图像变形实验报告

姓名：张钧　　日期：2020 年 9 月 18 日

摘要：本实验用$MATLAB$实现了图像变形算法中的RBF与IDW算法，本报告对两种算法做了相关的简单介绍与对应的比较分析。

一、 算法描述

图像变形，实际即为数字图像的几何变形过程。图像变形的步骤可以归为：第一步计算原图像中所有像素所需的位移（亦为本实验着重研究的问题），第二步则为对图像重新采样以创建输出图像。

给定一组离散平移向量的图像变形问题的本质即为同时插值两个函数的问题，其一为x轴方向上的位移处理，另一则为y轴方向上的位移处理。考虑输入数据为n组控制点对$(\boldsymbol{p_i}, \boldsymbol{q_i})$，我们需要找到对应的连续映射$f_1$和$f_2$满足：$f_1: R^2 \to R^2$，$f_2: R^2 \to R^2$；$f_1(\boldsymbol{p_i}) = q_{i1}$，$f_2(\boldsymbol{p_i}) = q_{i2}$，其中$q_{ij}$为$\boldsymbol{q_i}$的第$j$分量，$j = 1,2$。

1. IDW算法

在IDW算法中，考虑局部近似：$f_i(\boldsymbol{p_i}) = y_i$，将原图像中点$\boldsymbol{p}$映射到$f(\boldsymbol{p}) = \sum_{i=1}^{n} \omega_i(\boldsymbol{p}) f_i(\boldsymbol{p})$，其中权重$\omega_i(\boldsymbol{p_i}) = 1$，$\sum_{i=1}^{n} \omega_i(\boldsymbol{p}) = 1$，$\omega_i(\boldsymbol{p}) \geq 0$。一般地，取$\omega_i(\boldsymbol{p}) = \frac{\sigma_i(\boldsymbol{p})}{\sum_{j=1}^{n} \sigma_j(\boldsymbol{p})}$，其中$\sigma_i(\boldsymbol{p}) = \frac{1}{d(\boldsymbol{p},\boldsymbol{p_i})^{\mu}}$，其中$d(\boldsymbol{p}, \boldsymbol{p_i})$为$\boldsymbol{p}$与$\boldsymbol{p_i}$距离。

进一步地，可以修正$\sigma_i(\boldsymbol{p}) = [\frac{(R_i - d(\boldsymbol{p},\boldsymbol{p_i}))_+}{R_i d(\boldsymbol{p},\boldsymbol{p_i})}]^{\mu}$。其中$R_i$为用户指定的影响范围，可由多次试验得到效果较好数据；$x_+ = \max\{x, 0\}$。通常取$\mu = 2$，局部逼近通常用一阶或二阶多项式，系数由导数值得到。其误差函数为：$E_i(f) = \sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j})(f(\boldsymbol{p_j}) - y_j)^2$。即$\boldsymbol{f}(\boldsymbol{p}) = \sum_{i=1}^{n} \omega_i(\boldsymbol{p}) \boldsymbol{f_i}(\boldsymbol{p})$，其中$\boldsymbol{f_i}(\boldsymbol{p_i}) = \boldsymbol{q_i}$，$\boldsymbol{f_i}(\boldsymbol{p}) = \boldsymbol{q_i} + \boldsymbol{T_i}(\boldsymbol{p} - \boldsymbol{p_i})$。考虑对应误差函数$E_i(\boldsymbol{T}) = \sum_{j=1, j\neq i}^{n} \sigma_i(\boldsymbol{p_j}) || \boldsymbol{q_i} + \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix} (\boldsymbol{p_j} - \boldsymbol{p_i}) - \boldsymbol{q_j} ||^2$，分别对$\boldsymbol{T}$内元素求偏导，令偏导为 0 即可得到使得误差最小化的具体系数矩阵$\boldsymbol{T}$。

对于任意的IDW算法，由于对于每个像素均单独运算其映射，故总算法复杂度均为$O(nN)$，

其中n为控制点对数量，N为像素数量。

2. RBF算法

在RBF算法中，将原图像中点$\boldsymbol{p}$映射到$f(\boldsymbol{p})=\sum_{i=1}^{n}\alpha_i f_i\big(d(\boldsymbol{p},\boldsymbol{p_i})\big)+\boldsymbol{p_m}(\boldsymbol{p})$，其中$f_i$称为径向函数，$\boldsymbol{p_m}$为$m$次多项式。常用的径向函数有：$f(d)=(d^2+r^2)^{\mu}$，其中$r_i=min_{i\neq j}d_i(\boldsymbol{p_j})$，$\mu$通常取$\pm\frac{1}{2}$，当取$-1$时算法速度较快。

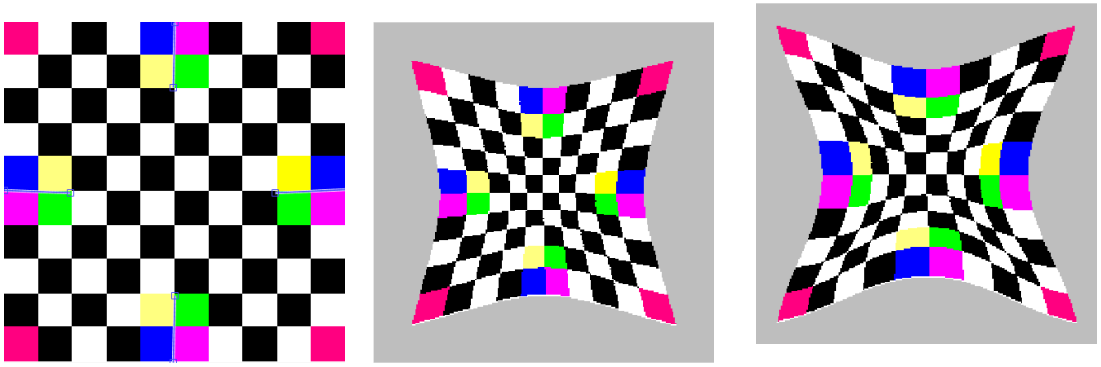
一般地，我们取$\boldsymbol{p_m}$为线性多项式$\boldsymbol{p}$，即：$f(\boldsymbol{p})=\sum_{i=1}^{n}\alpha_i(d(\boldsymbol{p},\boldsymbol{p_i})^2+{r_i}^2)^{\mu}+\boldsymbol{p}$，再由映射将$\boldsymbol{p_i}$映射至$\boldsymbol{q_i}$的初始条件可以反解出系数矩阵$\{\alpha_i\}$，代入记得所需映射。

对于任意的RBF算法，由于亦是对于每个像素的单独运算，故总算法复杂度亦为$O(nN)$。

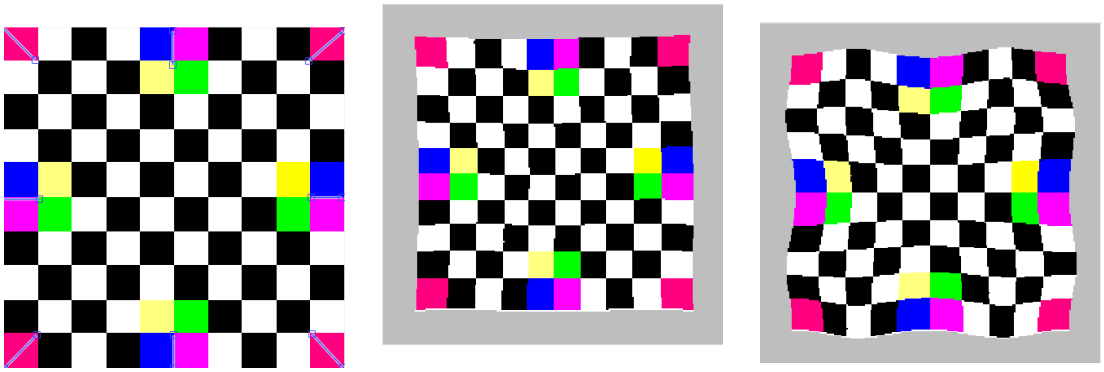
二、 实验结果

如下所示即为对原图作不同情形的控制点对选取下的RBF与IDW两种算法所得到的变形效果图（左至右依次为原图带控制点对，RBF处理图，IDW处理图）：

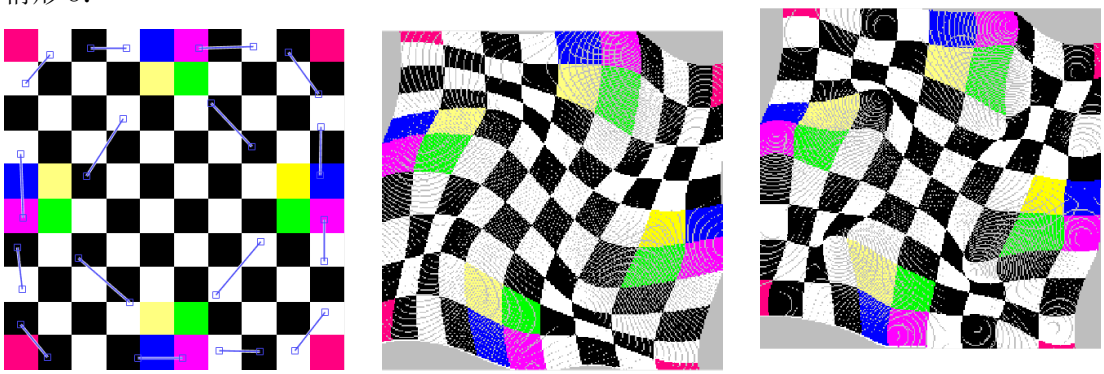
情形 1：



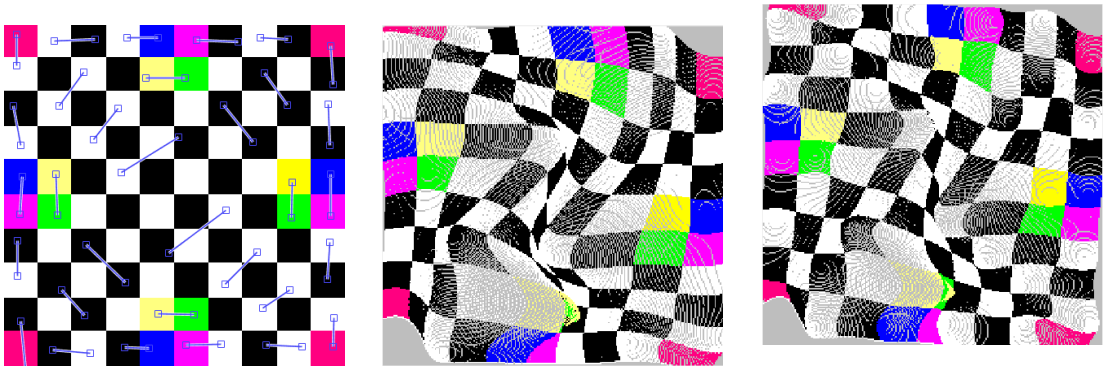
情形 2：



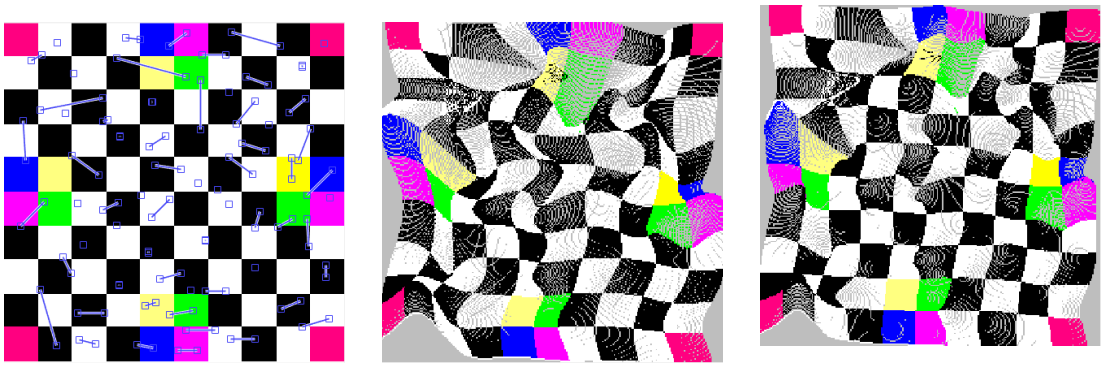
情形 3：



情形 4：



情形 5：



以上各情形对应消耗时间表：

情形	n	N	RBF耗时(s)	IDW耗时(s)
情形1	4	65536	0.221864	0.292928
情形2	8	65536	0.305643	0.323484
情形3	16	65536	0.465335	0.417470
情形4	32	65536	0.851313	0.624041
情形5	64	65536	1.486346	1.009167

三、 结果分析

1. 算法复杂性与结果的比较与分析

由第一部分算法所述，理论上两种算法复杂度均为$O(nN)$；而实际上，由第二部分实验结果所附时间表可知：当 n 逐渐增大时，无论是RBF还是IDW方法，在控制像素点数N不变时，算法耗时随控制点对数n的增大而增加，且逐渐接近正比关系，故可以大致认为实际复杂度为$O(nN)$。

2. 背景空洞或条纹出现的原因

观察到无论是IDW还是RBF算法，所得图像中总是充斥着大量的灰色空洞或条纹（此实验中所选背景色为灰色），这是由于无论是IDW还是RBF算法，均不能保证相邻的格子均被原图像

中的某一点映射的取整所得（此实验中在$MATLAB$代码中使用$round$函数作为取整函数四舍五入，可能造成此现象更为频繁），故所得图形不具有广义上的“连续性”，从而导致未被映射且取整部分呈现背景色，夹杂在被映射且取整部分中，造成灰色空洞或条纹现象。