Computer Vision and Deep Learning

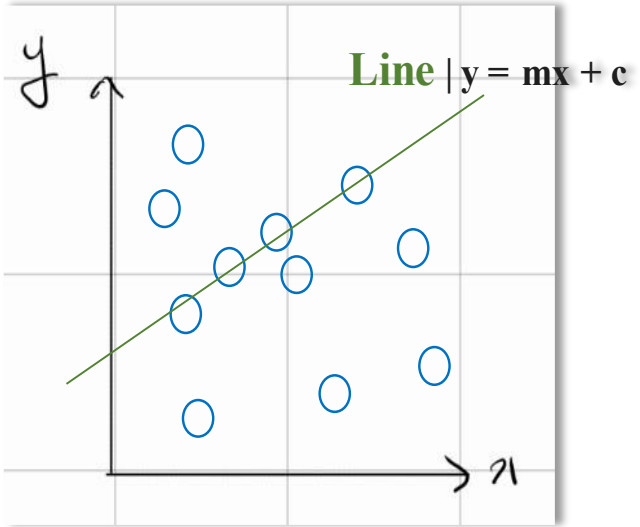
Ch 04

강명묵

Contents

Hough Transform

1. 다양한 **Noise**에서 **진짜 Edge**를 찾아가는 과정
2. **경계선**을 **적은 수의 파라미터**로 표현 가능



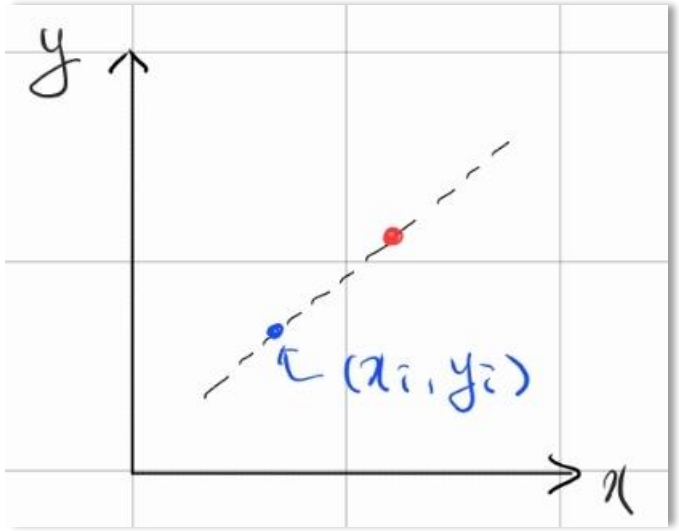
Given : Edge Point

Task : Detect line

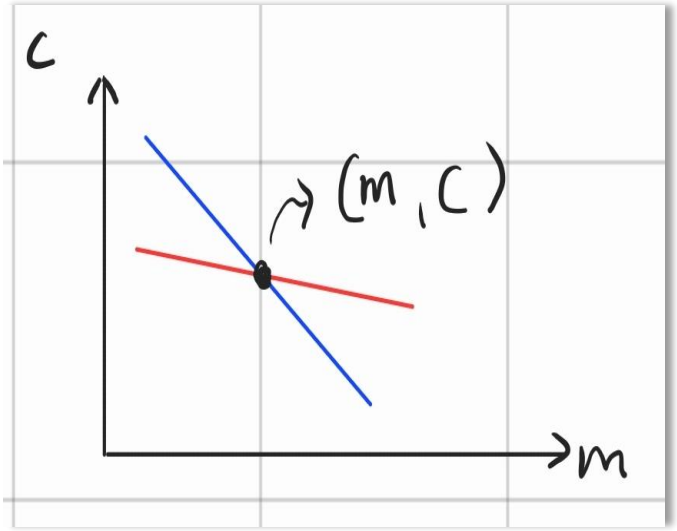
Consider point (x_i, y_i)

Hough Transform(Line)

Image Space



Parameter Space



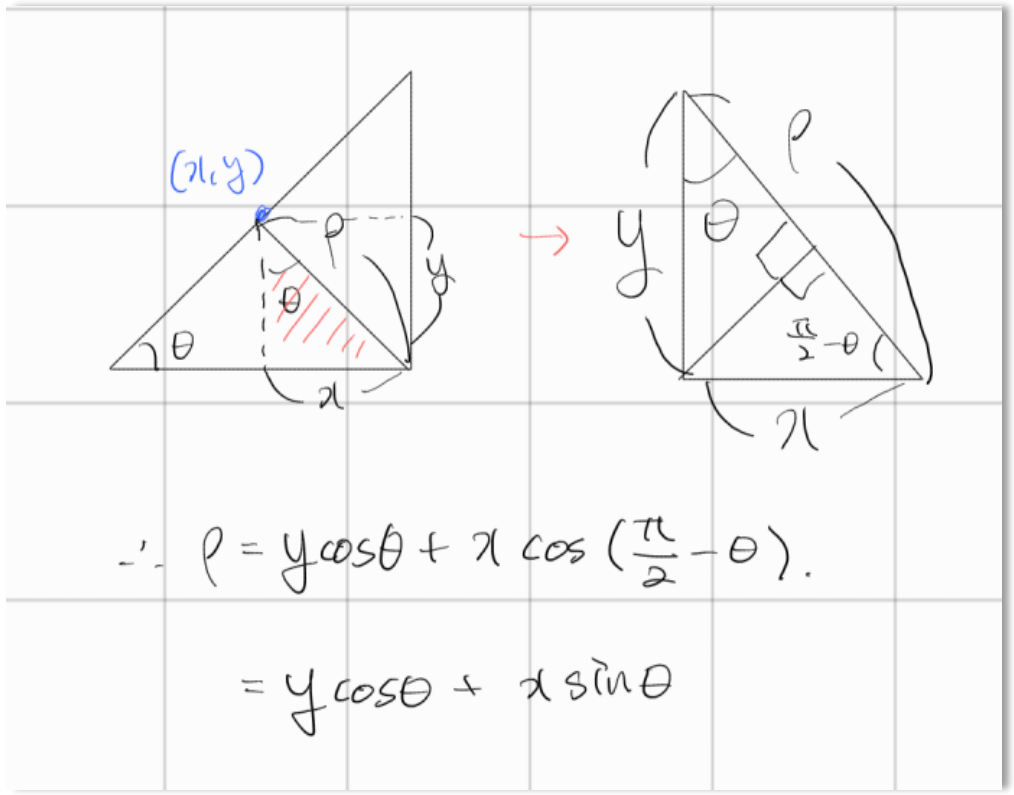
$y_i = mx_i + c$

$c = -mx_i + y_i$

Line ⟷ **Point**

Point ⟷ **Line**

Hough Transform(Line)

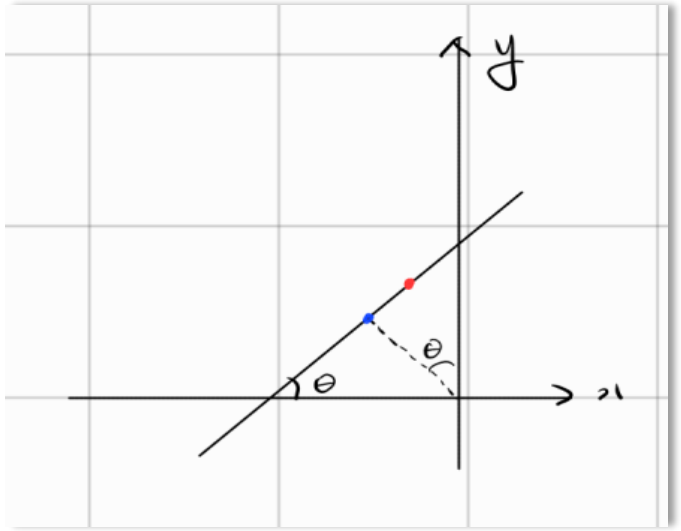


$$\therefore \rho = y\cos\theta + x\cos\left(\frac{\pi}{2} - \theta\right).$$

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

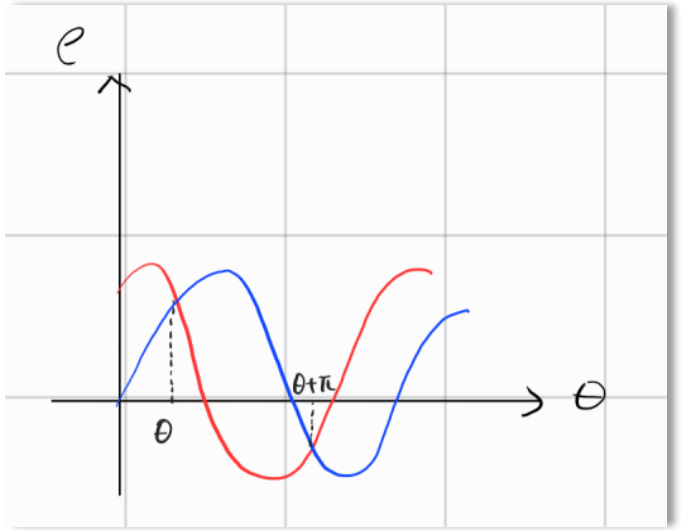
Hough Transform(Line)

Image Space



$$\rho = x_i \sin\theta + y_i \cos\theta$$

Parameter Space



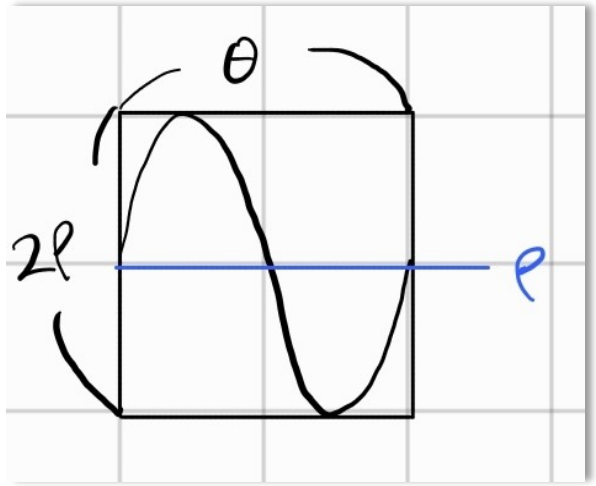
$$\rho = x_i \sin\theta + y_i \cos\theta$$

Line ⟷ **Point**

Point ⟷ **Line**

Implementation

Step 1
파라미터 공간 생성 및 투표

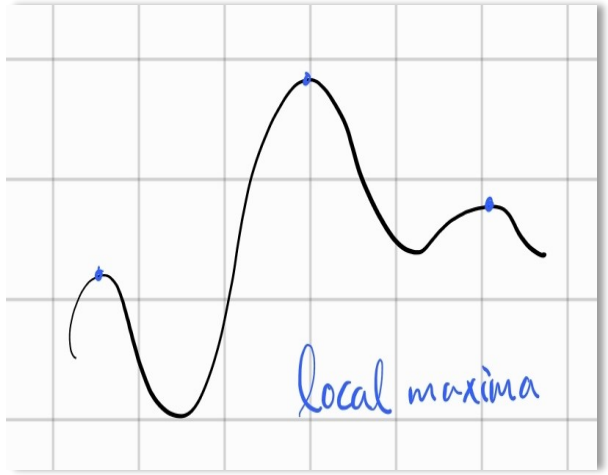


$0 \leq \theta \leq \pi$
θ를 1° 간격으로 이산화

```
Mat hough_space_line(Mat& img) {
    double max_rho = sqrt(pow(img.cols, 2) + pow(img.rows, 2));
    int max_rho_idx = cvRound(max_rho);
    Mat parameter_space = Mat::zeros(2 * max_rho_idx + 1, 180, CV_8UC1);

    for (int x = 0; x < img.cols; ++x) {
        for (int y = 0; y < img.rows; ++y) {
            if (img.at<uchar>(y, x) != 0) {
                for (int theta = 0; theta < 180; ++theta) {
                    // 변환된 파라미터 공간에서 직선의 방정식을 계산
                    double rho = x * cos(theta * CV_PI / 180) + y * sin(theta * CV_PI / 180);

                    // 직선이 이미지 내에 있는지 확인하고, 있다면 해당 위치의 값을 +1
                    int rhoIdx = cvRound(rho) + max_rho_idx;
                    if (rhoIdx >= 0 && rhoIdx < parameter_space.rows && (parameter_space.at<uchar>(rhoIdx,theta) != 255)) {
                        parameter_space.at<uchar>(rhoIdx, theta)++;
                    }
                }
            }
        }
    }

    return parameter_space;
}
```

Implementation

Step 2
지역 최댓값 찾기



Find local maxima

```
void findLocalMaxima(const Mat& src, Mat& dst, int neighborhoodSize, int threshold) {
    dst = Mat::zeros(src.size(), CV_8UC1);

    for (int y = 0; y < src.rows; ++y) {
        for (int x = 0; x < src.cols; ++x) {
            int value = src.at<uchar>(y, x);
            bool isMaxima = true;

            for (int i = -neighborhoodSize; i <= neighborhoodSize; ++i) {
                for (int j = -neighborhoodSize; j <= neighborhoodSize; ++j) {
                    int neighborRow = y + i;
                    int neighborCol = x + j;

                    if (neighborRow >= 0 && neighborRow < src.rows && neighborCol >= 0 && neighborCol < src.cols) {
                        if (src.at<uchar>(neighborRow, neighborCol) > value) {
                            isMaxima = false;
                            break;
                        }
                    }
                }
                if (!isMaxima) {
                    break;
                }
            }

            if (isMaxima && value >= threshold) {
                dst.at<uchar>(y, x) = 255;
            }
        }
    }
}
```

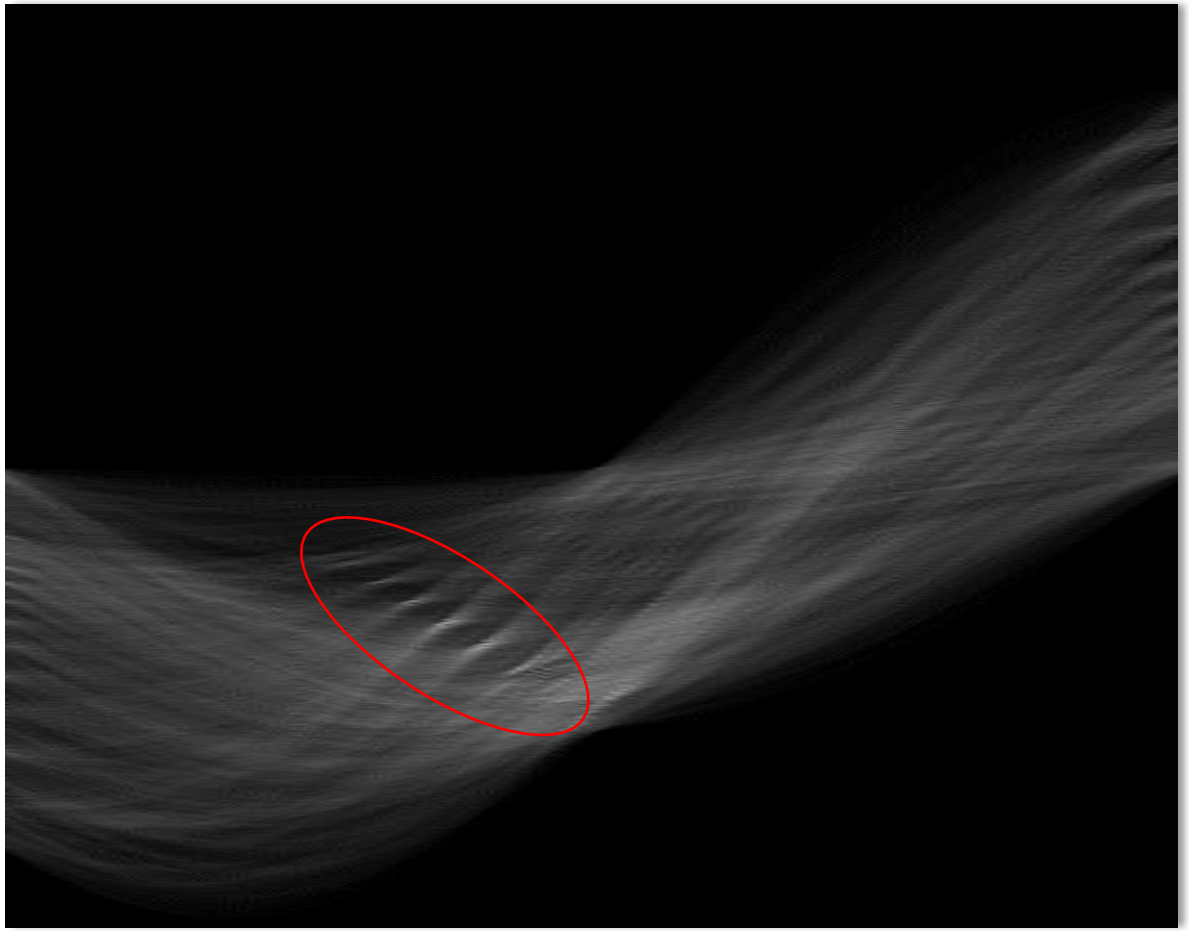
Implementation

Step 3
이미지에 직선 그리기

```
Mat draw_detected(Mat& img, const Mat& hough_space, int threshold) {
    Mat draw_img = img.clone();
    for (int rhoIdx = 0; rhoIdx < hough_space.rows; ++rhoIdx) {
        for (int theta = 0; theta < hough_space.cols; ++theta) {
            if (hough_space.at<uchar>(rhoIdx, theta) > threshold) {
                // 변환된 파라미터 공간에서 직선의 방정식으로 역변환
                double rho = rhoIdx - hough_space.rows / 2.0;
                double theta_rad = theta * CV_PI / 180.0;
                double a = cos(theta_rad);
                double b = sin(theta_rad);
                double x0 = rho * a;
                double y0 = rho * b;

                // 좌표 계산
                Point pt1(cvRound(x0 + 1000 * (-b)), cvRound(y0 + 1000 * (a)));
                Point pt2(cvRound(x0 - 1000 * (-b)), cvRound(y0 - 1000 * (a)));

                // 직선 그리기
                line(draw_img, pt1, pt2, Scalar(0, 0, 255), 2, LINE_AA);
            }
        }
    }
    return draw_img;
}
```

Implementation

Parameter space

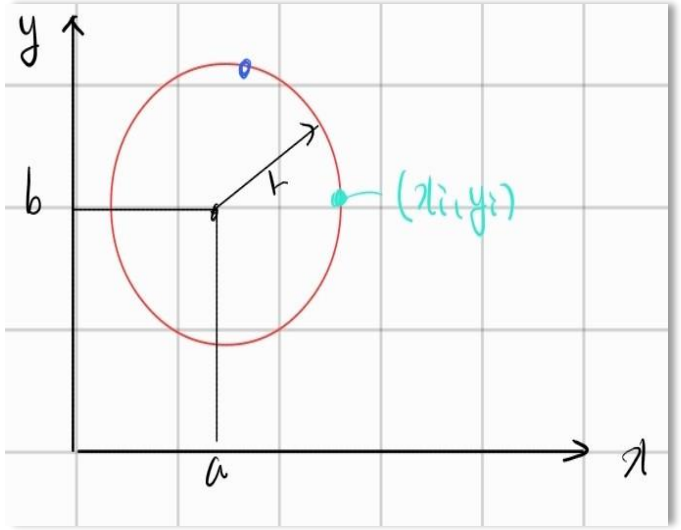


Parameter space (thresholding)

Implementation

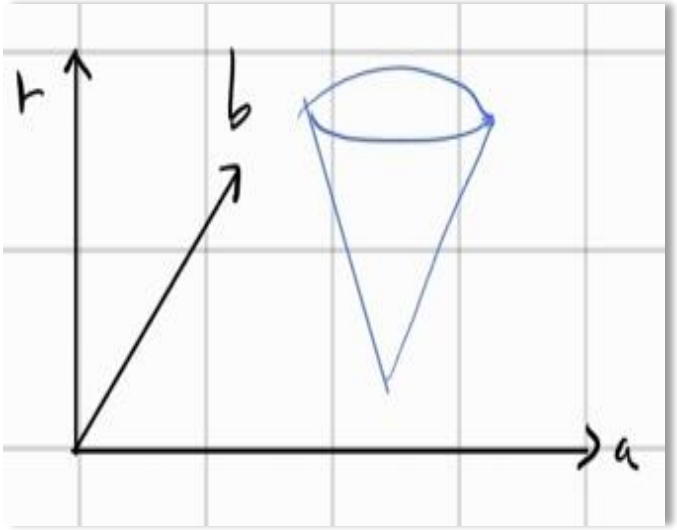
Hough Transform(Circle)

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

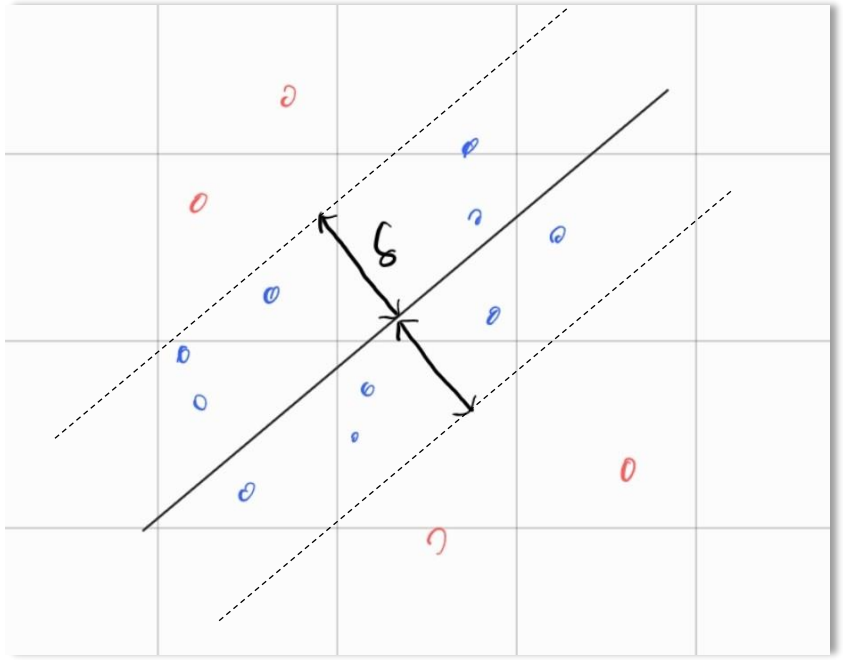
RANSAC

Random sample consensus
관측된 데이터들에 대하여 근사 모델의 파라미터를 **추정하는 방법 중 하나.**

1. 모델 추정
이미지에서 랜덤한 두 점 선택(직선의 방정식)

2. 모델 검증
직선의 방정식으로부터
δ의 범위 안에 있는 점 : inlier
δ의 범위 밖에 있는 점 : outlier

3. 반복
iteration만큼 1,2번을 반복하며
inlier가 최대가 되는 지점 찾기



inlier, outlier

감사합니다.