仿射變換 Affine Transformation

cupoy.com/event/cvdl/mission/1586226055069

1.仿射變換｜線代示錄

透過數學證明仿射變換為甚麼具有比例不變性,

以及依序推導仿射變換矩陣的設計

仿射變換 | 線代示錄

本文的閱讀等級：初級 設 $latex A&fg=000000$ 為一個 $latex n\times n&fg=000000$ 階實矩陣，$latex \mathbf{b}&fg=000000$ 是 $latex n&fg=000000$ 維實向量，定義於幾何向量空間 $latex \mathbb{R}^n&fg=000000$ 的仿射變換 (aff...

ccjou.wordpress.com

2.幾和變換矩陣設計｜線代示錄

裏面包含其他線性變換的證明與講解,

其中切變變換 (Shearing) 屬於 optional 的內容

幾何變換矩陣的設計 | 線代示錄

本文的閱讀等級：初級 矩陣之所以成為研究線性變換的一個有效工具乃基於兩個事實：線性變換完全由基底的映射行為所決定，以及線性複合變換可表示為矩陣乘積 (見“線性代數的第一堂課——矩陣乘法的定義”)。本文運用這兩個性質來設計二維歐幾里得空間裡常用的一些幾何變換矩陣。...

ccjou.wordpress.com

Geometric Transformation - Affine Transformation

如果對 OpenCV 實作不清楚可以參考官方實作程式碼

Geometric Transformations of Images — OpenCV 3.0.0-dev documentation

undefined

docs.opencv.org

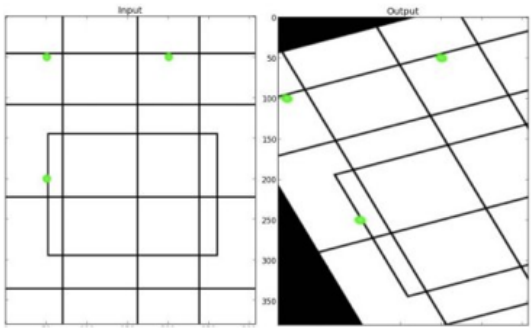
Affine Transformation

In affine transformation, all parallel lines in the original image will still be parallel in the output image. To find the transformation matrix, we need three points from input image and their corresponding locations in output image. Then **cv2.getAffineTransform** will create a 2x3 matrix which is to be passed to **cv2.warpAffine**.

Check below example, and also look at the points I selected (which are marked in Green color):

```
img = cv2.imread('drawing.png')
rows,cols,ch = img.shape

pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]])

M = cv2.getAffineTransform(pts1,pts2)

dst = cv2.warpAffine(img,M,(cols,rows))

plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

See the result:



Perspective Transformation

For perspective transformation, you need a 3x3 transformation matrix. Straight lines will remain straight even after the transformation. To find this transformation matrix, you need 4 points on the input image and corresponding points on the output image. Among these 4 points, 3 of them should not be collinear. Then transformation matrix can be found by the function **cv2.getPerspectiveTransform**. Then apply **cv2.warpPerspective** with this 3x3 transformation matrix.

OpenCV

Quick search

Go

Table Of Contents

Geometric Transformations of Images
» Goals
» Transformations
» Scaling
» Translation
» Rotation
» Affine Transformation
» Perspective Transformation
» Additional Resources
» Exercises

Previous topic

Image Thresholding

Next topic

Smoothing Images