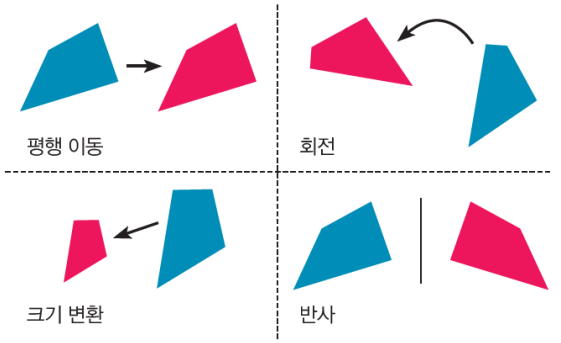
7장 기하학적 처리

1

- 기하학적 변환(geometric transformation)은 영상을 이동하거나 영상의 모양을 변형하는 처리



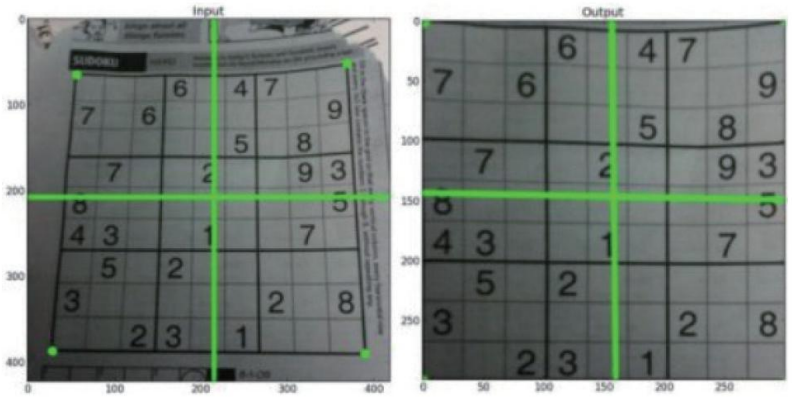
2

기본적인 기하학적 변환

- 평행이동translation : 영상을 평행 이동한다.
- 크기변환scaling : 영상의 크기를 변경한다.
- 회전rotation : 영상을 회전시킨다.

3

기하학적 변환의 이유

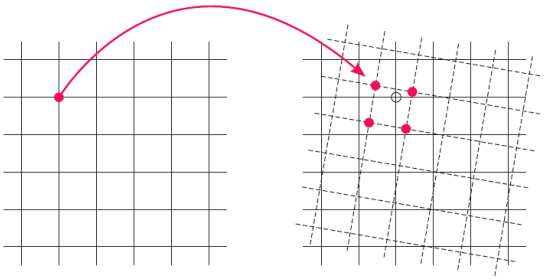


출처: OpenCV 튜토리얼

4

2가지의 절차

1. 입력 영상의 화소가 출력 영상에서 어디로 가느냐를 계산
2. 소수점 위치에 놓인 화소의 값을 주위 화소들을 이용하여 추정하는 절차



5

순방향 변환

- 입력 영상에서 (x, y) 위치에 있는 화소가 출력 영상 안의 새로운 좌표(*x′*, *y′*)로 재배치되는 것

$$I(x, y) \rightarrow O(x', y')$$

6

순방향 변환식

$$x' = x + T_x$$
$$y' = y + T_y$$

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

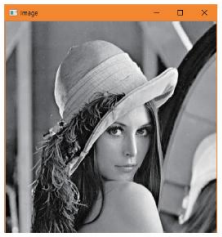
$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$
$$y' = x \cdot \sin\theta - y \cdot \cos\theta$$

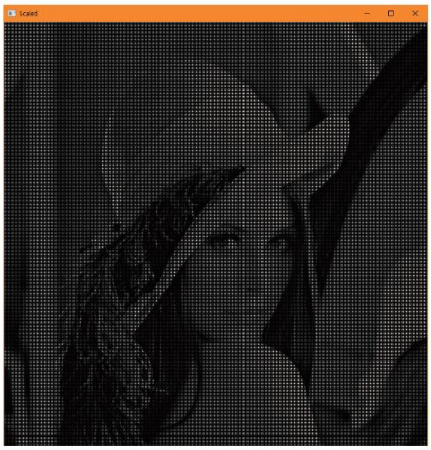
7

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
	Mat dst = Mat::zeros(Size(src.cols*2, src.rows*2), src.type());

	for (int y = 0; y < src.rows; y++) {
		for (int x = 0; x < src.cols; x++) {
			const int newX = x * 2;
			const int newY = y * 2;
			dst.at<uchar>(newY, newX) = src.at<uchar>(y, x);
		}
	}
	imshow("Image", src);
	imshow("Scaled", dst);
	waitKey(0);
	return 1;
}
```

8

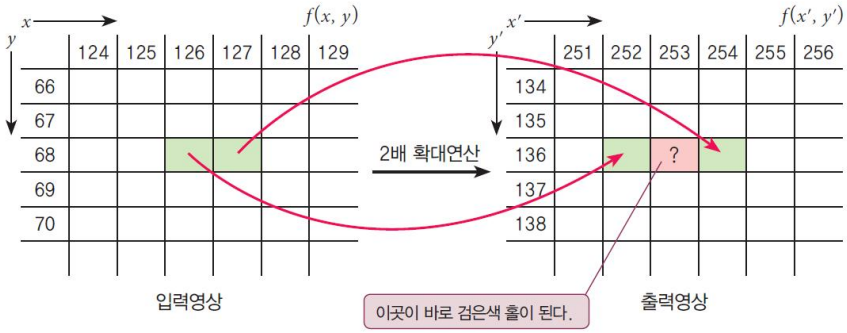
실행결과





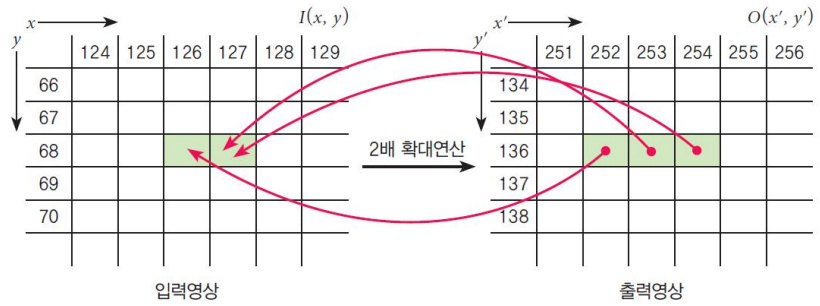
9

검정색 홀의 이유



10

역방향 사상



11

역방향 사상

$$O(x, y) \leftarrow I(x', y')$$

12

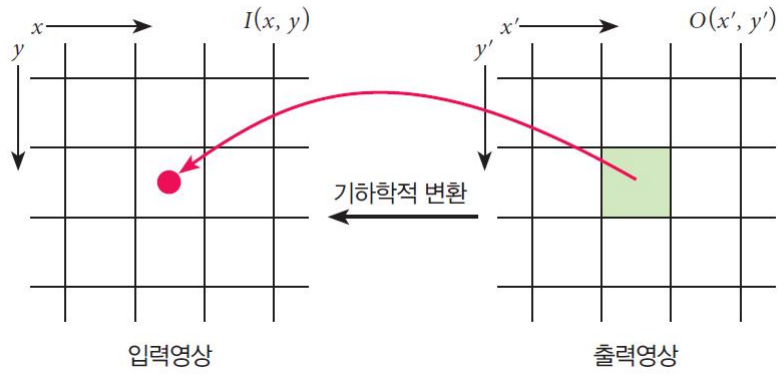
```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
	Mat dst = Mat::zeros(Size(src.cols*2, src.rows*2), src.type());

	for (int y = 0; y < dst.rows ; y++) {
		for (int x = 0; x < dst.cols; x++) {
			const int newX = x / 2.0;
			const int newY = y / 2.0;
			dst.at<uchar>(y, x) = src.at<uchar>(newY, newX);
		}
	}
	imshow("Image", src);
	imshow("Scaled", dst);
	waitKey(0);
	return 1;
}
```

13

실행결과

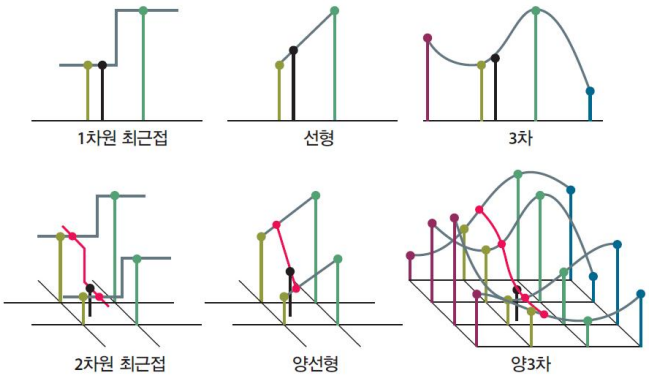
14

보간법



15

보간법의 종류



By Cmglee – Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53064904

16

최근접 보간법

- 변환된 위치와 가장 가까운 화소값을 사용하는 방법

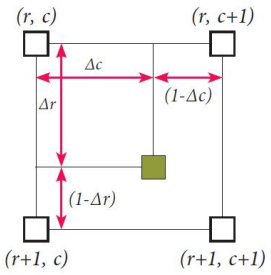
$$x_{nn} = (int)(x_{float} + 0.5)$$

$$y_{nn} = (int)(y_{float} + 0.5)$$

17

양선형 보간법

- 우리가 이미 알고 있는 4개의 인접 화소의 값을 이용한다.
- 양선형 보간법은 비례식을 이용하여 중간에 놓인 화소의 값을 추정하는 방법이다



18

양선형 보간법

$$O(r', c') = \mathbf{I}(r, c) \cdot (1 - \Delta r) \cdot (1 - \Delta c) + \mathbf{I}(r + 1, c) \cdot \Delta r \cdot (1 - \Delta c) + \mathbf{I}(r, c + 1) \cdot (1 - \Delta r) \cdot \Delta c + \mathbf{I}(r + 1, c + 1) \cdot \Delta r \cdot \Delta c$$

19

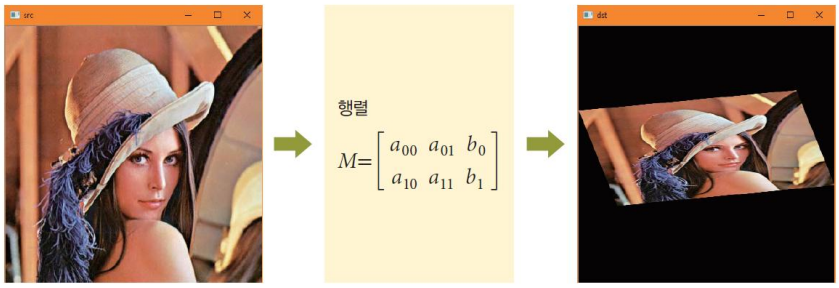
```
float Lerp(float s, float e, float t) {
        return s + (e - s) * t;
}

float Blerp(float c00, float c10, float c01, float c11, float tx, float ty) {
        return Lerp(Lerp(c00, c10, tx), Lerp(c01, c11, tx), ty);
}

float GetPixel(Mat img, int x, int y)
{
        if (x > 0 && y > 0 && x < img.cols && y < img.rows)
                return (float)(img.at<uchar>(y, x));
        else
                return 0.0;
}
```

20

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
	Mat dst = Mat::zeros(Size(src.cols*2, src.rows*2), src.type());

	for (int y = 0; y < dst.rows ; y++) {
		for (int x = 0; x < dst.cols; x++) {
			float gx = ((float)x) / 2.0;
			float gy = ((float)y) / 2.0;
			int gxi = (int)gx;
			int gyi = (int)gy;
			float c00 = GetPixel(src, gxi, gyi);
			float c10 = GetPixel(src, gxi + 1, gyi);
			float c01 = GetPixel(src, gxi, gyi + 1);
			float c11 = GetPixel(src, gxi + 1, gyi + 1);

			int value = (int)Blerp(c00, c10, c01, c11, gx - gxi, gy -
gyi);
			dst.at<uchar>(y, x) = value;
		}
	}
```

실행결과

OpenCV 함수를 사용한 기본 변환



23

OpenCV 함수의 보간법

- warpAffine (src, dst, M, dsize, flags = INTER_LINEAR)

- INTER_NEAREST = 0, // 최근접 보간법
- INTER_LINEAR = 1, // 양선형 보간법
- INTER_CUBIC = 2, // 3차 보간법
- INTER_AREA = 3,
- INTER_LANCZOS4 = 4,
- INTER_MAX = 7,
- WARP_FILL_OUTLIERS = 8,
- WARP_INVERSE_MAP = 16

24

크기 변환

```
resize (src, dst, dsize, fx = 0, fy = 0, interpolation = cv.INTER_LINEAR )
```

매개 변수	설명
src	입력 영상
dst	출력 영상
dsize	출력 영상 크기. 0이면 다음과 같이 계산된다. dsize=Size(round(fx*src.cols), round(fy*src.rows)
fx	x축 상의 배율. 만약 0이면 (double)dsize. width/src.cols로 계산된다.
fy	y축 상의 배율. 만약 0이면 (double)dsize. height/src.rows로 계산된다.
interpolation	보간법

25

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_COLOR);
	Mat dst = Mat();


	resize(src, dst, Size(), 2.0, 2.0);

	imshow("Image", src);
	imshow("Scaled", dst);
	waitKey(0);
	return 1;
}
```

26

평행이동

```
warpAffine (src, dst, M, dsize, flags = cv.INTER_LINEAR)
```

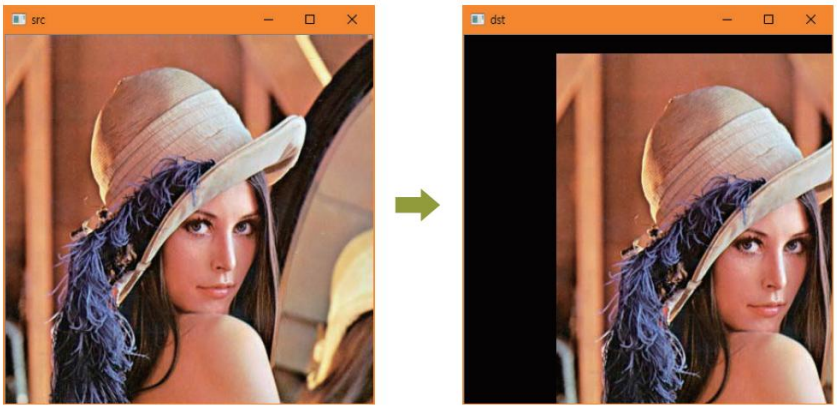
매개 변수	설명
src	입력 영상
dst	출력 영상
Mat	2x3 변환 행렬(CV_64FC1 type)
dsize	출력 영상의 크기
flags	보간법

27

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_COLOR);
	Mat dst = Mat();

	int tx = 100;
	int ty = 20;


	Mat tmat = (Mat_<double>(2, 3) << 1, 0, tx, 0, 1, ty);

	warpAffine(src, dst, tmat, src.size());

	imshow("src", src);
	imshow("dst", dst);
	waitKey(0);
	return 1;
}
```

28

실행결과



29

회전 변환

```
getRotationMatrix2D (center, angle, scale)
```

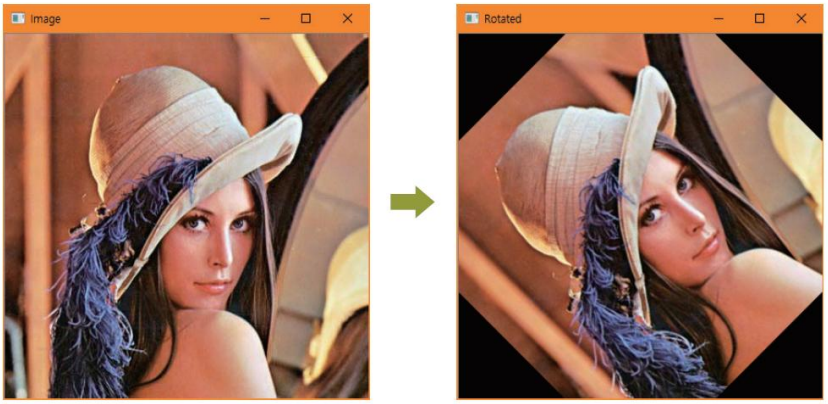
매개 변수	설명
center	입력 영상에서 회전의 중심
angle	회전 각도(단위: 도)
scale	배율

30

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_COLOR);
	Mat dst = Mat();
	Size dsize = Size(src.cols, src.rows);

	Point center = Point(src.cols / 2.0, src.rows / 2.0);
	Mat M = getRotationMatrix2D(center, 45, 1.0);

	warpAffine(src, dst, M, dsize, INTER_LINEAR);

	imshow("Image", src);
	imshow("Rotated", dst);
	waitKey(0);
	return 1;
}
```

31

실행결과



32

3점을 이용한 변환

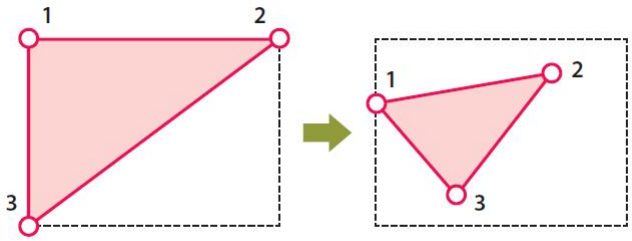


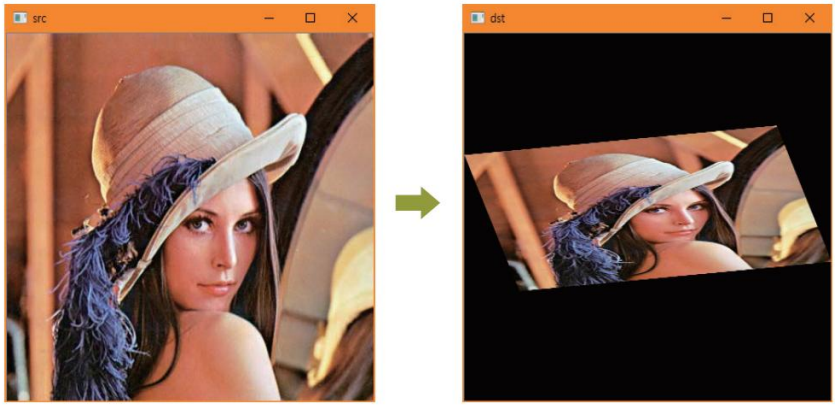
Image 1 Image 2

33

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_COLOR);
	Point2f srcTri[3];
	Point2f dstTri[3];
	Mat warp_mat(2, 3, CV_32FC1);

	Mat warp_dst;
	warp_dst = Mat::zeros(src.rows, src.cols, src.type());
	srcTri[0] = Point2f(0, 0);
	srcTri[1] = Point2f(src.cols - 1.0f, 0);
	srcTri[2] = Point2f(0, src.rows - 1.0f);
	dstTri[0] = Point2f(src.cols*0.0f, src.rows*0.33f);
	dstTri[1] = Point2f(src.cols*0.85f, src.rows*0.25f);
	dstTri[2] = Point2f(src.cols*0.15f, src.rows*0.7f);
	warp_mat = getAffineTransform(srcTri, dstTri);
	warpAffine(src, warp_dst, warp_mat, warp_dst.size());

	imshow("src", src);
	imshow("dst", warp_dst);
	waitKey(0);
	return 1;
}
```

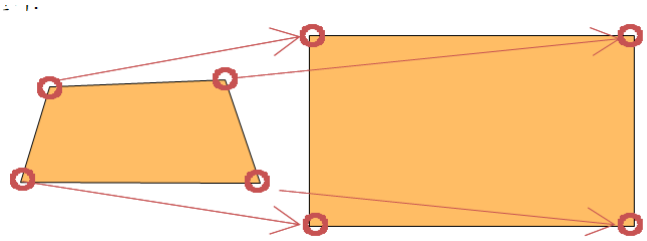
34

실행결과



35

원근 변환



36

원근 변환

```
void warpPerspective(InputArray src, OutputArray dst, InputArray M,
                     Size dsize)
```

매개 변수	설명
src	입력 영상
dst	출력 영상
M	3x3 변환 행렬
dsize	출력 영상의 크기

37

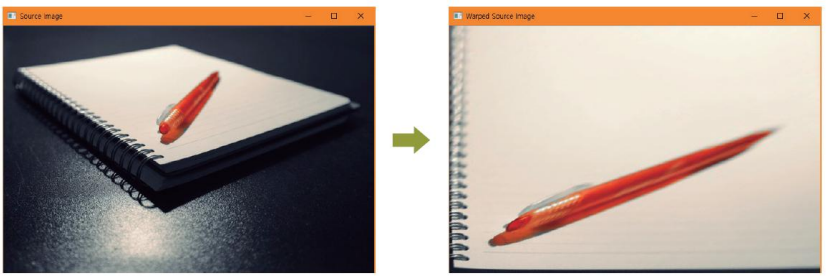
```
int main()
{
        Mat src = imread("d:/book.jpg");

        Point2f inputp[4];
        inputp[0] = Point2f(30, 81);
        inputp[1] = Point2f(274, 247);
        inputp[2] = Point2f(298, 40);
        inputp[3] = Point2f(598, 138);


        Point2f outputp[4];
        outputp[0] = Point2f(0, 0);
        outputp[1] = Point2f(0, src.rows);
        outputp[2] = Point2f(src.cols, 0);
        outputp[3] = Point2f(src.cols, src.rows);
```

38

```
        Mat h = getPerspectiveTransform(inputp, outputp);

        Mat out;
        warpPerspective(src, out, h, src.size());

        imshow("Source Image", src);
        imshow("Warped Source Image", out);

        waitKey(0);
}
```

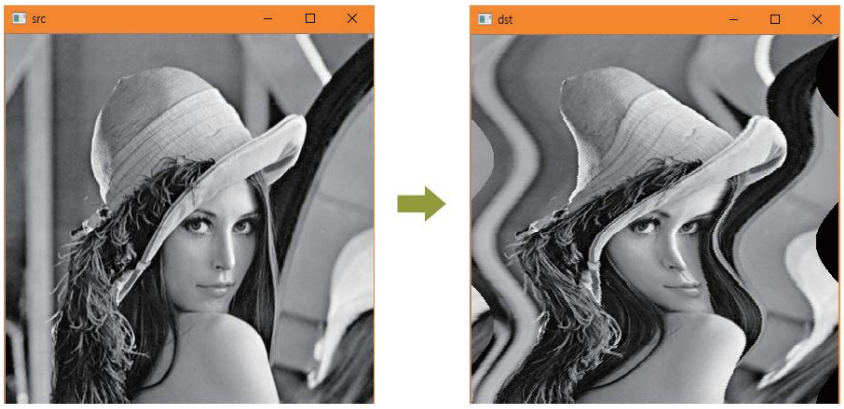
39

실행결과



40

영상 워핑



41

```
int main()
{
	Mat src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
	int rows = src.rows;
	int cols = src.cols;
	Mat dst = src.clone();

	for (int i = 0; i < rows; i++) {
		for (int j = 0; j < cols; j++) {
			int offset_x = (int)(25.0 * sin(2 * 3.14 * i / 180));
			int offset_y = 0;
			if (j + offset_x < rows)
				dst.at<uchar>(i, j) = src.at<uchar>(i, (j +
offset_x) % cols);
			else {
				dst.at<uchar>(i, j) = 0;
			}
		}
	}
```

42

Q & A