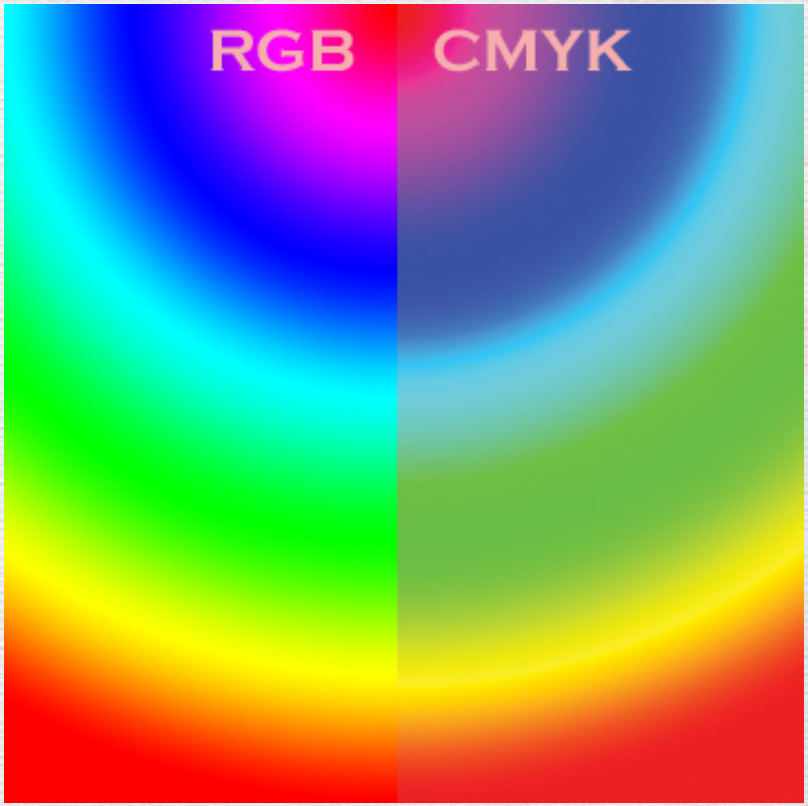


RGB

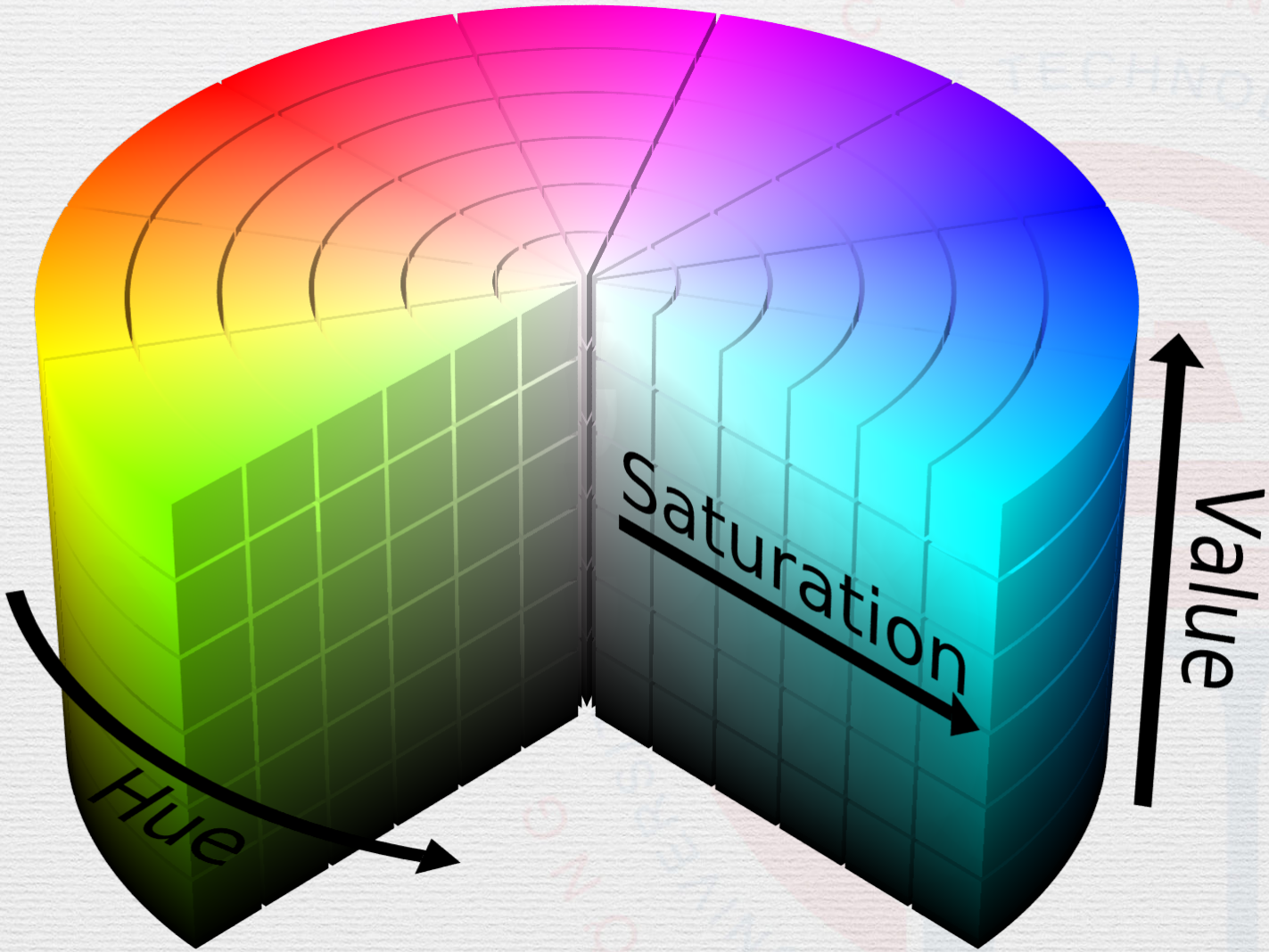


CMY(K)

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



HSV



BGR to HSV

```
def bgr_to_hsv(b, g, r):
    minc, maxc = min(r, g, b), max(r, g, b)
    if minc == maxc: return 0.0, 0.0, maxc
    diff = maxc - minc
    sat = diff / maxc
    rc, gc, bc = (maxc-r)/diff, (maxc-g)/diff, ...
    if r == maxc: return (bc-gc)/6%1, sat, maxc
    if g == maxc: return (2.0+rc-bc)/6%1, sat, maxc
    if b == maxc: return (4.0+gc-rc)/6%1, sat, maxc
```

HSV to BGR

```
def hsv_to_bgr(h, s, v):
    if s == 0.0: return v, v, v
    f = h * 6 % 1
    p = v * (1 - s)
    q = v * (1 - s*f)
    t = v * (1 - s*(1 - f))
    i = int(h*6%6)
    if i == 0: return p, t, v
    if i == 1: return p, v, q
    if i == 2: return t, v, p
    if i == 3: return v, q, p
    if i == 4: return v, p, t
    if i == 5: return q, p, v
```

Color Space Conversion

```
from itertools import starmap
from numpy import reshape, uint8

def convert_color(image, func):
    x, y, z = image.shape
    return reshape(
        list(starmap(func, reshape(image, (x*y,z)))),
        (x, y, z)).astype(uint8)
```

- `func` takes range 0–255 instead of 0–1
- For `@decorator` see source code
- Slow! Better stick to `cv2.cvtColor`

Chromatic Aberration

- Often seen in pictures taken by cheap lenses
- Modern game devs love it

Mitigation

- Mostly impossible to generalize (depend on lenses)
- Detection is easy though

Greyworld Balancing

Adjustments in CIE LAB color space

Implementation

```
def grayworld_assumption(img):
    result = cvtColor(img, COLOR_BGR2LAB)
    avg_a = average(result[:, :, 1])
    avg_b = average(result[:, :, 2])
    result[:, :, 1] = (
        result[:, :, 1]
        - (avg_a-128)*(result[:, :, 0]/255)*1.1)
    result[:, :, 2] = (
        result[:, :, 2]
        - (avg_b-128)*(result[:, :, 0]/255)*1.1)
    result = cvtColor(result, COLOR_LAB2BGR)
    return result
```

Pseudo Color Rendering

- Map grey intensity to color
- Get color image from greyscale image
- Enhance humans' visual intuition
- Common use: elevation, thermal, medical imaging

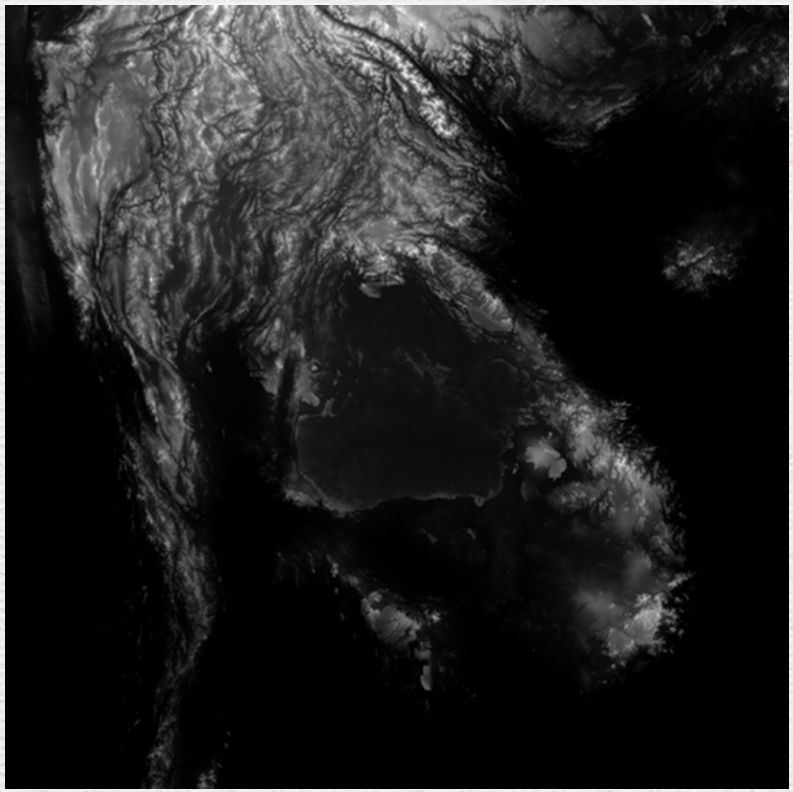
Implementation

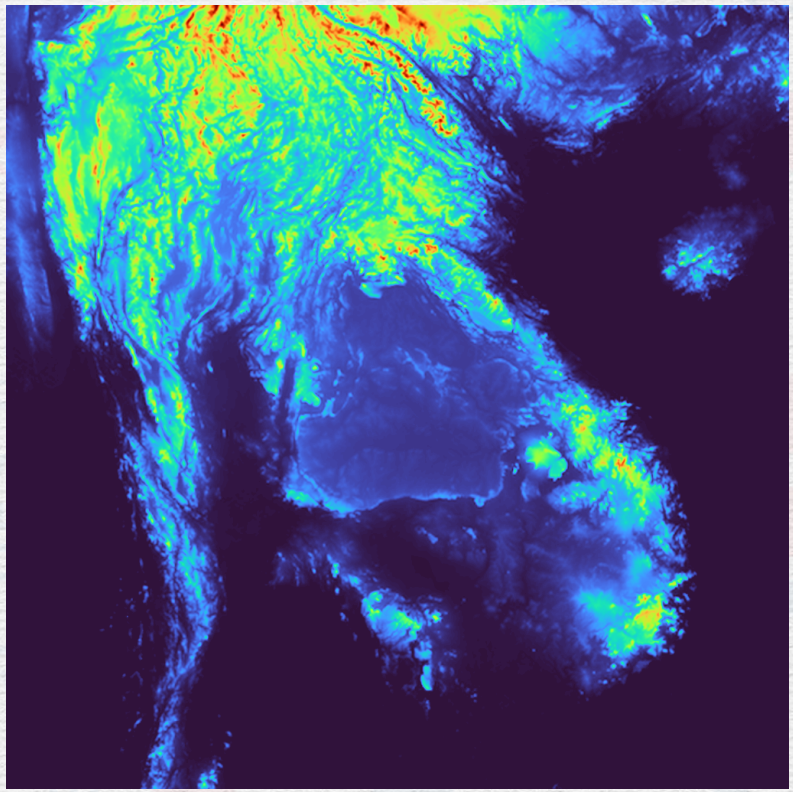
```
from numpy import stack, vectorize

def map_color(grey, mapping):
    r = vectorize(lambda i: mapping[i][0])
    g = vectorize(lambda i: mapping[i][1])
    b = vectorize(lambda i: mapping[i][2])
    return stack((b(grey), g(grey), r(grey)), -1)
```

cv2.applyColorMap is still faster!

Example: Elevation





Conclusion

- Colors are cool!
- Some amazing things can be achieved with only a few lines of code
- Pure Python is slow, Cython or C/C++ would be better in practice

Copying