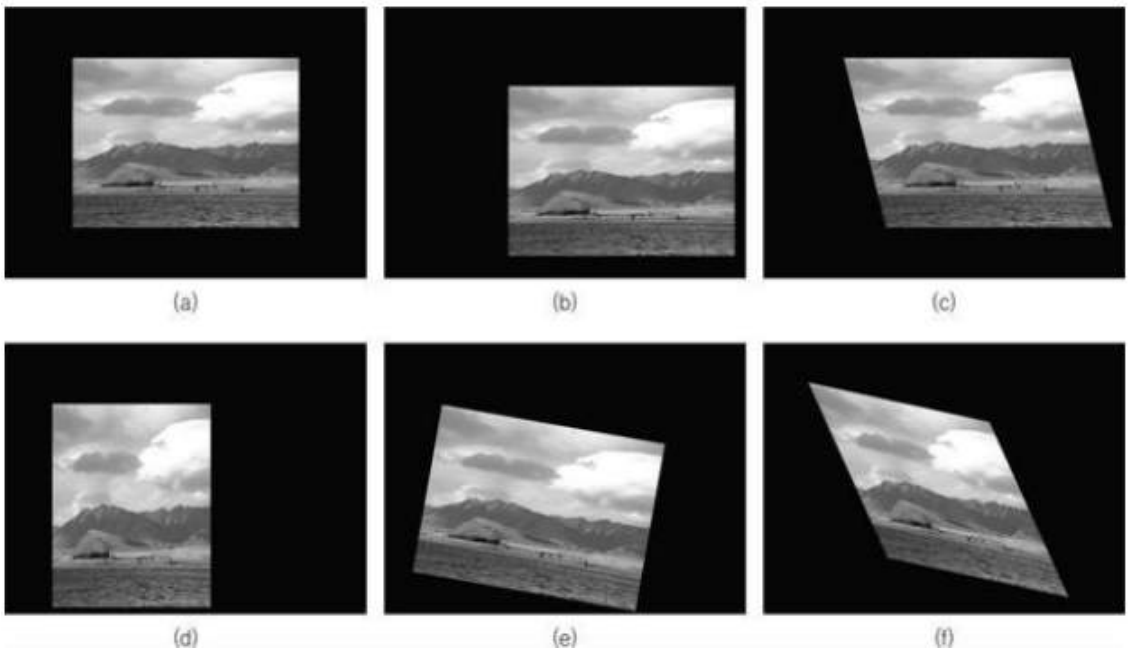
디지털 영상처리 연구실 연구보고서

정지우

<영상의 기하학적 변환>

- **어파인 변환(affine transformation)**이란 영상의 평행 이동, 확대 및 축소, 회전 등의 조합으로 만들 수 있는 기하학적 변환을 나타낸다.

+ affine은 프랑스어로써 유사한, 비슷한 것을 의미한다.



우선 기하학적 변환의 방법은 다음과 같은 함수의 형태로 나타낼 수 있다.

$$\begin{cases} x' = f_1(x, y) \\ y' = f_2(x, y) \end{cases}$$

- 입력영상에서 (x, y)좌표를 f라는 함수를 통해 (x', y')로 변환하는 작업이고 이러한 작업을 수행하기 위해서는 어파인 행렬을 알아야한다.

<어파인 변환>

$$\begin{cases} x' = f_1(x,y) = ax + by + c \\ y' = f_2(x,y) = dx + ey + f \end{cases}$$

이렇게 어파인 변환을 결정하는 6개의 파라미터가 존재한다.

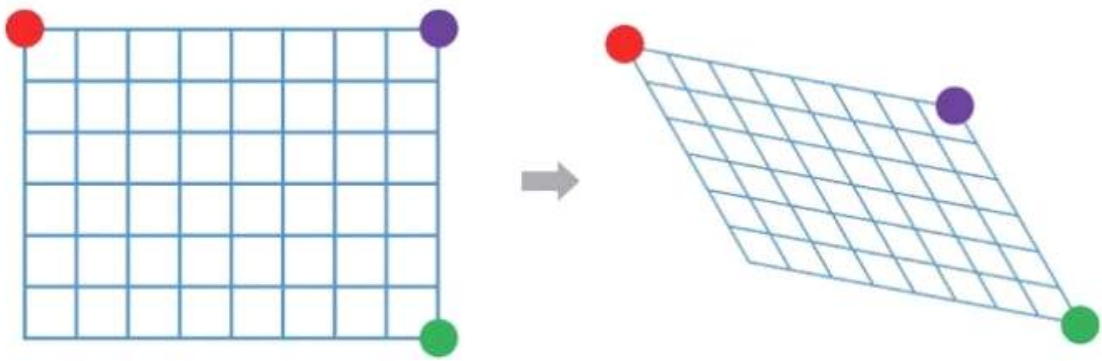
이를 행렬을 통해 하나의 수식으로 표현한다면,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ d & e \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ f \end{pmatrix}$$

(x, y)에 가상의 좌표 1을 하나 추가하여 (x, y, 1) 형태로 바꾸면,

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

이렇게 6개의 파라미터로 구성된 2x3 행렬을 어파인 변환 행렬이라고 부른다.



입력 영상과 어파인 변환 결과 영상으로부터 어파인 변환 행렬을 구하기 위해서 최소 3점의 이동 관계를 알아야 한다.

점 1개의 이동 관계로부터 x 좌표와 y 좌표에 대한 변환 수식 2개를 얻을 수 있고,

점 3개의 이동 관계로부터 총 6개의 방정식을 구할 수 있다.

즉, 점 3개의 이동 관계를 알고 있다면 6개의 원소로 정의되는 어파인 변환 행렬을 구할 수 있다.

<어파인 변환 행렬 생성 함수>

getAffineTransform();

```
Mat getAffineTransformat(const Point2f src[], const Point dst[]);

src : 입력 영상에서 3점의 좌표
dst : 결과 영상에서 3점의 좌표
반환값 : 2 x 3 어파인 변환 행렬
```

+) Point2f란?

OpenCV는 자주 사용하는 자료형에 대하여 Point_클래스 이름을 재정의하여 제공
ex. 정수형 int 자료형 - Point2i 클래스 -> 자주 사용하기때문에 Point 클래스로 재정의
float 자료형 - Point2f 클래스

```
typedef Point_<int>     Point2i;
typedef Point_<int64>   Point2l;
typedef Point_<float>   Point2f;
typedef Point_<double> Point2d;
typedef Point2i          Point;
```

<어파인 변환한 결과영상 생성 함수>

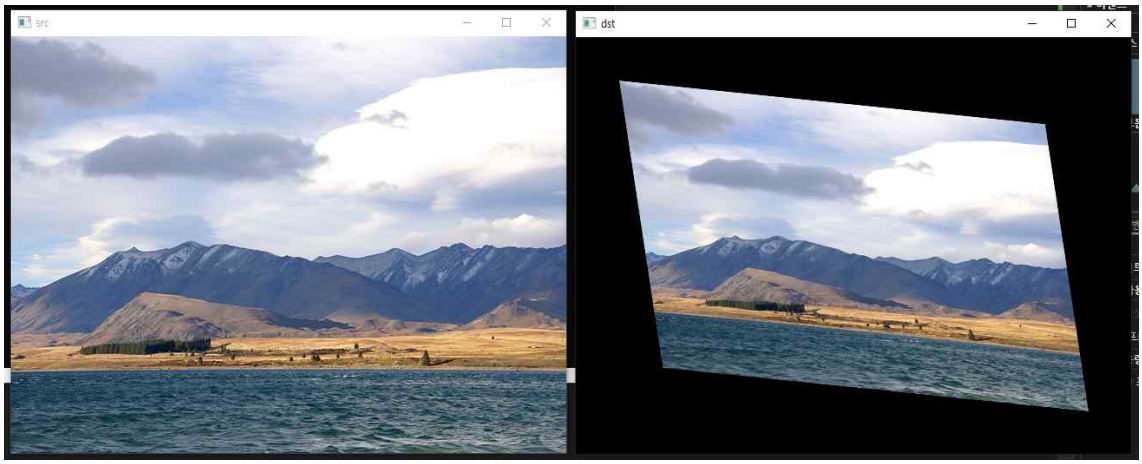
warpAffine();

```
void warpAffine(InputArray src, OutputArray dst, InputArray M, Size dsize, int flags = INTER_LINEAR,
    int borderMode = BORDER_CONSTANT, const Scalar& borderValue = Scalar());

src: 입력 영상
dst : 결과 영상
M : 2 x 3 어파인 변환 행렬
dsize : 결과 영상 크기
flags : 보간법 알고리즘
borderMode : 가장자리 픽셀 확장 방식
borderValue : bordermode가 BORDER_CONSTANT일 때 사용할 상수 값
```

- 입력, 결과 영상, 행렬, 크기만 지정해주면된다.

<어파인 변환 예제>

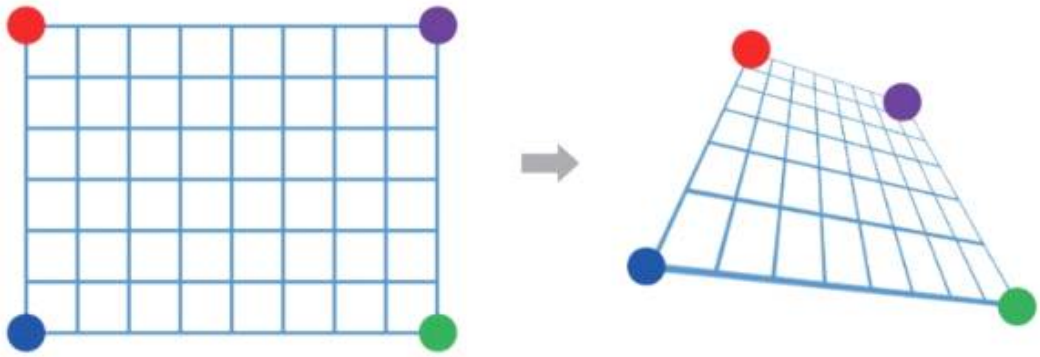
```
Point2f srcPts[3], dstPts[3];
srcPts[0] = Point2f(0, 0);
srcPts[1] = Point2f(src.cols - 1, 0);
srcPts[2] = Point2f(src.cols - 1, src.rows - 1);
dstPts[0] = Point2f(50, 50);
dstPts[1] = Point2f(src.cols - 100, 100);
dstPts[2] = Point2f(src.cols - 50, src.rows - 50);

Mat M = getAffineTransform(srcPts, dstPts);

Mat dst;
warpAffine(src, dst, M, Size());

imshow("src", src);
imshow("dst", dst);
```



<투시 변환>

투시 변환(perspective transformation)은 직사각형 형태의 영상을 임의의 볼록 사각형 형태로 변경할 수 있는 변환이다.

투시 변환에 의해 원본 영상에 있던 직선은 결과 영상에서도 직선성이 그대로 유지되지만, 두 직선의 평행 관계는 깨질 수 있다.



위 그림과 같이 4개의 점을 이용해서 투시변환 행렬을 만들 수 있다.

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = M_P \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Mp가 투시변환을 위한 행렬이고 9개의 파라미터를 가진다.

<투시 변환 행렬생성 함수>

getPerspectiveTransform();

```
Mat getPerspectiveTransform(InputArray src, InputArray dst, int solveMethod = DECOMP_LU);

src: 입력 영상에서 4점의 좌표
dst : 결과 영상에서 4점의 좌표
solveMethod : 계산 방법 지정
반환값 : 3 x 3 크기의 투시 변환 행렬
```

<투시 변환한 결과영상 생성 함수>

warpPerspective();

```
void warpPerspective(InputArray src, OutputArray dst, InputArray M, Size dsize, int flags = INTER_LINEAR,
    int borderMode = BORDER_CONSTANT, const Scalar& borderValue = Scalar());

src: 입력 영상
dst : 결과 영상으로 src와 같은 타입의 dsize의 크기를 갖는다.
M : 3 x 3 투시 변환 행렬
dsize : 결과 영상의 크기
flags : 보간법 알고리즘
borderMode : 가장자리 픽셀 확장 방식
borderValue : boderMode가 BORDER_CONSTANT일 때 사용할 상수 값
```

- 입력, 결과 영상, 행렬, 크기만 지정해주면된다.

<투시 변환 예제>

```
srcPts[cnt++] = Point2f(x, y);

circle(src, Point(x, y), 5, Scalar(0, 0, 255), -1);
imshow("src", src);

if (cnt == 4) {
    int w = 200, h = 300;

    dstPts[0] = Point2f(0, 0);
    dstPts[1] = Point2f(w - 1, 0);
    dstPts[2] = Point2f(w - 1, h - 1);
    dstPts[3] = Point2f(0, h - 1);

    Mat pers = getPerspectiveTransform(srcPts, dstPts);

    Mat dst;
    warpPerspective(src, dst, pers, Size(w, h));

    imshow("dst", dst);
}
```