Assignment-matlab 1

Haowei Kan

September 17, 2020

1 Introduction

The term "image warping" describes methods for deforming images to arbitrary shapes. The problem of image deformation can be formulated as follows:

Input: n pairs $(\mathbf{p}_i, \mathbf{q}_i)$ of control points, $\mathbf{p}_i, \mathbf{q}_i \in \mathbb{R}^2, i = 1, ..., n$.

Output: An at least continuous function $\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2$ with $\mathbf{f}(\mathbf{p}_i) = \mathbf{q}_i, i = 1, ..., n$.

The interpolation problem of scattered data of two variables can be formulated as:

Input: n data points $(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^2, y_i \in \mathbb{R}, i = 1, ..., n$

Output: An at least continuous function $f : \mathbb{R}^2 \to \mathbb{R}$ interpolating the given data points, i.e.$f(\mathbf{x}_i) = y_i, i = 1, ..., n$.

So we can solve the image deformation problem by treating each component of target points separately and then the interpolation can be used for it. Here we will introduce two image warping alogrithms based on this idea.

2 Algorithms

2.1 Inverse Distance Weighted Interpolation Methods

For each data point $\mathbf{p}_i$, a local approximation $f_i(\mathbf{p}) : \mathbb{R}^2 \to \mathbb{R}$ with $f_i(\mathbf{p}_i) = y_i, i = 1, ..., n$ is determined. The interpolation function is a weighted average of these local approximations, with weights dependent on the distance of the observed point from the given data points,

$$f(\mathbf{p}) = \sum_{i=1}^{n} w_i(\mathbf{p}) f_i(\mathbf{p}) \tag{1}$$

where $f_i(\mathbf{p}_i) = y_i, i = 1, ..., n$. $w_i : \mathbb{R}^2 \to \mathbb{R}$ is the weight function, which must satisfy the conditions

$$w_i(\mathbf{p}_i) = 1, \quad \sum_{i=1}^{n} w_i(\mathbf{p}) = 1, \quad \text{and } w_i(\mathbf{p}) \geq 0, i = 1, ..., n \tag{2}$$

The application of inverse distance-weighted interpolation to image warping gives

$$\mathbf{f}(\mathbf{p}) = \sum_{i=1}^{n} w_i(\mathbf{p}) \mathbf{f}_i(\mathbf{p}) \tag{3}$$

where $\mathbf{f}(\mathbf{p}_i) = \mathbf{q}_i, i = 1, ..., n$. $\mathbf{f}_i : \mathbb{R}^2 \to \mathbb{R}^2$ are the local approximations.

weight function

A simple weight function is proposed by Shepard, which is

$$w_i(\mathbf{p}) = \frac{\sigma_i(\mathbf{p})}{\sum\limits_{j=1}^{n} \sigma_j(\mathbf{p})} \text{ with } \sigma_i(\mathbf{p}) = \frac{1}{d(\mathbf{p}, \mathbf{p}_i)^{\mu}} \tag{4}$$

where $d(\mathbf{p}, \mathbf{p}_i)$ is the distance between $\mathbf{p}$ and $\mathbf{p}_i$.

The smoothness is determined by the exponent μ. $\mu > 1$ assures continuity of the derivatives.

local approximation

For the local approximations, linear or quadratic polynomials are normally used. We used the linear polynomials in our experiments, i.e.

$$\mathbf{f}_i(\mathbf{p}) = \mathbf{q}_i + \mathbf{T}_i(\mathbf{p} - \mathbf{p}_i) \tag{5}$$

We compute T by minimizing the squared error of the mapping of other nearby control points $\mathbf{p}_j$ with f_i, weighted with the $\sigma_i(\mathbf{p}_j)$ from Equation (4). The corresponding error function $E_i(f)$ is

$$E_i(T) = \sum_{j=1, j\neq i}^{n} \sigma_i(\mathbf{p}_j) \left\| \mathbf{q}_i + \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix} (\mathbf{p}_j - \mathbf{p}_i) - \mathbf{q}_j \right\|^2 \tag{6}$$

It can be also written in the matrix form, which is

$$E_i(T) = \left\| \tilde{P}_i T^T - \tilde{Q}_i \right\|_F^2 \tag{7}$$

where

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \\ \vdots & \vdots \\ p_{n1} & p_{n2} \end{pmatrix}, W_i = \begin{pmatrix} \sqrt{\sigma_i(\mathbf{p}_1)} & & & 0 \\ & \sqrt{\sigma_i(\mathbf{p}_2)} & & \\ & & \ddots & \\ & & & \sqrt{\sigma_i(\mathbf{p}_n)} \end{pmatrix}, P_i = \begin{pmatrix} p_{i1} & p_{i2} \\ p_{i1} & p_{i2} \\ \vdots & \vdots \\ p_{i1} & p_{i2} \end{pmatrix},$$

$$Q = \begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \\ \vdots & \vdots \\ q_{n1} & q_{n2} \end{pmatrix}, Q_i = \begin{pmatrix} q_{i1} & q_{i2} \\ q_{i1} & q_{i2} \\ \vdots & \vdots \\ q_{i1} & q_{i2} \end{pmatrix}, \tilde{P}_i = W_i(P - P_i), T = \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}, \tilde{Q}_i = W_i(Q - Q_i) \tag{8}$$

The minimum of the error function is obtained with the derivative with respect to T. Setting the derivative to zero, we have

$$T_i = ((\tilde{P}_i^T \tilde{P}_i)^{-1} \tilde{P}_i^T \tilde{Q}_i)^T \tag{9}$$

complexity analysis

With n the number of control points, we need $O(n)$ time to get matrix T. And for every pixels in the image, we need $O(n)$ time to compute the weight and get the result coordinate.So the complexity is $O(nN + n^2)$ where N is the number of pixels.

2.2 Radial Basis Functions Methods

Another popular approach to scattered data interpolation is to construct the interpolation function as a linear combination of basis functions, then determine the coefficients of the basis functions,

$$f(\mathbf{p}) = \sum_{i=1}^{n} \alpha_i f_i(d(\mathbf{p}, \mathbf{p}_i)) + p_m(\mathbf{p}) \tag{10}$$

The values of the basis funciton f_i depend only on the distance from the data point and are thus called radial. $p_m(\mathbf{p})$ is a polynomial of degree m.

Linear polynomials, where $m = 1$, give very good results—often better than higher degree. More easily, an identical transform is usually sufficient if no strong global rotations are involved. In this experiment, we will just let p_m be an identical transform, so the application of radial basis functions to the problem of deformation gives

$$\mathbf{f}(\mathbf{p}) = \sum_{i=1}^{n} \boldsymbol{\alpha}_i f(d(\mathbf{p}, \mathbf{p}_i)) + \mathbf{p} \tag{11}$$

In term of radial basis function part, the thin-plate spline, the Gaussian and the multiquadrics are all commonly used. In this experiment we will choose the multiquadrics, which is

$$f(d) = (d^2 + r^2)^{\mu/2} \tag{12}$$

And we used individual values r_i for each data points, computed from the distance to the nearest neighbor:

$$r_i = \min_{i \neq j} d_i(\mathbf{p}_j) \tag{13}$$

So the resulting mapping function is then

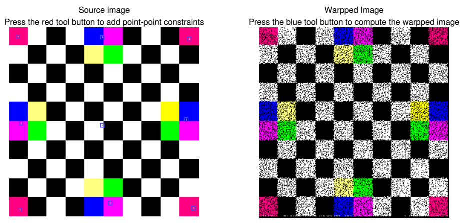
$$\mathbf{f}(\mathbf{p}) = \sum_{i=1}^{n} \boldsymbol{\alpha}_i (d(\mathbf{p}, \mathbf{p}_i)^2 + r_i^2)^{\mu/2} + \mathbf{p} \tag{14}$$

where $\boldsymbol{\alpha}_i$ can be calculated with $\mathbf{f}(\mathbf{p}_i) = \mathbf{q}_i$ by solving 2N linear equations.

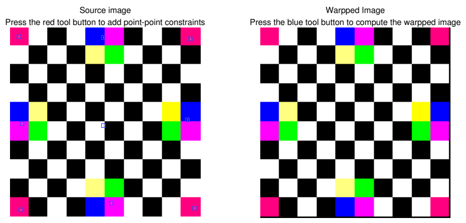
complexity analysis

For every pixel, we need $O(n)$ time to compute the result of radial basis functions. Besides, we need another $O(n^3)$ time to get the coefficients α_i. So it gives the complexity of $O(nN + n^3)$.

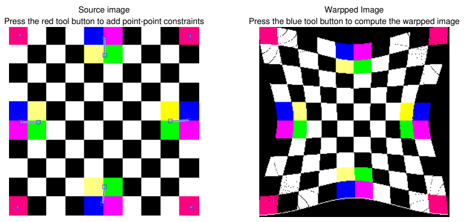
3 Results


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

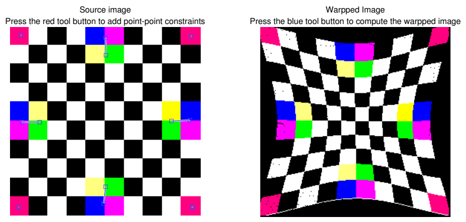
(a) IDW


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

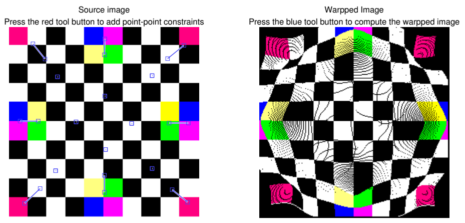
(b) RBF


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

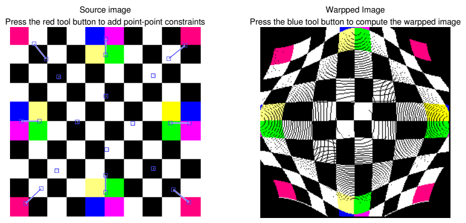
(c) IDW


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

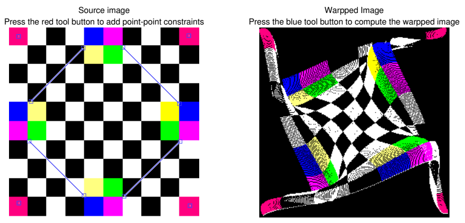
(d) RBF


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

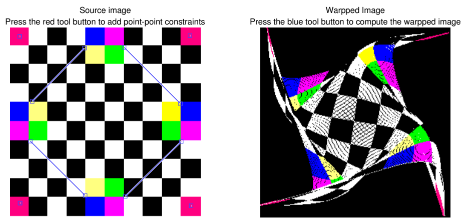
(e) IDW


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

(f) RBF


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

(g) IDW


Source image
Press the red tool button to add point-point constraints
Warpped Image
Press the blue tool button to compute the warpped image

(h) RBF

4 Performance Comparisons

The table below shows the runtime of two algorithm in a 256×256-pixel image with different control points numbers in our test.

n	4	8	16	24	32
IDW(s)	0.947608	1.506009	3.044657	3.725757	4.742689
RBF(s)	0.297725	0.357402	0.536027	0.549384	0.667023

we can see in the same image with the same control points, RBF method spent less time. And with the same number of control points, RBF method leads to a more precise result(see figure(a) and(b)).

Also, in the image test, we can see some disadvantages of IDW and RBF. One of them is that we cannot mapping the original image to every pixel of the result image, which causes the black hole or gap in the result image. The other disadvantage is that in some extreme situation, the IDW and RBF will cause the foldover in the result image(see figure (g) and (h)).

A Code

All the code of this assignment can be downloaded from `https://github.com/mathendy/MS-USTC/tree/master/Code%20Training/matlab-1_assignment`.