

影像處理 Image Processing

Homework 2 Geometric Transformation

學號: _____10820109_____

姓名: _____陳詩云_____

指導教授: 張元翔

I. Introduction

本作業的目標是研究和實作影像處理技術，學習將幾何轉換運用於影像。

II. Method

使用 OpenCV 實現魚眼效果和圖像合成。

III. Results

Part I

經過魚眼特效的區域可以看成一個圓形區域，圓形區域由原始影像中心的圓形區域擴張而成。



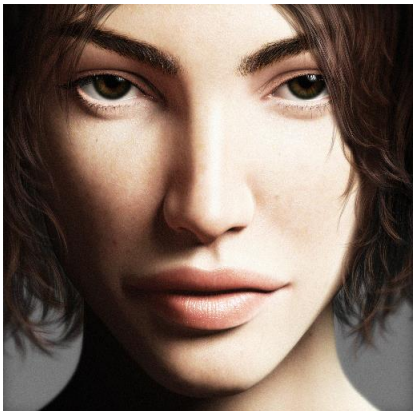
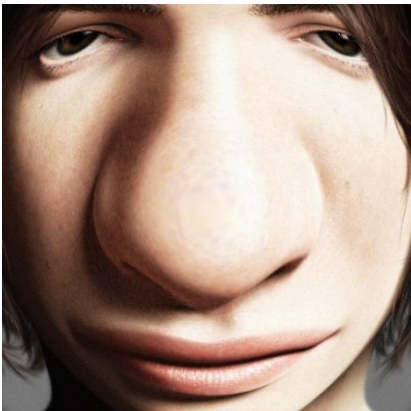
Original Image	Fisheye Effect
	
	



Fig. 1 Examples of the *fish-eye effect*.







Original Image	Fisheye Effect
	
	
	

Fig. 2 Examples of the *fish-eye effect* (acquired from Internet).

Part II

影像藉由透視轉換將座標位置映設到另一張影像的新座標位置。



Fig. 2 Examples of the *image composition*.

IV. Discussion

- (1) Part1: 從 Part1 可觀察到極座標的 r 越大、離幾何失真的中心越遠，影像位置的變動越大。
- (2) Part2: 原始影像可透過改變像素的空間位置，計算新空間位置的像素值，因此可以任意的將原始圖片合成在另一張影像，透過幾何轉換也可以將傾斜或變形的圖片矯正回來。

VI. Appendix

Python 程式

```
# 影像處理
# Homework 2 – part1
# 學號: ____10820109____
# 姓名: ____陳詩云____

import numpy as np
import cv2
import math
import numpy as np
import cv2
import math

img = cv2.imread( "Bug.bmp", -1)

def fisheyes( img_suc ) :
    img = img_suc.copy()
    m, n, c = img.shape
    xc = m / 2
    yc = n / 2
    # 到 4 個點最遠距離
    R = math.sqrt( xc**2 + yc**2 )
    x_prime = np.zeros( (m,n), dtype= "float32")
    y_prime = np.zeros( (m,n), dtype= "float32")

    for x in range( m ) :
        for y in range( n ) :
            r = (x - xc) ** 2 + (y - yc) ** 2
            sita = math.atan2(y - yc, x - xc)
            x_prime[x,y] = (r/R)*math.cos(sita) + xc
            y_prime[x,y] = (r/R)*math.sin(sita) + yc
    img_out = cv2.remap( img, y_prime, x_prime, cv2.INTER_LINEAR)
    return img_out

new_img = fisheyes( img )
cv2.imwrite( "fisheye_bug.jpg", new_img )
cv2.imshow('fish',new_img)
cv2.waitKey( 0 )
cv2.destroyAllWindows()
```

Python 程式

```
# 影像處理
# Homework 2 – part2
# 學號: _____10820109_____
# 姓名: _____陳詩云_____

import numpy as np
import cv2

def composition(img1,img2): #1: Background  2:iris
    pts1 = np.float32([[47, 191], [307, 219], [47, 440], [308, 407]])
    #pts1 = np.float32([[360, 94], [738, 46], [363, 291], [735, 306]]) back2
    #pts1 = np.float32([[360, 240], [498, 235], [358, 333], [498, 330]]) back3
    pts2 = np.float32([[0, 0], [300, 0], [0, 250], [300, 250]])

    M = cv2.getPerspectiveTransform(pts1, pts2)
    M_inv = np.linalg.inv(M)

    dst = cv2.warpPerspective(img1, M, (300, 250))
    iris2 = cv2.resize(img2, (300, 250))
    dst2 = cv2.warpPerspective(iris2, M_inv, (800, 560))

    copyImg = dst2.copy()
    h, w = copyImg.shape[:2]

    mask = np.ones([h+2,w+2,1],np.uint8)
    mask[h+1:w+1,h+1:w+1] = 0
    cv2.floodFill(copyImg, mask, (30, 30), (255, 255, 255), (100, 100, 100), (50, 50, 50),
cv2.FLOODFILL_FIXED_RANGE)

    img2gray = cv2.cvtColor(copyImg, cv2.COLOR_BGR2GRAY)
    ret, mask = cv2.threshold(img2gray, 0.5, 255, cv2.THRESH_BINARY_INV)
    rows, cols, channels = copyImg.shape
    y = 0
    x = 0
    roi = img1[y:0 + rows, x:0 + cols]
    img1_bg = cv2.bitwise_and(roi, roi, mask = mask)

    mask_inv = cv2.bitwise_not(mask)
    img2_fg = cv2.bitwise_and(copyImg, copyImg, mask=mask_inv)
    dst = cv2.add(img1_bg, img2_fg)
    img1[y:y + rows, x:x + cols] = dst
    return img1
```

```
img1 = cv2.imread('Background1.bmp')
img2 = cv2.imread('iris.jpg')
new_img = composition(img1,img2)
cv2.imshow('Background1.jpg', new_img)
cv2.waitKey(0)
cv2.imwrite( 'Background1.jpg', new_img)
cv2.destroyAllWindows()
```