Computer Vision Hw1 Color-to-Gray Conversion

電機四 B04901096 蔡昕宇

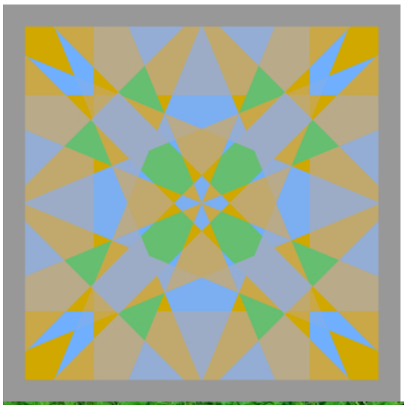
Conventional RGB2GRAY Conversion

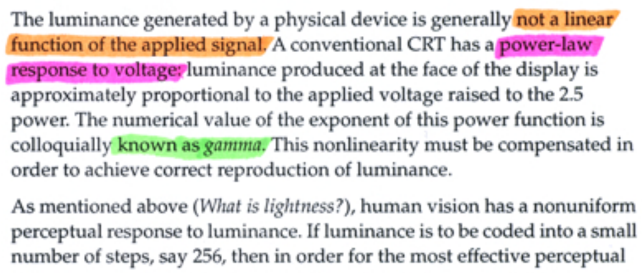
Simply apply dimension reduction[1] as follow:

$$Y = 0.299R + 0.587G + 0.114B$$

Results

Original RGB Image

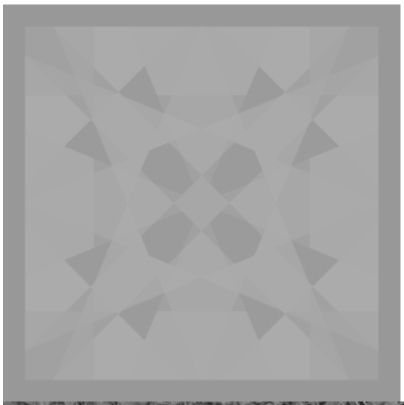


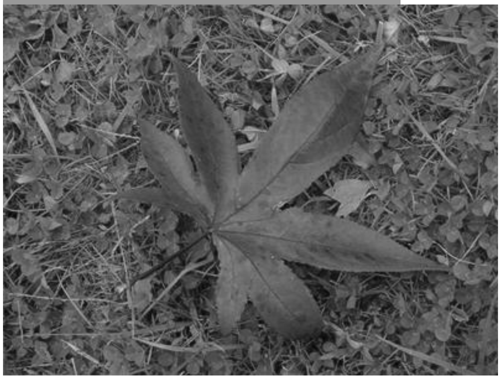


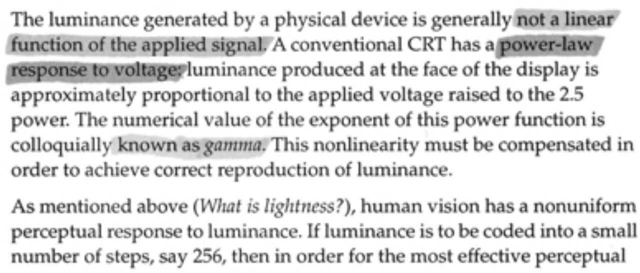
The luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power-law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the 2.5 power. The numerical value of the exponent of this power function is colloquially known as *gamma*. This nonlinearity must be compensated in order to achieve correct reproduction of luminance.

As mentioned above (*What is lightness?*), human vision has a nonuniform perceptual response to luminance. If luminance is to be coded into a small number of steps, say 256, then in order for the most effective perceptual

Gray Scale Image







The luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power-law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the 2.5 power. The numerical value of the exponent of this power function is colloquially known as *gamma*. This nonlinearity must be compensated in order to achieve correct reproduction of luminance.

As mentioned above (*What is lightness?*), human vision has a nonuniform perceptual response to luminance. If luminance is to be coded into a small number of steps, say 256, then in order for the most effective perceptual

Usage:

```
python3 color2gray.py --mode c -p -i [input image] -o [output directory]
```

Example:

```
python3 color2gray.py --mode c -p -i testdata/0a.png -o conventional
```

Converted images will be saved in a directory. In this case, it will be saved in the directory named as **conventional**.

[1] RGB2YUV, Wikipedia: https://en.wikipedia.org/wiki/YUV

Joint Bilateral Filter

Refer to Color Image Guided Bilateral Filter, given ***T*** as the guidance, the bilateral filter has the following formula:

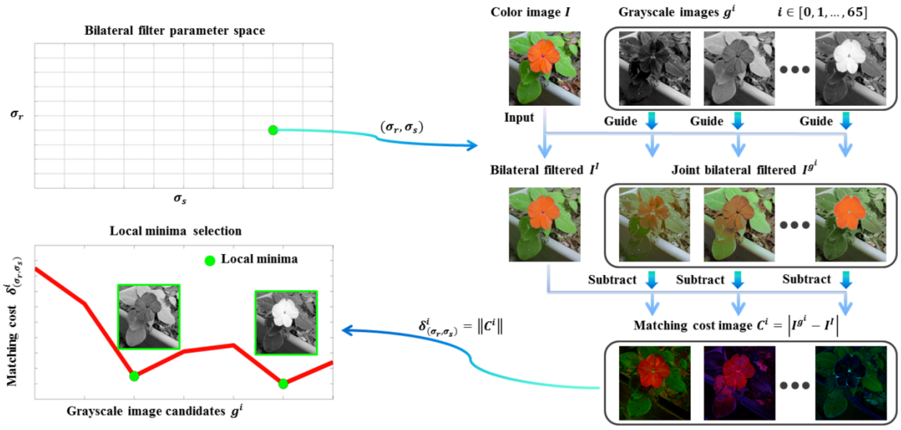
$$\frac{\sum_{q\in\Omega_p} G_s(p,q)G_r(T_p,T_q)I_q}{\sum_{q\in\Omega_p} G_s(p,q)G_r(T_p,T_q)}$$

for p, q is two different pixels' location in the guidance image.
In this task, ***T*** is a single-channel gray scale image. ***Gs*** and ***Gr*** will be:

$$G_s(p,q) = e^{-\frac{(p-q)^2}{2\sigma_s^2}} \qquad G_r(p,q) = e^{-\frac{(T_p-T_q)^2}{2\sigma_r^2}}$$

Advanced RGB2GRAY Conversion

According to the figure shown below from the paper ***Decolorization: Is rgb2gray() Out?***[2],


Bilateral filter parameter space
σ_r
σ_s
(σ_r, σ_s)
Color image I
Grayscale images g^i
$i \in [0, 1, \dots, 65]$
Input
Guide
Guide
Guide
Bilateral filtered I^I
Joint bilateral filtered I^{g^i}
Subtract
Subtract
Subtract
Matching cost image $C^i = |I^{g^i} - I^I|$
$\delta^i_{(\sigma_r,\sigma_s)} = \|C^i\|$
Local minima selection
Local minima
Matching cost $\delta^i_{(\sigma_r,\sigma_s)}$
Grayscale image candidates g^i

I followed these steps and implemented joint bilateral filter just as same as mentioned above.

Local Minimum Selection

The method is as same as that in the paper, which tells us to find the local minimum by calculating cost between bilateral filtered image I^I and joint bilateral filtered image I^{g^i}. There will be **66** weight combinations for gray conversion and **3** parameters for both σ_s and σ_r.

$$Y = w_r * R + w_g * G + w_b * B \qquad w_r, w_g, w_b \geq 0$$

such that $\quad w_r + w_g + w_b = 1 \qquad w \in \{0, 0.1, 0.2, \dots 1\}$

and $\quad \sigma_s \in \{1,2,3\} \qquad \sigma_r \in \{0.05, 0.1, 0.2\}$

The cost is calculated by **L1 distance** of two images, and compare that of each parameter.
The local minima corresponding to the weight w_r, w_g, w_b is determined if the cost is lower than its neighbor $w_r^{'}, w_g^{'}, w_b^{'}$ (i.e. the sum of the absolute distance of each dimension is equal to 0.2, so there are at most 6 neighbors for each), and it gets 1 vote. The corresponding weight can get get at most 9 votes. In this task, we keep the 3 most voted weight (maybe only 1 or 2).
I chose the window size: $2 * (3 * \sigma_s) + 1$ to do filtering, and complexity of filter computation is $O(n^2)$, for n is image side length.

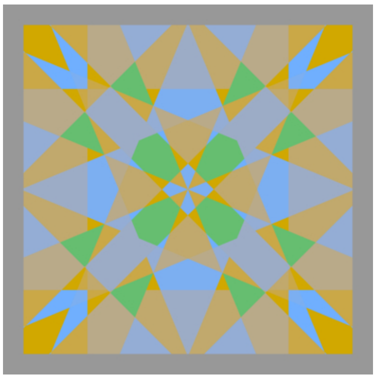
[2] Yibing Song, Linchao Bao, Xiaobin Xu, Qingxiong Yang, "*Decolorization: Is rgb2gray() Out?*", *SIGGRAPH Asia 2013 Technical Briefs — https://ybsong00.github.io/siga13tb/siga13tb_final.pdf*

Results

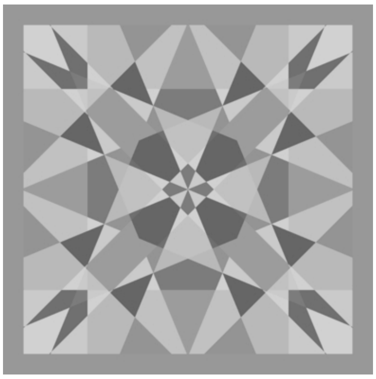
The following listed the 3 most voted image and the corresponding weight. (shown as w_r, w_g, w_b)

testdata/0a.png

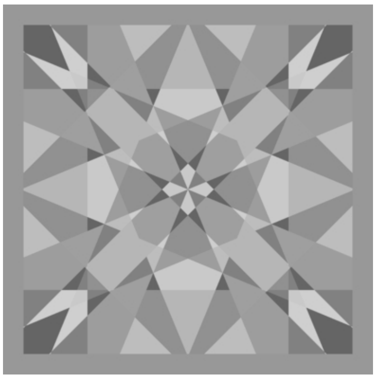
Original



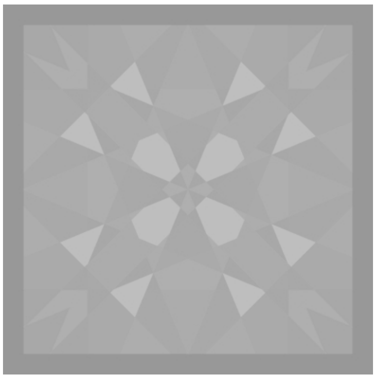
(1.0, 0.0, 0.0) 9 votes



(0.0, 0.6, 0.4) 6 votes

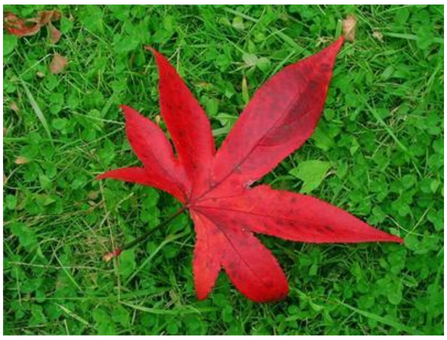


(0.0, 0.1, 0.0) 4 votes

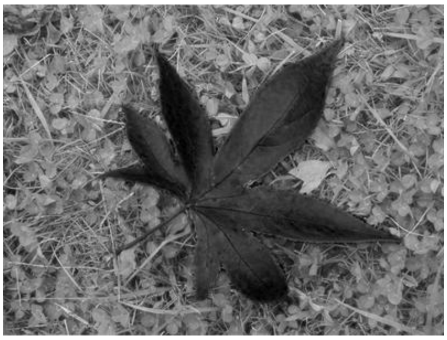


testdata/0b.png

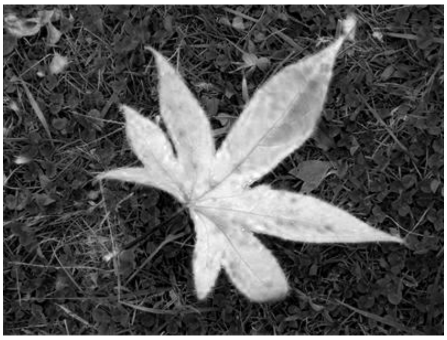
Original



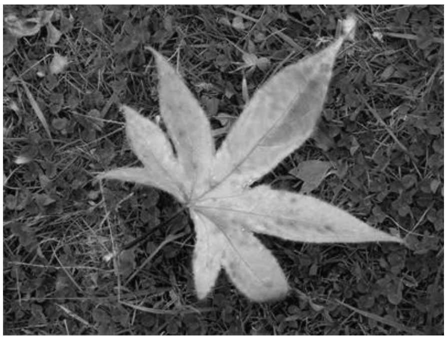
(0.0, 0.1, 0.0) 9 votes



(1.0, 0.0, 0.0) 4 votes

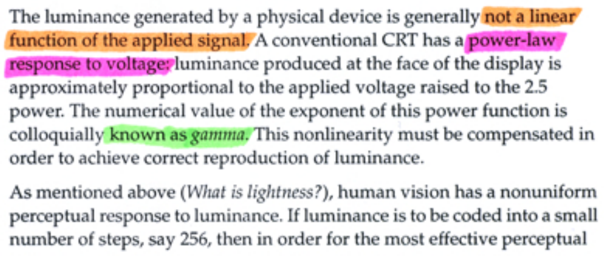


(0.8, 0.2, 0.0) 1 vote



testdata/0c.png

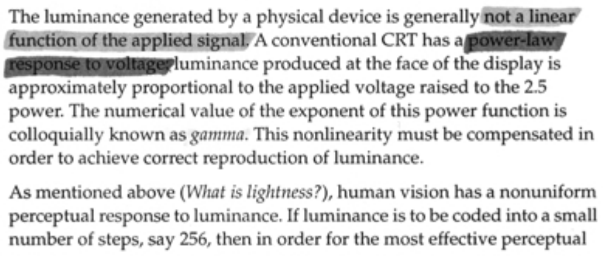
Original



The luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power-law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the 2.5 power. The numerical value of the exponent of this power function is colloquially known as *gamma*. This nonlinearity must be compensated in order to achieve correct reproduction of luminance.

As mentioned above (*What is lightness?*), human vision has a nonuniform perceptual response to luminance. If luminance is to be coded into a small number of steps, say 256, then in order for the most effective perceptual

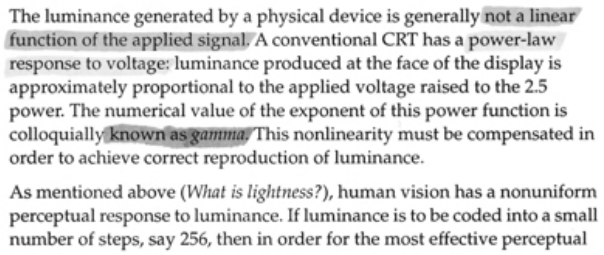
(0.0, 0.1, 0.0) 8 votes



The luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power-law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the 2.5 power. The numerical value of the exponent of this power function is colloquially known as *gamma*. This nonlinearity must be compensated in order to achieve correct reproduction of luminance.

As mentioned above (*What is lightness?*), human vision has a nonuniform perceptual response to luminance. If luminance is to be coded into a small number of steps, say 256, then in order for the most effective perceptual

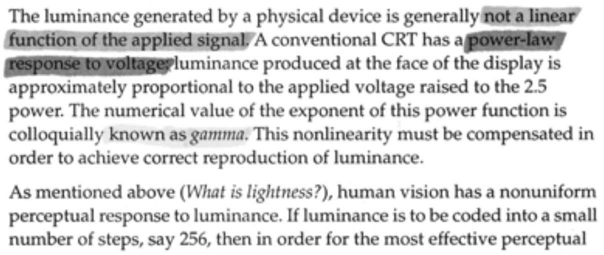
(0.6, 0.1, 0.3) 6 votes



The luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power-law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the 2.5 power. The numerical value of the exponent of this power function is colloquially known as *gamma*. This nonlinearity must be compensated in order to achieve correct reproduction of luminance.

As mentioned above (*What is lightness?*), human vision has a nonuniform perceptual response to luminance. If luminance is to be coded into a small number of steps, say 256, then in order for the most effective perceptual

(0.1, 0.8, 0.1) 2 votes

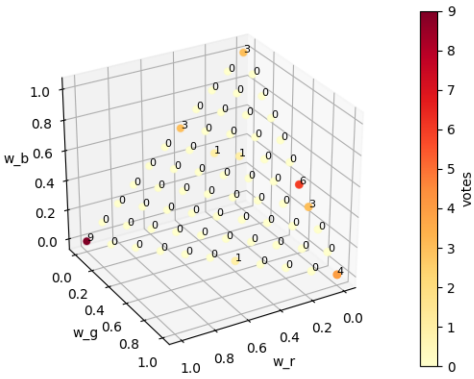


The luminance generated by a physical device is generally not a linear function of the applied signal. A conventional CRT has a power-law response to voltage: luminance produced at the face of the display is approximately proportional to the applied voltage raised to the 2.5 power. The numerical value of the exponent of this power function is colloquially known as *gamma*. This nonlinearity must be compensated in order to achieve correct reproduction of luminance.

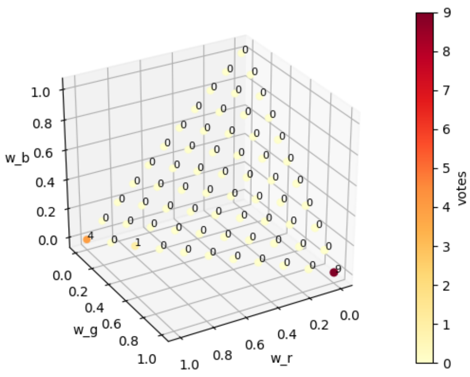
As mentioned above (*What is lightness?*), human vision has a nonuniform perceptual response to luminance. If luminance is to be coded into a small number of steps, say 256, then in order for the most effective perceptual

Voting Visualization

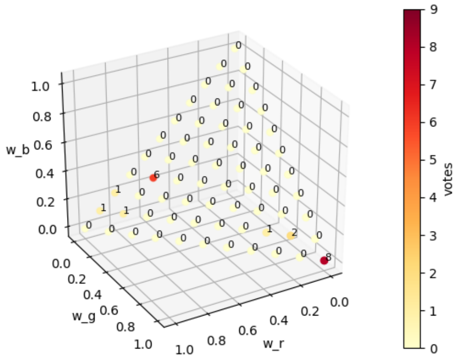
testdata/0a.png



testdata/0b.png



testdata/0c.png



Usage:

```
        python3 color2gray.py --mode a -p -i [input image] -o [output directory]
```

Example:

```
        python3 color2gray.py --mode a -p -i testdata/0a.png -o advanced
```

It will generate all bilateral filtered image for all weight combinations in corresponding directory, and also outputs a directory called **result** in the output directory, which includes converted gray scale images.

Requirement:

Python	3.6.5
Matplotlib	3.0.0
Opencv-contrib-python	3.4.0.12
Numpy	1.14.5