RBF 图像变形实验报告

陈萱 SA24001008

2024 年 9 月 22 日

1 问题描述

给定 n 个源点与目标点, 通过 RBF 基构造出使得图像平滑变形的函数.

2 程序思路说明

以径向基函数为基 $g_i(x) = e^{-\frac{|x-p_i|^2}{\sigma^2}}$, 其中半径 σ 与所变形图像的大小有关, 将要求的函数用改组基线性表示出来, 问题转化为求解函数在这组基下的系数, 由于单纯用 RBF 基不能表示出多项式形式, 故我们考虑在函数中加入低阶多项式, 同时加入限制条件, 如下图矩阵表达式所示, 可以理解为某种意义下先用最小二乘拟合出多项式。

$$f(x) = ax + by + c + \sum_{i=1}^{n} b_i g_i(x)$$

$$\begin{pmatrix} g_1(p_1) & g_2(p_1) & \cdots & g_n(p_1) & y_1 & x_1 & 1 \\ g_1(p_2) & g_2(p_2) & \cdots & g_n(p_2) & y_2 & x_2 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ g_1(p_n) & g_2(p_n) & \cdots & g_n(p_n) & y_n & x_n & 1 \\ y_1 & y_2 & \cdots & y_n & 0 & 0 & 0 \\ x_1 & x_2 & \cdots & x_n & 0 & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ b_{n+1} \\ b_{n+2} \\ b_{n+3} \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3 主要代码

注意由于 python 函数中 remap 的特性, 故我们的计算是逆流运算, 即计算从目标点到源点的变换函数

```
h, w = warped_image.shape[:2]
n=target_pts.shape[0]
X=np.zeros((h,w))
Y=np.zeros((h,w))
sigma = ((h+w)/12*0.2)**2
Lambda=0

M=np.ones((n+3,n+3))
I=np.eye(n)
f=np.array([[np.sum((target_pts[i]-target_pts[j])
    **2)for i in range(n)]for j in range(n)])
f=np.exp(-f/sigma)
#计算变换矩阵M
M[:n,:n]=f-Lambda*I
M[:n,n:n+2]=target_pts
M[n:n+2,:n]=target_pts.T
M[n:n+3,n:n+3]=np.array([0,0,0])
b=np.vstack((source_pts,np.array
    ([[0,0],[0,0],[0,0]])))
#求解线性方程组
x, res, rank, s = np.linalg.lstsq(M, b, rcond=None
    )
#计算每个像素移动到的位置（逆流）
for i in range(h):
    for j in range(w):
        v=np.array([i,j])
        basis=[np.exp(-np.sum((v-target_pts[k])
            **2)/sigma)for k in range(n)]
        basis=basis+[j,i,1]
```

```
            xy=basis @ x
            Y[i][j]=int(xy[0])
            X[i][j]=int(xy[1])
    warped_image=cv2.remap(warped_image,np.float32(Y),
        np.float32(X),interpolation=cv2.INTER_LINEAR)
```

4 结果展示