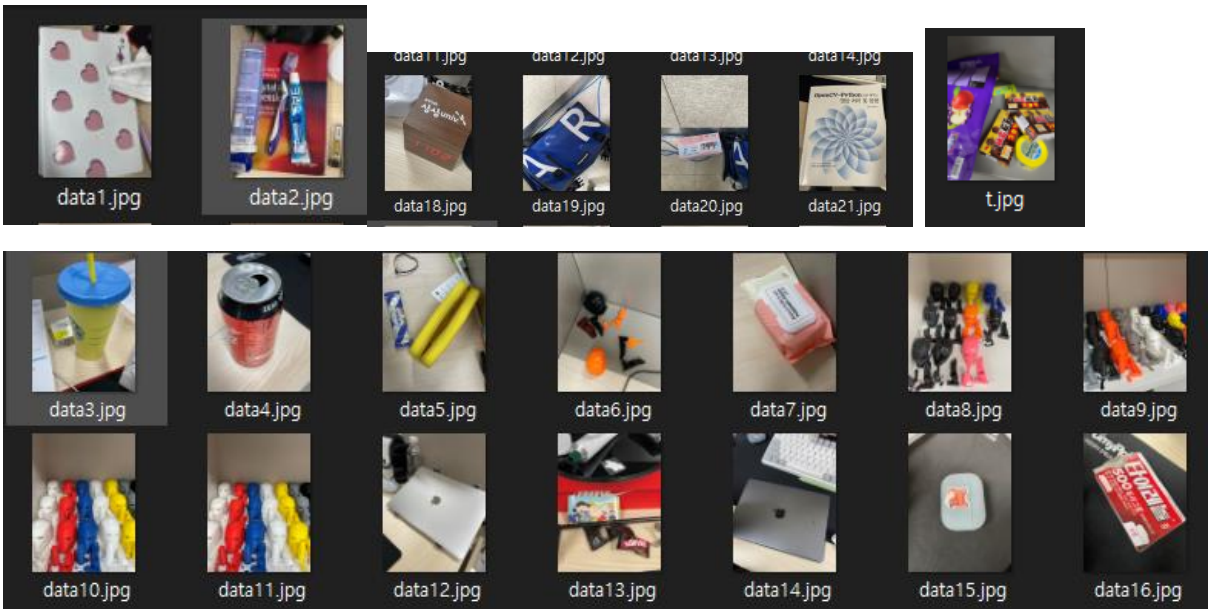
프로젝트 3. 카메라 출처 식별 및 위변조 식별 프로젝트

20195179 서원진

1. 디지털 카메라 (DSLR, 미러리스, 아이폰 등) 2~3대 선정 후 데이터셋 수집



제 핸드폰 아이폰 12 Pro 로 총 20장의 image를 Finger print dataset 로 사용했습니다.

색상 변조 사진은 동일한 기기로 촬영한 이미지입니다.

색상 변조 사진에 사용 한 사진

2. 구현 상세 (코드 핵심부분 설명)

-논문 프로젝트 구동

```
RP,_,_ = gF.getFingerprint(Images)
RP = Fu.rgb2gray1(RP)
sigmaRP = np.std(RP)
Fingerprint = Fu.WienerInDFT(RP, sigmaRP)
```

gF.getFingerprint()

Fingerprint 함수를 통해 카메라의 지문 reffrence patten을 얻습니다.

RP 를grayscale로 변환하고 표준편차를 구해 WiernerInDFT 함수에 넣습니다.

Fu.WinerInDFT

wavelet기반 de-noising을 해주는 필터로 노이즈를 제거합니다.

```
imx = './images/img10.jpg'
Noisex = Ft.NoiseExtractFromImage(imx, sigma=2.)
Noisex = Fu.WienerInDFT(Noisex, np.std(Noisex))  #주파수 영역의 입

# The optimal detector (see publication "Large Scale Test of Sen
Ix = cv.cvtColor(cv.imread(imx),# image in BGR format
                 cv.COLOR_BGR2GRAY)
#
C = Fu.crosscorr(Noisex,np.multiply(Ix, Fingerprint))
det, det0 = md.PCE(C)
for key in det.keys():
    print("{0}: {1}".format(key,det[key]))


eu.mesh(C)
```

Ft.NoiseExtractFromImage

변조 된 지 안 된 지 확인 할 해당 image의 noise(PRNU)를 뽑습니다 .

위에서 사용한 WinerInDFT 함수를 통해 denoising 합니다

Fu.crosscorr

image를 detect하기 위해 correlation을 사용합니다.

ix를 grayscale로 변환하고 노이즈와 와 아까 얻은 Fingerprint를 곱하여 계산한 값과 noisex를 비교합니다.

md.PCE

계산한 C를 통해 상관관계(PCE)를 계산하고 PFA, log10p_FA등을 딕셔너리 형태로 리턴합니다.

-논문 구현

```
Images = [im6,im7,im8,im9,im10,im11,im12,im13,im14,im15,im17,im18,im19,im20,im21,im22,im23,im24,im25,im26]

RP,_,_ = gF.getFingerprint(Images)  #Fingerprint 가 이미지 쉐입만큼 나온다.
print(RP.shape)
RP1 ,RP2,RP3=RP[:,:,0],RP[:,:,1],RP[:,:,2],
sigmaRP1, sigmaRP2, sigmaRP3 = np.std(RP1),np.std(RP2),np.std(RP3)
```

```
sigmaRP=np.array([sigmaRP1,sigmaRP2,sigmaRP3])
Fingerprint=np.array([Fingerprint1,Fingerprint2,Fingerprint3])

sio.savemat('Fingerprint.mat', {'RP': RP, 'sigmaRP': sigmaRP, 'Fingerprint': Fingerprint})

mat_file = sio.loadmat("Fingerprint.mat")
Fingerprint= mat_file["Fingerprint"]
```

img 의 reffrence patten fingerprint를 구합니다.

RP를 채널별로 표준편차를 구합니다 (sigma RP1 RP2 RP3)

Fingerpint를 RGB 채널별로 나눠서 Fingerprint1,Fingerprint2 ,Fingerprint3을 저장합니다.

mat 파일을 저장하기 위해 채널별로 나눈 RP sigmaRP Fingerprint를 각각 numpy array처리 해준 후 savemat합니다.

아까 저장한 mat file을 로드하고 matfile의 Fingerprint만 값만 가져와 저장합니다.

```
def huechange(img, shift):
    #plt.imshow(img)
    #plt.show()
    hsv = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    #print(hsv[:, :, 0])
    hsv[:, :, 0]=np.where(hsv[:, :, 0]>=180-shift,hsv[:, :, 0] -180+shift,hsv[:, :, 0] +shift)
    #print(hsv[:, :, 0])
    hnew = cv2.cvtColor(hsv, cv2.COLOR_HSV2RGB)
    # plt.imshow(hnew)
    # plt.show()
    return hnew
```

색상을 변조하기위한 함수입니다.

img RGB를 HSV로 변환하고 HSV의 색상 부분 hsv[:,:,0] 값만 바꾸기 위해 범위를 저렇게 정했습니다.

확인해보니 shift하면 0~256 사이의 값으로 들어가는 문제가 있었습니다.

np.where함수를 통해 hsv[:,:,0] 값 중 >180 에서 shift한 값을 빼 준 값이 크면

True 작으면 FALSE로 shift할 때 True는 -180을 빼 주고 shift FALSE는 안 빼 주고 shift합니다.

처음 원본의 hsv값은 180보다 큰 값이 없으나 shift 하고나서 opencv에선 hue가 0~180 값이 들어가야 하므로 조건을 나눠 shift합니다.

```python
def Corr(im):
    result = []
    for i in tqdm(range(0, 181,20)):
        # hnew[:,:,0]+=i
        #im = cv2.cvtColor(im, cv2.COLOR_HSV2BGR)
        # hnew1 = cv2.cvtColor(hnew1, cv2.COLOR_HSV2BGR)
        hnew1 = huechange(im, i)
        print("################### ",i ,"Shift ######################")
        Noisex = Ft.NoiseExtractFromImage(hnew1, sigma=2.)
        PCESUM = 0
        for j in range(3):
            Noisex2 = Fu.WienerInDFT(Noisex[:,:,j], np.std(Noisex[:,:,j]))
            C = Fu.crosscorr(Noisex2,np.multiply(hnew1[:,:,j],Fingerprint[j]))
            det, det0 = md.PCE(C)
            PCESUM += (det["PCE"])
        result.append(PCESUM)

    return result
```

result 빈 배열을 만들고 open cv는 hue범위가 0~180 이므로 for문을 180까지 하였고 20도씩 더하여 (20,40,80 …180) shift합니다.

hue change 함수를 통해 이미지를 20도씩 shift 하여 나온 hnew1의 노이즈를 뽑고

채널별로 비교해야 하므로 for문을 3번돌려 denoising 하여 나온 noisex2랑

shift한 이미지와 Fingerprint 곱한 거의 관계를 비교합니다.

비교하여 나온 관계를 PCE함수를 통해 상관관계를 계산합니다.

det에 저장된 PCE값을 각도 돌릴 때 마다 RGB별 PCE를 합하여 result 배열에 append합니다

즉 각도를 8번 돌리면 8개의 PCE의 합이 추가됩니다.

```python
img = cv2.imread('./data/je.jpg')
im = img[:,:,::-1]
# im=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

gak=80
hhh=huechange(im,gak)

result=Corr(hhh)

idx=np.argmax(np.array(result))
print("############detect###########")
print(idx)
if idx == 0:
    print("no hue modified")
else:
    print("modified")

changeimg = huechange(hhh, idx*20)

plt.subplot(1, 3, 1), plt.title("original"), plt.imshow(im)
plt.subplot(1, 3, 2), plt.title("shift"), plt.imshow(hhh)
plt.subplot(1, 3, 3), plt.title("foundimage"), plt.imshow(changeimg)
plt.show()
```

동일한 카메라에서 얻은 임의의 이미지를 RGB로 변환하여 huechange함수를 통해 색상을 변경한 후에 Corr 함수에 넣습니다. 위에 Corr함수에서 리턴한 result 배열 (각도 당 PCE의 합)중 가장 큰값의 인덱스를 idx로 저장하고 idx가 0이면 원본이므로 no hue modified를 출력해주고

0이 아니면 modified라고 출력했습니다.

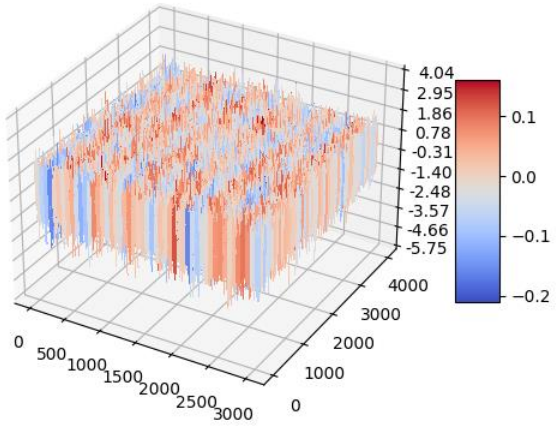
이제 변경정도 추정한 값을 통해 이미지를 확인하기위해

idx 에 20 을 곱하여 변경정도를 잘 추정해서 원본이랑 같은 지 확인합니다.

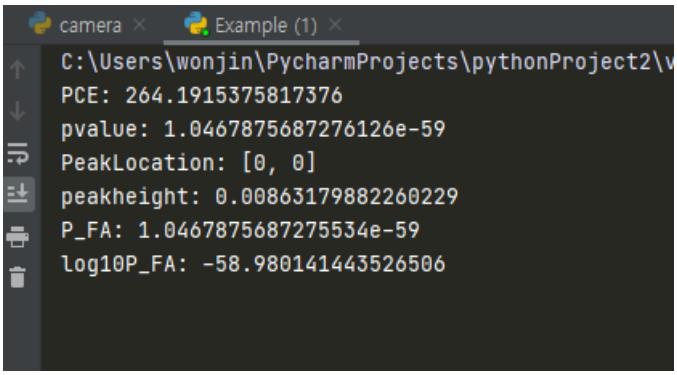
original (원본) shift(shift 한 이미지) foundimage(변경 정도 추정 이미지)

3. 실험 결과 보이기

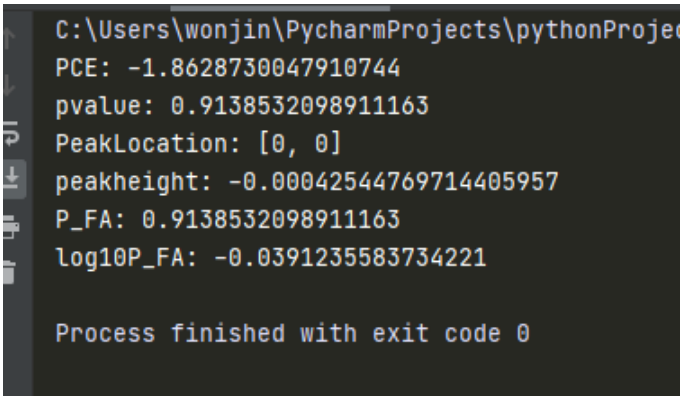
- reference pattern 생성, 패턴 보이기



- 수집한 DB를 기반으로 source camera identification 성능 확인

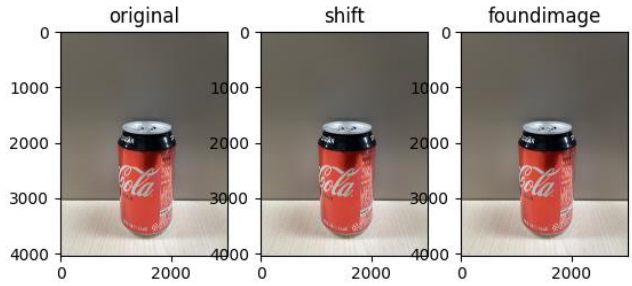


수집한 DB를 기반으로 동일기기 사진을 넣었을 때 source camera identification 성능



수집한 DB를 기반으로 다른 기기 사진을 넣었을 때 source camera identification 성능

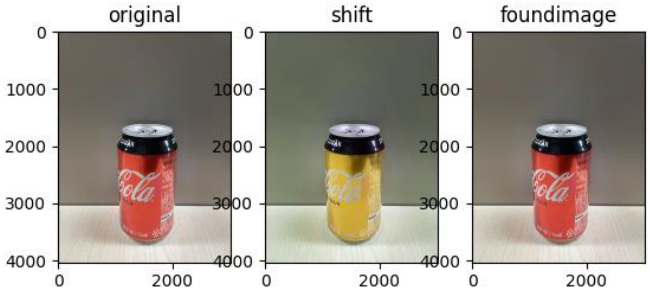
- 동일한 카메라에서 얻은 임의의 이미지의 색상을 변경한 후, 변경 정도를 추정한 값을 출력



변경 값 0도

```
100%|          | 10/10 [06:30<00:
############detect###########
PCESUM MAX INDEX : 0
no hue modified
```

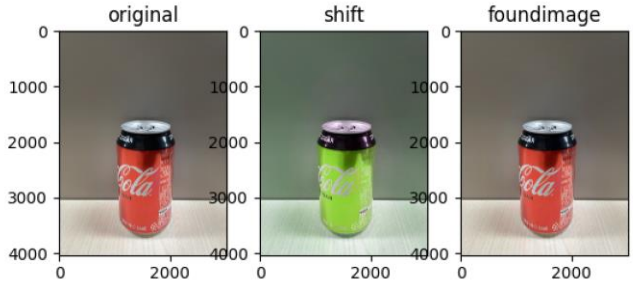
추정한 값 : 0도 PCESUM MAX (index) * 20 =0



변경 값 20도

```
############detect###########
PCESUM MAX INDEX : 8
modified
```

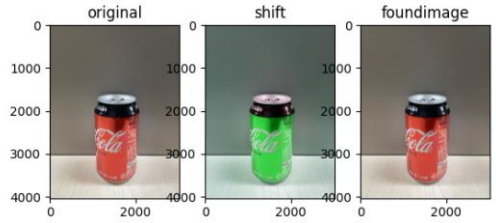
추정한 값 160도 PCESUM MAX (index) * 20 =160



변경 값 : 40

```
############detect###########
PCESUM MAX INDEX : 7
modified
```

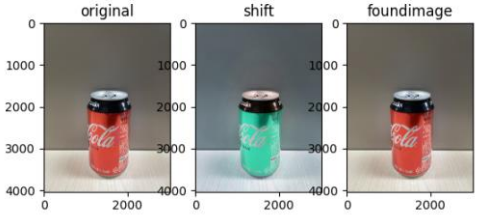
추정 값: 140도 PCESUM MAX Index * 20 =160



변경 값 :60도

```
############detect###########
PCESUM MAX INDEX : 6
modified
```

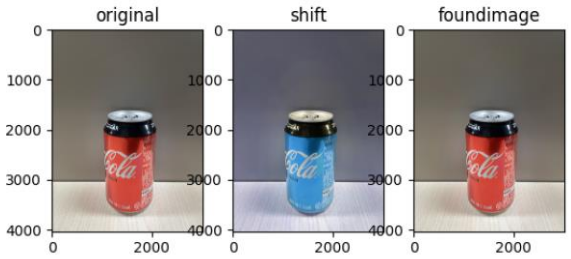
추정한 값 : 120도 PCESUM MAX Index * 20 =120



변경 값 : 80 도

```
############detect###########
PCESUM MAX INDEX : 5
modified
```

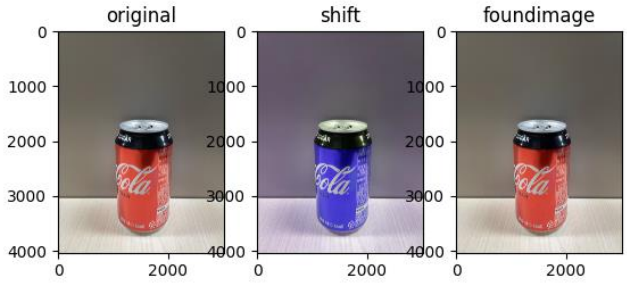
추정한 값 : 100도 PCESUM MAX Index * 20 =100



변경 값 : 100도

```
############detect###########
PCESUM MAX INDEX : 4
modified
```

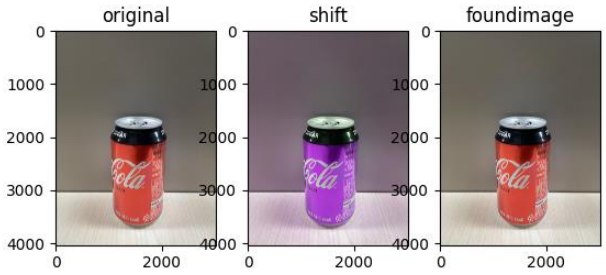
추정한 값 : 80도 PCESUM MAX Index * 20 =80



변경 값 120도

```
############detect############
PCESUM MAX INDEX : 3
modified
```

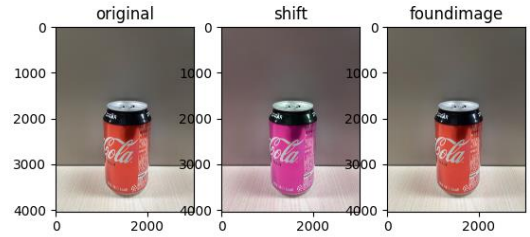
추정한 값 : 60도 PCESUM MAX Index * 20 =60



변경 값 140도

```
############detect############
PCESUM MAX INDEX : 2
modified
```

추정한 값 : 40도 PCESUM MAX Index * 20 =40



변경 값 160도

```
############detect############
PCESUM MAX INDEX : 1
modified
```

추정한 값 20도 PCESUM MAX Index *20 =20