レポート3
08-172022
永田 諒

gammaXYZ.py

実行結果：

コード：

```
import cv2
from grayscale import readImage
from gammaCV import gammaLUT

winRes = 'Result'
tracks = ('X', 'Y', 'Z')


def changeGamma(val):
    global img, lookUpTable
    imgXYZ = cv2.cvtColor(img, cv2.COLOR_BGR2XYZ)
    planes = cv2.split(imgXYZ)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgXYZ = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgXYZ, cv2.COLOR_XYZ2BGR))


def main():
    global img, lookUpTable
    img = readImage()
    trackHalf = 128
    trackMax = trackHalf * 2
    lookUpTable = gammaLUT(trackHalf, trackMax)
    cv2.namedWindow(winRes)
    for i in range(len(tracks)):
        cv2.createTrackbar(tracks[i], winRes, 0, trackMax,
changeGamma)
        cv2.setTrackbarPos(tracks[i], winRes, trackHalf)

    changeGamma(trackHalf)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


if __name__ == '__main__':
    main()
```

考察：
そもそもガンマ変換は何を目的にしたものであるのかがわからなかったので調べた。
人間の光受容体は刺激の物理強度にリニアではなく、物理強度の対数にフィットする。
これによって、人間の視覚は、低輝度の変化に敏感で、高輝度には鈍感である。
よって、リニア輝度でRGBの値を表現するとデジタル値の処理に付き纏う量子化誤差が低い輝度ほど目立つが、ガンマ補正を行うと、その傾向が薄れる。

つまり今回ガンマ変換を行なっている理由は、人間にわかりやすいようにパラメータの変化を行わせるためである。

さて今回のコードに関して言うと、トラックバーでパラメーターの変更が行われるたびに、色空間の更新が行われている。

いまいちわからないのはここ
一行ずつ、処理の流れを追ってみた。

```
    planes = cv2.split(imgXYZ)
    for i in range(len(tracks)):
       val = cv2.getTrackbarPos(tracks[i], winRes)
       planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgXYZ = cv2.merge(planes)
```

一番最初のsplitではマルチチャネルの配列をシングルチャネルの配列に分割している。
この操作により、X,Y,Zそれぞれのパラメーターを扱うことができる。
次にcv2.LUTでは、ルックアップテーブルを用いて、配列を変換している。
最後にmergeを使うことで、シングルチャネルに分けていた、配列をマルチチャネルに戻している。
以上の流れで、トラックバーからの入力を使って階調変換を行なっている。

sepia.py

コード：

```
import cv2
import numpy as np
from grayscale import readImage


def main():
    img = readImage()
    cv2.imshow('Source', img)
    default_h = 22
    default_s = 150
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    planes = cv2.split(imgHSV)
    planes[0] = np.ones(planes[0].shape[:2], np.uint8) * default_h
    planes[1] = np.ones(planes[1].shape[:2], np.uint8) * default_s
    imgHSV = cv2.merge(planes)
    cv2.imshow('Sepia', cv2.cvtColor(imgHSV, cv2.COLOR_HSV2BGR))
```

```
    cv2.waitKey(0)
    cv2.destroyAllWindows()


if __name__ == '__main__':
    main()
```

考察：
ここではセピア画像を生成している。
画像の色をHSV表現に変換したのち、デフォルトのHueとSatuationの値で固定することでセピア画像に変換している。

HSVは
H: 色相
S: 彩度
V: 明度
で表現される。この実装から、セピア画像は色相と彩度が、固定された画像だと考えることができる。

コード

RGB

```
def changeGamma(val):
    global img, lookUpTable
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    planes = cv2.split(imgRGB)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgRGB = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgRGB, cv2.COLOR_RGB2BGR))
```

HLS

```
def changeGamma(val):
    global img, lookUpTable
    imgHLS = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
    planes = cv2.split(imgHLS)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgHLS = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgHLS, cv2.COLOR_HLS2BGR))
```

HSV

```
def changeGamma(val):
    global img, lookUpTable
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    planes = cv2.split(imgHSV)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgHSV = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgHSV, cv2.COLOR_HSV2BGR))
```

CIE L*a*b*

```
def changeGamma(val):
    global img, lookUpTable
    imgLab = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)
```

```
    planes = cv2.split(imgLab)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgLab = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgLab, cv2.COLOR_Lab2BGR))
```

CIE L*u*v*

```
def changeGamma(val):
    global img, lookUpTable
    imgLuv = cv2.cvtColor(img, cv2.COLOR_BGR2Luv)
    planes = cv2.split(imgLuv)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgLuv = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgLuv, cv2.COLOR_Luv2BGR))
```

YCrCb

```
def changeGamma(val):
    global img, lookUpTable
    imgYCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
    planes = cv2.split(imgYCrCb)
    for i in range(len(tracks)):
        val = cv2.getTrackbarPos(tracks[i], winRes)
        planes[i] = cv2.LUT(planes[i], lookUpTable[val])
    imgYCrCb = cv2.merge(planes)
    cv2.imshow(winRes, cv2.cvtColor(imgYCrCb,
cv2.COLOR_YCrCb2BGR))
```

考察：
コードはほぼ同じなので、それぞれの意味を簡単にまとめる

RGB:
光の三原色を利用した表現方法、加法混合を表すのに使われる。

HLS:
H => 色相、L=>輝度、S=>彩度からなる。
HSVに似ているが、違いは、その値の算出方法である。

HSV:
例題で述べたので省略

CIE L*a*b*:
XYZから、知覚と装置の違いによる色差を表現するために考案された。

CIE L*u*v*:
XYZ表色系のxy色度図の波長感覚の均等性を改善したもの。
> https://ja.wikipedia.org/wiki/%E8%89%B2%E7%A9%BA%E9%96%93#RGB

CIE XYZ:
RGBで、負の値の概念を導入しないと表現できない色を表現できる。
原色ではなく、混色を使って色を表現する。

YCrCb:
アナログビデオ等で用いられる色の表現方法で、輝度信号Yと2つの色差信号を使って表現される色空間。