메디치소프트 기술연구소

2018.08.07

CH08. OpenCV: 도형 그리기

8.이미지 연산 처리를 이용한 이미지 합성 하기

- 8.1 테스트 환경

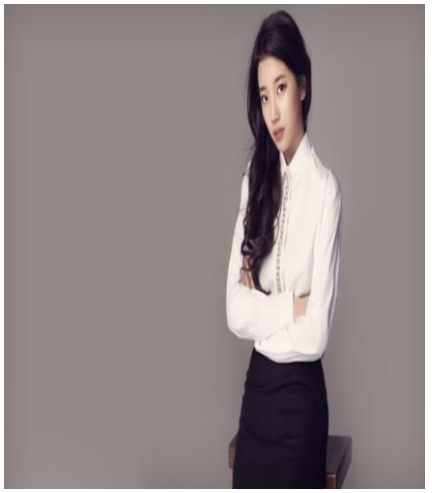
순 번	제 목	설치 버전
1	운영체제	Windows 10 64bit
2	프로그래밍언어	Python3.6.5

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-01 example.py] – 이미지 더하기

  - 다음은 08-01 example.py 예제의 code이다.

[그림] img1



[그림 img2

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-01 example.py] – 이미지 더하기

  >> img = img1 + img1

  Img1 과 img2를 더하여 새로운 이미지를 생성 한다. 단 img1 과 img2 동일 크기의 동일한 데이터 타입으로 작성되어야 한다. 또는 img2 가 하나의 단일한 값을 가져도 된다.

  이 연산에서 각 픽셀들을 더한 값이 255보다 크면 그 값을 256 으로 나눈 나머지가 픽셀값이된다.

  예) 257일 경우 , 256 으로 나눈 나머지가 1이므로, 픽셀값은 1이된다.

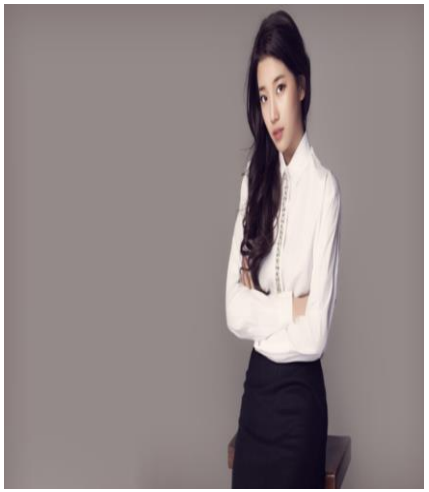
  이미지 합(+) 연산에 있어, 위 예처럼 Numpy array를 그대로 더하는 방법도 있지만, OpenCV의 add() 함수를 이용하는 방법도 있다.

  >> img = cv2.add(img1, img2)

  위 코드도 img1 과 img2 의 각 픽셀 값을 더한다. 하지만 Numpy array 연산과 다르게 더한 값이 255보다 크면 255 로 값이 정해진다.

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 합성 이미지

8. 이미지 연산 처리를 이용한 이미지 합성 하기

이미지 블렌딩(Image Blending)

이미지 **블렌딩은 가중치를 두어 합치는 방법**이다. 예를 들명 두 개의 이미지가 있고, 1번 이미지에서 2번 이미지로 전환 될때 1번에서 2번으로 서서히 변해가도록 할때 이미지 블렌딩은 좋은 기업이다.

함수 : $g(x) = (1-\alpha) f_0(x) + \alpha f_1(x)$

위 식에서 알파값이 0에서 1로 변해가면 f0의 효과는 감소하하고 f1의 효과는 커지게 된다.

Cv2.addWeighted() 함수는 위의 수식을 차용하여 새로운 이미지로 블렌딩하는 함수이다.

- 8.2 [08-02 example.py]

8. 이미지 연산 처리를 이용한 이미지 합성 하기

코드 해석

무한 루드 내에서 처리

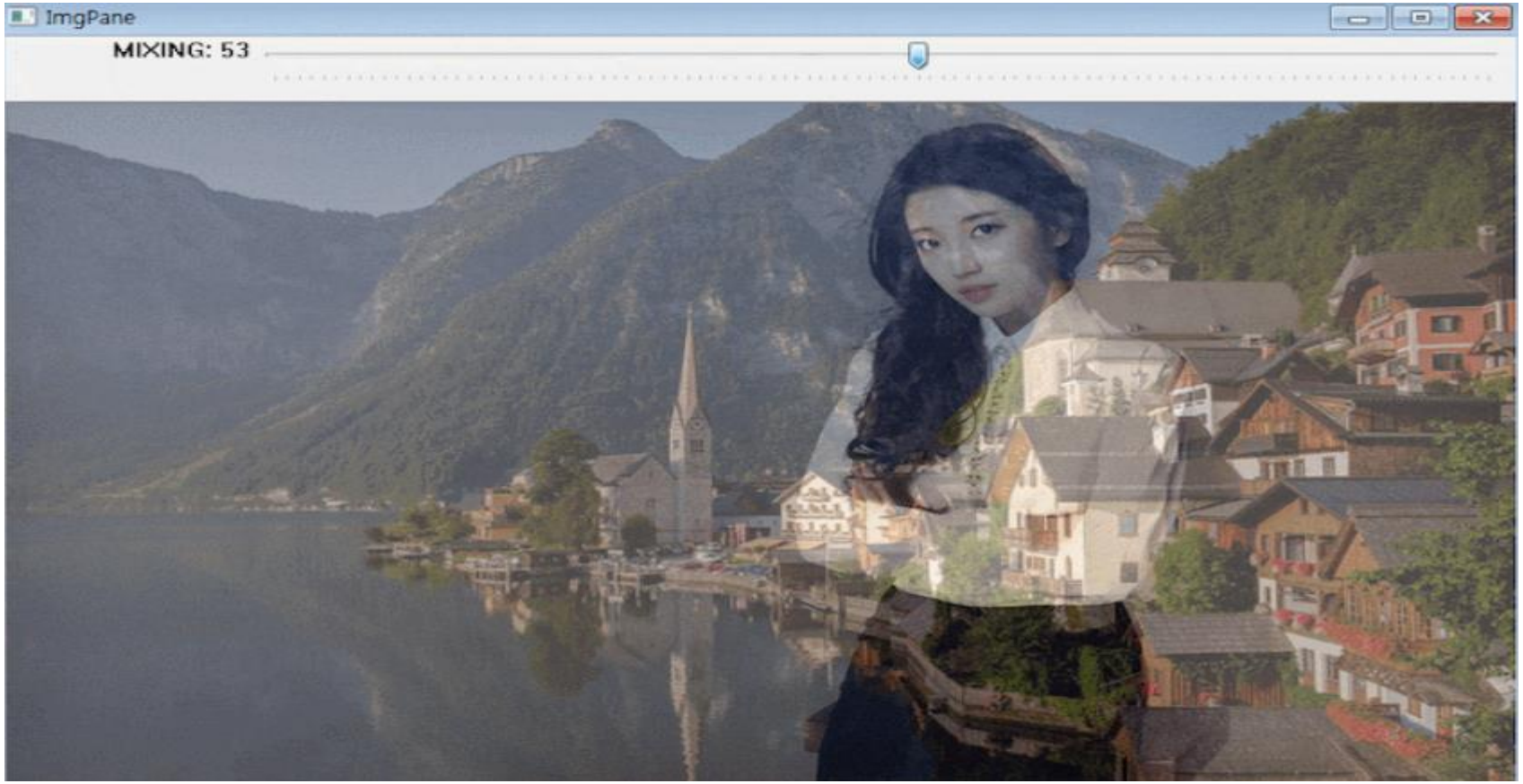
```
>> img=cv2.addWeighted(img1, float(100-mix)/100, img2, float(mix)/100,0)
```

트랙바로부터 얻은 mix 값은 0~100 까지 변동가능하며, img1 에서 가중치 float(100-mix)/100, img2에는 가중치 float(mix)로 주었습니다.

Mix 초기값이 0이므로, 화면에는 img1만 보이게 된다 트랙바를 움직이면 두 이미지가 서서히 블렌딩되는 것을 볼수있으며, 트랙바를 100으로 두게되면 img2만 보이게 된다.

아래 화면 위 코드를 실행시킨 후, 트랙바를 움직여 imx 값이 0,25,50,75,100 일때 화면에 디스플레이되는 이미지입니다.

8. 이미지 연산 처리를 이용한 이미지 합성 하기



8. 이미지 연산 처리를 이용한 이미지 합성 하기

이미지 비트 연산

이미지간 비트 연산은 다른 비트 연산과 마찬가지로 AND, OR,NOT, XOR 연산이있습니다. 이미지 비트 연산은 이미지에서 특정 영역을 추출하거나 직사각형 모양이 아닌 ROI를 정의하거나 할때 매우 유용하다.

우리가 해볼 것은 앞으로 예쁜 수지 사진에 아래의 OpenCV 로고를 추가해보는 것이다.

이미지 검정색 바탕이라 이 이미지 그대로 전투기 그림 위에 올리면 수지 사진이 답답해할 것 같아서 검정색 부분은 투명하도록 변경하는 코드이다.

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-03 example.py] – 이미지 로고 추가

  >> def bitOperation(hpos, vpos)

  로고가 이미지 위에 놓일 위치를 인자로 받습니다.메인 함수에서 bitoperation(10,10) 호출하였으므로, hpos=10, vpos=10 된다. OpenCV 로 원본 이미지 (10,10) 좌표 위치 하게된다.

  >>> rows, cols, channels = img2.shape

  >>> roi=img1[vpos:rows+vpos, hpos:cols+hpos]

  OenCV 로고 이미지 크기를 구하고, 전투기 이미지 위의 좌표(10,10)에서 Cols+10, rows+10)를 OpenCV로고를 위한 영역으로 ROI를 잡습니다. 분홍색 점선 표시 영역

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-03 example.py] – 이미지 로고 추가

```
>> img2gray=cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    ret, mask=cv2.threshold(img2gray, 10, 255, cv2, THRESH_BINARY)
    mask_inv =cv2.bitwise_not(mask)
```

로고 이미지를 흑백으로 변환하고, 이를 이용해 마스크를 만든다.

이미지의 특정 영역에 마스크를 씌운 후 연산을 하게 되면

특정한 효과를 낼 수가 있죠.

생성되는 마스크는 흑, 백으로 완전히 구분되게 한 마스크와 그 반대의 이미지를 생성한다.

Cv2.threshold()함수는 이미지 threasholding 배울 때 자세하게 설명

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-03 example.py] – 이미지 마스크


```
>> img1_bg=cv2.bitwise_and(roi,roi,mask=mask=mask_inv)
>> img2_fg=cv2.bitwise_and(img2, img2, mask=mask)
```

Cv2.bitwise_and(src1, src2,mask)는 mask의 값이 0 이 아닌 부분만 src1과 src2를 AND 연산합니다.mask 의 값이 0 인 부분은 mask로 그대로 쓴다.

Cv2.bitwise_and()함수의 인자로 사용되는 mask는 1채널 값이어야 하므로 대부분 흑백 이미지이다.

Mask 의 값이 0 이 아닌 부분은 곧 흰색 부분을 말하므로 mask의 검정색 부분은 연산을 하지 않고 검정색 그대로 이미지에 놓여지게 된다.

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-03 example.py] – 이미지 마스크

img1_bg

img2_fg


```
>> dst=cvs2.add(img1_bg,img2_fg)
```

검정색 픽셀값은 0이므로 두 이미지를 더하게 되면 검정색은 없어지고 검정색 아닌 색이 표출되게 된다.

Img1_bg에 img2_fg를 합치면 아래와 같은 이미지가 생성된다.

8. 이미지 연산 처리를 이용한 이미지 합성 하기

- 8.2 [08-03 example.py] – 이미지 마스크

  >>> img1[vpos:rows+vpos, hpos:cols+hpos] = dst

  최종 이미지가 화면에 디스플레이된다.

```
import ...


def addImage(imgfile1, imgfile2):
    img1 = cv2.imread(imgfile1)
    img2 = cv2.imread(imgfile2)

    cv2.imshow('img1', img1)
    cv2.imshow('img2', img2)

    add_img1 = img1 + img2
    add_img2 = cv2.add(img1, img2)

    cv2.imshow('img1+img2', add_img1)
    cv2.imshow('add(img1,img2)', add_img2)

    cv2.waitKey(0)
    cv2.destroyAllWindows()
addImage('images/hallstatt.jpg', 'images/suji.jpg')
```

```
import ...
def onMouse(x):
    pass
def imgBlending(imgfile1, imgfile2):
    img1=cv2.imread(imgfile1)
    img2=cv2.imread(imgfile2)

    cv2.namedWindow('ImgPane')
    cv2.createTrackbar('MIXING','ImgPane', 0, 100, onMouse)
    mix=cv2.getTrackbarPos('MIXING','ImgPane')
    while True:
        img=cv2.addWeighted(img1, float(100-mix)/100, img2, float(mix)/100,0)
        cv2.imshow('ImgPane',img)
        k=cv2.waitKey(1) & 0xFF
        mix=cv2.getTrackbarPos('MIXING','ImgPane')
cv2.destroyAllWindows()
imgBlending('images/hallstatt.jpg','images/suji.jpg')
```

```
import ...

def bitOperation(hpos, vpos):
    img1=cv2.imread('images/suji.jpg')
    img2=cv2.imread('images/logo.jpg')

    rows, cols, channels = img2.shape
    roi=img1[vpos:rows+vpos, hpos:cols+hpos]

    img2gray=cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
    ret, mask=cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
    mask_inv =cv2.bitwise_not(mask)

    img1_bg=cv2.bitwise_and(roi,roi,mask=mask_inv)

    img2_fg=cv2.bitwise_and(img2, img2, mask=mask)

    dst=cv2.add(img1_bg, img2_fg)
    img1[vpos:rows+vpos, hpos:cols+hpos] = dst

    cv2.imshow('result', img1)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

bitOperation(10,10)
```