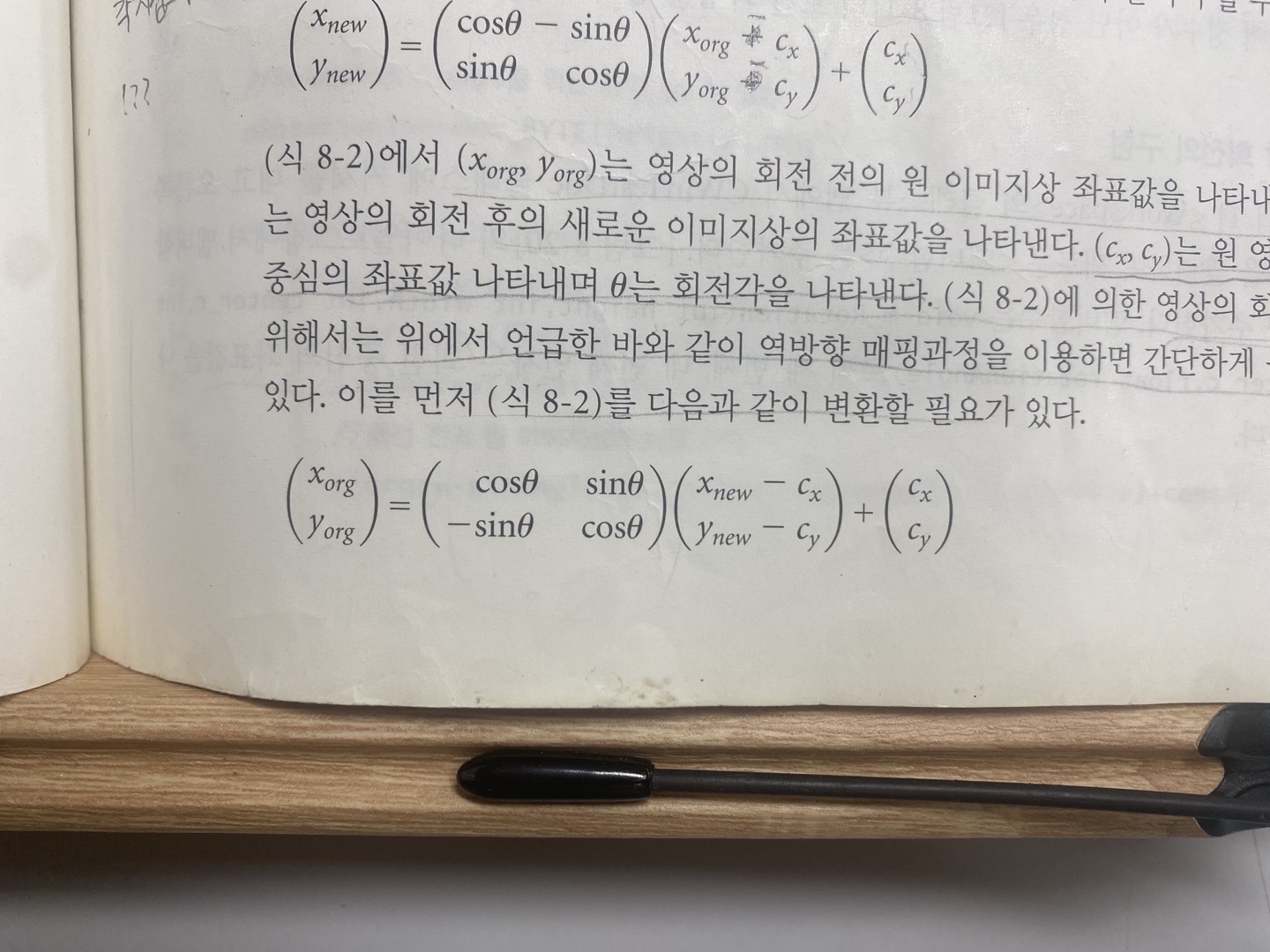
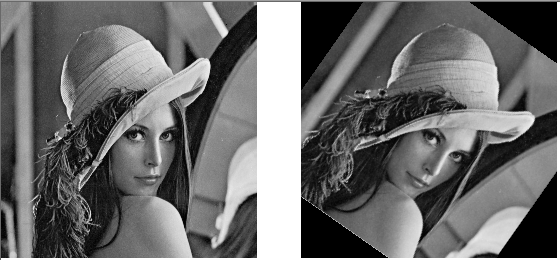
디지털 영상처리 연구실 연구보고서

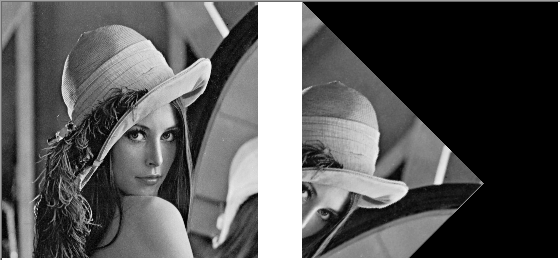
김우헌

##영상의 회전



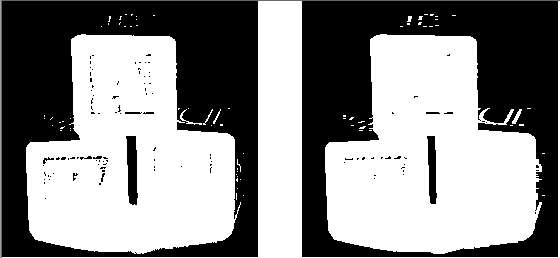


* Lenna image를 기준점(256/2,256/2)을 기준으로 35도 회전

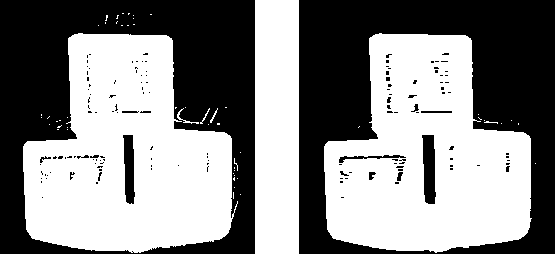


* Lenna image를 기준점(0,0)을 기준으로 45도 회전

##모폴로지



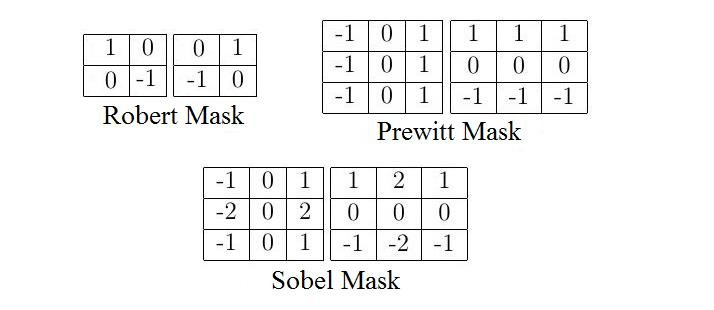
* 모폴로지 침식연산 -> 노이즈 제거



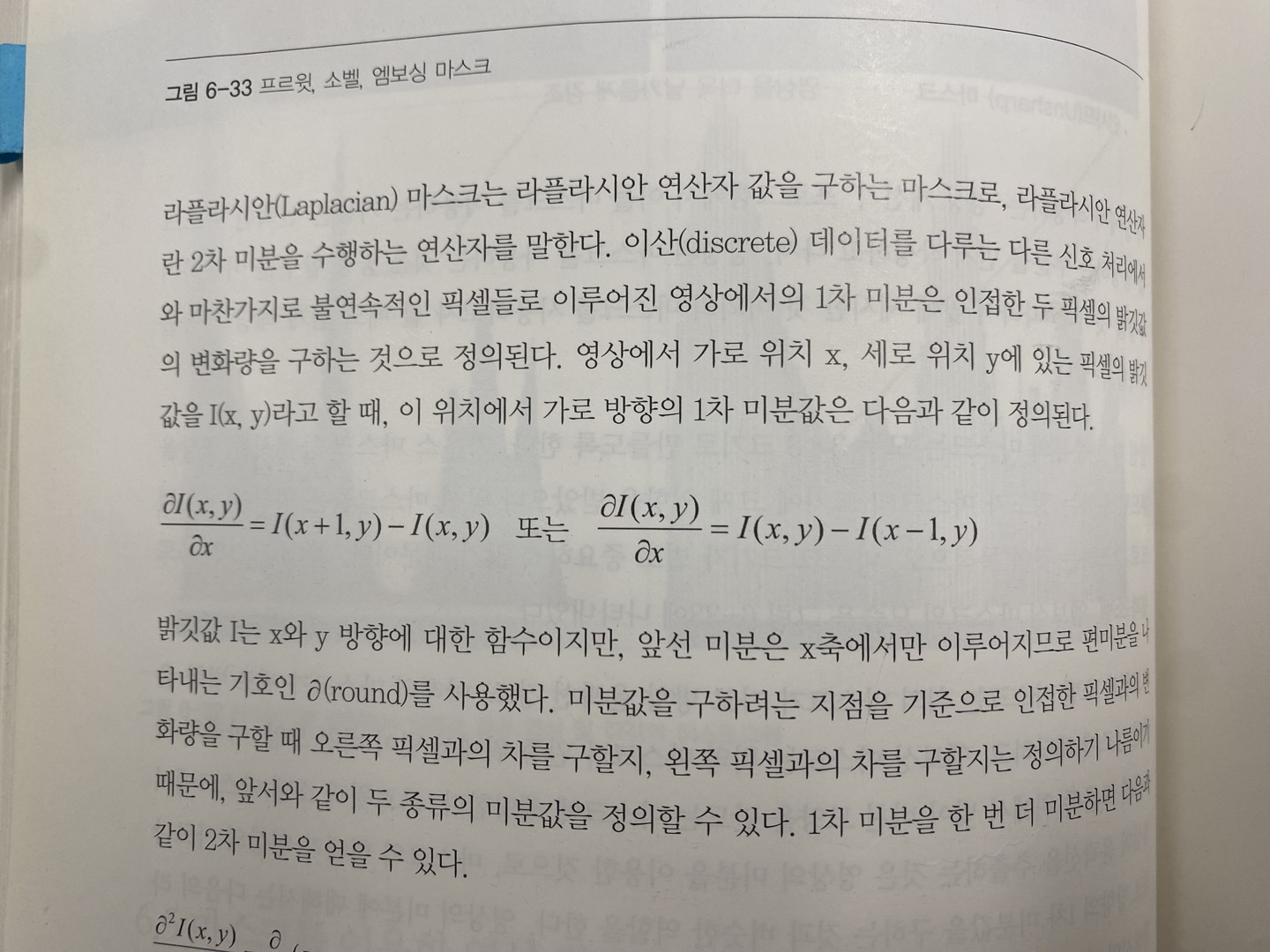
* 모폴로지 팽창 연산 -> 빈 공간 채워짐

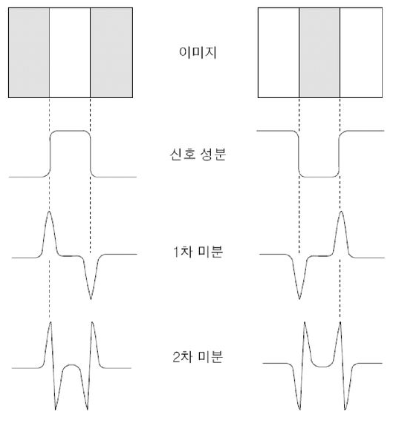
##경계추출

#1차 미분 필터



* 세로방향, 가로방향의 윤곽선 강조 -> 1차 미분값

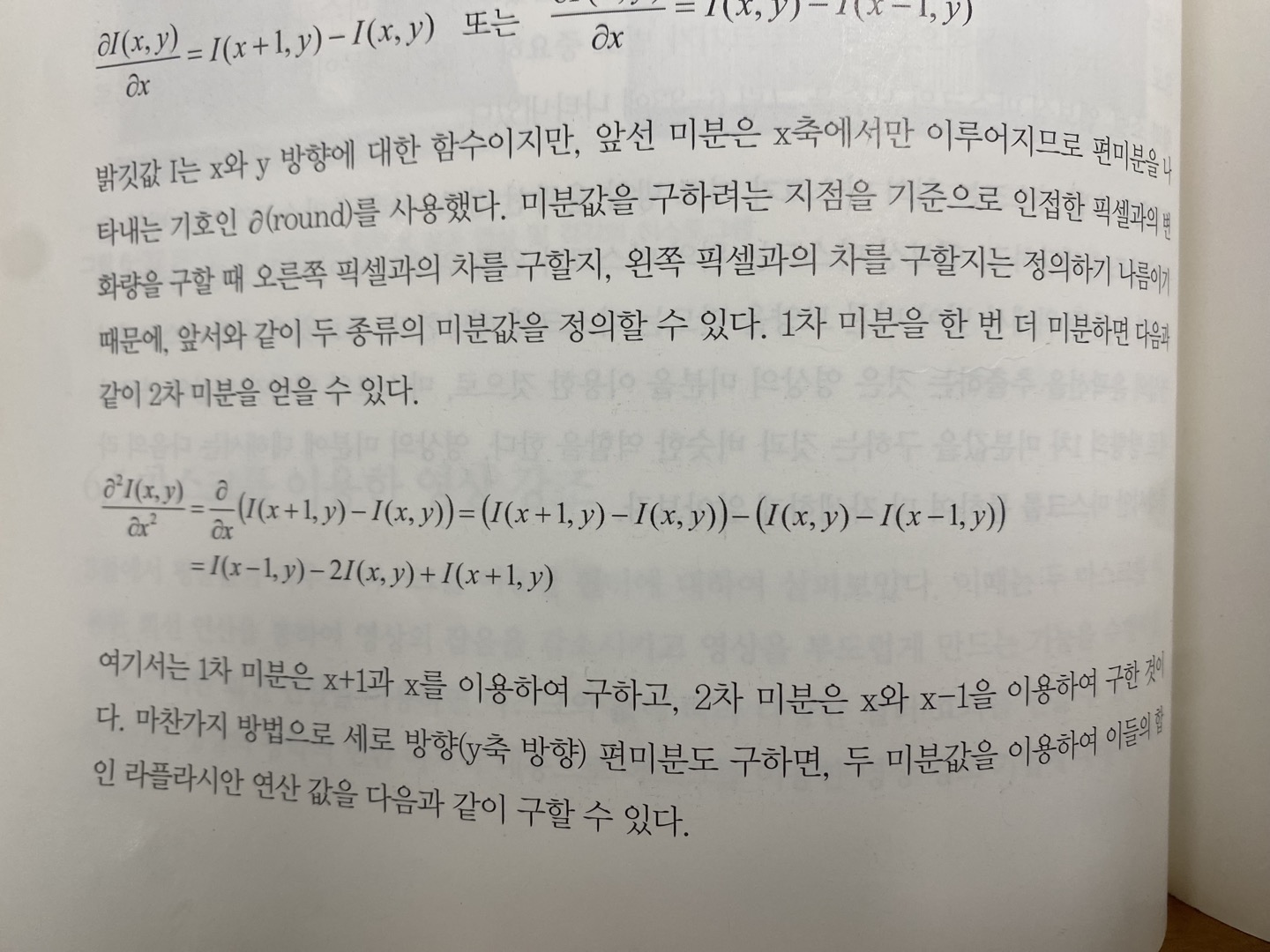




#2차 미분 필터

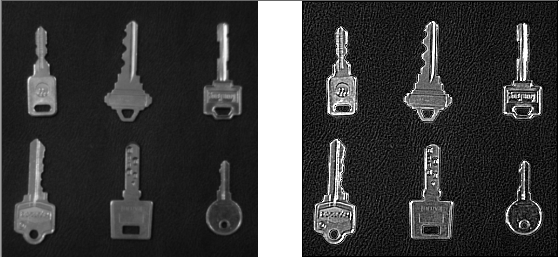
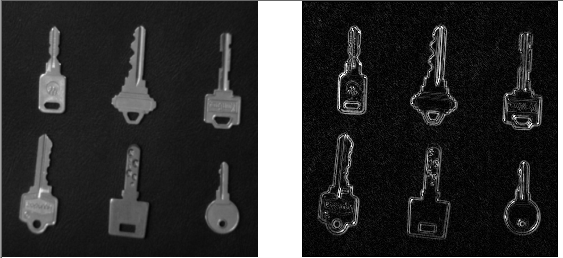
* 라플라시안 마스크=라플라시안 연산자 값을 구하는 마스크

\*\*라플라시안 연산 =2차 미분을 수행하는 연산자!

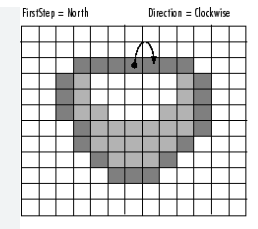


|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |



#영역 경계의 추적



-경계추적

|  |  |  |
| --- | --- | --- |
| 3 | 4 | 5 |
| 2 | X | 6 |
| 1 | 0 | 7 |

->관심픽셀의 주위 픽셀번호

->처음시작은 4번부터

-> N`=(N+5)&7;

|  |  |  |
| --- | --- | --- |
| (-1,-1) | (-1,0) | (-1,1) |
| (0,-1) | X | (0,1) |
| (1,-1) | (1,0) | (1,1) |

const POINT nei[8] = // clockwise neighbours

{

{1,0}, {1,-1}, {0,-1}, {-1,-1}, {-1,0}, {-1,1}, {0,1}, {1,1}

};

->nei[] 를 통한 경계추적 박스 설정

do

{

for (k = 0; k < 8; k++, n = ((n + 1) & 7))

{

short u = (short)(x + nei[n].x);

short v = (short)(y + nei[n].y);

if (u < 0 || u >= height || v < 0 || v >= width) continue;

if (m\_inimage[u][v] == c0) break; // 관심점의 주위를 돌다가 같은 밝기의

}

if (k == 8) break; // 고립점(영역내 추적할영역의 화소가 하나)

visited[x \* width + y] = 255; // 방문한 점으로 마크

xchain[border\_count] = x;

ychain[border\_count++] = y; // border\_count++통한 경계면의 좌표를 0부터 증가시킴

if (border\_count >= 10000) break;

x = x + nei[n].x;

y = y + nei[n].y; //중심이동

if (n % 2 == 1) diagonal\_count++;

n = (n + 5) & 7; //n값설정

}while (!(x == x0 && y == y0));//처음기준좌표

if (k == 8) continue; // 고립점(영역내 추적할영역의 화소가 하나)

stBorderInfo[numberBorder].x = new short[border\_count];

stBorderInfo[numberBorder].y = new short[border\_count];

// border\_count는 경계면의 개수

for (k = 0; k < border\_count; k++)

{

stBorderInfo[numberBorder].x[k] = xchain[k];

stBorderInfo[numberBorder].y[k] = ychain[k];

}

stBorderInfo[numberBorder].n = border\_count;

stBorderInfo[numberBorder++].dn = diagonal\_count;

->구조체의[numberBorder] 즉 경계면의 개수가 인식하기위한 인덱스로 작용

for (k = 0; k < numberBorder; k++)

{ for (int i = 0; i < stBorderInfo[k].n; i++)

{ x = stBorderInfo[k].x[i];

y = stBorderInfo[k].y[i];

m\_outimg[x][y] = 0;

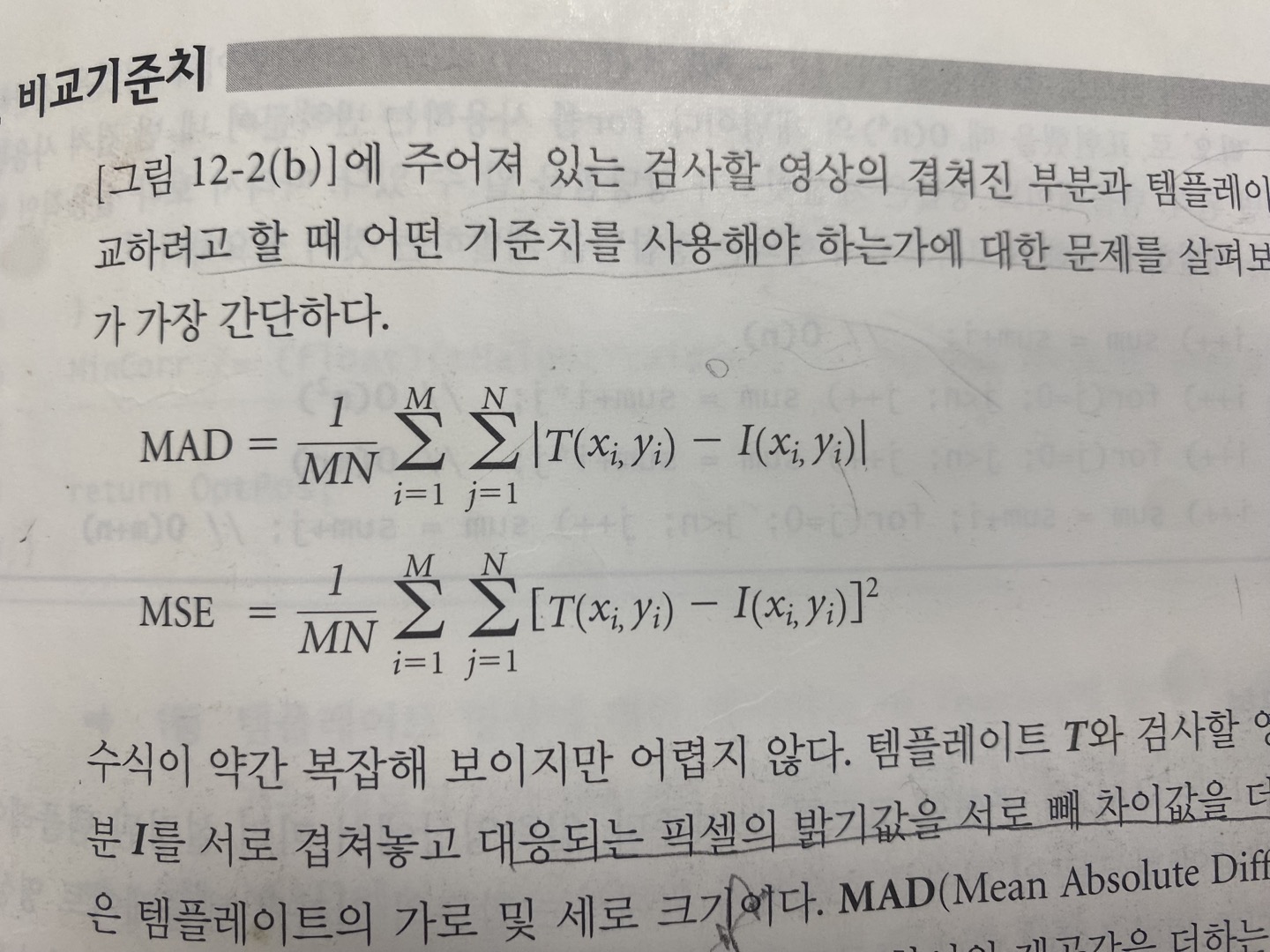
}

}

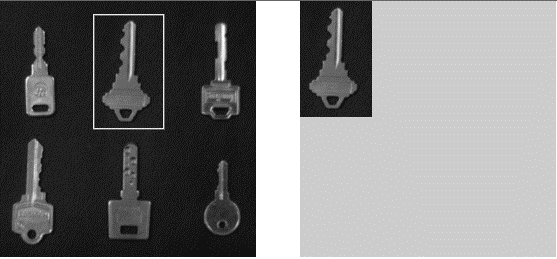
-> stBorderInfo 구조체를 이용하여 경계면 0으로 설정

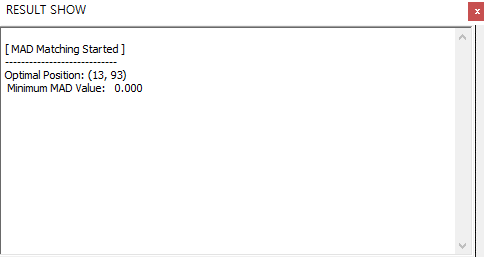
->경계추출

##템플레이트 정합

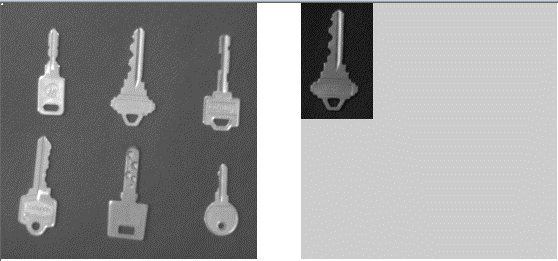


->픽셀차 합들의 최솟값->비교치 계산가능

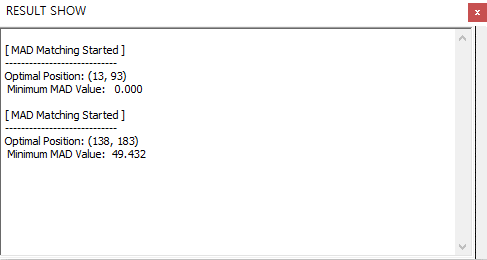


->템플레이트 정합 결과

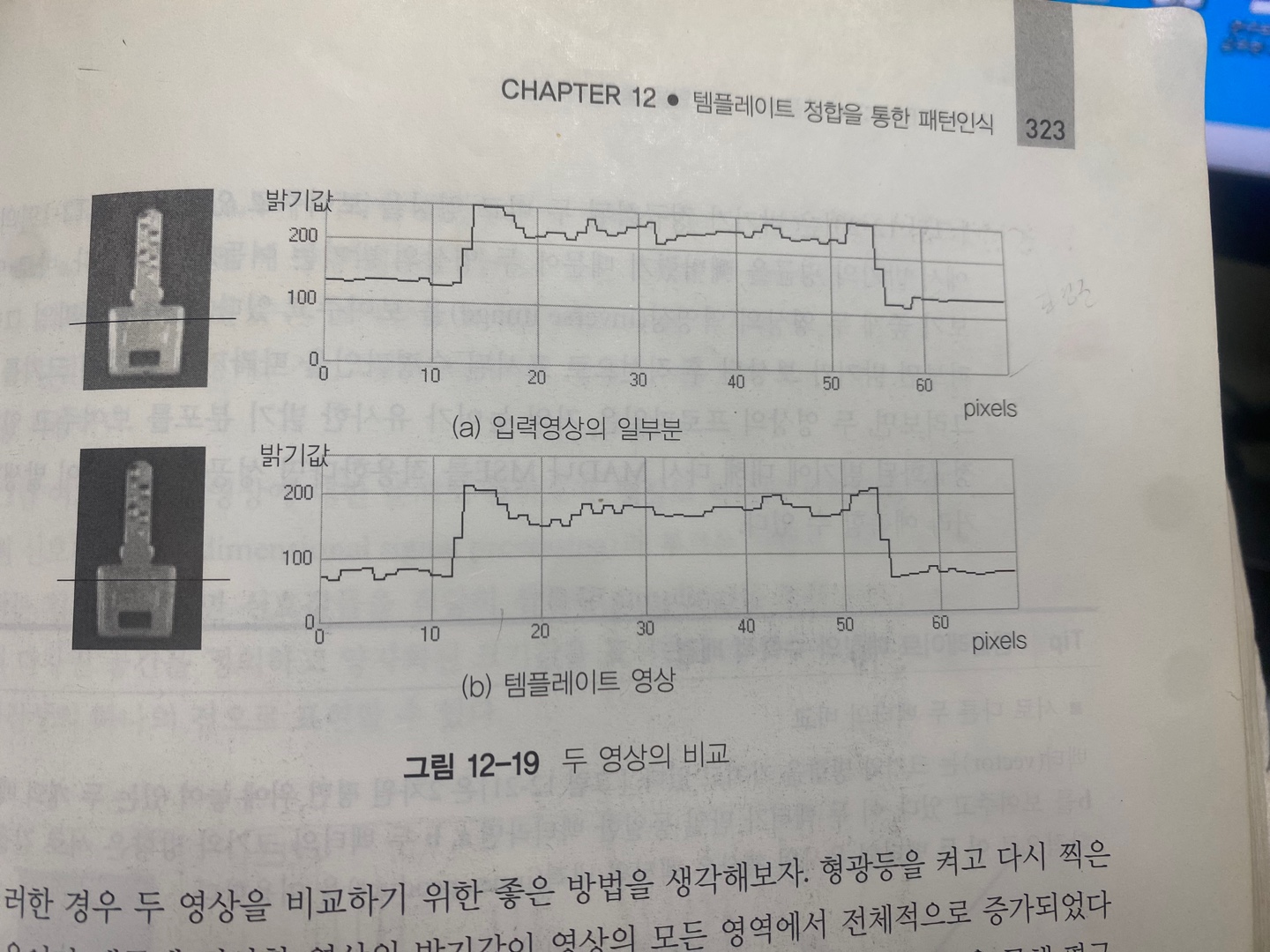
#농담정규화 정합



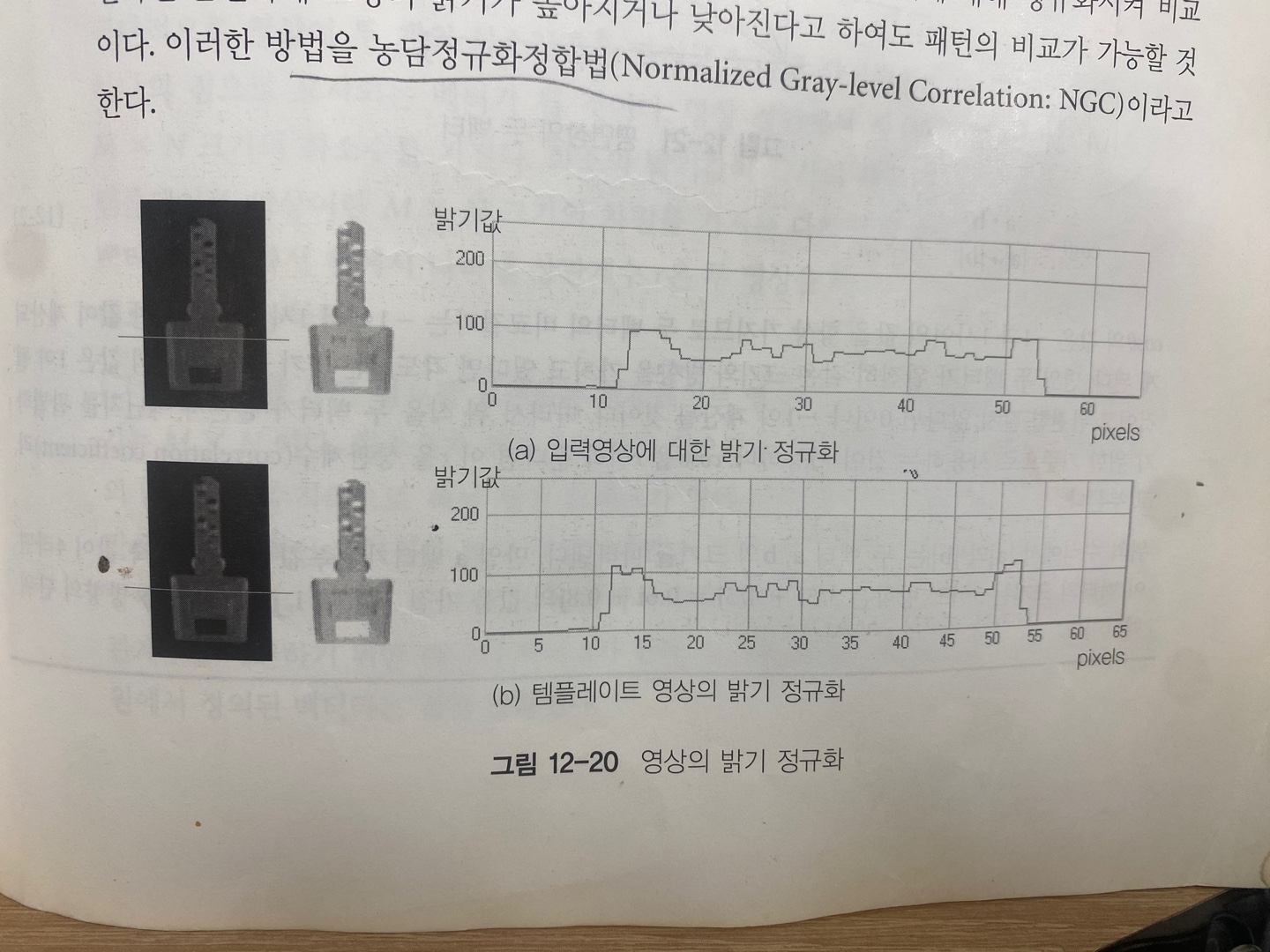
->밝기가 변화된 탐색할 영상 -> 기존의 템플레이트 영상



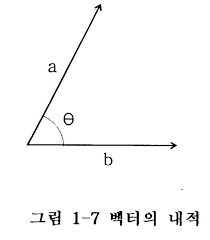
* 정합이 제대로 실행되지 않음



* 입력영상의 배경 밝기 값과 템플레이트 영상의 배경 밝기 값의 차에 의해 mad방식 실패!

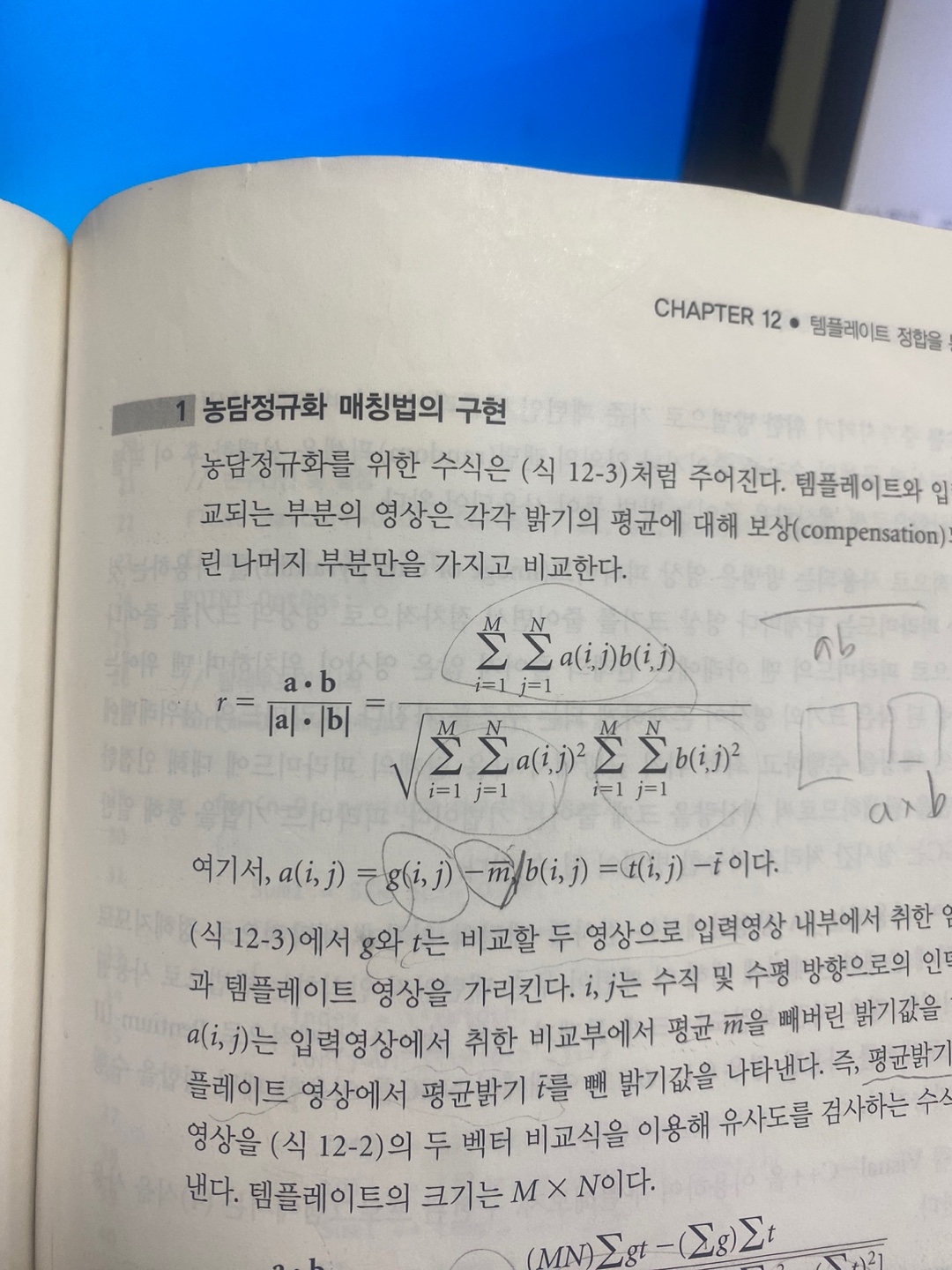
  
-> 빼주어 입력영상와 템플레이트영상에 밝기의 평균을 빼주어 정합 실시-> NGC=농담정규화정합법

#농담정규화의 매칭법

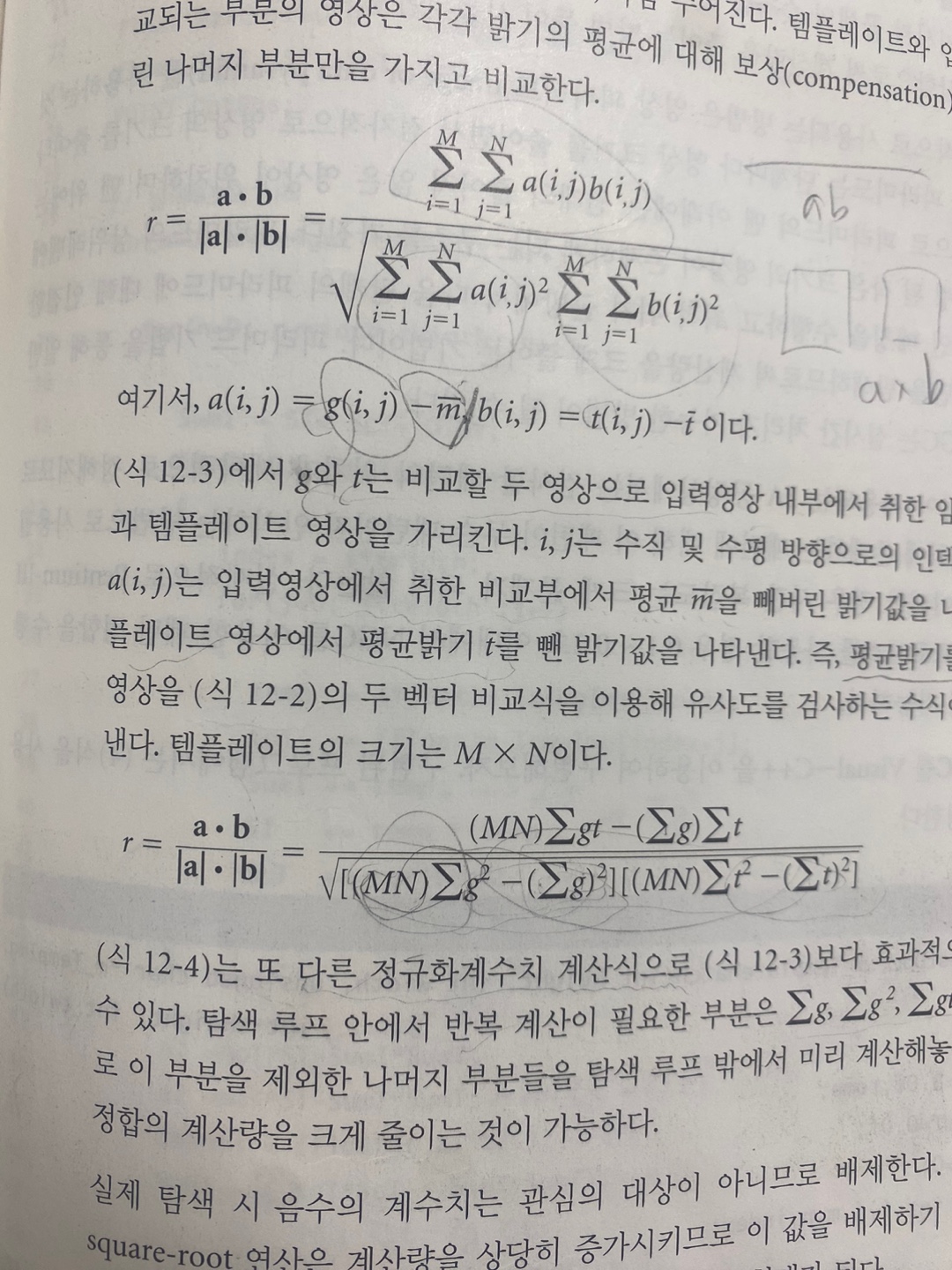


=R

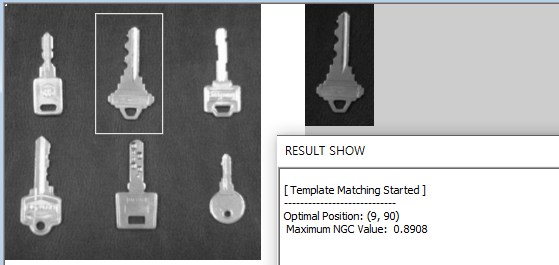
R=상관계수(correlation coefficient)



->유사도 검사식 유도



* NGC결과



* 기존의 MAD결과 (상단은 밝기변화가 없는 이미지,하단은 밝기가 변화된 이미지)

