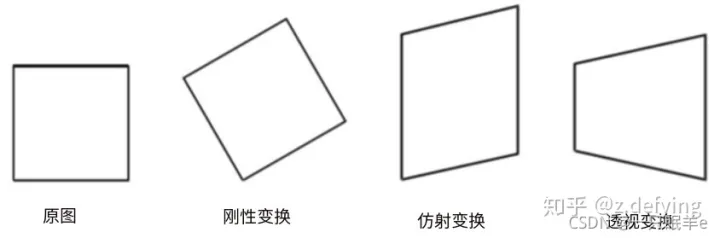
**概述**

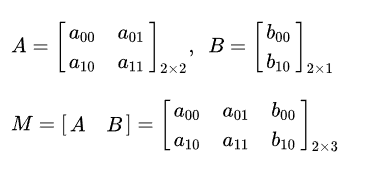
图像的几何变换主要包括：平移、旋转、缩放、剪切、仿射、透视等。  
图像的几何变换主要分为：刚性变换、相似变换、仿射变换和透视变换（投影变换）

* 刚性变换：平移+旋转
* 相似变换：缩放+剪切
* 仿射变换：从一个二维坐标系变换到另一个二维坐标系，属于线性变换。通过已知3对坐标点可以求得变换矩阵
* 透视变换：从一个二维坐标系变换到一个三维坐标系，属于非线性变换。通过已知4对坐标点可以求得变换矩阵。

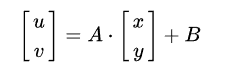


**仿射变换**

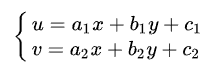
仿射变换（Affine Transformation）是指在向量空间中进行一次线性变换(乘以一个矩阵)和一次平移(加上一个向量)，变换到另一个向量空间的过程。  
仿射变换代表的是两幅图之间的映射关系，仿射变换矩阵为2x3的矩阵，如下图中的矩阵M，其中的B起着平移的作用，而A中的对角线决定缩放，反对角线决定旋转或错切。



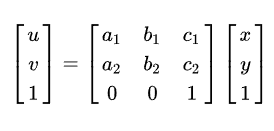
原像素点坐标(x,y)，经过仿射变换后的点的坐标是T，则矩阵仿射变换基本算法原理：



所以仿射变换是一种二维坐标（x,y）到二维坐标（u,v）之间的线性变换，其数学表达式如下：



这个矩阵是2×3的，但是这会改变原始图像的维度，为此，增加一个维度，构造齐次变换矩阵3×3



这就保持了图像的‘平直性’和‘平行性’。  
平直性：直线、圆弧不变  
平行性：平行关系不变，直线相对位置不变，但是夹角可能会改变。

**opencv实现**

仿射变化需要一个转换矩阵，但是由于仿射变换比较复杂，一般很难直接找到这个矩阵，opencv提供了根据源图像和目标图像上三个对应的点来自动创建变换矩阵，矩阵维度为 2x3。  
这个函数是：cv2.getAffineTransform(pos1,pos2)  
最后这个矩阵会被传给函数 cv2.warpAffine()来实现仿射变换。

**import** cv2

**import** numpy **as** np

img **=** cv2**.**imread('me.jpg')

height, width **=** img**.**shape[:2]

*# 在原图像和目标图像上各选择三个点*

mat\_src **=** np**.**float32([[0, 0],[0, height**-**1],[width**-**1, 0]])

mat\_dst **=** np**.**float32([[0, 0],[100, height**-**100],[width**-**100, 100]])

*# 得到变换矩阵*

mat\_trans **=** cv2**.**getAffineTransform(mat\_src, mat\_dst)

*# 进行仿射变换*

dst **=** cv2**.**warpAffine(img, mat\_trans, (width,height))

*# 显示*

imgs **=** np**.**hstack([img,dst])

cv2**.**namedWindow('imgs', cv2**.**WINDOW\_NORMAL)

cv2**.**imshow("imgs",imgs)

cv2**.**waitKey(0)



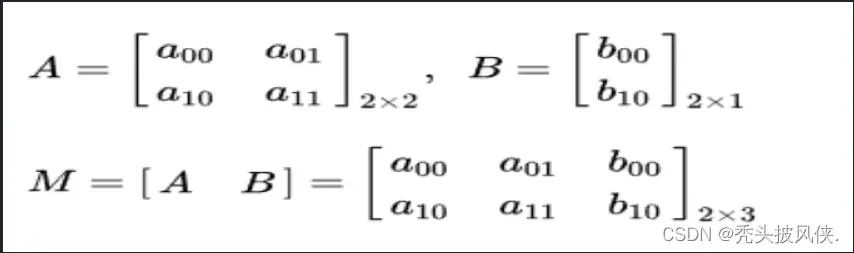
仿射变换原理介绍

仿射变换，也称为仿射映射，是指在几何学中，一个向量空间通过执行线性变换后平移变换到另一个向量空间。

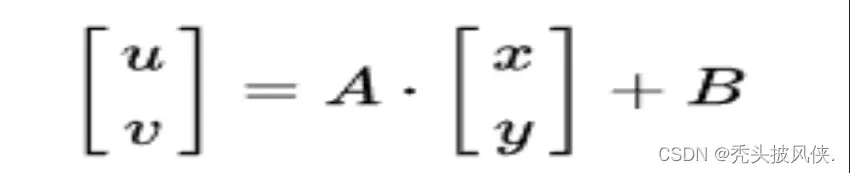
在有限维的情况，每个仿射变换可以由一个矩阵A和一个向量b给出，它可以写作A和一个附加的列b。一个仿射变换对应于一个矩阵和一个向量的乘法，而仿射变换的复合对应于普通的矩阵乘法，只要加入一个额外的行到矩阵的底下，这一行全部是0除了最右边是一个1，而列向量的底下要加上一个1。

在[opencv](https://aitechtogether.com/tag/opencv)中就相当于是对图像的坐标进行操作(x,y)，对坐标进行矩阵运算。运算过后会生成新的矩阵，仿射变换就相当于是原始图像和新生成图像之间的映射，下面给出运算矩阵。

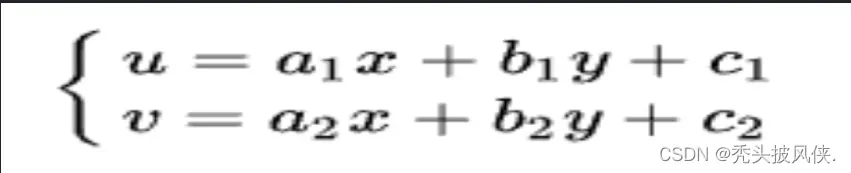
下面的M相当于就是一个运算矩阵。2行3列



根据定义，我们的坐标就可以根据A，B来进行运算

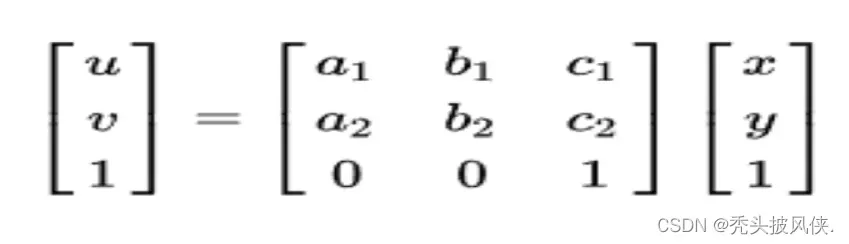


下面的方程可以很容易地推导出来



通过方程我们就可以很轻易的看出，c1和c2就相当于是让图像平移，而a和b这2个参数就是对图像进行旋转，缩放等操作。

由于图像是3维的，所以这里还需要增加下维度，构建齐次方程进行运算



以上仅为个人理解，如有错误请指出

cv2.warpAffine函数介绍

cv2.warpAffine函数用于仿射变换

def warpAffine(src, M, dsize, dst=None, flags=None, borderMode=None, borderValue=None)

Python

Copy

* src：输入图像
* M：运算矩阵，2行3列的，数据类型要求是float32位及以上
* dsize：运算后矩阵的大小，也就是输出图片的尺寸
* dst：输出图像
* flags：插值方法的组合，与resize函数中的插值一样，可以查看 cv2.resize
* borderMode：像素外推方法，详情参考官网
* borderValue：在恒定边框的情况下使用的borderValue值；默认情况下，它是 0

代码示例

下面代码对M矩阵给出不同的值，查看图像变换情况，首先改变c1和c2

M = np.float32([[1, 0, 50], [0, 1, 50]])

Python

Copy

上面的代码就相当于是对图像进行平移，向左平移50，向上平移50



改变下其他值，改变b1和a2

M = np.float32([[1, 0.2, 0], [0.2, 1, 0]])

Python

Copy

这个就好像是对图片进行了3维的旋转



改变b2和a1

M = np.float32([[1.3, 0, 0], [0, 1.3, 0]])

Python

Copy

这相当于放大图像。



完整代码

import cv2

import numpy as np

lp = cv2.resize(cv2.imread('../images/lp.jpg'), None, fx=0.7, fy=0.7)

h, w, channel = lp.shape

M = np.float32([[1, 0, 50], [0, 1, 50]])

new\_lp1 = cv2.warpAffine(lp, M, (w, h))

M = np.float32([[1, 0.2, 0], [0.2, 1, 0]])

new\_lp2 = cv2.warpAffine(lp, M, (w, h))

M = np.float32([[1.3, 0, 0], [0, 1.3, 0]])

new\_lp3 = cv2.warpAffine(lp, M, (w, h))

cv2.imshow('lp', np.hstack((lp, new\_lp1, new\_lp2, new\_lp3)))

cv2.waitKey(0)

cv2.destroyAllWindows()

