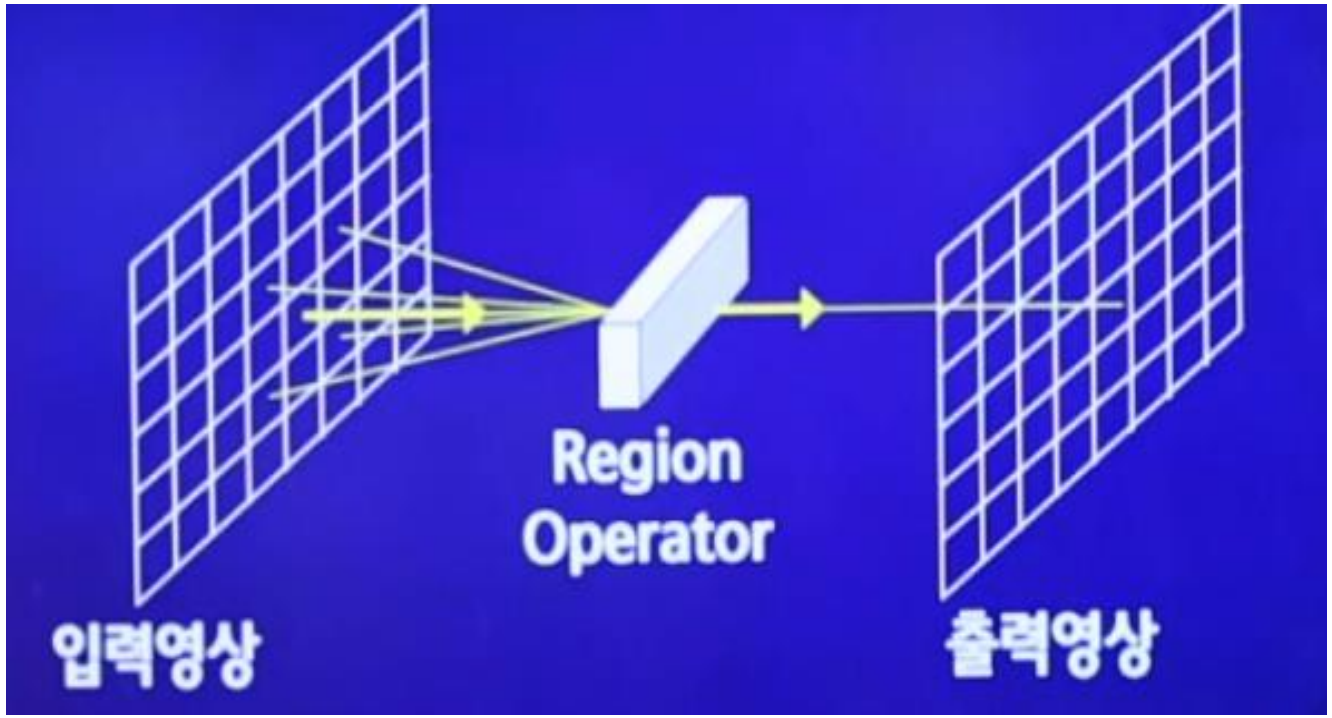
4-1. 컨볼루션 개념

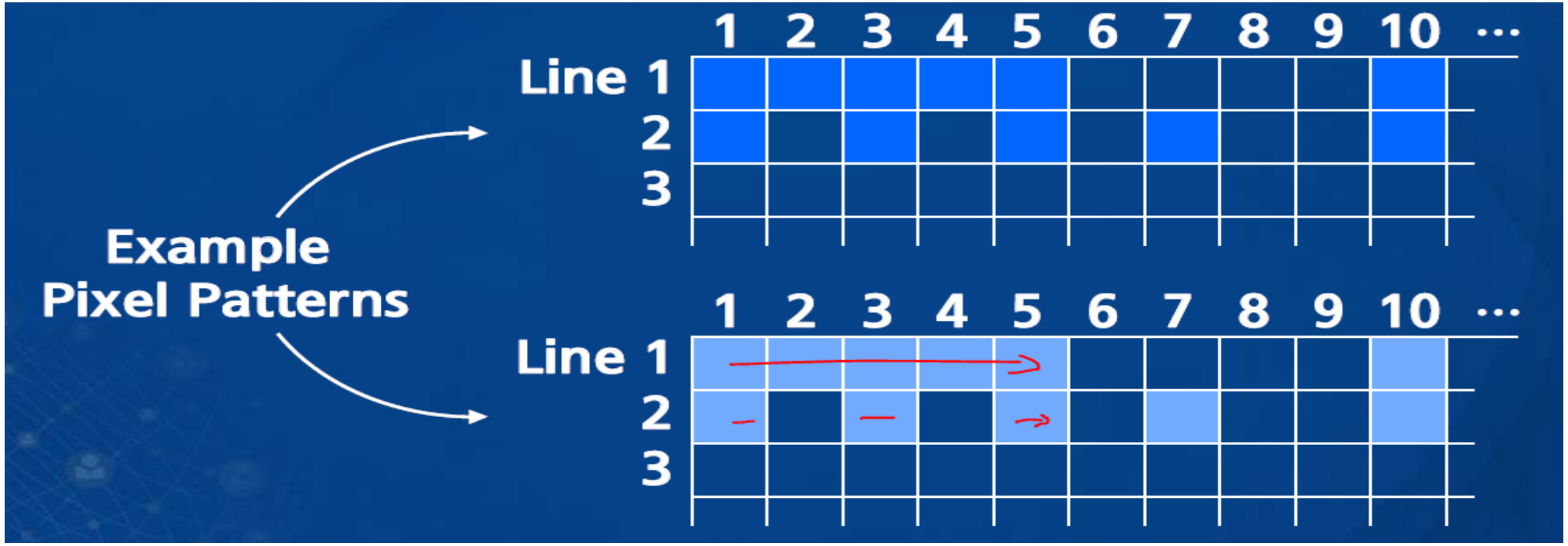
영상 영역 처리



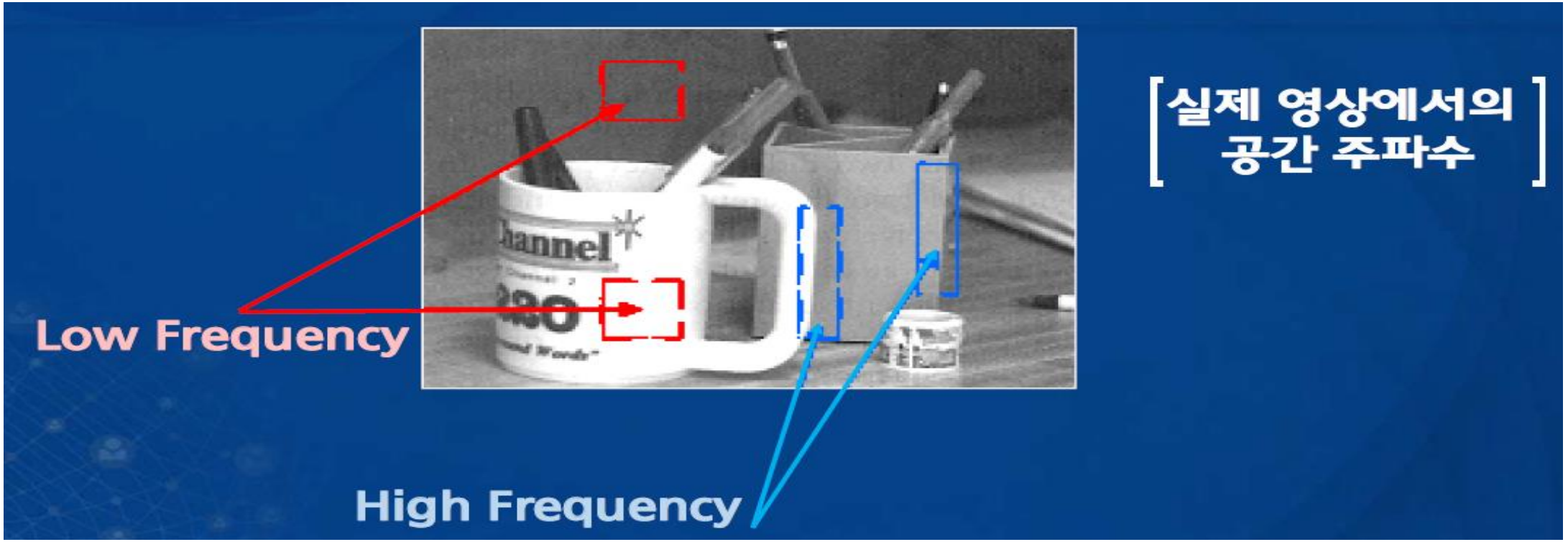
입력화소 주위에 있는 화소들을 사용하여 출력화소를 만드는 개념

영상 공간주파수

- 저주파(밝기 값 유지) 혹은 고주파(밝기 값 변화) 사용
- 임의의 방향에 대해 화소의 밝기 값의 변화 빈도수로 주파수 파악



영상 공간주파수



Low Frequency : 주위 색 비슷비슷함
High Frequency : 주위와 비교했을 때 색 변화가 보임

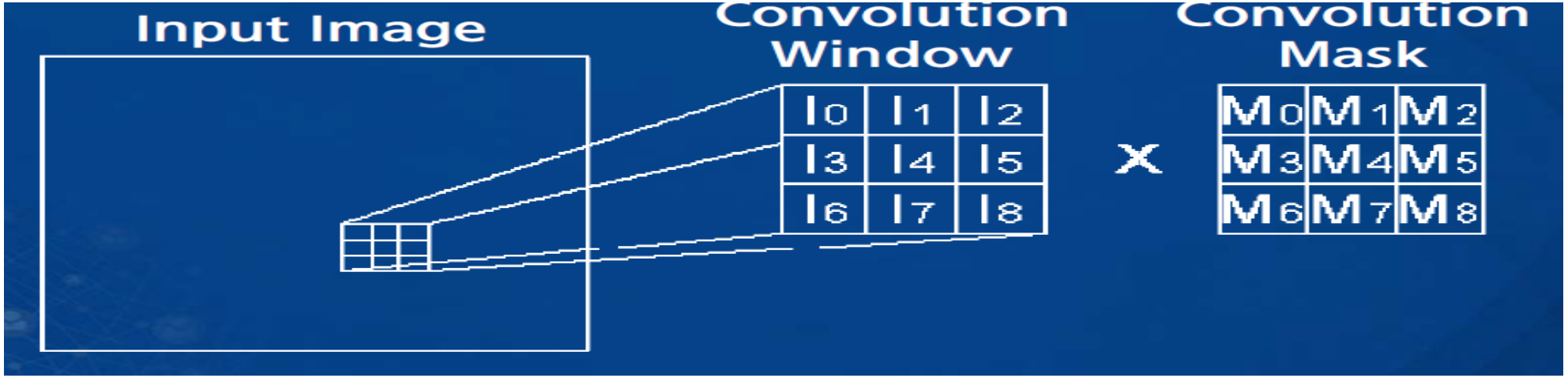
Convolution(=Sum of Product)

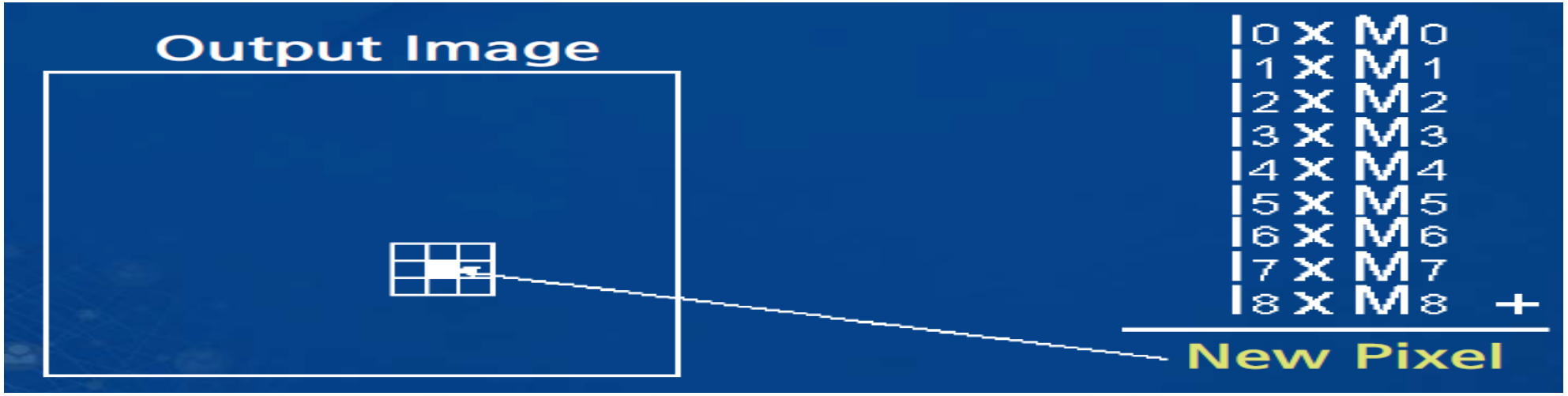
- Mask를 정의하여 영상 위에 덮어 씌우는 방법
- Mask 숫자와 영상의 겹치는 숫자끼리 곱한 값을 다 더함
- 계산된 값을 다음 연산에 영향을 미치지 않도록 새로운 배열에 저장해야 한다.
- Mask 사이즈는 반드시 홀수(가운데 값 정의 위해)
- 계속 마스크를 이동시키며 연산 수행

$$f(t) * h(t) = \int_{-\infty}^{\infty} f(a)h(t-a)da$$

$$R(x,y) = \sum_{u,v=0}^{k-1} g(x+c-u, y+c-v) \cdot m(u,v)$$

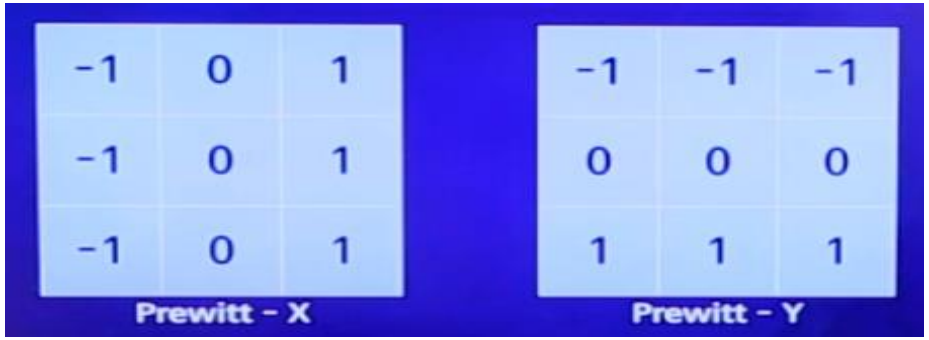
Convolution 계산

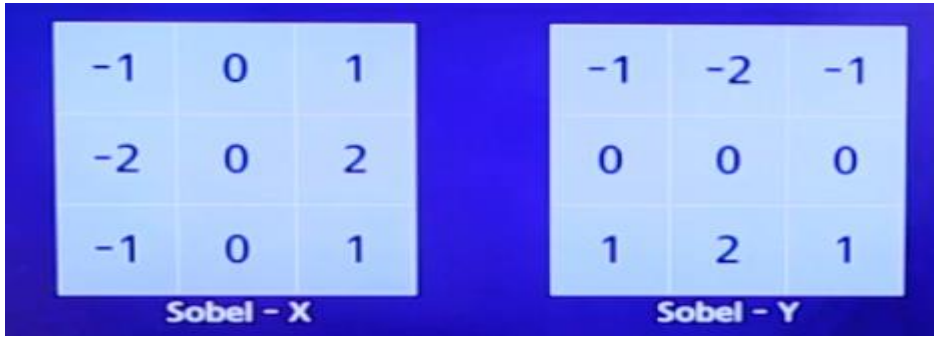




4-2. 영상 영역 처리의 종류 및 특징

컨볼루션 마스크



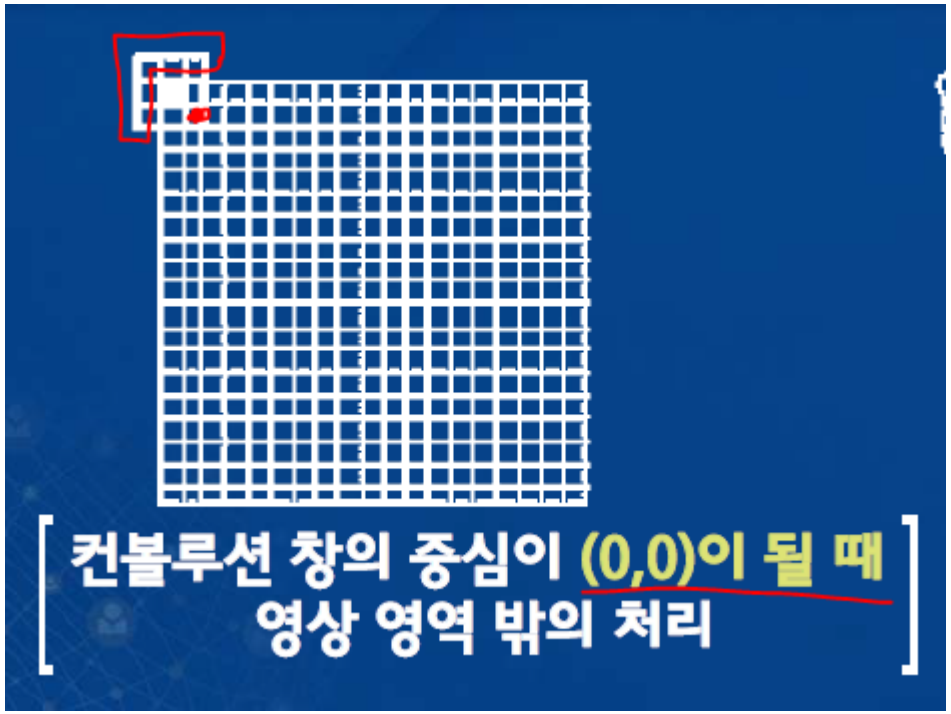


합이 1보다 큰/작은 경우
: 컨볼루션 결과 밝아/어두워진다.

합이 1인 경우
: 원래 밝기는 유지하고, 경계를 검출

합이 0이 되도록 유지
: 영상의 경계 선분을 찾음
= 경계가 없는 곳의 합을 0으로 만든다.

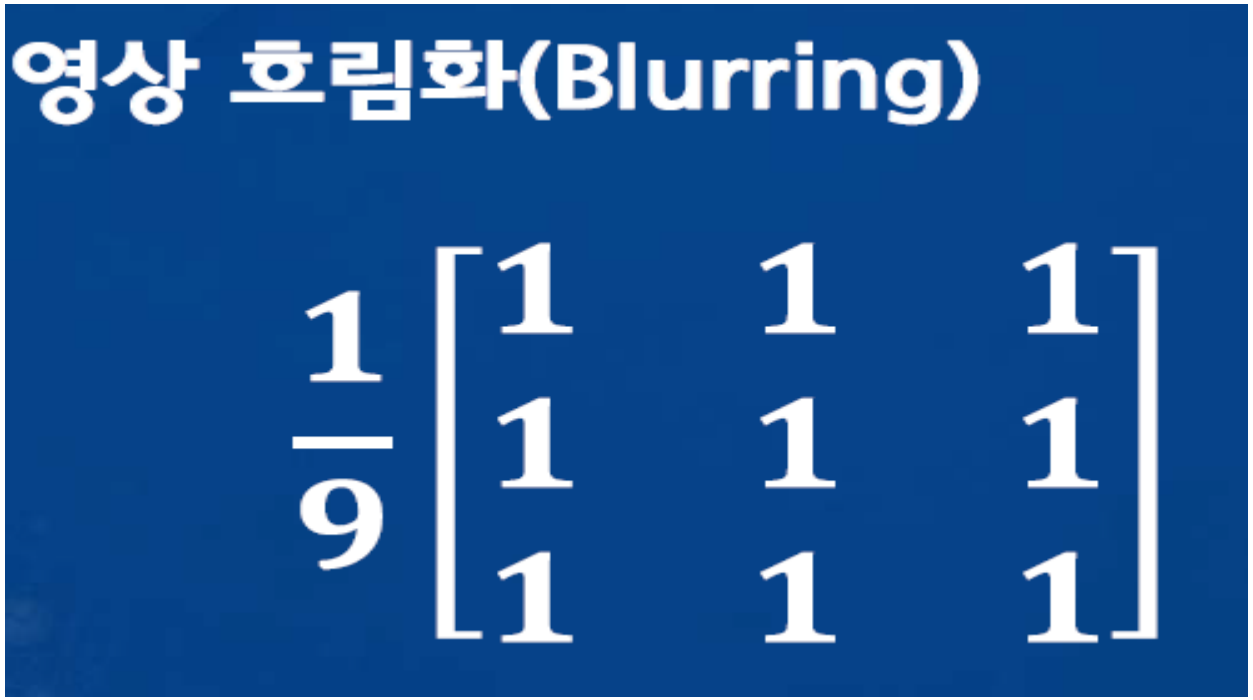
주의사항



Mask 사이즈의 ½ 만큼 상하좌우 1픽셀 마진 남겨놓고 진행

저역통과 필터(평활화)

- 저주파 성분만 통과, Low-Pass filter
- 고주파 : Edge, 노이즈 및 밝기 제거

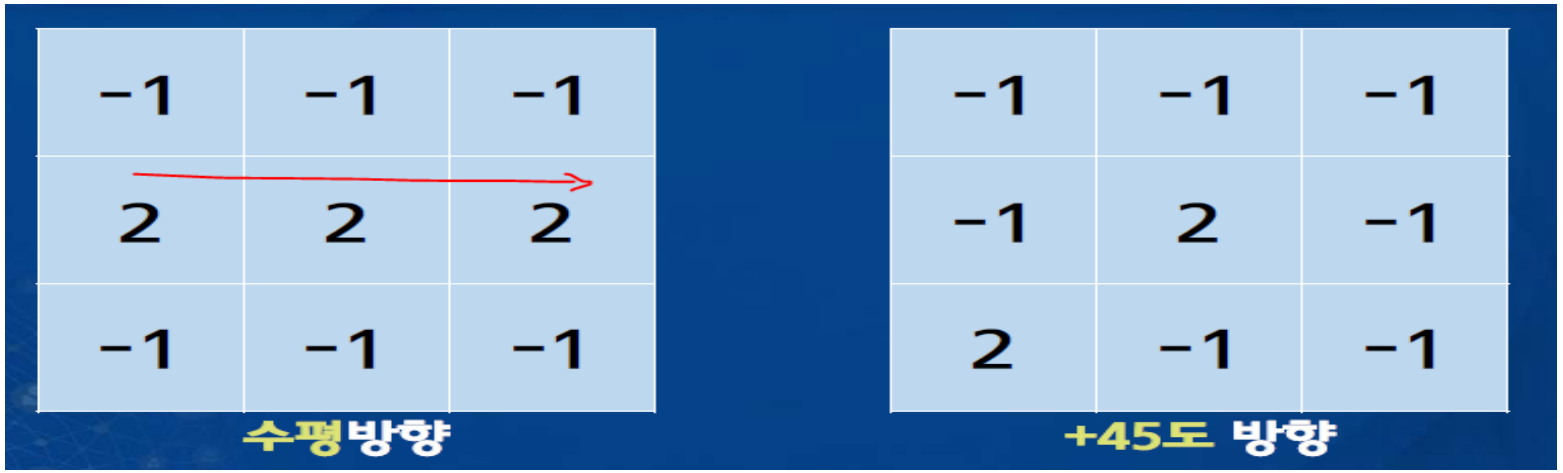


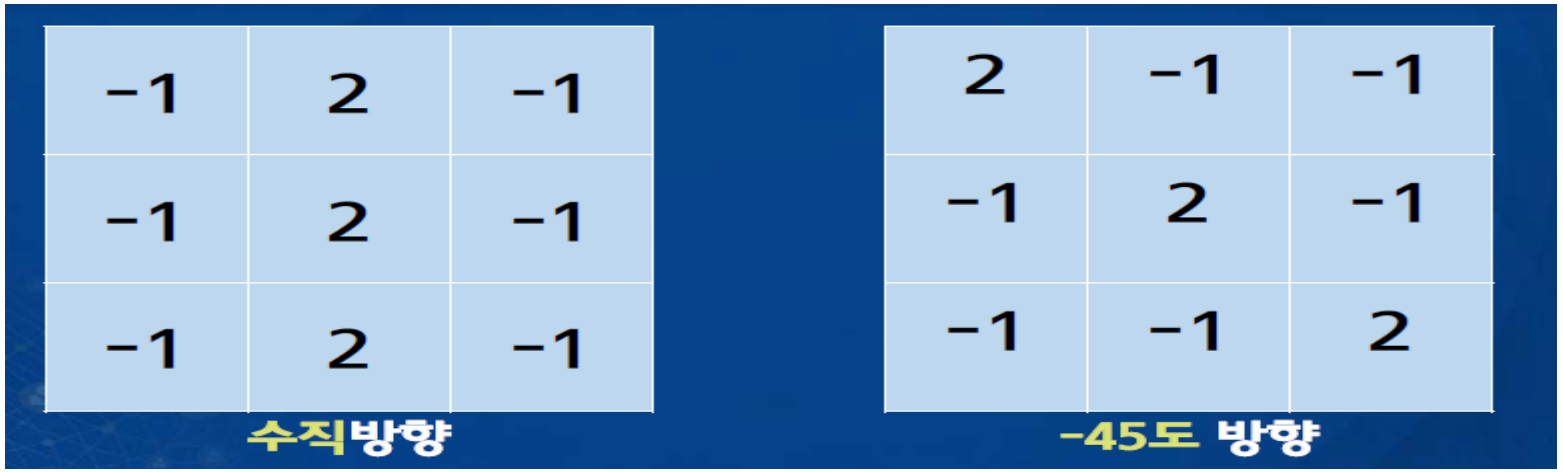
고역통과 필터

- 고주파 성분만 통과, High-Pass filter
- 영상의 edge 뿐 아니라 노이즈 성분도 증폭됨

=> 저역통과 후 고역통과 필터링 수행

경계선(Edge) 검출





모호한 경계를 구분하기 위해 필터링 된 영상을 이진화 수행

소스 코드

```
BYTE * Image = (BYTE *)malloc(ImgSize);
BYTE * Output = (BYTE *)malloc(ImgSize);
fread(Image, sizeof(BYTE), ImgSize, fp);
fclose(fp);
/* 영상처리 */
for (int i = 0; i < ImgSize; i++)
    Output[i] = 255;
```

1. Output 배열 정의 및 Mask 정의

```
const int size = 3;
double mask[size][size] = { -1.0, -1.0, -1.0,
                            0.0, 0.0, 0.0,
                            1.0, 1.0, 1.0 };
```

```
for (int i = 0; i < hInfo.biSizeImage; i++)
    if (Output[i] > 50) Output[i] = 255;
    else Output[i] = 0;
```

(마지막) 이진화

소스 코드

```
int margin = size / 2;
double SumProduct = 0.0;
for (int i = margin; i < hInfo.biHeight - margin; i++){ // Kernel Center의 Y좌표
    for (int j = margin; j < hInfo.biWidth - margin; j++){ // Kernel Center의 X좌표
        for (int m = -margin; m <= margin; m++){ // 커널 중심 기준 세로방향 이동
            for (int n = -margin; n <= margin; n++){ // 커널 중심 기준 가로방향 이동
                SumProduct += Image[(i+m)*hInfo.biWidth + (j+n)] *
                    mask[margin + m][margin + n];
            }
        }
        Output[i*hInfo.biWidth + j] = (BYTE)(abs(SumProduct) / 3.0);
        SumProduct = 0.0;
    }
}
```

위->아래

좌->우

중심 위->아래

중심 좌->우

컨벌루션 Mask 연산 수행, 중심값 계산

컨벌루션 Mask 연산 결과 output 배열에 저장, SumProduct 초기화

가능 범위 : -768~+768 -> 절대값 -> 3으로 나눠 0~255 맞춤