가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

Hough Transformation

- 직선 및 원 검출

미디어기술콘텐츠학과
강호철

허프 변환 직선 검출

- 허프 변환 직선 검출
  - 직선 성분을 찾기 위해 에지를 찾아내고, 에지 픽셀들이 일직선상에 배열되어 있는지를 확인해야 함
  - 영상에서 직선을 찾기 위한 용도로 허프 변환(Hough transform) 기법이 널리 사용됨
  - 허프 변환은 2차원 xy 좌표에서 직선의 방정식을 파라미터(parameter) 공간으로 변환하여 직선을 찾는 알고리즘임
  - 2차원 평면에서 직선의 방정식은 다음과 같이 나타낼 수 있음

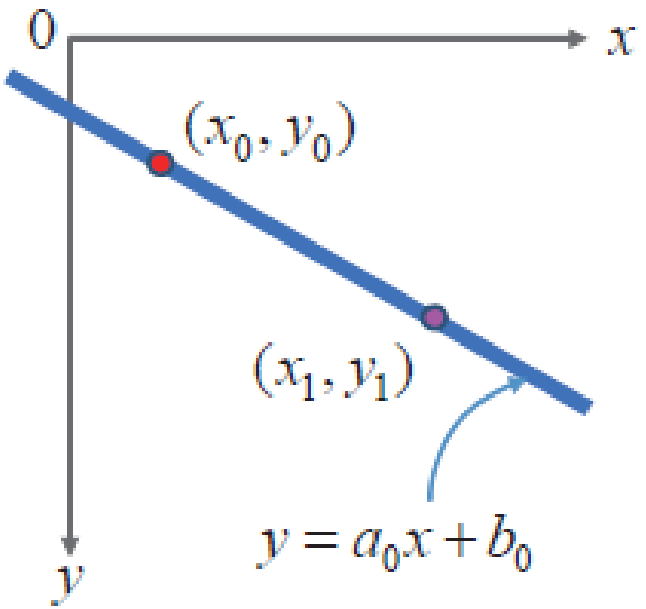
$$y = ax + b$$

  - 이 수식에서 a는 기울기이고, b는 y 절편임
  - xy좌표 공간의 직선 방정식을 ab좌표 공간으로 변형

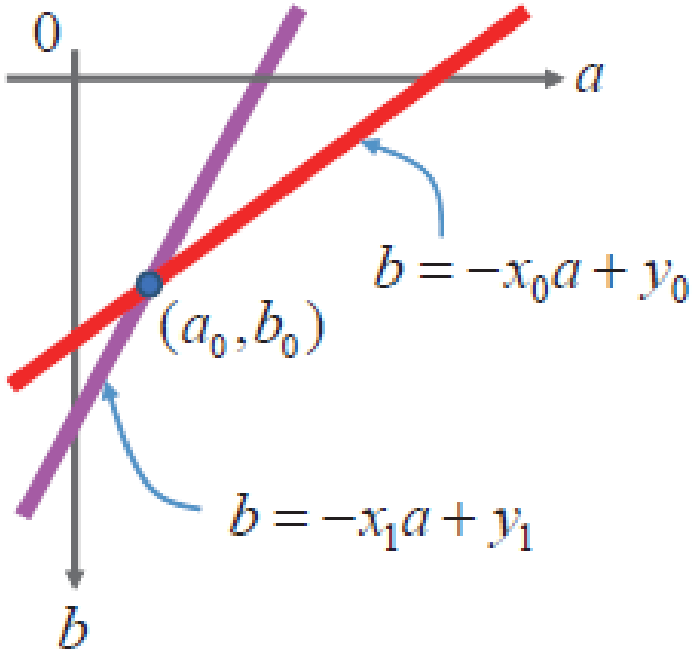
$$b = -xa + y$$

허프 변환 직선 검출

- 허프 변환 직선 검출



(a)



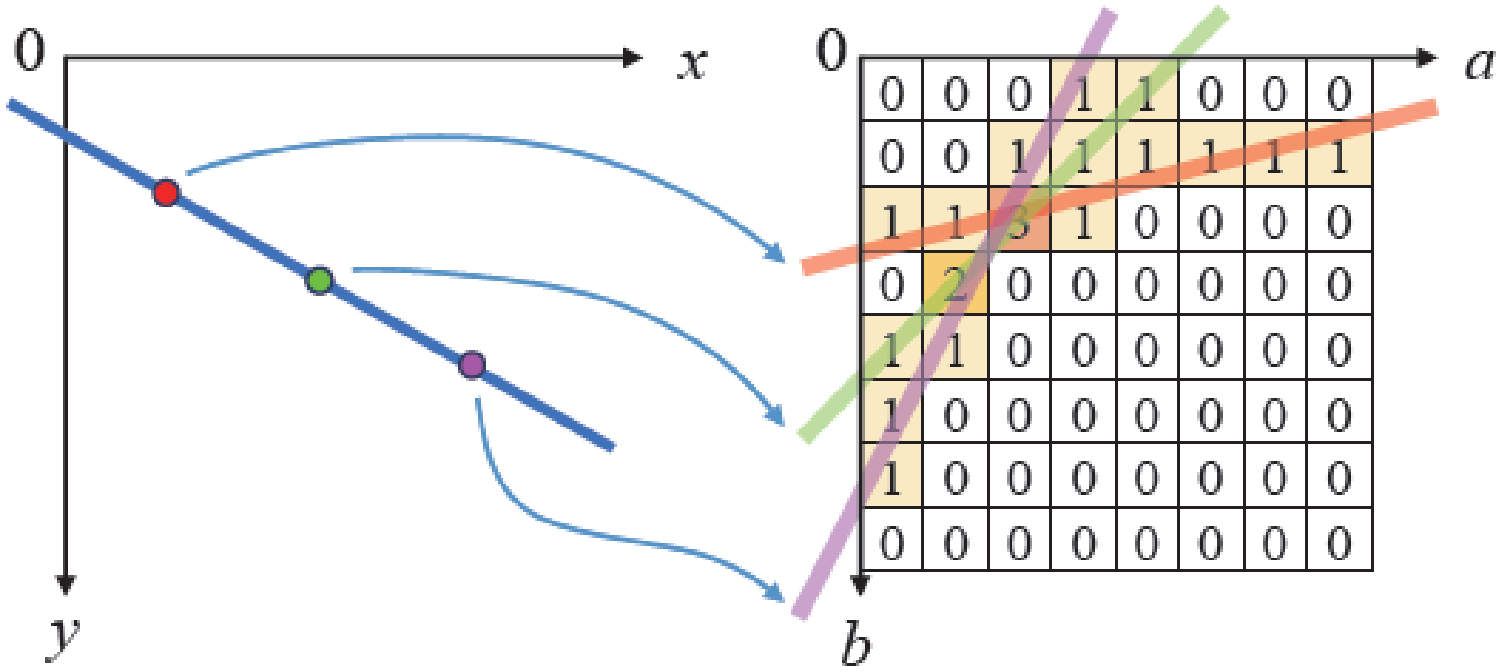
(b)

허프 변환 직선 검출

- 허프 변환 직선 검출
  - 허프 변환을 이용하여 직선의 방정식을 찾으려면 xy 공간에서 에지로 판별된 모든 점을 이용하여 ab 파라미터 공간에 직선을 표현함
  - 직선이 많이 교차되는 좌표를 모두 찾아야 함
  - 이때 직선이 많이 교차하는 점을 찾기 위해서 보통 축적 배열(accumulation array)을 사용함
  - 축적 배열은 0으로 초기화된 2차원 배열에서 직선이 지나가는 위치의 배열 원소 값을 1씩 증가시켜 생성함

허프 변환 직선 검출

- 허프 변환 직선 검출



허프 변환 직선 검출

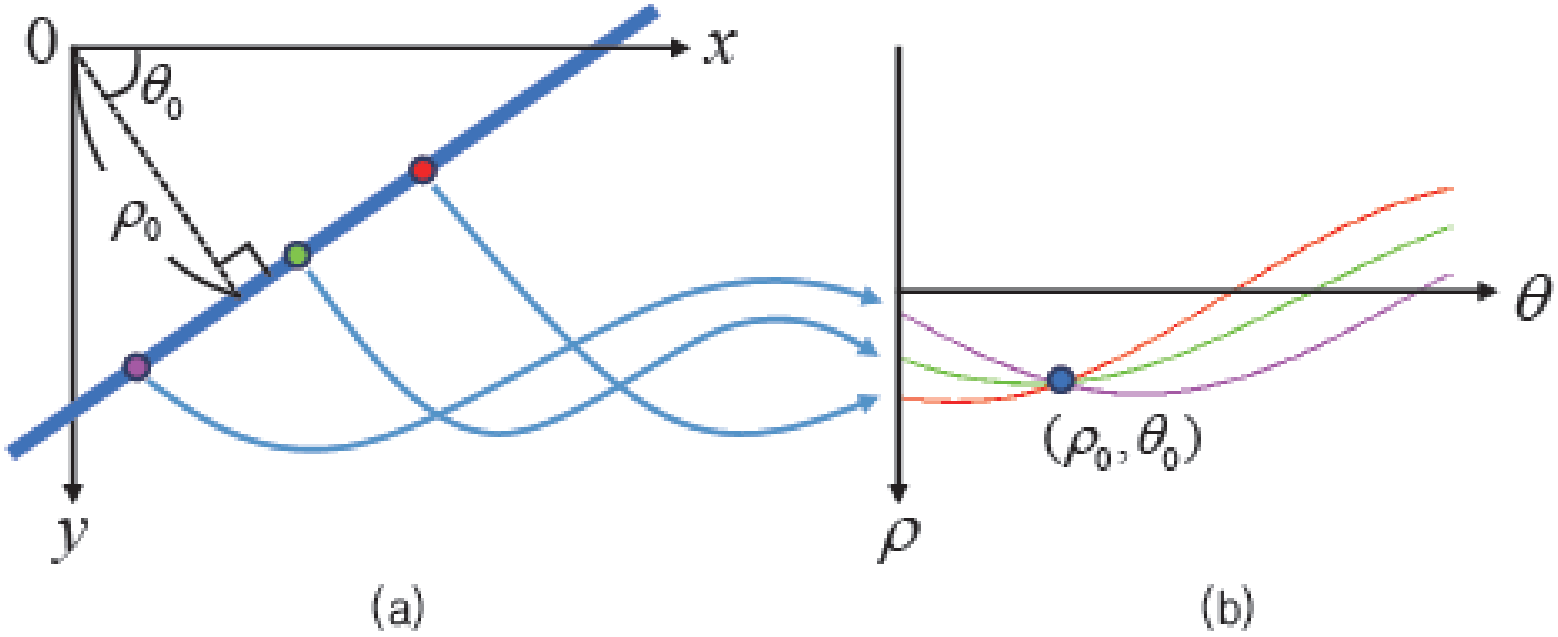
- 허프 변환 직선 검출
  - y = ax + b 직선의 방정식을 사용할 경우 모든 형태의 직선을 표현하기 어려움
    - y축과 평행한 수직선을 표현할 수 없음
    - 수직선을 표현하려면 기울기 a 값이 무한대가 되어야 하기 때문임
  - 허프 변환 구현 시 다음과 같이 극좌표계 형식의 직선의 방정식 사용

$$x\cos\theta + y\sin\theta = \rho$$

  - ρ 는 원점에서 직선까지의 수직 거리를 나타냄
  - θ 는 원점에서 직선에 수직선을 내렸을 때 x축과 이루는 각도를 의미함
  - xy 공간에서 한 점은 ρθ공간에서는 삼각함수 그래프 형태의 곡선으로 표현됨
  - ρθ 공간에서 한 점은 xy 공간에서 직선으로 나타나게 됨
  - 극좌표계 형식의 직선의 방정식을 사용하여 허프 변환을 수행할 경우에도 축적 배열을 사용함
  - 축적 배열에서 국지적 최댓값이 발생하는 위치에서의 ρ 와 θ 값을 찾아 직선의 방정식을 구할 수 있음

허프 변환 직선 검출

- 허프 변환 직선 검출



허프 변환 직선 검출

- OpenCV 함수

OpenCV를 통한 Hough변환

`cv2.HoughLines(image, rho, theta, threshold[, lines[, srn[, stn[, min_theta[, max_theta]]]]])` : Hough변환을 통하여 직선을 검출

parameter

- image: input(canny edge선 적용 후)
- rho: r값의 범위(0~1 실수)
- theata: θ값의 범위(0~180 정수)
- threshold: 만나는 점의 기준, 숫자가 적으면 많은 선이 검출되지만 정확도가 떨어지고, 숫자가 크면 정확도가 올라감

`cv2.HoughLinesP(image, rho, theta, threshold, minLineLength, maxLineGap)` : `cv2.HoughLines`는 모든 점에 대해서 계산하므로 시간이 오래걸린다. 따라서 모든 점을 대상으로 하는 것이 아니라 임의의 점을 이용하여 직선을 찾는 방법이다. 단, 임계값을 작게 하여야 한다. **선의 시작점과 끝점을 Return해주기 때문에 쉽게 화면에 표현할 수 있다.**

출처: https://wjddyd66.github.io/opencv/OpenCV(6)/#hough-%EB%B3%80%ED%99%98%EC%97%90-%EC%9D%98%ED%95%9C-%EC%A7%81%EC%84%A0-%EC%9B%90-%EA%B2%80%EC%B6%9C

- 실습

허프 변환 원 검출

- 허프 변환 원 검출
  - 중심 좌표가 (a,b)이고 반지름이 r인 원의 방정식은 다음과 같이 표현함

$$(x-a)^2+(y-b)^2=r^2$$

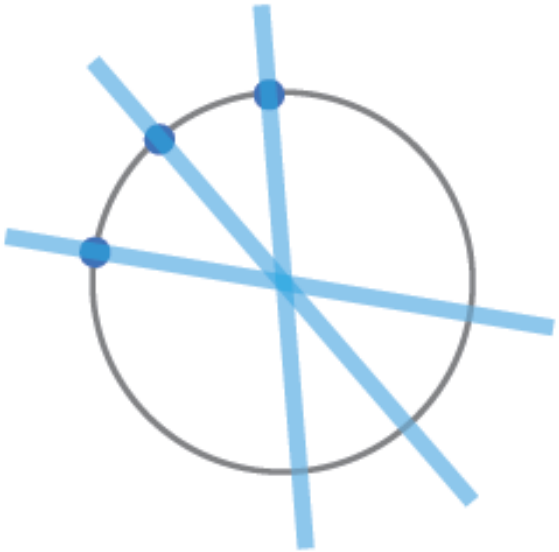
  - 이러한 원의 방정식은 세 개의 파라미터를 가지고 있음
  - 허프 변환을 그대로 적용하려면 3차원 파라미터 공간에서 축적 배열을 정의함
  - 가장 누적이 많은 위치를 찾아야 함
  - 3차원 파라미터 공간에서 축적 배열을 정의하고 사용하려면 너무 많은 메모리와 연산 시간을 필요로 하게 됨
  - OpenCV에서는 일반적인 허프 변환 대신 허프 그래디언트 방법(Hough gradient method)을 사용하여 원을 검출함

허프 변환 원 검출

- 허프 변환 원 검출
  - 허프 그래디언트 방법은 두 가지 단계로 구성됨
    - 첫 번째 단계에서는 영상에 존재하는 모든 원의 중심 좌표를 찾고
    - 두 번째 단계에서는 검출된 원의 중심으로부터 원에 적합한 반지름을 구함
  - 원의 중심 좌표를 찾는 과정에서 축적 배열이 사용됨
  - 허프 그래디언트 방법에서 사용하는 축적 배열은 파라미터 공간에서 만드는 것이 아니라 입력 영상과 동일한 xy 좌표 공간에서 2차원 배열로 만듦
  - 원의 중심을 찾기 위해 허프 그래디언트 방법은 입력 영상의 모든 에지 픽셀에서 그래디언트를 구함
  - 그래디언트 방향을 따르는 직선상의 축적 배열 값을 1씩 증가시킴

허프 변환 원 검출

- 허프 변환 원 검출
  - 원주상의 모든 점에 대해 그래디언트 방향의 직선을 그림
  - 직선상의 축적 배열 값을 증가시키면 결과적으로 원의 중심 위치에서 축적 배열 값이 크게 나타나게 됨
  - 일단 원의 중심을 찾은 후에는 다양한 반지름의 원에 대해 원주상에 충분히 많은 에지 픽셀이 존재하는지 확인하여 적절한 반지름을 선택함



허프 변환 직선 검출

- OpenCV 함수

```
cv2.HoughCircles(image, method, dp, minDist[, circles[, param1[,
param2[, minRadius[, maxRadius]]]]])
```

$$(x - x_{center})^2 + (y - y_{center})^2 = r^2$$

3개의 변수에 대하여 계산하는 것은 비효율적이므로 가장자리에서 기울기를 측정하여 원을 그리는데 관련이 있는 점인지를 확인할 수 있는 Hough Gradient Method를 사용

parameter

- image: input(canny edge선 적용 후)
- method: 검출방법, ex)HOUGH_GRADIENT
- dp: 해상도
- minDist: 검출한 원의 중심과의 최소거리. 값이 작으면 원이 아닌 것들도 검출이 되고, 너무 크면 원을 놓칠 수 있음
- param1: 내부적으로 사용하는 canny edge 검출기에 전달되는 Paramter
- param2: 이 값이 작을 수록 오류가 높아짐. 크면 검출률이 낮아짐.
- minRadius: 원의 최소 반지름.
- maxRadius: 원의 최대 반지름.

출처: https://wjddyd66.github.io/opencv/OpenCV(6)/#hough-%EB%B3%80%ED%99%98%EC%97%90-%EC%9D%98%ED%95%9C-%EC%A7%81%EC%84%A0-%EC%9B%90-%EA%B2%80%EC%B6%9C

- 실습

화이트 보드

영상처리 프로그래밍 기초

- Python으로 배우는 OpenCV 프로그래밍
  - 김동근 지음
  - 가메출판사, 2018

- OpenCV4 로 배우는 컴퓨터 비전과 머신러닝
  - 황선규 지음
  - 길벗, 2019