Find the correlation coefficients between color components in the images

```
for i,file in enumerate(file_list):
bgr = cv2.imread(path+file)
img = cv2.cvtColor(bgr,cv2.COLOR BGR2RGB)
img_list[i]=img
a,b,c = cv2.split(img_list[0])
```

우선, Opencv를 이용하여 이미지를 불러오면 rgb가 아닌 bgr로 불러들이기 때문에 cvtcolor를 이용하여 rgb파일로 전환시켜서 이미지를 저장했습니다.

correlation을 구하는 공식은 다음과 같습니다.

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

따라서 함수를 정의하여 각각의 채널별 계수를 구하였습니다.

<correlation coefficient>

```
def correl(x,y):
meanx = np.mean(x)
meany = np.mean(y)
stdx = np.std(x)
stdy = np.std(y)
ret = (np.mean((x-meanx)*(y-meany)))/(stdx*stdy)
return ret
```

<RGB2YUV>

```
def RGB2YUV(rgb):
m = np.array([[0.29900 ,-0.14713,0.61500],
              [0.58700,-0.28886,-0.51499],
              [0.11400,0.43600,-0.10001]])
yuv = np.dot(rgb, m)
yuv[:,:,1:] += 128.0
return yuv
```

식이 맞는지 확인하기 위해 Opencv의 cvtColor의 option 중 cv2.COLOR_RGB2YUV의 것과 비슷한 지 확인해 보겠습니다.

```
print(RGB2YUV(img_list[0]))
print(cv2.cvtColor(img_list[0],cv2.COLOR_RGB2YUV))
```

```
[[[ 54.093   174.21235 165.64366]
  [ 54.093   174.21235 165.64366]
  [ 53.093   174.21234 165.64366]
```

[그림1] 구현한 RGB2YUV

```
[[[ 54 174 166]
  [ 54 174 166]
  [ 53 174 166]
```

[그림2] Opencv의 RGB2YUV

```
while count < 10:
r_b_co[count] = correl(red_list[count][:,:,0],blue_list[count][:,:,2])
r_g_co[count] = correl(red_list[count][:,:,0],green_list[count][:,:,1])
b_g_co[count] = correl(blue_list[count][:,:,2],green_list[count][:,:,1])
y_u_co[count] = correl(y_list[count],u_list[count])
y_v_co[count] = correl(y_list[count],v_list[count])
u_v_co[count] = correl(u_list[count],v_list[count])
count +=1
```

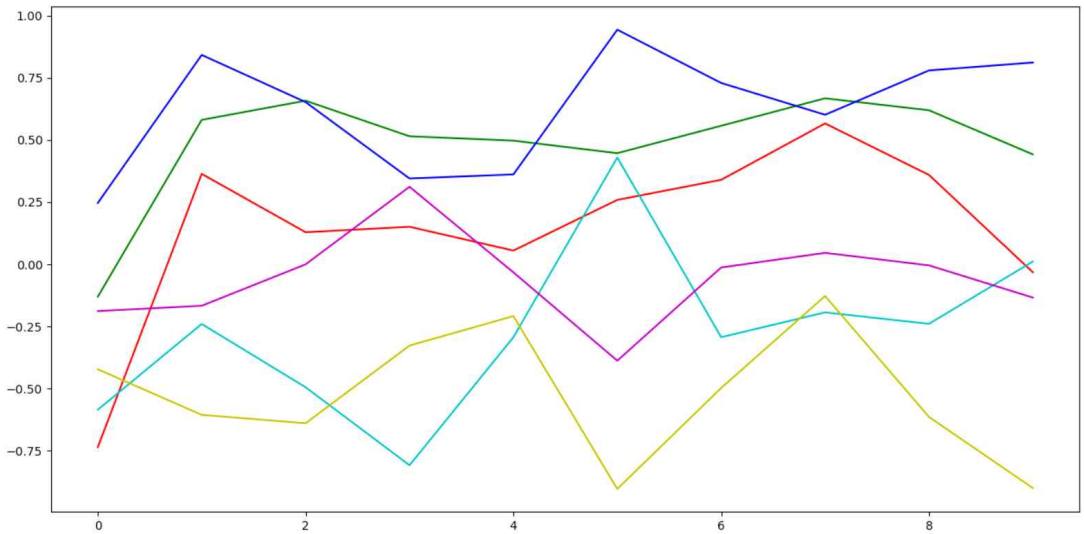
이 때, 각각의 이미지들은 모두 0,1,2번 index에 rgb채널을 가지고 있지만 자신의 채널을 제외하고 값이 0인 배열입니다. 따라서, red = 0, green = 1, blue = 2번 index만의 값을 가져와야 합니다.

다음은 각각의 이미지의 6가지 계수 모음입니다. 순서대로, red-blue, red-green, blue-green, y-u, y-v, u-v 상관계수들입니다.

```
-0.7357781153492424  -0.1305399333867224  0.24660384906527497  -0.5850065500711605  -0.18827576133723478  -0.4222914945077118
0.36363754407571225  0.5807066462529984  0.8420201453109005  -0.24013408809920914  -0.16692532231022253  -0.6057769117707337
0.12888853975804  0.6578370986368421  0.6522033241776436  -0.4947358864994259  -0.00028759121312053654  -0.6394400329390403
0.15076855541349118  0.5147989221612624  0.34506982399991276  -0.8083970363811266  0.31162707216909336  -0.3270449519997391
0.055119519848963267  0.4973051872654821  0.3613876750878166  -0.29360259184580567  -0.03243387732966827  -0.20845815809342794
0.2588625941413041  0.4467774281131026  0.9434619683639068  0.4294280194646464  -0.3879576110265869  -0.9037974288630352
0.33971375817934235  0.5570999193299784  0.7292478364387999  -0.2934298505704567  -0.01294683524952063  -0.4975603910095205
0.5664290712125133  0.6675439211587233  0.6013064129713256  -0.1935810055605036  0.045882705636068125  -0.1276559294092402
0.3594483447860317  0.6194553011345223  0.779658977938798  -0.2395212155559363  -0.004427792747205457  -0.6140587793883159
-0.032287043053545864  0.4419933052988414  0.8114087909553618  0.011158780378839215  -0.13397038745985929  -0.9003066485650079
```

다음은, correlation coefficients들을 그래프로 나타내어 분포를 확인해 보겠습니다.

```
plt.plot(r_b_co,'r')
plt.plot(r_g_co,'g')
plt.plot(b_g_co,'b')
plt.plot(y_u_co,'c')
plt.plot(y_v_co,'m')
plt.plot(u_v_co,'y')
```




다음과 RGB채널들 사이의 상관계수는 대체로 양수인 것을 알 수 있으며 YUV채널들 사이의 상관계수는 대체로 음수인 것을 알 수 있습니다. YUV채널의 절대값이 작고, RGB채널의 절대값이 클 것으로 상관계수를 예측하였으나 실제로는 각각의 이미지마다 많은 차이가 있음을 알 수 있었습니다.