차선 검출

SSL

색 정보 이용

1. RGB 색 좌표계를 이용한 흰색 추출

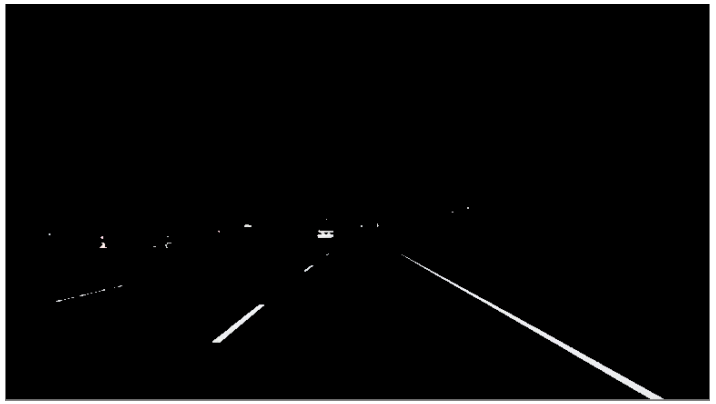
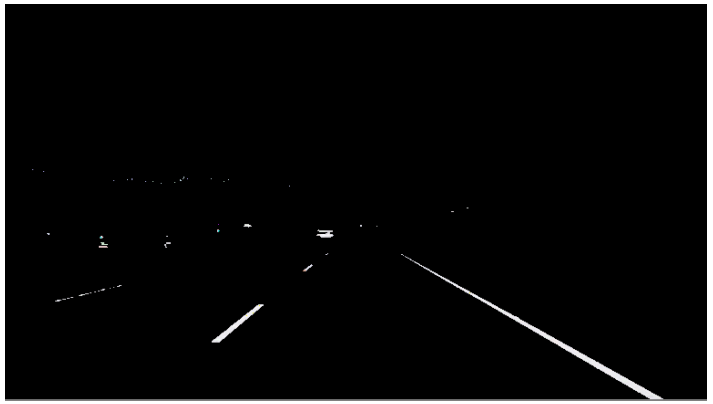
```
void GetwhiteRGB(PPMdata *ppm) {

    BYTE R, G, B;
    BYTE threshold = 200;

    for (int i = 0; i < ppm->height; i++)
        for (int j = 0; j < ppm->width * 3; j += 3)
        {
            R = ppm->pixels[i][j];
            G = ppm->pixels[i][j + 1];
            B = ppm->pixels[i][j + 2];

            if (R < threshold || G < threshold || B < threshold)
            {
                ppm->pixels[i][j] = 0;
                ppm->pixels[i][j + 1] = 0;
                ppm->pixels[i][j + 2] = 0;
            }
        }

}
```

2. HSI 색 좌표계를 이용한 흰색 추출

```
void extractwhite(PPMdata *ppm) {
    float I = 0;

    for(int i = 0; i < ppm->height; i++)
        for(int j =0; j < ppm->width*3; j+=3)
        {
            I = ppm->HSI[i][j + 2];
            if (I < 200)
                ppm->HSI[i][j + 2] = 0;
        }
    }
```

SSL

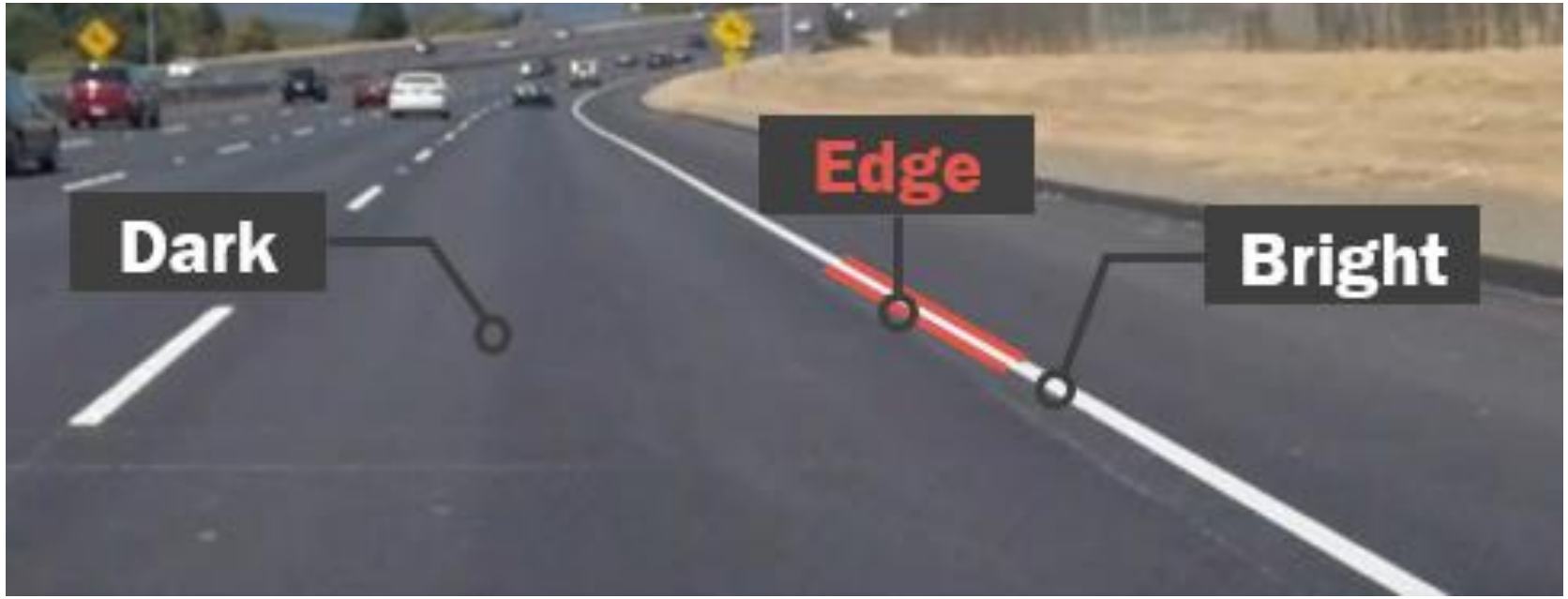
색 정보 이용 결론

색 정보만으로 차선 추출은 불가능하다.
(색정보로 흰색을 추출하면 흰색 중 차선을 뽑아낼 수 없다.)

직선 특징 이용

⇒ Edge 추출을 통한 차선을 검출



SSL

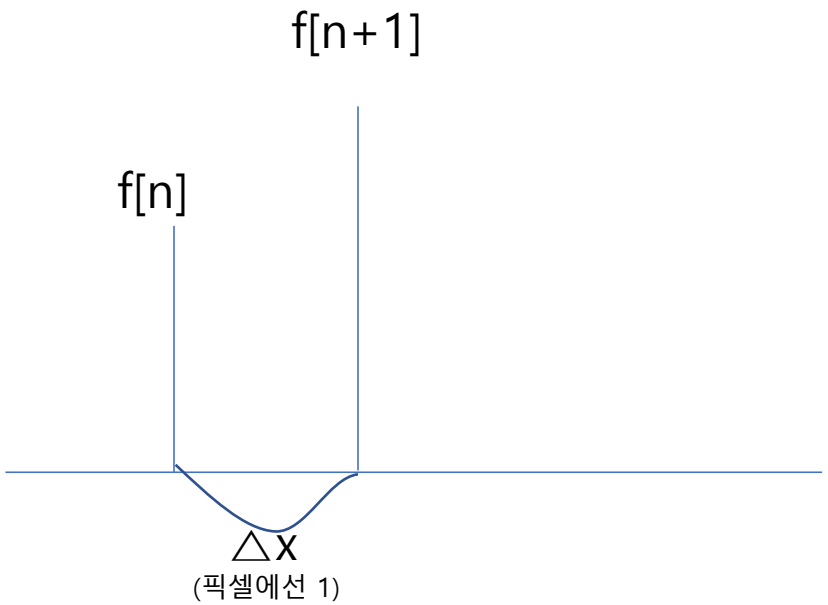
Edge 검출 방법

1. 1차 미분
 - Sobel 마스크

2. 2차 미분
 - 라플라시안 마스크

-Canny Edge 검출

-1차 미분



$$\frac{f[n+1]-f[n]}{1} = f[n+1] - f[n]]$$

픽셀 I 의 변화량은

$$\nabla I(m, n) = \sqrt{\nabla x(m, n)^2 + \nabla y(m, n)^2}$$
$$= |\nabla x(m, n)| + |\nabla y(m, n)|$$

$$\nabla x(m, n) = |I(m+1, n) - I(m, n)|$$
$$\nabla y(m, n) = |I(m, n+1) - I(m, n)|$$

SSL

1. 1차 미분

- 소벨 연산

(m-1, n-1)	(m, n-1)	(m+1, n-1)
(m-1, n)	(m, n)	(m+1 ,n)
(m-1, n+1)	(m, n+1)	(m+1, n+1)

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

$$\nabla x(m, n) = [I(m+1, n+1)) + 2 * I(m+1, n) + I(m+1, n-1)] - [I(m-1, n+1)] + 2I(m-1, n) + I(m-1, n-1)]$$
$$\nabla y(m, n) = [I(m+1, n+1)) + 2 * I(m, n+1) + I(m-1, n+1)] - [I(m+1, n-1)] + 2I(m, n-1) + I(m-1, n-1)]$$

$$I(m, n) = \nabla x(m, n) + \nabla y(m, n)$$

SSL

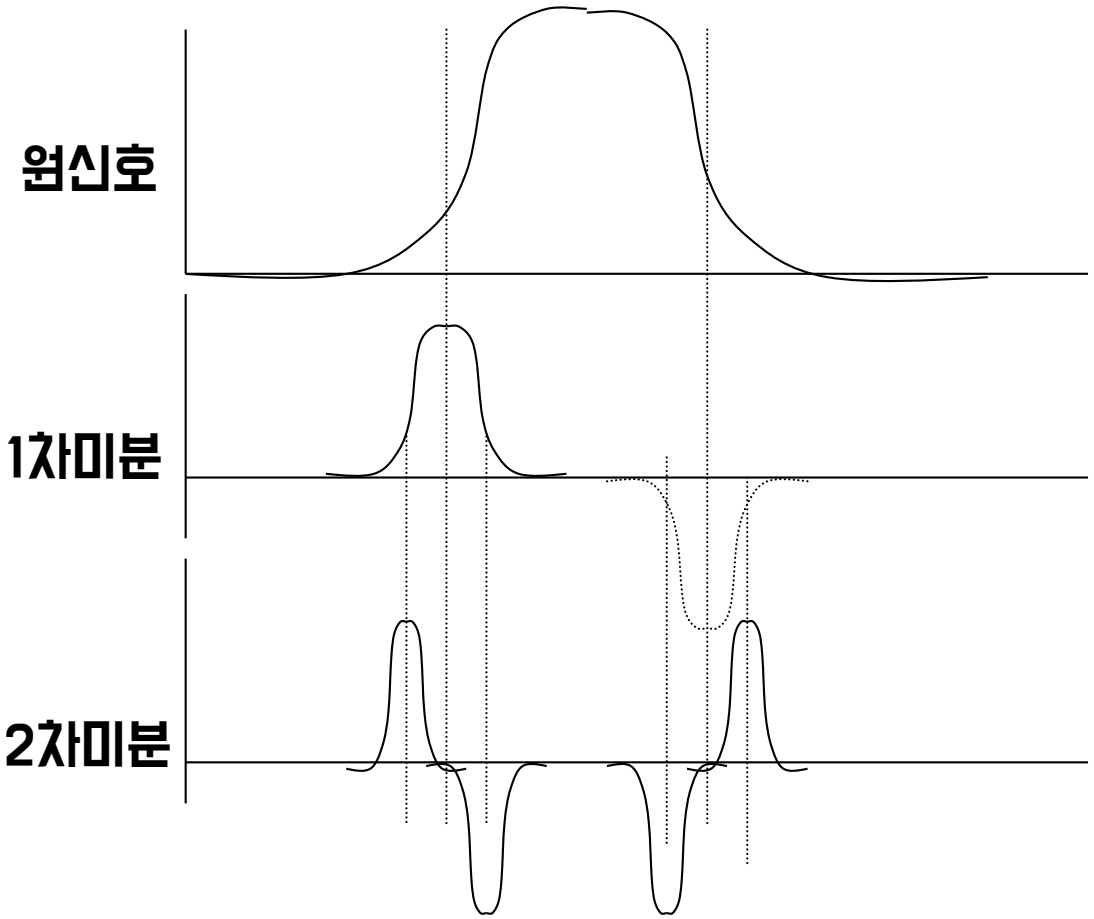
- Sobel 마스크를 이용한 1차 미분

```
int maskX[3][3] = { {-1, -2, -1}, {0, 0, 0}, {1, 2, 1} };
int maskY[3][3] = { {-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1} };

int Xsum = 0;
int Ysum = 0;
int sum = 0;

for (int i = 0; i < H-2; i++)
    for (int j = 0; j < W-2; j++){
        for(int k = 0; k < 3; k++)
            for (int e = 0; e < 3; e++)
            {
                Xsum += Img[i + k][j + e] * maskY[k][e];
                Ysum += Img[i + k][j + e] * maskY[k][e];
            }

        sum = (abs(Xsum) + abs(Ysum));

        if (sum > 255) {
            sum = 255;
        }
        else if (sum < 0) {
            sum = 0;
        }
        Img2[i+1][j+1] = sum;

        Xsum = 0;
        Ysum = 0;
    }
```

Edge가 두꺼워서 어디까지가 edge인지를 찾아야함

⇒ Object인식에는 적합하지 않다.

2. 2차 미분



1차 미분으로는 너무 많은 Edge후보 생성

실제 Edge점은 1차 미분의 최대값
=> 2차 미분하면 0인 것이 1차 미분의 최대값

2. 2차 미분

-라플라시안 연산

(m-1, n-1)	(m, n-1)	(m+1, n-1)
(m-1, n)	(m, n)	(m+1 ,n)
(m-1, n+1)	(m, n+1)	(m+1, n+1)

0	1	0
1	-4	1
0	1	0

$\nabla^2 I(x, y) = Ixx(x, y) + Iyy(x, y)$

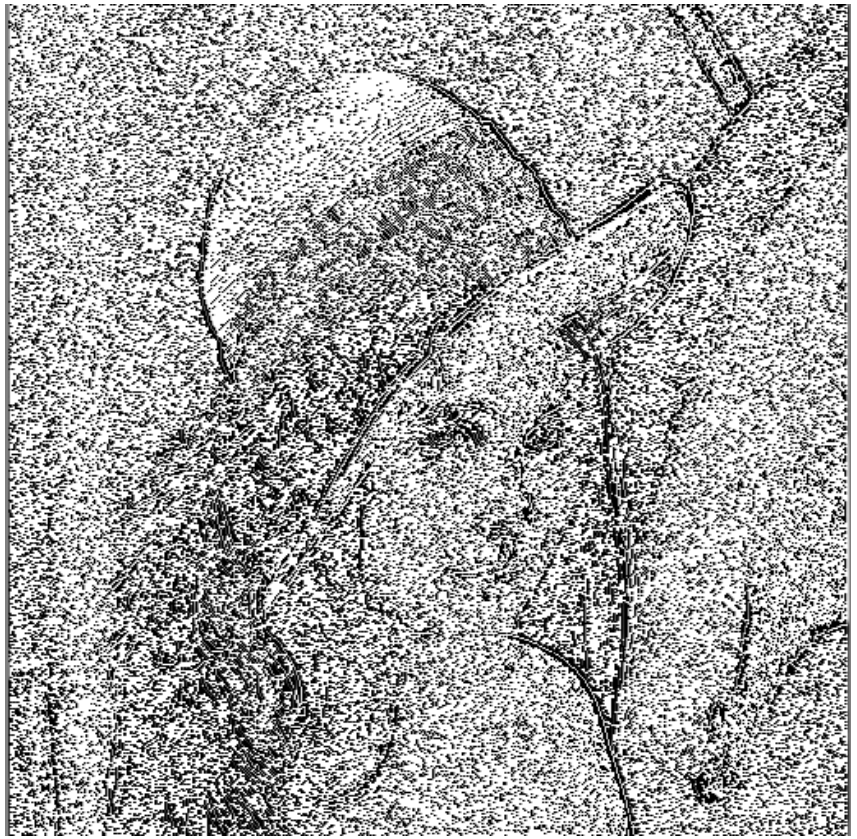
$Ixx(m, n) = Ix(m, n) - Ix(m-1, n)$
$Ix(m, n) = I(m+1, n) - I(m, n)$
$Iyy(m, n) = Iy(m, n) - Iy(m, n-1)$
$Iy(m, n) = I(m, n+1) - I(m, n)$

$\nabla^2 I(x, y) = I(m+1, n) + I(m, n+1) + I(m-1, n) + I(m, n-1) -4I(m, n)$

SSL

- 라플라시안 마스크를 이용한 2차 미분

```
// Laplacian Mask
int mask[3][3] = { { 0, 1, 0 }, { 1, -4, 1 }, { 0, 1, 0 } };
int sum = 0;

for (int i = 0; i < H - 2; i++)
for (int j = 0; j < W - 2; j++){
    for (int k = 0; k < 3; k++)
    for (int e = 0; e < 3; e++)
    {
        sum += Img[i + k][j + e] * mask[k][e];
    }

    Img2[i + 1][j + 1] = sum;
    sum = 0;
}

//Zero-crossing detection

for(int i = 0; i < H-2; i++)
for (int j = 0; j < W-2; j++)
{
    if (((Img2[i + 1][j + 1] * Img2[i + 2][j + 1]) < 0 ) || ((Img2[i + 1][j + 1] * Img2[i + 1][j + 2]) < 0))
        Img[i + 1][j + 1] = 255;
    else
        Img[i + 1][j] = 0;
}
```



Edge가 1개의 픽셀크기라서 얇은 Edge

=> Object인식에 적합 하지만 잡음에는 민감

SSL

Canny Edge 검출

1) 가우시안 필터를 이용한 노이즈 제거

SSL

2) 미분을 통한 Edge 검출

SSL

3) 비최대치 억제

1	2	2	4	7
2	2	6	7	5
2	6	7	6	2
5	7	6	2	1
7	5	2	1	1

1	2	2
2	2	6
2	6	7

2	6	7
6	7	6
7	6	2

4방향의 원소들보다 크면 자신이 Edge 그렇지 않으면 Edge가 아닌 것으로 간주

=> 얇은 Edge를 얻음

3) 비최대치 억제 결과

SSL

4-1) 임계값 사용

비최대치 억제 후에도 실제 Edge와 약간의 노이즈는 검출됨
이를 구별하기위해 2개의 임계 값 Low와 High를 사용
Low와 High를 기준으로 파란색 영역은 제거하고
주황색 영역(약한 Edge)과 빨간색 영역(강한 Edge)를 구분함

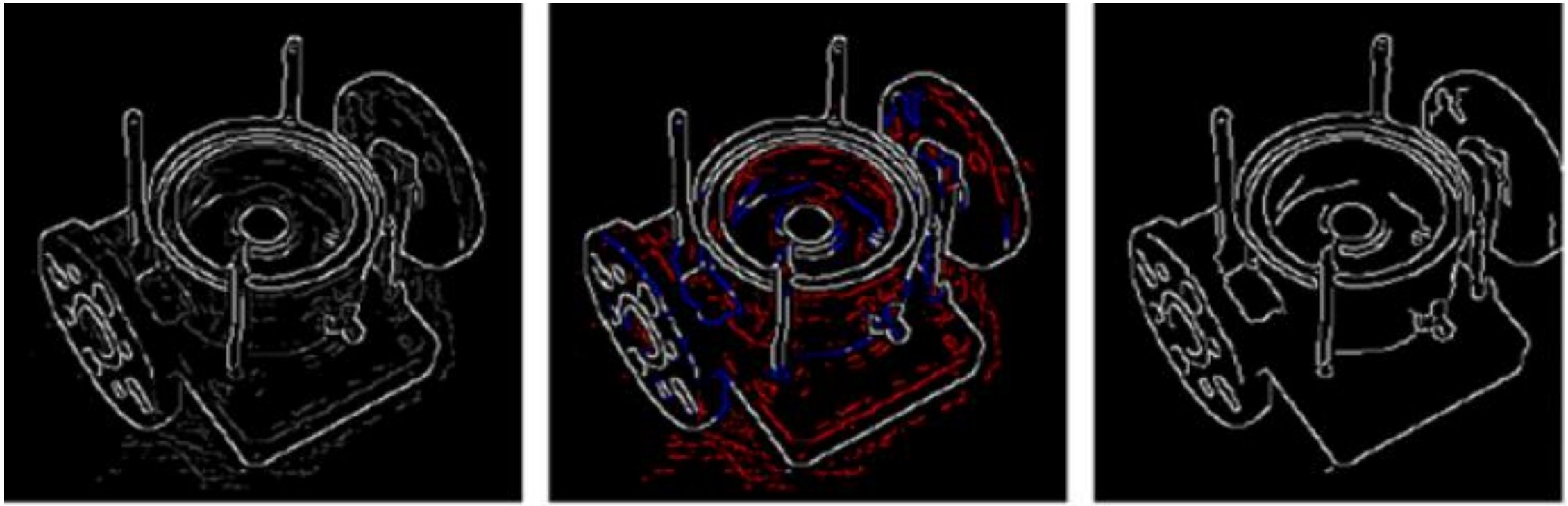
4-2) 약한 Edge 연관성 판별

강한 Edge는 최종 Edge에 추가

A	B	C
D	(m, n)	E
F	G	H

주황색 영역의 Edge는 강한 Edge와 연결된 경우에만 추가
⇒노이즈나 작은 변화량은 강한 Edge와 연관성이 떨어지므로

3) Edge연관 판별 및 Canny Edge 검출 결과



검출된 Edge를 통한 직선 검출

수식으로 표현할 수 있는 도형이라면 검출할 수 있다

=> Hough Trasnform 함수

참고문헌

1] https://blog.naver.com/windowsub0406/220892704332
(UDACITY Self-Driving Car nanodegree 강의 해설)

2] http://carstart.tistory.com/188
(Canny Edge 검출 설명)

3] 임베디드 소프트웨어 경진대회 Team 금호우 개발계획서

4] OpenCV로 배우는 영상 처리 및 응용