1. AFFINE TRANSFORMATION

*import* cv2  
*import* numpy  
  
image = cv2.imread('..\\pictures\\dog\_2.jpg')  
array = numpy.zeros(image.shape, image.dtype)  
  
height, width, channels = image.shape  
  
image1 = image.copy()  
image2 = image.copy()  
image3 = array.copy()  
image4 = image.copy()  
  
pts1 = ([50, 50], [width-21, 20], [25, height-19])  
pts2 = ([157, 188], [width//2, 10], [(width//4)\*3, (height//10)\*9])  
*for* img *in* [image2, array, image3]:  
 *for* i, (x, y) *in* enumerate(pts1):  
 cv2.circle(img, (x, y), 4, (0, 0, 255), -1)  
 *if* i != 1:  
 p, q = x + 10, y + 10  
 *else*:  
 p, q = x - 160, y + 25  
 cv2.putText(img, '({}, {})'.format(x, y), (p, q), cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 0, 255), 3)  
*for* i, (x, y) *in* enumerate(pts2):  
 cv2.circle(image3, (x, y), 4, (0, 255, 0), -1)  
 *if* i != 1:  
 p, q = x + 10, y + 10  
 *else*:  
 p, q = x + 10, y + 30  
 cv2.putText(image3, '({}, {})'.format(x, y), (p, q), cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 255, 0), 3)  
*for* i, j *in* zip(pts1, pts2):  
 cv2.arrowedLine(image3, tuple(i), tuple(j), (255, 0, 0), 2, cv2.LINE\_AA)  
  
M = cv2.getAffineTransform(numpy.float32(pts1), numpy.float32(pts2))  
image4 = cv2.warpAffine(image4, M, (width, height))

cv2.imshow('image1', image1)  
cv2.imshow('array', array)  
cv2.imshow('image2', image2)  
cv2.imshow('image3', image3)  
cv2.imshow('image4', image4)  
cv2.waitKey(0)  
  
cv2.imwrite('..\\Paper1\\affine\\Original\_Image.jpg', image1)  
cv2.imwrite('..\\Paper1\\affine\\Points\_1.jpg', image2)  
cv2.imwrite('..\\Paper1\\affine\\Mask\_Points\_1.jpg', array)  
cv2.imwrite('..\\Paper1\\affine\\New\_points.jpg', image3)  
cv2.imwrite('..\\Paper1\\affine\\Result\_Image.jpg', image4)  
  
cv2.destroyAllWindows()

1. PERSPECTIVE TRANSFORMATION

*import* cv2  
*import* numpy  
  
image = cv2.imread('..\\pictures\\flower\_5.jpg')  
array = numpy.zeros(image.shape, image.dtype)  
  
height, width, channels = image.shape  
  
image1 = image.copy()  
image2 = image.copy()  
image3 = array.copy()  
image4 = image.copy()  
  
pts1 = ([0, 0], [width-1, 0], [0, height-1], [width-1, height-1])  
pts2 = ([157, 188], [width//2, 10], [(width//4)\*3, (height//10)\*9], [700, height-51])  
*for* img *in* [image2, array, image3]:  
 *for* i, (x, y) *in* enumerate(pts1):  
 cv2.circle(img, (x, y), 8, (0, 0, 255), -1)  
 *if* i == 0:  
 p, q = x + 10, y + 25  
 *elif* i == 1:  
 p, q = x - 160, y + 25  
 *elif* i == 2:  
 p, q = x + 10, y - 10  
 *else*:  
 p, q = x - 180, y - 15  
 cv2.putText(img, '({}, {})'.format(x, y), (p, q), cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 0, 255), 3)  
*for* i, (x, y) *in* enumerate(pts2):  
 cv2.circle(image3, (x, y), 4, (0, 255, 0), -1)  
 *if* i == 0:  
 p, q = x + 10, y + 10  
 *elif* i == 1:  
 p, q = x - 160, y + 20  
 *elif* i == 2:  
 p, q = x - 200, y - 10  
 *else*:  
 p, q = x -150, y - 20  
 cv2.putText(image3, '({}, {})'.format(x, y), (p, q), cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 255, 0), 3)  
*for* i, j *in* zip(pts1, pts2):  
 cv2.arrowedLine(image3, tuple(i), tuple(j), (255, 0, 0), 2, cv2.LINE\_AA)  
  
M = cv2.getPerspectiveTransform(numpy.float32(pts1), numpy.float32(pts2))  
image4 = cv2.warpPerspective(image4, M, (width, height))  
  
cv2.imshow('image1', image1)  
cv2.imshow('array', array)  
cv2.imshow('image2', image2)  
cv2.imshow('image3', image3)  
cv2.imshow('image4', image4)  
cv2.waitKey(0)  
  
cv2.imwrite('..\\Paper1\\perspective\\Original\_Image.jpg', image1)  
cv2.imwrite('..\\Paper1\\perspective\\Points\_1.jpg', image2)  
cv2.imwrite('..\\Paper1\\perspective\\Mask\_Points\_1.jpg', array)  
cv2.imwrite('..\\Paper1\\perspective\\New\_points.jpg', image3)  
cv2.imwrite('..\\Paper1\\perspective\\Result\_Image.jpg', image4)  
  
cv2.destroyAllWindows()

1. IMAGE RESIZING

import cv2

image1 = cv2.imread('..\\pictures\\things\_3.jpg')  
image2 = cv2.imread('..\\pictures\\person\_4.jpg')  
  
interpolation\_type2 = ['cv2.INTER\_NEAREST',  
 'cv2.INTER\_LINEAR',  
 'cv2.INTER\_CUBIC',  
 'cv2.INTER\_AREA',  
 'cv2.INTER\_LANCZOS4',  
 'cv2.INTER\_LINEAR\_EXACT']  
  
# for increasing size  
interpolation\_big = [cv2.resize(image1, (800, 800), interpolation=eval(item)) *for* item *in* interpolation\_type2]  
  
cv2.imshow('original image', image1)  
cv2.imwrite('..\\Paper1\\resizing\\Original\_Image\_increase.jpg', image1)  
cv2.waitKey(0)  
*for* count *in* range(6):  
 cv2.imshow(interpolation\_type2[count] + ' ' + str(eval(interpolation\_type2[count])), interpolation\_big[count])  
 cv2.waitKey(0)  
 cv2.imwrite('..\\Paper1\\resizing\\Increase\_' + interpolation\_type2[count] + ' ' + str(eval(interpolation\_type2[count])) + '.jpg', interpolation\_big[count])  
 cv2.destroyWindow(interpolation\_type2[count] + ' ' + str(eval(interpolation\_type2[count])))  
cv2.destroyAllWindows()  
  
# for decreasing size  
interpolation\_small = [cv2.resize(image2, (500,500), interpolation=eval(item)) *for* item *in* interpolation\_type2]  
  
cv2.imshow('original image', image2)  
cv2.imwrite('..\\Paper1\\resizing\\Original\_Image\_decrease.jpg', image2)  
cv2.waitKey(0)  
*for* count *in* range(6):  
 cv2.imshow(interpolation\_type2[count] + ' ' + str(eval(interpolation\_type2[count])), interpolation\_small[count])  
 cv2.waitKey(0)  
 cv2.imwrite('..\\Paper1\\resizing\\Decrease\_' + interpolation\_type2[count] + ' ' + str(eval(interpolation\_type2[count])) + '.jpg', interpolation\_small[count])  
 cv2.destroyWindow(interpolation\_type2[count] + ' ' + str(eval(interpolation\_type2[count])))  
cv2.destroyAllWindows()

1. IMAGE FLIPPING

import cv2

import numpy

image = cv2.imread('..\\pictures\\vehicle\_5.jpg')  
  
# Method 1  
cv2.imshow('original image', image)  
cv2.imshow('Flip Horizontally', cv2.flip(image, 1))  
cv2.imshow('Flip Vertically', cv2.flip(image, 0))  
cv2.imshow('Flip Horizontally and Vertically', cv2.flip(image, -1))  
cv2.waitKey(0)  
cv2.imwrite('..\\Paper1\\flipping\\Original\_Image.jpg', image)  
cv2.imwrite('..\\Paper1\\flipping\\Flip\_Horizontally.jpg', cv2.flip(image, 1))  
cv2.imwrite('..\\Paper1\\flipping\\Flip\_Vertically.jpg', cv2.flip(image, 0))  
cv2.imwrite('..\\Paper1\\flipping\\Flip\_Horizontally\_Vertically.jpg', cv2.flip(image, -1))  
cv2.destroyAllWindows()  
  
# Method 2  
height, width, channels = image.shape  
  
mat\_default = numpy.array(([0, 0], [width - 1, 0], [0, height - 1]), dtype=numpy.float32)  
mat\_1 = numpy.array(([width - 1, 0], [0, 0], [width - 1, height - 1]), dtype=numpy.float32)  
mat\_0 = numpy.array(([0, height - 1], [width - 1, height - 1], [0, 0]), dtype=numpy.float32)  
mat\_minus1 = numpy.array(([width - 1, height - 1], [0, height - 1], [width - 1, 0]), dtype=numpy.float32)  
  
Mat\_1 = cv2.getAffineTransform(mat\_default, mat\_1)  
Mat\_0 = cv2.getAffineTransform(mat\_default, mat\_0)  
Mat\_minus1 = cv2.getAffineTransform(mat\_default, mat\_minus1)  
  
cv2.imshow('original image', image)  
cv2.imshow('Horizontally Using WarpAffine', cv2.warpAffine(image, Mat\_1, (width, height)))  
cv2.imshow('Vertically Using WarpAffine', cv2.warpAffine(image, Mat\_0, (width, height)))  
cv2.imshow('Horizontally and Vertically Using WarpAffine', cv2.warpAffine(image, Mat\_minus1, (width, height)))  
cv2.waitKey(0)  
cv2.destroyAllWindows()  
  
# Method 3  
height, width, channels = image.shape  
  
mat\_default = numpy.array(([0, 0], [width - 1, 0], [0, height - 1], [width - 1, height - 1]), dtype=numpy.float32)  
mat\_1 = numpy.array(([width - 1, 0], [0, 0], [width - 1, height - 1], [0, height - 1]), dtype=numpy.float32)  
mat\_0 = numpy.array(([0, height - 1], [width - 1, height - 1], [0, 0], [width - 1, 0]), dtype=numpy.float32)  
mat\_minus1 = numpy.array(([width - 1, height - 1], [0, height - 1], [width - 1, 0], [0, 0]), dtype=numpy.float32)  
  
Mat\_1 = cv2.getPerspectiveTransform(mat\_default, mat\_1)  
Mat\_0 = cv2.getPerspectiveTransform(mat\_default, mat\_0)  
Mat\_minus1 = cv2.getPerspectiveTransform(mat\_default, mat\_minus1)  
  
cv2.imshow('original image', image)  
cv2.imshow('Horizontally Using WarpPerspective', cv2.warpPerspective(image, Mat\_1, (width, height)))  
cv2.imshow('Vertically Using WarpPerspective', cv2.warpPerspective(image, Mat\_0, (width, height)))  
cv2.imshow('Horizontally and Vertically Using WarpPerspective', cv2.warpPerspective(image, Mat\_minus1, (width, height)))  
cv2.waitKey(0)  
cv2.destroyAllWindows()

1. IMAGE ROTATION

import cv2

import numpy

image = cv2.imread('..\\pictures\\dog\_3.jpg')  
  
cv2.imshow('original image', image)  
cv2.imwrite('..\\Paper1\\rotation\\Original\_Image.jpg', image)  
  
# Method 1  
rotate\_plus90 = cv2.rotate(image, cv2.ROTATE\_90\_CLOCKWISE)  
rotate\_minus90 = cv2.rotate(image, cv2.ROTATE\_90\_COUNTERCLOCKWISE)  
rotate\_180 = cv2.rotate(image, cv2.ROTATE\_180)  
cv2.imshow('ROTATE 90 CLOCKWISE', rotate\_plus90)  
cv2.imshow('ROTATE 90 COUNTERCLOCKWISE', rotate\_minus90)  
cv2.imshow('ROTATE 180', rotate\_180)  
cv2.imwrite('..\\Paper1\\rotation\\ROTATE\_90\_CLOCKWISE.jpg', rotate\_plus90)  
cv2.imwrite('..\\Paper1\\rotation\\ROTATE\_90\_COUNTERCLOCKWISE.jpg', rotate\_minus90)  
cv2.imwrite('..\\Paper1\\rotation\\ROTATE\_180.jpg', rotate\_180)  
cv2.waitKey(0)  
  
# Method 2  
height, width, channels = image.shape  
*for* angle *in* range(20, 360, 50):  
 image1 = image.copy()  
 mat = cv2.getRotationMatrix2D(((width // 2), (height // 2)), angle, 1)  
 rotate = cv2.warpAffine(image1, mat, (width, height))  
 cv2.imshow('Using WarpAffine', rotate)  
 cv2.imwrite('..\\Paper1\\rotation\\Rotation\_Angle\_{}.jpg'.format(angle), rotate)  
 cv2.waitKey(0)  
cv2.circle(image, ((width // 2), (height // 2)), 5, (0, 0, 255), -1, cv2.LINE\_AA)  
cv2.putText(image, 'Center of Rotation ({}, {})'.format((width // 2), (height // 2)), ((width // 2) - 250, (height // 2) + 40), cv2.FONT\_HERSHEY\_PLAIN, 2, (0, 0, 255), 4)  
cv2.imwrite('..\\Paper1\\rotation\\Rotation\_Center.jpg', image)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

1. IMAGE PYRAMIDS

import cv2

image1 = cv2.imread('..\\pictures\\landscape\_1.jpg')  
image2 = cv2.imread('..\\pictures\\landscape\_5.jpg')  
cv2.imwrite('..\\Paper1\\pyramid\\Original\_Image\_Down.jpg', image1)  
cv2.imwrite('..\\Paper1\\pyramid\\Original\_Image\_Up.jpg', image2)  
down = image1  
*for* i *in* range(3):  
 down = cv2.pyrDown(down)  
 cv2.imshow('Image 1 Down ' + str(down.shape), down)  
 cv2.imwrite('..\\Paper1\\pyramid\\Down\_pyramid\_{}\_({}, {}).jpg'.format((i+1), down.shape[1], down.shape[0]), down)  
up = image2  
*for* i *in* range(3):  
 up = cv2.pyrUp(up)  
 cv2.imshow('Image 2 Up ' + str(up.shape), up)  
 cv2.imwrite('..\\Paper1\\pyramid\\Up\_pyramid\_{}\_({}, {}).jpg'.format((i+1), up.shape[1], up.shape[0]), up)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

1. IMAGE TRANSLATION

import cv2

import numpy

image = cv2.imread('..\\pictures\\group\_3.jpg')  
  
height, width, channels = image.shape  
  
# Method 1  
mat\_1 = numpy.array(([1, 0, 500], [0, 1, 50]), dtype=numpy.float32)  
result1 = cv2.warpAffine(image, mat\_1, (width + 500, height + 50))  
cv2.imshow('original image', image)  
cv2.imshow('Translation without Matrix', result1)  
  
# Method 2  
mat\_default = numpy.array(([0, 0], [width - 1, 0], [0, height - 1]), dtype=numpy.float32)  
mat\_1 = numpy.array(([500, 50], [width + 499, 50], [500, height + 49]), dtype=numpy.float32)  
Mat\_1 = cv2.getAffineTransform(mat\_default, mat\_1)  
result2 = cv2.warpAffine(image, Mat\_1, (width + 500, height + 50))  
cv2.imshow('original image', image)  
cv2.imshow('Translation with Matrix', result2)  
  
cv2.imwrite('..\\Paper1\\translation\\Original\_Image.jpg', image)  
cv2.imwrite('..\\Paper1\\translation\\Without\_translation\_matrix.jpg', result1)  
cv2.imwrite('..\\Paper1\\translation\\With\_translation\_matrix.jpg', result2)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()

1. IMAGE GAMMA CORRECTION

import numpy

import cv2

image = cv2.imread('..\\pictures\\landscape\_3.jpg')  
cv2.imwrite('..\\Paper1\\gamma\\Original\_Image.jpg', image)  
Table = numpy.arange(256).astype(numpy.uint8)  
  
Gamma = 0.5  
result = []  
count = 0  
*while* Gamma <= 3.0:  
 Table1 = Table.copy()  
 *for* i *in* range(256):  
 Table1[i] = numpy.clip(pow(i / 255.0, Gamma) \* 255.0, 0, 255)  
  
 result.append(cv2.LUT(image, Table1))  
  
 cv2.imshow("Gamma correction for Gamma = " + str(Gamma), result[count])  
 cv2.imwrite('..\\Paper1\\gamma\\Gamma:{}.jpg'.format(Gamma), result[count])  
 cv2.waitKey(0)  
 Gamma += 0.5  
 count += 1  
  
cv2.destroyAllWindows()

1. IMAGE WARPING

*import* math  
import numpy

import cv2

image = cv2.imread('..\\pictures\\vehicle\_2.jpg')  
height, width, \_ = image.shape  
  
#####################  
# Vertical wave  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = int(25.0 \* math.sin(2 \* 3.14 \* i / 180))  
 offset\_y = 0  
 *if* j+offset\_x < width:  
 image\_output[i, j] = image[i, (j + offset\_x) % width]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Input', image)  
cv2.imshow('Vertical wave', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Vertical\_Wave.jpg', image\_output)  
  
#####################  
# Horizontal wave  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = 0  
 offset\_y = int(16.0 \* math.sin(2 \* 3.14 \* j / 150))  
 *if* i+offset\_y < height:  
 image\_output[i, j] = image[(i + offset\_y) % height, j]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Horizontal wave', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Horizontal\_Wave.jpg', image\_output)  
  
#####################  
# Both horizontal and vertical  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = int(20.0 \* math.sin(2 \* 3.14 \* i / 150))  
 offset\_y = int(20.0 \* math.cos(2 \* 3.14 \* j / 150))  
 *if* i+offset\_y < height *and* j+offset\_x < width:  
 image\_output[i, j] = image[(i + offset\_y) % height, (j + offset\_x) % width]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Multidirectional wave', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Multidirectional\_Wave.jpg', image\_output)  
  
#####################  
# Concave effect  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = int(128.0 \* math.sin(2 \* 3.14 \* i / (2 \* width)))  
 offset\_y = 0  
 *if* j+offset\_x < width:  
 image\_output[i, j] = image[i, (j + offset\_x) % width]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Concave', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Concave\_effect.jpg', image\_output)  
  
# Horizontal Disturbance  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = i + int(20.0 \* math.sin(2 \* 3.14 \* i / 30))  
 offset\_y = 0  
 *if* j < width:  
 image\_output[i, j] = image[int(offset\_x % height), j]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Horizontal Disturbances', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Horizontal\_Disturbances.jpg', image\_output)  
  
# Vertical Disturbance  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = 0  
 offset\_y = j + int(20.0 \* math.sin(2 \* 3.14 \* j / 30))  
 *if* i < height:  
 image\_output[i, j] = image[i, int(offset\_y % width)]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Vertical Disturbances', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Vertical\_Disturbances.jpg', image\_output)  
  
# Both Disturbances  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 offset\_x = i + int(20.0 \* math.sin(2 \* 3.14 \* i / 30))  
 offset\_y = j + int(20.0 \* math.sin(2 \* 3.14 \* j / 30))  
 *if* i < height:  
 image\_output[i, j] = image[int(offset\_x % height), int(offset\_y % width)]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Both Disturbances', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Both\_Disturbances.jpg', image\_output)  
  
# Swirl  
  
image\_output = numpy.zeros(image.shape, dtype=image.dtype)  
  
x0 = width // 2  
y0 = height // 2  
  
*for* i *in* range(height):  
 *for* j *in* range(width):  
 r = (((i - x0) \*\* 2) + ((j - y0) \*\* 2)) \*\* 0.5  
 theta = math.pi \* r / 512  
 offset\_x = (((i - x0) \* math.cos(theta)) + ((j - y0) \* math.sin(theta)) + x0)  
 offset\_y = (((x0 - i) \* math.sin(theta)) + ((j - y0) \* math.cos(theta)) + y0)  
 *if* i < height:  
 image\_output[i, j] = image[int(offset\_x % height), int(offset\_y % width)]  
 *else*:  
 image\_output[i, j] = 0  
  
cv2.imshow('Swirl', image\_output)  
cv2.imwrite('..\\Paper1\\warping\\Swirl.jpg', image\_output)  
  
cv2.imwrite('..\\Paper1\\warping\\Original\_Image.jpg', image)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()