09장 이동,대칭,회전,워핑 기하학적 변환

- 영상의 이동 기하학적 변환
- 영상의 대칭 기하학적 변환
- 영상의 회전 기하학적 변환
- 영상의 워핑 기하학적 변환

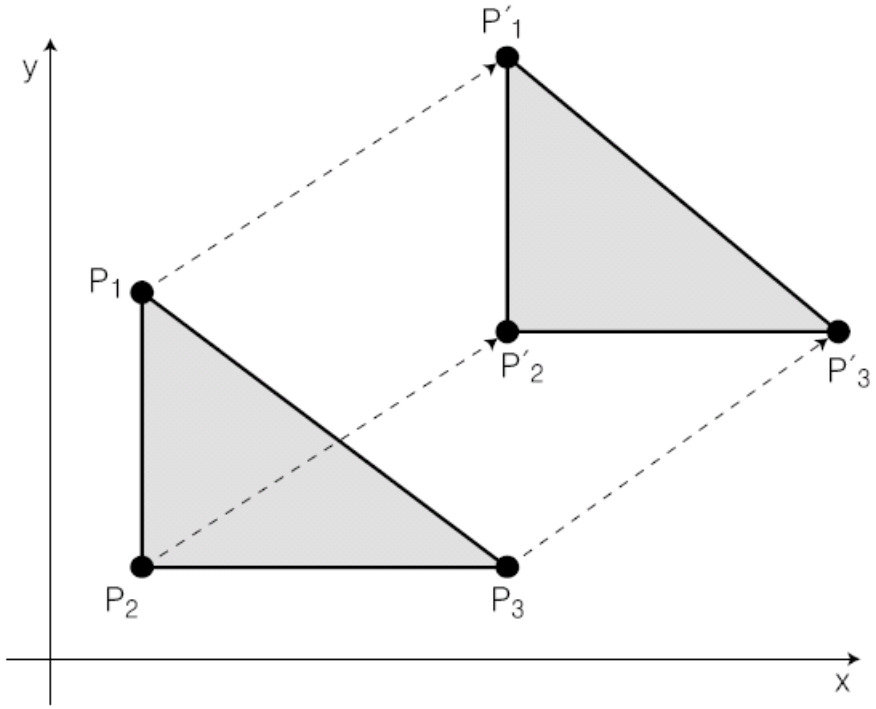
한빛미디어(주)

학습목표

- ✓ 이동의 기하학적 변환을 공부한다.
- ✓ 대칭 기하학적 변환을 공부한다.
- ✓ 회전 기하학적 변환의 원리를 학습한다.
- ✓ 회전 기하학적 변환에 고려할 사항을 소개한다.
- ✓ 워핑을 이해하고 수행 방법과 응용 분야를 소개한다.

Section 01 영상의 이동 기하학적 변환

- 이동(Translation) 기하학 변환
  - 디지털 영상을 평면의 한 위치에서 원하는 다른 위치로 옮기는 연산
  - 디지털 영상의 크기나 형태 등이 바뀌지 않아 원본 영상과 결과 같음.



[그림 9-1] 삼각형 물체의 이동과 공식

[실습하기 9-1] 이동 기하학 변환 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_TRANSLATION
Caption	영상 이동

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 영상 이동을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnTranslation
Doc Class	void	OnTranslation

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 9-1] 이동 기하학 변환 프로그램

```
void CImageProcessingDoc::OnTranslation()
{
   int i,j;
   int h_pos = 30, w_pos = 130;
   double **tempArray;

   m_Re_height = m_height;
   m_Re_width = m_width;
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];

   m_tempImage = Image2DMem(m_height, m_width);
   tempArray = Image2DMem(m_Re_height, m_Re_width);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         m_tempImage[i][j] = (double)m_InputImage[i*m_width + j];
      }
   }
```

[실습하기 9-1] 이동 기하학 변환 프로그램

```
    for(i=0 ; i<m_height - h_pos ; i++){
        for(j=0 ; j<m_width - w_pos ; j++){
            tempArray[i + h_pos][j + w_pos] = m_tempImage[i][j];
            // 입력 영상을 h_pos, w_pos만큼 이동
        }
    }

    for(i=0 ; i< m_Re_height ; i++){
        for(j=0 ; j< m_Re_width ; j++){
            m_OutputImage[i* m_Re_width + j]
                = (unsigned char)tempArray[i][j];
        }
    }

    delete [] m_tempImage;
    delete [] tempArray;
}
```

[실습하기 9-1] 이동 기하학 변환 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnTranslation()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnTranslation();

    Invalidate(TRUE);
}
```

[실습하기 9-1] 이동 기하학 변환 프로그램

⑤ 프로그램 실행 결과 영상

- 결과 영상이 오른쪽 아래로 이동함.

이동 기하학적 변환한 결과 영상

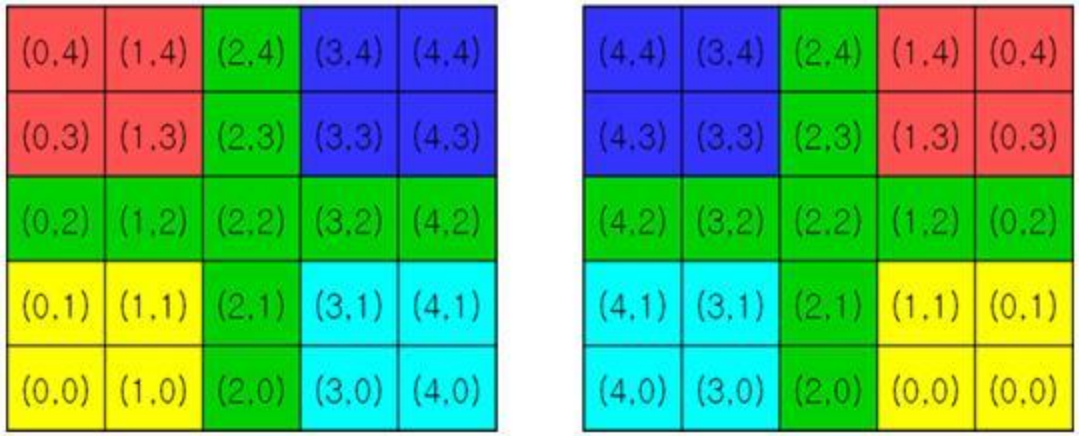
Section 02 영상의 대칭 기하학적 변환

- 대칭(Mirroring) 기하학 변환
  - 영상을 가로축이나 세로축으로 단순히 뒤집는 것
  - 세로축을 중심으로 뒤집는 것 → 좌우 대칭
  - 가로축을 중심으로 뒤집는 것 → 상하 대칭

좌우 대칭

- 좌우 대칭
  - 영상을 세로축을 중심으로 뒤집는 것
  - 즉, 영상 내의 한 수직선을 중심으로 왼쪽 화소와 오른쪽 화소를 서로 교환하는 것

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -(x - x_0) \\ y \end{bmatrix} + \begin{bmatrix} x_0 \\ 0 \end{bmatrix}$$



(a) 원시 영상 (b) 목적 영상

[그림 9-2] 이동 변환으로 화소 이동

[실습하기 9-2] 좌우 대칭 기하학 변환 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_MIRROR_HOR
Caption	영상 좌우 대칭

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 영상 좌우 대칭을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnMirrorHor
Doc Class	void	OnMirrorHor

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 9-2] 좌우 대칭 기하학 변환 프로그램

```
void CImageProcessingDoc::OnMirrorHor()
{
    int i,j;

    m_Re_height = m_height;
    m_Re_width = m_width;
    m_Re_size = m_Re_height * m_Re_width;

    m_OutputImage = new unsigned char [m_Re_size];

    for(i=0 ; i<m_height ; i++){
        for(j=0 ; j<m_width ; j++){
            m_OutputImage[i*m_width + m_width - j - 1] =
                m_InputImage[i*m_width + j];
            // 입력 영상의 배열 값을 출력 영상을 위한
            // 배열의 수평축 뒷자리부터 저장
        }
    }
}
```

[실습하기 9-2] 좌우 대칭 기하학 변환 프로그램

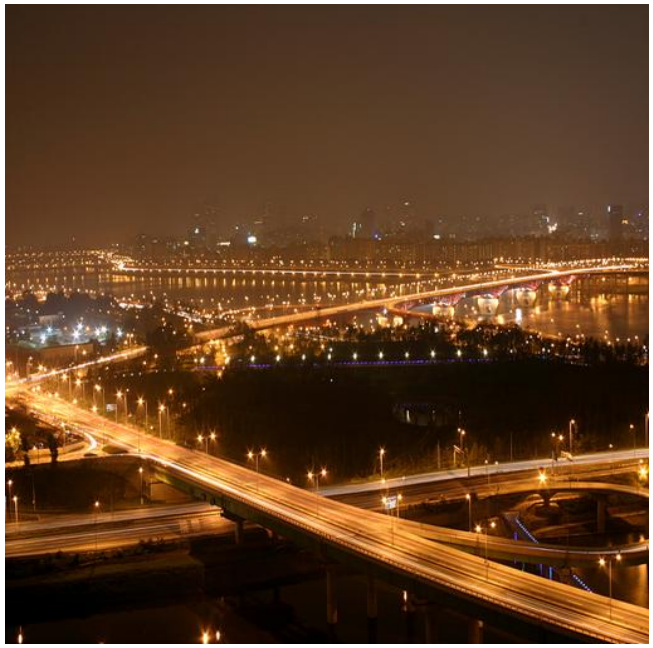
④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnMirrorHor()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnMirrorHor();

    Invalidate(TRUE);
}
```

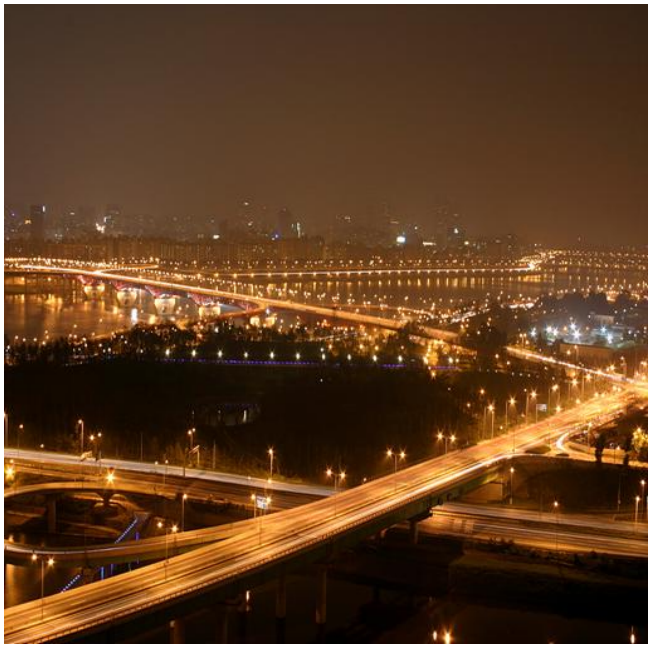
[실습하기 9-2] 좌우 대칭 기하학 변환 프로그램

⑤ 프로그램 실행 결과 영상

- 거울에 비치는 효과와 같다고 해서 좌우 대칭을 거울 영상이라고 함.



(a) 입력 영상



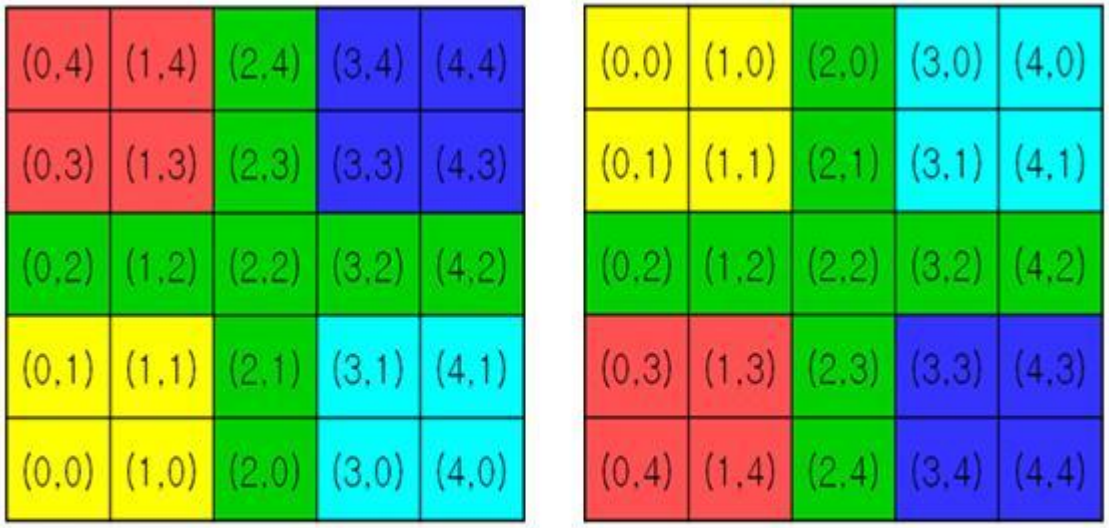
(b) 좌우 대칭 영상

좌우 대칭을 이용한 결과 영상

상하 대칭

- 상하 대칭
  - 영상을 가로축을 중심으로 뒤집는 것
  - 즉, 영상 내의 한 수평선을 중심으로 위쪽의 화소와 아래쪽의 화소를 교환하는 것

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ -(y - y_0) \end{bmatrix} + \begin{bmatrix} 0 \\ y_0 \end{bmatrix}$$



(a) 원시 영상 (b) 목적 영상

[그림 9-3] 좌우 대칭으로 화소 이동

[실습하기 9-3] 상하 대칭 기하학 변환 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_MIRROR_VER
Caption	영상 상하 대칭

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 영상 상하 대칭을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnMirrorVer
Doc Class	void	OnMirrorVer

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 9-3] 상하 대칭 기하학 변환 프로그램

```
void CImageProcessingDoc::OnMirrorVer()
{
    int i,j;

    m_Re_height = m_height;
    m_Re_width = m_width;
    m_Re_size = m_Re_height * m_Re_width;

    m_OutputImage = new unsigned char [m_Re_size];

    for(i=0 ; i<m_height ; i++){
        for(j=0 ; j<m_width ; j++){
            m_OutputImage[(m_height - i -1)*m_width + j]
                = m_InputImage[i*m_width + j];
            // 입력 영상의 값을 출력 영상을 위한 배열의 수직축 뒷자리부터 저장
        }
    }
}
```

[실습하기 9-3] 상하 대칭 기하학 변환 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnMirrorVer()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnMirrorVer();

    Invalidate(TRUE);
}
```

[실습하기 9-3] 상하 대칭 기하학 변환 프로그램

⑤ 프로그램 실행 결과 영상

- 입력 영상이 상하로 서로 뒤집힘.

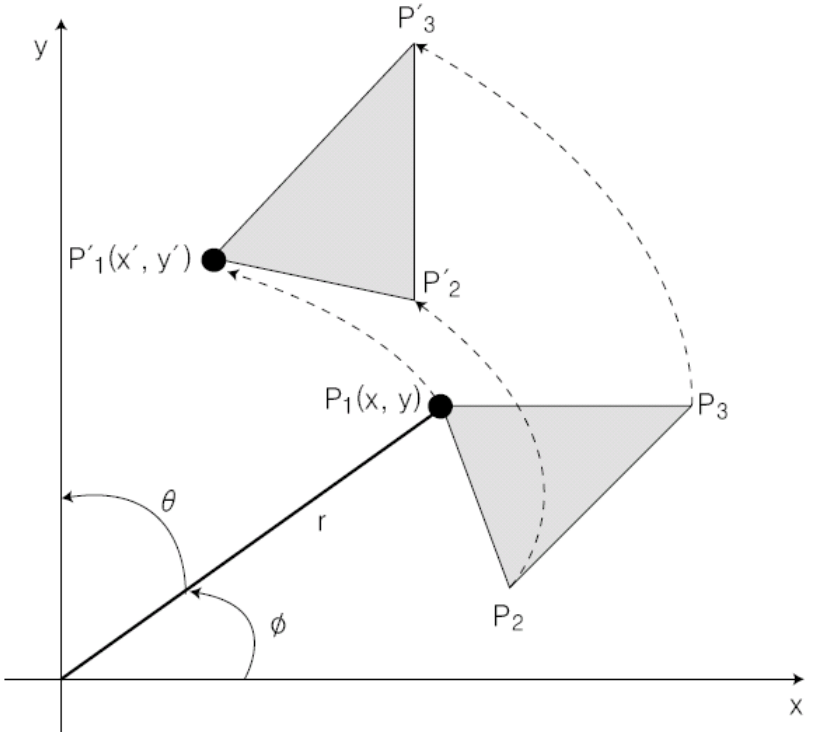
(a) 입력 영상 (b) 상하 대칭 영상

상하 대칭을 이용한 결과 영상

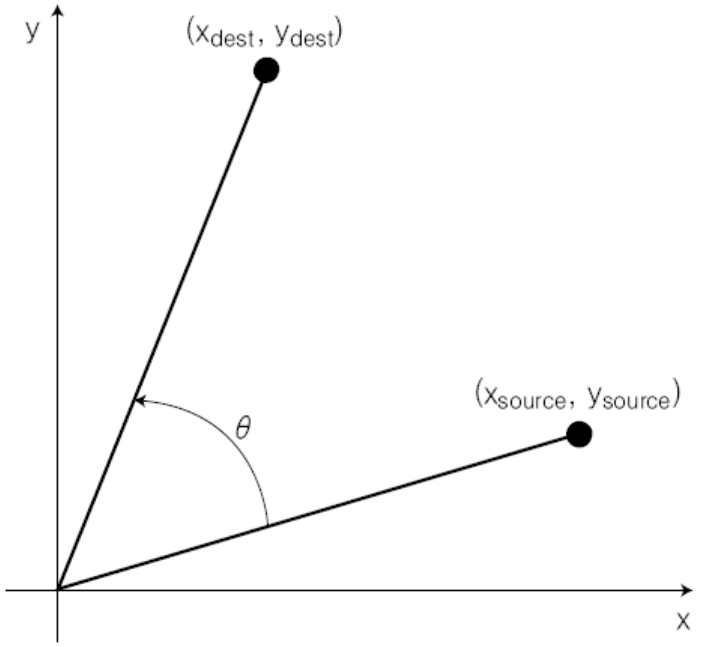
회전(Rotation) 기하학 변환

- 영상을 임의의 방향으로 특정한 각도 θ만큼 회전시키는 것

$$\begin{bmatrix} x_{dest} \\ y_{dest} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{source} \\ y_{source} \end{bmatrix}$$

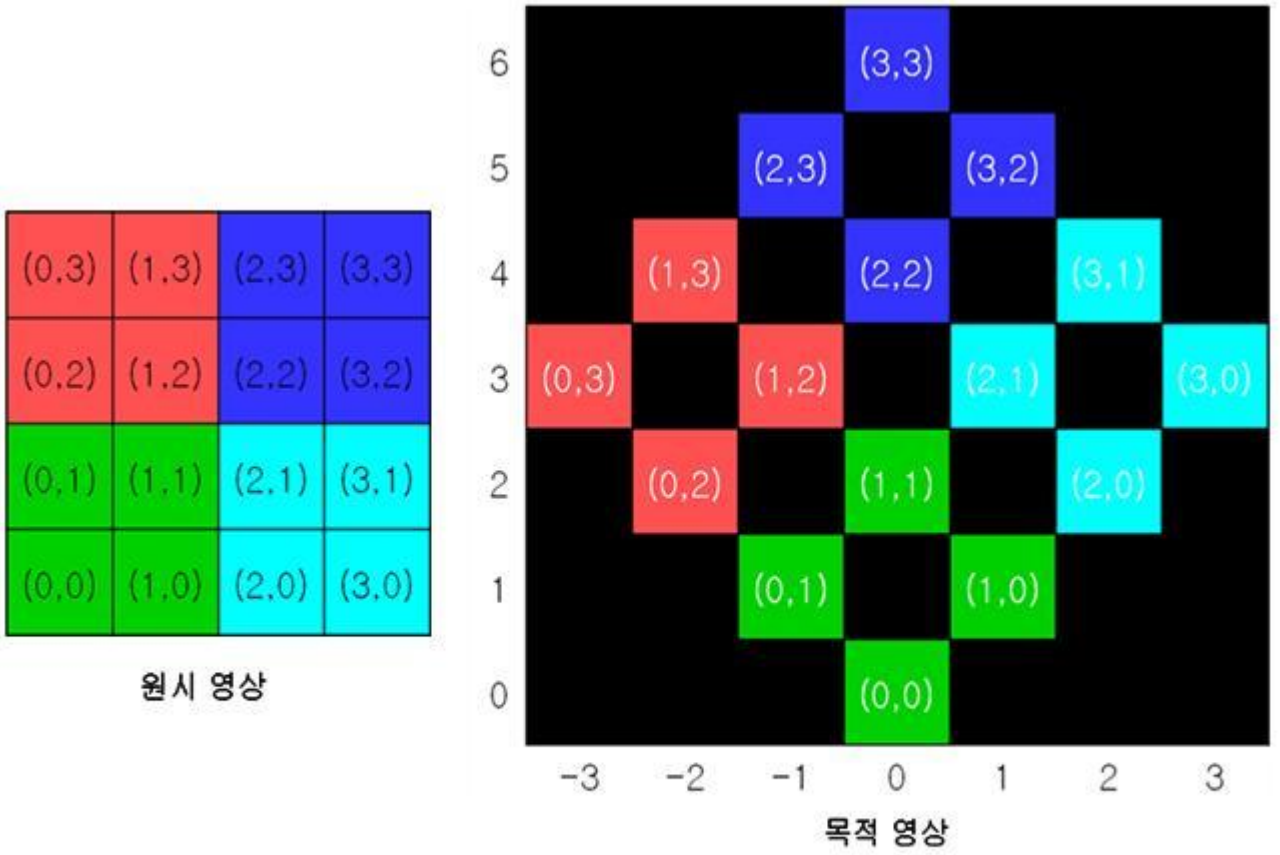


[그림 9-5] 삼각형 물체의 회전 변환



[그림 9-6] 회전 변환하여 좌표 변화

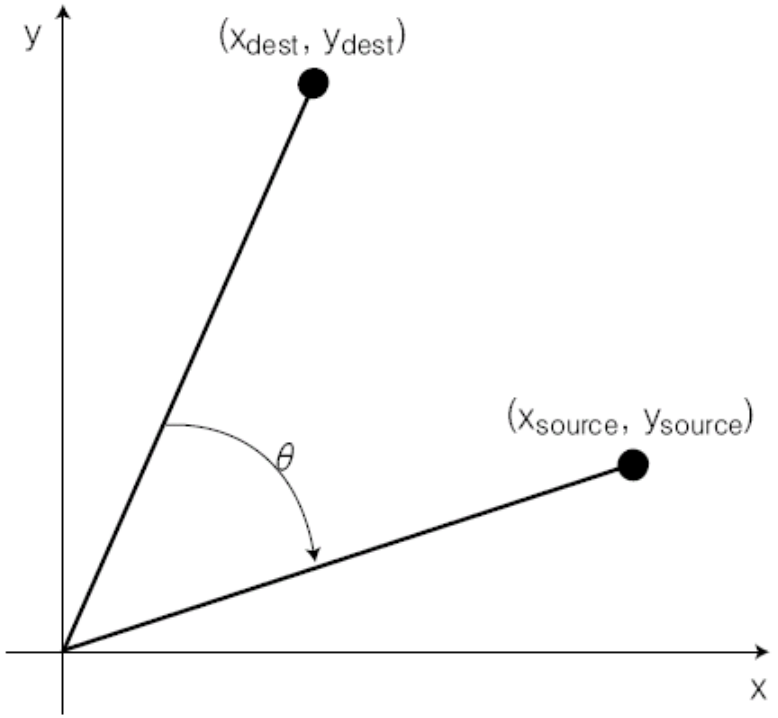
영상의 회전 기하학 변환(계속)



[그림 9-7] 회전 변환으로 이동된 화소

홀 문제를 해결하는 방안

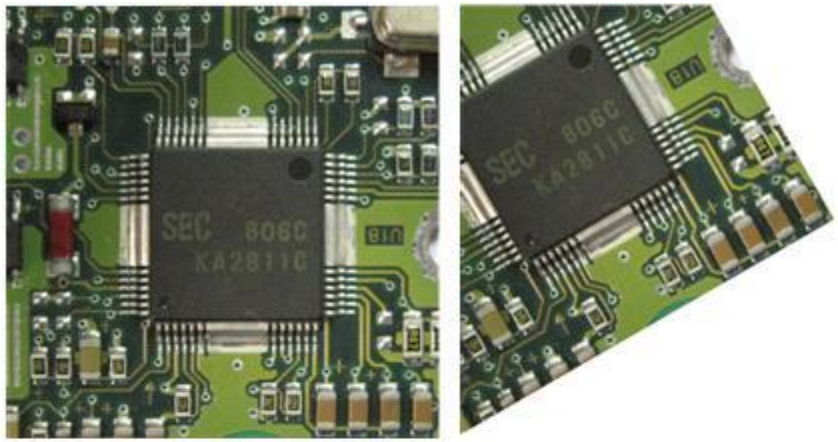
- 먼저 회전 기하학 변환을 역방향 사상으로 수행
- 영상의 회전 방향은 원하는 목적에 따라 다름
- 회전의 방향을 반시계 방향으로 설정한 예
  - 전방향 사상에서 회전 방향은 반시계방향, 역방향 사상에서 회전 방향은 시계방향이 됨



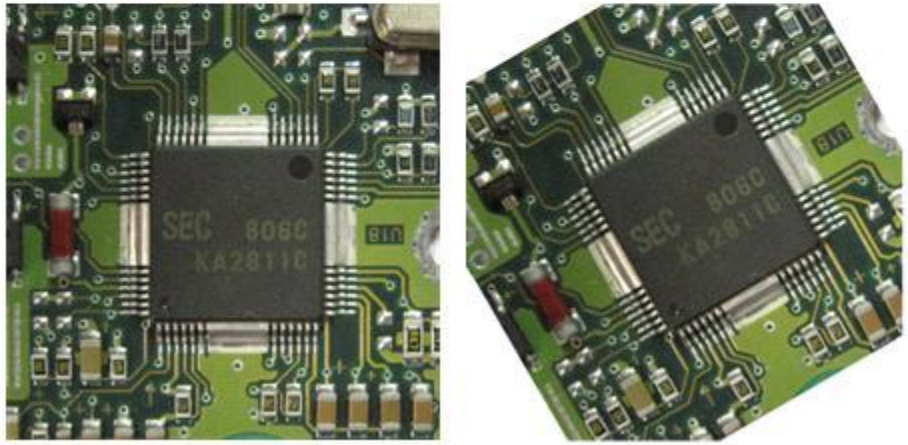
[그림 9-8] 홀 문제가 발생한 회전 결과 영상 [그림 9-9] 역방향 사상에서의 시계방향 회전

회전 결과 보이는 부분이 줄어드는 것을 방지하는 방법

- 회전 결과: 화소의 좌표 값이 음(-)
  - 실제 화소 좌표는 음의 값일 수 없으므로 해당 부분은 잘려 안 보이게 됨.



[그림 9-10] 원점을 기준으로 회전한 결과 영상



[그림 9-11] 영상의 중심점을 기준으로 회전한 결과 영상

회전 결과 보이는 부분이 줄어드는 것을 방지하는 방법(계속)

- 회전하려는 영상의 중심점이 (Cx, Cy)이고, 이 중심점을 기준으로 회전하는 전방향 사상 공식

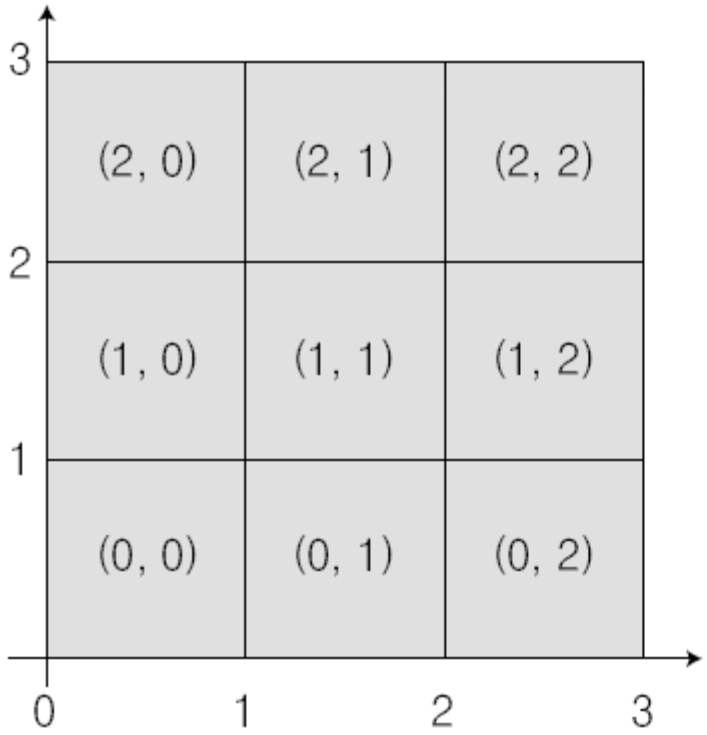
$$\begin{bmatrix} x_{dest} \\ y_{dest} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{source} - C_x \\ y_{source} - C_y \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix}$$

- 더 효율적으로 회전하기 위해 역방향 사상을 고려한 공식

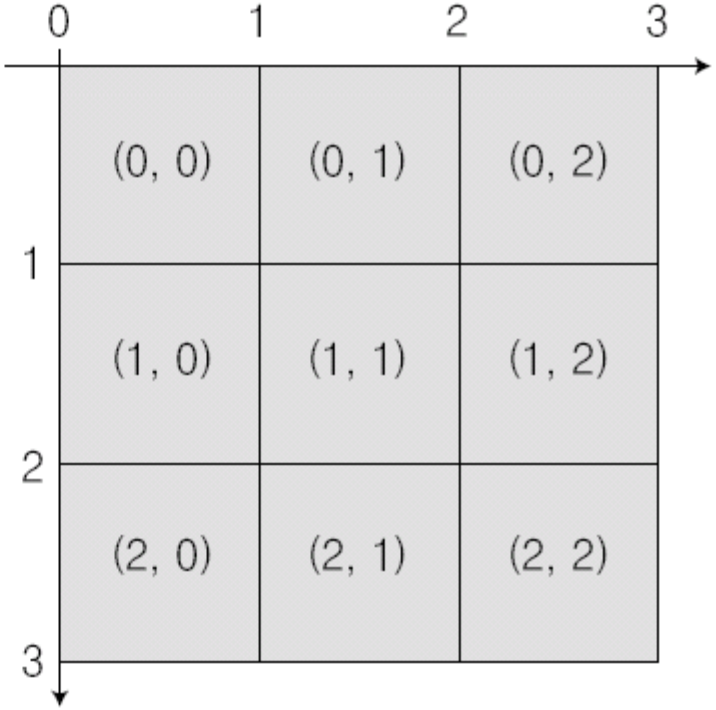
$$\begin{bmatrix} x_{source} \\ y_{source} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{dest} - C_x \\ y_{dest} - C_y \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix}$$

올바른 회전 방향이 되게 하는 해결 방안

- 영상 화소에서 좌표를 표시할 때는 왼쪽 위가 원점(수학 좌표에서는 왼쪽 아래가 원점).



(a) 수학 좌표계



(b) 화면 좌표계

[그림 9-12] 수학 좌표계와 화면 좌표계

올바른 회전 방향이 되게 하는 해결 방안(계속)

- 올바르게 회전하도록 하려면 화면 좌표를 수학적 좌표로 변환하여 회전한 뒤 다시 화면 좌표로 변환해야 함.
  - 이 과정을 반영한 전방향 사상 공식
    - **Hy**는 영상의 높이에서 **1**을 뺀 값

$$H_y = imageHeight - 1$$

$$x_{dest} = (x_{source} - C_x)\cos\theta - ((H_y - y_{source}) - C_y)\sin\theta + C_x$$

$$y_{dest} = (x_{source} - C_x)\sin\theta + (H_y - ((H_y - y_{source}) - C_y)\cos\theta) + C_y$$

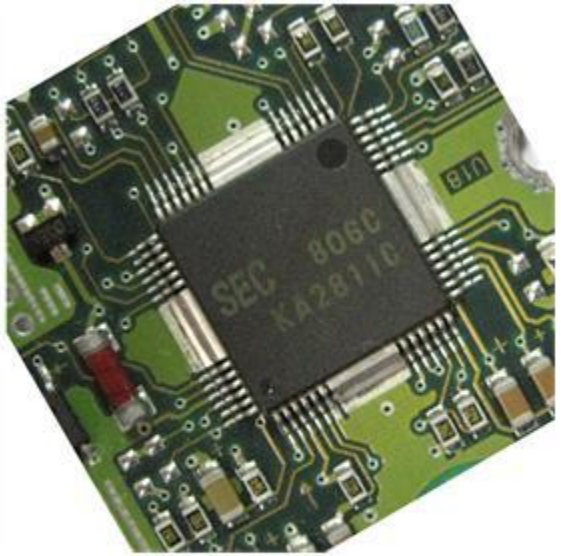
  - 역방향 사상 공식

$$x_{source} = (x_{dest} - C_x)\cos\theta + ((H_y - y_{dest}) - C_y)\sin\theta + C_x$$

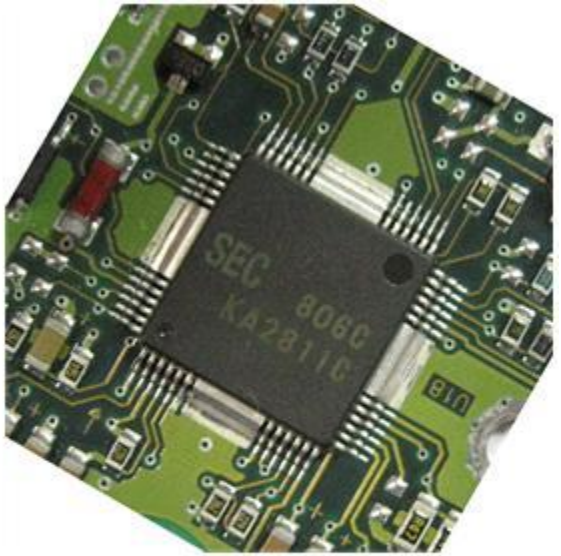
$$y_{source} = -(x_{dest} - C_x)\sin\theta + (H_y - ((H_y - y_{dest}) - C_y)\cos\theta) + C_y$$

올바른 회전 방향이 되게 하는 해결 방안(계속)

입력영상



좌표계 변환을 하지 않은 경우



좌표계 변환을 한 경우

[그림 9-13] 시계방향으로 회전하려고 좌표 변환

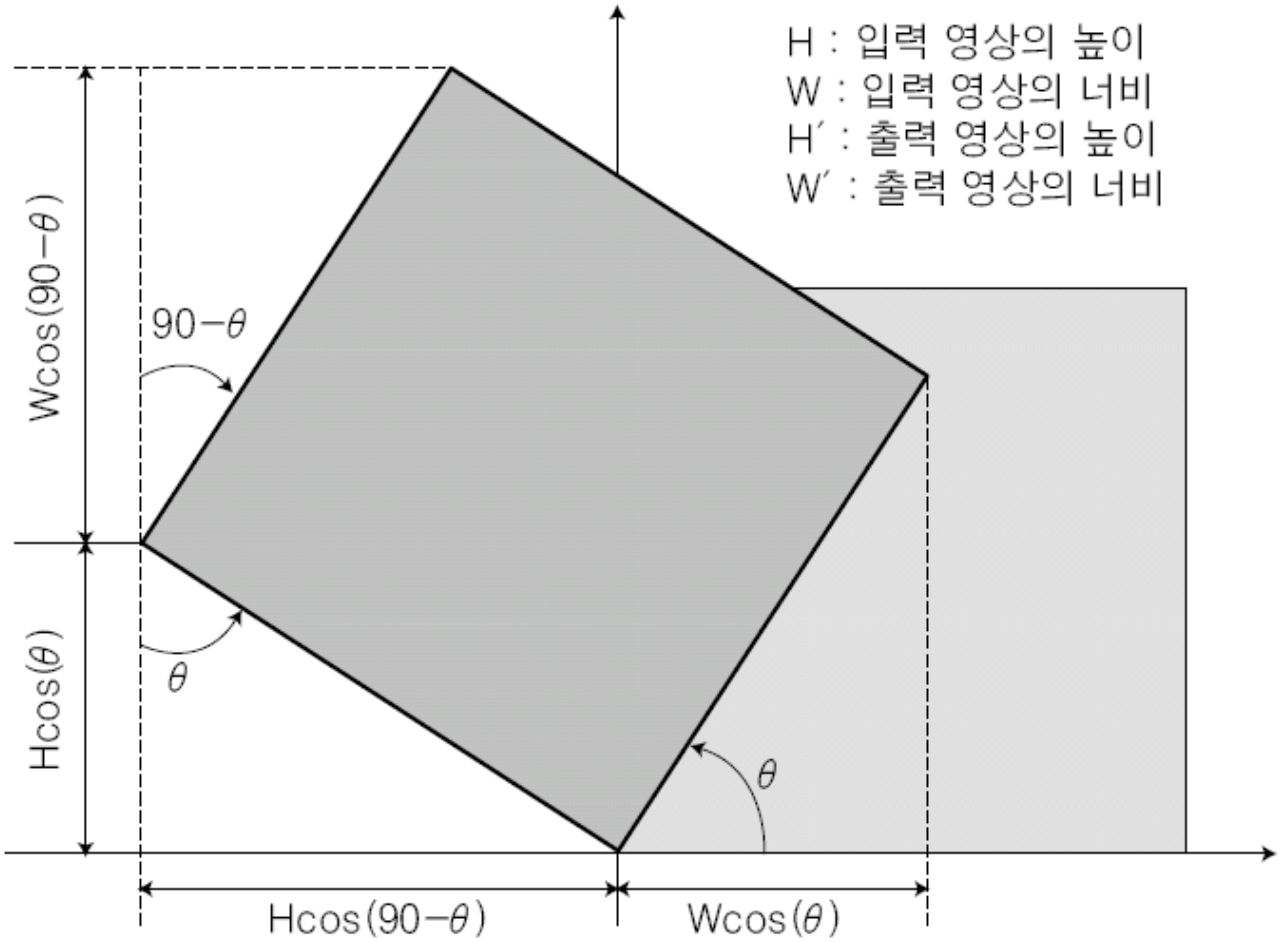
출력 영상의 크기를 고려한 회전 변환

- 회전 기하학 변환에서 입력 영상과 출력 영상의 크기를 같게 하면 출력 영상에서 잘려나가는 부분이 발생
- 회전의 기준을 원점에서 영상의 중심점으로 변경하면 이런 문제를 어느 정도 해결할 수는 있지만 그래도 잘려나가는 부분 발생
- 출력 영상에서 잘려 나가는 부분이 없게 하려면 출력 영상의 크기를 미리 계산해야 함.
- 출력 영상의 크기를 구하는 공식(회전 각도로 계산)
  - **H, W :** 원본 영상의 높이와 너비
  - **H', W':** 회전한 출력 영상의 높이와 너비

$$W' = H\cos(90-\theta) + \cos\theta$$

$$H' = H\cos\theta + W\cos(90-\theta)$$

출력 영상의 크기를 고려한 회전 변환(계속)



$$W' = H\cos(90-\theta) + W\cos(\theta)$$
$$H' = H\cos(\theta) + W\cos(90-\theta)$$

[그림 9-14] 회전 기하학적 변환의 출력 영상 크기

[실습하기 9-4] 영상의 회전 기하학 변환 프로그램

① **ResourceView 창에서 [Menu]-[IDR_IMAGETYPE] 더블클릭 → 메뉴 추가**

ID	ID_TRANSLATION
Caption	영상 이동

② **[MFC ClassWizard] 대화상자를 이용해 추가된 메뉴에서 영상 회전을 실행하는 함수 추가**

Class Name	Function Type	Function Name
View Class	void	OnMeanSub
Doc Class	void	OnMeanSub

③ **Doc 클래스에 다음 프로그램 추가**

[실습하기 9-4] 영상의 회전 기하학 변환 프로그램

```
void CImageProcessingDoc::OnRotation()
{
   int i, j, CenterH, CenterW, newH, newW, degree = 45;
   // degree = 회전할 각도
   double Radian, PI, **tempArray, Value;

   m_Re_height = m_height; // 회전된 영상의 높이
   m_Re_width = m_width; // 회전된 영상의 너비
   m_Re_size = m_Re_height * m_Re_width;

   m_OutputImage = new unsigned char [m_Re_size];
   PI = 3.14159265358979; // 회전각을 위한 PI 값

   Radian = (double)degree*PI / 180.0;
   // degree 값을 radian으로 변경
   CenterH = m_height / 2; // 영상의 중심 좌표
   CenterW = m_width / 2; // 영상의 중심 좌표

   m_tempImage = Image2DMem(m_height, m_width);
   tempArray = Image2DMem(m_Re_height, m_Re_width);

   for(i=0 ; i<m_height ; i++){
      for(j=0 ; j<m_width ; j++){
         m_tempImage[i][j] = (double)m_InputImage[i*m_width + j];
      }
   }
```

[실습하기 9-4] 영상의 회전 기하학 변환 프로그램

```
for(i=0 ; i<m_height ; i++){
    for(j=0 ; j<m_width ; j++){
        // 회전 변환 행렬을 이용하여 회전하게 될 좌표 값 계산
        newH = (int)((i-CenterH)*cos(Radian)
            - (j-CenterW)*sin(Radian) + CenterH);
        newW = (int)((i-CenterH)*sin(Radian)
            + (j-CenterW)*sin(Radian) + CenterW);

        if(newH < 0 || newH >= m_height){
            // 회전된 좌표가 출력 영상을 위한 배열 값을 넘어갈 때
            Value = 0;
        }
        else if(newW < 0 || newW >= m_width){
            // 회전된 좌표가 출력 영상을 위한 배열 값을 넘어갈 때
            Value = 0;
        }
        else{
            Value = m_tempImage[newH][newW];
        }
        tempArray[i][j] = Value;
    }
}
```

[실습하기 9-4] 영상의 회전 기하학 변환 프로그램

```
      for(i=0 ; i< m_Re_height ; i++){
      for(j=0 ; j< m_Re_width ; j++){
         m_OutputImage[i* m_Re_width + j]
            = (unsigned char)tempArray[i][j];
      }
   }

   delete [] m_tempImage;
   delete [] tempArray;
}
```

[실습하기 9-4] 영상의 회전 기하학 변환 프로그램

④ View 클래스에 다음 프로그램 추가

```
void CImageProcessingView::OnRotation()
{
    CImageProcessingDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDoc->OnRotation();

    Invalidate(TRUE);
}
```

[실습하기 9-4] 영상의 회전 기하학 변환 프로그램

⑤ 프로그램 실행 결과 영상

- 출력 영상의 크기를 계산해서 회전한 뒤 얻은 출력 영상으로 잘려나간 부분이 없음.

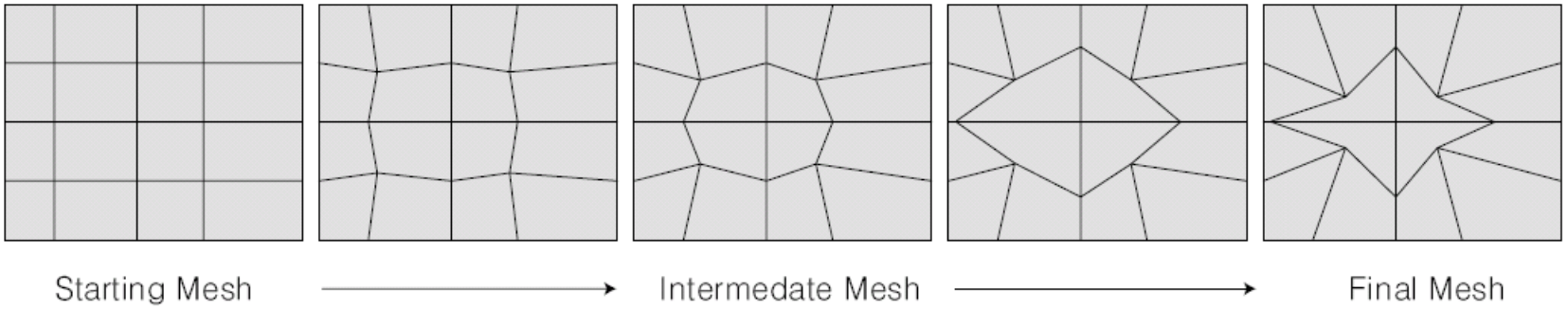
(a) 입력 영상

(b) 출력 영상

출력 영상의 크기를 고려하여 회전 기하학적 변환한 출력 영상

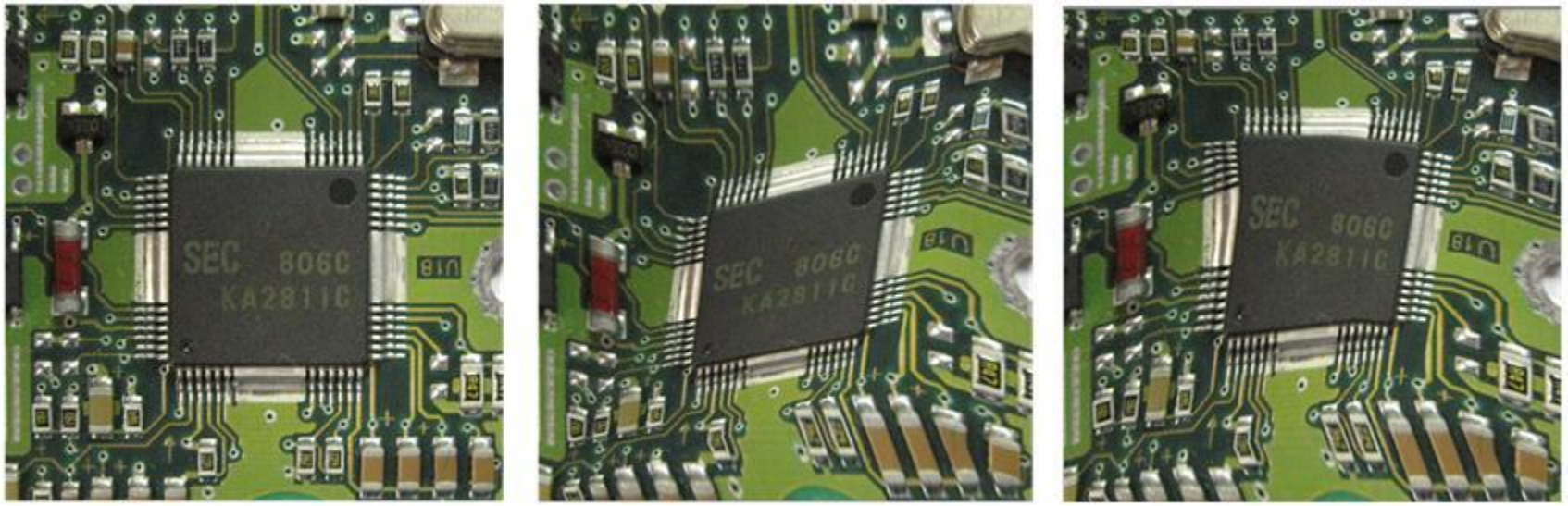
Section 04 영상의 워핑 기하학적 변환

- 워핑(Warping)
  - 비선형이나 왜곡 변환을 수행
  - 고무시트 변환(Rubber Sheet Transform)이라고도 함.
  - 화소별로 이동 정도를 달리해 고무판 위에 그린 영상을 임의로 구부린 듯한 효과를 냄.
- 매시 워핑(Mesh Warping)
  - 입력 영상을 작은 삼각형이나 사각형 격자(Mesh)로 나눠서 변형시켜 목적하는 결과 영상을 얻음 .
  - 격자의 모양인 다각형의 기하학적 변형 수행



[그림 9-15] 매시 워핑 알고리즘의 동작

영상의 워핑 기하학적 변환(계속)



[그림 9-16] 워핑 처리된 결과 영상

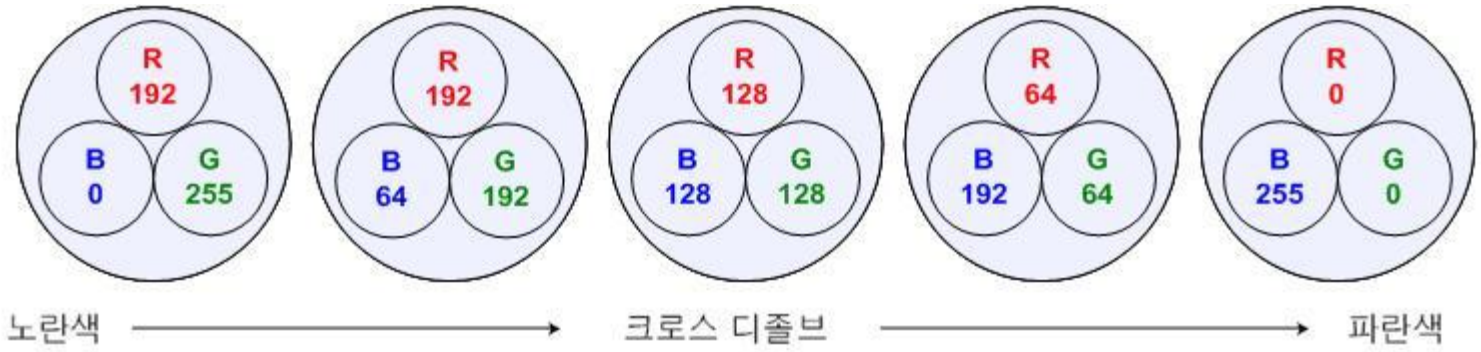
모핑 기술

- 변형(Metamorphosis)에서 유래된 모핑(Morphing)은 한 영상을 서서히 변화시켜 다른 영상으로 변환하는 기술
- 원본 영상과 최종 영상은 물론, 최종 영상으로 매끄럽게 변할 수 있도록 많은 중간 단계의 영상도 필요함.

초기영상 중간영상 최종영상

[그림 9-17] 중간 단계 영상을 생성하는 모핑의 수행 과정

- 중간 단계의 영상을 생성하려면 워핑과 합병의 두 단계를 거쳐야 함.
- 영상의 워핑 기술: 두 영상의 중요한 특징을 좀더 가깝게 정렬하는 데 사용
- 합병: 영상이 순차적으로 자연스럽게 융합되도록 하는 크로스 디졸브 (Cross-dissolve)로 처리됨.



[그림 9-18] 크로스 디졸브의 원리

- 워핑과 합병 처리로 모핑을 수행하는 과정

[그림 9-19] 모핑의 동작 원리

- 전이 모핑 : 초기 영상과 최종 영상이 달라 전혀 다른 내용으로 변하는 것
- 왜곡 모핑: 같은 내용에서 일부분이 변하는 것, 워핑의 개념과 비슷하나 중간 영상을 생성하여 변하는 과정을 보여준다는 차이가 있음.

요약

- 워핑(Warping)
  - 비선형이나 왜곡 변환 수행
  - 화소별로 이동 정도를 달리해 고무판 위에 그린 영상을 임의로 구부린 듯한 효과를 냄.
- 매시 워핑(Mesh Warping)
  - 입력 영상을 작은 삼각형이나 사각형 격자(Mesh)로 나눠서 변형시켜 목적하는 결과 영상을 얻음.
  - 격자의 모양인 다각형의 기하학적 변형을 수행
- 모핑(Morphing)
  - 변형(Metamorphosis)에서 유래
  - 영상을 서서히 변화시켜 다른 영상으로 변환하는 기술
  - 원본 영상과 최종 영상은 물론, 최종 영상으로 매끄럽게 변할 수 있도록 많은 중간 단계의 영상도 필요함.

요약

- 모핑에서 중간 단계의 영상을 생성하려면 워핑과 합병의 두 단계를 거쳐야 함.
  - 워핑 기술: 두 영상의 중요한 특징을 좀더 가깝게 정렬하는 데 사용
  - 합병: 영상이 순차적으로 자연스럽게 융합되도록 하는 크로스 디졸브(Cross-dissolve)로 처리됨.

- 전이 모핑
  - 초기 영상과 최종 영상이 달라 전혀 다른 내용으로 변하는 것

- 왜곡 모핑
  - 같은 내용에서 일부분이 변하는 것
  - 워핑의 개념과 비슷하나 중간 영상을 생성하여 변하는 과정을 보여 준다는 차이가 있음.

Thank you