基于区域合并的对象层次构建方法之

区域邻接图与最近邻域图

胡忠文 zhongwenhu@163.com

采用层次区域合并的方法获取区域层次结构的整体技术路线如图 1 所示：

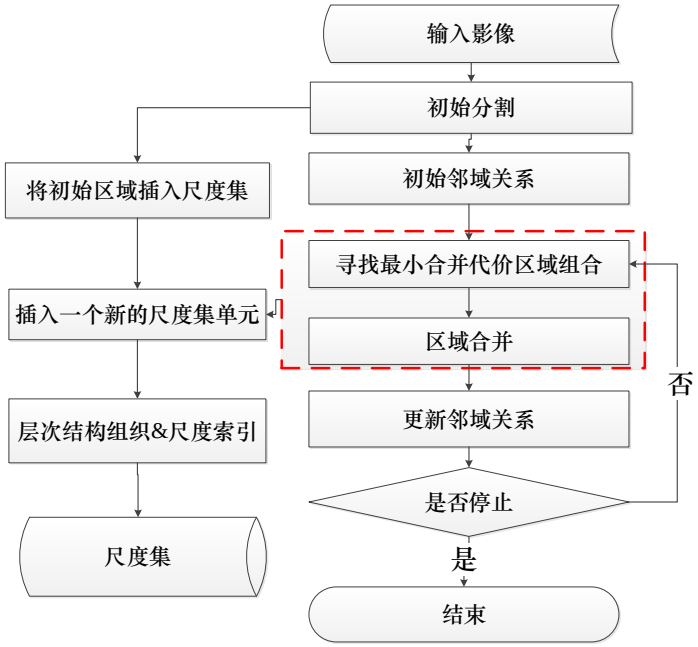

图 1 利用区域合并构建对象层次关系技术路线

其中涉及的几个关键技术问题：

(1) **初始分割：**初始分割是采用特定方法以较小的运算代价快速的将图像分解为一系列互不交叠同质区域。对图像进行初始分割可以极大的减小后续区域合并的基元数量，减少合并过程的运算量；同时，基于区域的合并方法可以更加有效的利用多种特征，如区域的统计特征、区域之间边缘的特征、区域纹理特征等。常用的初始分割方法包括分水岭算法、基于图论的方法、Meanshift 方法等。其中分水岭算法边缘定位准确、速度快，最为常用。

(2) **区域相邻关系的组织和维护：**在层次迭代的优化模型中，需要频繁的访问区域的邻接关系，并在每次区域合并之后对其进行维护更新。因此对其进行合理而高效的组织可以极大的方便整个优化过程，提升整个过程的执行效率。在本文中，介绍了采用区域邻接图(Region Adjacency Graph, RAG)记录区域之间的相互关系，采用最近邻域图(Nearest Neighbor Graph, NNG)维护合并队列的区域组织关系。

(3) **层次结构组织与尺度索引：**应用层次区域合并方法构建尺度集的过程中，需要利用

尺度参数对所获得的尺度集单元进行组织。对该部分的内容，本文不做重点解释，参考博士论文。

区域邻接图与最近邻域图

区域合并前，需要对区域的相邻关系以及区域距离等进行记录，以便寻找最优的合并区域；在合并过程中，还需要对这些关系进行不断的更新。本文中采用了区域邻接图(Region Adjacent Graph，RAG)记录区域的邻接关系；而在合并过程中，通过最近邻域图(Nearest Neighbor Graph, NNG)对 RAG 进行简化，以在较低运算代价下高效的完成相邻关系的记录和更新。同时，层次迭代的优化模型在每一步中均需要从所有可能合并的对象组合中找出代价最小的。**并且**，在合并中还需要注意，每次合并的两个区域应该满足局部相互最优准则，即两个区域互为对方的合并代价最小的区域。通过 NNG 的中特殊的环记录方式，可以在每一步的合并中从 NNG 中方便的找到局部相互最优合并对象。

1 区域邻接图（RAG）

区域邻接图是一种记录区域邻接关系的数据结构。对于一个有 K 个区域的初始分割结果，采用无向图$G = (V, E)$对其邻接关系进行记录，其中 $V=\{S_1,S_2,\ldots,S_K\}$是所有 K 个顶点的集合，E 是所有边界的集合。在图中，每一个区域都被表示为途中的一个节点，并且对于任意两个区域$(S_i, S_j)\in$V,如果 S_i, S_j 相邻，那么对应的节点之间就存在一条边相连。如图 2 所示，一个包含 5 个区域的初始分割，其可以用图 2(b)所示的区域邻接图表示。图 G (V,E)中，每一条边均赋予一定的权重，其对应于合并两个端点区域的代价值。那么对于整个初始分割而言，最相似的相邻区域在图中对应的节点权重也最小。在每一次合并中寻找最小合并代价准则即变为寻找图中最小权重的边。为了方便寻找最小代价的区域组合，图 G (V,E)中的所有边缘均是按照其权重从小到大排序。本文中，采用红黑树数据结构存储边缘集合 E。红黑树是一种自平衡二叉树，典型的用途是实现关联数组。红黑树是一种复杂的数据结构，但是它的操作有着良好的最坏运行时间，其可以在 O(log N)的时间里做查找、插入和删除(n 是指树中节点的个数). 对于节点集 V，采数组进行存储，并且每个节点均以指针的形式与其对应的边缘建立关联。

首先找到Σ4与Σ5合并代价最小

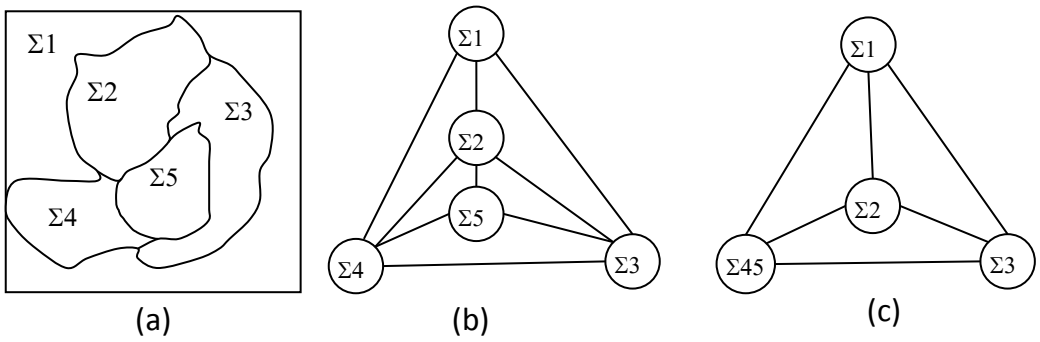

图 2 区域邻接图示意图 (a) 区域邻接关系； (b) 初始区域邻接图；

(c) 合并区域 S_4,S_5 之后的区域邻接图

在初始分割之后，需要对分割图像进行扫描，建立区域邻接关系，获取区域属性信息，并以此构建区域邻接图，最终在此基础上进行迭代的区域合并。对于初始有 K 个节点的邻接图，每一次的合并中均会减少一个节点，以及其对应的边，形成新的邻接图。由于影像分割的结果在空间上一定是相互连通的，因此区域邻接图 RAG 上只存在唯一的一个连通成分，即任意两个节点之间一定能够找到一条通路(边缘集合)使之相互连通。那么对于 N 个节点的区域邻接图，通过 N-1 次的区域合并一定只剩下一个节点，0 条边。那么利用 RAG 进行区域合并，构建尺度集的过程如下所示：

输入: 含有K个节点的RAG;
迭代: for i=0 to k-1;
在（K-i）个节点的RAG中寻找最小权重边缘;
合并对应的区域;
更新RAG;
往尺度集中插入一个基本单元;
输出：尺度集结构

在每一次的迭代中均需要合并节点、删除对应边以及由于区域合并而造成的重复的边缘。由于合并之后区域属性以及与其他区域的关系发生了变化，因此需要更新与周围其他节点相连的边的权重，同时这也意味着在每一次合并中，均需要重新计算合并后的区域与邻域的距离。如图 2 所示，初始 RAG 如图(b)所示，合并开始时首先找到了区域 S_4 和 S_5 合并代价最小，将两区域合并得到 S_{45}，删除原来连接 S_4 和 S_5 的边 $E(S_4,S_5)$。同时可以发现，由于 S_4 和 S_5 的合并，边 $E(S_2,S_4)$与 $E(S_2,S_5)$合并为一条边 $E(S_2,S_{45})$，同理，$E(S_3,S_4)$与 $E(S_3,S_5)$也合并为一条边 $E(S_3,S_{45})$，因此需要删除冗余的边界并更新保留边界的权重。边缘 $E(S_1,S_4)$由于区域 S_4 属性的改变，需要重新计算权重。在存储节点时，以指针的方式记录了所有与该节点相连的边缘在边界集合 E 中的位置，因此可以很方便的处理与指定节点相连的边。

层次结构与 RAG 结构：从 RAG 的结构以及合并过程，可以较为容易的获取尺度集的结构。初始 RAG 中的节点，分别对应于尺度集 0 截面的所有节点。在利用 RAG 合并过程中，每一步合并均会产生一个新的节点，以及一个合并关系，这新的节点以及合并关系便对应于尺度集中的一个基本单元。

在合并未进行到最后一步之前，尺度集树状结构中可能拥有多个分支，即存在多个根节点。但每次合并之后插入的尺度集基本单元的子节点一定能在现有尺度集中找到，直到最后一步时，层次结构最顶层仅存在两个分支，且最后一步合并的插入的基本单元正好能将这两个分支连接为一颗完整的树结构。

2 最近邻域图（NNG）

虽然采用 RAG 能够较为完全的记录所有的邻接关系，但是在实际的应用中，由于初始分割的区域个数众多，并且每个区域有多个邻接区域，因此造成 RAG 的节点数量和边数量巨大，从中搜寻最佳合并对象的计算复杂度非常大。然而，由于每个区域在合并时，均是优先与合并代价最小的区域进行合并，因此对于每个区域只需要在其邻域中寻找最佳合并对象并记录相关的边缘，并不需要记录其与所有邻域的 RAG 的边缘。本节中采用最近邻域图(Nearest Neighbor Graph, NNG)对 RAG 进行简化。最近邻域图为有向图，从每个节点出发连出一条有向的边，该边指向该节点的最佳合并对象。

对于一个给定的区域邻接图 RAG，定义其简化形式 NNG 为$G_m = (V_m, E_m)$为一个有向图。其中$V_m = V$表示原有的节点集合，而E_m是由一系列有指向的边构成的集合。

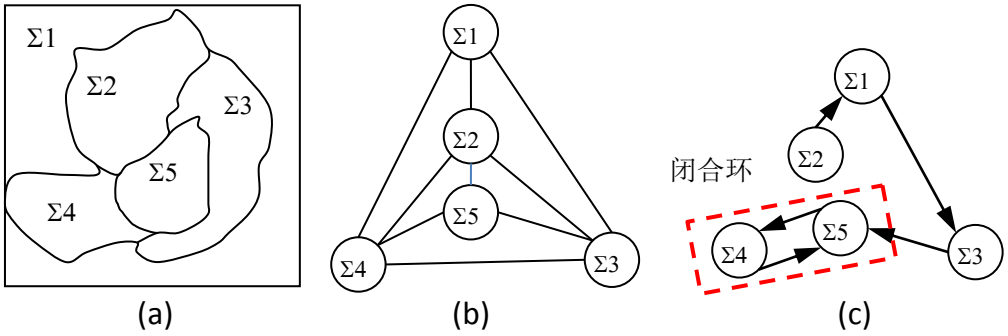

图 3 区域邻接图与最近邻域图　(a) 区域划分；(b) 区域邻接图；(c) 最近邻域图

如图 3 所示，对于图 3-6(a)所示初始分割结果，其采用区域邻接图表示如图 3(b)所示，其中共有 5 个节点和 7 条边。采用最近邻域图表示结果如图 3(c)所示。对于给定的任意节点 S，需要从 S 的邻域中找出与其合并代价最小的区域 N，并以有方向的边 $E_m(S,N)$从 S 指向 N。值得注意的是，当区域 S 周围存在多个邻域与其合并代价相同时，$E_m(S,N)$应该指向其中标

号最小的区域，以避免形成死循环。采用最近邻域图表示时，可以极大的减小数据量，如图3-6(c)所示的简单情况下，其初始 NNG 只需要记录 4 条边，在区域邻接情况更为复杂时减少的数据量更少。

采用 NNG 记录区域邻接关系时，从每个节点只引出一条边，因此对于有 N 个节点的图 G，其边的数目也是 N。采用这种有向图记录时具有如下的性质：

性质 1：在 NNG 中一定存在闭合的环，如图 3(c)中所示的闭合环，每一个闭合环记录的都是满足局部相互最优准则的相邻区域；

性质 2：闭合环有且只有两条边。由于在构建 NNG 时，遇到相同合并代价的时候优先选择标号较小的区域作为指向的区域，因此整个过程不可能形成大于两条边的闭合环；

性质 3：最优的合并区域一定连接成闭合环的，即连接在闭合环上的两个节点 S_1 和 S_2 均为对方的最佳合并对象；

性质 4：每个节点仅可能存在于一个闭合环上，因为每个节点只能有一个最佳的合并对象，因此以其为起点的边只能有一条，只可能构成一个闭合环；

性质 5：NNG 中闭合环的数量最小为 1，最大为节点数目 N 的 1/2(当图中的节点两两相互构成闭合环)。

由 NNG 的性质可以看出，闭合环代表的是区域合并的局部最优解，即对于一个闭合环上的两个节点，两者互为最优的合并对象。因此整个区域合并过程是在不断寻找闭合环然后进行合并的过程。由 NNG 性质第一条可知，NNG 中必然存在闭合环，因此可以保证合并过程中一定可以找到最优的合并区域，直到 NNG 中边的数目为 0。

同时，由于合并只可能出现在闭合环的节点之间，因此在寻找最优合并区域时，只需要在 NNG 中的闭合环中寻找，忽略那些单向连接的边，从而可以进一步缩减搜索空间。将闭合环以集合的形式进行存储(Closed Cycle Set, CCS)，由性质五可知，一个 NNG 中最多有 N/2 个闭合环，可以采用一条双向连接的边对其进行记录，因此 NNG 中边的最大数目为 N/2，而最小为 1。这极大减少了搜寻最佳合并对象时的搜索空间的大小。

采用 RAG 与 NNG 相结合的方式，对区域合并过程中的对象关系进行组织和更新，构建尺度集的步骤如下：

输入: 含有K个节点的RAG、NNG；

迭代: for i=0 to K-1；

在NNG闭合环中寻找最小权重边缘；

合并对应的区域；

更新RAG、NNG、CCS；

往尺度集中输入一个新的单元；

输出：尺度集结构。

每一次合并后，RAG 的更新较为简单，其过程如图 3-4 所示；NNG 的更新相对复杂，当闭合环上的两个节点合并后，需要根据 RAG 更新后各个边的权重变化情况，重新构成 NNG 中的有向边。其中涉及到 NNG 中闭合换的删除、增加以及 NNG 边的改变等情况如图 3-6 示意图所示：

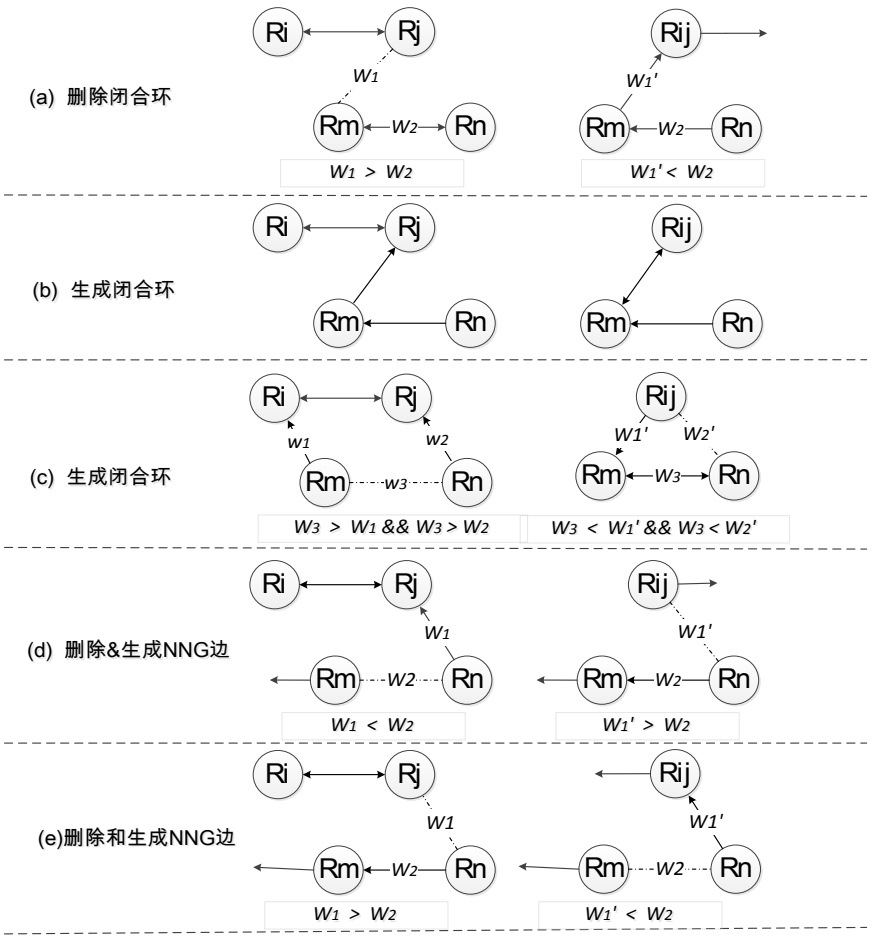

图 4 NNG 更新示意图。其中→表示所指向的对象为最佳合并对象，其间存在 NNG 边；---表示两区域相邻，但相互都不是最佳合并对象，其间存在 RAG 边，↔表示 NNG 中的闭合环，即两区域互为最佳合并对象。

每一次合并都是从 NNG 中的某个闭合环上节点的合并，如图 4 中的↔所指的节点。每

一次合并之后，RAG 减少一个节点，同时与该节点所有相关的边都要进行更新，重新计算合并距离；而合并距离的改变会引起 NNG 结构的改变。如图 4(a)所示，节点 *Ri* 和 *Rj*，*Rm* 和 *Rn* 分别互为最佳合并对象，构成两个闭合环，其中 *Rj* 和 *Rm* 相邻(虚线相连)，且合并代价 $w_1>w_2$。i 和 j 之间合并代价较小，优先合并，得到新的区域 *Rij*。当 *Ri* 和 *Rj* 合并之后，需要重新计算 *Rm* 和 *Rj* 之间的距离，得到 w_1'。此时由于 $w_1' > w_2$，对区域 *Rm* 而言，其最佳合并对象不再是 *Rn*，而是新合并之后的区域 *Rij*。由此，从图 4(a) 右图中可以看到，原本连接(*Rm*,*Rn*)的闭合环被拆散，形成 NNG 中的两条边。

区域的合并也会产生新的闭合环，如图 4(b)和(c)所示。在图 4(b)中，*Ri* 和 *Rj* 合并之后形成的新区域 *Rij*，其最佳合并区域为 *Rm*，而 *Rm* 的最佳合并区域为 *Rij*，由此形成了连接 *Rij* 和 *Rm* 的新的闭合环。在图 4(c)左图中，可以看到区域 *Rm* 和 *Rn* 为相邻区域，但他们的最佳合并区域分别为 *Ri* 和 *Rj* ($w_1<w_3$ 且 $w_2<w_3$)。由于区域 *Ri* 和 *Rj* 合并之后，新的区域与 *Rm* 和 *Rn* 之间的距离发生变化，使得 $w_1'>w_3$ 且 $w_2'>w_3$，使得 *Rm* 和 *Rn* 互相成为最佳合并的对象，由此形成了新的闭合环。

区域的合并可能引起 NNG 边的改变，但并不影响 NNG 中闭合环的情况，如图 4(d)和(e)所示。如图 4(d)所示，合并之前区域 *Rn* 月 *Rj* 的距离 w_1 小于 *Rn* 与 *Rm* 之间的距离 w_2，因此 *Rn* 与 *Rj* 之间存在 NNG 边。区域 *Ri* 和 *Rj* 合并之后，*Rn* 与新区域 *Rij* 之间距离 $w_1'>w_2$，由此，*Rn* 的最佳合并对象为 *Rm*，因此需要删除连接 *Rn* 与 *Rj* 之间的 NNG 边，增加 *Rn* 与 *Rm* 之间的 NNG 边。这一过程只涉及 NNG 边的操作，不对闭合环产生影响。同理，图 4(e)所示也只是 NNG 边的操作。

NNG 中的闭合环集合代表的均为相互最优的合并组合，每次只需从 NNG 闭合环集合中寻找代价值最小的两个区域进行合并，更新 RAG，NNG 和 CCS。NNG 的性质保证了一定存在闭合环，因此区域合并能够一直持续下去，直至 RAG 中只剩一个节点，0 条边，完成整个尺度集的构建，得到尺度集的最后一个节点。

3 RAG 与 NNG 效率分析

通过 NNG 的方式极大的减小了搜索最优合并区域组合的运算量和存储空间。RAG&NNG 联合使用的情况下，RAG 节点和边的数目依旧保持不变，但是在 RAG 中只需要存储区域的邻接关系，不再需要额外存储区域之间合并的代价值；NNG 中的数据节点和边的数量相比 RAG 减少了许多，因此在 NNG 中搜索最佳合并区域其搜索空间更小。如图 5 所示为某区域合并过程中 RAG 边和 NNG 边的数量虽区域合并过程变化的情况。由图中可

以看出，RAG 边和 NNG 边的数量均随着合并过程的进行不断减少，但是 NNG 边的数量明显少于 RAG 边的数量。在本文所陈述的实验之外，大量实验表明，NNG 边的数量通常只有 RAG 的 12%～13%，由此可以见采用 NNG 记录区域的合并次序关系，可以极大的减少每次合并过程中搜寻最优合并对象的搜索空间，提高算法效率。

由第 2 节可知，在更新 NNG 的过程中，需要考虑与对应节点相连的 RAG 边的数量，即对应区域邻接区域的个数。当所合并的两个区域邻接区域比较多时，其每次合并更新 NNG 时运算量较大。如图 3-9 所示为某初始分割中区域邻域个数的直方图。可见大多数区域的邻接区域个数在 20 个以内。通过大量实验验证，大部分区域的邻接区域个数在 2～15 之间，均值约为 5.7。可见，在使用 NNG 记录合并关系时，其每次更新时其合并代价是较小的。

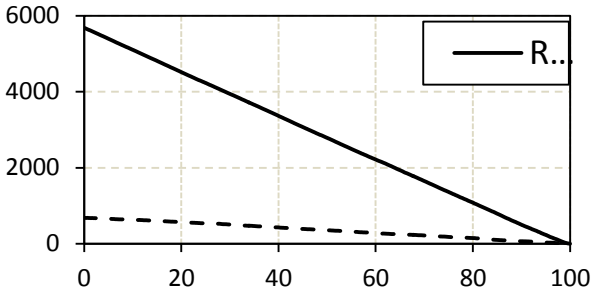

图 5 某区域合并中 RAG 边与 NNG 边数量变化趋势。

（其中纵轴表示 RAG 或 NNG 中边的数量，横轴表示区域合并进行过程的百分比）

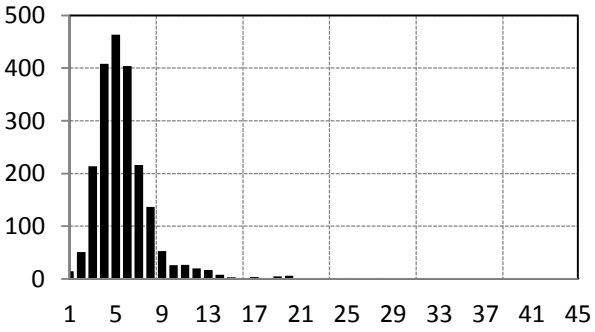

图 6 初始分割中区域邻域数量直方图。（其中纵轴表示出现频数，横轴表示邻域的个数）

4、结论

将 RAG 和 NNG 进行结合，可以有效而简单的组织对象的邻域关系，为层次区域合并提供有力的保障，效率超高！

还有一句话，理论看似简单，但实践起来还是灰常麻烦的…相关代码量约 1500 行