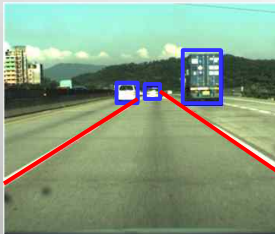



1

10.1 허프(Hough) 변환

- ◆ **에지 검출(8장)**
 - ❖ 미분 마스크를 이용해, 픽셀 값이 급격히 바뀌는 부분을 검출해내는 것
 - ❖ 영상 분석의 시작
- ◆ **직선 검출**
 - ❖ 영상 내에서 공간 구조를 분석하는데 유용한 도구
 - ❖ 영상 처리와 컴퓨터 비전분야에서 많은 연구 진행
 - ❖ 다양한 응용에 사용
 - 차선 및 장애물 자동인식 시스템 - 차선 검출
 - 스캐너의 기능을 대신해 주는 앱 - 네 개 모서리 검출

2

허프(Hough) 변환을 이용한 직선 검출

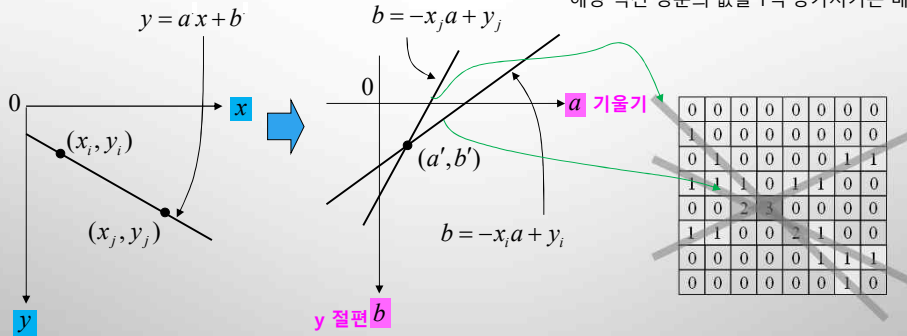
◆ 허프 변환(Hough transform)

- ❖ 2차원 영상 좌표에서의 직선의 방정식을 **파라미터(parameter) 공간으로 변환**하여 직선을 찾는 알고리즘

$$y = ax + b \Leftrightarrow b = -xa + y$$

누적 배열(accumulation array)

일반적인 2차원 배열을 이용하여 만든 후, 해당 직선 성분의 값을 1씩 증가시키는 배열



◆ 직선의 방정식 $y = ax + b$ 를 사용할 때의 문제점

- ❖ y 축과 평행한 수직선($x=c$)을 표현하지 못함 : 수직선일 경우에 기울기가 무한대

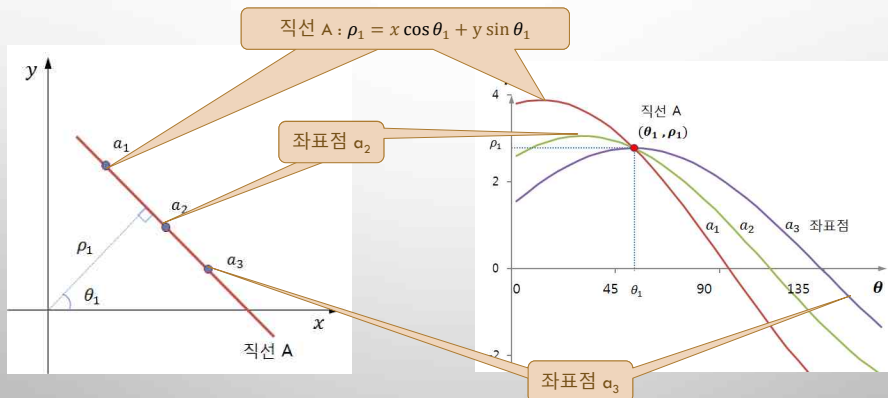
3

10.1 허프 변환의 좌표계

◆ 해결책

- ❖ 직교 좌표계로 표현되는 영상의 에지 점들을 극(polar) 좌표계로 옮겨, 검출하고자 하는 물체의 파라미터(ρ, θ)를 추출하는 방법

$$y = ax + b \Leftrightarrow \rho = x \cdot \cos \theta + y \cdot \sin \theta$$



- ❖ 직교좌표의 직선은 허프변환 좌표에서 한점 (ρ_1, θ_1) 으로 표현
- ❖ 직교좌표의 한 점은 허프변환 좌표에서 곡선으로 표현

4

10.1.2 허프 변환의 전체 과정

1. 극 좌표계에서 누적 행렬 구성
2. 영상 화소의 직선 검사
3. 직선 좌표에 대한 극좌표 누적 행렬 구성
4. 누적 행렬의 지역 최댓값 선정
5. 직선 선별 - 임계값 이상인 누적값 선택 및 정렬

◆극좌표계에서 누적 행렬(배열) 구성

❖ 직선을 극좌표(ρ, θ)로 인덱싱해서 값을 누적하기 위한 배열

$$-\rho_{\max} \leq \rho \leq \rho_{\max}, \quad \rho_{\max} = \text{rows} + \text{cols}, \quad h = \frac{\rho_{\max}}{\Delta \rho} * 2$$

$$0 \leq \theta < \theta_{\max}, \quad \theta_{\max} = \pi, \quad w = \frac{\pi}{\Delta \theta}$$

교재 오타

각도간격,
거리간격

영상의 높이, 너비

행렬의 높이, 너비

5

10.1.2 허프 변환의 누적 행렬 구성

거리간격, 각도간격

```
01 def accumulate(image, rho, theta):
02     h, w = image.shape[:2]
03     rows, cols = (h+w) * 2 // rho, int(np.pi / theta) # 누적행렬 너비, 높이
04     accumulate = np.zeros((rows, cols), np.int32) # 직선 누적행렬
05
06     sin_cos = [(np.sin(t*theta), np.cos(t*theta)) for t in range(cols)] # 삼각 함수값 저장
07     pts = np.where(image > 0) # 넘파이 함수 활용 - 직선좌표 찾기
08
09     polars = np.dot(sin_cos, pts).T # 행렬 곱으로 극좌표 계산
10     polars = (polars / rho + rows / 2).astype('int') # 해상도 변경 및 위치 조정
11
12     for row in polars:
13         for t, r in enumerate(row):
14             accumulate[r, t] += 1 # 각도, 수직 거리 가져옴
15             # 극좌표에 누적
16     return accumulate
```

누적 행렬

삼각 함수 행렬

직선 극좌표를 누적 행렬의 인덱스로 계산

직선 좌표들의 극좌표

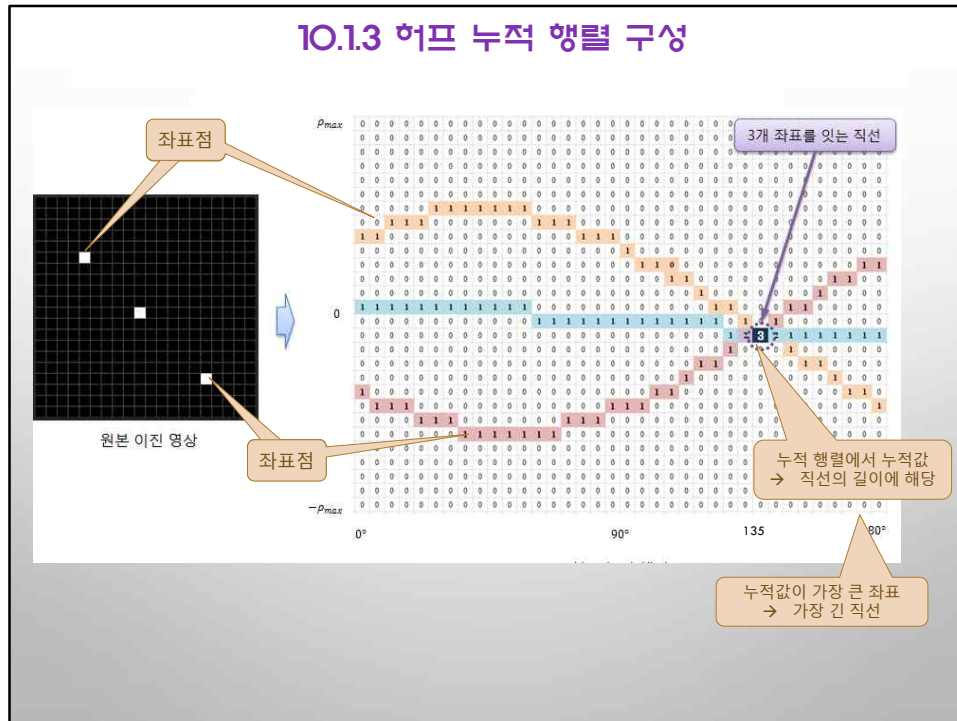
0~180도 각도에 대한 삼각함수 값 행렬

$$\begin{bmatrix} \sin \theta_0 & \cos \theta_0 \\ \sin \theta_1 & \cos \theta_1 \\ \sin \theta_2 & \cos \theta_2 \\ \dots & \dots \\ \sin \theta_m & \cos \theta_m \end{bmatrix} \begin{bmatrix} y_0 & y_1 & y_2 & \dots & y_n \\ x_0 & x_1 & x_2 & \dots & x_n \end{bmatrix} = \begin{bmatrix} \rho_0 \theta_0 & \rho_1 \theta_0 & \rho_2 \theta_0 & \dots & \rho_n \theta_0 \\ \rho_0 \theta_1 & \rho_1 \theta_1 & \rho_2 \theta_1 & \dots & \rho_n \theta_1 \\ \rho_0 \theta_2 & \rho_1 \theta_2 & \rho_2 \theta_2 & \dots & \rho_n \theta_2 \\ \dots & \dots & \dots & \dots & \dots \\ \rho_0 \theta_m & \rho_1 \theta_m & \rho_2 \theta_m & \dots & \rho_n \theta_m \end{bmatrix}$$

직선 좌표들

6

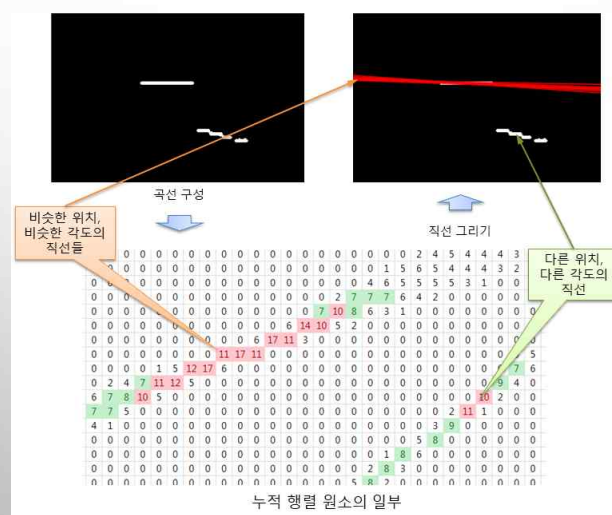
10.1.3 허프 누적 행렬 구성



7

10.1.3 허프 누적 행렬 구성

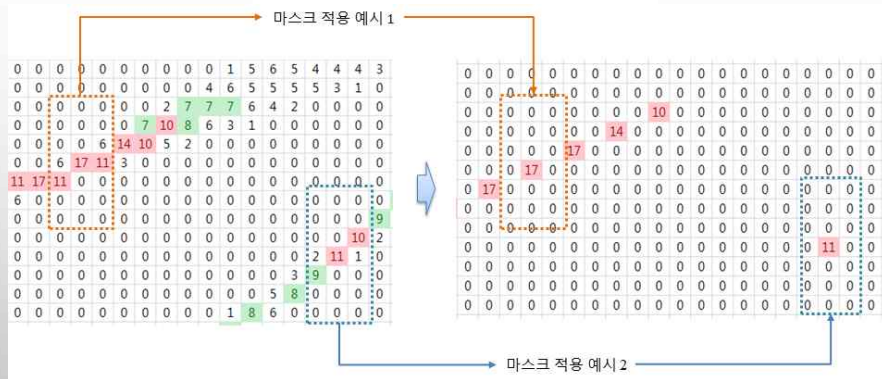
❖ 한 지점에서 여러 직선 검출의 문제



8

10.1.4 허프 누적 행렬의 지역 최대값 선정

◆ 마스크를 이용해 지역 최대값 선정



9

10.1.4 허프 누적 행렬의 지역 최대값 선정

```

01 def masking(accumulate, h, w, thresh):
02     rows, cols = accumulate.shape[:2]
03     rcenter, tcenter = h//2, w//2
04     dst = np.zeros(accumulate.shape, np.uint32)
05
06     for y in range(0, rows, h):
07         for x in range(0, cols, w):
08             roi = accumulate[y:y+h, x:x+w]
09             _, max, _, (x0, y0) = cv2.minMaxLoc(roi)
10             dst[y+y0, x+x0] = max
11     return dst

```

마스크 크기 절반

누적 행렬 조회

마스크 크기

최대값 위치

결과행렬의 최대값 위치

10

10.1.5 직선(극 좌표) 선택 및 정렬

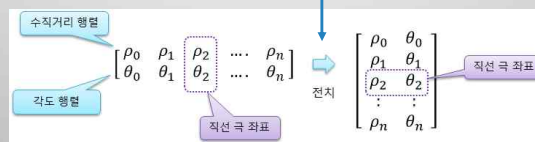
❖ 직선 선별 함수

```

01 def select_lines(acc_dst, rho, theta, thresh):
    rows = acc_dst.shape[0]
    03 r, t = np.where(acc_dst>thresh)           # 임계값 이상 인덱스 가져옴
    04
    05 rhos = ((r - (rows / 2)) * rho)          # 인덱스로 수직 거리 계산
    06 radians = t * theta                     # 인덱스로 각도 계산
    07 values = acc_dst[r, t]                  # 인덱스로 누적값 가져옴
    08
    09 idx = np.argsort(values)[::-1]           # 내림차순 정렬 인덱스
    10 lines = np.transpose([rhos, radians])     # 리스트 전치하여 행렬 생성
    11 lines = lines[idx, :]                   # 누적값 기준으로 극좌표 정렬
    12
    13 return np.expand_dims(lines, axis=1)     # 1번(열) 차원 증가

```

OpenCV 함수와 동일하게 변환 자료형 맞춤



11

10.1.7 최종 완성 프로그램

예제 10.1.1 허프 변환을 이용한 직선 검출 - 01.hough_lines.py

```

01 import numpy as np, cv2, math
02 from Common.hough import accumulate, masking, select_lines # 허프 변환 함수 임포트
03
04 def houghLines(src, rho, theta, thresh):
    05 acc_mat = accumulate(src, rho, theta)           # 허프 변환 함수
    06 acc_dst = masking(acc_mat, 7, 3, thresh)         # 직선 누적 행렬 계산
    07 lines = select_lines(acc_dst, rho, theta, thresh) # 마스크 처리 - 7행, 3열
    08 return lines                                     # 임계 직선 선택
    09
    10 def draw_houghLines(src, lines, nline):          # 검출 직선 그리기 함수
    11 dst = cv2.cvtColor(src, cv2.COLOR_GRAY2BGR)     # 컬러 영상 변환
    12 min_length = min(len(lines), nline)
    13
    14 for i in range(min_length):
    15     rho, radian = lines[i, 0, 0:2]                # 수직 거리, 각도 - 3차원 행렬임
    16     a, b = math.cos(radian), math.sin(radian)
    17     pt = (a * rho, b * rho)                       # 검출 직선상의 한 좌표
    18     delta = (-1000 * b, 1000 * a)                 # 직선상의 이동 위치
    19     pt1 = np.add(pt, delta).astype('int')
    20     pt2 = np.subtract(pt, delta).astype('int')
    21     cv2.line(dst, tuple(pt1), tuple(pt2), (0, 255, 0), 2, cv2.LINE_AA)
    22
    23 return dst
    24

```

허프 변환
전체 과정 수행 함수

직선위 2개 좌표 계산

12

10.1.7 최종 완성 프로그램

❖ 영상에 검출 직선 그리기 함수

```

25 image = cv2.imread("images/hough.jpg", cv2.IMREAD_GRAYSCALE)
26 if image is None: raise Exception("영상파일 읽기 에러")
27 blur = cv2.GaussianBlur(image, (5, 5), 2, 2)
28 canny = cv2.Canny(blur, 100, 200, 5)
29
30 rho, theta = 1, np.pi / 180
31 lines1 = houghLines(canny, rho, theta, 80)
32 lines2 = cv2.HoughLines(canny, rho, theta, 80)
33 dst1 = draw_houghLines(canny, lines1, 7)
34 dst2 = draw_houghLines(canny, lines2, 7)
35
36 cv2.imshow("image", image)
37 cv2.imshow("canny", canny);
38 cv2.imshow("detected lines", dst1)
39 cv2.imshow("detected lines_OpenCV", dst2)
40 cv2.waitKey(0)

```

Hough transform은 찾고자하는 형태(라인 또는 도형)의 개수에 따라 검색시간이 기하급수적으로 늘어날 수 있음. 따라서 형태를 표시하는 에지(edge)에 대해서만 검색하므로써 검색 시간을 줄일 수 있음.

가우시안 블러링

캐니 에지 추출

수직거리 간격, 각도 간격

저자 구현 함수

OpenCV 함수

직선 그리기

13

10.1.7 최종 완성 프로그램

◆ 실행결과

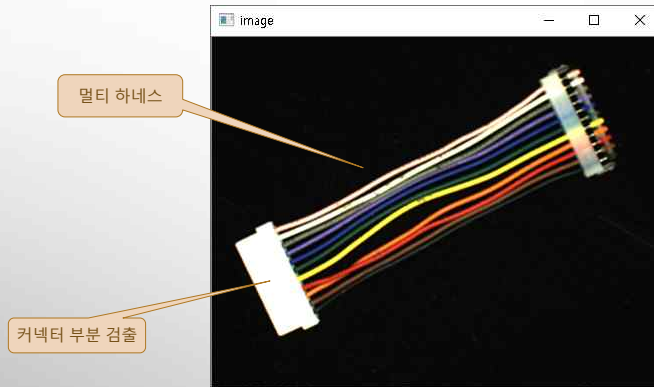


14

10.1.8 심화예제 -멀티 하네스의 전처리(기울기 보정)

❖ 멀티하네스란

- 전자제품 내부에서 다종의 전선들을 한 번에 연결시키는 케이블 커넥터



15

10.1.8 심화예제 -멀티 하네스의 전처리(기울기 보정)

심화예제 10.1.2 직선 검출을 이용한 멀티 하네스 기울기 보정 - 02.correct_object.py

```

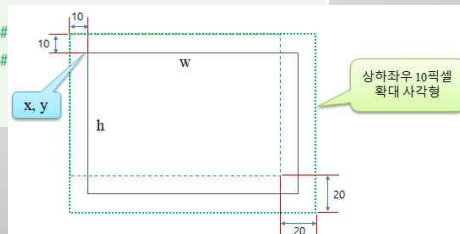
01 import numpy as np, cv2
02 from Common.hough import * # 허프 변환 관련 사용자 정의 함수 포함
03
04 def detect_maxObject(img):
05     results = cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
06     if int(cv2.__version__[0]) >= 4: # OpenCV 4.x - 2 원소 튜플 반환
07         contours = results[0]
08     else:
09         contours = results[1] # OpenCV 3.x - 3 원소 튜플 반환
10
11     areas = [cv2.contourArea(c) for c in contours] # 검출 객체들의 넓이 가져옴
12     idx = np.argsort(areas) # 오름차순 정렬 인덱스
13     max_rect = contours[idx[-1]] # 오름차순이라 마지막 원소가 최대
14
15     rect = cv2.boundingRect(max_rect) #
16     rect = np.add(rect, (-10, -10, 20, 20)) #
17     return rect
18

```

객체 외곽선 검출

넓이 리스트의 정렬 인덱스

검출 객체들의 넓이 가져옴



16

심화예제 - 멀티하네스 기울기 보정

```

19 image = cv2.imread("images/harness.jpg", cv2.IMREAD_COLOR)
20 if image is None: raise Exception("영상파일 읽기 에러")
21
22 rho, theta = 1, np.pi / 180 # 허프 변환 거리&각도 간격
23 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # 명암도 영상 변환
24 th_gray = cv2.threshold(gray, 240, 255, cv2.THRESH_BINARY)[1] # 이진 영상 변환
25 kernel = np.ones((3, 3), np.uint8)
26 morph = cv2.erode(th_gray, kernel, iterations=2) # 침식 연산- 2번 반복
27
28 x, y, w, h = detect_maxObject(np.copy(morph)) # 가장 큰 객체 rect 검출
29 roi = th_gray[y:y+h, x:x+w] # 검출 객체 관심영역
30
31 canny = cv2.Canny(roi, 40, 100) # 캐니 에지 검출
32 lines = houghLines(canny, rho, theta, 50) # 허프 직선 검출
33 # lines = cv2.HoughLines(canny, rho, theta, 50) # OpenCV 함수
34
35 cv2.rectangle(morph, (x, y, w, h), 100, 2) # 큰 객체 사각형 표시
36 canny_line = draw_houghLines(canny, lines, 1) # 직선 표시
37

```

배경 잡음 및
케이블 부분 제거

검출 객체 관심영역

직선 기울기 이용해
커넥터 기울기 계산

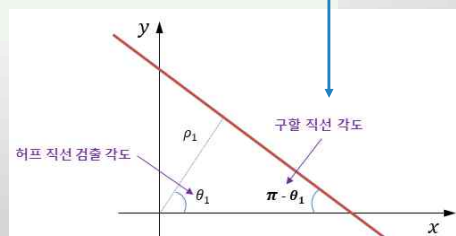
17

심화예제 - 멀티하네스 기울기 보정

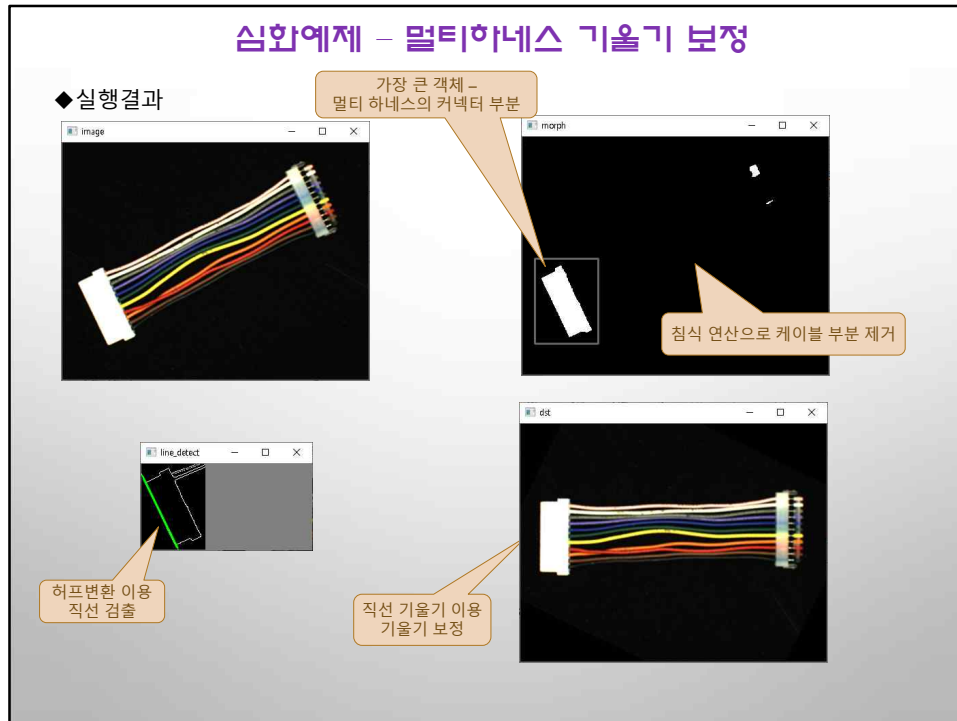
```

38 angle = (np.pi - lines[0, 0, 1]) * 180 / np.pi # 회전 각도 계산
39 h, w = image.shape[:2]
40 center = (w//2, h//2) # 입력 영상의 중심점
41 rot_map = cv2.getRotationMatrix2D(center, -angle, 1) # 역방향 회전 행렬 계산
42 dst = cv2.warpAffine(image, rot_map, (w, h), cv2.INTER_LINEAR) # 역회전 수행
43
44 cv2.imshow("image", image)
45 cv2.imshow("morph", morph)
46 cv2.imshow("line_detect", canny_line)
47 cv2.resizeWindow("line_detect", 250, canny_line.shape[0])
48 cv2.imshow("dst", dst)
49 cv2.waitKey(0)

```



18



19

단원 요약

◆허프 변환은 영상 내의 선, 원뿐만 아니라 임의의 형태를 지닌 물체를 감지해 내는 대표적인 기술로서 특히 직선 검출에 주로 사용된다. 직교 좌표계로 표현되는 영상의 에지 점들을 극좌표계로 옮겨, 검출하고자 하는 물체의 파라미터(ρ , θ)를 추출하는 방법이다.

$$y = ax + b \leftrightarrow \rho = x \cdot \cos \theta + y \cdot \sin \theta$$

◆허프 변환은 다음과 같은 세부적인 과정을 거쳐서 수행된다.

1. 극 좌표계에서 누적 행렬 구성
2. 영상 화소의 직선 검사
3. 직선 좌표에 대한 극좌표 누적 행렬 구성
4. 누적 행렬의 지역 최댓값 선정
5. 직선 선별 - 임계값 이상인 누적값 선택 및 내림 차순 정렬

20