화소 처리 응용

Applications of Pixel-based Image Processing

목차

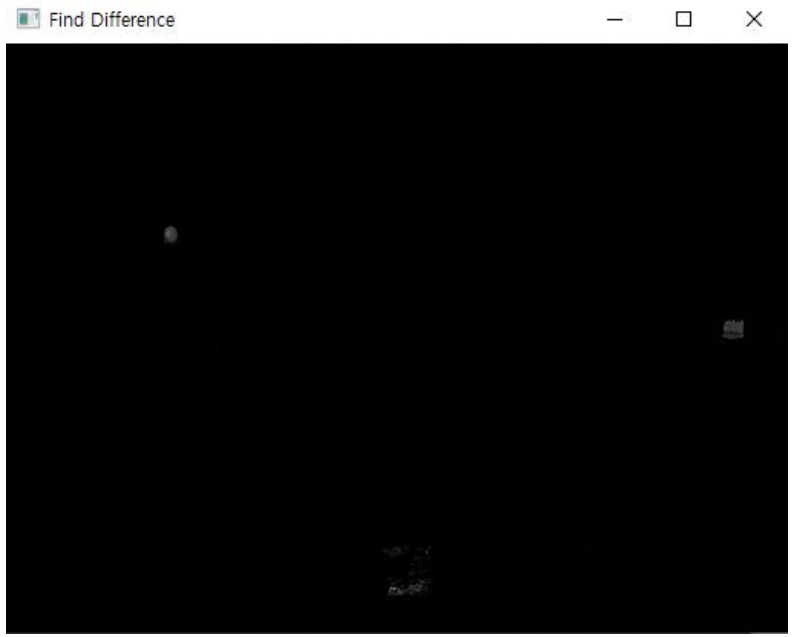
1. 영상에서 달라진 부분 탐색

2. 영상 검색
   - 히스토그램 비교
   - 히스토그램 역투영

두 영상에서 다른 부분을 찾아보자.

How can we find the difference between the two images?

```python
import cv2

img1 = cv2.imread("Lighthouse.jpg", cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread("Lighthouse3.jpg", cv2.IMREAD_GRAYSCALE)

dif_img = cv2.subtract(img1, img2)    # jpg 손실압축에 따른 차이
cv2.imshow("Find Difference", dif_img)

th, th_img = cv2.threshold(dif_img, 10, 255, cv2.THRESH_BINARY)
cv2.imshow("Find Difference Th", th_img)

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



히스토그램 응용 (영상 검색)

❖히스토그램 비교

compareHist (InputArray H1, InputArray H2, **int method**)

1) HISTCMP_CORREL ⬆

$$d(H_1,H_2)=\frac{\sum_i(H_1(i)-\overline{H_1})(H_2(i)-\overline{H_2})}{\sqrt{\sum_i(H_1(i)-\overline{H_1})^2\sum_i(H_2(i)-\overline{H_2})^2}}$$

2) HISTCMP_CHISQR ⬇

$$d(H_1,H_2)=\sum_i\frac{(H_1(i)-H_2(i))^2}{H_1(i)}$$

3) HISTCMP_INTERSECT ⬆

$$d(H_1,H_2)=\sum_i\min(H_1(i),H_2(i))$$

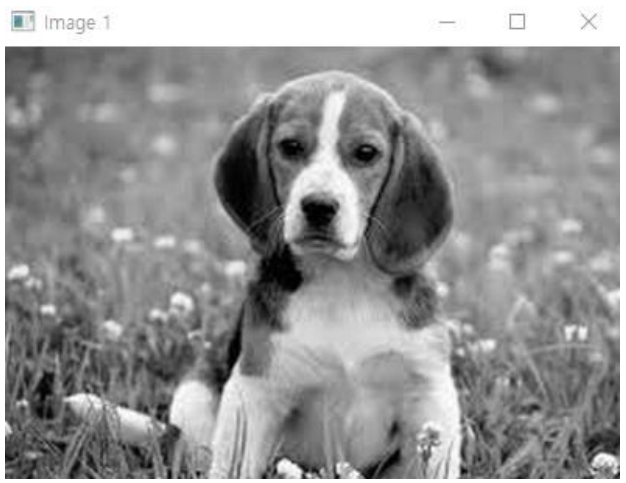
4) HISTCMP_BHATTACHARYYA ⬇

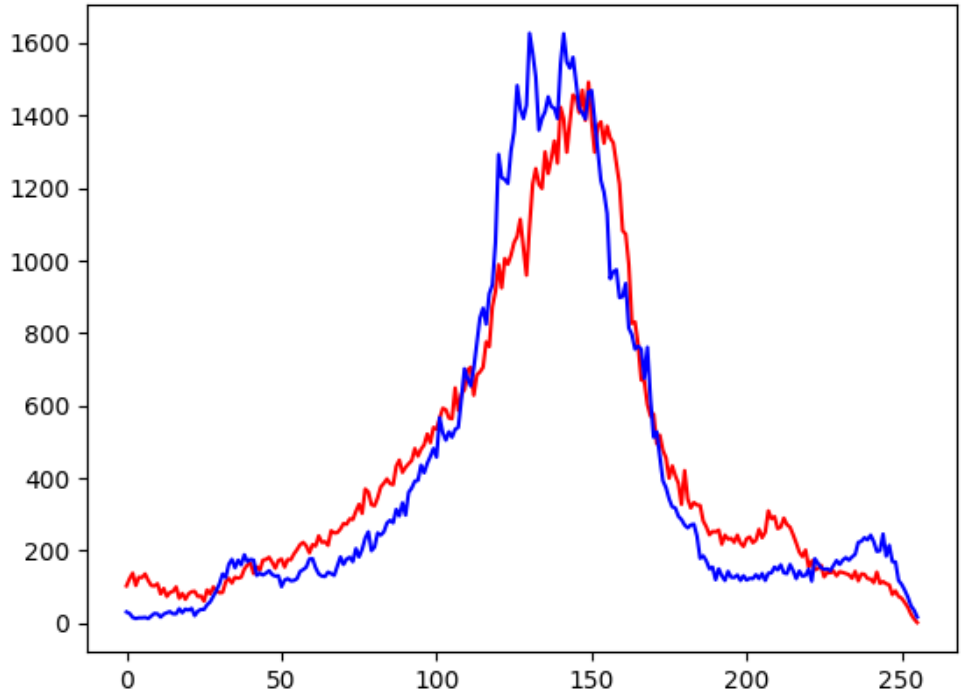
$0 \le d(H_1,H_2) \le 1$

$$d(H_1,H_2)=\sqrt{1-\frac{1}{\sqrt{\overline{H_1}\overline{H_2}N^2}}\sum_i\sqrt{H_1(i)H_2(i)}}$$

정규화된 히스토그램만 적용 가능 *N: bin의 개수*

```python
import cv2
import matplotlib.pyplot as plt

img1 = cv2.imread('./images/dog3.jpg',
cv2.IMREAD_GRAYSCALE)
img2 = cv2.imread('./images/dog8.jpg',
cv2.IMREAD_GRAYSCALE)
cv2.imshow("Image 1", img1)
cv2.imshow("Image 2", img2)

hist1 = cv2.calcHist([img1], [0], None, [256], [0,256])
hist2 = cv2.calcHist([img2], [0], None, [256], [0,256])
plt.plot(hist1, color='red')
plt.plot(hist2, color='blue')

corr1 = cv2.compareHist(hist1, hist2, cv2.HISTCMP_CORREL)
print("Correlation distance (큰 값이 유사)", corr1)

plt.show()
cv2.destroyAllWindows()
```





dog1

dog4

dog7

dog2

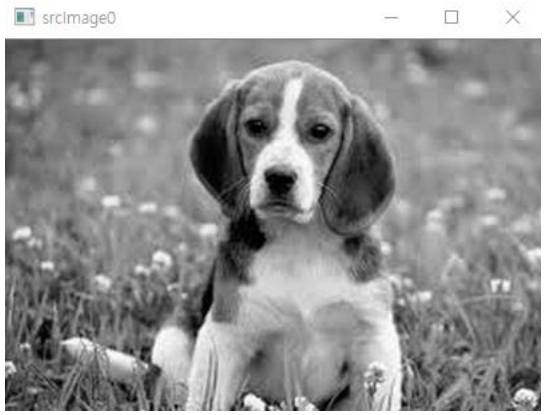
dog5

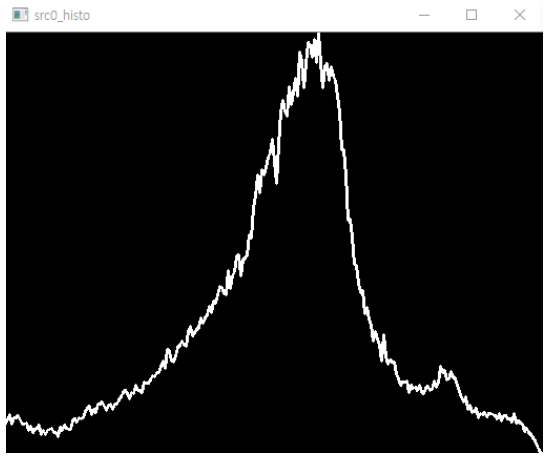
dog8

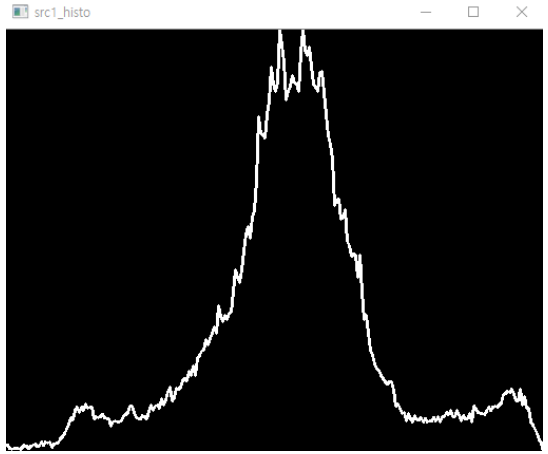
dog3

dog6

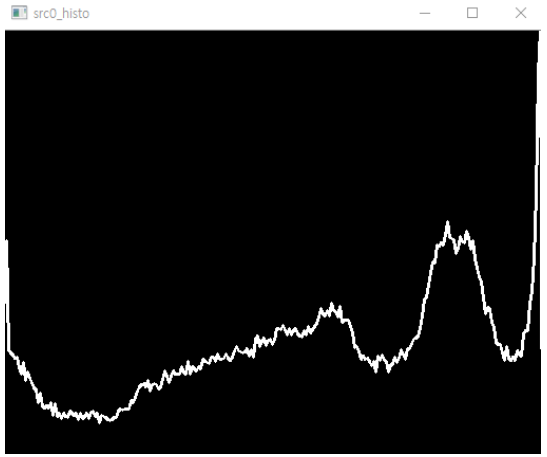
dog9

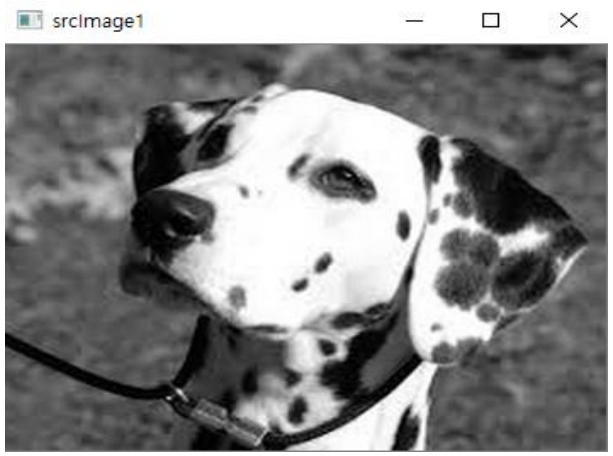


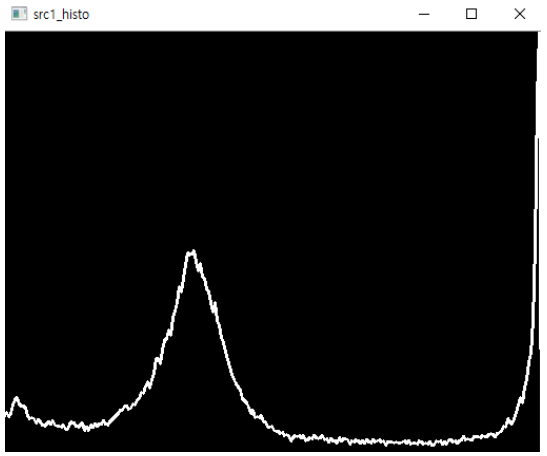





```
 --- Histogram Distance ---
Correl distance (큰값이 유사) = 0.965802
Chisqr distance (작은값 유사) = 8959.174834
Intersect distance (큰값이 유사) = 97301.000000
Intersect distance (Norm) = 0.867535
Earth Mover distance (작은값 유사) = 3.648142
```








```
 --- Histogram Distance ---
Correl distance (큰값이 유사) = 0.123256
Chisqr distance (작은값 유사) = 108239.590904
Intersect distance (큰값이 유사) = 63970.000000
Intersect distance (Norm) = 0.480270
Earth Mover distance (작은값 유사) = 14.489852
```

히스토그램 응용 (역투영)

❖히스토그램 역투영

- 명도 값을 해당 명도의 히스토그램 빈도로 변환

Ex) 8 levels gray image

2	4	4	3
2	1	3	3
1	0	1	2
0	1	1	2

h[i] = { 2, 5, 4, 3, 2, 0, 0, 0 }

histogram back projection →

4	2	2	3
4	5	3	3
5	2	5	4
2	5	5	4

✓ 히스토그램 빈도가 큰 명도 값이 역투영 결과 큰 값을 갖는다.
즉 역투영한 값이 크다는 것은 원래 히스토그램에 자주 나타난 명도 값을 의미함

히스토그램 역투영

h[i] = { 2, 5, 4, 3, 2, 0, 0, 0 }

2	4	4	3
2	1	3	3
1	0	1	2
0	1	1	2

Patch A

6	4	4	6
6	7	6	6
7	7	7	6
1	7	5	5

histogram back projection
by Patch A

0	2	2	0
0	0	0	0
0	0	0	0
5	0	0	0

1	4	4	6
1	7	2	2
3	7	2	2
1	3	1	1

histogram back projection
by Patch A

5	2	2	0
5	0	4	4
0	0	4	4
5	3	5	5

히스토그램 역투영

```
calcBackProject (
        const Mat* images,              // 1개 이상의 영상
        int nimages,                    // 영상의 개수
        const int* channels,            // 역투영에 사용할 채널
        const SparseMat& hist,          // 입력으로 사용되는 히스토그램
        OutputArray backProject,        // images 영상과 같은 크기와
                                        // 같은 깊이를 갖는 1-채널 역투영 행렬
        const float** ranges,
        double scale=1, bool uniform=true)
);
```

```python
import cv2

src = cv2.imread("./images/logo.jpg")
cv2.imshow("Image", src)

# 관심영역 추출
x, y, w, h = cv2.selectROI(src)
print(x, y, w, h)

hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)
roi = hsv[y:y+h, x:x+w]

channels = [0, 1]   # 명도는 사용하지 않음
h_bins = 128        # 색상(hue)의 범위 0~127
s_bins = 256        # 밝기(value)의 범위 0~127
histSize = [h_bins, s_bins]
ranges = [0, 128] + [0, 256]
hist = cv2.calcHist([roi], channels, None, histSize, ranges)

# 히스토그램 역투영으로 마스크 생성
backproj = cv2.calcBackProject([hsv], channels, hist, ranges, 1)
cv2.imshow('backprj', backproj)

# 마스크 연산
dst = cv2.copyTo(src, mask=backproj)
cv2.imshow('dst', dst)

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