医学图像配准

5.1 医学图像配准概述

图像配准(Image Registration)技术现已广泛应用于模式识别、计算机视觉、医学影像处理、遥感数据处理等诸多领域[1]。本章中我们主要讨论其在医学领域的应用。

图像配准是指在两幅图像相应点之间建立一个一一映射的过程,也就是说,将两幅图像中对应于空间同一位置的点联系起来,这里的映射一般称为变换(Transform)。

图 5-1 简单说明了一个二维图像配准的概念。对于两幅图像,设 $f(x)$表示参考图像(Fixed Image),$m(x)$表示浮动图像(Moving Image),其中 x 描述 N 维坐标(参数)空间中任意一点的几何坐标。配准过程可描述为:在 N 维参数空间中找到几何变换 T,使得匹配度评价函数 S 达到极值。配准的结果是使经过几何变换 T 的浮动图像 $m(x)$上的像素点,与固定图像 $f(x)$在相同的几何位置坐标上具有对应性关系。

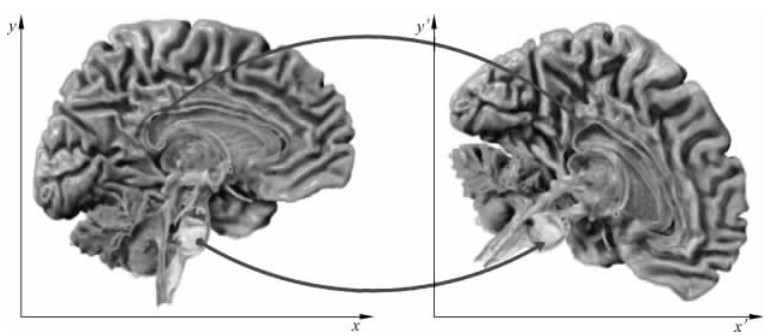


图 5-1 图像配准概念图示

$$\arg\max_{T} S(f(x), m(T(x)), T) \tag{5-1}$$

图像不同的获取方式形成了配准问题的不同方面,同时也代表着不同的应用领域。这些方式可归纳为以下 4 类情况[1]。

(1) 不同的时间(Multi-Temporal):对从相同场景、不同时间得到的图像进行配准处理,其目的是为了获得场景的变化信息或进行目标跟踪。如医学上对某器官病变情况的监视。

(2) 不同的视场(Multi-View):对于二维或三维的场景,使用不同的视角获取图像的局部,这种情况发生在设备成像范围有限,需要多次成像来获取场景的全貌的情形,如医学影像中的拼图。

(3) 不同的成像模式(Multi-Model):综合利用多源图像信息,获得更全面更详细的场景表达。医学图像领域的多模态配准问题多属于这一方面。

(4) 从场景到模型的配准(Scene To Model):将从传感器中获得的真实图像数据与虚拟、标准化的模型数据做对比,以实现基于模型的目标识别与跟踪。如在医学中,将病人的解剖图像与标准数字解剖图库(虚拟人技术)进行对比,对样本图像分类等。

图像配准在临床医学中的应用主要有以下一些方面。

(1) 不同成像设备间的图像信息融合,主要是高分辨率的结构图像(CT、MRI)与低分辨率的功能图像(SPECT、PET)融合,提高定位精度。

(2) 肿瘤的早期发现与定位,提高手术的精确性。

(3) 为数字血管剪影(DSA)提供前期和后期图像处理,白内障诊断。

(4) 脑部疾病诊断,如引导神经外科手术。

20 世纪以来医学成像技术经历了一个从静态到动态、从形态到功能、从平面到立体的发展过程,尤其在计算机技术高度发达之后,医学成像技术的发展给临床医学提供了 X 线、超声、计算机断层成像(CT)、数字减影血管造影(DSA)、单光子发射断层成像(SPECT)、磁共振成像(MRI)、数字荧光造影(DF)、正电子发射断层成像(PET)等形态和功能的影像信息。根据医学图像所提供的信息内涵,可将这些信息分为两大类:解剖结构图像(CT、MRI、B 超等)和功能图像(SPECT、PET 等)。这两类图像各有其优缺点:功能图像分辨率较差,但它提供的脏器功能代谢信息是解剖图像所不能替代的;解剖图像以较高的分辨率提供了脏器的解剖形态信息(功能图像无法提供脏器或病灶的解剖细节),但无法反映脏器的功能情况。目前这两类成像设备的研究都已取得了很大的进步,图像的空间分辨率和图像质量有很大的提高,但由成像原理不同所造成的图像信息局限性,使得单独使用某一类图像的效果并不理想,而多种图像的利用又必须借助医生的空间构想和推测去综合判定他们所要的信息,其准确性受到主观影响,更主要的是一些信息将可能被忽视。解决这个问题的最有效方法,就是以医学图像配准技术为基础,利用信息融合技术,将这两种图像结合起来,利用各自的信息优势,在一幅图像上同时表达来自人体的多方面信息。使人体内部的结构、功能等多方面的状况通过影像反映出来,从而更加直观地提供了人体解剖、生理及病理等信息。其中图像配准技术是图像融合的关键和难点[32]。

进入 21 世纪后,非刚性配准算法方面发展很快,Arno Klein et al.[4]使用 8 种不同的误差分析准则,对 80 个手动标记的人脑图像进行了超过 45000 次实验对 14 种非刚性配准方法进行了评估。不断地有基于新的数学模型的算法出现,如 Mark Holden[7]提到的 Miller et al. 使用微分同胚时空映射(Diffeomorphic Space-Time Mapping)来正则化速度场,以避免使用逐次超松弛迭代法(Successive Overrelaxation)求解 Navier-Stokes 方程时遇到的奇异点问题。同一时期也出现了一些新的面向非刚性图像配准的软件平台,如基于 ITK[27](Insight Segmentation and Registration ToolKit)的开源算法平台 elastix[25]在 ITK 的基础上对 ITK 做了增强与补充:加入采样策略模块和一些新的优化器。相对于 ITK 的配准算法框架,其提供了更多的用户控制与更好的错误处理机制,包括一些新的或者更灵活的代价函数及其线性组合能力,几何变换序列组合和多种空间变换一阶、二阶偏导数数值算法和图像重采样策略。这些改进使得 elastix 能够更好地支持基于 B 样条自由变换的非刚性配准算法。B. Glocker, N. Komodakis, G. Tziritas et al.[24]开发的 Drop 平台实现了基于马尔可夫随机场的三维非刚性密集图像配准算法。

集成计算机硬件的飞速发展更进一步推动了图像配准技术的研究,产生了许多成功的临床应用。目前,医学图像配准已发展成为图像处理领域里一个比较活跃的分支,每年在相关的

期刊和会议上都有大量关于医学图像配准的论文发表。

5.2 医学图像配准算法构成与分类

图像配准方法根据待配准图像的维度、模态、基于配准的特征以及采用的变换方法、图像的主体等[2]可以简单地分为以下几种。

1. 按图像维度分类

(1) 2D/2D[1]配准：主要用于断层扫描数据中切片数据的配准或者二维图像间的配准。

(2) 2D/3D 配准：主要用于空间数据和投影数据的配准(如 CT 图像和 X-ray 图像配准)或二维断层扫描数据和三维空间数据的配准。

(3) 3D/3D 配准：主要用于两个断层扫描数据间的配准或者断层扫描数据与任意给定空间数据的配准。

2. 按配准特征分类

(1) 基于外部特征：在病人身上添加人工标识物，在任意模态的医学图像上都可见，并可以被精确地标定位置。

(2) 基于内部特征：基于获得的图像本身进行配准，进一步可以细化为基于图像特征点(Landmark)、特征曲线(Curve)、特征曲面(Surface)的配准和基于图像体素(Voxel)的配准。

基于外部特征的配准很多需要在病人体内植入标识物，对病人不太友好，而且在扫描过程中严格保持不动，因此渐渐淡出主流方法。

基于内部特征点、特征曲线以及曲面的配准，需要预先对图像进行分割，以提取特征，进行下一步的配准，配准的精度受图像分割的影响比较大。基于体素的配准，无需对图像进行预处理，但是配准的计算量比较大，在实际的图像配准中，我们需要结合各种性能上的要求，在时间和精度间进行权衡，选取合适的配准方法。

3. 按变换模型分类

(1) 刚体变换：图像旋转和平移，保持体素间距离和角度不变。

(2) 仿射变换：将平行线映射为平行线。

(3) 投射变换：将直线映射为直线。

(4) 曲线变换：将直线映射为曲线。

4. 按图像模态分类

(1) 单模态图像配准：待配准图像属于同种模态(CT-CT，MR-MR，US-US 等)一般用于观察器官、组织的生长变化。

(2) 多模态图像配准：待配准图像属于不同模态(CT-MR，CT-PET，US-CT 等)多用于图像融合，综合多模态图像信息。

(3) 图像与图谱的配准：将病人图像与图谱(Atlas)进行配准，主要用于搜集某些结构信息。

(4) 病人与图像的配准：主要用于放射治疗(将实时的 X-ray 图像与预先获得的解剖图像配准)和计算机辅助手术中。

5. 按图像主体分类

(1) 同一主体(Intrasubject)：待配准图像来自同一个病人。

(2) 不同主体(Intersubject)：待配准图像来自不同的病人。

(3) 图谱(Atlas)：待配准的图像一幅来自病人，另一幅来自图谱。

以上是主流的配准方法分类，此外还有一些不常用的分类方法，如根据配准采用的优化方法分类等，限于篇幅就不一一阐述。下面我们将简要介绍一些本领域中典型的方法和问题，包括基于灰度的图像配准、基于特征的图像配准、刚性配准与仿射配准以及非刚性配准。

5.2.1 基于灰度的图像配准

基于图像灰度的配准也可称为基于图像内容的配准，这类方法在配准过程中直接利用图像本身具有的灰度统计信息来衡量图像的相似性程度，并通过最优化方法对描述图像相似性的代价函数进行寻优，从而最终实现图像的自动和高精度配准。相对于基于特征的图像配准方法，基于灰度的图像配准不需要预先对图像进行复杂的特征提取等操作，同时也避免了由稀疏特征集带来的有偏估计的问题。然而，受到图像噪声和交叠面积小等因素的影响，这类配准方法通常存在着严重的局部极值问题，同时在最优变换的搜索过程中需要巨大的计算量，因此目前的基于图像灰度的配准方法的应用范围还比较窄，算法的鲁棒性不高。

基于图像灰度的配准方法一般可以划分为 4 大方面的问题。

(1) 图像相似性测度：作为一种准则用来评价参考图和插值后得到的图像匹配的效果，它直接影响配准效果的好坏。

(2) 几何变换：将参考图像空间中像素点映射到浮动图像空间中去。几何变换的类型一般有刚性变换、仿射变换、投影变换、曲线变换等。

(3) 函数优化：图像配准可被看成一个多参数优化问题，给定一种配准准则，就可以定义一个以几何变换参数为自变量的多元目标函数，通过对该目标函数的最优化搜索得到配准时的几何变换参数。

(4) 图像插值：由于数字图像是对模拟图像的网格采样，只有格点处有像素值，而参考图像空间中的格点映射到浮动图像空间中去后可能不在格点上，为了得到这些浮动图中非格点处的像素值就有必要进行图像插值。一般常采用的插值方法有最近邻插值、线性插值、B 样条插值等。

如何衡量两幅图像的相似性程度是基于图像灰度配准的核心问题。当前有很多图像相似性计算方法，如均方误差测度、均方直方图测度、归一化相关系数测度、梯度差分测度等。此外，为了解决不同模态间的图像相似性测度问题，基于信息理论的互信息相似性准则也被引入到图像配准领域中来[5]。

灰度均方误差测度是最简单的一种图像相似性评价方法。假设 $F(x)$为参考图像灰度，$M(x)$为浮动图像灰度，N 为参与计算的像素个数统计，两幅图像的灰度均方误差测度可表示为：

$$\mathrm{MSD}(F,M)=\frac{1}{|\Omega|}\int_{\Omega}(F(x)-M(x))^2\mathrm{d}x \tag{5-2}$$

灰度归一化相关系数测度是在一个绝对的尺度范围内[−1,1]计算两幅图像的相关性，并给出相似度的一个线性的指标。令 $\overline{F}$ 与 $\overline{M}$ 分别为参考图像和浮动图像的像素灰度均值，则归一化相关系数测度可描述为：

$$\mathrm{CC}(F,M)=\frac{\mathrm{Cov}(F(x),M(x))}{\sqrt{\mathrm{Var}(F(x)\mathrm{Var}(M(x)))}}$$

$$\mathrm{Var}(I)=\frac{1}{|\Omega|}\int_{\Omega}(I(x)-\overline{I})^2\mathrm{d}x \tag{5-3}$$

$$\mathrm{Cov}(F,M)=\frac{1}{|\Omega|}\int_{\Omega}(F(x)-\overline{F})(M(x)-\overline{M})\mathrm{d}x$$

1. 多模态图像配准

图像体素相似性测度在多模图像配准方面拥有直观感觉的基础。假设图像非常相似，即计算对齐被研究患者影像中的结构所需的几何变换后，绝大多数的体素的差别只由噪声引起。可以证明上节所述的灰度均方误差测度只有在图像仅受到高斯噪声干扰时才是最优测度。而在多模配准中，这是永远不会出现的情况。一个略不严格的假设是图像灰度值间存在线性关系。这种情况下最优的相似性测度为上节提到的相关系数。一般地，体素间存在线性关系的假设对不同模态的图像也是不满足的，如图 5-2 所示。

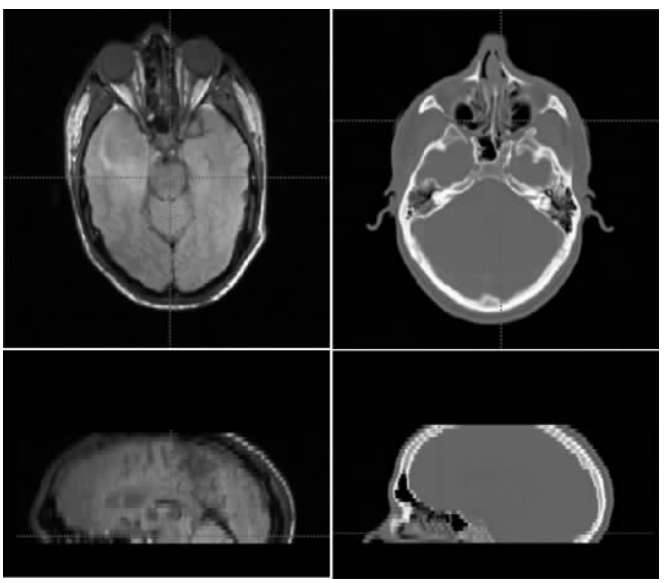


图 5-2 未配准的 MR(左)与 CT(右)图像示例

上面两幅是原始轴位图像，下面两幅是重建冠状面图像。注意，这些图像中对应的解剖组织灰度存在着巨大的差异。而且，虽然每次都针对颞骨扫描，CT 图像轴位视野比 MR 图像小得多。待配准图像视野差别是多模图像配准的难题之一。

互信息 (Mutual Information)[5]是信息论中的一个概念，通常用于描述两个系统间的统计相关性，或一个系统中包含的另一个系统中信息的多少，一般用熵(Entropy)来表示。

随机变量 A 的熵定义为

$$H(A)=-\sum_{a\in A}p_A(a)\log p_A(a) \tag{5-4}$$

两个随机变量 A、B 的联合熵定义为

$$H(A,B)=-\sum_{a\in A}\sum_{b\in B}p_{AB}(a,b)\log p_{AB}(a,b) \tag{5-5}$$

如果 A、B 互相独立，则

$$p_{AB}(a,b)=p_A(a)p_B(b) \tag{5-6}$$

$$H(A,B)=H(A)+H(B) \tag{5-7}$$

如果 A、B 不互相独立，则

$$H(A,B)<H(A)+H(B) \tag{5-8}$$

其差值称为 A、B 的互信息 $I(A,B)$

$$\begin{aligned} I(A,B) &= H(A)+H(B)-H(A,B) \\ &= \sum_{a\in A}\sum_{b\in B} p_{AB}(a,b)\log\frac{p_{AB}(a,b)}{p_A(a)\cdot p_B(b)} \end{aligned} \tag{5-9}$$

在医学图像配准中，待配准的两幅图像可能来自于不同的时间或不同的成像设备，但它们都基于共同的人体解剖信息，它们被看作两个随机变量时，二者显然不会相互独立，并且我们假设当它们的空间位置达到一致时，其互信息应为最大。这就是用互信息最大化作为相似性测度的原理所在。

将待配准的两幅图像看作二维随机变量 (A,B) 计算其互信息之前，首先需要估计该二维随机变量的边缘概率分布密度函数 $p_A(a)$、$p_B(b)$ 和联合概率分布密度函数 $p_{AB}(a,b)$。一个最直观的想法就是用图像的灰度级(联合)直方图来估计，即各灰度级出现的相对频率作为该灰度级像素的概率；常用的另一种估计方法是采用著名的 Parzen 窗概率密度估计(Parzen Window Density Estimate)$P^*(z)$：

$$p(z)\approx P^*(z)=\frac{1}{N_A}\sum_{z_i\in A}R(z-z_i) \tag{5-10}$$

其中，N_A 是图像 A 的像素个数(或者说随机变量 A 的样本数)；R 是窗函数，其定义域内积分等于 1。窗函数常采用高斯函数 $G_\Psi(z)$：

$$G_\Psi(z)\equiv(2\pi)^{-n/2}\mid\Psi\mid^{-1/2}\exp\left(-\frac{1}{2}z^T\Psi^{-1}z\right) \tag{5-11}$$

估计出边缘概率密度和联合概率密度函数后，可以很容易求出两图像的互信息。有文献指出待配准两图像重叠区的大小会影响互信息[5]，用规整化互信息 NMI(Normalized Mesure of Mutual Information)或熵相关系数 ECC 作为相似性测度函数能克服重叠区的影响，其中 NMI 和 ECC 计算公式如下。

$$\mathrm{NMI}(A,B)=\frac{H(A)+H(B)}{H(A,B)} \tag{5-12}$$

$$\mathrm{ECC}(A,B)=\frac{2I(A,B)}{H(A)+H(B)} \tag{5-13}$$

显然 NMI 和 ECC 有如下关系。

$$\mathrm{ECC}(A,B)=2-\frac{2}{\mathrm{NMI}(A,B)} \tag{5-14}$$

采用基于体素方法计算相似度时，对于高分辨率的三维体数据集，其包含的数据量极大，为减少计算量，常只采用部分或是随机采样数据点，而不是全部数据点。计算前先对原图像进行重采样。

2. 频域图像配准

频域图像配准方法利用了傅里叶变换多种良好的性质。平移变换、旋转变换、反演变换、分配律以及尺度变换在傅里叶域均有对应变换。而变换又可以用硬件或者快速傅里叶变换(FFT)高效实现。借助频域，傅里叶方法对于相关频率噪声有良好的鲁棒性。然而，这种方法只适用于刚性配准。以下先介绍使用傅里叶分析最基本的方法——相位相关，可以用于配准互相有平移的图像对。然后会介绍这种方法的扩展与几种相关的方法，这些方法能够配准平移加旋转的图像。

1975 年，Kuglin 与 Hines 提出了一种简易的用来配准互相有平移的图像对的方法，该方

法就是相位相关法。为了描述他们的方法，先定义几个需要的傅里叶分析中的术语。图像 $f(x,y)$ 的傅里叶变换是复函数；在频谱上每个频率 (ω_x,ω_y) 的函数值都有实部 $R(\omega_x,\omega_y)$ 与虚部 $I(\omega_x,\omega_y)$：

$$F(\omega_x,\omega_y)=R(\omega_x,\omega_y)+iI(\omega_x,\omega_y) \tag{5-15}$$

其中 $i=\sqrt{-1}$。也可以表示为指数形式

$$F(\omega_x,\omega_y)=|F(\omega_x,\omega_y)|\,e^{i\phi(\omega_x,\omega_y)} \tag{5-16}$$

其中 $|F(\omega_x,\omega_y)|$ 是傅里叶变换的幅值，$\phi(\omega_x,\omega_y)$ 是相角。图像中每个频率的功率等于幅值的平方

$$|F(\omega_x,\omega_y)|^2=R^2(\omega_x,\omega_y)+I^2(\omega_x,\omega_y) \tag{5-17}$$

相角表示每个频率对应的相移

$$\phi(\omega_x,\omega_y)=\arctan[I(\omega_x,\omega_y)/R(\omega_x,\omega_y)] \tag{5-18}$$

相位相关法根据傅里叶变换的平移性质，也称为移位定理。已知两幅仅相差位移 (d_x,d_y) 的图像，即

$$f_2(x,y)=f_1(x-d_x,y-d_y) \tag{5-19}$$

对应的傅里叶变换为

$$F_2(\omega_x,\omega_y)=e^{-j(\omega_x d_x+\omega_y d_y)}F_1(\omega_x,\omega_y) \tag{5-20}$$

换句话说，两幅图像幅值相同，相移与其位移直接相关。相差由 $e^{j(\phi_1-\phi_2)}$ 给出。如果按下式计算互功率谱

$$\frac{F_1(\omega_x,\omega_y)F_2^*(\omega_x,\omega_y)}{|F_1(\omega_x,\omega_y)F_2^*(\omega_x,\omega_y)|}=e^{(\omega_x d_x+\omega_y d_y)} \tag{5-21}$$

其中 F^* 为 F 的复共轭，移位定理确保互功率谱的相位与两幅图像之间的相移相等。此外，如果用空域形式表示互功率谱的相位，即对频域形式进行傅里叶逆变换，将会得到一个冲激函数，除了最佳配准位移处几乎处处为零。

所以使用频域方法对互相有平移的图像对的配准添上了检测互功率谱函数傅里叶逆变换峰值位置的步骤。因为各个频率分量相差相等，则窄带噪声，即仅在很小频率范围内分布，不会影响峰值的位置。因此这种方法尤其适合具有这种噪声的图像。于是，对于不同光照条件下获取的图像，该技术非常有效，因为光照条件变化缓慢，频谱上集中在低空间频率。类似地，这种方法具有一定程度的场景不变性，并且因为该方法对频率能量变化不敏感，该方法对于通过不同传感器采集的图像也很有用。这种在相关中只利用相位信息的性质有时也称做图像白化。与其他方法相比，白化对于亮度线性变化具有不变性，使相位相关对于场景变化相对独立。

另一方面，如果图像具有明显的白噪声，即覆盖所有频率的噪声，因为每个频率上的相差都受到了影响，峰值的位置将不准确。Kuglin 与 Hines 建议在进行傅里叶逆变换之前在相差中加入广义权重函数得到一系列相关方法，包括相位相关与传统互相关方法。按照这个思路，可以根据想要屏蔽的噪声选择适当的权重函数。

作为相位相关技术的扩展，De Castro 与 Morandi 提出一种可以配准同时存在位移与旋转变换的图像对的方法。由于旋转操作本身不存在位移，同理可推出将旋转表示为极坐标下的平移相位相关法。旋转与平移结合则表示一种更为复杂的变换。De Castro 与 Morandi 提出了以下分两步先求出旋转角度再计算平移位移量的方法。

旋转对于傅里叶变换具有不变性。图像进行旋转则其傅里叶变换旋转相同的角度。如果

角度已知，可对互功率谱旋转后使用相位相关法求出位移。然而，由于角度不已知，将功率谱相位看作旋转角估计值ϕ并用极坐标(r,θ)简化等式。得到函数

$$G(r,\theta;\phi)=\frac{F_1(r,\theta)F_2^*(r,\theta-\phi)}{|F_1(r,\theta)F_2^*(r,\theta-\phi)|} \tag{5-22}$$

该函数在实际旋转角度下应该具有只经过平移的图像对应的形式。因此，先求出是互功率谱相位的傅里叶逆变换最接近冲激函数的角度ϕ，然后即可根据冲激的位置求出位移。

实现上述方法的过程中，应当注意求经旋转变换后的图像的值是需要进行某种形式的插值，因为变换后图像上的点不在原图像离散网格上。虽然可以在第一次空域旋转后计算变换图像解决这个问题，但是会大大提高计算量。De Castro 与 Morandi 将变换应用于零填充(Zero-Padded)图像，提高了分辨率并改善了旋转后变换的近似。这种方法由于测试每个ϕ很麻烦，同样计算量很大。Alliney 与 Morandi 提出一种只需要一维傅里叶变换的方法来计算相位相关函数。使用图像在 x 轴与 y 轴上的投影，傅里叶变换可以由投影切片定理得到。x 投影与 y 投影的一维变换分别就是二维变换 $\omega_x=0$ 对应的行与 ω_y 对应的列。虽然得到了节省了一些计算量，但是除了对相对较小的位移外该方法不再具有鲁棒性。

作为一类算法，频域配准方法提供了噪声敏感度与计算复杂度方面的优点。Lee et al. 提出了一种使用图像倒频谱(对数功率谱的功率谱)功率的类似的方法用于早期检测青光眼的图像的配准。首先通过求出是功率谱相差最小的角度(如果只存在平移位移理论上应该为 0)使图像平行。然后用类似相位相关的方法确定平移变换。这种方法相比 De Castro 与 Morandide 的方法在节省计算量方面具有优势，因为它利用对数性质将两幅图像相加而不是相乘。两种方法都比经典的相关法更准确，更具有鲁棒性，计算量也更小。然而，由于傅里叶变换依赖于其不变性性质，频域方法只适用于某些明确界定的变换，如旋转与平移。如果存在大量空域局部变量(即使变换只是很小程度的刚性变换)，频域的方法也会失效。

3. 块匹配 (Block Matching)

通过对相似性测度进行优化来搜索全局变换的方法只有当定义变换的参数很少时实用，如刚性或仿射变换中。在非刚性变换中参数很多，一般都反映一个高度复杂的有很多局部最优点的相似性函数。B. Likar 与 F. Pernuš[51] 提出一种解决办法是先将待配准图像分割成子图像再应用局部刚性或仿射变换配准。通过近似所有的局部变换就可求出平滑连续的全局弹性变换。

B. Likar 与 F. Pernuš 在他们提出的能实现弹性配准的分层配准方案中使用了基于全图像的配准，即对一幅图像变形后，计算两幅待配准图像所有几何对应像素对的相似性。该方案包含 4 个主要的层(如图 5-3 所示)。每一层中，在上一层配准后的图像都分割为更小的子图像。然后通过改变仿射变换参数使归一化互信息最大化来配准每个子图像对。一旦所有的子图像都配准并且通过了三项一致性测试后，应用将已配准子图像看作点对得到的弹性薄板样条插值得到一个全局一致的配准。该方法详细细节，即互信息最大化、联合概率估计、局部配准一致性测试与弹性插值，在这里不再赘述，详见文献[51]。

S. Ourselin et al.[52] 不同于其他使用块匹配策略进行非刚性配准，提出一种刚性模式的块匹配算法。他们指出基于灰度的配准方法存在以下问题：第一，所有的相似性测度对于变换参数是高度非凸的，这导致进行全局优化几乎不会直接到达最优点；第二，两幅图像灰度间存在全局关系的假设可能会被多种图像伪影破坏。他们提出了块匹配策略结合鲁棒的变换估计方法解决这些问题。

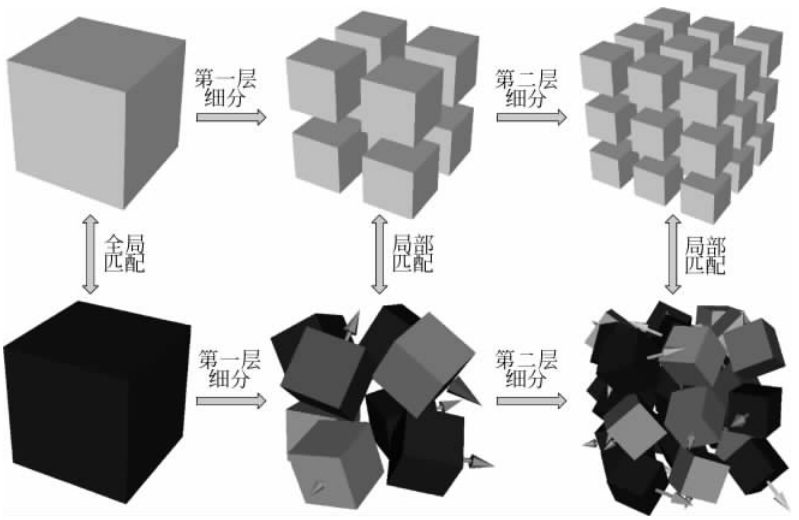


图 5-3　块匹配算法弹性配准方法

用该方法估计刚性变换时，主要问题在于所求的位移一般比非刚性的大得多。因此，需要将块在更大的范围内移动，这可能导致最后得到的位移场包括一些严重的离群点。这些离群点可能会严重影响刚性变换的估计。鉴于此，变换估计方法的鲁棒性至关重要。

算法将两幅图像作为输入：参考图像 I 与浮动图像 J。输出为变换 T 与对齐 I 的图像 $J'=JoT^{-1}$。完整的过程对两个连续的操作进行迭代。

(1) 使用块匹配策略计算 I 与当前浮动图像 J' 之间的位移；

(2) 使用鲁棒估计方法利用(1)求出的位移确定刚性变换 S。根据 $T\leftarrow SoT$ 更新当前变换，使用新的 T 对图像 J 进行一次重采样得到新的浮动图像 J'。

在迭代多尺度策略下应用这两步操作。

使用块匹配策略匹配(图 5-4)的基本原理为在邻域 Ω 中移动浮动图像块 $\mathcal{A}$ 并与参考图像中有对应位置的块 $\mathcal{B}$ 比较(如)。在一给定尺度级下，块大小与搜索邻域的大小是常数，分别表示为 $N_x\times N_y\times N_z$ 与 $\Omega_x\times\Omega_y\times\Omega_z$。对于给定相似性测度，最优的匹配块 $\mathcal{B}$ 可以在块 $\mathcal{A}$ 与 $\mathcal{B}$ 中心之间定义匹配对 (a_i,b_i)。为了改善计算时间，分别沿 x,y,z 方向在每个像素 Σ_x，Σ_y，Σ_z 上考虑暂时匹配来对搜索集 Ω 进行下采样。这些参数决定了位移场的分辨率。同样，测试块 $\mathcal{A}$ 中心也从浮动图像一个子块中提取。引入下采样因子 $(\Delta_x,\Delta_y,\Delta_z)$，该因子决定位移场的密度。

因为参数 (N,Ω,Σ,Δ) 为像素距离，所以恒为整数。考虑可能的图像各向异性，使用相对值以大致对应实数距离。

与基于全局准则的方法相反，块匹配改善确知的点对。然而，应该怀疑其中一些是不可靠的。当发生位移的块是均一的或包括参数图像中没有的结构时，坏匹配经常发生。为了处理离群点，先移除测度低于某给定阈值的配对。这样就检测出初始离群点集。在进行变换参数鲁棒估计时移除剩下的离群点。

当选择大搜索面积 Ω 时，我们希望算法能使用粗位移而当 Σ 与 Δ 取较小的值时进行精确配对。另外，块匹配过程的计算复杂度与 $(N\times\Omega)/(\Delta\times\Sigma)$ 成正比。为寻求性能与复杂度之间的平衡，采取多精度策略。开始时以粗精度为 N、Ω、Δ、Σ 设置大值。在细化精度时，所有参数

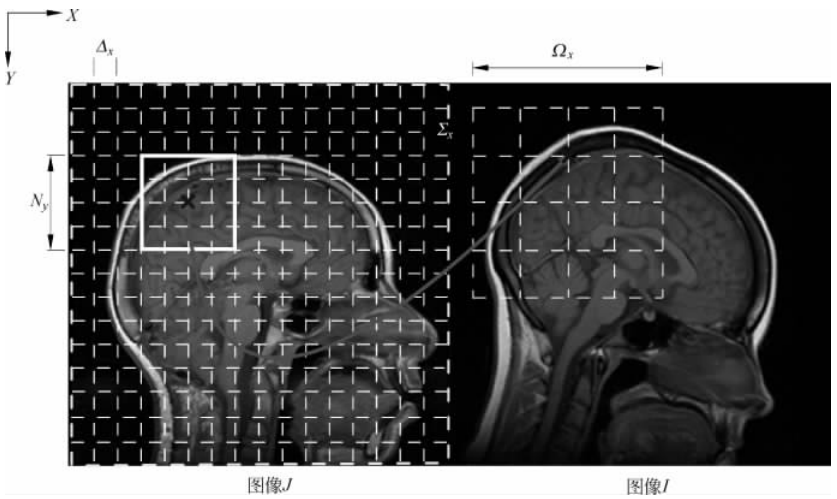


图 5-4 块匹配策略图示

在给定方向上，参数有下列意义：N—块大小；Ω—搜索集的大小；Σ—位移场的分辨率；Δ—位移场的密度

减半；按照这样的方法，计算复杂度与精度级无关。

块匹配步骤后使用最小绝对残差(Least Trimmed Squares，LTS)回归计算最适合位移场的刚性变换 T。已经证明这种最小化策略比经典最小二乘法(LS)对离群点鲁棒性好得多。它通过求解下列最小化问题实现。

$$\min_{T}\sum_{i=1}^{h}\| r_{1,n}\|^{2}$$

其中，$\| r_{1,n}\|^{2}$ 为留数的平方有序欧氏范数；$r_i=a_i-T(b_i)$；n 为位移向量总数；h 取 $h=\lfloor n/2\rfloor$ 以取得 50%的失效点。

LTS 衡准可以使用简单的迭代 LS 近似进行最小化。这种方法一般不超过 10 次迭代，由于一次 LS 最小化的解是解析的，具有可以非常高效的计算的优点。然而，算法作者指出应注意这种方法并不保证能够收敛到 LTS 衡准的全局最小值，有时只收敛到局部极小值。虽然他们在实验中没有观察到迭代 LS 陷入到异常解的情况，在以后的实现中还会研究更加鲁棒的 LTS 最小化策略。

4. 基于灰度图像配准的加速问题

O. Fluck et al. 对基于 GPU 灰度图像配准技术进行了总结[53]。医学影像数量上前所未有的增长与快速诊断的临床需求，推动了更加高效的图像配准方法的研究。而配准算法研究的两个前沿：一个是在头脑中构造出能高效计算的算法；另一个是尽量利用硬件的计算能力。在这里着重讨论后者。特别要讨论的是使用图像处理单元(GPU)的缩短配准时间的研究。被游戏产业与持续的对更复杂画面效果的需求所推动，图像硬件已经从一个固定功能流水线的简单接口演变为可编程的超级计算机。现在个人电脑上的 GPU 在峰值计算能力与存储器带宽方面已经远远超出了 CPU。自从引入可编程图形流水线后，就有一个活跃的研究社区致力于如何在图形硬件上实施科学(非图形)计算。浮点数据的非标准实现与双精度浮点数据型的却是曾经是数值算法的一大问题。但现在双精度数据格式已经开始出现，而且使用单精度浮点数据格式的模拟技术已经被应用于 GPU。一些计算甚至可以用只有 16 位的浮点数进行。因为 GPU 本身是用来从三维场景生成着色后的二维图像，从图像推导出变换参数的

逆问题并不能直接解决。这种配准方法需要自定义的并行化和在传统组件上修改。

像素灰度插值是图像配准中常见步骤。因为通常在图像所有像素上进行，在普通串行处理器上就变得很耗时。幸运的是，该操作是图形流水线的基本部分。因此，GPU 具有线性插值执行效率比为 CPU 写的代码更高效的特殊硬件。CPU 与 GPU 另一个重要差别在于访问存储器的灵活性。一直以来，对图形编程存储器随机访问不是很必要，主要进行本地化访问。因此一般对本地化存储器访问优化查找纹理缓存。更麻烦的是，缺乏对任意存储器位置（分散）的写操作。这就是需要对导入 GPU 的算法进行重设计的主要原因。另一个性能瓶颈经常由主存储器（内存）与图像存储器（显存）之间的数据传输引起。内存与显存间的吞吐率（Throughput Rate）相当的低。然而，一旦图像数据存储到显存后，配准方法即可利用 GPU 超高的带宽。

GPU 上的图形流水线一开始仅仅局限于允许高效硬件实现的有限配置下固定功能。通过高级编程语言中的“着色”，这些限制后来被去掉，在不同的阶段可编程性更好，使为每个游戏创造独特的图形风格变得更加可能。这使得研究者可以做出能在着色流水线不同阶段运行的通用算法。

不同的图形应用程序编程接口（Application Programming Interfaces，API）对应不同的高级编程语言。OpenGL 着色语言（GLSL）是 OpenGL 的高级着色语言，DirectX 对应的是 HLSL，而 Cg 编译器的输出在 OpenGL 或 DirectX 下都可以运行。图形 API 与着色语言在功能上是等价的。

另一方面，GPU 编程也有不面向图形的通用编程语言。一般地，这些 API 一样或更适合图像配准任务。Brook 是最早的通用语言之一，要求开发者严格遵循一个流处理模型（图 5-5 比较了流处理编程模型与顺序编程模型）。流处理是符合现代 GPU（或 CELL 处理器）架构的用于高度并行计算的编程模式。输入为相同类型数据的序列（流），然后被“核函数”逐元素处理。并行是外显的，即程序员需要定义它。Brook 并不是仅为图形 API 设计的，对于各种流架构都是适用的。对于 GPU，Brook 建立在图形 API 上，因此支持各种后端处理生成同时适用于 DirectX（或 CPU 参考代码）与 OpenGL 的代码。因为 Brook 隐藏了图形相关的代码，GPU 供应商开发了一种新的绕开图形编程的 GPU 流处理语言。CTM（Close to the Metal）是 ATI（现在的 AMD）开发的一种类似汇编的语言。CTM 下一代语言称为 CAL（Compute Abstract Layer）。开放的指令集架构（Instruction Set Architecture，ISA）可以直接使用或者

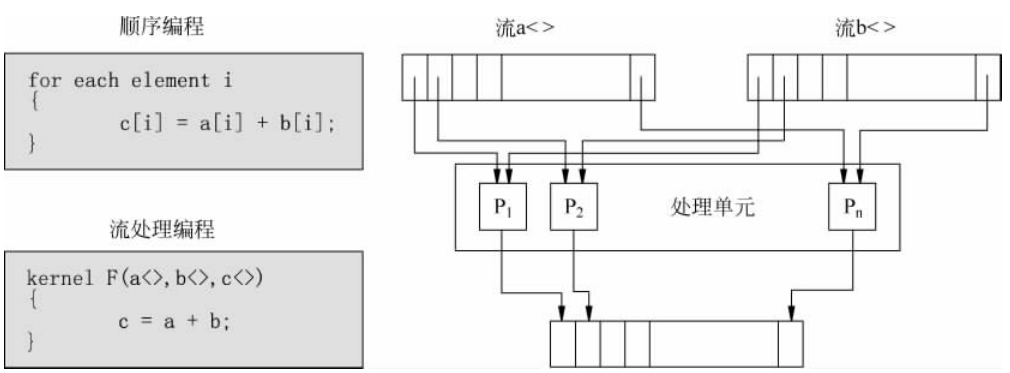


图 5-5 流编程模型

左图为顺序编程示例代码与流处理表示的比较。右图为流处理方案图示

作为编译器的输出。Brook + 是 Brook 将 CAL 开放的 ISA 作为输出的扩展。CUDA (Compute Unified Device Architecture)是 Nvidia 开发的 C 语言专有扩展并只在 Nvidia 的硬件上运行。OpenCL 是一种更通用的框架,不仅支持 GPU 也支持 CPU。DirectX API 中的一部分 DirectCompute 是另一种通用编程框架。图 5-6 为不同语言的概况。

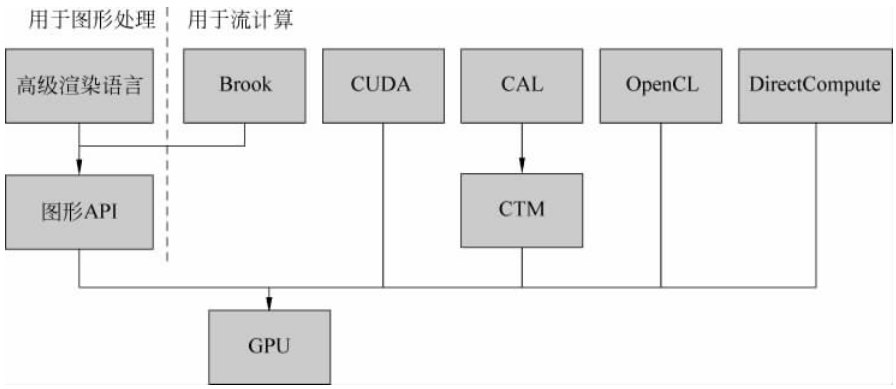


图 5-6 GPU 编程语言

图形 API 在 GPU 上部署更高级的着色语言(虚线左边)。GLSL 与 HLSL 分别为 OpenGL 与 Direct3D 的一部分,都可以使用 Cg。虚线右边是特别为 GPU 计算设计的语言,其中 Brook (该类第一种)通过 OpenGL 与 GPU 通信。

图形处理器针对处理当今视频游戏中的复杂的三维场景设计。图形流水线由高度可并行部件组成,所以设计 GPU 利用这些资源。图形流水线的输入是顶点——二维或三维图像中描述着色场景几何结构的点——与粘贴到顶点描述物体上的纹理(图像)。

在第一个可编程阶段顶点阶段,输入的顶点按照 GPU 上运行的顶点程序(顶点着色器)变形。构成线段、三角形甚至更复杂的多边形的顶点的连接只在下一个阶段中通过对几何着色器编程使用。该阶段中,对几何基元(以后均称做几何基元)进行变形并生成新的基元。然后对生成的基元进行光栅化。

光栅化阶段用片段填充基元。与像素相同,片段也对应与屏幕上特定位置,但不同的是如果片段所在基元部分是被另一个基元挡住,则没有必要绘制到屏幕上。然后生成的片段进入片段阶段,每个片段的外观由片段程序(或片段染色器)决定。在片段处理过程中,纹理可以粘贴到基元上。取决于缩放尺度,纹理的一个像素(纹素)可能精确地粘贴到光栅化基元的一个片段上。纹理和片段大小不一致的情况下,GPU 可以自动在纹理元素之间插值。支持的插值模式有最邻近插值法与线性插值法,而更复杂的插值策略可以使用染色程序实现。当允许混合时,可能结合若干片段计算一个像素的颜色。最后阶段,片段转化为实际像素。顶点、几何基元与片段数据以数据-并行的模式独立处理。图 5-7 表示了这个过程。

CUDA 是 Nvidia 开发的 C 语言的扩展。与 GPU 代码与 CPU 代码严格分离的面向图形的方法相比,CUDA 核函数可以包含在 GPU 代码与 CPU 代码混合的.cu 文件中。至于处理阶段,代码文件的内容分配给 CPU 或 GPU 不同的编译器。或者说,CUDA 提供了明确分离 GPU 代码与 CPU 代码的底层驱动接口。为了符合底层硬件,线程被分层次安排(见图 5-8)。CUDA 中的核函数对应于图形 API 中的着色器,片段上片段染色器的计算对应于 CUDA 上的一个线程(Thread)(见图 5-8)。

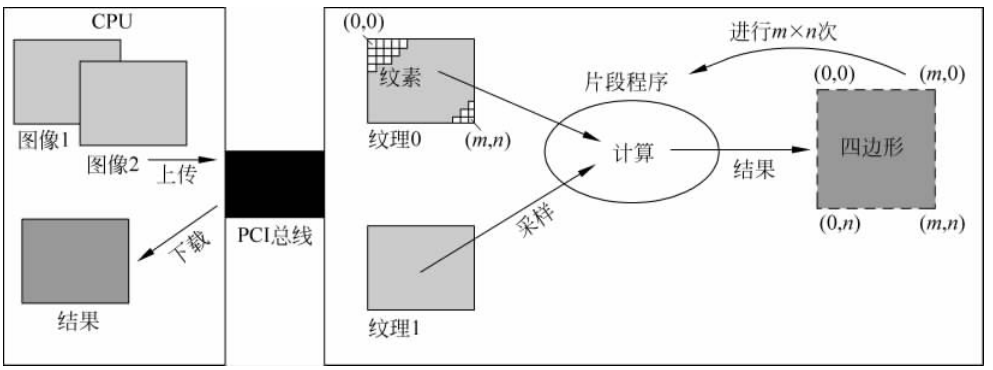


图 5-7　通过图形 API 初始化的 GPU 计算

基元由空间中的顶点定义。每个顶点调用一个顶点程序。基元投影在图像平面上的面积用像素衡量。输出缓存中每个像素调用一次载入的片段着色程序。

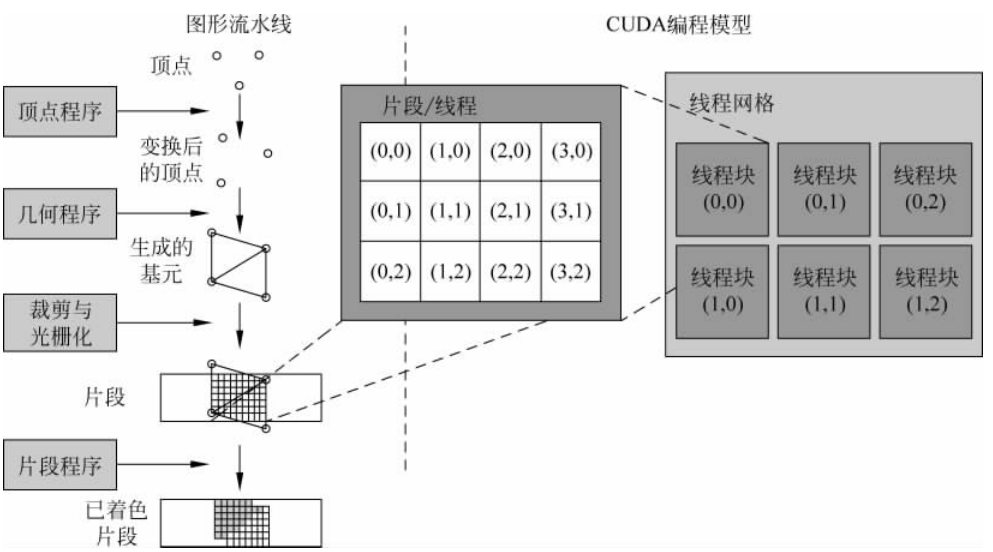


图 5-8　关于 CUDA 架构现代图形流水线的设计

使用 CUDA，程序员可以避免使用图形相关的代码。另外，当 GPU 若干多处理器上每个线程块运行时，块中的线程可以访问共享存储器。

OpenCL 是如 GPU 等流处理器的计算框架，但也与支持 CELL 架构一样支持多核 CPU。OpenCL 由苹果开发，现在由管理 OpenGL 的 Khronos Group 管理。概念上，OpenCL 与 CUDA 驱动接口非常类似而且当 OpenCL 应用于 Nvidia GPU 上时性能方面的考虑与 CUDA 是一样的。然而 OpenCL 的表示略有不同。OpenCL 中的一个工作项目就是 CUDA 中的一个线程，而处理元素在 CUDA 中叫做标量处理器。CUDA 线程组成多处理器中的线程块。OpenCL 中局部存储器在 CUDA 中叫做共享存储器，OpenCL 中私有存储器在 CUDA 中称为寄存器。对于 Nvidia 的 GPU，CUDA 是 OpenCL 与 DirectX Compute 的基础，DirectX Compute 是 DirectX 从 11 版本开始的计算接口。

与如 OpenGL 这种图形接口不同，CUDA 提供对有若干种存储器类型的显存的直接访问（见图 5-9）。能访问各种存储器对很多配准算法有好处。与线程 ID（例如 CUDA）或纹理坐标

(着色语言)无关的并行访问位置通常都是用这些方法。例如,存储器写操作的位置可能由执行核函数的像素灰度决定。这种情况其中一种常见例子就是基于直方图的互信息相似性测度。这些问题中,算法可以利用能在一组线程(即线程块)之间通信的共享存储器或全局存储器。而开发者必须注意并行线程对存储器访问冲突的可能性。

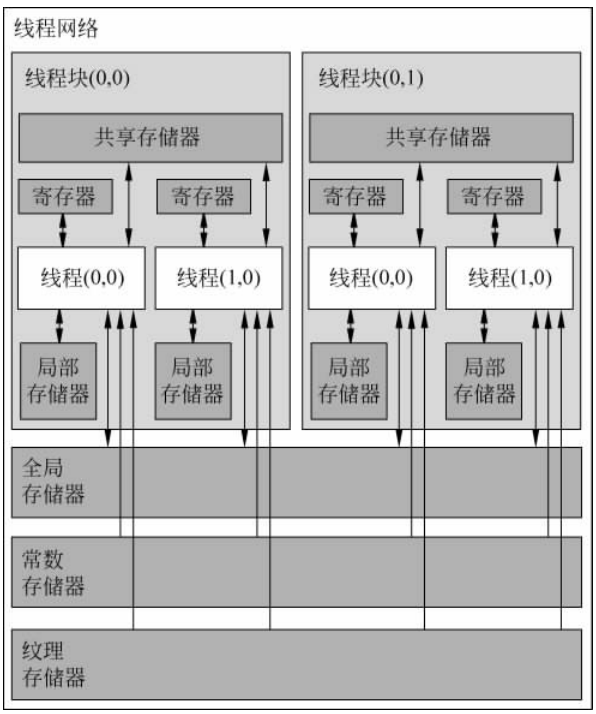


图 5-9 CUDA 一个线程网格概貌

线程组成线程块,线程块并行地运行在多处理器上。一个线程块中的线程都可以访问共享存储器,局部存储器与寄存器只能由拥有它的线程访问。

A. Kubias et al.[54]提出了一种在 GPU 上高效地进行刚性 2D/3D 图像配准的方法。其中算法的两个主要部分,生成数字重建放射图像(Digital Reconstruction Radiogragh,DRR)与相似性测度的计算,都在 GPU 上运行。如果 DRR 与 X 射线图像几乎一样,作者称 DRR 是拟真的或视觉相似的。通过提高 DRR 与 X 射线图像的相似度,配准得到简化。作者通过对当前的 X 射线 C 型臂系统的一些电子后处理建模提高相似度。生成 DRR 后,使用 GPU 并行计算 8 种基于灰度的 DRR 与 X 射线图像之间相似性测度。使用 8 种相似性测度的组合而不只使用一种是为了保证配准结果的精确度与鲁棒性。

2D/3D 图像配准中,术前体数据(如 CT 或 MRT)与术中 X 射线图像配准。在对体数据平移与旋转后,DRR 图像 $I_{\mathrm{DRR}}(x)$由当前变换 x 变换后体数据得到。DRR 由下面描述的方法生成。后来,DRR 图像 $I_{\mathrm{DRR}}(x)$与 X 射线图像 I_{FLL}用相似性测度 $S(I_{\mathrm{FLL}},I_{\mathrm{DRR}}(x))$比较。使用的 X 射线图像 I_{FLL}之前由 C 型臂系统得到。

使用特定的优化方法,DRR 与 X 射线图像之间的相似度根据式(5-23)提高直到找到最优参数向量 x^*。计算相似度的方法后面有描述。优化方法由全局优化器自适应随机搜索(Adaptive Random Search)与局部优化器最优邻域(Best Neighbor)组合而成。

$$x^* = \arg\max_x S(I_{\mathrm{FLL}},I_{\mathrm{DRR}}(x)) \tag{5-23}$$

以下先介绍生成拟真 DRR 的理论背景。

X 射线图像中，像素值取决于两个因素，探测器上 X 射线入射强度与电子后处理。

X 射线入射强度 J 取决于三个参数，即初始强度 J_0、X 射线源与探测器之间的距离与 X 射线通过的材料对 X 射线的衰减。衰减系数 μ_i 描述 X 射线在通过材料中衰减的百分比。例如，水的衰减系数 $\mu_{\text{Water}}=0.25\text{cm}^{-1}$。CT 锥束中，使用 CT 值 HU（Hounsfield Units）表示通过组织的衰减。CT 值通常取值范围为[−1000，7000]。例如，水的 CT 值为 0。CT 值可以用衰减系数通过下式计算

$$\text{HU}=1000\frac{\mu-\mu_{\text{Water}}}{\mu_{\text{Water}}} \tag{5-24}$$

如果 X 射线在衰减系数为 μ 的匀质物体中通过距离 d，初始强度 J_0 衰减为 J

$$J=J_0\text{e}^{-\mu d} \tag{5-25}$$

然而，人体并不是匀质物体，包含有若干种具有不同衰减系数 μ_i 的组织。另外，每种组织具有特定的厚度 d_i。衰减后强度 J 可以精确计算并近似为

$$J=J_0\text{e}^{-\int_0^d\mu\text{ds}}\approx J_0\text{e}^{-\sum_i\mu_i d_i} \tag{5-26}$$

如果体数据由等距离散体素构成并且每个体素大小相同，衰减系数 J 的计算可以简化为

$$J=J_0\text{e}^{-\sum_i\mu_i\Delta d_i} \tag{5-27}$$

X 射线到达探测器之后，可以在探测器每个点(x,y)上测量强度 $J(x,y)$。然后，通过执行一些 C 型臂系统电子后处理将测得的强度 $J(x,y)$转化为灰度值。因此，接下来作者对当前 C 型臂系统电子后处理进行了建模。首先，测得的强度 $J(x,y)$需要进行归一化，使得平均值对应某预期值 J_{desired}。其中，在 DRR 内层圆或者矩形 Ω 中，计算 Ω 中$|\Omega|$个灰度值的均值

$$\bar{J}=\frac{1}{|\Omega|}\sum_{(x,y)\in\Omega}J(x,y) \tag{5-28}$$

然后，计算出的均值 $\bar{J}$ 要缩放到预期值 J_{desired}。因此，每个灰度值 $J(x,y)$乘一个因子 J_{mult} 后加上一个常数 J_{add}

$$\begin{aligned}J'(x,y)&=\frac{J_{\text{desired}}}{\bar{J}}J(x,y)+J_{\text{add}}\\&=J_{\text{mult}}J(x,y)+J_{\text{add}}\end{aligned} \tag{5-29}$$

最后，得到的值插入到查找表 γ 中得到最终灰度值

$$J''(x,y)=\gamma(J'(x,y)) \tag{5-30}$$

在生成拟真 DRR 的实现中，部署了两个顶点着色器（结合两个简单的片段着色器）和另一个片段着色器。另外，作者还使用了帧缓存对象（Framebuffer Objects），从而能够直接对纹理进行渲染。

如上面理论背景提到，为了得到拟真 DRR 必须已知 J_{mult}、J_{add}与 γ。然而 J_{add}与 γ 由 X 射线 C 型臂系统规格决定，J_{mult}由当前 DRR 决定。因此，因子 J_{mult}必须在生成拟真 DRR 之前计算。然后，计算简单（非拟真）DRR，进一步计算均值 $\bar{J}$ 与因子 J_{mult}。为得到简单 DRR，去掉建模的电子后处理。根据接近最优参数向量 x^* 初始参数向量 x_0 生成简单 DRR。

生成简单 DRR 后，使用模板测试可以很容易计算出均值 $\bar{J}$。模板测试对存储有简单 DRR 的二维纹理内层圆或矩形 Ω 进行掩模处理。为了得到 Ω 的均值，使用 mipmap 可以高效地对掩模处理后的二维纹理进行下采样。要在纹理中求平均值这是一个很快很有技巧性的方法。然后，最小化的二维纹理下载到 CPU 上并计算最终的均值。乘数因子 J_{mult}可以用 $\bar{J}$ 按

式计算。

最后，片段着色器根据输入参数 J_{mult}、J_{add} 与带查找表 γ 的一维纹理生成拟真 DRR。生成拟真 DRR 的片段着色器输出含有拟真 DRR 的纹理和在另一个存储 DRR 的 ROI(感兴趣区域)的输出纹理。ROI 表示计算相似性测度时 DRR 被包含的部分。因为 ROI 与体数据的边界有关，自动确定 ROI 是很重要的。因此，计算相似性测度时不包括体数据之外的部分因为会干扰相似性测度值。

计算 DRR 与 X 射线图像之间的相似度时，选取一种相似性测度代入 $S(I_{\text{FLL}}, I_{\text{DRR}}(x))$。原作者使用基于灰度的相似性测度：SSD(Sum of Squared Differences)，

$$\text{SSD} = \sum_{(i,j)\in T} (I_{\text{FLL}}(i,j) - I_{\text{DRR}}^{T(i,j)})^2 \tag{5-31}$$

SAD(Sum of Absolute Differences)，

$$\text{SAD} = \sum_{(i,j)\in T} |I_{\text{FLL}}(i,j) - I_{\text{DRR}}^{T}(i,j)| \tag{5-32}$$

VOD(Variance of Differences)，

$$\text{VOD} = \frac{1}{1+\sigma^2} \tag{5-33}$$

其中

$$\sigma^2 = \frac{\sum_{(i,j)\in T} [(I_{\text{FLL}}(i,j) - I_{\text{DRR}}^{T}(i,y))]^2}{N^2 - 1}$$

RIU(Ratio of Image Uniformity)，

$$\text{RIU} = \frac{\sqrt{\frac{1}{N}\sum_{(i,j)\in T}(R(i,j)-\bar{R})^2}}{\bar{R}} \tag{5-34}$$

其中

$$R(i,j) = \frac{I_{\text{FLL}}|_{(i,j)\in T}}{I_{\text{DRR}}^{T}|_{(i,j)\in T}}, \quad \bar{R} = \frac{1}{N}\sum_{(i,j)\in T} R(i,j)$$

NCC(Normalized Cross Correlation)，

$$\text{NCC} = -\frac{\sum_{(i,j)\in T}(I_{\text{FLL}}(i,j) - \overline{I_{\text{FLL}}(i,j)}) \cdot (I_{\text{DRR}}(i,j) - \overline{I_{\text{DRR}}(i,j)})^{\text{T}}}{\sqrt{\sum_{(i,j)\in T}(I_{\text{FLL}}(i,j) - \overline{I_{\text{FLL}}(i,j)})^2} \cdot \sqrt{\sum_{(i,j)\in T}(I_{\text{DRR}}(i,j) - \overline{I_{\text{DRR}}(i,j)})^2}} \tag{5-35}$$

GDI(Gradient Difference)，

$$\text{GDI}(s) = \sum_{(i,j)\in T} \frac{A_v}{A_v + (I_{\text{diff}V}(i,j))^2} + \frac{A_h}{A_h + (I_{\text{diff}H}(i,j))^2} \tag{5-36}$$

其中 A_v、A_h 为常数，

$$I_{\text{diff}V} = \frac{dI_{\text{FLL}}}{di} - s\frac{dI_{\text{DRR}}}{di}, \quad I_{\text{diff}H} = \frac{dI_{\text{FLL}}}{dj} - s\frac{dI_{\text{DRR}}}{dj}$$

GC(Gradient Cross Correlation)，

$$\text{GC} = \frac{1}{2}\left[\text{NCC}\left(\frac{dI_{\text{FLL}}}{di}, \frac{dI_{\text{DRR}}}{di}\right) + \text{NCC}\left(\frac{dI_{\text{FLL}}}{dj}, \frac{dI_{\text{DRR}}}{dj}\right)\right] \tag{5-37}$$

与 PI(Pattern Intensity)，

$$\text{PI}_{r,\sigma}(s) = \sum_{(i,j)\in T}\sum_{d^2<r^2} \frac{\sigma^2}{\sigma^2 + (I_{\text{dif}}(i,j) - I_{\text{dif}}(v,w))^2} \tag{5-38}$$

其中 $d^2=(i-v)^2+(j-w)^2$，$I_{dif}(i,j)=I_{FLL}(i,j)-sI_{DRR}(i,j)$。下面介绍这些相似性测度如何在 GPU 上高效地计算。

除这 8 种相似性测度外，还计算上列 8 种相似性测度的平均值。该相似性测度称为 AESM(Average of Eight Similarity Measures)。计算 AESM 之前需要对这些相似性测度值进行融合，使得它们的值具有可比性。因此，每个相似性测度取值 0 到 1 之间，其中 0 表示没有相似性，1 表示最相似。然后计算 8 种相似性测度的平均值并作为一种新的测度使用。

尽管当今 GPU 的性能已经很好了，大多数相似性测度计算都在 CPU 上进行。为了在 GPU 上运行，必须适合 GPU 的流水线架构。随之会产生一些问题。由于流水线架构的缘故，着色程序不能改变输入纹理的值而且不能读取输出纹理的值。输出值数量上有所限制因为它依赖于附加到着色程序的输出纹理的数量。另外，着色程序中静态与动态指令的数量是受限的，所以复杂的计算必须分成几个着色程序进行。一些操作如求平均值在着色程序中不能高效地实现。因此，这些操作由一些工具如 mipmap 实现。

因为 GPU 存在上述限制，计算 8 种相似性测度的实现分成三步：第一步，计算后续运算中所需的若干平均值；第二步，计算不需要空域信息的简单基于灰度的相似性测度，即 SSD、SAD、VOD、NCC 与 RIU；最后一步，计算需要空域信息的较复杂的于灰度的相似性测度，即 GDI、GC 与 PI。

使用三个连续的片段着色器可以很容易地进行计算。片段着色器序列如图 5-10 所示。所有的片段着色器至少包括三个输入，即 DRR、X 射线图像与 ROI。三个输入分别存储在三个二维纹理中。而存储 DRR 与 ROI 的纹理已经在先前生成 DRR 时加载在 GPU 上。最后两个片段着色器还需要第一个片段着色器的计算结果。

第一个片段着色器并行计算不同均值所需的中间结果。为了后来区分不同的均值，这些中间结果存储在两个输出纹理的不同通道中。之后，使用 mipmap 对这些纹理高效地进行下采样。得到的小纹理下载到 CPU 上并进行最终均值的计算。这些均值也和其他均值一起作为后续片段着色器的输入。

$$S_{SSD}(I_{FLL},I_{DRR})=\frac{1}{|M|}\sum_{(i,j)\in M}(I_{FLL}(i,j)-I_{DRR}(i,j))^2 \tag{5-39}$$

第二个片段着色器并行计算 SSD、SAD、VOD、NCC 与 RIU 5 种相似性测度的中间结果。为了在 GPU 上计算，计算方式与在 CPU 上完全不同。例如，式(5-39)中相似性测度 SSD 在 CPU 上计算时在比较的图像上循环运行并对单次差异求和。整体求和后，再除以两幅图像重叠部分$|M|$中的像素数。而在 GPU 上，只需要在着色程序中计算每两个像素之间的差异。这些计算出的差异作为中间结果存储在输出纹理中。接下来求和与除法运算则使用 mipmap 求均值高效地执行。相似性测度的最终值在 CPU 上从这些下采样纹理中计算。下面是第二着色器代码的节选。

```
//从输入纹理中读入平均值
…
float4 vecMeanFLL = float4(value0,value0,value0,value0);
float4 vecMeanDRR = float4(valuel,valuel,valuel,valuel);
float4 vecMeanRIU = float4(value2,value2,value2,value2);
float4 vecMeanFLLMinusDRR = float4(value3,value3,value3,value3);
//读入 FLL、DRR 值，并计算差异
float4 fll = tex2D(texture0,coords);
```

```
float4 drr = tex2D(texturel,coords);
float4 DiffFLLMinusDRR = fll - drr;
//计算 SSD,SAD,RIU,VOD 的中间结果
float4 resultSSD = pow((DiffFLLMinusDRR),2);
float4 resultSAD = abs(DiffFLLMinusDRR);
float4 resultRIU = pow((fll + BIAS) / (drr + BIAS) - vecMeanRIU,2);
float4 resultVOD = pow((DiffFLLMinusDRR) - vecMeanFLLMinusDRR,2);
//中间结果写入 Output0
setOutput(output0,resultSSD,resultSAD,resultRIU,resultVOD);
```

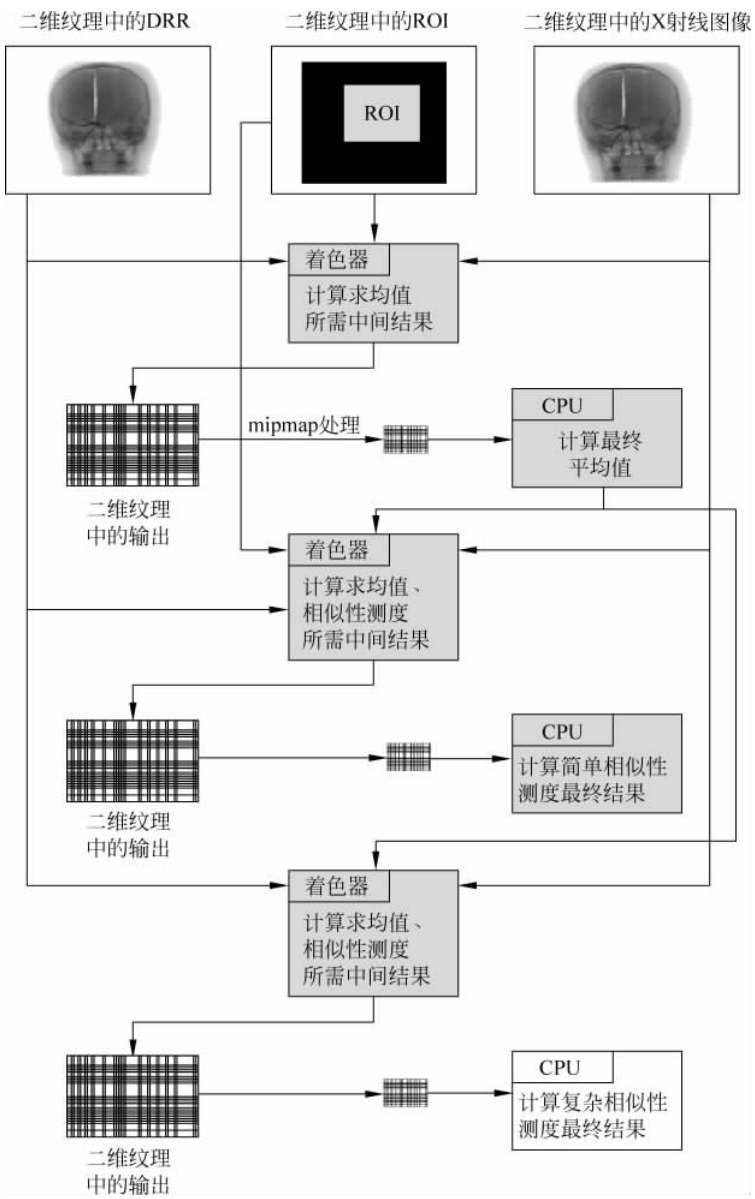


图 5-10 在 GPU 上计算相似性测度

为了计算 8 种相似性测度，连续运行三个片段着色器。对这三个片段着色器输出生成 mipmap 后，使用 CPU 计算最终结果。

类似地，第三片段着色器并行计算 GDI、GC 与 PI 三种基于灰度的相似性测度的中间结果。作者提供了表 5-1 说明加速效果。

表 5-1 GPU 算法与 CPU+GPU 混合算法的比较，CPU+GPU 混合算法生成 DRR 仍然采用 GPU 算法，但计算相似性测度在 CPU 上执行。两种算法中，都计算了 8 种相似性测度（即 AESM）。测量时间在 5000 次循环中平均

X 射线图像	CT 体数据	CPU+GPU 混合算法	GPU 算法	DRR	AESM	
				GPU	CPU	GPU
1024^2	1024^3	5065.6 ms	792.0 ms	757.7 ms	4307.9 ms	34.3 ms
512^2	512^3	1265.4 ms	235.6 ms	216.2 ms	1049.2 ms	19.4 ms
256^2	256^3	328.9 ms	88.6 ms	72.7 ms	256.2 ms	15.9 ms
128^2	128^3	74.6 ms	16.3 ms	11.4 ms	63.2 ms	4.9 ms
64^2	64^3	23.9 ms	8.4 ms	3.7 ms	30.2 ms	4.7 ms

伦敦大学学院开发的 Nifty Reg 配准平台包含进行刚性、仿射与非线性医学图像配准的程序。其中算法都包含两个版本，CPU 版本与 GPU 版本。刚性与仿射变换配准使用 Ourselin et al. 提出的算法[57,58]，非刚性配准基于 Modat et al. 的工作[55]。

Ourselin et al. 提出了一种基于块匹配与最小绝对残差的 Aladin 算法[57,58]。第一步，块匹配提供目标与源图像之间的匹配点对集。第二步，使用这个匹配点对集评估出最好的刚性或者仿射变换。对这两步循环直到收敛到最佳变换。Nifty Reg 的实现中，使用 NCC 提取目标块与源块之间的最优匹配。块的边长是固定的并设为 4 个体素。使用由粗到细的方法，即第一次在下采样图像（使用高斯滤波器对图像重采样）上进行配准，最后在全分辨率图像上进行。

非刚性图像配准算法实现基于 Rueckert et al.[18] 提出的自由变形法。然而，为了加速配准过程，对配准算法进行了重构。源图像的变形使用三次 B 样条产生形变场。具体地讲，就是在目标图像上定义等间距的控制点网格并且移动每个点使得可以改变到源图像的映射。为了评估两输入图像之间变形的质量，使用归一化互信息（NMI）与弯曲能量（BE）组成的一个目标函数。使用 NMI 与 BE 的解析上的导数利用共轭梯度法在对目标函数进行优化。

快速自由变形法使用 CUDA 实现。框架如图 5-11 所示。

第一步进行三次 B 样条插值与三线性插值确定新的体素位置与灰度。每个体素的位移与灰度插值的计算是独立的，所以它们并行硬件实现就是直接的。然而，计算需要动态存储资源，每个计算线程需要 22 个寄存器左右。因为 GPU 的存储器有限，每个线程需要更多的寄存器意味着并行的线程更少，使得不能充分利用设备的计算资源。活动线程与允许的最大值（随硬件改变）的比值称做占用率，而高效的实现方法应该使占用率最大化。每个核函数需要 22 个寄存器导致 42%的占用率。因此这一步分成了两个核函数，第一个只处理 B 样条插值，第二个只处理三线性插值。则寄存器需求分别为 16 与 12，占用率上升到 67%与 83%。这样的方法在这个问题中使计算时间减少了 36.8%。

第二步包括填充整个联合直方图与计算不同熵值。这一步 GPU 实现没有运算时间与串行执行相比表现出明显的降低。而且这一步只占总运算时间 2.2%左右。另外，GPU 实现需要使用单精度浮点数，会影响这一步的精度（原注：较新的显卡提供双精度，但是性能相当低下。编注：Nvidia 新的 Fermi 架构的双精度性能已经有了大幅提升，可以达到单精度的一半）。由于这些原因这一步在 CPU 上执行。尽管考虑设备与主机之间的数据传输对总的计算时间影响也不大。

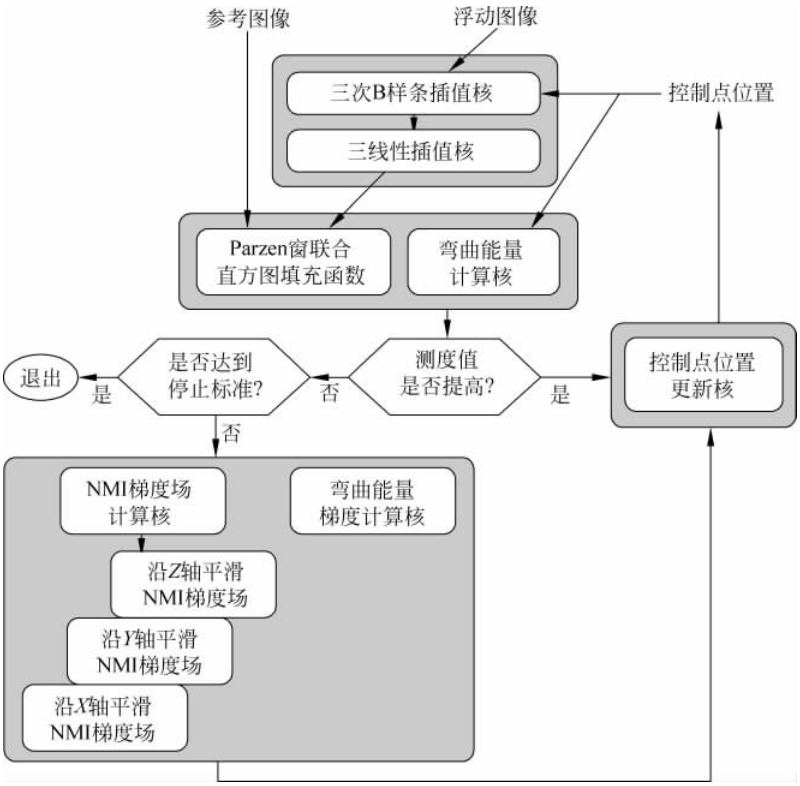


图 5-11 快速变形算法 GPU 实现框架

第三步中每个体素计算梯度值并且应用卷积窗函数。与第一步相同,将计算分散到若干个核函数以改进占用率。第一个核函数计算梯度值。然后使用三个核函数对梯度进行平滑,每个对应一个方向。经试验,这些核函数直接计算三次样条曲线比先计算再从内存中读取快。

最后一步,对梯度进行归一化并使用共轭梯度优化更新控制点。第一个核函数从整个梯度场中提取最大值。梯度场划分为若干部分并提取各部分最大值。最后保留极大值中最大的。最后由一个核函数根据归一化的梯度值更新控制点的位置。

他们的实现最后一个特点就是收敛标准的选取,之前的实现都是只进行不多的几次迭代,而他们的算法则一直迭代到收敛为止,已得到较好的配准结果。

作者使用 Nvidia 8800 GTX GPU,包括 128 个处理器与 768MB 的显存。显存的大小限制不允许读入控制点间距 Δ 比较小的非常大的图像集。最后作者在每个轴向 $\delta=2.5$ 体素的 2563 个体素的图像上进行了测试。这些规格对处理核磁大脑图像是可以接受的。另外,最近发布的一些显卡的显存已有所扩大,如已到 6GB。

表 5-2 所示为 Nifty 算法实现中每个函数并行 GPU 计算与串行 CPU 相比速度提升。

表 5-2 Nifty 算法实现中每个函数并行 GPU 计算与串行 CPU 相比速度提升

函 数	速 度 提 升	函 数	速 度 提 升
计算变形场	11.27×	计算弯曲能量	9.13×
三线性重采样	12.83×	计算 NMI 场	26.30×
三次样条重采样	10.12×	计算弯曲能量梯度	13.33×

表 5-3 为一篇高性能配准综述文献[59]中对其提到的部分 GPU 硬件加速算法实现的归纳和性能数据对比。

表 5-3 一些基于硬件加速的图像配准算法统计与对比

变换		测度	优化器	硬件	年	性能	注释	小组
线性	刚性	SSD	Simplex	GeForce 6800	2006	98.0	使用 OpenGL 与 GLSL 编写	Köhn
	刚性	SSD	梯度下降(B)	GeForce 6800	2006	858	使用 OpenGL 与 GLSL 编写	Köhn
	刚性	GC	?	Quadro FX 1400,FX 3400,GTX 7800	2006	—	2D/3D 配准	Ino
	刚性	多种	自定义	GeForce 6800 GT	2006	—	多种测度(如 SSD′CC′GC)	Khamene
	刚性	多种	ARS+BN	GeForce 6800 GS	2008	—	2D/3D,多种测度(如 SSD′CC′GC),自适应随机搜索+最优邻域	Kubias
	刚性	MI	Simplex	GTX 8800(16 个多处理器/128 核)	2007	6.17	CUDA 1.0(不支持三维纹理),用采样估计 MI	Shams
	刚性	SSD	Simplex	GTX 8800(16 个多处理器/128 核)	2008	6.05	CUDA 2.0	Plishker
	仿射	MI	梯度下降(A)	GTX 8800(16 个多处理器/128 核)	2008	—	用采样估计 MI	Lin
	刚性	MI	Powell	GTX 280(30 个多处理器/240 核)	2009	4.06	CUDA 2.0	Shams
非线性	Bezier 曲线	MI	Powell	GeForce3 64 MB	2002	—		Soza
	非参数	SSD	梯度下降	GeForce FX 5800 Ultra	2004	—	2D/2D,使用了多重网格求解器	Strzodka
	非参数	SSD	梯度下降(B)	GeForce 6800	2006	465	使用 OpenGL 与 GLSL 编写	Köhn
	非参数	MI+KL	梯度下降(C)	GTX 7800	2007	2860	结合 MI 与 Kullback-Leiber 测度,使用 OpenGL 与 GLSL 编写	Vetter
	非参数	MI+KL	梯度下降(C)	GTX 8800 Ultra(16 个多处理器/128 核)	2008	324		Fan
	非参数	SSD	迭代	Quadro FX 1400	2007	1050	使用 Cg 编写,2008 年发表	Courty
	非参数	SSD	迭代	GTS 8800(12 个多处理器/96 核)	2007	11.7	使用 Brook 编写,计算性能未包含 SSD	Sharp
	非参数	CC	迭代	Quadro FX 5600(16 个多处理器/128 核)	2008	9.25	CUDA 0.9	Özçelik
	非参数	SSD	梯度下降(C)	GTX 8800(16 个多处理器/128 核)	2008	3710	CUDA 0.2	Plishker
	非参数	MI	穷尽法	Quadro FX 5600(16 个多处理器/128 核)	2009	—	2D/2D,超大二维图像	Ruiz

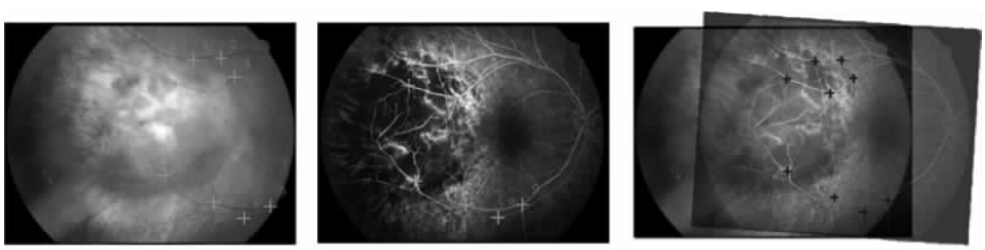
注：性能为以毫秒每次迭代每百万体素(ms/MVoxel/itr)为单位的归一化性能。

5.2.2 基于特征的图像配准

基于特征(Feature-Based)的图像配准,通常也可称为基于点集(Point Set-Based)的配准,是一类利用从图像中提取的特征进行配准的方法。这类方法常用于三维模型表面重建、图像导航等领域。在基于特征的配准过程中,图像由一些离散的特征所组成的集合来描述,配准算法通过找到特征间的对应关系,进而直接估计出待配准图像间的几何变换参数。用数学语言来描述:假设浮动图像的特征集为 $G=\{g_i\}$,参考图像的特征集为 $G=\{f_i\}$,特征间的对应关系集为 $C=\{(f_i,g_i)\}$,基于特征的配准过程就是要找到关系集 C 和几何变换参数 Θ,使得特征间距离测度函数式(5-40)最小二乘误差达到最小。

$$E(\Theta;C)=\sum_{(f_k,g_k)\in C}[D(T(g_k;\Theta),f_k)]^2 \tag{5-40}$$

图 5-12 为基于标记点的视网膜图像配准与融合。



(a) 参考图像 (b) 浮动图像 (c) 配准并融合后的图像

图 5-12 基于标记点的视网膜图像配准与融合

基于特征的配准本质上是求解一个非线性最小二乘问题。在配准过程中,算法主要面临着两大问题。

(1) 特征点的匹配问题;

(2) 几何变换模型的参数估计问题。

相对于基于图像内容(Area-Based/Intensity-Based)的配准方法,基于特征配准方法在以下几个方面具有优势。

(1) 图像特征集在数量上要比图像像素集小很多,从而配准过程的计算量很少;

(2) 图像特征是经过选择的,其精度和敏感度都要强于图像像素,因此配准的精度较高;

(3) 图像特征在一定程度上可以避免图像纹理噪声的影响,因此算法的鲁棒性较好。

根据特征选择和特征匹配方法的不同所衍生出的具体配准方法是多种多样的,但它们一般都包括了几个必要的环节[3]。

(1) 图像预处理(Preprocessing):图像预处理环节主要为特征的选择和匹配做准备,使用的方法通常包括图像尺度校正、图像去噪和图像分割等;

(2) 特征选择(Feature Detection):在特征选择环节,算法从参考图像和浮动图像上提取边缘、线段、表观等特征,特征提取过程可手动交互或自动完成;

(3) 特征匹配(Feature Matching):在特征匹配环节,算法根据特征集间的相似性测度,建立参考图像特征与浮动图像特征间的一一对应关系;

(4) 几何模型参数估计(Transform Model Estimation):几何模型参数估计环节将根据图像特征间的对应关系,得到参考图像与浮动图像间的像素映射关系;

(5) 图像重采样(Image Resampling)：在图像重采样环节，算法根据由步骤(4)得到图像映射关系，对浮动图像进行几何变换并在坐标格上进行图像插值。

特征的选择与提取是基于特征的图像配准算法中一个重要的环节。为了便于特征的匹配和几何变换模型参数的估计，用于配准的特征应当是明显的、独特的、易于提取的并且在配准图像上有较好的分布特性。目前可以用于图像配准的特征种类很多，包括点、直线、曲线、表面、区域等。由于点特征易于描述，且其空间坐标可以直接用来计算几何变换模型的参数，因此点特征是图像配准中最为常用一类特征。目前，流行的点特征提取算法主要有：角点提取(Harris，SUSAN)和 SIFT 点特征提取。

特征的匹配是基于特征配准算法的核心问题和难点。在特征匹配过程中，算法需要找到特征点间的对应关系(Correspondences)。在图像配准应用中，通常需要一一对应的特征匹配关系。特征匹配在计算机视觉中被分为宽基线匹配(Wide Baseline Correspondence)和窄基线匹配(Short Baseline Correspondence)。前者对应于大视场改变、图像内容变化较大的情况；后者对应于成像环境变化不大、图像内容交叠较多的情形。目前常用的点特征匹配方法有：基于邻域的互相关方法、Hausdorff 距离法、松弛标定法、迭代最近邻方法(ICP)、最大期望方法(EM)等。

1. 随机抽样一致性策略

随机抽样一致性模式 RANSAC(RANdom SAmple Consensus)由 Martin A. Fishler 与 Robert C. Bolles[38]早在 1981 年提出。作为一种数据拟合、模型参数估计模式，RANSAC 具有广泛的应用。这里仅简单介绍它在提取特征的匹配问题方面的应用。与之前的使用尽可能多的数据取得初始解然后尝试消除不可靠的数据点的模式不同，RANSAC 使用尽可能少的初始数据并在可能的时候加入一致的数据。

RANSAC 模式数学表达：

给定一个最少需要 n 个数据点来确定其自由参数的模型和一个数据点集 P 且 P 中点的个数大于 n。任意从 P 中选取具有 n 个点的子集 $S1$ 并初始化模型参数，并用该模型 $M1$ 决定从点集 P 中确定处于 $M1$ 一定误差范围内的数据点子集 $S1^*$。$S1^*$ 称为 $S1$ 的一致性集合。

如果 $S1^*$ 中元素的个数大于某个阈值 t(t 为 P 中严重误差点个数估计值的函数)，则使用 $S1^*$ 计算(可能使用均方法)一个新的模型 $M1^*$。

如果 $S1^*$ 中元素的个数小于 t，随机选取新的子集 $S2$ 并重复上述过程。如果经过预先给定次数尝试后，仍未找到一个元素数大于等于 t 的一致性集合，则用找到的最大的一致性集合求解模型或者以失败结束。

2. 弹性配准分层次属性匹配机制(HAMMER)

HAMMER 算法(Hierarchical Attribute Matching Mechanism for Elastic Registration)是 D. G. Shen 与 C. Davatzikos[17]提出的一种应用于脑核磁共振影像的弹性配准算法。实验证明结果在不同患者的重合影像具有高重合度。该算法不同于其他基于最大化图像相似性的体积变形方法，被称做 HAMMER 的主要创新点在于：首先使用一种属性向量，即定义在图像每个体素上并从组织图中计算出的几何矩不变量(Geometric Moment Invariants，GMIs)的集合，来反映不同尺度上内在的解剖结构。如果属性向量足够丰富的话即可区分图像中的不同部分，有助于在变形过程中建立解剖对应关系。并且通过减少潜在匹配对的混淆度，也有助于减少局域极小值。然后为了避免陷入局部极小值，即次优化的劣匹配，HAMMER 使用能量函数的逐次近似。高维能量函数被较低维平滑能量函数优化，较低维能量函数构造时即明

显含有较少的局部极小值点。这种构造方法通过选取具有区分度高的属性向量作为主导特征实现，然后就会大幅度地减少寻找匹配对的混淆度。

HAMMER 使用一系列低维能量函数以最终逼近如下多变量能量函数：

$$E=\sum_{x\in V_T}\omega_T(x)\left(\frac{\sum_{z\in n(x)}\varepsilon(z)(1-m(a_T(z),a_s(h(z))))}{\sum_{z\in n(x)}\varepsilon(z)}\right)$$
$$+\sum_{y\in V_s}\omega_s(y)\left(\frac{\sum_{z\in n(y)}\varepsilon(z)(1-m(a_T(h^{-1}(z)),a_s(z)))}{\sum_{z\in n(y)}\varepsilon(z)}\right)+\beta\sum_x\|\nabla^2u(x)\| \tag{5-41}$$

第一项是模板空间里相似匹配之和 f，以计算从模板到对象的变换 $h(\cdot)$。对每个模板体素 x，在其球形邻域 $n(x)$计算属性向量相似度。对于一个相邻的模板体素 z，将其属性向量与其对象中对应变形体素 $h(z)$的属性向量比较。$a_T(z)$是模板体素 z 的属性向量，而 $a_S(h(z))$为 $h(z)$处对象体素的属性向量。函数 $m(\cdot,\cdot)$衡量两个属性向量之间的相似度，取值从 0 到 1。因此 $1-m(\cdot,\cdot)$衡量两个属性向量之间的差异。$\varepsilon(\cdot)$是权值，因为边界体素是重要的解剖特征并且在有足够对比度的图像中很容易找到，所以为边界体素分配较大权值。$\sum_{z\in n(x)}\varepsilon(z)$ 项用来归一化。加权参数 $\omega_T(x)$为模板图像中每个体素 x 分配相对权值。给具有大区分度属性向量的体素分配较大权值，这些体素通常相对其他体素可以更稳定地被识别出来。这些体素将作为由模板到对象变形的初始主导点。通过层次化地集中处理能量函数中各项(由主到次)，该模型可以分层次地将模板向对象变形。

第二项与第一项相似，但是定义在对象域，并且用来约束有对象到模板的逆变换($h^{-1}(\cdot)$)。因为权值 $\omega_T(x)$、$\omega_S(y)$由图像细节决定，而图像细节在模板图像和对象图像中不同，这两项侧重的方面不同。第一项侧重于将“模板向对象”变形，而第二项反之。这个构造使得 HAMMER 对于初始化与次优解具有鲁棒性。当然，引入这项会增加计算量。为了克服这一点，选择性地只关心一小部分称为主导体素的对象体素，即具有显著属性向量的体素。虽然不能保证完全三维变换的一致性，但是保证了驱动力的一致性。因为驱动力能很有效地决定两图像间的对应关系并且在体积内其他位置可以简单地进行插值，这种方法经实验证明是有效的。

第三项是位移场的平滑约束。其中，∇^2为拉普拉斯(Laplacian)算子。此项与广泛应用的归一项相似。参数 β控制形变场的平滑度。而大的 β可能会导致形变场的过度平滑，在此应用中 β取 0.5。

属性向量是 HAMMER 至关重要的组成部分。对每个体素 x 定义属性向量 $a(x)$，由三个向量组成，即

$$a(x)=[a_1(x)\quad a_2(x)\quad a_3(x)]$$

$a_1(x)$是边界类型向量，若采取硬组织分割方法，则为 1×1 的向量，可取 7 个离散值，对应 7 种边界类型，即非边界和灰质(GM)、白质(WM)与脑脊液(CSF)之间 6 种组合边界；若采取模糊分割，每个体素分类结果为向量$[C_{GM},C_{WM},C_{CSF}]^T$，其中 C_{GM}、C_{WM}与 C_{CSF}取 0 到 1.0 之间的实数，且满足 $C_{GM}+C_{WM}+C_{CSF}=1.0$，则 $a_1(x)$为 3×1 向量，如可设为体素 x 与其相邻体素

z 的组织分类结果的差，其中 z 为使该差向量幅值最大的相邻体素。

$a_2(x)$为 1×1 向量(但一般也可有多个分量)，表示 x 的灰度，取值范围即为灰度值范围。为方便起见，$a_2(x)$归一化到 0 和 1 之间。

$a_3(x)$包括各组织在不同尺度上的 GMI。以硬分类举例，组织被标记为 GM、WM 与 CSF 三种。每个尺度每种组织，可从三维零阶、二阶、三阶标准矩计算出 13 种旋转不变量。因此，$a_3(x)$为 $3\times13\times L$ 的向量，其中 L 为采用的尺度个数。以下为 GMI 的数学定义。为方便起见，假设坐标原点已经平移到考虑的体素 x，而该坐标系下隶属函数(GM、WM 或者为 CSF)为 $f_{\text{tissue}}(x_1,x_2,x_3)$，其中 x_1、x_2、x_3 为体素坐标。则函数 $f_{\text{tissue}}(x_1,x_2,x_3)$的三维$(p+q+r)$阶标准矩定义为

$$M_{p,q,r}=\iiint_{(x_1)^2+(x_2)^2+(x_3)^2<R^2} x_1^p x_2^q x_3^r f_{\text{tissue}}(x_1,x_2,x_3)\,\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3 \tag{5-42}$$

其中 R 为原点(即体素 x)周围球形邻域的半径。下面列出四个由零阶和二阶矩构造的旋转不变量，另外 9 个，由三阶与二阶矩构成，因为结构过于复杂在此略去，详见文献[39]。

$$I_1=M_{0,0,0}$$

$$I_2=M_{2,0,0}+M_{0,2,0}+M_{0,0,2}$$

$$I_3=M_{2,0,0}M_{0,2,0}+M_{2,0,0}M_{0,2,0}+M_{2,0,0}M_{0,2,0}-M_{1,0,1}^2-M_{1,1,0}^2-M_{0,1,1}^2$$

$$I_4=M_{2,0,0}M_{0,2,0}M_{0,0,2}-M_{0,2,0}M_{1,0,1}^2-M_{0,0,2}M_{1,1,0}^2-M_{2,0,0}M_{0,1,1}^2+2M_{1,0,1}M_{1,1,0}M_{0,1,1}$$

定义两个体素 x 与 y 的相似度为

$$m(a(x),a(y))=\begin{cases}0, & \text{if } a_1(x)\neq a_1(y)\\ c([a_2(x)\quad a_3(x)],[a_2(y)\quad a_3(y)]), & \text{otherwise}\end{cases} \tag{5-43}$$

其中 $c([a_2(x)\quad a_3(x)],[a_2(y)\quad a_3(y)])$是属性向量第二与第三部分的相似度。如果有训练集，则 $c([a_2(x)\quad a_3(x)],[a_2(y)\quad a_3(y)])$可以支持向量机等学习方法用训练样本进行训练。没有训练集时，则简单地将 a_2 与 a_3 中每个元素归一化至 0 到 1 之间，然后定义 $c([a_2(x)\quad a_3(x)],[a_2(y)\quad a_3(y)])$为

$$c([a_2(x)\quad a_3(x)],[a_2(y)\quad a_3(y)])=(1-|a_2(x)-a_2(y)|)\times\prod_{i=1}^{K}(1-|a_3^i(x)-a_3^i(y)|) \tag{5-44}$$

其中 $a_3^i(x)$为总共有 K 个元素的 $a_3(x)$的第 i 个元素。

具体算法步骤如下。

(1) 使用全局仿射变化将对象图像变换至模板空间，仿射变换矩阵由对象的标准矩确定。

(2) 在模板和对象图像每个体素上计算属性向量。因为已经将组织标记为 GM、WM 与 CSF，可以对每个体素计算出局部结构的属性向量。注意每个组织可计算出 13 种 GMI。但实际情况中酌情选取使用。

(3) 在对象中根据属性向量确定主导边界体素集$\{y_i\,|\,1\leqslant i\leqslant N_S\}$。

(4) 分层次在模板中选取主导边界体素集$\{x_i\,|\,1\leqslant i\leqslant N_T\}$，用来主导模板的变形。初始模板主导边界体素集只包括区分度大的模板体素，如脑沟的根部与脑回的顶部。在后期迭代中，其他的边界体素如脑室壁加入到该集合。最终阶段，体素集包括模板中所有的体素。

(5) 对每个对象主导体素 y_i，在其邻域(半径为 D_S 的球体)内搜索属性向量最相似的偏移的模板主导体素 $h(x_j)$。如果对象主导体素 y_i 与模板主导体素 x_j 之间的相似度在阈值 t_{Voxels}

(初始 t_{Voxels} 较高,随时间逐渐降低)以上,则在形变模板主导体素 $h(x_j)$ 沿从 $h(x_j)$ 到 y_i 方向施加力。

(6) 对每个模板驱动体素 x_i,在其邻域(半径为 D_T 的球体)内搜索所有具有相似属性向量的对象体素(寻找的初始相似度较高,随着模型的收敛逐渐降低)。然后试验性地将 x_i 所在的区域向各个已找到的相似对象体素变形,并在区域内综合所有体素的相似度。如果对象体素的最大相似度大于一个特定的阈值(t_{Voxels}),该区域最终会变形到具有最大相似度的对象体素。另外,如果存在由对象主导体素向模板主导体素 x_i 施加的力((5)中确定的),则该区域还会受这些力发生形变。

(7) 分层次使用由模板主导体素计算出的局域与全局仿射变换改进位移场。

(8) 使用式(5-41)中的第三能量项描述的平滑约束改进位移场。

(9) 如果形变过程收敛,则停止。否则,到步骤(4)。

3. 基于图像局部特征描述子的配准

21 世纪初,出现了一批基于图像局部特征描述子的配准方法,如 SIFT[40]、RIFT[41]、MSER[42]、DAISY[43]等。由于其鲁棒性与灵活性,这些新兴的算法得到广泛的应用,同时基于图像局部特征描述子的配准方法也成为了研究热点。

1) 尺度不变特征描述子(Scale-Invariant Feature Transform,SIFT)

该描述子由 D. G. Lowe 提出。由于对图像缩放、旋转具有不变性,并对亮度变化与三维视角变化具有部分不变性,降低由遮挡、背景杂波或者噪声引起描述子失效的概率,可以从图像中高效地提取大量的区分度高的描述子等优点,该描述子具有广泛的应用,受该算子思想启发很多种其他特征描述子不断出现。

SIFT 算法如下 4 大步骤。

- 检测尺度-空间的极值:算法第一阶段是在全部尺度和空间位置上搜索极值。采用了很有效率的高斯差分(Difference of Gaussians)算子近似尺度归一化的高斯对数算子。
- 精确特征点定位:在每个候选特征点处,采用了精确的模型来拟合各个特征点的位置和尺度,并通过稳定性量度筛选特征点。
- 为特征点分配特征方向:根据局部图像梯度方向为每个特征位置分配一个或多个图像。
- 计算局部图像描述子:在每个特征点特定尺度上周围区域中计算局部图像梯度。这些梯度再经过变换成为允许明显程度的局部形状畸变以及照射条件变化的描述子。

(1) 检测尺度-空间的极值

可以证明高斯差分算子是尺度归一化的高斯对数算子:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{k\sigma - \sigma}$$

所以

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{5-45}$$

所以检测尺度-空间极值时采用高效计算简便的高斯差分算子来代替高斯对数算子是可行的。

将尺度空间每个尺度层级分割为 s 间隔。令 $k=2^{\frac{1}{s}}$，则每个尺度层级中需计算 $s+3$ 张高斯平滑($k^i\sigma$)图像，最终在整个尺度层级中搜索局部极值。图 5-13 为尺度层级示意图。

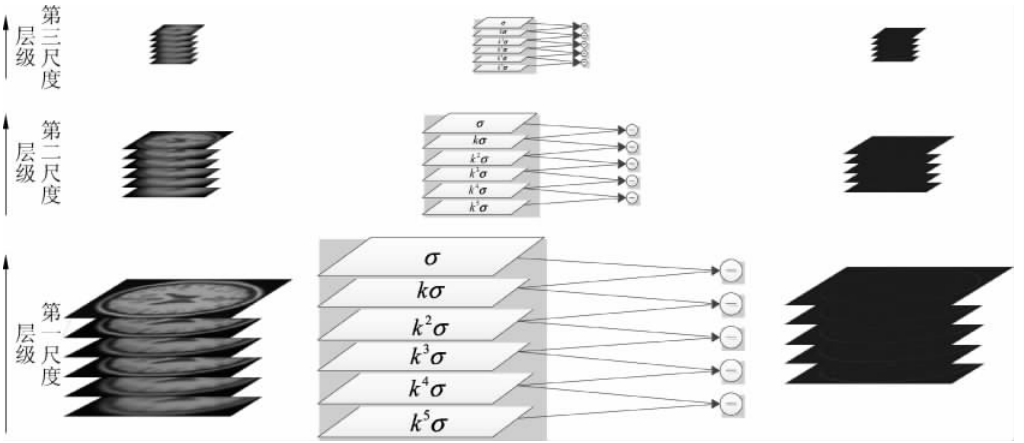


图 5-13　尺度层级示意

① 极值的检测

每个样本点和同一图像中 8 个邻接点以及同一尺度层级相邻模糊度图像中各 9 个邻接点进行比较，取比较结果是都大于或都小于的样本。

② 尺度取样频率

经过实验，取三个不同的尺度能兼顾特征的个数和匹配重复率。

③ 空域取样频率

经过实验，$\sigma=1.6$ 时最合适。

(2) 特征点精确定位

① 尺度-空间函数的泰勒展开

$$\begin{cases} D(x) = D + \dfrac{\partial D^{\mathrm{T}}}{\partial x}x + \dfrac{1}{2}x^{\mathrm{T}}\dfrac{\partial^2 D}{\partial x^2}x \\ \hat{x} = -\dfrac{\partial^2 D^{-1}}{\partial x^2}\dfrac{\partial D}{\partial x} \\ D(\hat{x}) = D + \dfrac{1}{2}\dfrac{\partial D^{\mathrm{T}}}{x}\hat{x} \end{cases} \tag{5-46}$$

其中

$$\frac{\partial D}{\partial x} = [D_x \quad D_y \quad D_s]^{\mathrm{T}}$$

$$\frac{\partial^2 D}{\partial x^2} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xs} \\ D_{yx} & D_{yy} & D_{ys} \\ D_{sx} & D_{sy} & D_{ss} \end{bmatrix}$$

② 去除边界效应

一个峰值偏差的高斯差分函数会在边界的法向产生很大的主曲率，而在其正交方向很小。主曲率可以通过 Hessian 矩阵 H 计算，

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{5-47}$$

主曲率与 H 的特征值 α、β 呈正比。由定理 $\mathrm{Tr}(H)=\alpha+\beta$，$\mathrm{Det}(H)=\alpha\beta$，又

$$\mathrm{Tr}(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$\mathrm{Det}(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

令 $\alpha=r\beta$，有

$$\frac{\mathrm{Tr}^2(H)}{\mathrm{Det}(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2\alpha}{r} \tag{5-48}$$

取阈值 $r_0=10$，通过判断 $\dfrac{\mathrm{Tr}^2(H)}{\mathrm{Det}(H)}=\dfrac{(r_0+1)^2}{r_0}$ 去除主曲率之比大于 10 的特征点。

(3) 分配特征方向

特征点的尺度用来选择计算特征方向时所选用的高斯平滑图像 L，选取最邻近的尺度。使用如下公式预先计算特征点周围区域梯度幅值 m 和辐角 θ，

$$m(x,y) = \sqrt{[L(x+1,y)-L(x-1,y)]^2+[L(x,y+1)-L(x,y-1)]^2} \tag{5-49}$$

$$\theta(x,y) = \arctan\{[L(x+1,y)-L(x-1,y)]^2/[L(x,y+1)-L(x,y-1)]\} \tag{5-50}$$

构造一个方向直方图。将 360°分为 36 个样条即每 10°一个样条。每个样条将其梯度幅值与一个 σ 为特征点尺度 1.5 倍的高斯圆形窗的乘积作为权重加入直方图。最后直方图的峰值即代表特征点的主方向，如果不止一个峰，则加入高度为主峰 80%以上的其他峰。每个峰在峰值处及左右取三个样条做二次拟合，以得到更精确的峰值。

(4) 计算局部图像描述子

选择特征点的尺度对应的高斯平滑图像来对特征点周围的梯度幅值和方向进行取样。用一个 σ 为描述子窗宽一半的高斯权重函数为每一个样本点的幅值分配权重。对 4×4 样本区域构造直方图，每个区域按 0°、45°、90°、135°、180°、225°、270°、315°分为 8 个样条，用三线性插值将每个梯度样本的值分配到相邻的直方图样条中。最后每个特征点描述子为长度 4×4×8=128 的向量。图 5-14 为特征点描述子示意图。

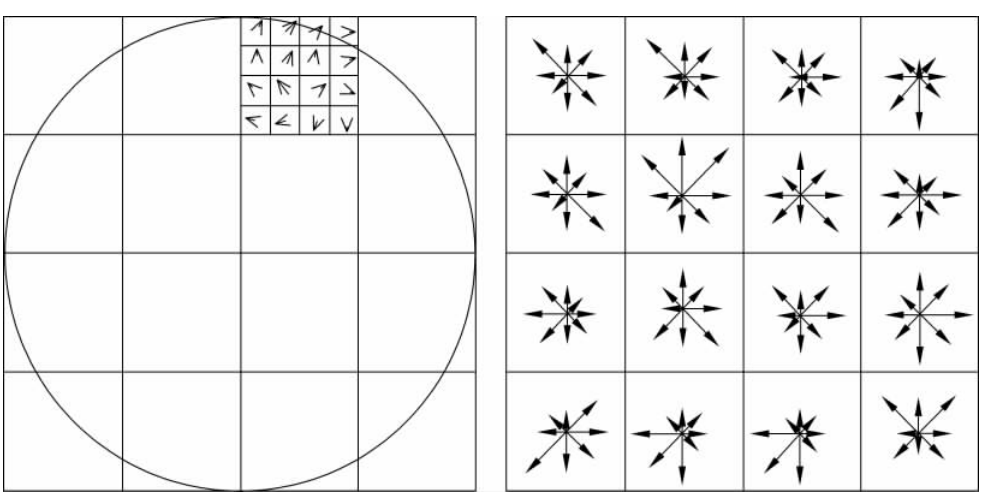


图 5-14 特征点描述子示意

按如上所述的方法提取特征后即可使用欧氏距离寻找最邻近特征来进行匹配，然后可进行进一步处理(计算仿射变换参数、识别等)。

2) 旋转不变特征描述子(Rotation-Invariant Feature Transform，RIFT)

S. Lazebnik et al.[41]提出一种适用于识别包括视点改变与非刚性变形的大范围变换的纹理表面图像的纹理表示方法。在特征提取阶段，先找到仿射 Harris 与拉普拉斯区域的离散集合。每个区域被看作具有特征椭圆形与特殊外观模式的纹理元素。该模式通过形状归一化与计算两种新型的描述子，spin images 与 RIFT 描述子，具有仿射不变性。当不需要仿射不变性时，原先的椭圆形可作为另外一种用来识别纹理的区分特征。

(1) 检测局部仿射区域

该方法使用一种适用于仿射的拉普拉斯色块检测器[45]。虽然文献[42,46,47,48]中有一些其他的仿射与尺度不变检测器，该方法中选择拉普拉斯检测器因为该检测器一般检测到的区域都是远离物体边界感觉比较显著的色块状区域。检测器确定尺度空间中归一化拉普拉斯算子取局部极大值的位置然后进行仿射适应过程[45,47]。得到的椭圆区域可以通过映射到单位圆进行归一化。然而，因为单位圆具有旋转与翻转不变性，归一化操作具有内在的正交模糊(Othogonal Ambiguity)。该算法通过用旋转不变描述子表示每个归一化块的表观解决模糊问题。

(2) 描述子

作者使用了依赖不同种类图像信息的两种互补的描述子：一种是基于归一化灰度的 spin images[49]，一种是基于梯度方向的 RIFT 描述子。

(3) 灰度域 spin images

灰度域 spin images 是一种对某特定参考(中心)点邻域内灰度值分布进行编码的二维直方图。直方图两个维度为到中心点的距离 d 与灰度值 i。对应固定的 d 的 spin images“切片”就是距离中心 d 的灰度值直方图。因为参数 d 和 i 在图像邻域正交变换下具有不变性，spin images 为表示仿射归一化块提供了合适程度的不变性。作者进行的实验距离划分为 10 个样条(Bin)，灰度划分为 10 个样条，得到 100 维的描述子。

作者实现的 spin images 为“软直方图”，其中统计区域里每个像素分配到多个样条中。特别地，在 x 处对坐标为(d,i)的样条贡献定义如下

$$\exp\left(-\frac{(|x-x_0|-d)^2}{2\alpha^2}-\frac{|I(x)-i|^2}{2\beta^2}\right) \tag{5-51}$$

其中，x_0 为中心像素位置；α 与 β 是表示二维直方图样条“软宽度”的参数。注意，软直方图可以看作从灰度值 I 与距离 d 联合密度的 Parzen 估计(使用高斯窗)的一组采样。

要使图像灰度函数具有仿射变换(即 $I\rightarrow aI+b$ 形式的变换)不变性，对 spin images 的统计范围内灰度值范围进行归一化即可。为了降低潜在的对噪声归一化的敏感性和减少重采样伪影(这些问题对于只有几个像素宽的块尤其严重)，该算法在计算 spin images 之前用高斯核略微平滑了归一化的块。

(4) RIFT 描述子

原作者推广了 Lowe 的 SIFT[40]提出了一种旋转不变描述子以取得归一化块局部表观的互补表示。原来的 SIFT 在检索任务方面有卓越的性能，然而，由于它依赖于归一化块的主方向，在此问题中并不适用。旋转不变特征变换(Rotation-Invariant Feature Transform)，RIFT 构造如下：圆形归一化块分成等宽的同心圆环并在每个环内计算梯度方向直方图。为保持旋

转不变性，每个点的方向角度都相对由中心指向外的方向计算。原作者用了 4 个环 8 个直方图方向，得到 32 维描述子。

(5) 讨论

spin images 与 RIFT 描述子遵循了原始 SIFT 与其他特征如 shape contexts[50]同样的构造策略：将统计区域分割成子区域并在各子区域中计算表观属性(最常用的是像素值与梯度方向)。直方图提供了对图像模式变形的稳定性，分割统计区域则抵消了潜在的空域信息的丢失。这样就达到了更大几何不变性与更强的区分度之间矛盾条件的平衡。直觉上，与在整个统计区域上计算函数传统描述子如滤波器组和差分不变子相比，基于这种平衡策略的描述子会更多同时更鲁棒看起来似乎有道理。也有实验文献[44]证明了这一点，同时在原作者早期研究阶段发现 spin images 相较于旋转不变的“类 Gabor”线性滤波器取得较好的结果。作者在后文对 spin images 与 RIFT 也进行了比较评估，spin images 性能稍优于 RIFT(可能由于 spin images 具有更高维数)。然而，在同一的识别框架中结合这两种描述子能得到比单独使用任何一种更好的结果。这也许反映了 spin images 与 RIFT 依赖于互补的图像信息——前者使用归一化灰度值而后者使用梯度。

3) 最大稳定极值区域(Maximally Stable Extremal Region，MSER)

J. Matas et al.[42]提出的 MSER 算法是与 SIFT 完全不同的算法。采用一种类似于形态学分水岭的方法从图像中提取最稳定极值区域作为特征。特征描述子原作者并未详细描述，只是提到使用一种区分性区域的互相关矩阵对角化与基于高阶矩的一种旋转不变描述子结合的仿射不变描述子。

图像 I：映射 $I:\mathcal{D}\subset\mathbb{Z}^2\to\mathcal{S}$。如果满足以下条件则可以明确定义极值区域：

(1) $\mathcal{S}$为全有序集，即具有自反性、反对称性与传递性的二元关系$\leqslant$存在；

(2) 定义了邻接关系 $A\subset\mathcal{D}\times\mathcal{D}$。如果使用 4-邻域，定义为对任意 $p,q\in\mathcal{Q}$当且仅当 $\sum_{i=1}^{d}|p_i-q_i|\leqslant 1$，$p,q$ 为邻点(pAq)。

区域$\mathcal{Q}$：$\mathcal{D}$的连续子集，即对任意 $p,q\in\mathcal{Q}$存在序列 $p,a_1,a_2,\cdots,a_n,q$ 并且满足 pAa_1，a_iAa_{i+1}，a_nAq。

区域(外)边界：$\partial\mathcal{Q}=\{q\in D\mid\mathcal{Q}:\exists\, p\in\mathcal{Q}:qAp\}$，即$\mathcal{Q}$的边界$\partial\mathcal{Q}$为与$\mathcal{Q}$至少一个像素相邻但是不属于$\mathcal{Q}$的像素的集合。

极值区域：对于区域$\mathcal{Q}$，$p\in\mathcal{Q}$，$q\in\partial\mathcal{Q}$，$I(p)>I(q)$(极大值区域)或 $I(p)<I(q)$(极小值区域)。

最稳定极值区域：令$\mathcal{Q}_1,\cdots,\mathcal{Q}_{i-1},\mathcal{Q}_i,\cdots$是一系列嵌套的极值区域，即$\mathcal{Q}_i\subset\mathcal{Q}_{i+1}$。极值区域$\mathcal{Q}_{i*}$是最稳定的当且仅当 $q(i)=|\mathcal{Q}_{i+\Delta}-\mathcal{Q}_{i-\Delta}|/|\mathcal{Q}_i|$在 $i*$ 处为极小值。其中，$|\cdot|$为集合的基数，Δ 为算法参数。

实现时即可采用依次增加或减少灰度阈值提取嵌套极大值区域或极小值区域序列，并检查 q 值，得到极小值时即得到一个最稳定极值区域。

MSER 算法的主要优点有：

(1) 具有仿射不变性；

(2) 对图像域内保粘切(连续)变换具有协变性；

(3) 稳定性；

(4) 多尺度检测；

(5) 极值区域枚举算法复杂度为 $O(n \log\log n)$(几乎为线性)。

4) DAISY 描述子

Tola et al.[43]提出的 DAISY 算子受到 SIFT 与 GLOH 算法(GLOH 算法与 SIFT 类似,只不过计算方向直方图使用极坐标)的启发,在长基线匹配有较好的性能,同时,作者在构造描述子时注意提高计算效率,使描述子计算效率大大降低。

DAISY 描述子的定义如下。

首先计算 H 幅方向图,G_i,$1 \leqslant i \leqslant H$,每个量化方向各一幅,其中如果$(u,v)$处图像在方向 o 梯度模值大于零,则 $G_o(u,v)$取值即为该模值,否则取 0。这样构造保留了灰度变化的极性。方向图记作 $G_o = \left(\frac{\partial I}{\partial o}\right)^+$,其中 I 是输入图像,o 是导数方向,$(\cdot)^+$定义为$(a)^+ = \max(a,0)$。

然后每个方向图与不同 Σ 值的高斯核卷积,得到用于不同面积区域的卷积后方向图 $G_o^{\Sigma} = G_{\Sigma} * \left(\frac{\partial I}{\partial o}\right)^+$,其中 G_{Σ} 为高斯核。使用不同的 Σ 值控制区域面积大小。

由于卷积运算使用可分离的高斯滤波器实现尤其表现出高效率,作者设计下面步骤初衷是降低运算量。另外,因为大高斯核卷积可以从几个连续较小高斯核卷积得到,我们可以低成本地计算不同大小的方向图。特别地,已知 $G_o^{\Sigma_1}$,按如下方法可以高效地计算 $G_0^{\Sigma_2}$($\Sigma_2 > \Sigma_1$)。

$$G_o^{\Sigma_2} = G_{\Sigma_2} * \left(\frac{\partial I}{\partial o}\right)^+ = G_{\Sigma} * G_{\Sigma_1} * \left(\frac{\partial I}{\partial o}\right)^+ = G_{\Sigma} * G_o^{\Sigma_1} \tag{5-52}$$

其中 $\Sigma = \sqrt{\Sigma_2^2 - \Sigma_1^2}$。

如图 5-15 所示,在每个像素位置,DAISY 为由从同心圆上卷积后方向图中得到的值组成的向量,在各卷积后方向图中高斯平滑度与圆的半径成正比。正如从图中看到的,这种构造使得描述子看起来像一朵花,描述子也由此得名。

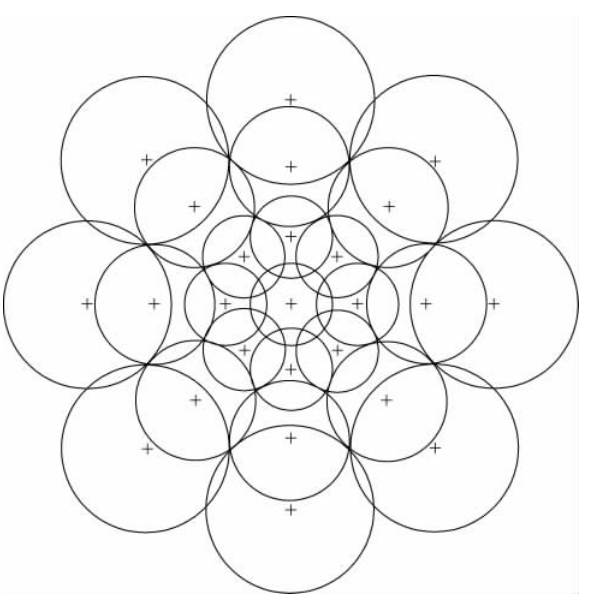


图 5-15 DAISY 描述子[43]

每个圆代表半径和高斯核标准差成正比的区域。“+”号表示对卷积后方向图采样的位置,每层圆的圆心都在以计算描述子的像素位置为圆心的同心圆上。区域互相重叠使得各区域间平滑过渡并且具有一定的旋转鲁棒性。半径逐层增加是为了在旋转轴上均匀采样,这对于旋转不变性是很必要的。

令 $h_\Sigma(u,v)$为标准差为 Σ 的高斯核卷积后方向图在(u,v)处的值组成的向量。

$$h_\Sigma(u,v) = [G_1^\Sigma(u,v),\cdots,G_H^\Sigma(u,v)]^T \tag{5-53}$$

其中 $G_1^\Sigma(u,v),G_2^\Sigma(u,v),\cdots,G_H^\Sigma(u,v)$表示不同方向上 Σ-卷积后方向图。将这些向量归一化为单位矢,记作$\tilde{h}_\Sigma(u,v)$。对每个直方图独立实施归一化使得正确表示障碍物附近的像素。如果对描述子进行整体归一化,从不同视角观察时靠近障碍的点的描述子会有较大的变化。

作者指出分别归一化直方图在均质区域中应用会出现问题。然而,考虑到作者设计描述子时考虑的是立体图像应用,即使在最坏情况的场景中,DAISY 性能依然高于标准基于区域的测度如归一化相关系数(NCC)。因为 DAISY 描述子相对来说比较大,上述问题不会经常发生。另外,使用全局优化算法往往也可修复导致的错误。如果确实需要大面积区域的鲁棒性,一种解决方案为先计算未归一的描述子然后在计算相异性之前对描述子可见部分进行全局归一化。然而在匹配阶段需要对每个像素每个可能深度进行两次额外的归一化会导致计算时间的增加。对于立体图像以外的应用,归一化的方法应该根据具体应用修改归一化方法,原作者未做进一步讨论。

如果 Q 表示不同环状结构的层数,(u_0,v_0)处完整的 DAISY 描述子$\mathcal{D}(u_0,v_0)$定义为向量的联结:

$$\mathcal{D}(u_0,v_0) = \begin{bmatrix} \tilde{h}T_{\Sigma_1}(u_0,v_0),\tilde{h}T_{\Sigma_1}(l_1(u_0,v_0,R_1)),\cdots,\tilde{h}T_{\Sigma_1}(l_T(u_0,v_0,R_1)), \\ \tilde{h}T_{\Sigma_2}(l_1(u_0,v_0,R_2)),\cdots,\tilde{h}T_{\Sigma_2}(l_T(u_0,v_0,R_2)), \\ \vdots \\ \tilde{h}T_{\Sigma_Q}(l_1(u_0,v_0,R_Q)),\cdots,\tilde{h}T_{\Sigma_Q}(l_T(u_0,v_0,R_Q)) \end{bmatrix}^T \tag{5-54}$$

其中为方向被量化为表 1 中 T 个值,由 j 决定的方向上到点(u,v)距离为 R 的位置。

表 5.4 为 DAISY 描述子参数。

表 5-4 DAISY 描述子参数

参 数 名 称	符 号	描述与默认值
半径	R	从中心点到最外网格点的距离(15)
半径量化值	Q	不同 Σ 卷积后方向图层数(3)
角度量化值	T	单层直方图个数(8)
直方图量化值	H	每个直方图样条数(8)
网格点序号	S	描述子中使用的直方图数$=Q*T+1$
描述子长度	$\mathcal{D}_s$	描述子向量总大小$=S*H$

DAISY 采用极坐标网格替代 SIFT 的常规直角网格,有文献[44]表明,极坐标网格有更好的分布特性。这一点,相较于 SIFT,DAISY 与不使用主元分析(Principle Component Analysis,PCA)的 GLOH 更接近。将各向同性的高斯核与极坐标网格结合也使 DAISY 自然而然地不受旋转扰动的影响。互相重叠的区域保证描述子沿着旋转轴平滑地变化,并且通过增大重叠面积,可以使描述子更加鲁棒直至失去区分度。

如上所述,圆形设计与使用各向同性核的一个重要优点在于当需要计算不同方向上的描述子时,不需要重新计算卷积后方向图,它们依然有效,只需简单地旋转采样网格重新计算描述子。然后直方图也需要循环移位来反映相对梯度方向的变化,而这仅仅是很小的开销。

之前描述高斯核的方差与描述子区域的面积成正比。准确的定义如下。

$$\sigma_i = \frac{R(i+1)}{2Q} \tag{5-55}$$

其中 i 表示极坐标网格的第 i 层(见图 5-15)。极坐标表示的直方图位置如下。

$$r_i = \frac{R(i+1)}{Q} \tag{5-56}$$

$$\theta_j = \frac{2\pi j}{T} \tag{5-57}$$

5.2.3 刚性与仿射配准

刚性配准与仿射配准是医学图像配准方法中最基本的两类情况。这两种配准形式主要由其所采用的几何变换模型决定。刚性变换也称为欧式变换,在几何空间中它保证了图像像素(特征点)间的相对位置不变,是一种简单而可靠的几何形变模型。仿射变换是线性变换与平移变换的复合变换,它在刚性变换的基础上加入了全局缩放和错切变换的因素。在仿射变换过程中,平行的直线依然被映射为平行的直线。刚性变换与仿射变换都可以用一个齐次矩阵描述,在三维空间中,这类变换可表示为 4×4 方阵。

对于大多数医学图像的几何形变问题,刚性约束是一类非常好的近似;同时其所需要计算的参数也比较少,因此刚性配准在临床医学图像配准中有着非常广泛的应用,常被用于人脑结构图像与功能图像的配准。

图 5-16 给出了几种三维旋转变换的示例。

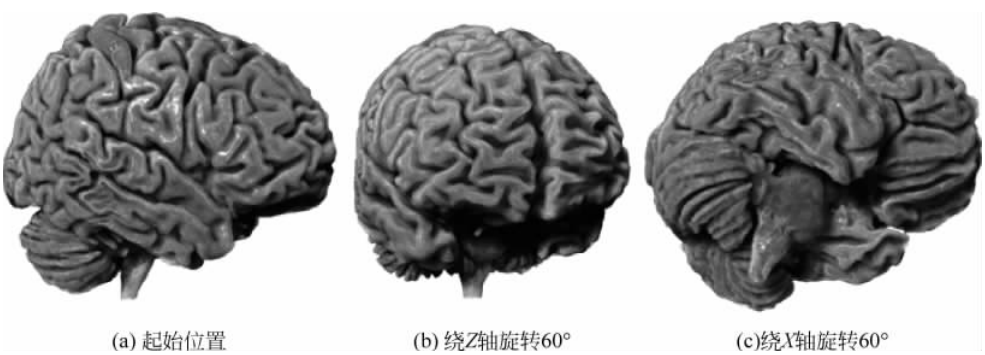


图 5-16 三维旋转变换示例

1) 线性变换与一对一变换

线性变换:许多作者认为仿射(Affine)变换是线性的。从严格意义上讲,这是不正确的。线性变换是一种满足以下条件的特殊变换:

$$L(\alpha x_A + \beta x'_A) = \alpha L(x_A) + \beta L(x'_A) \quad \forall x_A, \quad x'_A \in R \tag{5-58}$$

而仿射变换的平移部分不符合这一条件。确切地说,仿射变化是一种线性变换与平移变换的复合变换。

更进一步,反射(Reflection)变换也是一种线性变换,但是它们在图像配准中很少用到。举例而言,如果在图像指导的神经外科中所使用的配准算法包含反射变换,那么就有可能导致在穿颅手术中定位到错误的一边。如果怀疑到算法中可能包括反射变换,那么就一定要在应用之前得出确证,以免造成事故。

一对一变换:在对同一目标的配准中,病人通过不同的设备进行成像。这种配准中所要求的变换 T 似乎是一对一的。这意味着图像 A 中的点经过变换后与图像 B 中唯一确定的点对应,反之亦然。在有些情况下,这种规则并不适用。首先,如果配准图像的维数并不相同,例

如X光照片和CT成像之间的配准，一对一变换就是不可能的。其次，在一幅图像中的采样数据并没有反映在另一幅图像的采样数据中。

对于各种非仿射变换配准，一对一的变换都是不适用的。举例而言，在对不同目标的配准中，或者在对同一目标手术前和手术后的配准中，A图像中就可能存在B图像中所不包含的结构，反之亦然。

2）刚性变换

刚性变换（Rigid-Body Transform）是一类只包含旋转和平移的几何变换，它是更一般化的仿射变换的一个真子集。在三维空间中，刚性变换可表示为平移矩阵和旋转矩阵的乘积：

$$M = T \cdot R_x \cdot R_y \cdot R_z \tag{5-59}$$

其中，平移矩阵 T 描述了刚体变换在 x，y，z 轴方向上的偏移。

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-60}$$

旋转矩阵 R_x、R_y、R_z 分别描述刚体变换绕 x 轴（Pitch）、y 轴（Roll）、z 轴（Yaw）的旋转。

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-61}$$

三维刚性变换有6个自由度，包括三个平移变量（t_x，t_y，t_z）和三个旋转变量（α，β，γ）。根据三种旋转变换Pitch、Roll和Yaw的次序的不同，以及平移变换与旋转变换先后次序的不同，三维刚性变换可能有12种可能的变换形式。在医学图像配准应用中，通常按照Roll-Pitch-Yaw的次序进行三维旋转变换。

在实践中，普通的三维欧氏旋转变换在配准优化过程中往往表现不佳。一些研究[27]发现基于四元数（Quaternion）的三维旋转比普通的3D旋转矩阵在基于梯度的优化过程中表现出更好的稳定性和计算效率。式（5-62）描述了基于四元数的三维旋转变换公式，其中 α 为旋转角度，u 为旋转轴。

$$q = w + xi + yj + zk = w + (x, y, z) = \cos\left(\frac{\alpha}{2}\right) + u\sin\left(\frac{\alpha}{2}\right) \tag{5-62}$$

式（5-63）为由四元数 $q=w+xi+yj+zk$ 描述的正交旋转变换矩阵。

$$\begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2wy + 2xz \\ 2wz + 2xy & w^2 - x^2 + y^2 - z^2 & 2yz - 2wx \\ 2xz - 2wy & 2wx + 2yz & w^2 - x^2 - y^2 + z^2 \end{bmatrix} \tag{5-63}$$

在实践中，将Versor空间中的三维旋转变换和Vector空间的三维平移变换结合实现的刚体变换，与基于四元数的变换方法相比，不仅降低了几何变换空间的维数，同时也加快了优化过程算法收敛的速度。

对于2D-3D刚性配准，我们需要同时考虑刚性变换和3D物体在平面上的投影。我们可以将刚性变换、平移变换和投影变换在各向同性坐标系中统一成一个4×4的矩阵：

$$T_{\text{rigid}}=\begin{bmatrix}\cos\beta\cos\gamma & \cos\alpha\sin\gamma+\sin\alpha\sin\beta\cos\gamma & \sin\alpha\sin\gamma-\cos\alpha\sin\beta\cos\gamma & t_x\\ \cos\beta\sin\gamma & \cos\alpha\cos\gamma-\sin\alpha\sin\beta\sin\gamma & \sin\alpha\cos\gamma+\cos\alpha\sin\beta\sin\gamma & t_y\\ \sin\beta & -\sin\alpha\cos\beta & \cos\alpha\cos\beta & t_z\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{5-64}$$

通常认为投影是$(x、y、z)$沿着 z 轴方向投影到 $u、v$ 平面。这种投影变换可以通过图像系统的内在参数$(u_0、v_0、k_u、k_v)$来表示。对于 X 光投影而言，我们如下解释这些参数：u_0 和 v_0 定义了光线穿透点(在$(u、v)$平面上，该点的法矢指向 X 光光源)，k_v 和 k_u 分别表示像素点在水平方向(u)和垂直方向(v)的大小。另外，它们也可以作为未知量，这样就会给配准算法增加四个自由度。投影变换矩阵 $T_{\text{projection}}$可以表示成 4×3 的矩阵，它将沿着 z 轴方向投影：

$$T_{\text{projection}}=\begin{bmatrix}k_u & 0 & u_0 & 0\\ 0 & k_v & v_0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix} \tag{5-65}$$

在各向同性坐标系$(x、y、z、1)^T$中的 3D 坐标同这个矩阵相乘，得到向量$(\lambda_u,\lambda_v,\lambda)^{\mathrm{T}}$。其中 λ 代表缩放因子，投影到 2D 平面的结果可以由该向量中的前两项分别除以缩放因子 λ 得到。

2D-3D 刚性配准所需要的变换 $T_{\text{2D-3D}}$可以由投影变换和刚体变换组合而成：

$$T_{\text{2D-3D}}=T_{\text{projection}}\times T_{\text{rigid}} \tag{5-66}$$

当我们考虑骨质或者与之接近的结构时，这种刚性变换可以正确地将反映相同内容的图像对应起来。这种假设对在处理脑部图像的时候非常适用，因为脑颅骨限制了大脑的形变。

然而对身体内大多数组织而言，刚性变换远远满足不了配准的需要，我们需要更多的自由度来足够精确地描述组织的形变。相关非线性几何变换模型将在 5.2.4 小节进行介绍。

3) 仿射变换

仿射变换(Affine Transform)同时包含了平移、旋转、缩放、投影的错切变换，可以描述更为复杂的线性几何形变。在二维空间中，仿射变换有 6 个自由度；在三维空间中，仿射变换有 12 个自由度。式(5-67)为三维空间中的仿射变换公式。

$$\begin{bmatrix}x'\\ y'\\ z'\\ 1\end{bmatrix}=\begin{bmatrix}e_{11} & e_{12} & e_{13} & t_x\\ e_{21} & e_{22} & e_{23} & t_y\\ e_{31} & e_{32} & e_{33} & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix} \tag{5-67}$$

相对于刚性变换，仿射变换不需要正弦角度和余弦角度的计算，每个独立的变量可直接作为齐次变换矩阵上的元素，因此仿射变换在配准过程中的计算效率较高。一般的仿射变换可用于校正由 CT 台架倾斜引起的剪切或 MRI 梯度线圈不完善产生的图像畸变。

显然基准点的定位不可能完全正确。Fitzpatrick et al.[60]将其称之为基准定位误差 FLE (Fiducial Localisation Error)。其最小二乘余量称做基准配准误差 FRE (Fiducial Registration Error)，Sibson[61]对其分布进行了描述。Fitzpatrick et al. 强调 FRE 的值不重要，正如基准点并不是感兴趣点一样，而是他们定义的目标配准误差 TRE (Target Registration Error)，即对特定目标由 FLE 引起的误差。准确地说，如果 FLE 是 ε，则 Procrustes 问题的解对应的旋转变换与平移变换取决于 ε：$T_\varepsilon:=(R_\varepsilon,t_\varepsilon)$。则对目标 x 配准误差 TRE 为$|T_\varepsilon(x)\Leftrightarrow T(x)|$，对 ε 为一阶小量，并近似按照 $1/\sqrt{N}$，其中 N 为基准点个数[60]。结果可以总结如下，在 x 处(D 维坐标$(x_1,\cdots,x_D)$)TRE 的平方期望(一阶近似)为

$$\langle \mathrm{TRE}(x)^2 \rangle \cong \langle \mathrm{TLE} \rangle^2 \left(\frac{1}{N} + \frac{1}{D} \sum_{i}^{D} \sum_{j \neq i}^{D} \frac{x_I^2}{\Lambda_i^2 + \Lambda_j^2} \right) \tag{5-68}$$

其中 Λ 是标记位置的奇异值，并与标记相对点集分布主轴的位置有关。

5.2.4 非刚性配准

刚性变换和仿射变换可以很好地描述二维/三维空间上的线性几何变换关系，基于上述两种模型的配准方法仍是目前医学临床应用的主流。然而，对于一些包含有非线性几何形变的图像配准问题，如三维乳房核磁图像的配准、开颅手术术前术后的图像对比、四维动态心脏图像的配准以及人体胸腔肺部图像的配准等，刚性配准已经不能满足人们的需求，解决上述问题需要一些自由度更大、能够描述更加复杂的非线性软组织形变的几何变换模型。

非刚性配准(Non-Rigid Registration)是一类在配准图像(模型)间寻找非线性几何映射关系的配准方法。除了将图像对齐这一传统目标，非刚性配准更强调获得图像间密集的几何变换域。通常，非刚性配准算法可由三大模块组成：用于描述图像间非刚性形变的几何变换模块；描述图像间相似性程度的测度模块；根据图像相似性函数寻找几何变换最优解的优化模块。

非线性几何变换是非刚性配准的核心组成部分。目前可用于非刚性配准的几何变换模型种类繁多，通常可将它们分为两大类：基于物理模型的变换和基于函数描述的变换。基于物理模型的变换主要来源于物理学中的连续介质力学(Continuous Mechanics)理论，其又可划分为两个子类：弹性(Elasticity)变换和流体(Fluid Flow)变换。基于函数描述的变换来源于插值和近似理论，它们采用具有少量参数的基函数来描述复杂、稠密的非线性几何变换域。目前常用于非刚性配准的基函数主要有：径向基函数、B样条函数和小波函数等。

通常在进行非刚性配准前，首先需要对两幅待配准图像进行刚性配准，以使得在非刚性配准前两幅图像的初始几何偏差达到最小。另外在非刚性配准过程中，可以采用一些多分辨率和多重网格的策略，以提高配准过程的速度和精度，并避免局部极值问题产生。

1. 基于样条函数的非刚性配准

基于样条函数的配准方法主要利用控制点和样条函数来描述非线性几何变换域。将位于样条函数栅格上的控制点赋予一定的权值，当这些控制点产生空间位移时，样条函数随即被弯曲。在控制点周围，图像空间随着样条函数的弯曲而产生不同程度的扭曲。通过样条函数插值，这样的空间扭曲是平滑的。基于样条函数的图像变形主要由控制点决定，通常这些控制点可以是手动标记的图像特征点，如明显的局部解剖结构；此外控制点亦可均匀地分布于图像变换空间中，如控制点栅格阵列，它们只对图像周围的空间进行变换，与图像内容无关。

基于B样条的自由形式变换(FFDs)是目前最为流行的非刚性图像变换方法。它由Sederberg和Parry[34]首先提出，Rueckert[18]将上述变换模型首次应用于乳房核磁图像的非刚性配准。FFDs变换的基本思路是通过控制点栅格阵列中各点的自由平移变换，来对其包围的图像空间进行变形。相对于一些传统的非刚性形变模型，如高次多项式变换模型，FFDs变换的自由度更高、可以实现局部控制并且变换域具有 C^2 连续性。在三维空间中，基于B样条函数的自由形式变换可描述为三个一维三次B样条函数的张量积：

$$\Phi(u,v,w) = \sum_{l=0,m=0,n=0}^{3,3,3} \beta_{l,3}(u)\beta_{m,3}(v)\beta_{n,3}(w) \times P_{i+l,j+m,k+n} \tag{5-69}$$

其中 β_i 为第 i 次B样条的基函数：

$$\begin{cases}\beta_{0,3}(u) = \dfrac{(1-u)^3}{6} \\ \beta_{1,3}(u) = \dfrac{3u^3-6u^2+4}{6} \\ \beta_{2,3}(u) = \dfrac{-3u^3+3u^2+3u+1}{6} \\ \beta_{3,3}(u) = \dfrac{u^3}{6}\end{cases} \tag{5-70}$$

薄板样条(Thin-plate Spline)[9]变换是目前流行另一种非刚性几何形变模型。薄板样条是唯一一种可以将几何映射分解为刚性映射和非刚性映射的样条函数,基于薄板样条变换可描述为多个径向基函数的线性组合:

$$u(x) = Ax + B + I\sum_{i=1}^{N} b_i\theta(|\phi_i - x|) \tag{5-71}$$

其中变换矩阵 A 和 B 定义了一个仿射变换,系数向量 b_i 描述了薄板样条变换的非刚性变换部分,θ 函数为一个径向基函数。通常用于薄板样条变换的控制点通过手动指定,在保证控制点间一一对应的约束条件下,薄板样条变换模型参数可利用最小二乘方法直接求解计算,这一过程等价于最小化一个薄板样条弯曲能量函数。相对于基于 B 样条的自由形式变换,薄板样条变换是一种全局变换方法,它无法实现局部的变换控制。图 5-17 为应用薄板样条函数对三维大脑图像进行非刚性形变示意图。

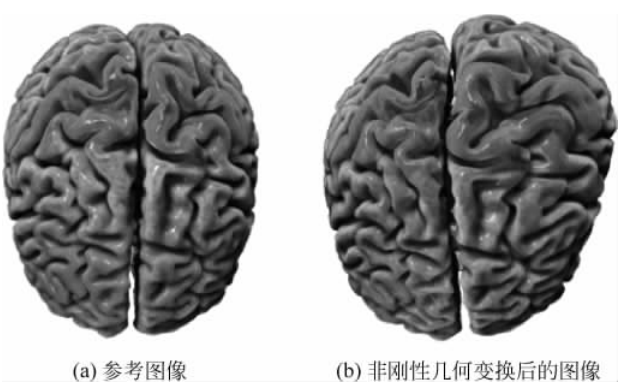


图 5-17 应用薄板样条函数对三维大脑图像进行非刚性形变

Davis et al.[9]为基于标记点配准提出了弹性体样条 EBS(Elastic Body Spline)。EBS 是描述线性弹性的 Navier-Cauchy 偏微分方程

$$\mu\nabla^2 u(x) + (\mu+\lambda)\nabla(\nabla\cdot u(x)) + f(x) = 0 \tag{5-72}$$

的解,其中,$u(x)$为 x 处的位移矢量;$f(x)$表示主导配准的单位体积内的体积力。一般地,假设多项式变形力 $f(x)=cr^{2k+1}$径向对称。当 $r=\|x\|$(欧氏模)且 c 为常向量考虑 $f(x)=cr$ 与 $f(x)=(c/r)$。则得到具有如下形式的形变场

$$u(x) = \sum_{i=1}^{N} R(x-p_i)c_i \tag{5-73}$$

其中,p_i 是 N 个目标标记点中第 i 个的位置;c_i 为与形变力有关的系数。简单地说,上式为移位基函数 R 的线性组合,表示为 3×3 矩阵。因此偏微分方程三项都是耦合的。使用伽辽金(Galerkin)向量法求解这三个耦合方程。该方法将三个耦合偏微分方程变为三个独立的径向

对称双调和方程并得到以下两个分别对应体积力 $f(x)=cr$ 与 $f(x)=(c/r)$ 的解，

$$R(x) = [(12(1-\upsilon)-1)r^2 I - 3xx^T] \tag{5-74}$$

$$R(x) = \left[(8(1-\upsilon)-1)rI - \frac{1}{r}xx^T\right] \tag{5-75}$$

其中，$\upsilon=(\lambda/[2(\lambda+\mu)])$ 是泊松比；I 是 3×3 矩阵；xx^T 是外积。体积力的系数 $W=[c_1^T\cdots c_N^T]^T$ 可以从线性方程系统 $W=K^{-1}U$ 求出，其中 K 是元素为 $R(p_i-p_j)$ 的 $3N\times3N$ 矩阵，$U=[u_1^T\cdots u_N^T]^T$ 是位移向量。使用奇异值分解求解 W。比较体积样条与三维薄板样条分别具有形式 $R(x)=r^3 I$ 与 $R(x)=rI$。所以除去式(5-74)与式(5-75)中的乘性常数，包含 xx^T 的项可看作是对体积样条与三维薄板样条的修正，所以也符合 Navier-Cauchy 物理模型。

2. 弹性配准

弹性配准(Elastic Registration)[35]是一种基于物理模型的图像配准方法。在配准过程中，假设待配准图像在外力作用下只会产生弹性的形变。这样的形变过程可由 Navier 线性弹性方程来描述：

$$\mu\nabla^2\vec{u} + (\lambda+u)\nabla(\nabla\times\vec{u}) + \vec{f} = 0 \tag{5-76}$$

其中，$\vec{u}$ 描述非线性几何变换域；$\vec{f}$ 为作用在图像上的外力；λ 和 μ 为弹性形变约束参数，它们主要由配准图像的材料属性来决定。∇ 为梯度算符，∇^2 为 Laplace 算符。式(5-76)给出的偏微分方程可利用有限差分和逐次超松弛迭代(Successive Over-relaxation)算法求解。

3. 基于光流模型的配准

光流模型[20]的概念最早在计算机视觉领域中提出，它主要用于图像序列中两个连续帧间的运动估计。光流模型基于这样的假设：图像中给定点的灰度值在短时间间隔内是恒定不变的。光流约束方程可描述为：

$$I(x,y,z,t) = I(x+\delta x, y+\delta y, z+\delta z, t+\delta t) \tag{5-77}$$

将上式等号左右两边进行泰勒展开，并忽略高阶项，光流约束方程可表示为：

$$\Delta I + \nabla I \cdot u = 0 \tag{5-78}$$

其中，ΔI 为图像间的时间微分；∇I 为图像的空间灰度梯度；u 描述了图像间的非线性变换关系。

4. 非刚性几何变换的约束

对于很多配准应用，保证非刚性几何变换域的平滑性和可逆性十分重要。具有平滑和可逆性质的空间几何变换通常称为微分同胚(Diffeomorphism)变换。然而，对于大多数非刚性几何变换方法来说，在配准过程中都会存在一定程度的图像交叉和折叠问题。特别是对于那些高维的形变模型，如弹性模型、光流模型，在变换模型内部完全保证几何变换的平滑可逆是一个病态问题。为了解决上述问题，目前常用的方法是在配准代价函数中加入一个几何变换的约束项。

目前对非刚性几何变换施加约束的方法很多，如在对称配准算法中采用的逆向变换一致性约束[36]，利用几何变换的雅可比矩阵正定性来保持拓扑关系的不变[37]。目前常用的一种方法是在配准代价函数中加入薄板样条能量约束项[18]。

$$C_{\text{regular}} = \iiint_{\Omega}\left[\left(\frac{\partial^2 T}{\partial x^2}\right)^2 + \left(\frac{\partial^2 T}{\partial y^2}\right)^2 + \left(\frac{\partial^2 T}{\partial z^2}\right)^2 + 2\left[\left(\frac{\partial^2 T}{\partial xy}\right)^2 + \left(\frac{\partial^2 T}{\partial xz}\right)^2 + \left(\frac{\partial^2 T}{\partial yz}\right)^2\right]\right]\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{5-79}$$

5.3 MITK 配准总体框架

MITK 配准框架的设计与实现借鉴了众多优秀算法平台的成功经验，包括 ITK 的基于图像灰度的配准算法框架模块化设计思路、RGRL 的点集配准框架设计思路等，同时在此基础上我们也提出和实现了不少自己观点和方法。在 MITK 配准算法框架简洁、紧凑的实现中，同时融合了基于图像灰度和基于特征的配准方法。以上方法在统一的类继承架构下实现，使得整体框架在为用户二次开发提供了一致的数据和算法接口的同时，具有较低代码冗余度和高可扩展性。在配准框架的局部，我们采用 CPU SSE 等硬件加速方法对算法进行优化，提高了整个配准框架的运行效率。

5.3.1 框架设计

很多实际应用中一个简单的刚性变换就可以很好地描述图像间的空间对应关系，例如同一主体(Intra-Subject)的脑部图像配准。目前，刚性配准技术已经发展得比较成熟，并且已经进入临床应用。因此 MITK 目前版本中主要是考虑这些技术的实现。当然也有很多情况下需要用到各种比较复杂的非刚性变换。相比较而言，非刚性配准还处于很活跃的研究阶段[7]。实际配准过程中，可以根据不同的特点和要求采用简单的刚性变换(Rigid Transform)、仿射变换(Affine Transform)，或较复杂的弹性形变(Elastic Deformation)等。

由于医学图像配准的算法比较复杂，而且方法种类很多，新的方法也层出不穷，为了给使用者提供一个开放的、易于扩充的架构，便于以后添加新的配准算法，我们在 MITK 中根据配准算法的一般步骤，分几何变换(Transform)、图像插值(Image Interpolator)、相似性测度(Similarity Metric)、优化(Optimizer)四个相对独立的模块来实现整个配准算法，各模块间的相互关系如图 5-18 所示。

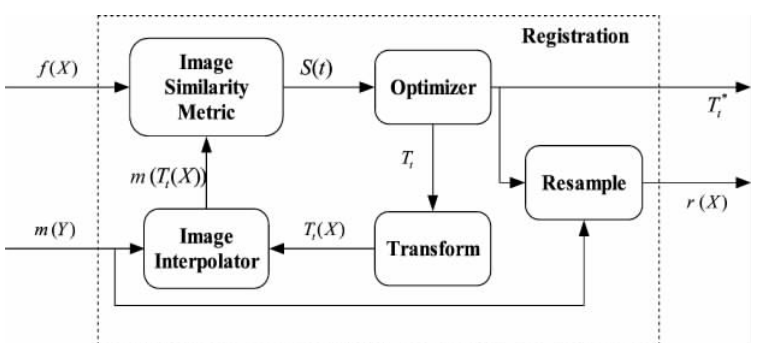


图 5-18 MITK 配准算法框架图

其中 S 为对任意两幅图像定义的一目标函数，用来衡量两图像的匹配效果，一般用相似性测度表示。

从算法框图中可以看出整个配准算法的流程：

(1) 输入待配准的两幅图像，分别记为参考图 $f(X)$(Fixed Volume)和浮动图 $m(Y)$(Moving Volume)；

(2) 对参考图指定区域 X 进行几何坐标变换(Transform)得到新的区域 $T_t(X)$坐标，其

中 t 表示变换参数；

(3) 通过一定的插值方法(Image Interpolator)得到浮动图在区域 $T_t(X)$的取值 $m(T_t(X))$；

(4) 在相似性测度模块计算参考图 $f(X)$和插值图 $m(T_t(X))$的相似度，它是一个关于几何变换参数的函数 $S(t)$；

(5) 相似度函数 $S(t)$输入优化模块中进行最优化计算得到最终变换参数，这个过程在计算中一般通过迭代来实现，即重复步骤(2)～(4)直到取得最大相似度时终止迭代循环；

(6) 整个配准算法模块输出配准时所采用几何变换的最优变换参数以及浮动图在最优变换下的插值图像。重采样模块 Resample 由 Transform 和 Image Interpolator 组成。

可以看出在此过程中，图像配准实际上已经转为如下描述的最优化问题：寻找一个最优变换 T_t，使相似度函数 $S(t)$取得最大值。下面分别对以上各算法模块作简要介绍。

几何变换(Transform)将参考图像空间 X(Fixed Volume Space)中像素点映射到浮动图像空间 Y(Moving Volume Space)中去。这和配准中图像变换的直观理解有点不一样，刚好是它的逆过程。因为正变换中从浮动图像空间变换到参考图像空间后可能会产生“空洞”，不利于后续的插值处理，逆变换则可以避免这一点。几何变换的类型一般有刚性变换、仿射变换、投影变换、曲线变换等。各种变换都是通过一组变换参数 t 来表示，例如刚性变换可用 3 个坐标轴方向上的平移参数和 3 个绕坐标轴旋转角度共 6 个参数来表示。几何变换模块在 MITK 中用 Transform 抽象类表示，具体的变换算法由 Transform 派生类实现。

图像插值(Image Interpolator)是为了估计浮动图中非网格点处的像素值。由于数字图像是对模拟图像的网格采样，只有格点处有像素值，而参考图像空间中的格点映射到浮动图像空间中去后可能不在格点上，为了得到这些浮动图中非格点处的像素值就有必要进行图像插值。一般常采用的插值方法有最近邻插值、线性插值、B 样条插值等。图像插值模块有两个输入一个输出：浮动图像 $m(Y)$(Moving Volume)和几何变换结果 $T_t(X)$(Transform 的输出)作为输入；插值结果 $I(X)=m(T_t(X))$作为输出。插值结果也是一个 Volume，因此 MITK 中将该抽象类(Interpolator)作为 VolumeToVolumeFilter 的派生类。

相似性测度(Similarity Metric Measure)作为一种准则用来评价参考图和插值后得到的图像匹配的效果，可以说这是整个配准框架中最关键的部分，它直接影响配准效果的好坏[1]。相似性测度是一个以几何变换参数 t 为自变量的单值函数：$S(t)=S(f(X),m(T_t(X)))$。该模块以参考图 $f(X)$和浮动图插值后得到的图像 $m(T_t(X))$作为输入，输出一个表示两幅图像相似(相关)程度的标量值，可以看作一个以变换参数 t 为自变量的函数，该函数输入到优化模块中作为代价函数(Cost Function)求取最优变换。一些非刚性配准算法中，除相似性测度外，还要考虑形变约束，两者之和作为优化的代价函数，即

$$C=-C_{\text{similarity}}+\lambda C_{\text{deformation}} \tag{5-80}$$

函数优化(Cost Function Optimizer)如同图像处理与分析领域中的很多问题一样，图像配准也被归结成一个多参数优化问题。如前所述，给定一种配准准则，就可以定义一个以几何变换参数为自变量的多元目标函数，通过对该目标函数的最优化搜索得到配准时的几何变换参数。由于优化问题是一个比较经典的数学问题，有很多解决方法，这里关键的就是根据目标函数的特点选择合适的优化算法和策略，准确、高效、快速地得到结果。

MITK 中上述算法模块分别用 Transform、InterpolateFilter、Metric、Optimizer 4 个抽象类表示。根据图，Transform 得到几何变换参数后对参考图定义域进行几何变换，输出变换后的坐标；InterpolateFilter 输入 Transform 变换后的坐标及浮动图进行插值运算，输出新的图

像像素值；Metric 输入参考图和插值图，输出相似度函数；Optimizer 对输入的相似度函数进行优化，输出优化后的变换参数。这些模块在配准类 RegistrationFilter 中通过相应的接口组合到一起，完成整个配准算法流程，最后输出参考图变换后的重采样图像。显然 RegistrationFilter、InterpolateFilter 输入输出均为三维体数据 Volume，可以作为 VolumeToVolumeFilter 的派生类继承其数据成员和成员函数。Transform、Metric、Optimizer 作为 ProcesssObject 的派生类。各算法模块类的继承结构及相互组合关系如图 5-19、图 5-20 所示。

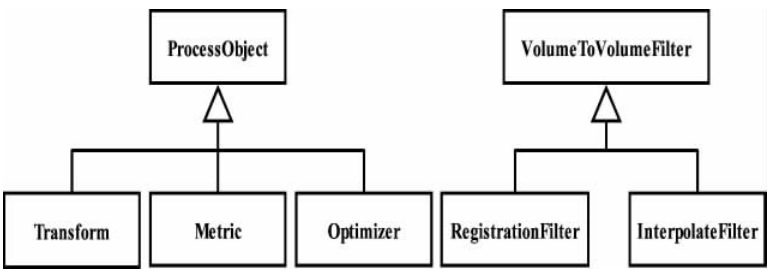


图 5-19　MITK 配准算法各模块类继承结构

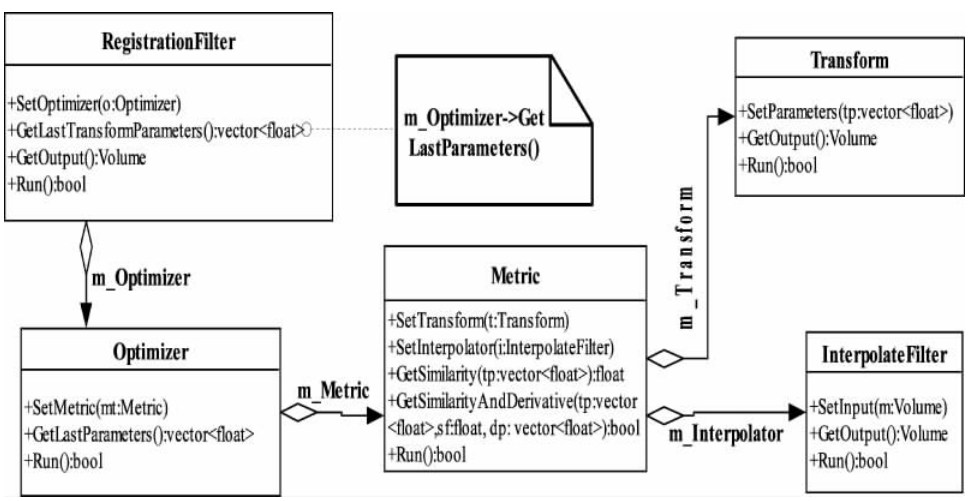


图 5-20　MITK 配准算法框架各模块类关系图

以上 Transform、InterpolateFilter、Metric、Optimizer 4 模块是从高层概念上实现了相应的算法，提供基本的数据成员和成员函数接口。不同类型的各种具体算法通过它们的派生类在虚函数中实现。

用户通过设置具体的几何变换类型、插值方法、相似性度量准则、优化策略就能实现不同的配准算法。下面几节详细介绍 MITK 中采用的具体几何变换类型、插值方法、相似性度量准则和优化策略。

5.3.2　图像相似性测度

相似性测度(Similarity Metric)定量化地衡量了两幅图像匹配的效果，它是图像配准过程中十分重要的一部分[5]。一般情况下，待配准的图像是在不同时间、不同条件，甚至不同成像技术下获取的，图像描述的信息可能存在本质的差别，这种情况下就没有绝对的配准问题，那

么我们的任务就是寻找一种准则，使两幅图像在这种准则下达到最佳的匹配效果。这里的准则称之为相似性测度，在一些非刚性配准中还加上形变约束。准则的选择和配准目的、具体的图像形态、几何变换类型等有关。例如，有些准则允许很大的几何变换搜索范围，而有些准则要求初始位置和最优的配准结果比较接近才能得到正确结果；有些准则仅仅适用于同一模态图像间的配准，而有些准则能处理不同模态的图像配准。遗憾的是现在还没有一个明确的准则来指导在各种情况下如何选择配准的相似度量准则，更不存在各种情况下都通用的相似度准则。

既然配准只是在某种准则下取得相对最优，准则的选择直接影响着配准的效果。因此，如何选择合适的相似性度量准则就成为图像配准中一个十分关键的研究问题，大量的研究论文也表明了这一点。从发表的论文来看，主要有两种相似性度量准则：基于特征(Feature-based)和基于体素(Voxel-based)。基于特征的准则一般是最小化两图像相应特征间的距离，常用的特征有对应解剖结构中的控制点、二维边缘线、三维表面等。这种准则下，通常先要提取特征，利用这些局部的特征信息进行配准。特征提取的准确性直接影响着配准的精度。基于体素的方法是目前的研究热点。从理论上来讲，这种方法应该是最灵活的，因为它利用了图像中的所有信息。从总体上来看，这类准则中较常见的有：相关性测度，包括相关系数、傅里叶域的互相关和相位相关等；总体平均差最小化；灰度比的方差最小化；互信息最大化。其中基于互信息最大化的方法获得了很大的成功[5]，大量文献表明，该方法不仅适用于单模态图像配准，对多模态图像配准问题也能取得不错的结果。

MITK 中目前实现了下列基于体素的相似性度量准则：

- 灰度均方误差(Mean Squares Metric)
- 归一化相关系数(Normalized Correlation Metric)
- Pattern Intensity Metric
- 互信息(Mutual Information Metric)
- 点集均方误差 (Point Set Euclidean Distance Metric)

下面我们先简要介绍这几种准则，然后看看其在 MITK 中的实现。为了书写方便，我们记参考图 $f(X)$和变换插值后的浮动图 $m(T_t(X))$为 A 和 B。A_i，B_i 分别是图像 A，B 第 i 个像素的灰度值，N 是计算区域的像素个数。

1. *灰度均方误差*

mitkMeanSquaresMetric 类计算图像 A，B 在给定区域的灰度平均差。

$$\mathrm{MS}(A,B)=\frac{1}{N}\sum_{i}^{N}(A_i-B_i)^2 \tag{5-81}$$

理想情况下的最优测度是 0，这是基于以下假设：两图像对应像素点灰度值相同。因此该准则只适用于同模态图像配准。该准则计算简单，相对来说可以在一个比较大的范围内搜索匹配。但该准则对图像灰度值的线性变化比较敏感。

2. *归一化相关系数*

mitkNormalizedCorrelationMetric 计算图像 A，B 的归一化互相关系数。

$$\mathrm{NC}(A,B)=\frac{\sum_{i}^{N}(A_i\cdot B_i)}{\sqrt{\sum_{i}^{N}A_i^2\cdot\sum_{i}^{N}B_i^2}} \tag{5-82}$$

理想情况下该准则的最优值是 1。该准则也仅限于单模态图像配准，并且产生尖峰状极值，搜索范围较小。

3. Pattern Intensity

mitkPatternIntensityMetric 计算灰度差，并代入钟形函数$\frac{1}{1+x^2}$求和。

$$\mathrm{PI}(A,B) = \sum_{1}^{N} \frac{1}{1+\lambda(A_i - B_i)^2} \tag{5-83}$$

其中 λ 是比例系数，控制搜索范围。该准则的具体性质可参考 Penney[63] 和 Holden[64] 的论文。该准则也仅限于单模态图像配准，对图像灰度值的线性变化比较敏感。

4. 互信息

mitkMutualInformationMetric 计算两幅灰度图像间的互信息测度

$$\mathrm{MI}(A,B) = \sum_{a\in A}\sum_{b\in B} p_{AB}(a,b)\log_2\left[\frac{p_{AB}(a,b)}{p_A(a)p_B(b)}\right] \tag{5-84}$$

两幅图像间的互信息通过计算图像的熵(公式(5-4))和图像间的联合熵(公式(5-5))得到。在实践中，我们采用高斯密度函数作为图像灰度分布密度估计函数(Probability Distribution Function)。为了提高算法的计算效率，我们实现了文献[64]中提出的随机最大互信息(Stochastic Maximization of Mutual Information)方法，即只计算两幅配准图像的随机采样的互信息值。在随机抽样算法实现中，我们采用了 SFMT(SIMD-oriented Fast Mersenne Twister)随机数生成算法。为了便于设置优化过程的收敛条件，我们还实现了归一化的互信息测度方法 mitkNormailizedMutualInformationMetric，相关计算见公式(5-12)。

5. 点集均方误差测度

除了以上基于体素的相似性测度，MITK 中还实现了基于图像特征点集的相似性测度 mitkEuclideanDistanceMetric，用来度量参考图和浮动图的标志点集中对应点的均方距离误差。其中 N 代表标志点集中的点的数目。A_i，B_i分别代表参考图和浮动图中第 i 个标志点的坐标，距离的定义采用常用的欧氏距离。

$$\mathrm{PS}(A,B) = \frac{1}{N}\sum_{i}^{N} \| A_i - B_i \|^2 \tag{5-85}$$

mitkEuclideanDistanceMetric 主要的函数接口如下：

```
输入固定点集
virtual void SetFixedPointSet(vector<double> *fixedPointSet)
输入浮动点集
virtual void SetMovingPointSet(vector<double> *movingPointSet)
计算点集欧氏距离
double GetPointSetSimilarity()
```

6. 相似性测度在 MITK 中的实现

从图可以看出，算法上相似性测度模块 Metric 接受 Fixed Volume 和 Image Interpolator 的输出作为输入，即输入两个 Volume 体数据，输出的是计算得到的相似度量值。在 MITK 具体实现中，我们用 Metric 这一抽象基类来实现相似性测度，其框架如图 5-21 所示。

我们将 Transform 和 Interpolator 作为 Metric 的成员变量封装起来，在 bool Execute() 函数中一次性完成几何变换、图像插值以及求相似度的工作。各种不同的相似性测度都通过 Metric 派生出来，并在虚函数 Execute()中具体实现，如图 5-22 所示。

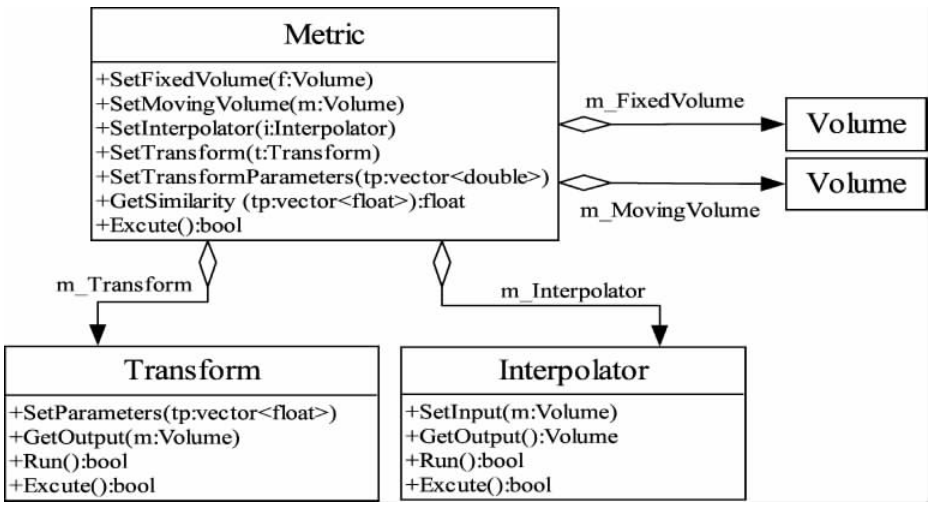


图 5-21 MITK 图像相似性测度算法类图

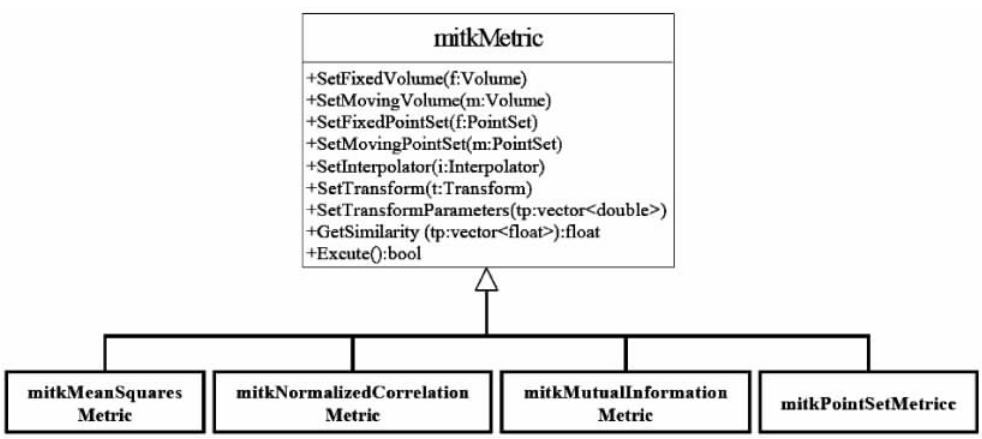


图 5-22 MITK 图像相似性测度算法类层次结构图

对于各种相似性测度,用户接口都是一致的。

输入参考图数据信息

```
void SetFixedVolume(mitkVolume * fixedVolume)
```

输入浮动图数据信息

```
void SetMovingVolume(mitkVolume * movingVolume)
```

指定几何变换类型

```
void SetTransform(mitkTransform * transform)
```

指定变换参数

```
void SetTransformParameters(const VectorParameterType & parameters)
```

指定图像插值的类型

```
void SetInterpolator(mitkInterpolator * interpolator)
```

获取图像间的相似性测度值

```
Virtual bool GetSimilarity ( const VectorParameterType & parameters, ScalarParameterType &similarity)
```

获取图像间的相似性测度值以及变换的 Jacobian 矩阵

```
virtual bool GetSimilarityAndDerivative(const VectorParameterType & parameters, ScalarParameterType &similarity, VectorParameterType & derivative);
```

5.3.3 几何变换

空间映射 T 描述了一幅图像中的位置与另一幅图像中的相应位置之间的关系。这里的图像可以是二维的(2D),也可以是三维的(3D),所以这种映射可能是从二维空间到二维空间、从三维空间到三维空间,或者是在三维和二维之间的变换。然而在所有上述情况中,象源体——人体的部分或全部——都是三维的。所以,对大多数情况而言,二维空间内部的变换无法满足配准的要求。目前最广泛的配准应用中就包含三维图像对之间的配准。另外,一个更加重要的应用就是二维图像与三维图像之间的配准(2D-3D 配准)。在 2D-3D 的配准过程中,T 包含了从 3D 物体到 2D 平面的投影,以及 3D-3D 的变换。

由于医学成像中的体数据通常为各向异性(Anisotropic),即切片间距与切片上像素间距不同,因此配准过程中几何变换需在物理空间进行。图像坐标系与物理坐标系的转换关系可由下面的等式(5-86)来描述。

$$\text{物理坐标} = \text{像素坐标} \times \text{像素间距} + \text{图像原点坐标} \tag{5-86}$$

在 MITK 中,我们实现了一系列不同自由度的几何变换算法子类,包括缩放变换(mitkScaleTransform)、平移变换(mitkTranslationTransform)、刚体变换(mitkRigidTransform)、相似性变换(mitkSimilarityTransform)和仿射变换(mitkAffineTransform)。对上述几何变换模型的数学描述详见 5.2.3 一节。

以下介绍几何变换算法在 MITK 中的实现。

从图的配准算法框架图中可以看出,变换模块 Transform 接受优化模块 Optimizer 的输出作为输入,即输入是一个变换参数集合 t。而 Transform 的输出是一个数据集,它记载着变换后点的坐标位置,其组织形式与 Volume 相同。这个输出进一步作为插值模块 Interpolator 的输入参数之一。除此之外,Transform 还提供了单点变换的接口,用以满足那些无需对全图像空间进行变换的需求,Transform 类的核心是一个 4×4 的变换矩阵,功能基本上可以实现任意线性变换。抽象类 Transform 的框架图如图 5-23 示。

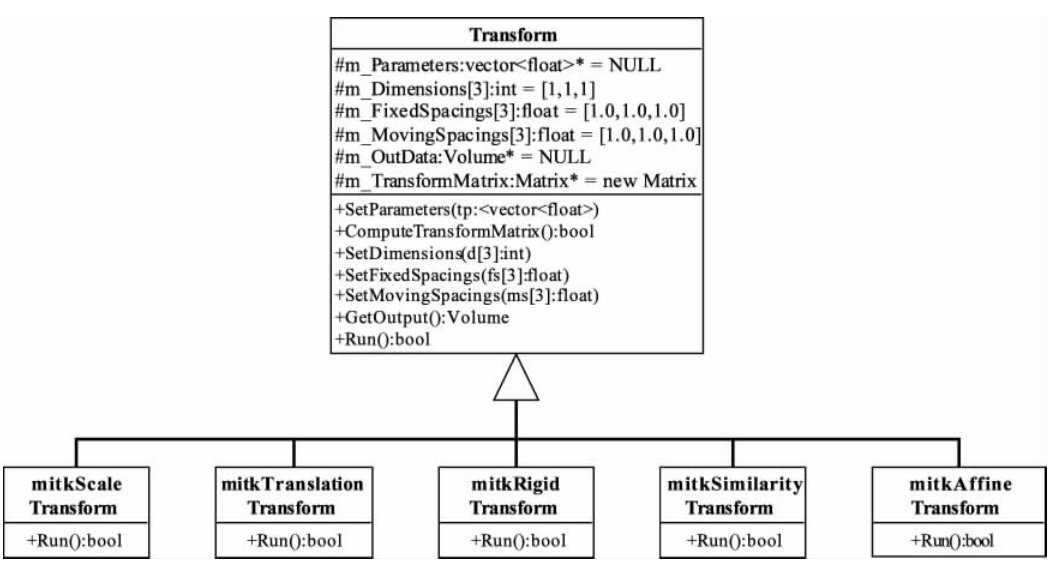


图 5-23 MITK 几何变换算法类层次结构图

对于各种变换算法，其用户接口都是一致的。

```
设置变换的参数
virtual void SetParameters(const VectorParameterType& parameters)
设置变换图像数据的大小
void SetRegion(const VectorIndexType& r)
获取像素(x,y,z)变换后的 Jacobian 矩阵
virtual const MatrixType & GetJacobian(const VectorParameterType& inPoint)
获取 4×4 的变换矩阵
mitkMatrixType * GetTransformMatrix()
计算像素(x,y,z)经几何变换后的坐标
virtual void TransformPoint(const VectorIndexType & inPoint,VectorParameterType & outPoint)
```

具体的变换算法实现将在子类中，由 virtual bool Excute()的重载函数实现。在变换参数和图像数据大小设置完成之后，就可以通过 Run()函数调用 Exctute()，从而完成变换工作。最终，变换的结果可以通过函数 Volume * GetOutput()得到。

变换核函数的实现代码如下。

```
void mitkTransform :: _transform(const ScalarParameterType * inPoint, ScalarParameterType *
outPoint)
{
    // Matrix based linear transform kernel

    if(!m_FlagComputeTransform)
        this->ComputeTransformMatrix();

    mitkMatrixScalarType *m = *m_TransformMatrix;
    if(m_SpaceDimension == 2)
    {
        outPoint[0] = m[0] * inPoint[0] + m[4] * inPoint[1] + m[12];
        outPoint[1] = m[1] * inPoint[0] + m[5] * inPoint[1] + m[13];
    }
    else
    {
        outPoint[0] = m[0] * inPoint[0] + m[4] * inPoint[1] + m[8]  * inPoint[2] + m[12];
        outPoint[1] = m[1] * inPoint[0] + m[5] * inPoint[1] + m[9]  * inPoint[2] + m[13];
        outPoint[2] = m[2] * inPoint[0] + m[6] * inPoint[1] + m[10] * inPoint[2] + m[14];
    }
}
```

5.3.4 图像插值

在配准过程中，相似性测度(Metric)通常是比较固定图像(Fixed Image)和变换图像(Moving Image)之间对应点的灰度值。当一个点通过某种变换，从一个空间映射到另一个空间时，目标点的坐标通常不在网格点上。在这种情况下，插值算法就需要用来估计目标点的灰度值。变换示意图见图 5-24。

插值方法影响着图像的平滑性、优化的搜索空间以及总体的计算时间。因为在一个优化周期之中，插值算法将被执行成千上万次。所以，在选择插值方法时，我们需要在计算复杂性和图像平滑性之间做一个权衡，常用的图像插值方法有以下几种：最近邻插值、线性插值、PV 插值等。下面我们一一简单地进行介绍。

1. 最近邻插值

最近邻域法是指把距离非相网点(u,v)最近的 u-v 坐标系中的格网点(u',v')的灰度值设为(u,v)点灰度值。如图 5-25 所示，最近邻插值计算量小，不足之处是会降低图像的平滑性，

视觉效果比较差，产生锯齿边界。

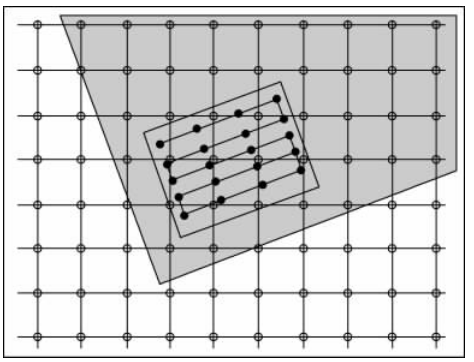


图 5-24　固定图像网格点与变换后的浮动图像非网格点

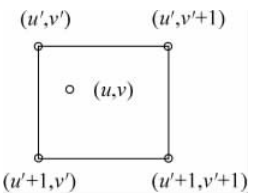


图 5-25　最近邻域插值

2. 线性插值

双线性内插法是指如图 5-26 所示，采用在(u,v)周围 4 个格网点的灰度值进行内插，其关系式为：

$$f(u,v)=(1-\alpha')(1-\beta')f(x,y)+\beta'(1-\alpha')f(x,y+1)+\alpha'(1-\beta')f(x+1,y)+\alpha'\beta'f(x+1,y+1) \tag{5-87}$$

线性内插法因其运算简单，精度高，是最常用的图像插值方法之一。另一方面，线性内插具有低通滤波特性，会使图像边缘产生模糊。

3. PV 插值

PV(Partial Volume)插值是指如图 5-27 所示，通过周围 4 个点距对应点的距离分配内插权值，以使它们贡献于联合灰度分布统计。假设两幅待配准的图像为 F(浮动图像)和 R(参考图像)。令 T_α是由参数α 确定的变换矩阵，它将被作用于 F。假设 T_α将 F 中的点(a,b)映射到 R 中的坐标$(i+\Delta_i,j+\Delta_j)$，其中(i,j)是 R 中的网格点，并且 $0\leqslant\Delta_i,\Delta_j<1$。

如果 f 是实函数，满足：

(1) $f(x)\geqslant 0$，x 是实数；

(2) $\sum_{n=-\infty}^{\infty} f(n+\Delta)=1$，$n$ 是整数，$0\leqslant\Delta\leqslant 1$。

那么，对于 F 中的任何点(a,b)，联合灰度分布为：

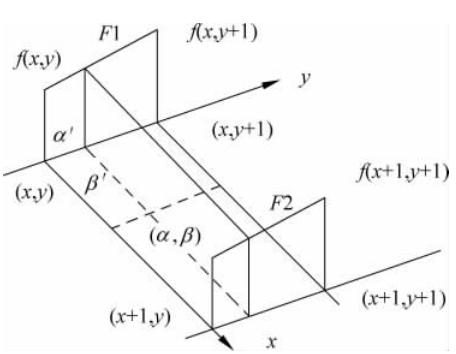


图 5-26　双线性内插法

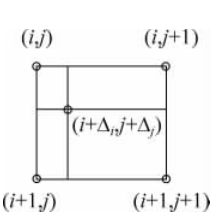


图 5-27　PV 插值

$$h(F(a,b),R(i+p,j+q)) = h(F(a,b),R(i+p,j+q)) + f(p-\Delta_i) * f(q-\Delta_j) \quad (5\text{-}88)$$

其中，p，q 是整数；f 是核函数。

PV 插值的结果并不产生新的像素值，而是计算新的像素对图像联合灰度分布的贡献，因此 PV 插值主要用于基于互信息的图像配准[5]。

4. 插值算法在 MITK 中的实现

从图 5-29 中可以看出，插值模块 InterpolateFilter 接受变换模块 Transform 的输出作为其输入。另外，它还有一个 mitkVolume 的输入参数。经过插值运算后，InterpolateFilter 输出 mitkVolume 类型的参数，并将其作为 Metric 模块的输入。此外，InterpolateFilter 也提供了单点插值的接口。

在 MITK 中，我们实现了 4 种不同精度和复杂度的插值算法子类：最近邻插值（mitkNearestNeighborInterpolateFilter）、线性插值（mitkLinearInterpolateFilter）、立方插值（mitkCubicInterpolateFilter）和 B-样条插值（mitkBSplineIntepolateFilter）。抽象类 mitkInterpolateFilter 的接口及继承结构如图 5-28 所示。

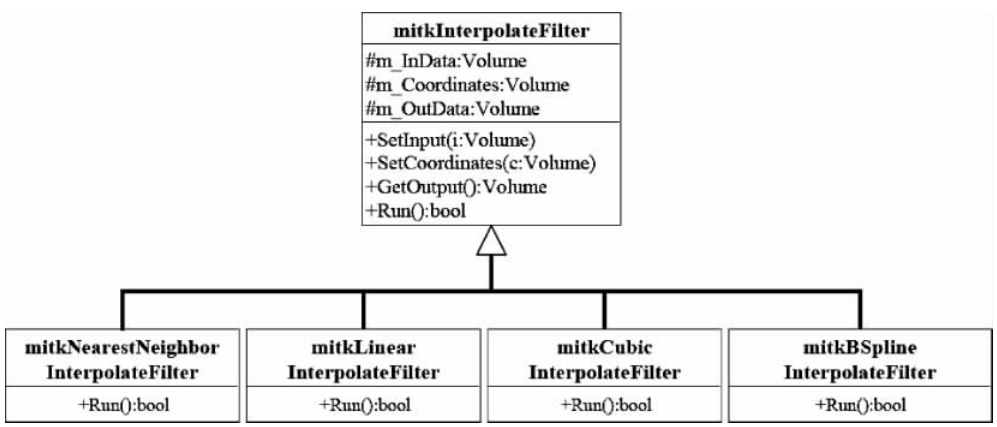


图 5-28 MITK 插值算法类层次结构图

InterpolateFilter 模块函数介绍如下。

```
设置原始图像数据
void SetInput(mitkVolume * inData)
获取图像插值结果
mitkVolume* GetOutPut()
获取单个像素点插值结果(单通道)
virtual bool InterpolatePoint(const VectorParameterType & index,ScalarPixelType &value)
获取单个像素点插值结果(多通道)
virtual bool InterpolatePoint(const VectorParameterType & index,VectorPixelType & value)
设置输出图像数据格式
void SetOutputDatatype(int dataType)
```

在 Interpolator 中设置完图像数据和变换结果之后，就可以通过 Run()调用 Execute()执行具体的插值操作。

Interpolator 中双线性插值核函数如下。

```
void mitkLinearInterpolateFilter::BilinearKernel(float x,float y,float z,float* value)
                                                    //对 x,y,z 处的像素进行双线性插值
```

```
{
  long a,b,c;
  float x11,x12,x21,x22;

  a = (long)x;
  b = (long)y;
  c = (long)z;
                                    //进行边界检测,判断邻域是否全在原始图像空间内,否则进行修正
  if (a == m_InDimensions[0])
      a--;
  if (m_InDimensions[1] - b <= 1)
      b--;
  if (c == m_InDimensions[2])
      c--;
          for(int i = 0; i < m_NumberOfChannels; i++)     //支持多通道数据插值
  {
      x11 = m_InputData[m_NumberOfChannels * (c * m_ImageSize + b * m_InDimensions[0] +
a  ) + i];
      x12 = m_InputData[m_NumberOfChannels * (c * m_ImageSize + (b+1) * m_InDimensions[0] +
a  ) + i];
      x21 = m_InputData[m_NumberOfChannels * (c * m_ImageSize + b * m_InDimensions[0] +
a+1) + i];
      x22 = m_InputData[m_NumberOfChannels * (c * m_ImageSize + (b+1) * m_InDimensions[0] +
a+1) + i];

      value[i] = (b + 1 - y) * ((x - a) * x21 + (a + 1 - x) * x11) + (y - b) * ((x - a) *
x22 + (a + 1 - x) * x12);
  }
}
```

5.3.5 函数优化

根据相似性测度选择的不同，配准变换的参数求解方式可分成两类：一是从获得的数据用联立方程组直接计算得到的；二是以对定义在参数空间的能量函数最优化搜索得到的。前者完全限制在基于特征(Feature-Based)的配准应用中。在后者中，所有的配准问题都变成一个能量函数的极值求解问题，能量函数是由需要被优化的变换参数表示的，一般是拟凸的，能用标准的优化算法求解极值。图像配准问题本质上是多参数优化问题，所以优化算法的选择至关重要。常用的优化算法有：Powell 法、下山单纯形法、Arent 法、Levenberg-Marquadrt 法、Newton-Raphson 迭代法、随机搜索法、梯度下降法、遗传算法、模拟退火法、几何 hash 法、半穷尽搜索法。在实际应用中，经常使用附加的多分辨率和多尺度方法加速收敛，降低需要求解的变换参数数目，避免局部最小值，并且多种优化算法混合使用，即开始时使用粗略的快速算法，然后使用精确的慢速算法。

图像配准中的优化函数就是相似性测度函数 $S(t)$，其中 t 是 n 维向量，表示几何变换中的 n 个参数。这里的优化函数 $S(t)$有几个显著特点。

(1) 一般来说，$S(t)$不够平滑，存在很多局部极值点。这给优化算法的选择提出了很高的要求，由于这些局部极值点的存在，优化结果会对初始值(初始变换)比较敏感，导致配准结果不够鲁棒。

(2) $S(t)$中各参数的变化对函数结果的影响因子差别较大。比如在刚性变换下，旋转比平移引起的图像变化要大。因此，最好给每个参数分配一个缩放因子，使各参数以合适的步长进行迭代搜索，减少搜索时间、提高计算精度。另外还要优化各参数的搜索顺序，比如考虑到

成像过程中，病人在 xy 平面的旋转和平移比其他方向的旋转和平移都要大，故可以先搜索 xy 平面的旋转和平移参数。

(3) 在有些情况下，$S(t)$的全局最优并不带来最好的配准效果，由于相似度函数并不包括配准图像的所有有效信息，两幅图像大面积不匹配时反而比正确匹配时具有更大的相似度。因此要合理选择优化参数的取值范围。

从已有的文献来看，比较多的采用 Powell 方法和梯度下降法进行优化，MITK 中 PowellOptimizer，以及 GradientDescentOptimizer 实现了这两种算法。Powell 的方法比较多的用于基于互信息的配准，而梯度下降法对基于图像灰度均方误差的配准效果比较明显。

MITK 中函数优化模块用 Optimizer 表示，继承自 ObjectProcess 类，其最关键的输入就是优化函数(相似性测度)，这通过函数 void SetMetric(mitkMetric * metric)设定，在 Metric 类的成员函数 float GetSimilarity(const vector<float> * parameters)中得到具体体现。除此之外比较重要的接口函数有：

```
设定各几何变换参数的尺度因子
void SetScales(const vector<float> *scales)
设置初始变换参数
void SetInitialParameters( const vector<double> *param )
```

各种具体的优化算法也同 Metric 模块类似，通过 Optimizer 的派生类在 bool Execute() 函数中实现。如图 5-29 所示，在 MITK 中，我们实现了若干种具有不同寻优能力和计算复杂度的优化算法子类，包括：

- 单纯形优化算法 (Amoeba Optimizer)
- 梯度下降优化算法 (Gradient Descent Optimizer)
- 鲍威尔优化算法 (Powell Optimizer)
- 共轭梯度优化算法 (Conjugate Gradient Optimizer)

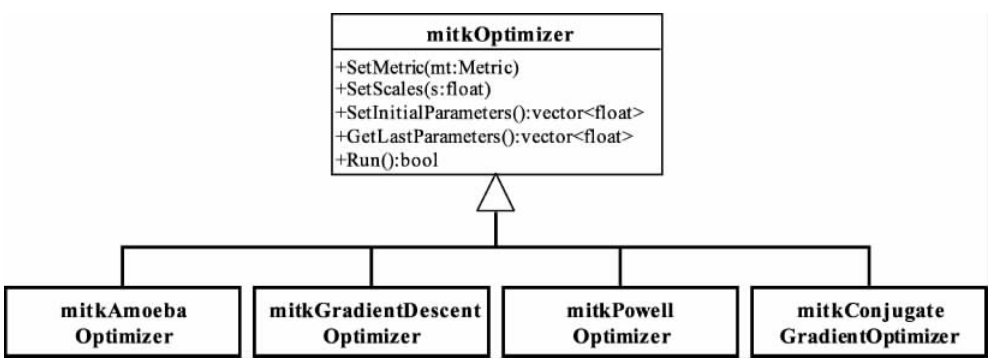


图 5-29 MITK 优化算法类层次结构图

1. 梯度下降最优化

mitkGradientDescentOptimizer 实现了一种梯度下降优化算法。在每次迭代过程中，当前参数位置为上一次迭代的参数坐标在参数空间中沿梯度方向上移动一个步长，如式(5-89)所示，其中 λ 为迭代步长。梯度下降算法实现的关键是

$$p_{k+1} = p_k + \lambda_k \frac{\partial f(x)}{\partial x_k} \tag{5-89}$$

计算图像的梯度，在实践中我们采用了一种基于递归滤波(Recursive Filtering)的近似高斯卷

积(式(5-90))算法来得到图像的梯度。这样做的好处是可以使代价函数的梯度更加平滑同时,配准迭代的计算量较小。

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{5-90}$$

2. 下山单纯形最优化

mitkAmoebaOptimizer 实现了 Nelder-Meade 的下山单纯形最优化算法。单纯形算法的优势在于它不需要计算梯度,在优化过程中,它首先建立一个由 $n+1$ 个顶点构成的单纯形(Simplex),代价函数在每个顶点位置上进行一次计算;随后,算法利用反射(Reflection)、膨胀(Expansion)、收缩(Contraction)等操作对单纯形进行变换,搜索代价函数的最小值。

5.4 基于点集的图像配准原理与实现

相对于基于图像灰度的方法,基于特征的配准方法具有计算量小、配准精度高、可控性强等优点,正如前面 5.2.2 小节所提到的。预先进行的图像分割和特征提取是基于特征配准的前提步骤,而特征的描述方式与存储结构是基于特征配准方法的基础设施。目前,大多数相关研究都将特征点作为配准的依据,因此在 MITK 基于特征的配准算法的实现中,我们将焦点放在了点集配准(PointSet Registration)问题上。

在 MITK 配准框架中,我们实现了两类特征点配准方法:手动标记点配准算法(mitkLandmarkRegistrationFilter)和自动点集配准算法(mitkPointSetRegistrationFilter)。对于特征集的描述,mitkPointSet 类和 mitkKdTreePointSet 类提供了特征点的数据结构和接口算法,如输入输出、包围盒计算、最近邻点索引计算等。特别是 mitkKdTreePointSet 实现了一种基于 kd 树的最近邻点快速计算方法,极大地提高了点集迭代配准算法的运行效率。

mitkLandmarkRegistrationFilter 类封装了一种从点集直接计算图像几何变换参数的标记点配准方法。算法输入用户预先提取或交互得到标记点以及标记点间的一一对应关系,利用最小二乘方法估计出标记点集(配准图像)间的几何变换参数。鉴于稀疏特征点对于复杂变换的描述能力有限,MITK 标记点配准算法只着眼于对包含刚性变换、相似性变换和仿射变换问题的参数估计。

mitkPointSetRegistrationFilter 类实现了一个基于点集的自动图像配准框架。该框架由 4 个功能模块组成:优化、参数估计、几何变换和点集距离测度。优化模块是点集配准框架的核心,整个点集间的迭代收敛过程都由其来组织完成,包括初始几何变换估计、点集间对应关系的计算、算法收敛判断等。参数估计模块根据点集间的对应关系对几何变换参数进行闭型(Closed-form)计算。几何变换模块将由参数估计模块得到的几何变换作用于浮动点集(Moving PointSet)。距离测度模块计算参考点集(Fixed PointSet)和浮动点集间的距离测度,并将测度结果反馈给优化模块。

5.4.1 基于 kd 树的点集描述

在介绍 kd 树理论之前,先给出一些定义[62]。已知两个多维空间域 Domain$=\mathcal{R}^{k_d}$ 和范围 Range$=\mathcal{R}^{k_r}$,将标本(Exemplar)定义为 Domain×Range 的成员,有限的标本的集合定义为标本集。已知标本集 E 与目标域向量 d,则 d 的一个最近邻是对任意标本 $(d',r')\in E$ 满足 Nonnearer(E,d,d')。注意有可能有多个适合的标本。此不定性反映了问题的要求:任何最近邻

均可。Non-nearer 如下定义。

$$\text{Non-nearer}(E,d,d') \Leftrightarrow \forall (d'',r'') \in E \mid d-d' \mid \leqslant \mid d-d'' \mid \tag{5-91}$$

式(5-91)中距离测度为欧氏距离,虽然可以使用任何其他 L_p 范数。

$$\mid d-d' \mid = \sqrt{\sum_{i=1}^{i=k_d} (d_i - d'_i)^2} \tag{5-92}$$

其中 d_i 是向量 d 第 i 个分量。

kd 树是一种用来存储 k 维空间中有限点集的数据结构。kd 树是一种二叉树。表 5-5 表示每个节点中包含的内容。

表 5-5 kd 树节点中的字段

字段名称	字段类型	描 述
dom-elt	域向量	k_d 维空间中的一个点
range-elt	范围向量	k_r 维空间中的一个点
split	整数	划分维度
left	kd 树	表示划分平面左侧的点的 kd 树
right	kd 树	表示划分平面右侧的点的 kd 树

标本集 E 表示为 kd 树中的节点集,每个节点表示一个标本。dom-elt 字段表示标本的域向量,range-elt 字段表示标本的范围向量。dom-elt 分量是节点的坐标。根据节点处的划分超平面将空间划分成两个子空间。所有在“左侧”子空间中的点都由 left 子树表示,在“右侧”子空间的点由 right 子树表示。划分超平面是经过 dom-elt 的平面,与由 split 字段确定的方向垂直。令 split 字段的值为 i,则一个点在 dom-elt 的左侧当且仅当它的第 i 个分量小于第 i 个 dom-elt 分量,互补的定义对右侧区域成立。如果一个节点没有子节点,则不需要划分超平面。

图 5-30 给出了 4 个 dom-elt 点(2,5)、(3,8)、(6,3)、(8,9)的 kd 树的表示。dom-elt 值为(2,5)的根节点将平面在 y 轴方向分为两个子空间。点(6,3)位于下方的子空间$\{(x,y) \mid y<5\}$,所以在左子树中。图 5-31 给出了节点划分平面的方法。

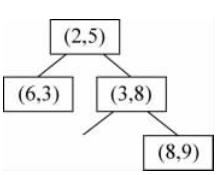


图 5-30 一棵具有 4 个节点的 2d 树

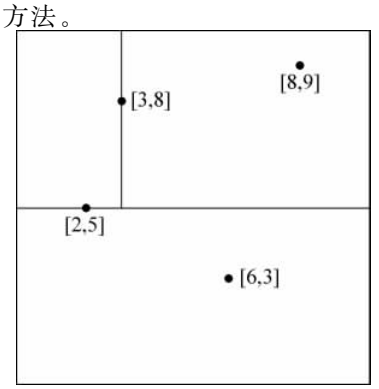


图 5-31 在图 5-30 中的树划分平面的方法

划分平面没有明确表示出来。节点(2,5)沿 $y=5$ 划分平面,节点(3,8)沿 $x=3$ 划分平面。

给定标本集 E,可按照表中所列的算法构造 kd 树。第 2 步中选择 pivot 步骤先检查标本集并从中选择“好”的域向量作为树的根节点。无论选择哪个标本作为根节点都不会影响 kd 树的正确性,但是树的最大深度与超区域的形状会受到影响。选择根节点的标准有两条:构

造出的树尽可能平衡的子节点对应的超区域基本具有相同比例。第一条标准很重要，因为严重不平衡的树访问时时间复杂度可能为 $O(N)$，而不是 $O(\log N)$。第二个标准是为了最大搜索最近邻时的截断概率。选取具有最大方差的维度的中位数的策略可以得到完全平衡的树。但是当遇到标本分布严重不对称的情况时，超区域就会呈现狭长的形状，搜索时有可能无法得到最优解，此时则需要考虑牺牲一部分树的对称性。

下面描述基于 kd 树的最近邻搜索算法。开始时在包含目标点的叶子节点中选取近似最近邻。如图 5-32 所示，目标点用 X 标记，包含目标点的区域对应的叶子节点着黑色。如此例所示，初始近似不必是最近邻，但我们至少知道任何潜在的最近邻必须更接近目标，所以必须在以 X 为中心经过叶子节点的圆内。现在回到当前节点的父节点。图中父节点是黑色的点。计算此父节点的其他子节点中是否可能存在比目前找到的更近的。如图 5-33 所示，此例中是不可能的，因为圆周没有与此父节点其他子节点占据的（加阴影的）区域相交。如果其他子节点中不存在更近邻点，算法即可立即深入更上一层，否则必须递归地考查其他子节点。此例中，检查的下一个父节点需要深入考查，因为它覆盖的面积（即在中部横线上方所有区域）与目前最近邻有交点。

根据上述 kd 树理论，MITK 中实现了基于 kd 树的点集描述 mitkKdTreePointSet 与点集的最邻近查找算法。mitkKdTreePointSet 的类图如图 5-34 所示。主要函数如下。

函数 Initialize()：初始化点集（构造 kd 树）；

函数 GetNearestPoint(double * query_point, double * result_point, int &i)：取得查询点的一个最邻近点。query_point 为查询点，result_point 返回查询结果点的指针，i 返回查询结果点的索引。

函数 GetKNearestPoints(double * query_point, unsigned int k, vector<int> &indices)：取得查询点的 k 个最邻近点。query_point 为存储在数组中的查询点，k 为要搜索的点的个数，indices 返回包括查询结果点索引的向量。

函数 GetKNearestPoints(vector<double> * query_point, unsigned int k, vector <int> &indices)：取得查询点的 k 个最邻近点。query_point 为存储在向量中的查询点，k 为要搜索的点的个数，indices 为返回的包括查询结果点索引的向量。

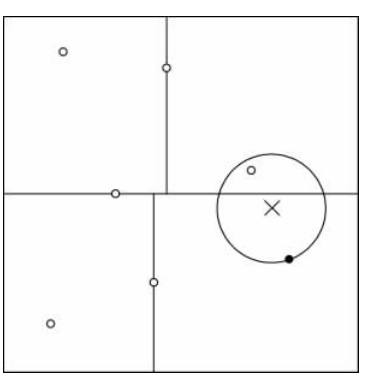


图 5-32 黑点即为包含目标点（叉）的子节点对应的点，任何更近的点必须在此圆内

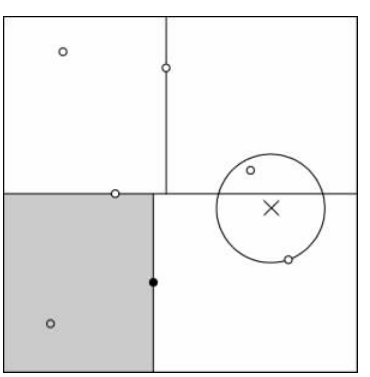


图 5-33 黑点是目前找到的最近的点的父节点。本例中黑点其他的子节点（加灰色阴影）不需要再进行搜索

函数 GetNearestPoints(mitkPointSet * query_pointset, vector<int> &indices)：取得查询点的 k 个最近点。query_pointset 为存储在 mitkPointSet 中的查询点，indices 为返回的包括查询结果点索引的向量。

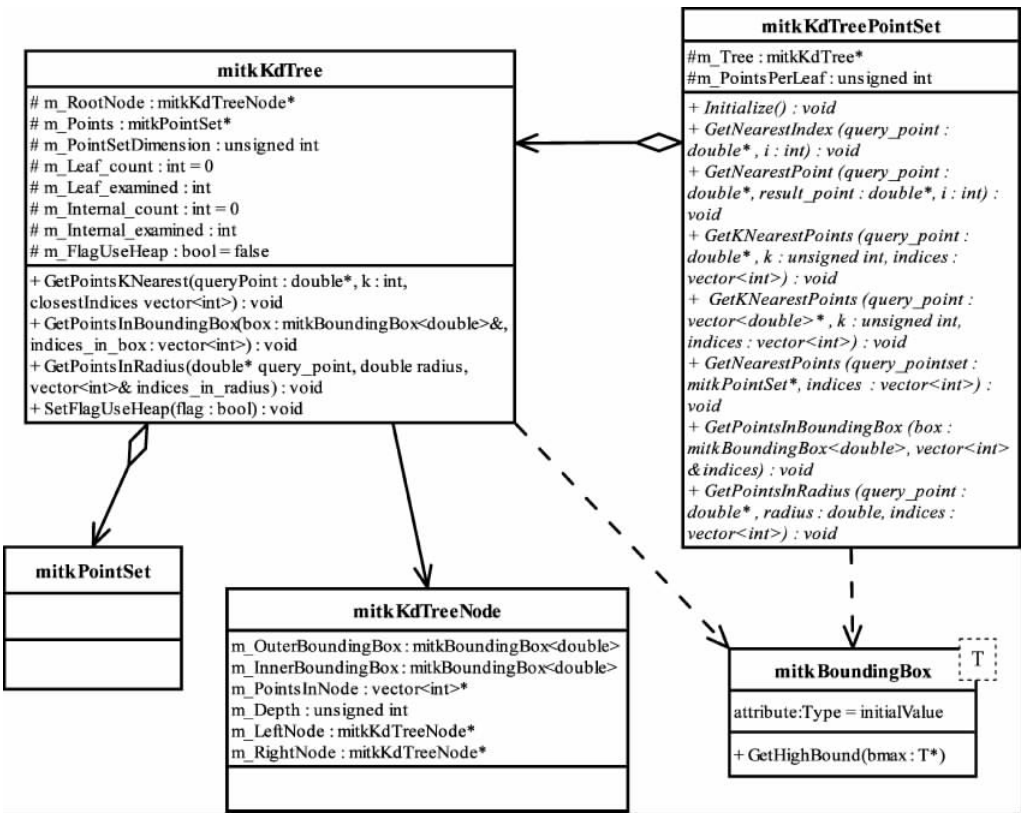


图 5-34 mitkKdTree 类图

函数 GetPointsInBoundingBox(mitkBoundingBox<double>& box, vector<int>&indices)：取得边界立方体内的点。box 为查询的边界立方体，indices 为返回的包括查询结果点索引的向量。

函数 GetPointsInRadius(double* query_point,double radius,vector<int>&indices)：取得以查询点为中心给定半径圆周或球面内的点。query_point 为查询点，radius 给定圆周或球面的半径，indices 为返回的包括查询结果点索引的向量。

最邻近搜索函数 GetPointsKNearest、K_nearest_with_heap，使用栈的 K_nearest_with_stack 函数结构与 K_nearest_with_heap 类似，不再具体分析。

```
// 最邻近搜索算法主函数
void mitkKdTree::GetPointsKNearest(double* queryPoint, int k, vector<int>& closestIndices)
{
    // 如果使用近似查询，则必须使用堆
    int max_leaves = 4;
    vector<double> sq_distances(k,1e+10);   // 可以缓冲以获取(少许)效率提升
    int num_found = 0;                      // 初始化查询结果与中间变量
    m_Leaf_examined = m_Internal_examined = 0;
    if ( m_FlagUseHeap )                    // 如果使用堆的标记非 0,则使用堆；否则使用栈
        this->K_nearest_with_heap(queryPoint,k,m_RootNode,closestIndices,sq_distances,num_
found,max_leaves );
    else
        this->K_nearest_with_stack(queryPoint,k,m_RootNode,closestIndices,sq_distances,num_
found );
```

```
    assert( num_found >= 0 );                    // 检查结果是否正确(非负),否则终止报错
    sq_distances.clear();
}
// 使用堆的 K 近邻算法
void mitkKdTree::K_nearest_with_heap
        (double* query_point,                    // 查询点
        int k,                                   // 搜索 k 个最近点
        mitkKdTreeNode* root,                    // kd 树根节点
        vector<int>& closest_indices,            // k 个最近点索引
        vector<double>& sq_distances,            // qp 与 nnp 之间的距离
        int& num_found,                          // 找到的点的个数
        int max_leaves)                          // 最大的叶子节点序号
{
    vector<mitk_kd_heap_entry> heap_vec;
    heap_vec.reserve( 100 );                     // 设置堆容量远大于所需量以避免错误
    double left_box_sq_dist,right_box_sq_dist;
    double sq_dist;
    // 沿着树从根节点向叶子节点深度优先查询
    mitkKdTreeNode* current = root;
    bool inside;
    while ( current->m_LeftNode )                //只要左节点非空,一直进行深度优先查询
    {
        m_Internal_examined ++;
        // 判断查询点是否在左节点划分区域内
        if ( mitk_point_dist_sq( query_point,current->m_LeftNode->m_OuterBoundingBox,
inside) < 1.0e-5 )
        {
            // 计算查询点到对应右节点外边界的距离并加入向量以备后续检查
            right_box_sq_dist = mitk_point_dist_sq( query_point,current->m_RightNode->m_
InnerBoundingBox,inside);
            heap_vec.push_back( mitk_kd_heap_entry( right_box_sq_dist,current->m_RightNode
) );
            // 前进到左节点
            current = current->m_LeftNode ;
        }
        else // 否则属于右节点划分区域
        {
            // 计算查询点到对应左节点外边界的距离并加入向量以备后续检查
left_box_sq_dist = mitk_point_dist_sq( query_point, current->m_LeftNode->m_
InnerBoundingBox,inside);
            heap_vec.push_back( mitk_kd_heap_entry( left_box_sq_dist,current->m_LeftNode
) );
            // 前进到右节点
            current = current->m_RightNode ;
        }
    }
    // 为沿途经过为选择的子节点比较距离构建堆
    make_heap( heap_vec.begin(),heap_vec.end() );
    sq_dist = 0;
    bool first_leaf = true;
    do
    {
        // 如果找到的个数比 k 小或者目前找到最近的点(距离最大)的距离大于距离
        if ( num_found < k || sq_dist < sq_distances[ num_found-1 ] )
        {
            // 检查左节点是否非空
            if ( ! current->m_LeftNode )
            {
                // 叶子节点
                m_Leaf_examined ++;
```

```
                // 将已找到的最近点排序,将距离大的放在后面
                 Update_closest( query_point,k,current,closest_indices,sq_distances,num_
found );
                // 第一次访问叶子节点
if ( first_leaf )
                {
                    // 检查能否以当前叶子节点返回
                    first_leaf = false;
                     if ( this->Bounded_at_leaf( query_point,k,current,sq_distances,num_
found ) )
                    {
                        heap_vec.clear();
                        return;
                    }
                }
                if (max_leaves != -1 && m_Leaf_examined >= max_leaves)
                {
                    heap_vec.clear();
                    return;
                }
            }
            else
            {
                m_Internal_examined ++;        // 检查过的内节点个数加 1
                // 计算查询点到左节点内边界的距离
                left_box_sq_dist = mitk_point_dist_sq( query_point,current->m_LeftNode->m
_InnerBoundingBox,inside);
                // 如果已找到邻近点个数小于 k 或者该左节点距离小于已找到值
                if ( num_found < k || sq_distances[ num_found-1 ] > left_box_sq_dist )
                {
                    // 将该左节点添加进堆
                    heap_vec.push_back( mitk_kd_heap_entry( left_box_sq_dist,current->m_
LeftNode ) );
                    push_heap( heap_vec.begin(),heap_vec.end() );
                };
                // 计算查询点到右节点内边界的距离
                right_box_sq_dist = mitk_point_dist_sq( query_point,current->m_RightNode-
>m_InnerBoundingBox,inside);
                // 如果已找到邻近点个数小于 k 或者该右节点距离小于已找到值
                if ( num_found < k || sq_distances[ num_found-1 ] > right_box_sq_dist )
                {
                    heap_vec.push_back( mitk_kd_heap_entry( right_box_sq_dist,current->m_
RightNode ) );
                    push_heap( heap_vec.begin(),heap_vec.end() );
                }
            }
        }
        if ( heap_vec.size() == 0 ) return;
        else
        {
            pop_heap( heap_vec.begin(),heap_vec.end() );
            sq_dist = heap_vec[ heap_vec.size()-1 ].dist_;
            current = heap_vec[ heap_vec.size()-1 ].p_;
            heap_vec.pop_back();
        }
    }
    while ( true );
}
```

5.4.2 迭代最近邻点配准算法

迭代最近邻点(Iterative Closest Point)算法由 Besl 和 McKay[14]提出,是目前最为流行的点集模型迭代配准方法,被广泛用于机器视觉、遥感图像和医学图像处理等众多领域。

ICP 算法的基本思路是:对于从点集 $A(a_i, i=1,2,\cdots,n)$到点集 $B(b_j, j=1,2,\cdots,m)$的配准,第 1 步,算法计算出点集 A 中的任一点 a_i 到点集 B 的欧式最短距离 d(式(5-93)),并获得相应的临时匹配点对(a_i, b_j)。第 2 步,算法

$$d(a_i, B) = \min d(a_i, b_j) \tag{5-93}$$

根据已得到的两个点集间特征点的对应关系,利用最小二乘解法计算出几何变换 T(T 通常为刚体变换,即只包含旋转和平移),使得点集间的距离目标函数 f 达到最小。几何变换 T 随后被作用于点集 A。根据两个点集间的距离误差函数 f(式(5-94)):

(1) 若 f 小于精度阈值,则算法收敛;

(2) 若 f 大于精度阈值,则重复第 1 步,直到算法收敛。

$$f = \frac{1}{N}\sum_{i}^{N} \| b_i - T(a_i) \|^2 \tag{5-94}$$

ICP 算法的优点在于其有机地将特征点匹配和几何变换参数估计结合在一起,通过迭代的方式使得两个点集间的距离逐渐缩小,并最终实现精确配准。然而 ICP 算法也存在很多缺陷,具体可总结为以下三点。

(1) 特征点匹配过程计算效率低,算法的计算复杂度为 $O(m^k n^k)$,其中 k 为点集的几何空间维数;

(2) 算法存在局部收敛问题,因此对于交叠面积较小、几何变换较大的点集(图像),若没有较为接近的初始几何变换估计,配准算法往往无法收敛到全局最优点;

(3) 算法对噪声点敏感,对于含有大量结构噪声的图像,配准的精度会受到较大的影响,有时甚至会使配准失败。

MITK 中实现了迭代最近邻点优化器。mitkIterativeClosestPointOptimizer 的类图如图 5-35 所示,主要变量与函数如下。

函数 GetEstimator():取得变换参数估计器 Estimator,返回 mitkPointSetEstimator * 类型的估计器的指针。

函数 SetEstimator(mitkPointSetEstimator * estimator):设置变换参数估计器 Estimator。

函数 SetInitializedByCentroidsFlag(bool flag):设置开始时将浮动点集的形心平移至固定点集的形心。flag 为真时则以对齐形心开始配准过程。

函数 SetMaxMeanDistance(double val):设置两次迭代之间最大平均距离。如果平均距离低于阈值 val,则迭代结束。

函数中 Execute 包含迭代最近邻点算法主要流程。

```
bool mitkIterativeClosestPointOptimizer::Execute()
{
    this->Initialize();                                    // 初始化
    mitkPointSetMetric* metric = mitkPointSetMetric::SafeDownCast(this->GetMetric());
    mitkTransform* transform = metric->GetTransform();
    // 指向固定点集的指针
    mitkPointSet* fixedPointSet = metric->GetFixedPointSet();
    // 指向浮动点集的指针
```

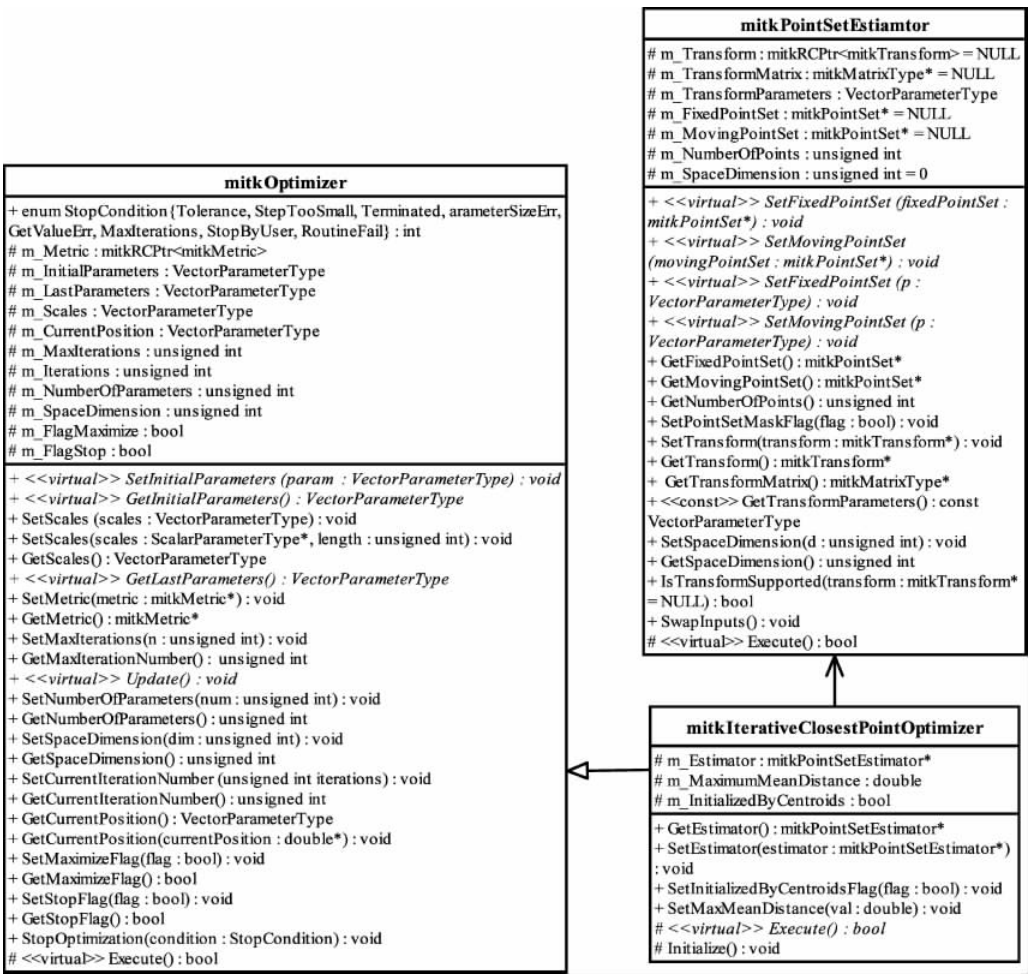


图 5-35 mitkIterativeClosestPointOptimizer 类图

```
mitkPointSet * movingPointSet = metric->GetMovingPointSet();
// 为最终变换备份浮动点集数据
vector<double> * copyMovingPointSetData = movingPointSet->GetPointData();
unsigned int copyMovingPointSetDimension = movingPointSet->GetPointSetDimension();
// 如果对齐形心变换开始标记为真,计算初始变换
if(m_InitializedByCentroids)
{
    // 计算固定点集的形心
    double fixedCentroid[3];
    fixedPointSet->GetCentroid(fixedCentroid);
    // 计算浮动点集的形心
    double movingCentroid[3];
    movingPointSet->GetCentroid(movingCentroid);
    // 更新变换矩阵
    mitkMatrixType * transformMatrix = transform-> GetTransformMatrix();
    transformMatrix->ele[12] = movingCentroid[0] - fixedCentroid[0];
    transformMatrix->ele[13] = movingCentroid[1] - fixedCentroid[1];
    transformMatrix->ele[14] = movingCentroid[2] - fixedCentroid[2];
    transformMatrix->Inverse();                    // 对浮动点集进行初始变换
    transform->TransformPointSet(movingPointSet);
}
```

```
vector<int>* closestp = new vector<int>(movingPointSet->GetNumberOfPoints(),0);
                                                    // 固定点集的点集掩膜
vector<int>* pre_closestp = new vector<int>;    // 固定点集上一个点集掩膜
double MeanDistance = 0;
do
{
    *pre_closestp = *closestp;
    if(fixedPointSet->GetPointSetMaskFlag() == true)
    {
        fixedPointSet->SetPointSetMaskFlag(false);
    }
    // 求固定点集与浮动点集之间的匹配
    metric->GetClosestPointPairs(closestp);
    if( *pre_closestp == *closestp)
    {
        this->StopOptimization(Terminated);
    }
    if( this->GetStopFlag() )
    {
        break;
    }
    fixedPointSet->SetPointSetMask(closestp);
    fixedPointSet->SetPointSetMaskFlag(true);
    // 避免变换矩阵计算中的零向量
    if((*closestp)[0] == (*closestp)[1] && movingPointSet-> GetNumberOfPoints() < 10)
    {
        double* fixPointSetPtr = fixedPointSet->GetPointDataPtr();
        fixPointSetPtr[fixedPointSet->GetPointSetDimension()*(*closestp)[1]]++;
    }
    // 使用四元数法计算变换矩阵
    m_Estimator->Run();
    // 求出变换后固定点集与最邻近浮动点集之间的相似性
    MeanDistance = metric->GetPointSetSimilarity();
    this->SetProgressRate((unsigned long)m_Iterations);
    this->UpdateObservers();
    m_Iterations++;
    if( m_Iterations == m_MaxIterations)
    {
        this->StopOptimization(MaxIterations);
    }
    if( this->GetStopFlag() )
    {
        break;
    }
    // 变换浮动点集
    fixedPointSet->SetPointSetMaskFlag(false);
    transform->TransformPointSet(movingPointSet);
}
while(1);
this->SetProgressRate((unsigned long)m_MaxIterations);
this->UpdateObservers();
fixedPointSet->SetPointSetMaskFlag(true);
// 估计最终变换参数
movingPointSet->SetPointData(copyMovingPointSetData,copyMovingPointSetDimension);
m_Estimator->Run();
// 计算相似性与最终变换参数
metric->GetPointSetSimilarity();
transform->ConvertTransformMaxtrixToVector();
m_LastParameters = metric->GetTransformParameters();
// 释放内存
```

```
    delete copyMovingPointSetData;
    delete closestp;
    delete pre_closestp;
    return true;
}
```

5.4.3 几何变换参数的最小二乘估计

在基于特征的图像配准过程中，当得到了点集间的对应关系并确定了几何变换模型之后，下一步是要知道两个点集所在的子空间的映射关系参数。这一步一般可以利用解线性方程组或线性回归等数学方法得到。在多数情况中，人们更关心两个图像子空间的旋转关系，如刚性配准，此时可以利用奇异值分解(SVD)或单位四元数(Unit Quaternion)的方法来计算旋转变换矩阵。对于上述问题，Umeyama[33]利用拉格朗日乘子法给出了一个严格的闭形解法(Closed-form Solution)。

设 $A\in R^{m\times p}$，$B\in R^{m\times p}$，在正交 Procrustes 问题中，旋转矩阵满足正交性 $R^TR=I$，要知道 B 能否旋转为 A，可求解如下问题。

$$\min_R \| A-RB \|^2 = \mathrm{tr}(A^TA)+\mathrm{tr}(B^TB)-2\mathrm{tr}(R^TB^TA) \tag{5-95}$$

等价于求取 $\mathrm{tr}(R^TB^TA)$的最大值，而使 $\mathrm{tr}(R^TB^TA)$最大化的 R 可以通过求 B^TA 的 SVD 解出。于是对于具有对应关系的点集 $A=\{a_i\}$，$B=\{b_i\}(i=0,1,2,\cdots,N-1)$，利用 SVD 求其的旋转矩阵可分三步完成。

(1) 计算正交矩阵 M；

$$M=\sum_{i=0}^{n}(x_i-\bar{x})(y_i-\bar{y})^T \tag{5-96}$$

(2) 对 M 进行 SVD 分解；

$$\mathrm{SVD}(M)=\mathrm{UDV}^T \tag{5-97}$$

(3) 计算点集间的旋转矩阵 R 及平移参数 t、缩放参数 s。

$$R=\mathrm{USV}^T \tag{5-98}$$

$$t=\bar{y}-sR\bar{x} \tag{5-99}$$

$$s=\frac{1}{\frac{1}{n}\sum_{i=1}^{n}\| x_i-\bar{x} \|^2}\mathrm{tr}(\mathrm{DS}) \tag{5-100}$$

MITK 中实现了两种点集变换估计器 mitkPointSetSVDEstimator 与 mitkPointSetQuaternionEstimator。在此仅对 mitkPointSetSVDEstimator 介绍。

mitkPointSetSVDEstimator 的类图如图 5-36 所示。主要函数如下。

函数 IsTransformSupported(mitkTransform * transform = NULL)：判断变换 transform 是否是 MITK 支持的变换类型。

函数 Execute()：运行 SVD 估计。

函数 Execute()中包含 SVD 估计主要流程。

```
bool mitkPointSetSVDEstimator::Execute()
{
    if ( !(mitkPointSetEstimator::Execute()) )
    {
        return false;
```

```
}
int i;
int m_ImageDimension = 3;
int transformMode = this->GetTransform()->GetTransformMode();
// 如果没有匹配点对,停止退出
if ( m_NumberOfPoints == 0 )
{
    m_TransformMatrix->IdentityMatrix();
    return false;
}
// 求出每个点集的形心
double source_centroid[3];
double target_centroid[3];
m_FixedPointSet->GetCentroid(source_centroid);
m_MovingPointSet->GetCentroid(target_centroid);
// 如果只有一组点对,在此停止
if ( m_NumberOfPoints == 1 )
{
    m_TransformMatrix->IdentityMatrix();
    this->m_TransformMatrix->ele[12] = target_centroid[0] - source_centroid[0];
    this->m_TransformMatrix->ele[13] = target_centroid[1] - source_centroid[1];
    this->m_TransformMatrix->ele[14] = target_centroid[2] - source_centroid[2];
    return true;
}
// 建立 3×3 矩阵 M
double M[3][3];
double AAT[3][3];
for( i = 0; i < 3; i++)
{
    AAT[i][0] = M[i][0] = 0.0F; // 用 0 初始化 M
    AAT[i][1] = M[i][1] = 0.0F;
    AAT[i][2] = M[i][2] = 0.0F;
}
double a[3],b[3];
double sa = 0.0F,sb = 0.0F;
for(unsigned int pt = 0; pt < m_NumberOfPoints; pt++)
{
    // 计算原始源点集形心(a)
    m_FixedPointSet->GetPoint(pt,a);
    a[0] -= source_centroid[0];
    a[1] -= source_centroid[1];
    a[2] -= source_centroid[2];
    // 计算原始目标点集形心(b)
    m_MovingPointSet->GetPoint(pt,b);
    b[0] -= target_centroid[0];
    b[1] -= target_centroid[1];
    b[2] -= target_centroid[2];
    // 将积 a * T(b)在 M 中积分
    for( i = 0; i < 3; i++)
    {
        M[i][0] += a[i] * b[0];
        M[i][1] += a[i] * b[1];
        M[i][2] += a[i] * b[2];
    }
    // 积分尺度因子(如果需要)
```

```
        sa += a[0]*a[0] + a[1]*a[1] + a[2]*a[2];
        sb += b[0]*b[0] + b[1]*b[1] + b[2]*b[2];
    }
    else
    {
        // 计算尺度因子(如果需要)
        double scale = (double)sqrt(sb/sa);
        double w[3],U[3][3],VT[3][3],UxVT[3][3],R[3][3];
        SingularValueDecomposition3x3(M,U,w,VT);
        Multiply3×3(U,VT,UxVT);
        double det_UxVT = Determinant3x3(UxVT);
        for(i=0;i<3;i++)
        {
            U[2][i] = U[2][i] * det_UxVT;
        }
        Multiply3×3(U,VT,R);
        this->m_TransformMatrix->ele[0] = R[0][0];
        this->m_TransformMatrix->ele[1] = R[0][1];
        this->m_TransformMatrix->ele[2] = R[0][2];
        this->m_TransformMatrix->ele[4] = R[1][0];
        this->m_TransformMatrix->ele[5] = R[1][1];
        this->m_TransformMatrix->ele[6] = R[1][2];
        this->m_TransformMatrix->ele[8] = R[2][0];
        this->m_TransformMatrix->ele[9] = R[2][1];
        this->m_TransformMatrix->ele[10] = R[2][2];
        // 加入尺度因子(如果需要)
        if ( transformMode != MITK_TRANSFORM_RIGID)
        {
            for( i = 0; i < 3; i++)
            {
                this->m_TransformMatrix->ele[i*4 + 0] *= scale;
                this->m_TransformMatrix->ele[i*4 + 1] *= scale;
                this->m_TransformMatrix->ele[i*4 + 2] *= scale;
            }
        }
    }
    // 平移变换由变换后源点集形心与目标点集形心之差求出
    double sx,sy,sz;
    sx = this->m_TransformMatrix->ele[0] * source_centroid[0] + this->m_TransformMatrix
->ele[4] * source_centroid[1] + this->m_TransformMatrix->ele[8] * source_centroid[2];
    sy = this->m_TransformMatrix->ele[1] * source_centroid[0] + this->m_TransformMatrix
->ele[5] * source_centroid[1] + this->m_TransformMatrix->ele[9] * source_centroid[2];
    sz = this->m_TransformMatrix->ele[2] * source_centroid[0] + this->m_TransformMatrix
->ele[6] * source_centroid[1] + this->m_TransformMatrix->ele[10] * source_centroid[2];
    this->m_TransformMatrix->ele[12] = target_centroid[0] - sx;
    this->m_TransformMatrix->ele[13] = target_centroid[1] - sy;
    this->m_TransformMatrix->ele[14] = target_centroid[2] - sz;
    // 填充4×4矩阵的最下一行
    this->m_TransformMatrix->ele[3] = 0.0;
    this->m_TransformMatrix->ele[7] = 0.0;
    this->m_TransformMatrix->ele[11] = 0.0;
    this->m_TransformMatrix->ele[15] = 1.0;
    m_TransformMatrix->Inverse();
    return true;
}
```

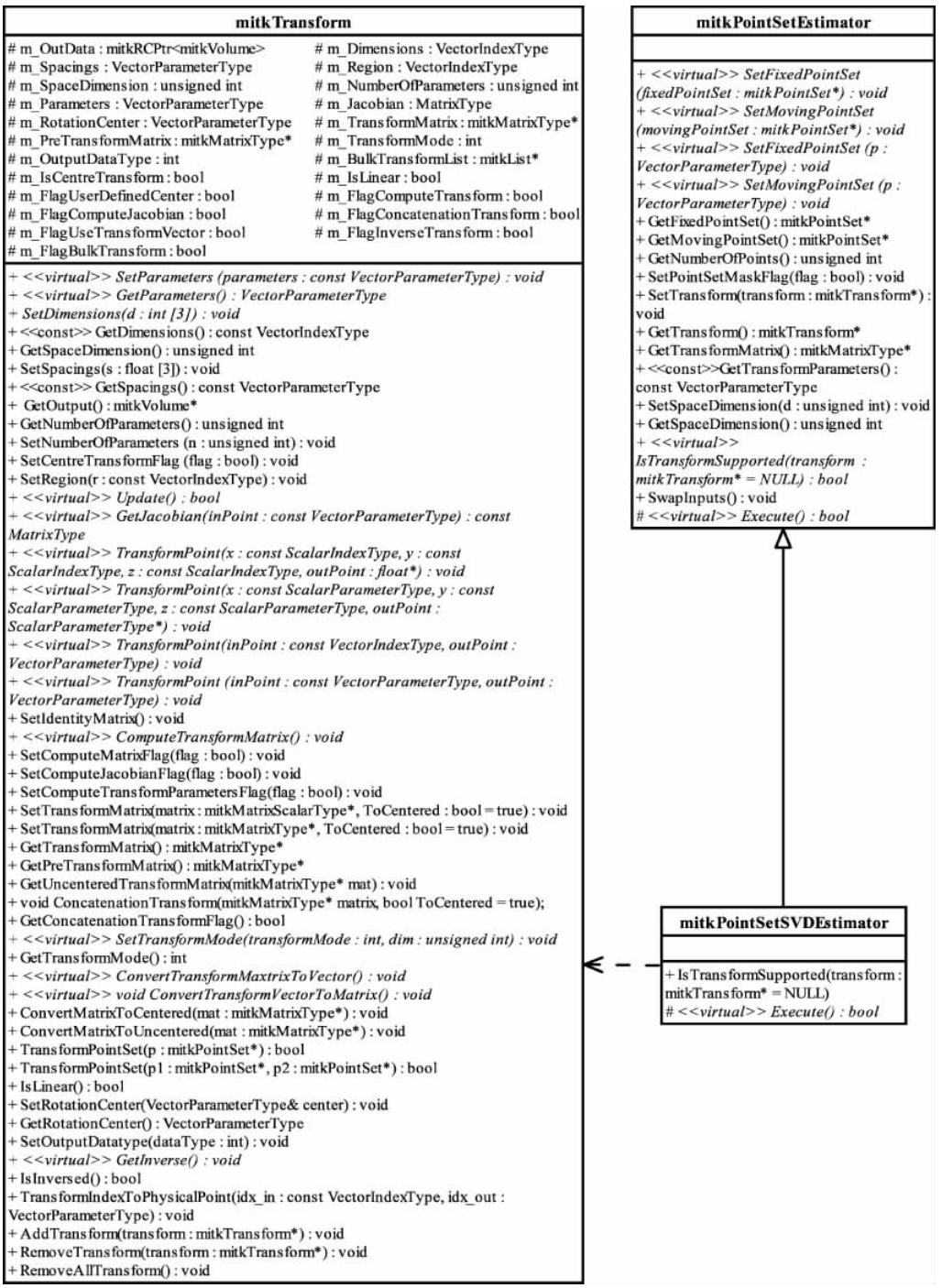


图 5-36　mitkPointSetSVDEstimator 的类图

5.5 基于特征区域的图像配准原理与实现

长期以来，基于图像灰度的密集匹配算法和基于图像点特征的稀疏算法在其特定领域都有着成功的应用和广泛的讨论。但它们都存在着一些不足：传统基于图像灰度的方法计算量非常大，对于图像中非共性结构又较为敏感；而基于点特征的方法虽然可以弥补这方面的不足，但点特征提取和描述算子的稳定性又常常容易受到图像局部噪声的影响。

在本节中我们将思路转向另一个角度来解决图像配准问题中稳定特征提取的问题。我们发现，即使图像的灰度值存在比较大的差异，仍然有一些区域在局部保持相对的显著性，我们称之为显著性特征区域(Salient Feature Region，SFR)。对于这些的显著性特征区域，我们认为它可以直观地理解为图像内容比较复杂(如血管密度比较大、局部结构信息比较多等)的区域。如果能提取并配准图像中的这些稳定的显著性特征区域，整体的图像配准问题也就能迎刃而解了。

围绕显著性区域特征这一问题，本节将简要介绍以下三个方面的理论和算法设计成果，包括：显著性区域特征提取、区域特征匹配以及 MITK 基于区域特征的配准子框架的设计与实现。

5.5.1 区域特征提取

特征区域的提取是计算机视觉领域一个比较基础的研究内容，它相对于特征点提取包含更多的图像信息，因而能够更好地表达图像的内容和结构特征，它在医学图像配准领域也有非常重要的研究及应用价值。通过一些血管结构和视网膜特征提取配准实验，我们发现即使图像的灰度值存在比较大的差异，仍然有一些区域在局部保持相对的显著性，我们称之为显著性特征区域[65,66]，从直观上来讲，SFR 代表图像中那些内容较为丰富的区域。

特征区域提取的流程包括主要三个独立的步骤：对初始图像进行子区域划分并进行局部显著性函数的计算；局部显著性函数拟合；Fls 极值提取及子区域扩张，如图 5-37 所示。

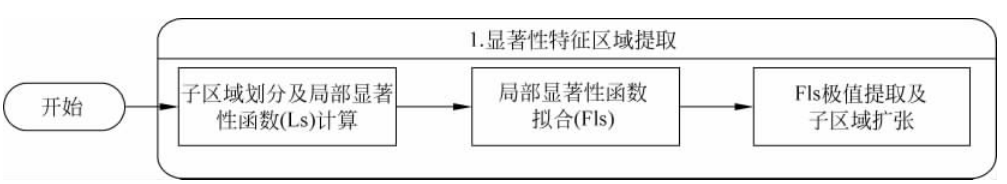


图 5-37 显著性特征区域提取流程图

1. 区域局部显著性函数定义

我们希望我们定义的区域显著性函数能够对图像灰度值变化、背景变化以及病理变化保持比较好的鲁棒性。Kadir[67]等提出了一个基于区域灰度熵的区域显著性函数定义，Kadir 的算法需要遍历图像中的每一个像素，计算在一个区域半径范围$[S_{min}, S_{max}]$内的显著性函数极值确定特征尺度，最后在图像空间搜索局部极值确定特征区域的位置，因此该算法的实际运行时间不能让人满意。

基于 Kadir 等的算法，我们认为区域的局部显著性可以由两方面构成。

(1) 局部灰度值差异显著性，由区域内灰度值的方差来描述，方差越大表示区域的灰度值

差异越显著，反之亦然。为了使其能够保持对灰度值变化的鲁棒性，我们采用自适应标准差(Adaptive Standard Deviation，ASD)来描述区域 R 的局部灰度值差异显著性，定义如下：

$$\mathrm{Asd}(R)=\frac{\sigma}{\mu} \tag{5-101}$$

其中，σ 是区域内灰度值的标准差；μ 是区域的灰度均值。根据式，我们定义的局部灰度值差异显著性函数 $\mathrm{Asd}(R)$ 对灰度值的线性变化保持不变性。

(2) 局部结构复杂度显著性。根据信息论中的定义，熵是描述一个统计变量无序、混乱程度的度量，因此我们采用边缘分布的熵值作为描述局部结构复杂度显著性的度量。为了避免耗时的局部边缘精确分割步骤，我们采用局部梯度场的熵值作为局部结构复杂度显著性的近似度量。在我们的方法中，整个圆周角$[0,2\pi)$被均匀的划分为 36 等份以统计局部梯度场的分布。对于区域 R 中任意一个像素 X_i，梯度方向 $\mathrm{Direction}(X_i)$的定义如式所示：

$$\mathrm{Direction}(X_i)=\begin{cases}\left\lceil\dfrac{\arctan(g(X_i))+\pi/2}{2\pi/36}\right\rceil, & g_x(X_i)\geqslant 0\\ \left\lceil\dfrac{\arctan(g(X_i))+\pi/2}{2\pi/36}\right\rceil, & g_x(X_i)<0\end{cases} \tag{5-102}$$

其中，$\lceil x\rceil$表示不小于 x 的最小整数，$g(X_i)=(g_x(X_i),g_y(X_i))$是 X_i 的梯度向量，区域 R 的局部结构复杂度显著性函数 $\mathrm{Lge}(R)$定义如下：

$$\mathrm{Lge}(R)=-\sum_{i=1}^{36}p_i(R)\log_2 p_i(R) \tag{5-103}$$

式中：

$$p_i(R)=\frac{\int_{R_i}|g(X_i)|\,\mathrm{d}X_i}{\int_R|g(X)|\,\mathrm{d}X} \tag{5-104}$$

$$R_i=\{X_i\mid X_i\in R\wedge \mathrm{Direction}(X_i)=i\} \tag{5-105}$$

在这里，我们采用了幅值加权的梯度场分布来代替传统的基于像素数目的分布。采用幅值加权的策略是为了减小梯度幅值比较小的像素带来的不稳定因素，因为这些像素容易受到灰度值变化以及背景变化的干扰。从式可以看出，我们定义的局部结构复杂度显著性函数 $\mathrm{Lge}(R)$也具有对灰度值线性变化的不变性。

结合区域 R 的局部灰度值差异显著性与局部结构复杂度显著性，我们定义的区域局部显著性函数 $\mathrm{Ls}(R)$如下：

$$\mathrm{Ls}(R)=\mathrm{Asd}(R)*\mathrm{Lge}(R) \tag{5-106}$$

与 Kadir 的区域显著性函数相比，式中一个较大的变化是对于 $\mathrm{Asd}(R)$的引入，引入 $\mathrm{Asd}(R)$的区域显著性函数在图像结构上区分力更大。这一特性使得我们定义的区域显著性函数在遇到图像背景变化、病理变化等导致的低质量图像时，仍然有比较好的鲁棒性。

在定义了区域局部显著性函数的基础上，我们采用了一个有效的子区域扩张策略来提取图像中的显著性特征区域，我们的特征区域提取主要包括以下几个步骤。

1) 子区域划分及其局部显著性函数计算

为了避免全图像空间的遍历搜索以减少计算量，我们首先将图像分成 $M\times N$ 个方形的子区域。子区域的边长是一个经验参数，通常设置为 10 左右的整数以保证子区域的定位精度同时保持其梯度场分布具有统计意义。进一步，我们设置了以下阈值条件来减少不必要的子区

域局部显著性函数计算。

(1) 灰度均值 μ 小于 20 或者自适应标准差 Asd(R)小于 0.04 的子区域可以被认为是背景区域予以排除。

(2) 计算子区域梯度场分布$\{p_i(R)\mid i=1,2,\cdots,36\}$中最大的分量 $p_{i\max}(R)$,如式所示。$p_{i\max}(R)$小于 0.1 的子区域可以被认为是无结构意义的均匀区域予以排除。

$$i\max = \arg\max_i p_i(R), i \in \{1,2,\cdots,36\} \tag{5-107}$$

我们将满足以上阈值条件要求的子区域作为候选特征子区域,根据式计算它们的局部显著性函数 Ls(R)。

2) 局部显著性函数拟合

步骤 1)中得到的局部显著性函数 Ls(R)中一般都存在明显的不连续性以及奇异值,这种不连续性以及奇异值的存在会影响后续的特征区域提取精度。为了消除不连续性及奇异值带来的影响,我们采用高斯函数对局部显著性函数 Ls(R)进行拟合以获得区域显著性函数(Fitted Local Saliency)Fls(R),如下所示。

$$\mathrm{Fls}(R_{ab}) = \sum_{i=1}^{M}\sum_{i=1}^{N}\frac{\mathrm{Ls}(R_{ij})}{2\pi\sigma^2}\exp^{-((a-i)^2+(b-j)^2)/(2\sigma^2)} \tag{5-108}$$

其中,R_{ab}表示在 M 𝓛 N 的子区域阵列中坐标为(a,b)的子区域。我们选择 $\sigma = 1.5$ 的高斯函数进行拟合,以实现中心子区域与周围子区域的局部显著性间的折中,在计算时采用 5×5 的离散化模板进行卷积。与局部显著性函数 Ls(R)相比,拟合后的区域显著性函数 Fls(R)更加光滑、连续性更好,能描述更大范围内的区域显著性。

3) 特征子区域确定及扩张

满足候选特征子区域条件的区域显著性函数 Fls(R)的极值子区域被确定为特征区域的中心子区域,然后我们采用子区域扩张的策略来确定特征区域的大小。具体步骤如下:以中心子区域为初始特征区域,在 $M \times N$ 的子区域阵列中各向同性地向外扩张,寻找满足式(5-109)所述条件的一个最大的方形子区域阵列 Ω。

$$\mathrm{Fls}(R_{ij}) \geqslant \lambda * \mathrm{Fls}(R_{ab}) \quad \forall R_{ij} \in \Omega \tag{5-109}$$

其中,R_{ab}是中心子区域,$\lambda \in [0,1]$是控制子区域扩张的经验参数。我们选择 Ω 的内接圆作为半完成的特征区域,以保持对图像旋转的不变性,最终的特征区域半径在此内接圆的基础上再扩张 1 倍。采用这样的扩张策略是为了吸收邻域内更多的结构信息以增加特征区域的显著性及可区分性,同时使特征区域之间保持大小的一致性。

5.5.2 区域特征匹配

特征描述及匹配是所有基于特征的配准方法的另一个关键步骤,它直接决定算法的性能,如运行时间、配准成功率以及配准精度等。通常特征描述及匹配步骤包含以下三个要素。

(1) 构造一个高维的特征描述子(Feature Descriptor);

(2) 定义一个衡量特征描述子(向量)间相似性的度量函数(Evaluation Metric);

(3) 采用有效的匹配策略(Matching Strategy)完成特征向量间的匹配。

我们提出了一个新颖的特征描述子及对应的相似性度量函数以及由粗到细的有效匹配策略来提高特征区域匹配算法的性能(如图 5-38 所示)。接下来我们详细介绍区域特征描述及匹配步骤中的三个要素,即特征描述子、相似性度量和特征区域匹配。

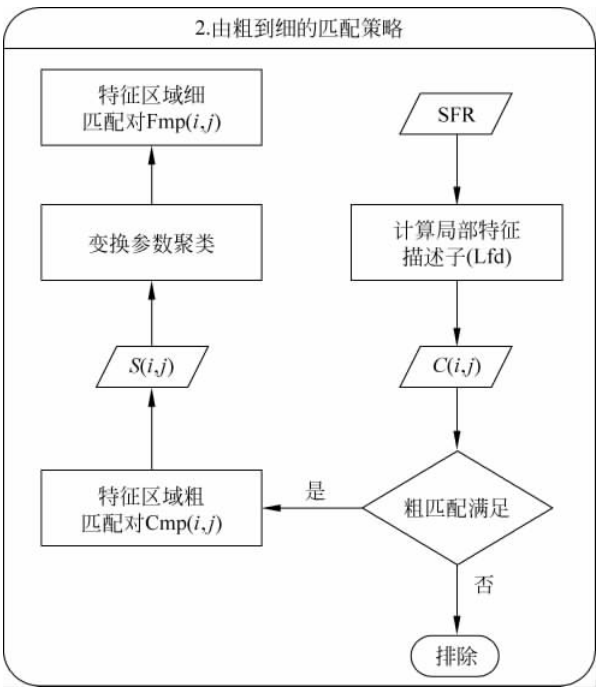


图 5-38　区域特征匹配算法流程图

1. 特征描述子

特征描述子主要可以分为两大类：基于图像内容的描述子，如 SIFT、PCA-SIFT 以及 SURF 等；基于几何结构信息及拓扑结构的描述子。SIFT 以及 SURF 是目前在模式识别领域使用最广泛的特征描述子，这些描述子主要基于区域内统计变量的概率分布构建的，基本不含几何结构信息。因此在描述具有相似分布而不同几何结构的区域特征时，描述子的区分性能会显著低，针对这个不足，我们提出了一个结合区域梯度场分布及对应几何结构信息的 72 维尺度不变特征描述子 Lfd(R)。

$$\mathrm{Lfd}(R)=(p_{\omega 1}(R),\cdots,p_{\omega 36}(R),da_1(R),\cdots,da_{36}(R)) \tag{5-110}$$

特征描述子 Lfd(R)的前 36 个分量是距离平方加权$\{p_{\omega i}(R)\mid i=1,2,\cdots,36\}$的梯度场幅值分布，定义如下。

$$p_{\omega i}(R)=\frac{\int_{R_i}(1-|X_i-X_c|^2/r^2)\mid g(X_i)\mid \mathrm{d}X_i}{\int_{R}(1-|X-X_c|^2/r^2)\mid g(X)\mid \mathrm{d}X} \tag{5-111}$$

其中，X_c 是特征区域R 的中心；r 是区域半径。考虑到特征区域定位的不完全一致性，我们采用距离平方加权来减小远离区域中心的像素在特征计算上的贡献，以提高特征匹配的成功率。

特征描述子 Lfd(R)的后 36 个分量 $da_i(R)\in[0,2\pi)$是从区域中心 X_c 到 R_i 几何中心 X_{ci} 的方向角，描述了同一梯度方向的点集 R_i 的几何结构信息。特征描述子 Lfd(R)的构造过程示意如图 5-39 所示。

特征描述子 Lfd(R)是基于像素的分布信息和相应的角度信息构建的，在理论上对特征区域的尺度变换保持不变性。这使其对于大小不完全一致的特征区域间的匹配具有一定的鲁棒性，并且由于吸收了区域的几何结构信息，使得特征区域匹配算法性能得到了一定程度的

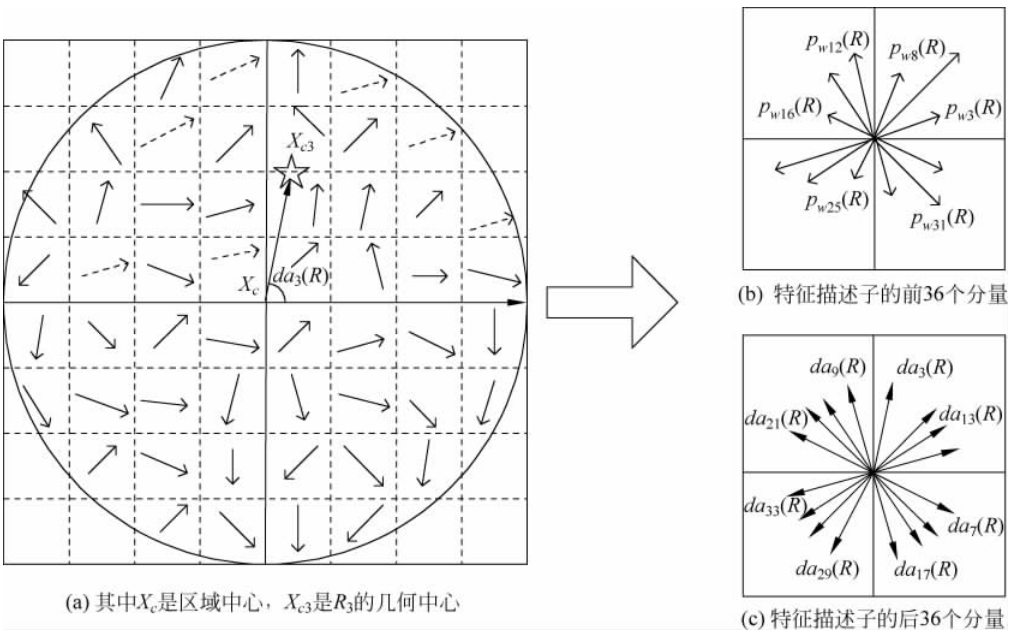


图 5-39 特征描述子 Lfd(R)构造示意

提升。

2. 相似性度量

考虑到特征区域中包括比较丰富的边缘信息，因此在特征描述子中存在几个主要优势方向(Predominant Directions)，如果将所有分量一致对待，采用高维欧氏空间中的距离度量会降低特征描述子的区分度。为了有效地衡量两个特征描述子 Lfd(R_1)与 Lfd(R_2)间的相似性，我们定义了一个基于 K-L 散度的相似性度量函数，如下所示。

$$\begin{aligned}\mathrm{Dist}(\mathrm{Lfd}(R_1),\mathrm{Lfd}(R_2)) = \sum_{i=1}^{36}\Big\{ &\mathrm{Eud}(da_i(R_1),da_i(R_2)) \cdot \max(p_{\omega i}(R_1),p_{\omega i}(R_2)) \\ &\cdot \log\Big(\frac{\max(p_{\omega i}(R_1),p_{\omega i}(R_2))}{\min(p_{\omega i}(R_1),p_{\omega i}(R_2))}\Big)\Big\} \end{aligned} \tag{5-112}$$

式中，Eud($da_i(R_1)$,$da_i(R_2)$)是 $da_i(R_1)$与 $da_i(R_2)$间的夹角。

K-L 散度是一类更广泛的 f-散度中的特例，在描述两个统计变量分布的相似程度上非常有效。K-L 散度不具有对称性，在我们的相似性度量函数中进行了修正。我们采用两个概率分布中的较大值作为分子项及加权项，较小值作为分母项，使其具有对称性，这样在遇到两幅图像的配准逆问题时能保证同样的特征区域匹配结果，有助于保持配准的逆变换连续性。

3. 特征区域匹配

通常提取的视网膜图像特征区域数目在一百以下，特殊情况下(图像尺寸较大、图像结构特征非常丰富)特征区域数目能达到几百的数量级。采用快速、有效的匹配策略对于减少算法运行时间也非常重要，我们提出了一个由粗到细的特征区域匹配流程，具体步骤如下。

1) 特征区域的粗匹配

我们用 $C(i,j)$表示固定图像中的第 i 个特征区域与浮动图像中的第 j 个特征区域间的匹配关系。遍历两幅图像中所有特征区域间的匹配关系 $C(i,j)$，对于满足式(5-113)所述条件的 $C(i,j)$，我们认为它是一个特征区域粗匹配对，记为 $\mathrm{Cmp}(i,j)$。

$$\frac{\min(\mathrm{Asd}(R_i),\mathrm{Asd}(R_j))}{\max(\mathrm{Asd}(R_i),\mathrm{Asd}(R_j))}\cdot\frac{\min(\mathrm{Lge}(R_i),\mathrm{Lge}(R_j))}{\max(\mathrm{Lge}(R_i),\mathrm{Lge}(R_j))}>T \tag{5-113}$$

其中 $T\in[0,1]$是一个全局相似度阈值。

通过全局相似度阈值 T 的限制，特征区域粗匹配对 $\mathrm{Cmp}(i,j)$的数量在特殊情况下也可以控制在几千以下。对于每一个特征区域粗匹配对 $\mathrm{Cmp}(i,j)$，根据式(2-12)计算 R_i 和 R_j 之间的相似度 $S(i,j)$，以及旋转角 θ_{ij}，如下所示。

$$S(i,j)=\mathrm{Dist}(\mathrm{Lfd}(R_i),\mathrm{Lfd}(R_jk)) \tag{5-114}$$

$$\theta_{ij}=\frac{2k\pi}{36} \tag{5-115}$$

其中：

$$k=\arg\min_k(\mathrm{Dist}(\mathrm{Lfd}(R_i),\mathrm{Lfd}(R_jk))),k\in\{0,1,\cdots,35\} \tag{5-116}$$

式中，$\mathrm{Lfd}(R_jk)$是将特征区域 R_j 逆时针旋转 k 个细分角度得到的新区域。与 SIFT 不同，我们并没有为每一个特征区域指定一个主方向，这是因为主方向计算本身就存在着一定的误差，这个误差在一定程度上会降低特征匹配算法的性能。由此导致的额外计算开销是可以接受的，因为 $\mathrm{Cmp}(i,j)$的数目被限制在一个比较低的级同时 $\mathrm{Lfd}(R_jk)$可以很容易地由 $\mathrm{Lfd}(R_j)$通过分量循环平移 k 位得到。在我们的实验中，特征区域粗匹配的运行时间通常可以控制在 1 秒之内。

2) 特征区域的精细匹配

每一个特征区域粗匹配对 $\mathrm{Cmp}(i,j)$可以确定 3 个二维全局刚体变换参数：二维的平移 $[t_x,t_y]$(由对应的特征区域中心确定)，以及旋转角 μ_{ij}。若点$[x,y]$是浮动图像的中心，那么由 $\mathrm{Cmp}(i,j)$确定的刚体变换模型可以表示如下：

$$\begin{bmatrix}u\\v\end{bmatrix}=\begin{bmatrix}\cos\theta_{ij} & -\sin\theta_{ij}\\ \sin\theta_{ij} & \cos\theta_{ij}\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}t_x\\t_y\end{bmatrix} \tag{5-117}$$

其中，$[u,v]$是变换后的浮动图像中心。

我们采用在全局刚体变换参数空间上的聚类(Clustering)分析方法来完成特征区域的精细匹配，主要步骤包括以下几步。

(1) 按 $S(i,j)$的升序排列特征区域粗匹配对 $\mathrm{Cmp}(i,j)$，选择前 N 个$\mathrm{Cmp}(i,j)$ 作为输入的聚类样本，一般 N 取 2000 就可以充分保证视网膜图像配准的成功率。

(2) 对变换后的浮动图像中心，在二维欧氏空间上采用最近邻聚类算法进行分析(类内距离阈值一般设为图像长度的二十分之一)。元素数目最多的类中包含的特征区域粗匹配对 $\mathrm{Cmp}(i,j)$被认为是特征区域细匹配对，记为 $\mathrm{Fmp}(i,j)$。

(3) 对特征区域细匹配对 $\mathrm{Fmp}(i,j)$中可能存在的重复区域，比较它们的相似度 $S(i,j)$，剔除 $S(i,j)$比较大的 $\mathrm{Fmp}(i,j)$，使得 $\mathrm{Fmp}(i,j)$中包含的特征区域 R_i 和 R_j 保持双向一对一映射关系。

5.5.3 MITK-REG-SFR 配准子框架的设计

基于 MITK 的统一设计风格，MITK-REG-SFR 的框架设计遵循以下的目标：在整体上保持与现有图像配准框架的一致性，提供与其他图像配准子框架关联(Association)的良好接口以支持混合配准策略；在内部实现各功能模块间的灵活组合；在程序设计上提供良好的扩展性和组件重用性。

考虑到以上对 MITK-REG-SFR 的设计要求，并对本文工作中基于特征区域的图像配准算法的处理流程进行抽象，我们设计了 6 个相对独立的模块来搭建 MITK-REG-SFR 的框架，如图 5-40 所示，包括以下几个部分。

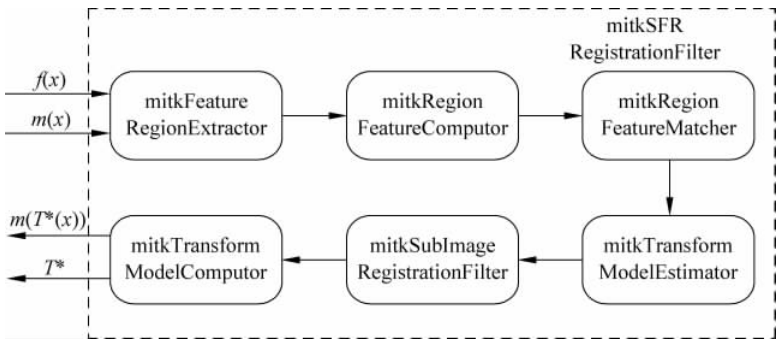


图 5-40 MITK-REG-SFR 算法框架和数据处理流程

(1) 特征区域提取：mitkFeatureRegionExtractor，根据设定的区域显著性函数，对输入的固定图像 $f(x)$ 和浮动图像 $m(x)$ 提取特征区域，输出特征区域的几何信息(位置，大小，主轴方向)。区域显著性函数包括：图像灰度熵、图像梯度熵、图像自适应标准差梯度熵、结合图像梯度场散度的梯度熵。特征区域的形状包括：正方形、圆形(二维图像)、立方体、球体(三维图像)。

(2) 区域特征计算：mitkRegionFeatureComputor，接受输入的特征区域的几何信息，统计特征区域内的像素(体素)的分布信息及几何结构信息，构造一个高维的特征向量来描述图像的特征区域，输出高维特征向量集合。

(3) 区域特征匹配：mitkRegionFeatureMatcher，接受输入的特征向量集合，对两个特征向量集合中的元素进行相似性度量计算，按照一定的匹配准则，进行特征匹配，输出特征向量匹配关系集合。相似性度量包括：K-L 散度度量(Kullback-Leibler Divergence)[82]以及自定义的包含几何结构特征的度量。匹配准则包括：最小距离显著性准则[51]、双边映射一致性准则(参见式(3-25))。

(4) 变换模型估计：mitkTransformModelEstimator，接受输入的特征向量匹配关系集合，进行变换模型的估计并剔除错误的特征向量匹配关系，输出变换模型及正确的特征向量匹配关系集合。变换模型的估计采用基于采样一致性的方法，包括：随机采样一致性(Random Sample Consensus，RANSAC)及有序采样一致性(Ordered Sample Consensus，OSAC)。变换模型可以是简单的刚体变换模型、仿射模型，也可以是高阶的基函数模型。

(5) 子图像配准：mitkSubImageRegistrationFilter，接受输入的变换模型及特征向量匹配关系集合，采用基于内容的图像配准方法对子图像进行局部配准。设定局部配准成功准则，对精确配准后的变换参数，采用基于采样一致性的方法或几何变换空间的参数聚类方法剔除错误的局部配准，输出局部正确、成功配准的变换参数集合及对应的特征区域匹配关系。

(6) 全局变换模型计算：mitkTransformModelComputor，接受输入的局部配准的变换参数集合及对应的特征区域匹配关系，计算全局高阶变换模型及配准后的图像。变换模型可以是简单的刚体变换模型、仿射模型，也可以是高阶的多项式变换模型、基于基函数的形变模型。

上述模块在配准类 mitkSFRRegistrationFilter 中通过相应的接口组合到一起，完成整个基于特征区域的配准算法流程，最后输出图像配准的变换参数 T^* 及配准后的图像 $m(T^*(x))$。

5.6 应用实例与分析

下面我们用三个简单的例子来说明如何使用 MITK 中的配准算法模块来搭建配准应用程序，并通过 3DMed 进行可视化。

5.6.1 基于灰度的三维核磁图像刚性配准

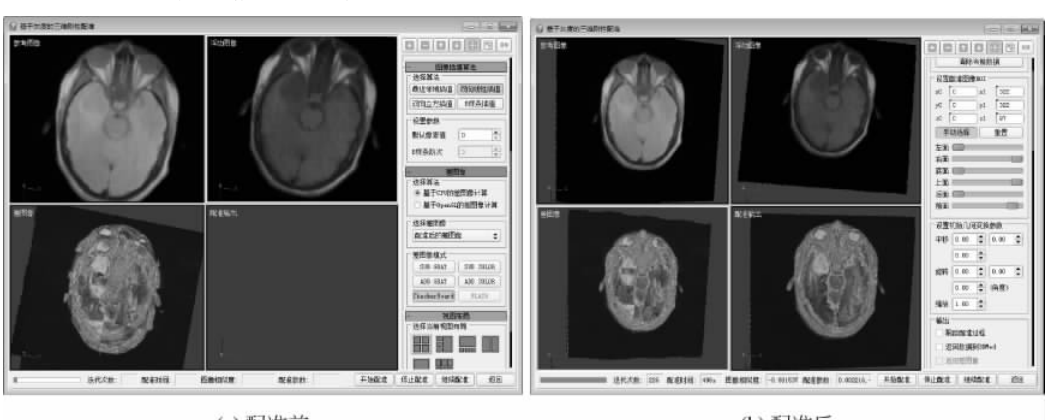
由于临床诊断与研究中，刚性约束是一类非常好的近似；同时其所需要计算的参数也比较少，因此刚性配准在临床医学图像配准中的有着非常广泛的应用，常被用于人脑结构图像与功能图像的配准。在此以三维核磁 PD 加权像与 T1 加权像刚性配准为例。

本例中使用规范化四元数刚性变换模块、互信息测度模块、线性插值模块、梯度下降优化器，输出配准结果同时输出差图像以反映配准结果。

```
#include "mitkRegistrationFilter.h"                    // mitk 配准滤波器
#include "mitkVersorRigid3DTransform.h"                // mitk 规范化四元数刚性变换
#include "mitkMutualInformationMetric.h"               // mitk 互信息测度
#include "mitkLinearInterpolateFilter.h"               // mitk 线性插值器
#include "mitkGradientDescentOptimizer.h"              // mitk 梯度下降优化器
#include "mitkRegistrationObserver.h"                  // mitk 配准监视器
#include "mitkSubtractImageFilter.h"                   // mitk 差图像滤波器
void main()
{
    mitkVolume * fixedVolume = NULL;
    mitkVolume * movingVolume = NULL;
    // 读入图像
    fixedVolume = ReadImageFileToVolume("image1.im0");
    movingVolume = ReadImageFileToVolume("image2.im0");
    // 建立并配置配准滤波器、变换模块、测度模块、内插模块与优化模块
    mitkRegistrationFilter * registration = new mitkRegistrationFilter;
    mitkVersorRigid3DTransform * transform = new mitkVersorRigid3DTransform;
    mitkMutualInformationMetric * metric = new mitkMutualInformationMetric;
    metric->SetMIMethod(MITK_MI_METRIC_HIST_NMI);
    metric->SetHistogramSize(32,32);
    metric->SetNumberOfSpatialSamples(567);
    mitkLinearInterpolateFilter * interpolator = new mitkLinearInterpolateFilter;
    interpolator->SetOutputDatatype(MITK_UNSIGNED_CHAR);
    mitkGradientDescentOptimizer * optimizer = new mitkGradientDescentOptimizer;
    // 设置输入
    registration->SetFixedVolume(fixedVolume);
    registration->SetMovingVolume(movingVolume);
    // 设置配准方法
    registration->SetTransform(transform);
    registration->SetInterpolator(interpolator);
    registration->SetMetric(metric);
    registration->SetOptimizer(optimizer);
    // 设置配准监视器
```

```
    registration->AddObserver(new mitkRegistrationObserver(registration));
    // 设置优化参数
    optimizer->SetMaxIterations(60);
    optimizer->SetGradientMagnitudeTolerance(1e-4);
    optimizer->SetMaximumStepLength(1.000);
    optimizer->SetMinimumStepLength(0.0001);
    // 设置初始变换参数
    double parameters[] = {0,0,0,0,0,0};            // 三维刚性
    registration->SetInitialParameters(parameters,sizeof(parameters)/sizeof(double));
    // 设置初始尺度
    double scales[] = {1,1,1,0.001,0.001,0.001};   // 三维刚性
    optimizer->SetScales(scales,sizeof(scales)/sizeof(double));
    // 定义配准区域
    int region[] = {30,180,30,127,30,127};
    registration->SetRegisrationRegion(region);
    // 进行配准
    registration->Run();
    // 取得配准结果
    mitkVolume* outVolume = registration->GetOutput();
    // 计算棋盘差异
    mitkSubtractImageFilter* subtract = new mitkSubtractImageFilter(MITK_SUBSTRACT_MODE_
CHECKERBOARD);
    subtract->SetInput1(fixedVolume);
    subtract->SetInput2(movingVolume);
    subtract->SetCheckImageDim(8,8);
    subtract->SetOutputDatatype(MITK_UNSIGNED_SHORT);
    subtract->Run();
    mitkVolume* subtractVolume = subtract->GetOutput();
    // 将输出与差图像写到文件
    WriteVolumeToImageFile("output_image.im0",outVolume);
    WriteVolumeToDICOMFile("check_image.dicom",subtractVolume);
    // 释放内存
    registration->Delete();
    subtract->Delete();
}
```

通过 3DMed 三维刚性配准插件可以很方便地对配准流程进行实时交互以及对配准数据进行联合可视化。图 5-41 为本实例三维脑核磁图像配准实验结果截图。图 5-42 为本实例迭代优化过程中,图像互信息测度值曲线。



(a) 配准前　　(b) 配准后

图 5-41　3DMed 三维刚性配准插件界面

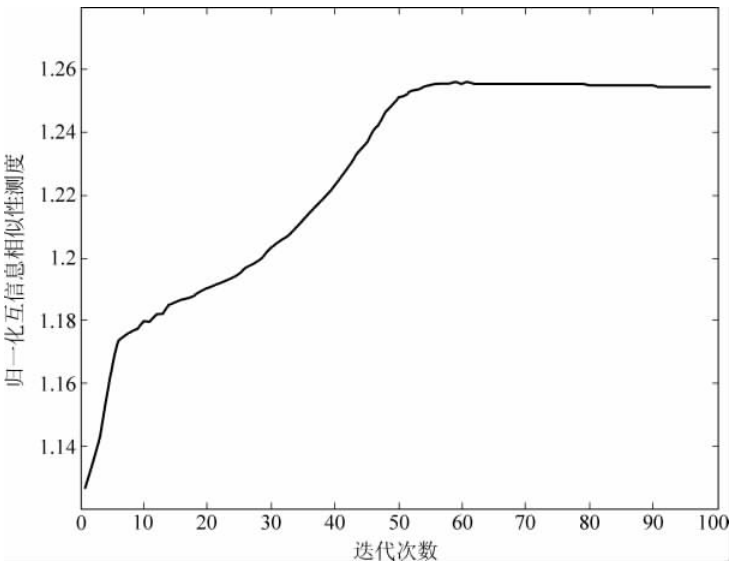


图 5-42　归一化互信息随迭代次数增加而收敛

5.6.2　基于标记点的弹性图像配准

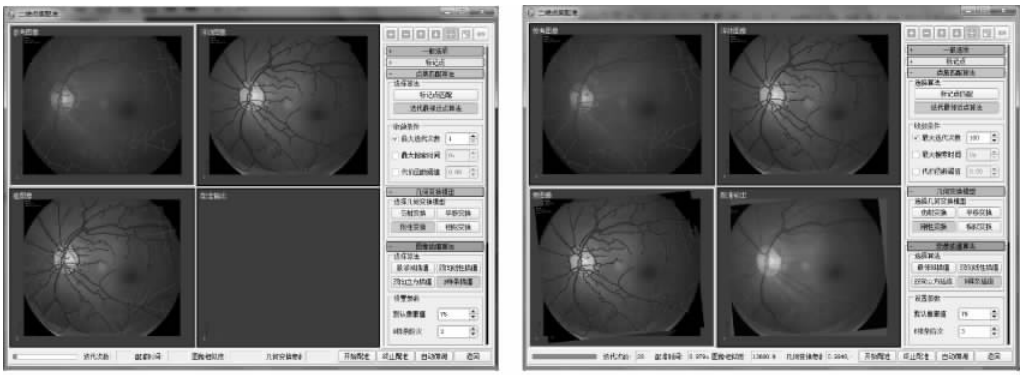
在临床眼底疾病诊断与治疗方面，图像配准技术越来越重要。对两张或以上在不同时间、不同角度以及用不同的传感器拍摄的眼底图像进行快速全自动的配准算法存在巨大的需求。眼底图像配准是一个挑战性的任务：第一，眼底是曲面，使用未校准的弱透视（Weak-Perspective）相机可能会导致非线性形变；第二，由于巨大的视角变换图像重叠面积可能很小；第三，眼底图像可能存在大面积非均匀光照无纹理区域，使得提取眼底血管非常困难；第四，较长时间间隔或从患病眼球中拍摄的两幅图像可能在结构和眼底颜色上都有生理性的变化。在这里，以半自动的基于 B 样条的弹性图像配准方法为例，其中标记点使用 3DMed 配准插件手动提取。

本例中使用薄板样条变换模块、欧氏距离测度模块、B 样条内插模块、点集四元数估计模块，最后输出配准结果。

```
#include "mitkPointSetRegistrationFilter.h"        // mitk 点集配准滤波器
#include "mitkIterativeClosestPointOptimizer.h"    // mitk 点集匹配优化器
#include "mitkThinPlateSplineTransform.h"          // mitk 薄板样条变换器
#include "mitkEuclideanDistanceMetric.h"           // mitk 欧氏距离测度
#include "mitkBSplineInterpolateFilter.h"          // mitk B 样条内插器
#include "mitkPointSetQuaternionEstimator.h"       // mitk 点集四元数估计器
#include "mitkPointSetSVDEstimator.h"              // mitk 点集 SVD 估计器
#include "mitkRegistrationObserver.h"              // mitk 配准监视器
#include "mitkPointSet.h"                          // mitk 点集类
void main()
{
    unsigned int ImageDimension = 2;
    mitkVolume * fixedVolume = NULL;
```

```
    mitkVolume * movingVolume = NULL;
    // 读入图像
    fixedVolume = ReadImageFileToVolume("image1.im0");
    movingVolume = ReadImageFileToVolume("image2.im0");
    // 加载标记点
    mitkPointSet * fixedPointSet = new mitkPointSet;
    fixedPointSet->ReadPointDataFromFile("landmark_set1.txt",3,MITK_FLOAT);
    mitkPointSet * movingPointSet = new mitkPointSet;
    movingPointSet->ReadPointDataFromFile("landmark_set2.txt",3,MITK_FLOAT);
    // 配置点集匹配算法为最近邻点匹配
    mitkIterativeClosestPointOptimizer * optimizer = new mitkIterativeClosestPointOptimizer;
    // 配置变换为薄板样条变换
    mitkThinPlateSplineTransform * transform = new mitkThinPlateSplineTransform(ImageDimension);
    // 配置测度为欧氏距离
    mitkEuclideanDistanceMetric * metric = new mitkEuclideanDistanceMetric;
    // 配置估计器为四元数估计器
    mitkPointSetQuaternionEstimator * estimator = new mitkPointSetQuaternionEstimator;
    // 配置内插器为线性内插器
    mitkLinearInterpolateFilter * interpolator = new mitkLinearInterpolateFilter;
    interpolator->SetDefaultPixelValue(0.0f);
    interpolator->SetOutputDatatype(MITK_UNSIGNED_CHAR);
    mitkPointSetRegistrationFilter * registration = new mitkPointSetRegistrationFilter;
    // 设置输入程序
    registration->SetFixedVolume(fixedVolume);
    registration->SetMovingVolume(movingVolume);
    // 设置输入点集
    registration->SetFixedPointSet(fixedPointSet);
    registration->SetMovingPointSet(movingPointSet);
    // 设置配准变换、内插、测度、估计、优化、观察者模块
    registration->SetTransform(transform);
    registration->SetInterpolator(interpolator);
    registration->SetMetric(metric);
    registration->SetEstimator(estimator);
    registration->SetOptimizer(optimizer);
    registration->AddObserver(new mitkRegistrationObserver(registration));
    // 进行点集配准
    registration->Run();
    // 获取配准结果
    mitkVolume * outVolume = registration->GetOutput();
    // 输出图像到文件
    WriteVolumeToImageFile("output.im0",outVolume);
    // 释放内存
    registration->Delete();
}
```

如图 5-43 3DMed 二维点集配准插件界面截图所示，本实例输入两幅不同时间采集的视网膜眼底图像，并输入两个预先分割的视网膜血管点集数据，通过点集迭代匹配算法将两个点集对齐，并输出经过几何变换和图像重采样后的配准图像。



(a) 配准前　　(b) 配准后

图 5-43　3DMed 二维配准插件界面

5.6.3　基于 B 样条自由形变模型的非刚性图像配准

十几年来，大脑图像的非刚性配准一直是活跃的研究领域。该领域具有广泛的临床应用，特别地，用于群体分析与统计参数映射(Statistical Parametric Mapping)的功能图像的空域归一化。作为一种结构度量方法，通过将解剖模板与个体解剖结构比对，大脑图像的非刚性配准在计算解剖学方面也有应用。在损伤-缺损研究中也可作为一种图像数据挖掘方法，还可用来立体定位神经外科中将解剖图谱映射到患者图像。这里使用基于 B 样条自由形变模型的非刚性图像配准的方法。

本例中使用 B 样条自由变形变换模块、均方测度模块、线性插值模块、适用于边界约束问题有限内存拟牛顿法 L-BFGS-B 优化模块，最后输出配准结果与形变场图像。

```
#include "mitkRegistrationFilter.h"                    // mitk 配准滤波器
#include "mitkBSplineFreeFormTransform.h"              // mitk B 样条自由变形变换器
#include "mitkMeanSquaresMetric.h"                     // mitk 均方测度
#include "mitkLinearInterpolateFilter.h"               // mitk 线性插值器
#include "mitkLBFGSBOptimizer.h"                       // mitk L-BFGS-B 优化器
#include "mitkRegistrationObserver.h"                  // mitk 配准监视器
#include "mitkSubtractImageFilter.h"                   // mitk 差图像滤波器
#include "mitkDeformFieldGenerator.h"                  // mitk 形变场生成器
void main()
{
    unsigned int ImageDimension = 2;
    // 读入图像
    mitkVolume * fixedVolume1 = ReadImageFileToVolume("image1.im0");
    mitkVolume * movingVolume1 = ReadImageFileToVolume("image2.im0");
    // 配置变换器、测度模块与插值器
    mitkBSplineFreeFormTransform * transform = new mitkBSplineFreeFormTransform(ImageDimension);
    transform->AutoConfigureBSplineLattice(10,fixedVolume);
    mitkMeanSquaresMetric * metric = new mitkMeanSquaresMetric;
    mitkLinearInterpolateFilter * interpolator = new mitkLinearInterpolateFilter;
    interpolator->SetOutputDatatype(MITK_UNSIGNED_CHAR);
    // 配置 L-BFGS-B 优化器
```

```
    mitkLBFGSBOptimizer * optimizer = new mitkLBFGSBOptimizer;
    optimizer->SetLowerBound(VectorParameterType(transform->GetNumberOfParameters(),0.0));
    optimizer->SetUpperBound(VectorParameterType(transform->GetNumberOfParameters(),0.0));
    optimizer->SetBoundSelection(VectorModeType(transform->GetNumberOfParameters(),long(0)));
    optimizer->SetCostFunctionConvergenceFactor( 1.e7 );
    optimizer->SetProjectedGradientTolerance( 1e-6 );
    optimizer->SetMaximumEvaluations(200);
    optimizer->SetMaximumCorrections(20);
    optimizer->SetMaximizeFlag(false);
    optimizer->SetMaxIterations(100);
    mitkRegistrationFilter* registration = new mitkRegistrationFilter;
    // 设置输入
    registration->SetFixedVolume(fixedVolume);
    registration->SetMovingVolume(movingVolume);
    // 设置配准子模块
    registration->SetTransform(transform);
    registration->SetInterpolator(interpolator);
    registration->SetMetric(metric);
    registration->SetOptimizer(optimizer);
    // 设置配准监视器
    registration->AddObserver(new mitkRegistrationObserver (registration));
    // 进行配准
    registration->Run();
    // 获取配准结果
    mitkVolume* outVolume = registration->GetOutput();
    // 计算形变场
    mitkDeformFieldGenerator* deformGen = new mitkDeformFieldGenerator;
    deformGen->SetTransform(transform);
    deformGen->SetDeformFieldMode(MITK_DEFORMFIELD_MODE_VECTOR);
    deformGen->SetGridSize(20,20,20);
    deformGen->Run();
    mitkVolume* deformfield = deformGen->GetOutput();
    // 输出图像到文件
    WriteVolumeToImageFile("out.im0",outVolume);
    WriteVolumeToImageFile("field.im0",deformfield);
    // 释放内存
    registration->Delete();
    deformGen->Delete();
}
```

图 5-44 给出了本小节非刚性图像配准程序的结果图示。配准算法程序输入两幅二维核磁全脑切片图像(图 5-44(a)、(b)),通过 L-BFGS-B 优化器对一个均方误差代价函数进行最优化,从而找到两幅待配准图像间的几何偏移场(图 5-44(d))以及非刚性几何校正并重采样图像(图 5-44(c))。在本例中采用基于三次 B 样条函数的自由形变模型(Free-Form Deformation Model)作为该偏移场的先验模型,FFD 控制点网格的空间分布密度为 10×10。B 样条 FFD 模型的优点在于其全局稀疏性和局部控制能力,此外几何变换场的光滑性可由 B 样条函数自身的优良性质保证,因此无需外加光滑约束条件。

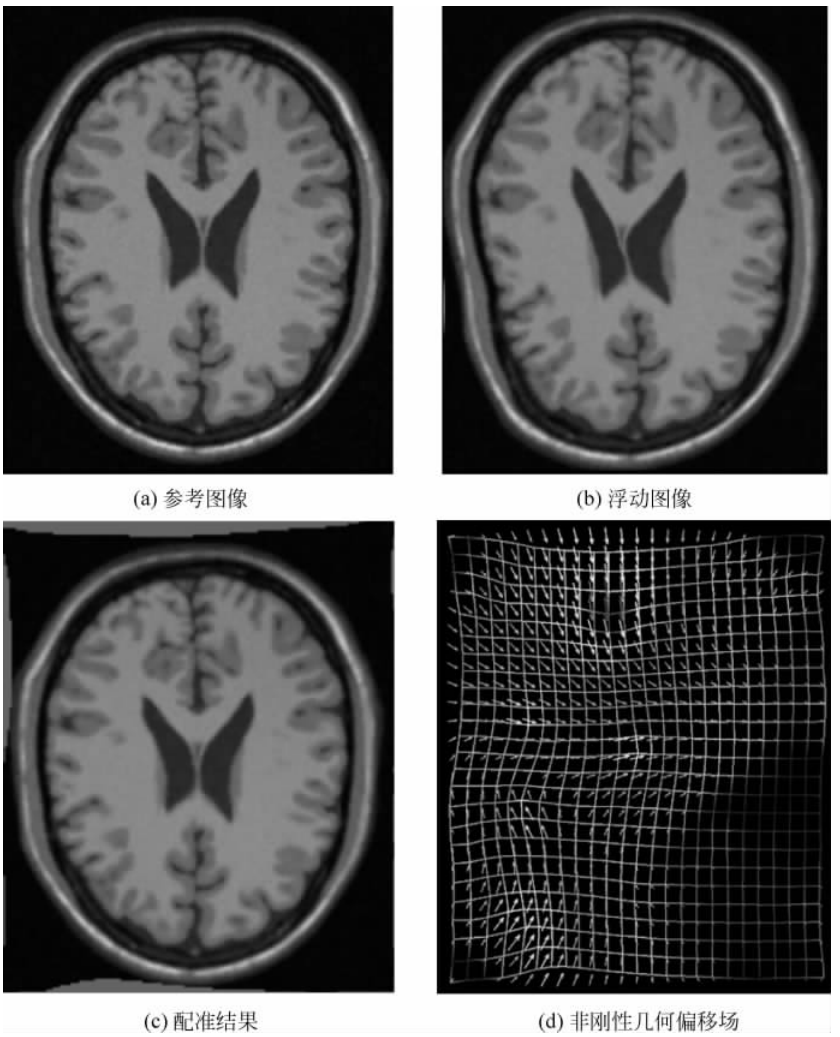


图 5-44　基于 B 样条自由形变模型的非刚性图像配准结果图示

5.7 小　　结

本章首先介绍了医学图像配准领域一些主要的问题和典型方法，包括基于灰度的图像配准、基于特征的图像配准、刚性配准与仿射配准、非刚性配准等。特别地针对近年来的一些热门问题，譬如配准算法的 GPU 加速问题、基于图像局部特征描述子的配准方法，本章都做了较为详细的分析和对比。随后，本章深入介绍了 MITK 配准算法框架的设计与实现，包括不同图像配准算法在 MITK 中的实现策略。通过算法分析与抽象，MITK 配准框架在高层概念上将整个配准问题被分成 4 个相互独立的模块：Transform、Interpolate、Metric、Optimizer，每个模块下再派生出各种不同类型的具体实现，最后将这 4 个模块按照固定的流程搭配在一起就可以组合出多种多样的配准算法。这样一种框架结构保证了配准算法的扩充性，新的算法可以通过相应模块下的派生类实现，而且各种算法的性能也能够在这一致的框架结构中方便地进行比较。MITK 配准框架与算法集合不仅是长期以来我们在本领域从事科学研究和

理论实践的重要工具，同时它也反映了我们在图像配准方向上的一些努力和感受。在本章的5.5小节，向读者介绍了我们近期在基于显著性区域特征图像配准方面的理论和算法研究成果。在本章的最后，我们简要介绍了三个MITK配准算法使用程序示例，希望其对读者在熟悉和使用MITK的过程中能有所帮助。

通过近年来的不断发展和改进，MITK中的配准框架和算法集合已初具规模，可以应用于临床中常见的图像配准问题，包括二维/三维图像配准、刚性和非刚性配准、多模态配准以及点集匹配等。当然在配准算法的鲁棒性以及非刚性配准方面，MITK配准算法还有许多不足和有待改进的方面。随着各种多模态非刚性配准算法的出现和完善，比如基于流体模型、基于有限元的方法等，这些新的算法也将逐渐加入到MITK中。

思考与练习

1. 查阅配准的综述文献以及最新的有关配准的文章，比较各种配准方法的优缺点，了解医学图像配准的热点和趋势。

2. 试简单分析一下采用正向变换将参考图像映射到浮动图像空间进行最近邻插值可能会产生“空洞”的原因。

3. 查阅有关互信息配准的文章，深入理解基于互信息配准的优点以及不足。

4. 在采用优化器对参数进行迭代优化时，对不同的参数如旋转、平移等，是否存在优化的先后顺序选择的规律？试用实验来验证你的推断。

5. 你觉得目前的MITK在配准的框架设计上，是否存在不足之处。

参 考 文 献

[1] L G Brown. A Survey of Image Registration Techniques. ACM Computing Surveys, 1992, 24(4): 325-376.

[2] J B A Maintz, M A Viergever. A survey of medical image registration. Medical Image Analysis, 1998, 2(1): 1-36.

[3] D L G Hill, P G Batchelor, M Holden et al. Medical image registration. Physics in Medicine and Biology, 2001, 46(1): 1-45.

[4] A Klein, J Andersson, B A Ardekani et al. Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration. NeuroImage, 2009, 46(3): 786-802.

[5] J P W Pluim, J B A Maintz, M A Viergever. Mutual-Information-Based Registration of Medical Images: A Survey. IEEE Transactions on Medical Imaging, 2003, 22(8): 986-1004.

[6] G Hermosillo, C Chefd ' Hotel, O Faugeras. Variational Methods for Multimodal Image Matching. International Journal of Computer Vision, 2002, 50(3): 329-343.

[7] M Holden. A review of geometric transformations for nonrigid body registration. IEEE Transactions on Medical Imaging, 2008, 27(1): 111-128.

[8] L Zagorchev, A Goshtasby. A comparative study of transformation functions for nonrigid image registration. IEEE Transactions on Image Processing, 2006, 15(3): 529-538.

[9] M H Davis, A Khotanzad, D P Flamig et al. A physics-based coordinate transformation for 3-D image matching. IEEE Transactions on Medical Imaging, 1997, 16(3): 317-328.

[10] F Maes, D Vandermeulen, P Suetens. Comparative evaluation of multiresolution optimization strategies for multimodality image registration by maximization of mutual formation. Medical Image Analysis,

1999,3(4): 373-386.

[11] S Klein, M Staring, J P W Pluim. Evaluation of optimization methods for nonrigid medical image registration using mutual information and B-splines. IEEE Transactions on Image Processing, 2007, 16 (12): 2879-2890.

[12] T M Lehmann, C Gonner, K Spitzer. Survey: Interpolation Methods in Medical Image Processing. IEEE Transactions on Medical Imaging, 1999, 18(11): 1049-1075.

[13] P Thevenaz, T Blu, M Unser. Interpolation revisited. IEEE Transactions on Medical Imaging, 2000, 19 (7): 739-758.

[14] P J Besl, N D McKay. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1992, 14(2): 239-256.

[15] H Chui, A Rangarajan. A new point matching algorithm for non-rigid registration. Computer Vision and Image Understanding, 2003, 89(2-3): 114-141.

[16] X Huang, N Paragios, D N Metaxas. Shape registration in implicit spaces using information theory and free form deformations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(8): 1303-1318.

[17] D Shen, C Davatzikos. HAMMER: Hierarchical attribute matching mechanism for elastic registration. IEEE Transactions on Medical Imaging, 2002, 21(11): 1421-1439.

[18] D Rueckert, L I Sonoda, C Hayes et al. Non Rigid registration using free-form deformations: application to breast MR images. IEEE Transactions on Medical Imaging, 1999, 18(8): 712-721.

[19] D Mattes, D R Haynor, H Vesselle et al. PET-CT image registration in the chest using free-form deformations. IEEE Transactions on Medical Imaging, 2003, 22(1): 120-128.

[20] J P Thirion. Image matching as a diffusion process: an analogy with Maxwell's demons. Medical Image Analysis, 1998, 2(3): 243-260.

[21] M F Beg, M I Miller, A Trouve et al. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. International Journal of Computer Vision, 2005, 61(2): 139-157.

[22] B B Avants, C L Epstein, M Grossman et al. Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain. Medical Image Analysis, 2008, 12(1): 26-41.

[23] G E Christensen, R D Rabbitt, M I Miller. Deformable templates using large deformation kinematics. IEEE Transactions on Image Processing, 1996, 5(10): 1435-1447.

[24] B Glocker, N Komodakis, G Tziritas et al. Dense image registration through MRFs and efficient linear programming. Medical Image Analysis, 2008, 12(6): 731-741.

[25] S Klein, M Staring, K Murphy et al. Elastix : A Toolbox for Intensity-Based Medical Image Registration. IEEE Transactions on Medical Imaging, 2010, 29(1): 196-205.

[26] L Ibanez, L Ng, J Gee et al. Registration patterns: the generic framework for image registration of the insight toolkit. In: Proceedings of IEEE International Symposium on Biomedical Imaging 2002, 2002, 345-348.

[27] L Ibanez, W Schroeder, L Ng et al. The ITK software guide. 2003.

[28] I N Bankman. Handbook of Medical Imaging: Processing and Analysis. San Diego, CA: Academic Press, 2000.

[29] A A Goshtasby. 2-D and 3-D Image Registration: for Medical, Remote Sensing, and Industrial Applications. Hoboken, NJ: Wiley Interscience, 2005.

[30] Q S Chen. Image Registration and Its Applications in Medical Imaging [Thesis]. Brussels, Vrije Universiteit Brussel Faculty of Applied Sciences, 1993.

[31] J Hajnal, D Hawkes, D Hill. Medical image registration. Boca Raton, FL: CRC, 2001.

[32] 田捷, 包尚联, 周明全. 医学影像处理与分析. 北京: 电子工业出版社, 2003.

[33] S Umeyama. Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence,1991,13(4): 376-380.

[34] T Sederberg,S R Parry. Free-form deformation of solid geometric models. SIGGRAPH Computer Graphics,1986,20(4): 151-160.

[35] R Bajcsy,S Kovačič. Multiresolution elastic matching. Computer Vision Graphics and Image Processing, 1989,46(1): 1-21.

[36] G E Christensen,H J Johnson. Consistent image registration. IEEE Transactions on Medical Imaging, 2001,20(7): 568-582.

[37] O Musse,F Heitz,J P Armspach. Topology preserving deformable image matching using constrained hierarchical parametric models. IEEE Transactions on Image Process,2001,10(7): 1081-1093.

[38] M A Fischler,R C Bolles. Random sample consensus : a paradigm for model-fitting with applications to image-analysis and automated cartography. Communications of the ACM,1981,24(6): 381-395.

[39] C H Lo,H S Don. 3-d moment forms - their construction and application to object identification and positioning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989, 11 (10): 1053-1064.

[40] D G Lowe. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision,2004,60(2): 91-110.

[41] S Lazebnik, C Schmid et al. A sparse texture representation using local affine regions. IEEE Transactions on Pattern Analysis and Machine Intelligence,2005,27(8): 1265-1278.

[42] J Matas,O Chum et al. Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing,2004,22(10): 761-767.

[43] E Tola,V Lepetit et al. DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence,2010,32(5): 815-830.

[44] K Mikolajczyk,C Schmid. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence,2005,27(10): 1615-1630.

[45] J Gårding, T Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. International Journal of Computer Vision,1996,17(2): 163-191.

[46] T Kadir, A Zisserman, M Brady. An affine invariant salient region detector. In Proceedings on European Conference on Computer Vision 2004,2004,228-241.

[47] K Mikolajczyk,C Schmid. An affine invariant interest point detector. In Proceedings on European Conference on Computer Vision 2002,2002,128-142.

[48] T Tuytelaars, L Van Gool. Matching widely separated views based on affinely invariant neighbourhoods. International Journal of Computer Vision,2004,59(1): 61-85.

[49] A Johnson,M Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence,1999,21(5): 433-449.

[50] S Belongie,J Malik et al. Shape context: A new descriptor for shape matching and object recognition. Advances in Neural Information Processing Systems 2001,2001,831-837.

[51] B Likar,F Pernuš. A hierarchical approach to elastic registration based on mutual information. Image and Vision Computing,2001,19(1-2): 33-44.

[52] S Ourselin,A Roche et al. Block matching: A general framework to improve robustness of rigid registration of medical images. Medical Image Computing and Computer-Assisted Intervention—MICCAI 2000,2000,557-566.

[53] O Fluck et al. A survey of medical image registration on graphics hardware. Computer Methods and Programs in Biomedicine,2010,in press.

[54] A Kubias,F Deinzer et al. 2D/3D image registration on the GPU. Pattern Recognition and Image Analysis,2008,18(3): 381-389.

[55] M Modat, G R Ridgway et al. Fast free-form deformation using graphics processing units. Computer Methods and Programs in Biomedicine, 2010, 98(3): 278-284.

[56] NiftyReg, http://www.cs.ucl.ac.uk/staff/m.modat/Marcs_Page/Software.html.

[57] Sebastien Ourselin, A Roche, G Subsol, Xavier Pennec, Nicholas Ayache. Reconstructing a 3D structure from serial histological sections. Image and Vision Computing, 2001, 19(1-2): 25-31.

[58] S Ourselin et al. Robust registration of multi-modal images: Towards real-time clinical applications. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2002, 2002, 140-147.

[59] R Shams, P Sadeghi et al. A Survey of Medical Image Registration on Multicore and the GPU. IEEE Signal Processing Magazine, 2010, 27(2): 50-60.

[60] J M Fitzpatrick, J B West et al. Predicting error in rigid-body point-based registration. IEEE Transactions on Medical Imaging, 1998, 17(5): 694-702.

[61] R Sibson. Studies in the robustness of multidimensional scaling: Perturbational analysis of classical scaling. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 1979, 41(2): 217-229.

[62] A Moore. An introductory tutorial on kd-trees. Technical Report No. 209, Computer Laboratory, University of Cambridge, 1991.

[63] G P Penney, J Weese, J A Little, P Desmedt, D L G Hill, D J Hawkes. A comparision of similarity measures for use in 2d-3d medical image registration. IEEE Transactions on Medical Imaging, 1998, 17(4): 586-595.

[64] M Holden, D L G Hill, E R E Denton, J M Jarosz, T C S Cox, D J Hawkes. Voxel similarity measures for 3d serial mr brain image registration. Information Processing in Medical Imaging 1999, 1999, 472-477.

[65] J Zheng, J Tian, K Deng et al. Salient Feature Region: A New Method for Retinal Image Registration. Information Technology in Biomedicine, IEEE Transactions on, 2011, 15(2): 221-232.

[66] 郑健. 集成化医学影像算法平台中基于特征区域的医学图像配准算法研究[博士论文]. 北京: 中科院自动化所, 2010.

[67] T Kadir, M Brady. Saliency, scale and image description. International Journal of Computer Vision, 45(2): 83-105, 2001.