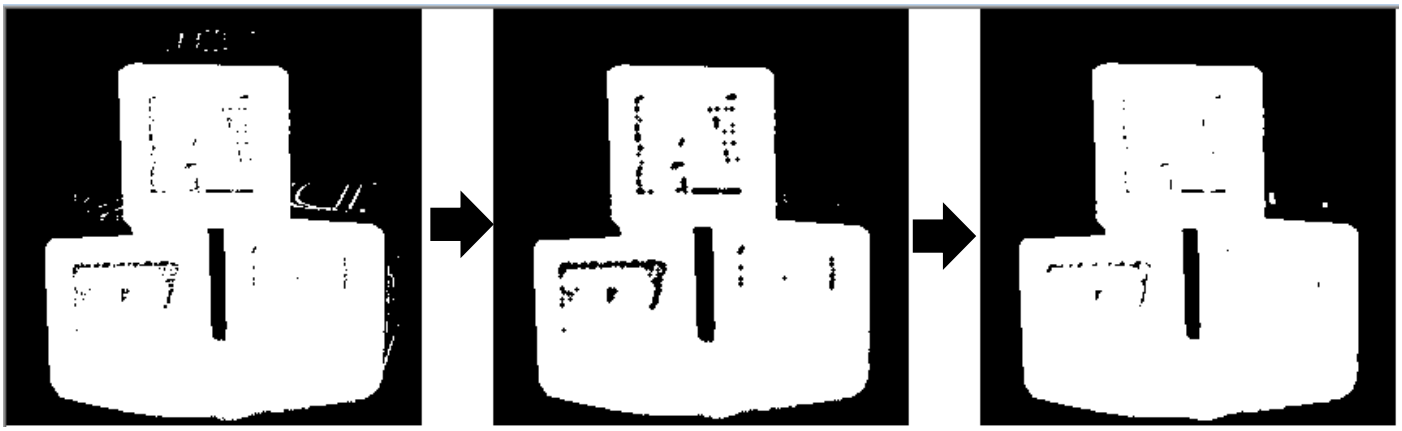
디지털 영상처리 연구실 연구보고서

김우헌

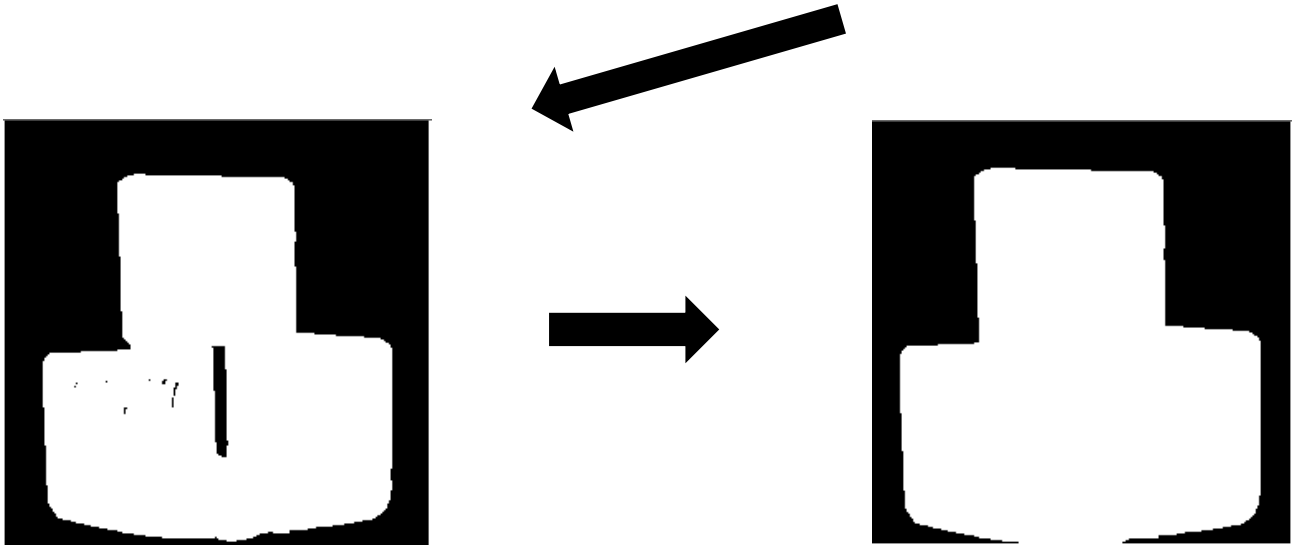
##모폴로지

1	1	1
1	1	1
1	1	1

침식과 팽창연산을 위한 마스크



->팽창 ->침식



->침식연산 3회실시 후 모습 -> 침식연산 5회 실시 후 모습잡음이 완전히 사라진 모습

➔ 팽창연산후 침식연산 효과로 제거연산 역할 하는 것을 확인 하였습니다.

##canny edge 알고리즘

#그래디언트?

-> 프리윗필터,소벨필터는 x축,y축방향 으로 각각 편미분을 사용하는데 이두개를 하나의 벡터로 표현한 것이 그래디언트입니다.

1	0	0	1
0	-1	-1	0

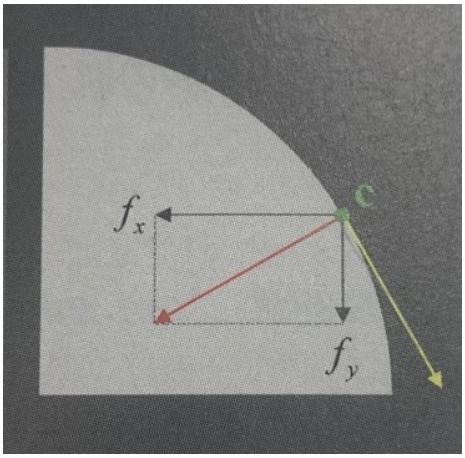
Robert Mask

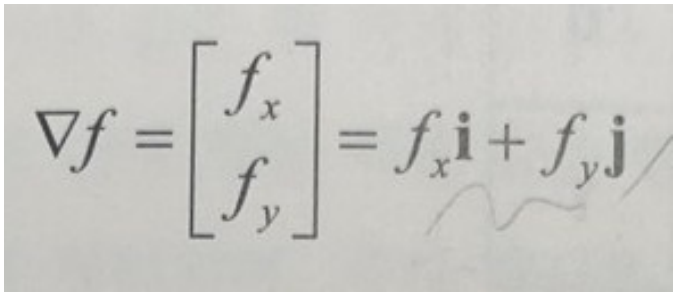
-1	0	1	1	1	1
-1	0	1	0	0	0
-1	0	1	-1	-1	-1

Prewitt Mask

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

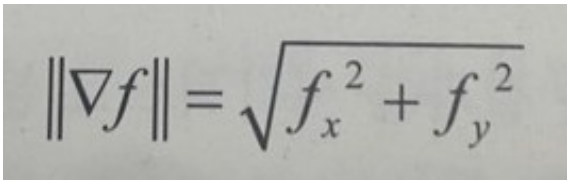
Sobel Mask



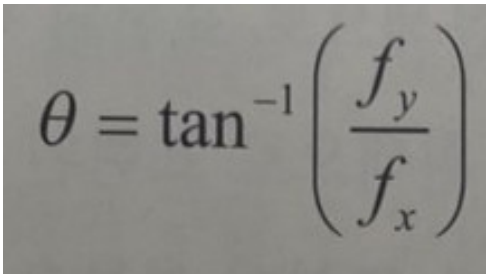


$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = f_x \mathbf{i} + f_y \mathbf{j}$$

->이때의 그래디언트 크기와 각도는



$$\|\nabla f\| = \sqrt{f_x^{\,2} + f_y^{\,2}}$$



$$\theta = \tan^{-1}\left(\frac{f_y}{f_x}\right)$$

➔ 그래디언트 벡터의 방향은 해당위치에서 밝기가 가장 밝아지는 방향

➔ 그래디언트 벡터의 수직인 방향=엣지의 방향

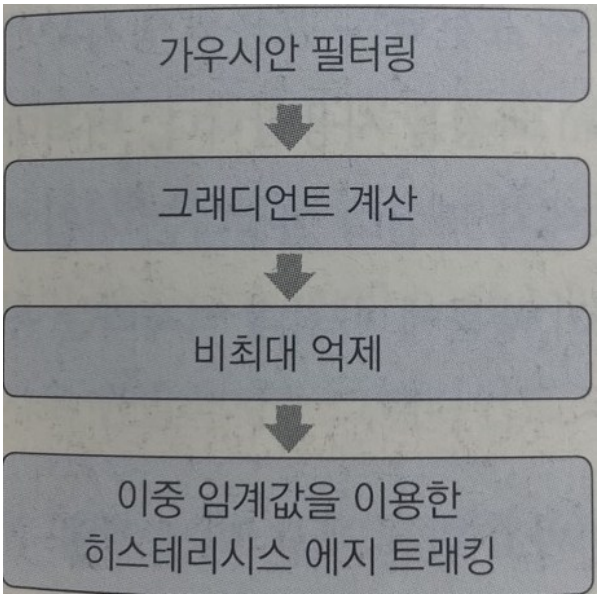
#소벨 필터

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

Sobel Mask

->edge가 두껍게 나타나고 정확하지 않음(그래디언트 크기만 이용)

#케니엣지 수행과정



->이 과정을 통해 정확한 검출과 위치, 얇은 엣지를 추출 할 수 있는 케니 엣지 검출 알고리즘

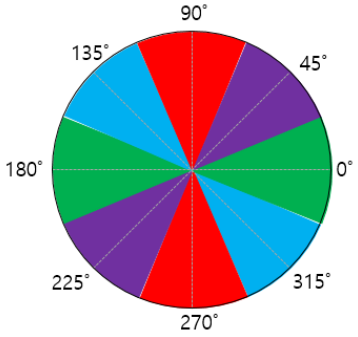
1. 가우시안 필터링

➔ 잡음 제거 목적 -> 적절한 표준편차 이용하여 필터링

2. 그래디언트 계산

➔ 소벨 필터는 그래디언트의 크기로만 엣지를 탐색

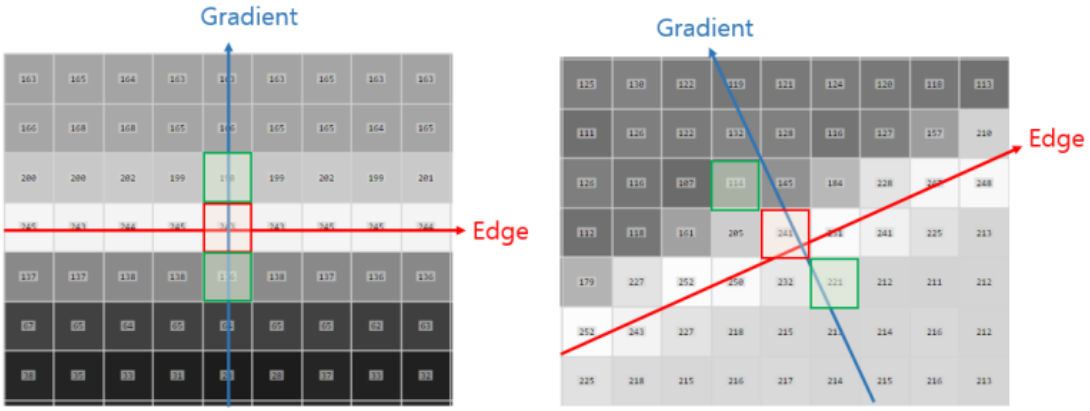
➔ 케니 엣지 탐색은 소벨필터로 필터링 한 후 그래디언트 크기와 방향 모두 고려



-> 그래디언트 방향은 (0,45,90,135)로 그룹을 만들어 단순화

3. 비최대억제 (NMS)

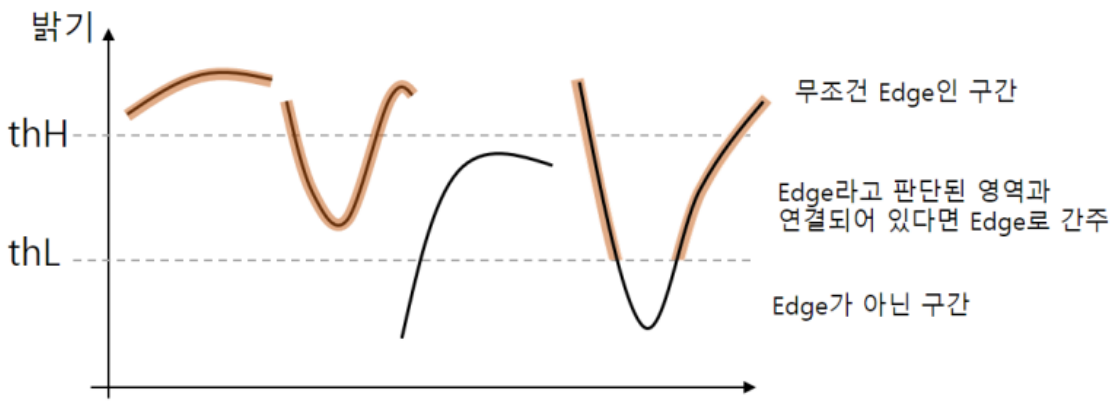
➔ EDGE의 두께를 얇게하기 위해서 에지가 아닌 픽셀을 제거하는 방법



➔ 현재 화소보다 그래디언트 크기가 크지 않으면 제거되는 방식

4. 이중 임계값을 통한 히스테리시스 에지 트래킹

➔ 실제 에지인지 아닌지 판단하는 단계 ->두개의 경계값으로 에지 판단



#C++을 사용한 CANNY EDGE

1. 가우시안 마스크

```
int MaskBox[3][3] = { {1,5,1},
                      {5,15,5},
                      {1,5,1} };
```

->가우시안 마스크를 적용하여 잡음제거

2. 그래디언트 계산

```
for (mr = 0; mr< 3; mr++)
                        for(mc=0;mc<3;mc++)
                        {
                                newvale1 += gx[mr][mc] * m_inimage[i + mr-1][j + mc-1];
                                newvale2 += gy[mr][mc] * m_inimage[i + mr-1][j + mc-1];

                        }

                sobelsum[i * width + j] = sqrt(newvale1 *newvale1 + newvale2 * newvale2);
```

➔ 소벨 필터 적용후 I,j위치에서 그래디언트 크기 (sobelsum) 구함

```
theta = atan2(newvale2, newvale1) * 180 / 3.141592;

if ((theta > -22.5&& theta >22.5)||theta > 157.5 || theta < -157.5)
        {
                sobelang[where] = 0;
        }
 else if ((theta >= 22.5 && theta < 67.5) || (theta >= -157.5 && theta < -112.5))
        {
                sobelang[where] = 45;
```

```
        }
        else if ((theta >= 67.5 && theta <= 112.5) || (theta >= -112.5 && theta <= -67.5))
                        {
                sobelang[where] = 90;
        }
        else
        {
                sobelang[where] = 135;
        }
```

➔ 그래디언트 각도 theta 를 구한 후 그룹화하여 단순화하여 sobelang[where]에 저장

3. 비최대 억제

```
switch (sobelang[where])
{
    case 0:
        if (sobelsum[where]>sobelsum[where-1]&&sobelsum[where]>sobelsum[where+1])
        {
            outimage[where] = 255;
        }

        break;
    case 45:
        if (sobelsum[where] > sobelsum[where-width+1] && sobelsum[where] > sobelsum[where + width- 1])
        {
            outimage[where] = 255;
        }

        break;
    case 90:
        if (sobelsum[where] > sobelsum[where - width] && sobelsum[where] > sobelsum[where + width])
        {
            outimage[where] = 255;
        }

        break;
    case 135:
        if (sobelsum[where]>sobelsum[where-width-1]&&sobelsum[where]>sobelsum[where + width+1])
        {
            outimage[where] = 255;
        }
```

➔ 그룹별로 outimage에 단일 엣지를 저장합니다.

->비최대 억제를 통하여 얇아진 엣지 추출

4. 이중 임계값을 통한 히스테리시스 에지 트래킹

```
if (outimage[where])
{
    if (sobelsum[where] > threshi)
    {
        outimage1[where] = 255;
    }
    else if (sobelsum[where] > threslo)
    {
        bool edge = TRUE;
        switch (sobelang[where])
        {
        case 0:
            if ((sobelsum[where - width] > threshi) || sobelsum[where + width] > threshi)
            {
                outimage1[where] = 255;
            }
            break;
        case 45:
            if ((sobelsum[where - width - 1] > threshi) || sobelsum[where + width + 1] > threshi)
            {
                outimage1[where] = 255;
            }
            break;

        case 90:
            if ((sobelsum[where - 1] > threshi) || sobelsum[where + 1] > threshi)
            {
                outimage1[where] = 255;
            }
            break;
        case 135:
            if ((sobelsum[where - width + 1] > threshi) || sobelsum[where + width - 1] > threshi)
            {
                outimage1[where] = 255;
            }
            break;
        }
    }
}
```

->비최대 억제를 통해 저장된값(outimage)이 255인 경우에 그래디언트 값이 thres hi보다 클경우 엣지

Thres hi보다 작고 thres lo보다 큰경우 엣지가 이어져 있으면 엣지라고 판단 !

-> Thres hi =70, thres low =30

➔ Thres hi =90, thres low =30

-> Thres hi =120, thres low =30

#open cv를 통한 canny edge 탐색

-> Thres hi =100, thres low =150

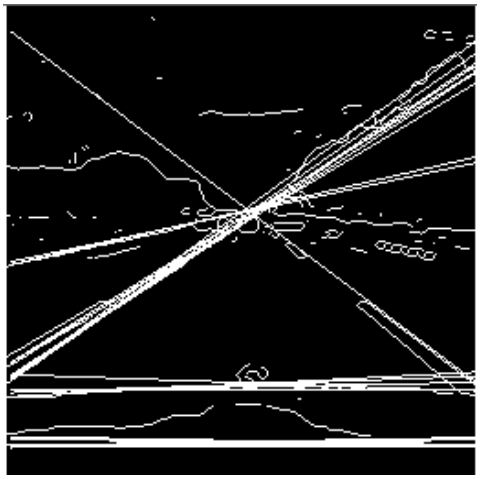
Thres hi =150, thres low =50

##허프 변환

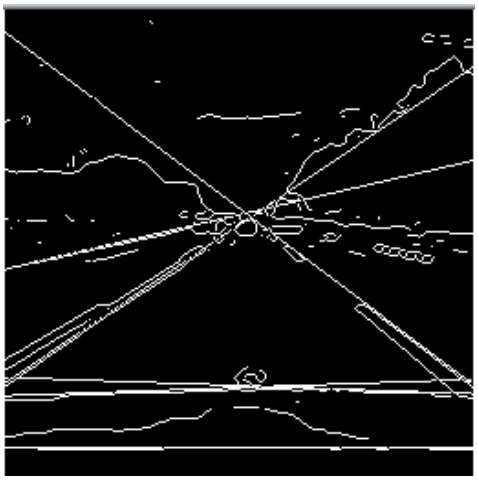
```
for (d = 4; d <= 360; d++)
      for ( k = 0; k < 360; k++)
      {
           for ( n = -3; n <=3; n++)
               for (m = -3; m <= 3; m++)
                {
                  if (k + n >= 0 && k + n < 360 && d + m >= 4 && d + m <= 360)
                       if (max < H[k + n][d + m])
                          {      max = H[k + n][d + m];   }

                 }
           for (n = -3; n <= 3; n++)
              for (m = -3; m <= 3; m++)
                      {
                         if (k + n >= 0 && k + n < 360 && d + m >= 4 && d + m <= 360)
                             if (max != H[k + n][d + m])
                                    H[k + n][d + m]=0;


                      }
                      max = 0;

      }
```

➔ 파라메타 공간 H[k][d] 에서 3x3공간내 최대값을 제외한 값들은 0으로 매핑(중복방지)

>원본 이미지



->Thres 60에 기존의 허프변환

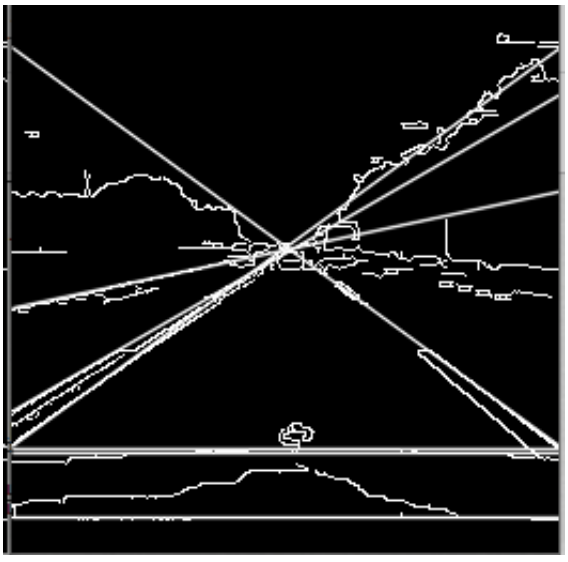


->파라메타 공간 수정후 thres60의 허프변환

#open cv를 이용한 허프변환

void HoughLines(src, lines, rho, theta, threshold, srn, stn, min_theta, max_theta)

src	입력 영상, 에지영상을 주로 입력함, 8UC1
lines	직선 식 정보(rho, theta), vector<Vec2f> 또는 vector<Vec3f> 타입 입력
rho	축적 배열에서의 rho 해상도(픽셀)|
theta	축적 배열에서 theta 해상도(rad 단위)
threshold	축적 배열에서의 직선 판단 임계치
srn	멀티 스케일 허프 변환에서 rho 해상도를 나누는 값, 기본값: 0, 만약 srn과 stn이 0이 아닌 값이 입력될 경우 멀티 스케일 허프 변환, 모두 0일 경우 표준 허프 변환 수행
stn	멀티 스케일 허프 변환에서 theta 해상도를 나누는 값, 기본값: 0
min_theta	검출할 직선의 최소 theta 값, 기본값: 0
max_theta	검출할 직선의 최대 theta 값, 기본값 : CV_PI

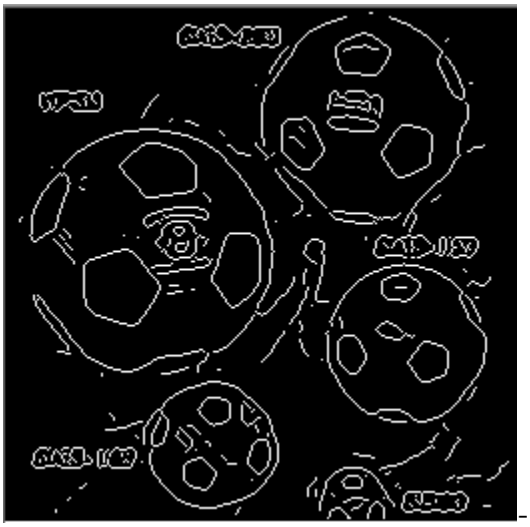


->HoughLines(edge, lines, 1, CV_PI / 30, 70) -> 6도 간격으로설정하여 중복 최소화, thres=70

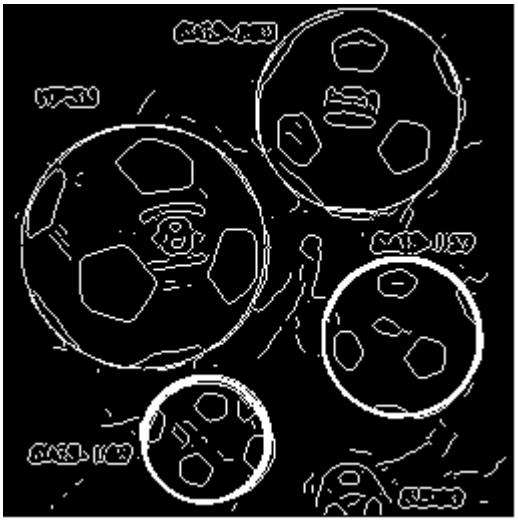
#원의 허프변환

```
for (rr= 0; rr < height; rr++)
    for (cc = 0; cc < width; cc++)
        for( k = r_min; k < r_max; k++)
        {
          for (n = -3; n <= 3; n++)
              for (m = -3; m <= 3; m++)
                  for (l = -3; l <= 3; l++)
              {
                  if (rr + n >= 0 && rr + n < height && cc + m >=0 && cc + m < width&&k+l>r_min&&k+l< r_max)

                      if (max < H[rr + n][cc + m][k+l])
                      {
                          max = H[rr + n][cc + m][k + l];
                      }


              }
             for (n = -3; n <= 3; n++)
              for (m = -3; m <= 3; m++)
                  for (l = -3; l <= 3; l++)
                 {
                    if( rr + n >= 0 && rr + n < height && cc + m >= 0 && cc + m < width && k + l>r_min && k + l< r_max)


                    if (max != H[rr + n][cc + m][k + l]) H[rr + n][cc + m][k + l] = 0;


                 }
             max = 0;

        }
```

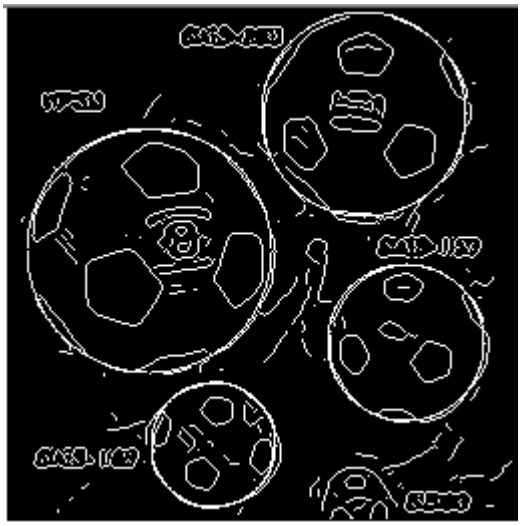
->직선의 허프변환과 마찬가지로 파라메타 공간H[i][j][k]에서 최대값을 제외한 값들을 0으로 매핑!



->원본 이미지



-> Thres 110에서의 기존 허프변환

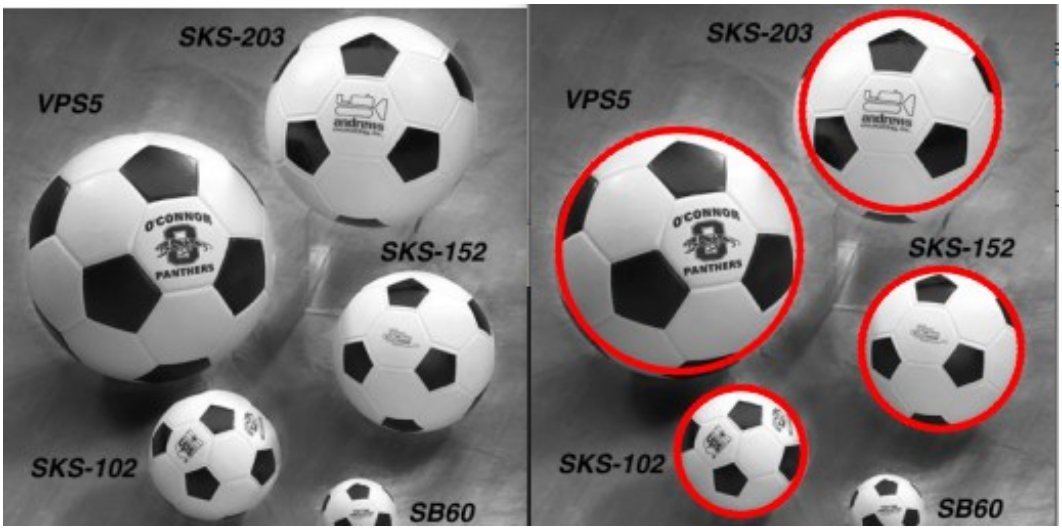


->파라메타 공간변화후 thres110에서의 허프변환

#OPEN CV에서의 허프변환

void HoughCircles(image, circles, method, dp, minDist, param1, param2, minRadius, maxRadius)

image	입력 영상, 에지 영상이 아닌 원본 grayscale 영상
circles	검출된 원 정보를 저장할 출력 벡터, vector<Vec3f>, vector<Vec4f>를 입력으로 함
method	HOUGH_GRADIENT만 지정 가능함
dp	입력 영상과 축적 배열의 크기 비율 dp = 1일 경우 축적배열의 크기와 영상과 같음 dp = 2일 경우 축적 배열의 크기가 영상의 가로,세로 크기를 2로 나눈 사이즈임
minDist	인접한 원 중심의 최소 거리
param1	Canny Edge 검출기의 높은 임계값, 낮은 임계값은 param1의 절반의 값으로 설정됨 기본값: 100
param2	축적 배열에서 원 검출을 위한 임계값 기본값: 100



-> dp1,mindist 50,param1 200,param2 40