

실습 6			
학번	20170766	이름	이건우
제출일	2022. 05. 19	전자 메일	dmz0128@naver.com
주제	바코드 검출		
실습목표	<ul style="list-style-type: none"> <li>영상 필터링, 영상 분할, 모풀로지 변환, 연결 요소 생성 등을 통해 다양한 이미지에 대해 바코드 영역을 검출한다.</li> </ul>		
해결 과정	<ul style="list-style-type: none"> <li>해결 과정             <ol style="list-style-type: none"> <li>컬러 영상을 그레이 스케일 영상으로 변환한다.</li> <li>수평 및 수직 방향으로 경계 강도를 계산한다.</li> <li>수평 방향을 기준으로 바코드 후보 영역을 검출한다.                     <ol style="list-style-type: none"> <li>수평 방향의 경계 강도 영상에서 수직 방향의 경계 강도 영상을 뺀다.</li> <li>뺄셈 결과에서 음수 값을 갖는 위치에서의 값을 모두 0.0으로 변경한다.</li> <li>적절한 처리 후 작업을 수행한다.                         <ol style="list-style-type: none"> <li>평활화를 수행한다.</li> <li>임계화를 수행한다. 수평 방향으로 긴 사각형 모양을 갖는 구조적 연산을 사용하여 닫힘 연산을 수행한 후, 다시 작은 사각형 모양을 갖는 구조적 요소를 사용하여 침식과 팽창 연산을 반복하여 적용한다.</li> </ol> </li> <li>연결 요소를 생성한 후 가장 크기가 큰 한 개의 요소를 선택한다.</li> </ol> </li> <li>수직 방향을 기준으로 바코드 후보 영역을 검출한다.                     <ol style="list-style-type: none"> <li>수직 방향의 경계 강도 영상에서 수평 방향의 경계 강도 영상을 뺀다.</li> <li>뺄셈 결과에서 음수 값을 갖는 위치에서의 값을 모두 0.0으로 변경한다.</li> <li>적절한 처리 후 작업을 수행한다.                         <ol style="list-style-type: none"> <li>평활화를 수행한다.</li> <li>임계화를 수행한다. 수직 방향으로 긴 사각형 모양을 갖는 구조적 연산을 사용하여 닫힘 연산을 수행한 후, 다시 작은 사각형 모양을 갖는 구조적 요소를 사용하여 침식과 팽창 연산을 반복하여 적용한다.</li> </ol> </li> <li>연결 요소를 생성한 후 가장 크기가 큰 한 개의 요소를 선택한다.</li> <li>수평 방향 및 수직 방향의 처리 과정에서 선택한 두 개의 연결 요소의 크기를 비교한다. 이 중에서 크기가 큰 연결 요소를 최종 바코드 영역으로 검출한다.</li> </ol> </li> <li>원본 영상에서 검출한 바코드 영역을 표시한 후 결과 영상을 저장한다,</li> <li>지정한 폴더에 포함된 모든 영상에 대해 1~5단계를 반복한다.</li> </ol> </li> </ul>		
설계 및 구현	<pre># 소벨 연산자를 사용하여 수평 및 수직 방향으로 경계 강도를 계산 sharpx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=-1) sharpx = cv2.convertScaleAbs(sharpx) sharpy = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=-1) sharpy = cv2.convertScaleAbs(sharpy)</pre> <ul style="list-style-type: none"> <li>Sobel 연산을 사용하여 수평 및 수직방향으로 각각 경계 강도를 계산한 후 계산 결과에 절대값을 구한다.</li> </ul>		

```

# 수평 방향의 경계 강도 영상에 수직 방향의 경계 강도 영상을 빼 후보 영역을 검출
dstx = cv2.subtract(sharpX, sharpY)
# 가우시안 블러 적용
dstx = cv2.GaussianBlur(dstx, (15, 13), 0)
# 임계화 수행
th, dstx = cv2.threshold(dstx, 100, 200, cv2.THRESH_BINARY)

```

- 수평 방향의 경계 강도 영상에 수직 방향의 경계 강도 영상을 빼 후보영역을 검출한다.
- Gaussian blur를 적용하여 영상 평활화를 수행한다. 수평(15, 13), 수직(13, 15)일 때 가장 좋은 결과를 얻을 수 있었다.
- Threshold를 사용하여 임계화를 수행했다. Threshold 값이 100일 때 가장 좋은 결과를 얻을 수 있었다.

```

# 모폴로지 변환(닫힘 연산) 적용, 직사각형
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (57, 13))
dstx = cv2.morphologyEx(dstx, cv2.MORPH_CLOSE, kernel)
# 3번 반복하여 침식과 팽창 연산을 적용, 직사각형
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (13, 13))
dstx = cv2.erode(dstx, kernel, iterations=3)
dstx = cv2.dilate(dstx, kernel, iterations=3)

```

- 긴 사각형 모양을 갖는 구조적 연산을 사용하여 morphology 닫힘 연산을 수행했다.
- 정사각형 모양을 갖는 구조적 연산을 사용하여 침식과 팽창 연산을 3번 반복해주었다.

```

# 연결요소를 찾아 가장 큰 연결 요소만
(contours, hierarchy) = cv2.findContours(dstx, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
if not contours:
    contour_x = np.array([[0, 0], [0, 0], [0, 0], [0, 0]])
else:
    contour_x = sorted(contours, key=cv2.contourArea, reverse=True)[0]

```

- 연결 요소를 찾아 가장 큰 연결요소만 구해 주었다.

```

# 수직방향 동일 진행
dsty = cv2.subtract(sharpY, sharpX)
dsty = cv2.GaussianBlur(dsty, (13, 15), 0)
th, dsty = cv2.threshold(dsty, 100, 200, cv2.THRESH_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (13, 57))
dsty = cv2.morphologyEx(dsty, cv2.MORPH_CLOSE, kernel)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (13, 13))
dsty = cv2.erode(dsty, kernel, iterations=3)
dsty = cv2.dilate(dsty, kernel, iterations=3)

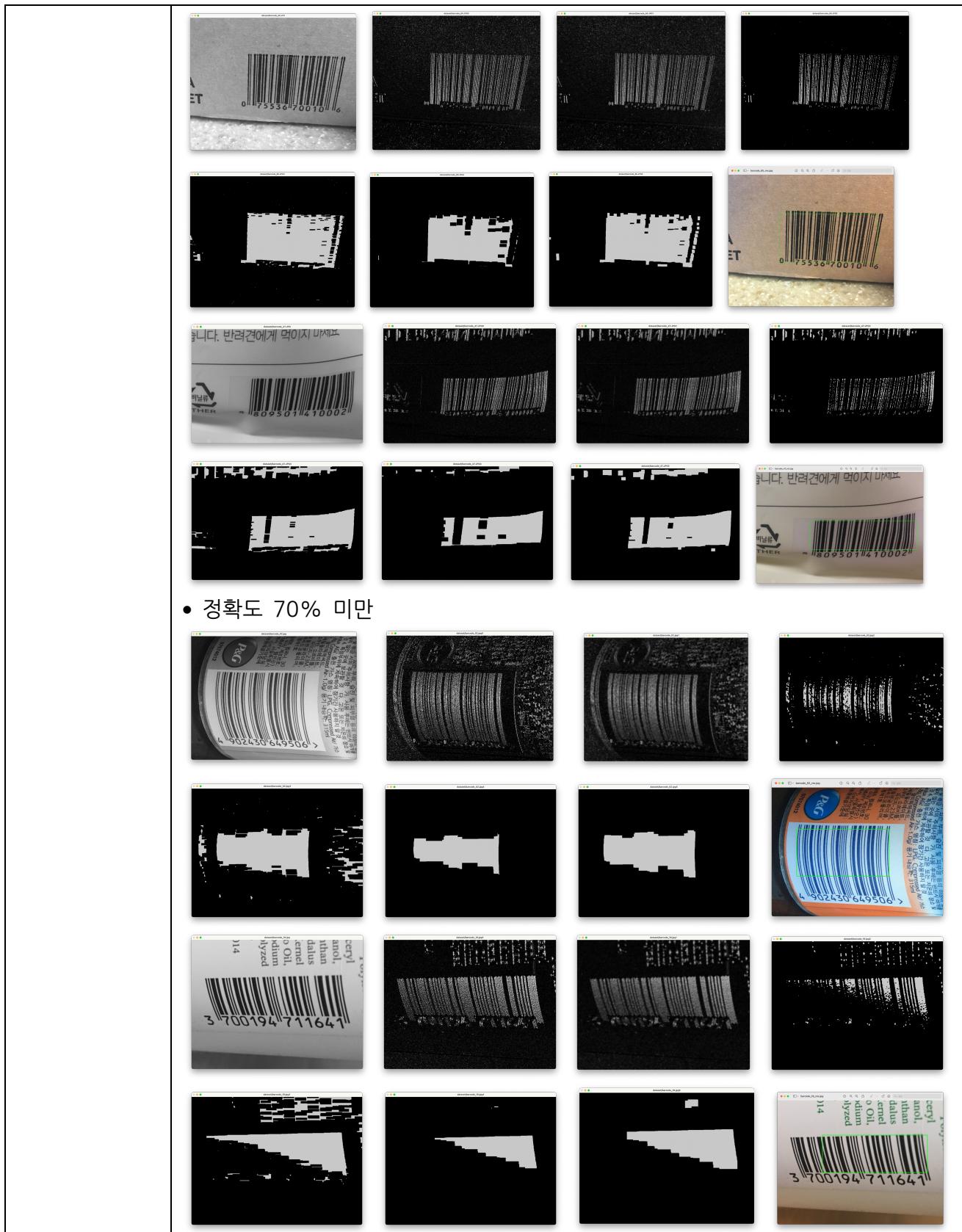
(contours, hierarchy) = cv2.findContours(dsty, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
if not contours:
    contour_y = np.array([[0, 0], [0, 0], [0, 0], [0, 0]])
else:
    contour_y = sorted(contours, key=cv2.contourArea, reverse=True)[0]

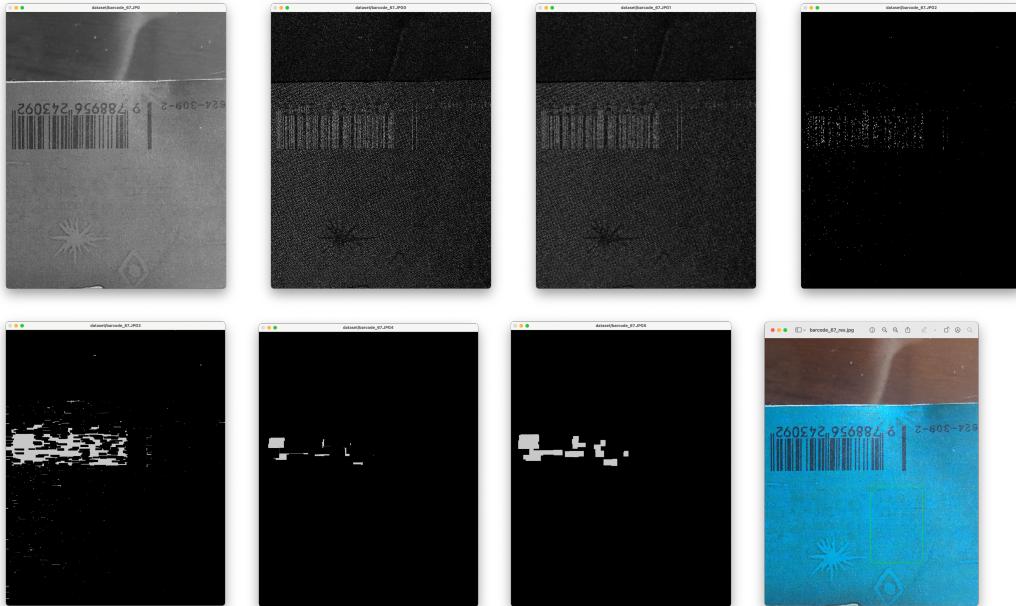
```

- 수직 방향으로 수평방향과 동일하게 진행해주었다.

	<pre> # 수평 방향과 수직 방향을 비교하여 연결 요소의 크기가 더 큰 방향의 값을 최종 바코드 영역으로 판단 if len(contour_x) &gt; len(contour_y):     rect = cv2.minAreaRect(contour_x)     box = cv2.boxPoints(rect)     box = np.int0(box)      # ref.dat에 값과 비교 시 좌표 순서 안 맞는게 있어서 코드 추가     if box[2][1] &lt; box[0][1]:         temp = box[2][1]         box[2][1] = box[0][1]         box[0][1] = temp     if box[2][0] &lt; box[0][0]:         temp = box[2][0]         box[2][0] = box[0][0]         box[0][0] = temp     else:         rect = cv2.minAreaRect(contour_y)         box = cv2.boxPoints(rect)         box = np.int0(box)          if box[2][1] &lt; box[0][1]:             temp = box[2][1]             box[2][1] = box[0][1]             box[0][1] = temp         if box[2][0] &lt; box[0][0]:             temp = box[2][0]             box[2][0] = box[0][0]             box[0][0] = temp  points = np.concatenate([box[0], box[2]]) </pre> <ul style="list-style-type: none"> <li>- 수평 방향과 수직 방향을 비교하여 연결 요소의 크기가 더 큰 방향의 값을 최종 바코드 영역으로 판단하여 좌측 상단, 우측 하단 좌표 값을 반환해주었다.</li> <li>- 내부 if 구문은 정확도 계산 시에 ref.dat 값과 detect.dat 값과 비교할 때 좌표의 순서가 안 맞아 값이 이상하게 나오는 경우를 대비하여 좌표의 자리를 정렬해 주는 구문이다.</li> </ul>
실험 결과	<ul style="list-style-type: none"> <li>• 평균 정확도 <ul style="list-style-type: none"> <li>(base) leegw ~/Downloads/barcode_detect&gt; python accuracy.py --reference ref.dat --detect detect.dat total number of list: 73 total number of list: 73 Accuracy data in accuracy.dat ===== average Precision: 0.92 average Recall: 0.9 average F1_Score: 0.91 average IOU_Score: 0.84</li> </ul> </li> <li>- F1 스코어 기준 평균 91%의 정확도가 나왔다.</li> <li>• 소요 시간  <ul style="list-style-type: none"> <li>- 모든 영상을 처리하는데 약 13초가 소요되었다.</li> </ul> </li> </ul>
정확도 분석	<ul style="list-style-type: none"> <li>• 정확도 90% 이상  </li> </ul>





	
<ul style="list-style-type: none"> <li>원인 분석</li> </ul>	<ul style="list-style-type: none"> <li>- 정확도 평균 90% 이상의 경우, 바코드가 뚜렷하고 왜곡이 없고 적당한 크기를 가지고 있음을 알 수 있다.</li> <li>- 정확도 평균 90%미만 70% 이상의 경우, 바코드가 조금 왜곡되어 있거나, 크기가 너무 크거나, 살짝 흐릿한 경우이다.</li> <li>- 정확도 평균 70% 미만의 경우, 바코드의 왜곡이 너무 심하거나 바코드와 배경의 대비가 적은 경우이다. 왜곡이 심한 경우는 왜곡된 영역을 바코드로 인식하지 못하였고, 대비가 적은 경우는 전체가 바코드 자체를 인식하지 못했다.</li> </ul>
<ul style="list-style-type: none"> <li>결론</li> </ul>	<ul style="list-style-type: none"> <li>- 바코드가 뚜렷하고, 적당한 크기를 가지고, 왜곡이 없고, 배경과의 대비가 클수록 검출이 잘 되었다.</li> <li>- 바코드가 왜곡이 되었을 경우와 배경과의 대비가 크지 않은 경우 현저하게 정확도가 떨어졌다.</li> </ul>
<b>느낀 점</b>	<ul style="list-style-type: none"> <li>• 바코드를 검출하는 과정에 대해서 이해가 높아졌다.</li> <li>• 바코드 검출 시에 정확도를 올리는 법에 대해 알 수 있었다.</li> <li>• 바코드가 왜곡된 경우와 배경과의 대비가 적은 경우에 대해서 바코드 검출을 잘 하지 못하였다.</li> <li>• 바코드의 왜곡된 경우와 배경과의 대비가 적은 경우에 대해서 바코드를 확실하게 검출할 수 있도록 탐구를 해야겠다.</li> </ul>