

디지털 영상처리 연구실 연구보고서

김우현

##모폴로지 연산

```
pDoc->m_AllocStructureElementBinary(6);  
pDoc->m_SetStructureElementBinary(0, 6, 0);  
pDoc->m_SetStructureElementBinary(1, 0, -1);  
pDoc->m_SetStructureElementBinary(2, 0, 0);  
pDoc->m_SetStructureElementBinary(3, 0, 1);  
pDoc->m_SetStructureElementBinary(4, -1, 0);  
pDoc->m_SetStructureElementBinary(5, 1, 0);
```



	1	
1	1	1
	1	

팽창연산을 위한 마스크

```
pDoc->m_MorphologyBinaryDilation(256, 256);  
  
pDoc->m_FreeStructureElementBinary();
```

```
pDoc->m_AllocStructureElementBinary(10);  
pDoc->m_SetStructureElementBinary(0, 10, 0);  
pDoc->m_SetStructureElementBinary(1, -1, -1);  
pDoc->m_SetStructureElementBinary(2, 0, 0);  
pDoc->m_SetStructureElementBinary(3, -1, 0);  
pDoc->m_SetStructureElementBinary(4, -1, 1);  
pDoc->m_SetStructureElementBinary(5, 0, -1);  
pDoc->m_SetStructureElementBinary(6, 1, -1);  
pDoc->m_SetStructureElementBinary(7, 1, 1);  
pDoc->m_SetStructureElementBinary(8, 1, 0);  
pDoc->m_SetStructureElementBinary(9, 0, 1);
```

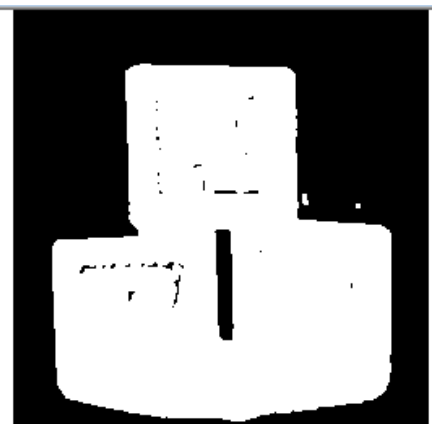
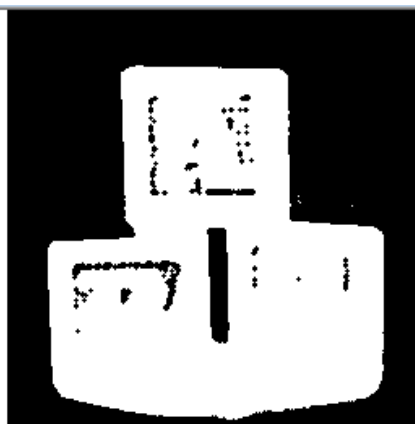


1	1	1
1	1	1
1	1	1

침식연산을 위한 마스크

```
pDoc->m_MorphologyBinaryErosion(256, 256);
```

```
pDoc->m_FreeStructureElementBinary();
```

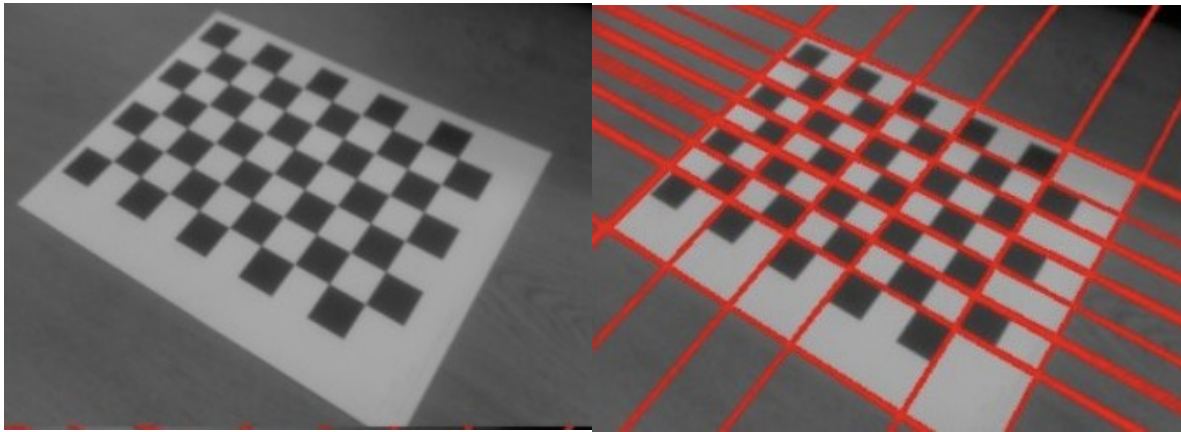


-> 팽창

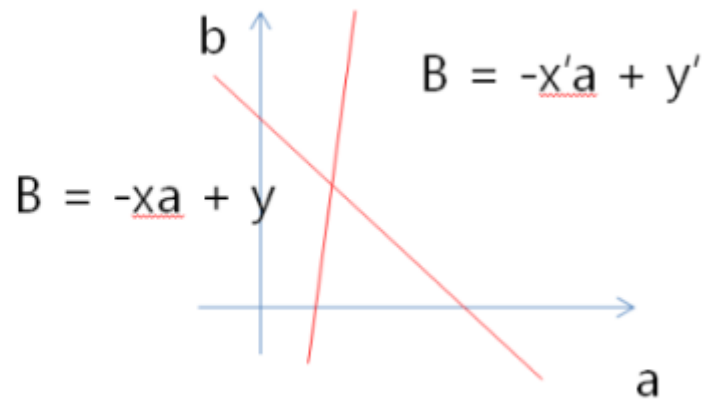
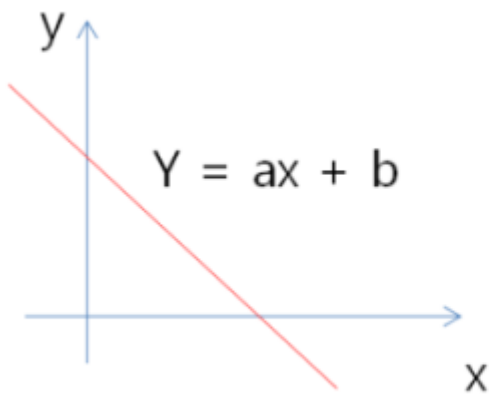
-> 침식

##허프변환

→ edge들을 그룹화



#직선추출 허프변환 원리

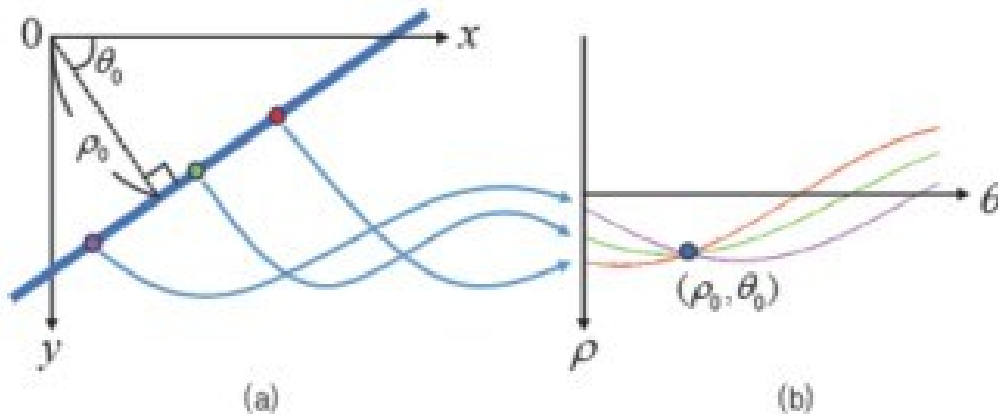


->문제점발생

->y=c와 같은 직선들은 파라메타 공간으로 표현불가

---> 삼각함수를 이용한 직선 방정식 이용

$$x \cos \theta_0 + y \sin \theta_0 = \rho_0$$



-> 이때 x-y 평면에서의 범위는 $0 < \rho < d, 0 < \theta < 180$ 입니다.

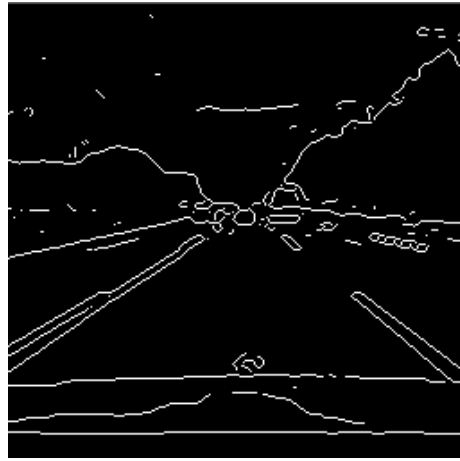
#HT 코드구현

```
for (i = 0; i < height; i++)
{
    index = i * width;
    for (j = 0; j < width; j++)
    {
        if (orgImg[index + j] == 255)
        {
            for (k = 0; k < 360; k++)
            {
                d = (int)(j * LUT_COS[k] + i * LUT_SIN[k]);
                if (d >= 4 && d <= 360) H[k][d]++; //H[360][362]
            }
        }
    }
}
```

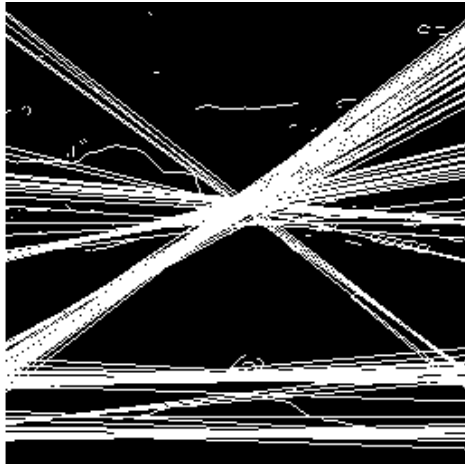
-> $H[k][d]$ 로 카운팅 =파라메타 공간에서의 보팅과정

```
for (d = 4; d <= 360; d++)
{
    for (k = 0; k < 360; k++)
    {
        if (H[k][d] > thres)
        {
            for (j = 0; j < width; j++)
            {
                i = (int)((d - j * LUT_COS[k]) / LUT_SIN[k]);
                if (i < height && i > 0) outImg[i * width + j] = 255;
            }
        }
    }
}
```

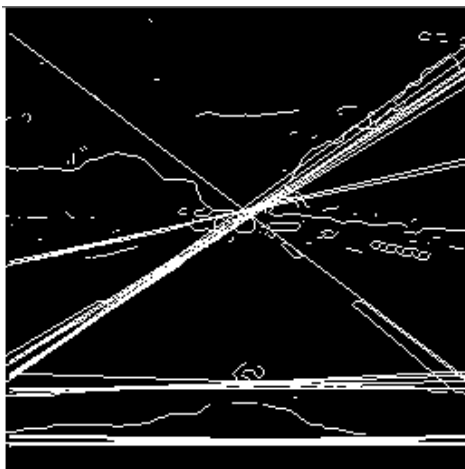
➔ 파라메타공간 $H[k][d] > \text{thres}$ 인 부분에서 k,d값을 이용하여 image의 x,y좌표를 구하여 직선 그리는 과정



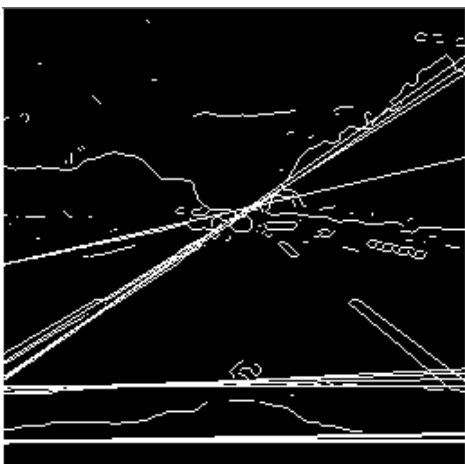
-> ORGNAL IMAGE



-> thres=30



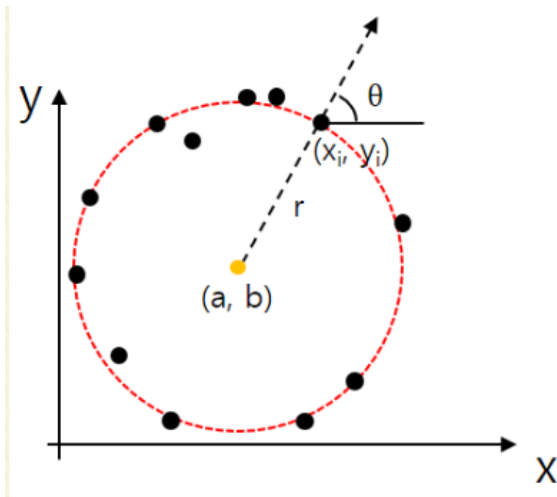
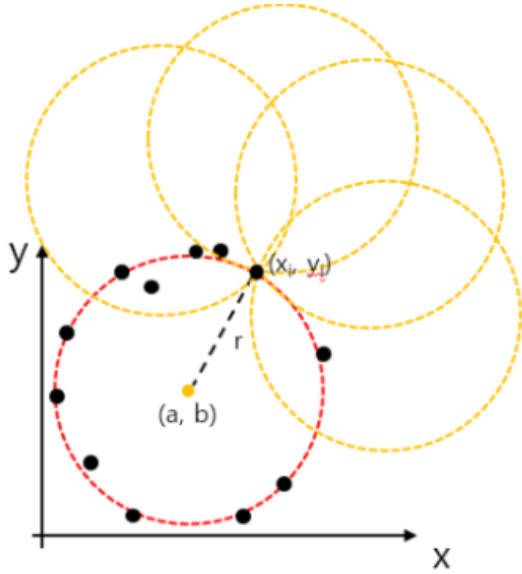
-> thres=60



-> thres=70

#원추출 허프 변환

->직선과는 달리 3개의 파라미터 존재(중심좌표 cx, cy 와 반지름 r)->보팅을 위한 허프공간=3차원



$$\begin{aligned} x - a &= r \cdot \cos\theta \\ y - b &= r \cdot \sin\theta \end{aligned}$$

#HT 원추출 코드 구현

```
if (orgImg[index + j] == 255)
{
    for (k = r_min; k < r_max; k++)
    {
        for (ang = 0; ang < 360; ang++)
        {
            rr = (int)(i - k * LUT_COS[ang]);
            cc = (int)(j - k * LUT_SIN[ang]);
            if (rr < height && rr > 0 && cc < width && cc > 0) H[rr][cc][k]++;
        }
    }
}
```

- i, j 값으로 rr, cc(원중심), k값(반지름) 파라메타 공간에서의 보팅
- rmin, rmax를 설정하여 계산시간 단축

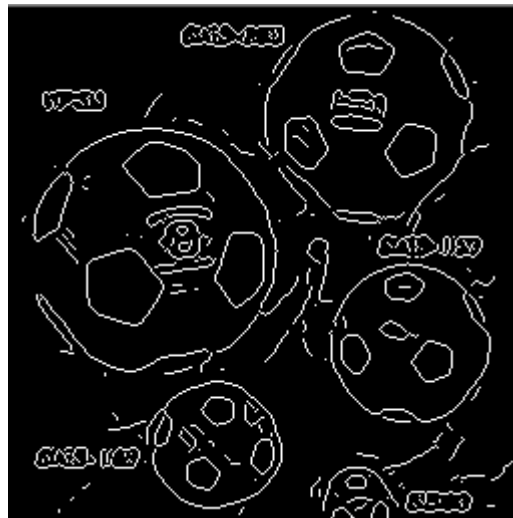
```

for (rr = 0; rr < height; rr++)
{
    for (cc = 0; cc < width; cc++)
    {
        for (k = r_min; k < r_max; k++)
        {
            if (H[rr][cc][k] > thres)
            {
                for (ang = 0; ang < 360; ang++)
                {
                    i = (int)(rr + k * LUT_COS[ang]);
                    j = (int)(cc + k * LUT_SIN[ang]);
                    if (i > 0 && i < height && j > 0 && j < width) outImg[i * width + j] = 255;
                }
            }
        }
    }
}

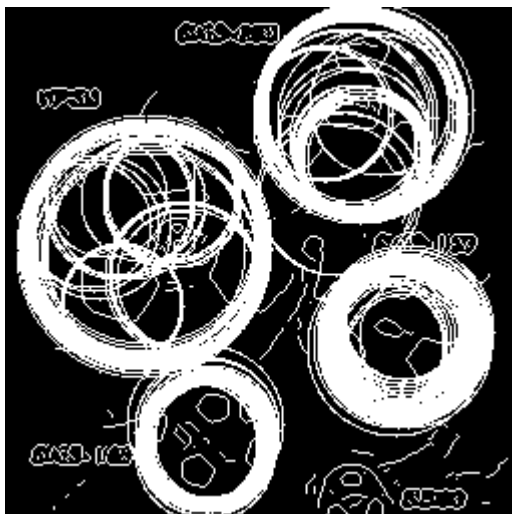
```

➔ 파라미터 3차원공간에서 $H[rr][cc][k]$ 값이 $thres$ 보다 클 경우에 추출하려는 대상으로 인식->디스플레이

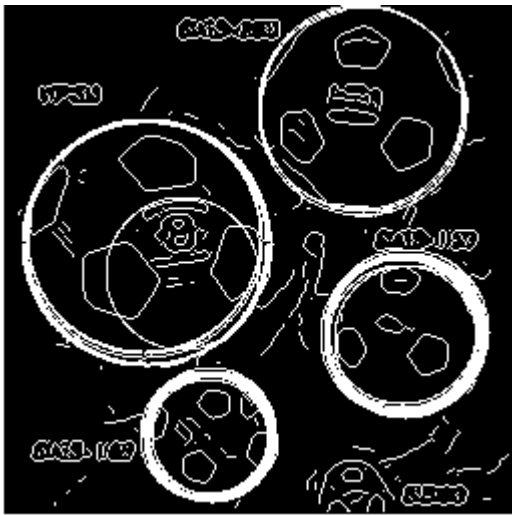
#원추출 결과



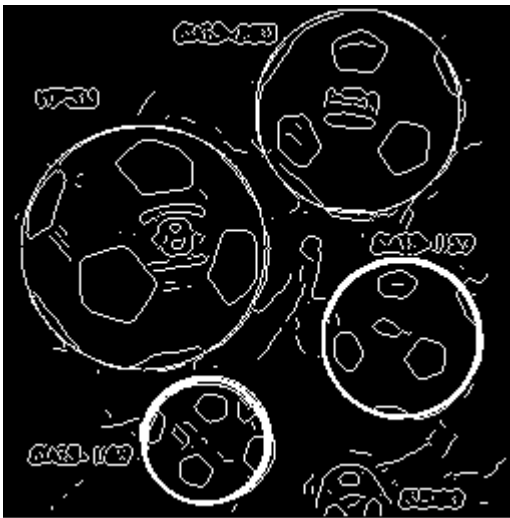
ORIGINAL IMAGE



→ THRES=70



-> THRES=90

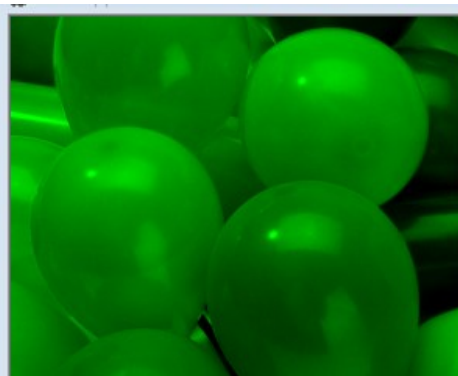
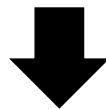


-> THRES=110

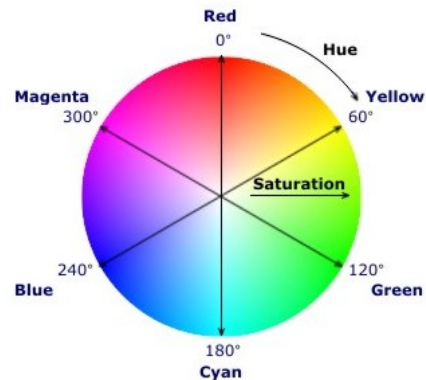
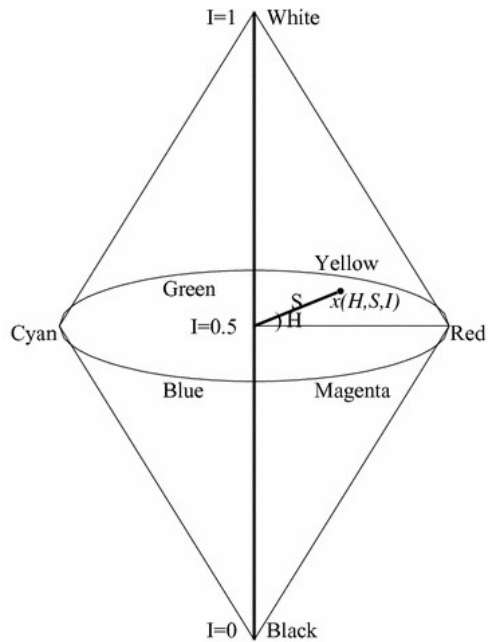
##컬러영상처리

#RGB 모델

```
int index, i, j;  
for (int k = 2; k >= 0; k++)  
{  
    for (i = 0; i < height * rsize; i++) pDoc->m_OutImg[i] = 0;  
    for (i = 0; i < height; i++)  
    {  
        index = (height - i - 1) * rsize;  
        for (j = 0; j < width; j++)  
            pDoc->m_OutImg[index + 3 * j + k] = pDoc->m_InImg[index + 3 * j + k];  
    }  
    pDoc->CopyClipboard(pDoc->m_OutImg, height, width, 24);  
}
```



#HSI 색 모델



- ◆ H(색상) → 0~360도
- ◆ S(채도) → 순색에 첨가된 백색광의 비율 → 0~1사이 (클수록 포화=순색)
- ◆ I(명도) → 빛의세기 → 0~1사이

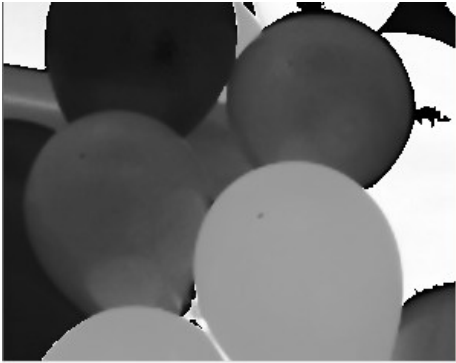
```
for (i = 0; i < height; i++)
{
    index = (height - i - 1) * rwsz;
    for (j = 0; j < width; j++)
    {
        r = (float)InImg[index + 3 * j + 2] / 255.0f;
        g = (float)InImg[index + 3 * j + 1] / 255.0f;
        b = (float)InImg[index + 3 * j] / 255.0f;

        min = (r < g ? (r) : (g));
        min = (min < b ? (min) : (b));    ///RGB중 최솟값찾음

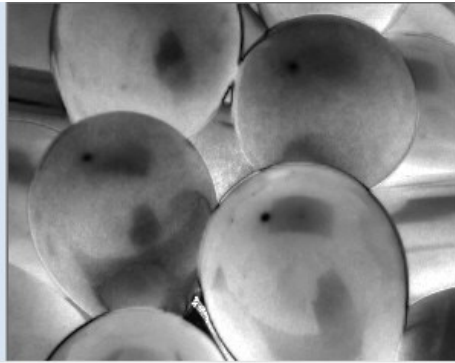
        iv = (r + g + b) / 3.0f;          //밝기구함

        s = 1.0f - (3.0f / (r + g + b)) * min; //채도 구함
        angle = (r - 0.5f * g - 0.5f * b) / ((float)sqrt((r - g) * (r - g) + (r - b) * (g - b)));
        h = (float)acos(angle);           //색상구함
        h *= 57.29577951f;

        if (b > g) h = 360.0f - h;
        ih = (int)(h * 255.0 / 360.0);
        OutImg[index + 3 * j] = (unsigned char)ih;
        OutImg[index + 3 * j + 1] = (unsigned char)(s * 255.0);
        OutImg[index + 3 * j + 2] = (unsigned char)(iv * 255.0);
    }
}
```



H



S



I

#컬러이미지의 라플라시안

->컬러이미지의 라플라시안 마스크는 RGB컬러->HSI컬러로 바꾼 후

-> I (밝기)에 대해 라플라시안 연산을 진행한후에

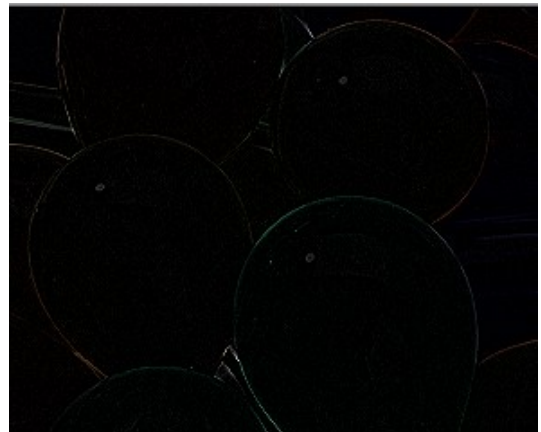
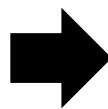
-> 밝기 연산 결과값과 기존의 H,S를 이용하여 다시 RGB컬러로 바꾼 후

-> 디스플레이 하여 줍니다.

```
for (i = n; i < height - n; i++)    //n=1
{
    index1 = i * width;
    for (j = n; j < width - n; j++)
    {
        sum = 0.0f;

        for (k = -n; k <= n; k++)
        {
            index2 = (i + k) * width;
            index3 = (k + n) * winsize;
            for (l = -n; l <= n; l++)
                sum += InImg[index2 + (j + l)] * Mask[index3 + l + n]; //마스크컨볼루션연산
        }
        CLIP(sum, 0, 255);
        OutImg[index1 + j] = (unsigned char)sum;
    }
}
```

0	-1	0
-1	4	-1
0	-1	0

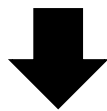
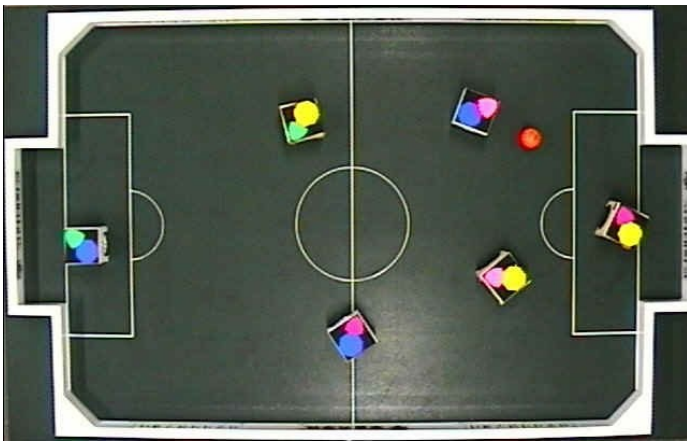


#컬러이미지의 이진화

➔ 그레이 이미지의 이진화와 비슷한 방법으로 컬러 이미지도 이진화가 가능합니다.

```
for (i = 0; i < 256; i++) LUT[i] = i > m_ThresValueDisp ? (unsigned char)255 : 0;
for (i = 0; i < height; i++)
{
    index = (height - i - 1) * rsize;
    for (j = 0; j < width; j++)
    {
        index2 = index + 3 * j;
        m_TmpImg[index2] = LUT[(int)(pDoc->m_OutImg[index2]));
        m_TmpImg[index2 + 1] = LUT[(int)(pDoc->m_OutImg[index2 + 1]));
        m_TmpImg[index2 + 2] = LUT[(int)(pDoc->m_OutImg[index2 + 2]));
    }
}
```

➔ 동적으로 변하는 m_ThresValueDisp 값에 따라 rgb채널별로 이진화 진행



#컬러 히스토그램 정합

$$\emptyset = \min \sum(I, M) -$$