Image Warping

陈雨竹 PB19000160

2022 年 3 月 5 日

1 问题描述

在 MATLAB 上实现基于径向基函数插值方法的图像拖拽变形，并实现实时用户交互，得到类似图 1的界面。

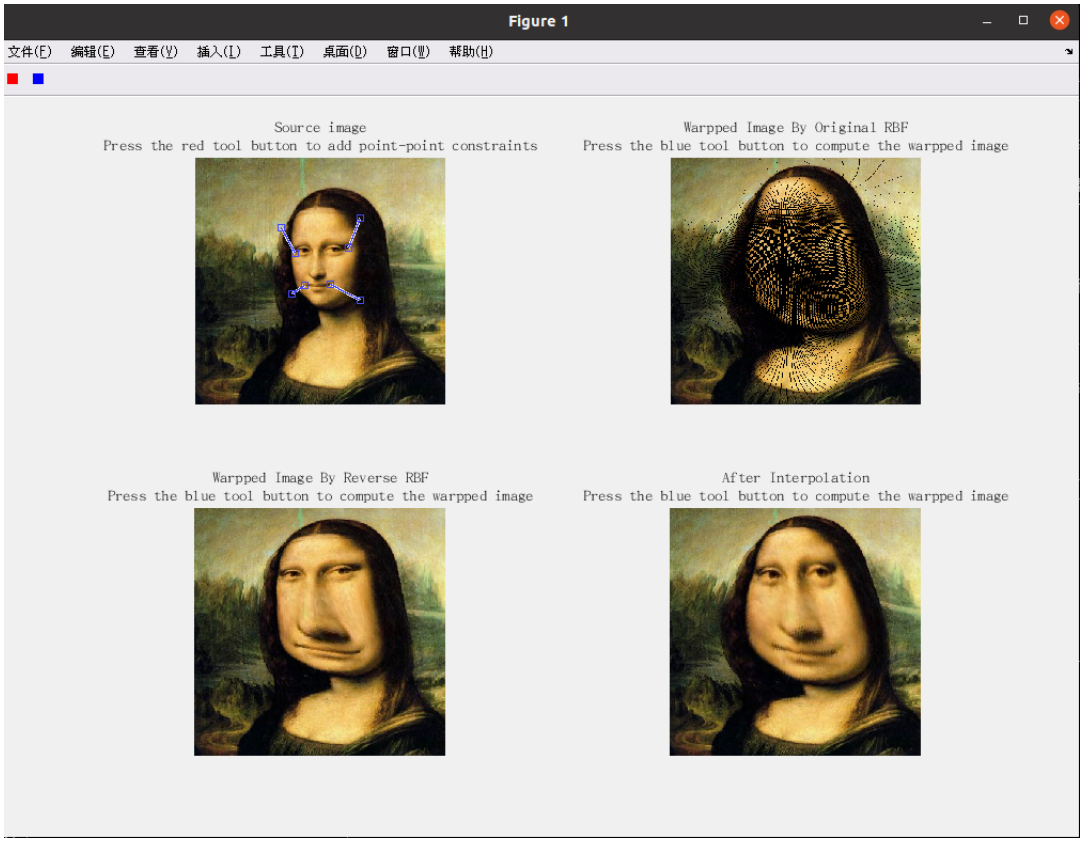

图 1: Image Warping Result

2 实现方法

编程环境为 Ubuntu 20.04.3 LTS MATLAB R2021a (9.10.0.1602886)。

本文使用两种基于径向基函数的方法实现 Image Warping。其核心问题为对于给定初始点 (psrc):$\{x_i\}_{i=1}^n$ 以及对应的目标点 (pdst):$\{y_i\}_{i=1}^n$ 构造函数 f 使得 $f(x_i) = y_i$ 对 $i = 1, 2, \cdots, n$ 成立。在已有 f 时，只需令源图像 $\mathbf{x}$ 处的像素赋给目标图像 $\mathbf{f(x)}$ 处即可。

2.1 RBF Image Warping

考虑径向基函数

$$b_i(\mathbf{x}) = \frac{1}{|\mathbf{x} - \mathbf{p_i}|^2 + d}$$

从而构造

$$f(\mathbf{x}) = \mathbf{x} + \sum_{i=1}^{n} b_i(\mathbf{x})\mathbf{a_i}$$

其中系数 $\mathbf{a_i}$ 可以由 $f(\mathbf{x_i}) = \mathbf{y_i}$ 构造如下线性方程组进行求解。

$$(1./((|\mathbf{x_i} - \mathbf{x_j}|^2)_{i,j=1}^n + d))(\mathbf{a_i})_{i=1}^n = (\mathbf{y_i})_{i=1}^n$$

在已有这样的 f 时，将源图像所有整点 $\mathbf{x}$ 的像素值赋给目标图像 $f(\mathbf{x})$ 取整处的像素值即可。

但这样做会导致 $f(\mathbf{x})$ 在 $\mathbf{x}$ 为整点时并不能覆盖所有的整点，呈现在图像上就是得到的图像有缝隙。

为解决这样的问题，需要对缝隙中的点进行插值。这里使用平均值插值的方法，具体实现时在对目标图像着色时记录被着色的像素点，对于未着色的点，若认定为图像的边界则着色为黑色 $(R, G, B) = (0, 0, 0)$，若不是图像边界则取其像素值为周边八个点中已着色点的均值。

2.2 Reverse RBF Image Warping

解决上述缝隙的另一种方法是用假设有一个处理好的图像 dst，将 dst 按照指定拖拽方法的逆过程进行得到源图像。

此时需要解的方程为 $f(\mathbf{y_i}) = \mathbf{x_i}$，对应的线性方程组为如下方程组。

$$(1./((|\mathbf{y_i} - \mathbf{y_j}|^2)_{i,j=1}^n + d))(\mathbf{a_i})_{i=1}^n = (\mathbf{x_i})_{i=1}^n$$

在已有这样的 f 时，将目标图像所有整点 $\mathbf{x}$ 的像素值赋为源图像 $f(\mathbf{x})$ 取整处的像素值即可。由于目标图像的每个整点都会被遍历，得到的计算结果并不会出现缝隙。这样做可能会出现算出的 $f(\mathbf{x})$ 越界的问题，对于这种情况，本文的处理方法是令其像素值为黑色 $(R, G, B) = (0, 0, 0)$。

3 实验结果

在具体实现中，取 d 为所有拖动距离的平方和，得到的结果如图 1。

在该结果中可以看出，RBF Image Warping 方法得到的图像可能出现大量缝隙，但在插值之后仍然可以得到伸缩的结果图像。Reverse RBF Image Warping 方法确实不会出现缝隙，而且可以得到合理的拉伸图像。这两种方法得到的拉伸结果图像有所不同，但都属于合理的拉伸结果。

此外，为了对比两种方法在放大/缩小时对于边界的处理情况，本文对这两种方法在网格图下进行测试，结果如图 2。

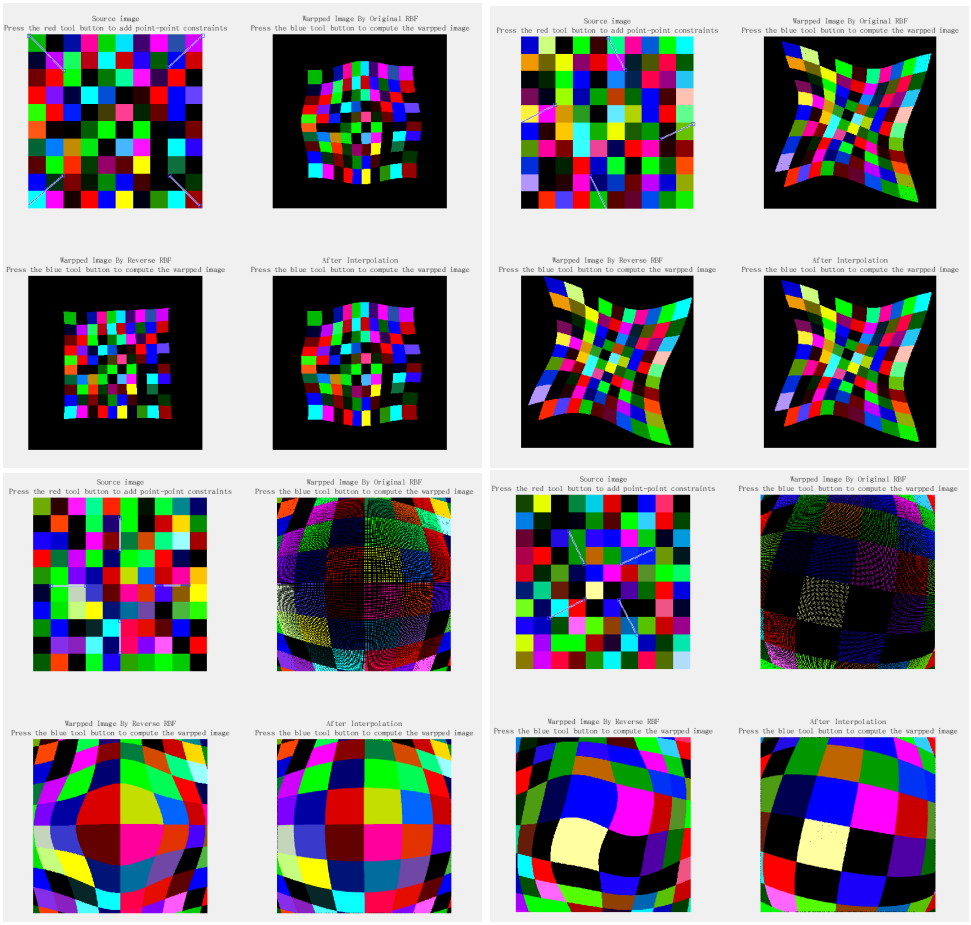

图 2: 方格图测试

可以看出，在缩小时两种方法都得到了合理的结果，且原 RBF 方法几乎不需要进行插值操作。但在放大时，原 RBF 方法会导致出现大量缝隙，从而需要对大量点进行差值处理。由于方格图比较均匀，插值效果较好，但如果放大比例更大且图像规律性更小，原 RBF 方法可能出现插值像素不正确或无法判定是否为边界的情况。

4 参考文献

[1]Nur Arad and Daniel Reisfeld. Image Warping Using Few Anchor Points and Radial Functions. Computer Graphics Forum, 14(1): 35-46, 1995.