히스토그램 역투영(BackProjection)

◆ 히스토그램 역투영(BACKPROJECTION)

❖ 히스토그램을 매핑함수로 사용하여, **화소값**을 **신뢰도값**으로 변환

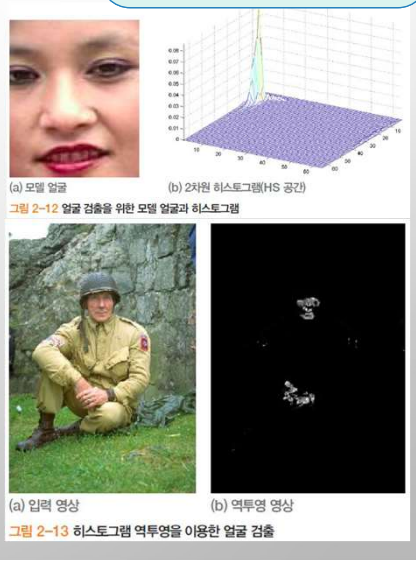
피부색 픽셀이 다수인 모델 얼굴에 대한 히스토그램
-> 가장 높은 히스토그램을 가지는 색상은 피부색
-> 즉, 히스토그램 비율만큼의 피부색 신뢰도를 가짐

❖ 얼굴검출 예

- 모델얼굴에서 구한 히스토그램 H_M은 화소의 컬러값을 얼굴에 해당하는 신뢰도값으로 변환해 줌
- 실제로는 비율 히스토그램 H_R을 사용

$$h_r(j,i) = \min\left(\frac{\hat{h}_m(j,i)}{\hat{h}_i(j,i)}, 1.0\right), \quad 0 \le j, \ i \le q-1$$

- 히스토그램 역투영 결과
  - ✓ 얼굴영역은 높은 신뢰도값, 손영역도 높은값
- 한계: 비슷한 색분포를 갖는 다른 물체구별 못함. 검출 대상이 여러 색분포를 갖는 경우 오류 가능성
- 장점: 배경을 조정할 수 있는 상황에 적합 (이동과 회전에 불변, 가림 OCCLUSION에 강인)



57

히스토그램 역투영

◆ 히스토그램 역투영

❖ **cv2.calcBackProject(images, channels, hist, ranges, scale[, dst]) ➔ dst**

- **images 입력영상의 channel에 대해서 histSize 크기의 히스토그램을 계산하여 hist 결과 히스토그램으로 반환하는 함수**
- **scale은 출력 역투영 행렬에 적용될 스케일 값. 디폴트는 1.**

❖ 예

```
src = np.array([[0,0,0,0],[1,1,3,5],[6,1,1,3],[4,3,1,7]], dtype=np.uint8)
hist1 = cv2.calcHist(images=[src], channels=[0], mask=None, histSize=[4],
ranges=[0,8])                                                [[9.][3.][2.][2.]]
backP1 = cv2.calcBackProject([src], [0], hist1, [0,8], scale=1)
                          [[9 9 9 9] [9 9 3 2] [2 9 9 3] [2 3 9 2]]
hist2 = cv2.calcHist(images=[src], channels=[0], mask=None, histSize=[4],
ranges=[0,4])                                                [[4.][5.][0.][3.]]
backP2 = cv2.calcBackProject([src], [0], hist2, [0,4], scale=1)
                          [[4 4 4 4] [5 5 3 0] [0 5 5 3] [0 3 5 0]]
hist3 = cv2.calcHist(images=[src], channels=[0], mask=None, histSize=[8],
ranges=[0,8])                                [[4.][5.][0.][3.][1.][1.][1.][1.]]
backP3 = cv2.calcBackProject([src], [0], hist3, [0,8], scale=1)
                          [[4 4 4 4] [5 5 3 1] [1 5 5 3] [1 3 5 1]]
```

58

예제 5.11 : 히스토그램 역투영 – hue 채널

```
src = cv2.imread('./data/fruits.jpg')

hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv)

roi = cv2.selectROI(src)
print('roi =', roi)
roi_h = h[roi[1]:roi[1]+roi[3], roi[0]:roi[0]+roi[2]]
hist = cv2.calcHist([roi_h], [0], None,[64], [0, 256]) #hist[r][c] = hist[y][x]
backP= cv2.calcBackProject([h.astype(np.float32)], [0], hist, [0, 256], scale=1.0)
##minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(backP)
##T = maxVal -1 # threshold

hist = cv2.sort(hist, cv2.SORT_EVERY_COLUMN+cv2.SORT_DESCENDING)
k = 1
T = hist[k][0] -1 # threshold
print('T =', T)
ret, dst = cv2.threshold(backP, T, 255, cv2.THRESH_BINARY)

cv2.imshow('dst',  dst)
```