

FDU 数字图像处理 7. 图像配准

本文参考以下教材:

- Digital Image Processing (4th Edition, R. Gonzalez, R. Woods) Chapter 2
- 数字图像处理 (第四版, R. Gonzalez, R. Woods) 第 2 章
- 医学图像配准技术与应用 (吕晓琪, 张宝华, 杨立东, 赵瑛) 第 2 章
- [Medical Image Registration \(D. Hill, P. Batchelor, M. Holden, D. Hawkes\)](#)
- Medical Image Registration (J. Hajnal, D. Hill, D. Hawkes) Chapter 13

欢迎批评指正!

7.1 图像内插

内插 (interpolation) 是用已知数据来估计未知位置的值的过
程, 它通常在图像放大、缩小、旋转和几何校正等任务中使用。

下面用一个简单的例子开始这一主题的探讨:

假设一幅大小为 500×500 像素的灰度图像要放大 1.5 倍, 即放大到 750×750 像素。

一种简单的放大方法是, 创建一个大小为 750×750 像素的假想网格, 网格的像素间隔与原图像的像素间隔相同。

然后收缩这个网格, 使它完全与原图像重叠。

显然, 收缩后的 750×750 网格的像素间隔要小于原图像的像素间隔。

我们可以基于原图像的灰度给新图像的像素赋值, 最后将图像展开到指定的大小, 得到放大后的图像。

赋值的方法有以下几种:

- ① **最邻近内插 (nearest neighbor interpolation):**
将原图像中最近邻的灰度作为新图像中待求位置的灰度。
这种方法简单, 但会产生一些人为失真, 例如严重的直边失真。
- ② **双线性内插 (bilinear interpolation):**
使用尺寸为 $M \times N$ 的原图像 f 中的 4 个最近邻的灰度来计算新图像 g 中待求位置 (x, y) 的灰度。
取 x, y 的小数部分为 dx, dy , 记四个邻近点的灰度值为 $I_{11}, I_{12}, I_{22}, I_{21}$ (左上, 右上, 右下, 左下)

$$\begin{aligned} dx &= x - \lfloor x \rfloor \\ dy &= y - \lfloor y \rfloor \\ I_{11} &= f(\lfloor x \rfloor, \lfloor y \rfloor) \\ I_{12} &= f(\lfloor x \rfloor, \min\{\lfloor y \rfloor + 1, N - 1\}) \\ I_{21} &= f(\min\{\lfloor x \rfloor + 1, M - 1\}, \lfloor y \rfloor) \\ I_{22} &= f(\min\{\lfloor x \rfloor + 1, M - 1\}, \min\{\lfloor y \rfloor + 1, N - 1\}) \\ g(x, y) &= I_{11}(1 - dx)(1 - dy) + I_{12}(1 - dx)dy + I_{21}dx(1 - dy) + I_{22}dxdy \end{aligned}$$

- ③ **双三次内插 (bicubic interpolation):**
使用原图像 f 中的 16 个最近邻的灰度来计算新图像中待求位置 (x, y) 的灰度。

$$g(x, y) = \sum_{i,j=0}^3 a_{ij} f(x_i, y_j)$$

其中 16 个系数 a_{ij} ($i, j = 0, 1, 2, 3$) 由点 (x, y) 的 16 个最近邻点 (x_i, y_j) ($i, j = 0, 1, 2, 3$) 的梯度和 Hessian 矩阵求出。

BiCubic 基函数为:

$$W(t) := \begin{cases} \frac{3}{2}|t|^3 - \frac{5}{2}|t|^2 + 1 & \text{if } 0 \leq |t| \leq 1 \\ -\frac{1}{2}|t|^3 + \frac{5}{2}|t|^2 - 4|t| + 2 & \text{if } 1 < |t| < 2 \\ 0 & \text{otherwise} \end{cases}$$

$$a_{ij} = W(x - x_i)W(y - y_j) \quad (i, j = 0, 1, 2, 3)$$

例 2.4 图像缩小和放大的内插方法的比较。

图 2.27(a)与图 2.23(d)相同，它是由图 2.23(a)中的 930dpi 图像降低到 72dpi (大小由原来的 2136×2140 像素缩小到 165×166 像素)，然后将分辨率降低的图像再次放大到原尺寸得到的。为了生成图 2.23(d)，采用最近邻内插法进行了缩小与放大。如前所述，图 2.27(a)中的结果很差。图 2.27(b)和图 2.27(c)是分别采用双线性内插和双三次内插缩小和放大得到的。使用双线性内插得到的结果明显好于使用最近邻内插得到的结果，但结果图像稍微有点模糊。使用双三次内插得到的结果要清晰得多，如图 2.27(c)所示。



a b c

图 2.27 (a)分辨率降低到 72dpi 后，使用最近邻内插放大到原分辨率 930dpi 的图像。该图像与图 2.23(d)相同；(b)分辨率降低到 72dpi 后，使用双线性内插放大的图像；(c)与图像(b)相同但使用双三次内插的图像 ■

内插时可以使用更多的邻点，并且存在使用样条和小波的复杂技术，采用这些技术可以得到更好的结果。对于三维图形而言，在生成图像的过程中保留细节非常重要（见 Hughes and Andries[2013]）；通用数字图像处理通常不需要额外的计算开销，因此常使用双线性内插和双三次内插。

7.2 仿射变换

几何变换 (geometric transformations) 改变图像中像素的空间排列。

这些变换通常称为**橡皮膜变换** (rubber-sheet transformations)

因为它们类似于在一块橡皮膜上打印图像，然后根据预定义的一组规则来拉伸或收缩橡皮膜。

数字图像的几何变换由两种基本运算组成：

- ① 坐标的**空间变换** (spatial transformation)
二维图像的空间变换可表示为：

$$\begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}$$

- ② **灰度内插** (interpolation)，即为空间变换后的像素赋灰度值。
在进行这些变换时，我们可考虑最近邻、双线性 and 双三次内插技术。

我们主要关注**仿射变换** (affine transformation)，它保留二维空间中的点、直线和平面。

它包含缩放变换 (scaling)、平移变换 (translation)、旋转变换 (rotation) 和剪切变换 (shearing)

我们可用**齐次坐标** (homogeneous coordinates) 来表示仿射变换：

$$\begin{aligned}
 \text{2D: } \begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \\ 1 \end{bmatrix} &= A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\
 \text{3D: } \begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \\ z_{\text{new}} \\ 1 \end{bmatrix} &= A \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
 \end{aligned}$$

我们主要有两种基本方法来实现空间变换:

• ① 正向映射 (forward mapping)

扫描输入图像的像素 (x, y) , 计算输出图像中像素的空间位置 $(x_{\text{new}}, y_{\text{new}})$

但一个问题是, 输入图像中的某些像素可能变换到输出图像中的同一位置,

这就产生了如何把多个输出值合并为单个输出像素值的问题.

此外, 某些输出位置可能根本没有要赋值的像素.

• ② 反向映射 (inverse mapping)

扫描输出像素的位置 $(x_{\text{new}}, y_{\text{new}})$, 并计算输入图像中的相应位置 (x, y) (注意处理越界/缺失位置)







然后在最近的输入像素之间进行内插, 求出输出像素的灰度值.

就实现而言, 反向映射要比正向映射更有效, 因而被许多商业软件采用.

(Python 实现参见 Homework 07)

TABLE 2.3

Affine transformations based on Eq. (2-45).

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x \cos \theta - y \sin \theta$ $y' = x \sin \theta + y \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

例 2.9 图像旋转和灰度内插。

本例的目的是说明如何用仿射变换来旋转图像。图 2.40(a)显示了一幅简单的图像,图 2.40(b)至(d)是(使用反向映射)将原图像旋转 -21° 后的结果(在表 2.3 中,顺时针转角是负的)。灰度赋值分别是用最近邻、双线性和双三次内插计算得到的。图像旋转的一个关键问题是保持直线特性。如图 2.40(f)到(h)中放大的边缘部分所示,最近邻内插产生了锯齿最大的边缘,并且如 2.4 节所示,双线性内插产生了明显改进的结果。同样,使用双三次内插产生了更好的结果。事实上,比较图 2.40(f)至(h)中一系列放大后的细节,会发现在最后一幅图像中,从白色(255)到黑色(0)的过渡是平滑的,因为边缘区域有更多的值,并且这些值的分布更为平衡。虽然双线性内插和双三次内插导致的小灰度差在人的视觉分析中不是很明显,但它们在图像数据处理中非常重要,如旋转后的图像中的自动边缘跟踪。

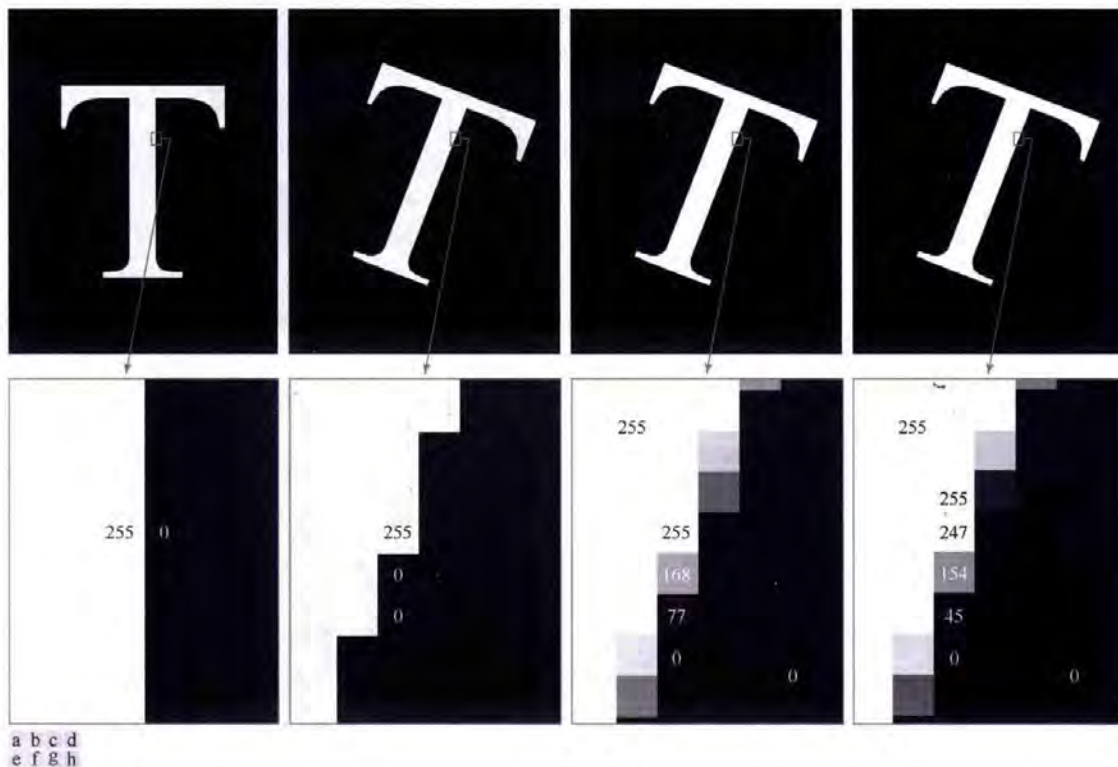


图 2.40 (a)字母 T 的 541×421 图像; (b)旋转 -21° 并用最近邻内插赋灰度值后的图像; (c)旋转 -21° 并用双线性内插赋灰度值后的图像; (d)旋转 -21° 并用双三次内插赋灰度值后的图像; (e)~(h)放大部分(每个方格是一个像素,显示的数字是灰度值)

包含旋转图像的空间矩形要大于包含原图像的矩形,如图 2.41(a)和(b)所示。处理这一问题时我们有两种选择:(1)修剪旋转图像,使其大小与原图像的大小相同,如图 2.41(c)所示;(2)保留包含整个旋转图像的较大图像,如图 2.41(d)所示。在图 2.40 中,我们使用了第一种选择,因为旋转并未使得感兴趣的目标落到原矩形的边界之外。旋转图像中不包含图像数据的几个区域必须填充某个值,通常填充 0 值(黑色)。注意,逆时针方向旋转为正旋转,这是图像坐标系的建立方式(见图 2.19)和表 2.3 中定义的旋转方式的结果。

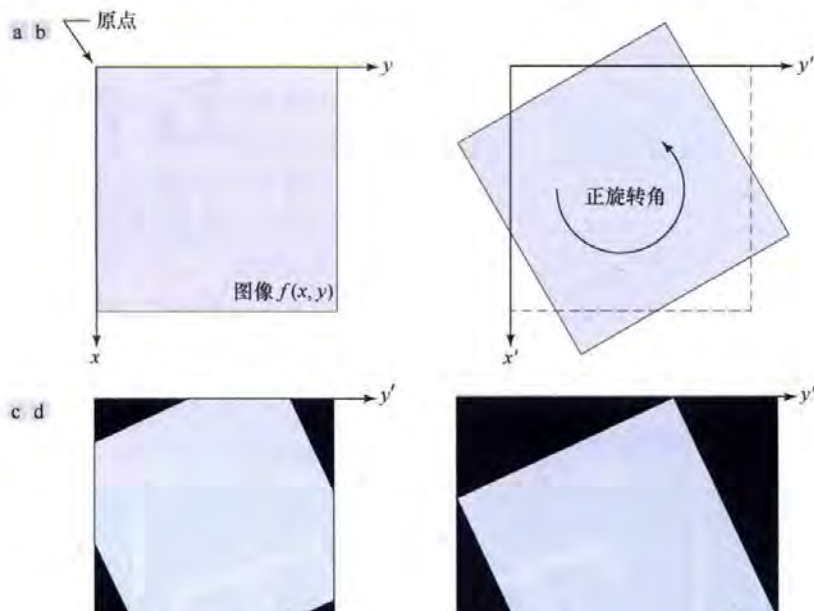




图 2.41 (a)一幅数字图像; (b)旋转后的图像 (注意正旋转角的逆时针方向); (c)裁剪后匹配原图像区域的旋转图像; (d)放大后容纳整个旋转图像的图像

(医学图像配准技术与应用, 2.2.4 节)

两种常用的仿射变换:

- ① 相似变换: (4 个自由度)

$$\begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \\ 1 \end{bmatrix} = A(r, \theta, t_x, t_y) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} r \cos(\theta) & r \sin(\theta) & t_x \\ -r \sin(\theta) & r \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

其中 $r > 0$ 代表各向同性缩放。

可以证明相似变换可以保持两条曲线在交点处的角度, 即保角性 (因此相似变换又称同形变换)

图 2.31 分别给出用 $s = 1.5$, $\theta = -90^\circ$ 和 $t = [1, 0]^T$; $s = 1$, $\theta = 180^\circ$ 和 $t = [4, 8]^T$; $s = 0.5$, $\theta = 0^\circ$ 和 $t = [5, 7]^T$ 定义的相似变换对左边的多边形目标变换得到的 3 个结果。

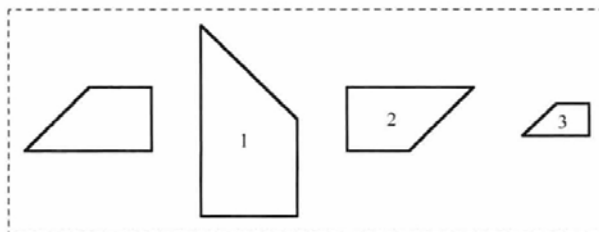


图 2.31 对区域进行相似变换得到的结果

- ② 等距变换: (4 个自由度)

$$\begin{bmatrix} x_{\text{new}} \\ y_{\text{new}} \\ 1 \end{bmatrix} = A(e, \theta, t_x, t_y) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} e \cos(\theta) & \sin(\theta) & t_x \\ -e \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

其中 $e = \pm 1$ (若 $e = 1$, 则保持朝向; 若 $e = -1$, 则反转朝向, 即镜像变换)

可以证明等距变换保持了任意两点之间的距离 (与相似变换相比, 少了各向同性缩放)

图 2.32 分别给出用 $e = 1$, $\theta = -90^\circ$ 和 $t = [2, 0]^T$; $e = -1$, $\theta = 180^\circ$ 和 $t = [4, 8]^T$; $e = 1$, $\theta = 180^\circ$ 和 $t = [5, 6]^T$ 定义的等距变换对左边的多边形目标变换得到的结果。

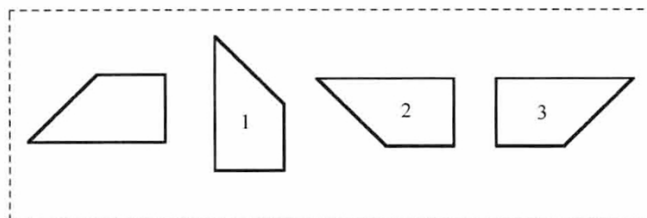


图 2.32 对多边形目标进行等距变换得到的结果

7.3 非线性变换

7.3.1 局部仿射

给定 n 个局部区域 R_i , 分别对应仿射变换 T_i

记 (x, y) 到区域 R_i 的距离为 $d_i(x, y)$

定义 $(x, y) \notin \bigcup_{i=1}^n R_i$ 关于区域 R_i 的权重为:

(良好的权重定义要求同一点对应的权重之和为 1)

$$w_i(x, y) := \frac{\frac{1}{(d_i(x, y))^p}}{\sum_{k=1}^n \frac{1}{(d_k(x, y))^p}}$$

其中 p 的一个比较好的选择是 $p = 1.5$

则局部仿射变换 T 由以下公式给出:

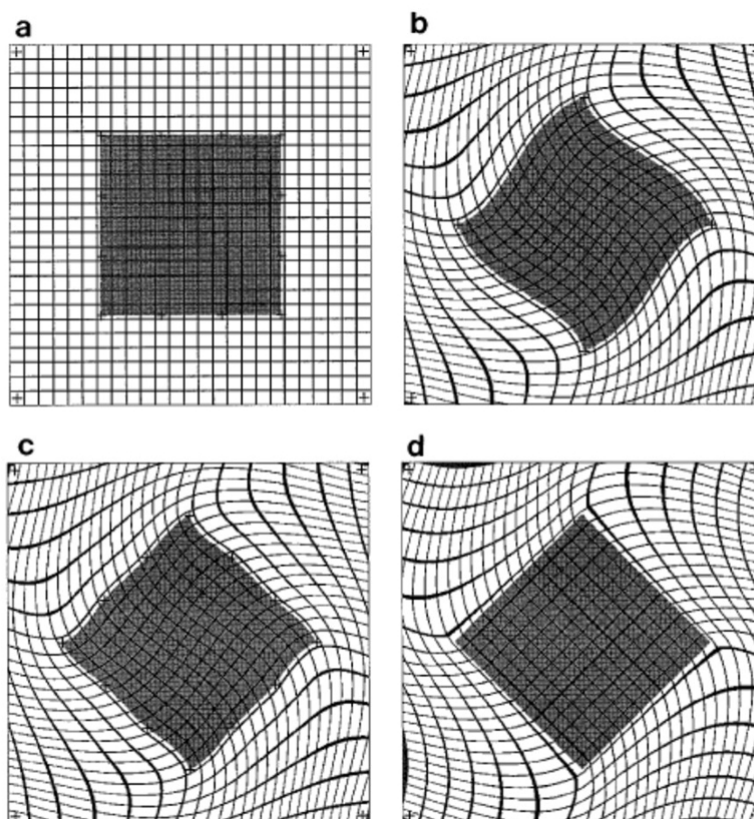
$$T(x, y) := \begin{cases} T_i(x, y) & \text{if } (x, y) \in R_i \text{ for a certain } i = 1, \dots, n \\ \sum_{i=1}^n w_i(x, y) T_i(x, y) & \text{otherwise} \end{cases}$$

基本思想: 区域控制 + 全局影响.

(Python 实现参见 Homework 07)

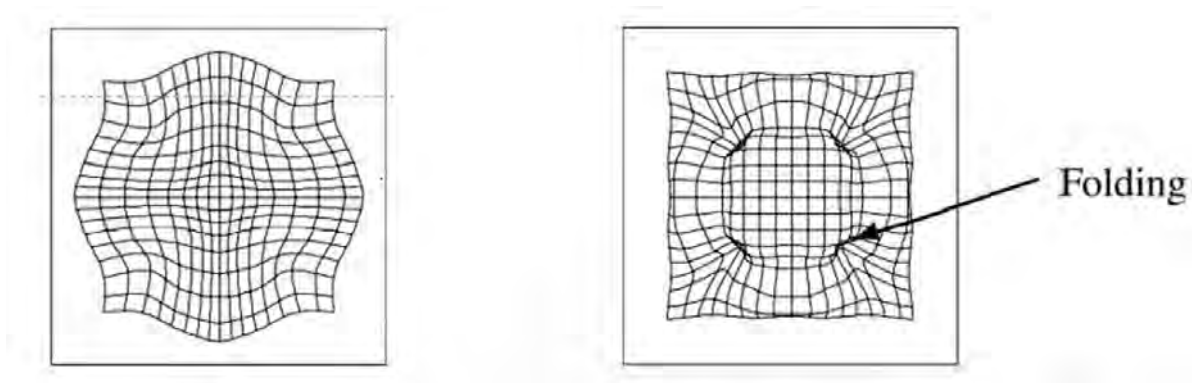
优化方法:

- 将区域控制调整为点控制 (即将区域 R_i 调整为单个控制点 $\phi_i = (x_i, y_i)$)
- 将全局影响调整为局部影响 (即将全局性权重调整为局部性权重, 使得控制点只对局部区域有影响, 减少计算开销)



缺点: 不保证光滑, 容易产生褶皱问题.

非线性变换导致两个点映射到同一个位置 (导致没有逆变换), 称为褶皱 (folding) 问题:



我们可以通过将变换函数光滑化来解决某些折叠问题，但不能保证解决所有折叠问题。例如我们可以将某个有折叠问题的局部仿射变换拆分成若干个小的局部仿射变换去做，以保证每一步都没有折叠问题 (庄老师说 PPT 里的结果这么好就是因为使用了这种 "continuization" 的做法)

$$T_{\text{sum}} = T_n \circ T_{n-1} \circ T_{n-2} \circ \cdots \circ T_2 \circ T_1$$

局部仿射变换是否有折叠问题可以根据其 Jacobi 行列式是否小于零来检验。

7.3.2 基于样条的变换

我们可以使用**样条** (spline) 来建模空间变换:

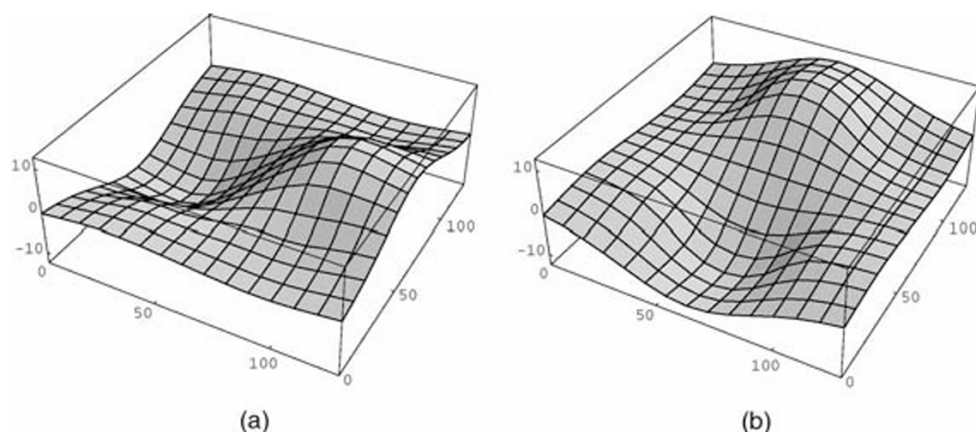


FIGURE 13.2

An example of a nonrigid transformation required to warp a square into a circle. The corresponding transformation is shown as two separate surfaces defining (a) the displacement in the horizontal direction and (b) the displacement in the vertical direction.

许多使用样条的配准技术基于这样一个假设:

我们可以在源图像和目标图像中识别出一组对应点，称为**控制点** (control point) 或**地标** (landmark)

我们在这些控制点处使用插值，而在控制点之间提供一个平滑变化的位移场。

控制点的选取主要有两种方法:

- ① 使用在两幅图像中都能识别的解剖学或几何学地标作为控制点，并通过优化像素相似性度量来更新控制点位置。
- ② 将控制点等距排列在图像上，形成一个规则的网格。此时控制点仅用作变换的参数化，并不对应于解剖学或几何学地标。因此这类控制点通常被称为**伪地标** (pseudo-landmark)

7.3.3 薄板样条

薄板样条 (thin-plate spline) 是基于径向基函数 (radial basis functions) 的一类样条。

它可被定义为若干个径向基函数的线性组合。

(点控制 + 全局影响 + 任意控制点)

薄板样条函数可以将映射分解为仿射变换和非线性变换，

图像中一个控制点的移动将影响到整个图像控制点的移动。

给定目标图像的 n 个控制点 $\phi_1 = (x_1, y_1), \dots, \phi_n = (x_n, y_n)$

(对应的源图像的 n 个控制点为 $\phi'_1 = (x'_1, y'_1), \dots, \phi'_n = (x'_n, y'_n)$)

定义径向基函数 $\sigma(d)$ 为:

$$\sigma(d) := d^2 \log(d)$$

其中 d 代表像素点与控制点之间的某种距离。

基于薄板样条的插值函数即为:

$$\begin{aligned} T(x, y) &= \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}^T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \sum_{i=1}^n \begin{bmatrix} b_{i1} \\ b_{i2} \end{bmatrix} \sigma(\|\phi_i - (x, y)\|_2) \\ &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}^T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ \vdots & \vdots \\ b_{n1} & b_{n2} \end{bmatrix}^T \begin{bmatrix} \sigma(\|\phi_1 - (x, y)\|_2) \\ \vdots \\ \sigma(\|\phi_n - (x, y)\|_2) \end{bmatrix} \\ &= A^T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + B^T \begin{bmatrix} \sigma(\|\phi_1 - (x, y)\|_2) \\ \vdots \\ \sigma(\|\phi_n - (x, y)\|_2) \end{bmatrix} \end{aligned}$$

其中 $A \in \mathbb{R}^{3 \times 2}$ 和 $B \in \mathbb{R}^{n \times 2}$ 为参数矩阵，共计 $2(n+3)$ 个参数。

目标图像和源图像的 n 对控制点 $(\phi_1, \phi'_1), \dots, (\phi_n, \phi'_n)$ 的对应可以提供 $2n$ 个方程。

(这是因为薄板样条要求目标图像的控制点和源图像的控制点完全重合)

考虑到总参数量为 $2(n+3)$ ，故我们增加 6 个方程以保证解的唯一性:

$$B^T \Phi = \begin{bmatrix} b_{11} & b_{12} \\ \vdots & \vdots \\ b_{n1} & b_{n2} \end{bmatrix}^T \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n b_{i1} x_i & \sum_{i=1}^n b_{i1} y_i & \sum_{i=1}^n b_{i1} \\ \sum_{i=1}^n b_{i2} x_i & \sum_{i=1}^n b_{i2} y_i & \sum_{i=1}^n b_{i2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = 0_{2 \times 3}$$

我们记:

$$\begin{aligned} \Phi &= \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \in \mathbb{R}^{n \times 3} \\ \Phi' &= \begin{bmatrix} x'_1 & y'_1 \\ \vdots & \vdots \\ x'_n & y'_n \end{bmatrix} \in \mathbb{R}^{n \times 2} \\ \Sigma &= [\sigma(\|\phi_i - \phi_j\|_2)]_{i,j=1}^n = \begin{bmatrix} \sigma(\|\phi_1 - \phi_1\|_2) & \cdots & \sigma(\|\phi_1 - \phi_n\|_2) \\ \vdots & & \vdots \\ \sigma(\|\phi_n - \phi_1\|_2) & \cdots & \sigma(\|\phi_n - \phi_n\|_2) \end{bmatrix} \in \mathbb{R}^{n \times n} \end{aligned}$$

(注意: Σ 的对角元均为 0, 因为 $\sigma(0) = \lim_{d \rightarrow 0} d^2 \log(d) = 0$)

则可得到以下线性方程组:

$$\begin{bmatrix} \Sigma & \Phi \\ \Phi^T & 0_{3 \times 3} \end{bmatrix} \begin{bmatrix} B \\ A \end{bmatrix} = \begin{bmatrix} \Phi' \\ 0_{3 \times 2} \end{bmatrix}$$

求解上述线性方程组即可得到参数矩阵 $A \in \mathbb{R}^{3 \times 2}$ 和 $B \in \mathbb{R}^{n \times 2}$

7.3.4 B-样条

(医学图像配准技术与应用, 6.1 节)

使用径向基函数会导致每个控制点对变换都有全局影响, 这可能会导致局部形变的建模变得困难. 此外, 当控制点的数量较多时, 径向基函数样条的计算代价也会变得非常昂贵.

一个替代方案是使用**自由形状变换** (Free-Form Deformation, FFD)

(点控制 + 局部影响 + 规则化控制点)

其基本思想是通过操作一个控制点网格来变形物体.

由此产生的形变控制了 3D 物体的形状, 并产生平滑连续的变换.

与允许控制点任意配置的径向基函数样条不同,

基于样条的 FFD 需要一个规则的控制点网格, 且控制点之间均匀间隔.

考虑一个基于 B-样条的 FFD 模型.

我们将二维图像表示为一个网格图, 每个网格交叉点代表一个控制点.

设 x 轴方向上网格间距为 δ_x , 分为 n_x 段, 而 y 轴方向上网格间距为 δ_y , 分为 n_y 段.

(医学图像配准通常选择 $\delta_x = \delta_y = 4$)

记控制点网格为 $\Phi = [\phi_{i,j}]$ (其中 $0 \leq i \leq n_x, 0 \leq j \leq n_y$)

为配准图像中的控制点, 我们设其在周围四个控制点的网格范围内移动.

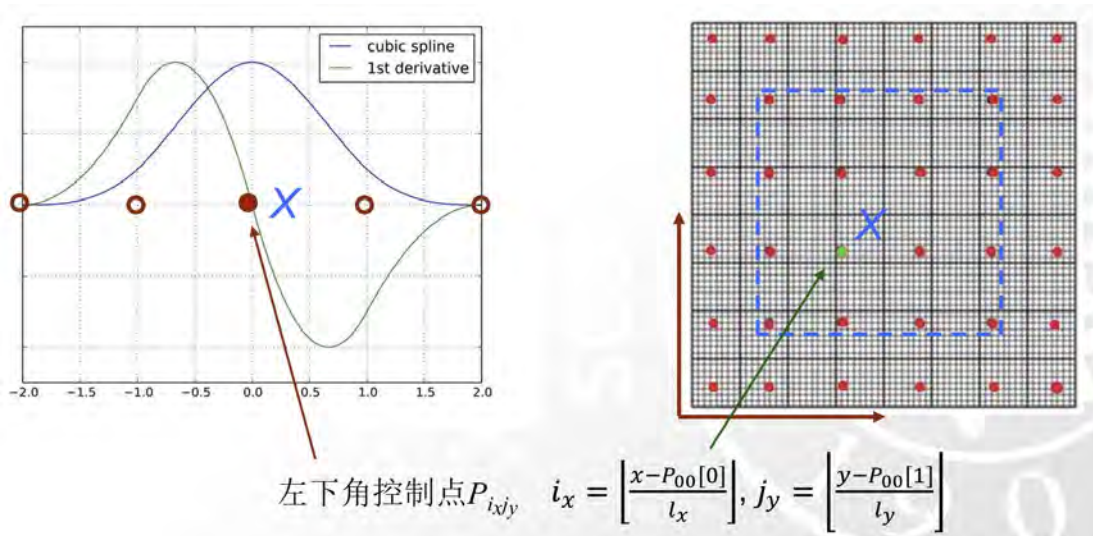
根据 B-样条理论, 点 (x, y) 到 $(x + \Delta x, y + \Delta y)$ 的位移量可表示为:

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \sum_{l=0}^3 \sum_{k=0}^3 B_l(u) B_k(v) \phi_{i+l, j+k} \quad \text{where} \quad \begin{cases} i = \left\lfloor \frac{x}{\delta_x} \right\rfloor - 1 \\ u = \frac{x}{\delta_x} - \left\lfloor \frac{x}{\delta_x} \right\rfloor \\ j = \left\lfloor \frac{y}{\delta_y} \right\rfloor - 1 \\ v = \frac{y}{\delta_y} - \left\lfloor \frac{y}{\delta_y} \right\rfloor \end{cases}$$

三次 B-样条函数的定义为: (其中 $u \in [0, 1)$)

(事实上它们是三次 B-样条函数的 4 段, 这样拆分可以方便代码实现)

$$\begin{aligned} B_0(u) &:= \frac{1}{6}(1-u)^3 \\ B_1(u) &:= \frac{1}{6}(3u^3 - 6u^2 + 4) \\ B_2(u) &:= \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1) \\ B_3(u) &:= \frac{1}{6}u^3 \end{aligned}$$



与薄板样条不同，B-样条由于其具有**局部支撑性**，在计算位移量时仅与周围 4×4 个控制点有关。单个控制点位置的改变影响的只是该控制点周围网络区域内的点，而不影响更远的区域。（控制点越多，移动单个控制点所需改变的区域）

我们根据相似度函数的差分来迭代地移动单个控制点的位置。

(Python 实现参见 Homework 07)

三维情形:

每个点的位移量仅与周围 $4 \times 4 \times 4 = 64$ 个控制点有关。

FFD-3D Regular lattice (with fixed spacing): l_x, l_y, l_z
 Control point indices $\{i, j, k\} \in [0, n_x/y/z - 1]$
 whose coordinate: P_{ijk} ; displacement: ϕ_{ijk}
 Given FFD T , deformation for $X = [x, y, z]^T \in \Omega$:
 $Y = T(X) = X + Q_{local}(X)$ where $Q_{local}(X) = \sum_{i,j,k} \phi_{ijk} W(X, P_{ijk})$

$$Q_{local}(X) = \sum_{i=-1}^2 \sum_{j=-1}^2 \sum_{k=-1}^2 \phi_{i+l_x, j+l_y, k+l_z} \beta^{(3)}(u_i) \beta^{(3)}(v_j) \beta^{(3)}(w_k)$$

$$u_i = \frac{x - P_{[ijk;X][0]}}{l_x}; v_j = \frac{y - P_{[ijk;X][1]}}{l_y}; w_k = \frac{z - P_{[ijk;X][2]}}{l_z}$$

当前控制点 $P_{[ijk;X]} = P_{i+l_x, j+l_y, k+l_z}$

左下角控制点 $P_{i_x j_y k_z}$ $i_x = \left\lfloor \frac{x - P_{000}[0]}{l_x} \right\rfloor, j_y = \left\lfloor \frac{y - P_{000}[1]}{l_y} \right\rfloor, k_z = \left\lfloor \frac{z - P_{000}[2]}{l_z} \right\rfloor$

FFD-3D: 另一种表达方式

$$Q_{local}(X) = \sum_{i=-1}^2 \sum_{j=-1}^2 \sum_{k=-1}^2 \phi_{i+l_x, j+l_y, k+l_z} \beta_i(u) \beta_j(v) \beta_k(w)$$

$$u = \frac{x - P_{000}[0]}{l_x} - i_x; v = \frac{y - P_{000}[1]}{l_y} - j_y; w = \frac{z - P_{000}[2]}{l_z} - k_z$$

左下角控制点 $P_{i_x j_y k_z}$ $i_x = \left\lfloor \frac{x - P_{000}[0]}{l_x} \right\rfloor, j_y = \left\lfloor \frac{y - P_{000}[1]}{l_y} \right\rfloor, k_z = \left\lfloor \frac{z - P_{000}[2]}{l_z} \right\rfloor$

$u, v, w \in [0, 1]$

使用 $\beta_{-1}(u), \beta_0(u), \beta_1(u), \beta_2(u)$

替代原来的 $\beta^{(3)}(a)$, $a = \frac{|x - P_i|}{l}$

$$\beta^{(3)}(a) = \begin{cases} \frac{1}{6}(4 - 6a^2 + 3|a|^3), & 0 \leq |a| < 1 \\ \frac{1}{6}(2 - |a|)^3, & 1 \leq |a| < 2 \\ 0, & 2 \leq |a| \end{cases}$$

$u, v, w \in [0, 1]$

使用 $\beta_{-1}(u), \beta_0(u), \beta_1(u), \beta_2(u)$

替代原来的 $\beta^{(3)}(a)$, $a = \frac{|x - P_i|}{l}$

$$\beta^{(3)}(a) = \begin{cases} \frac{1}{6}(4 - 6a^2 + 3|a|^3), & 0 \leq |a| < 1 \\ \frac{1}{6}(2 - |a|)^3, & 1 \leq |a| < 2 \\ 0, & 2 \leq |a| \end{cases}$$

$\beta_{-1}(u) = \beta^{(3)}(a)$, 其中 $a = u + 1$

$\beta_0(u) = \beta^{(3)}(a)$, 其中 $a = u$

$\beta_1(u) = \beta^{(3)}(a)$, 其中 $a = 1 - u$

$\beta_2(u) = \beta^{(3)}(a)$, 其中 $a = 2 - u$

$\beta_{-1}(u) = (1 - u)^3 / 6$

$\beta_0(u) = (3u^3 - 6u^2 + 4) / 6$

$\beta_1(u) = (-3u^3 + 3u^2 + 3u + 1) / 6$

$\beta_2(u) = u^3 / 6$

7.4 图像配准

图像配准 (image registration) 用于对齐同一场景的两幅或多幅图像。

组合或比较这些图像都要求对几何失真进行补偿，

其中几何失真由视角、距离、方向、传感器分辨率、物体位置的移动和其他因素导致。

解决上述问题的主要方法之一是使用**控制点** (control point)，即其精确位置在输入图像和参照图像中已知的对应点。

选取控制点的方法有多种，既可以交互地选择，也可以用自动检测算法进行选择。

我们主要有两种基本方法来实现空间变换:

- ① 正向映射 (forward mapping)

扫描输入图像的像素 (x, y) , 计算输出图像中像素的空间位置 $(x_{\text{new}}, y_{\text{new}})$

但一个问题是, 输入图像中的某些像素可能变换到输出图像中的同一位置, 这就产生了如何把多个输出值合并为单个输出像素值的问题.

此外, 某些输出位置可能根本没有要赋值的像素.

- ② 反向映射 (inverse mapping)

扫描输出像素的位置 $(x_{\text{new}}, y_{\text{new}})$, 并计算输入图像中的相应位置 (x, y) (注意处理越界/缺失位置)

然后在最近的输入像素之间进行内插, 求出输出像素的灰度值.

就实现而言, 反向映射要比正向映射更有效, 因而被许多商业软件采用.

配准过程 $\hat{T} = \underset{T}{\operatorname{argmax}} (\mathcal{S}(I_1, I_2, T) + \mathcal{R}(\cdot))$

\mathcal{S} : Similarity; T : Transformation; \mathcal{R} : Regularization

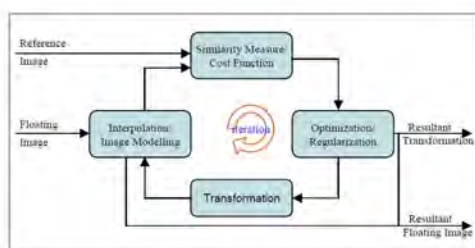


Figure 3.5: Framework of image registration

[Zhuang, X. PhD thesis 2010, UCL]

配准过程 $\hat{T} = \underset{T}{\operatorname{argmax}} (\mathcal{S}(I_1, I_2, T) + \mathcal{R}(\cdot))$

\mathcal{S} : Similarity; T : Transformation; \mathcal{R} : Regularization

- 空间变换 (affine)
- 相似性测度 (目标函数: 灰度值SSD; 正则化项: 无)
- 图像建模和插值算法 (线性插值)
- 优化过程 (最优化目标函数求解最优空间变换参数: 梯度下降)
 - 梯度: 有限差分 ($\frac{\partial f(x)}{\partial x} \approx \frac{f(x+h) - f(x-h)}{2h}$)

图像灰度值差 (sum square difference, SSD) 相似性度量

$$\mathcal{S} = \text{SSD} = \frac{1}{N} \sum_{x \in \Omega_{A,T(B)}} |I_A(x) - I_B(T(x))|^2$$

The End