我们开发了用于3D神经元跟踪的全路径修剪2.0（APP2）。 最重要的想法是使用长片段第一层次过程而不是APP中的原始首位搜索过程来修剪神经元形态的初始重建树。 为了进一步提高APP2的鲁棒性，我们计算了直接为灰度图像的所有图像体素的距离变换，而不需要在调用常规距离变换之前对图像进行二值化。 我们还设计了一种基于算法的快速算法来计算初始重建树，而不预先计算大图。 这种方法允许我们跟踪大图像。 我们在700 3D显微镜图像上对APP2进行了测试，发现APP2在大多数情况下可以比以前的几种方法产生更令人满意的结果。

以前在All-Path-Pruning（APP）方法中讨论了复杂神经元形态和低质量图像数据的神经元跟踪自动化（Peng et al。，2011）。 APP的关键思想是产生覆盖3D图像中神经元的所有潜在信号的初始重建，然后对不需要的分支进行线性时间修剪，直到产生最小紧凑的表示，同时所有神经元信号的覆盖范围为 保持。

在这项研究中，我们提出了一个新版本的APP算法，称为APP2（图1），其目标是在更短的时间内产生更准确和更健壮的重建。 为了实现这一目标，我们在三个组件中开发新的算法：一种从灰度图像直接生成神经元信号的距离变换的方法，无需基于阈值的二值化（图1B）; （ii）生成初始重建的方法（图1C）; （iii）产生最终重建的分层修剪方法（图1D）。

与APP相比，APP2的所有三个组件都是新颖的，特别是APP2比APP更有效的修剪过程。 APP2的距离变换步骤和初始重建步骤是有用的增强，也可以基本上用于APP。 因此，为了评估我们算法的效率，我们将跟踪结果与灰度距离变换（GWDT）或无GWDT进行比较。 我们还比较了APP2和APP方法的结果。 为了检验我们的算法的鲁棒性，我们使用信号删除测试。 我们还对来自不同来源的许多数据集进行了我们的算法测试，包括DIADEM数据集和许多其他具有重噪声的果蝇和蜻蜓神经元数据集。

在我们的实验中，我们发现在大多数情况下，APP2能够在短时间内产生合理的跟踪结果，通常在几秒钟之内就可以获得大型3D图像。 我们的新结果通常比使用其他现有竞争方法产生的结果更好。

在本文中，我们专注于3D光学显微镜图像（通常是共聚焦或多光子激光扫描显微镜图像）的神经元跟踪。 以3D灰度图像作为输入，神经元跟踪方法产生该图像中神经元形态的数字表示。 已经发现追踪在图像中可能重叠的多个神经元已经被根本不明了，并且可以使用诸如dBrainbow（Hampel等人，2011）的生物组织标记方法来最终解决。 在后一种情况下，问题减少到跟踪一组单通道图像，每个单通道图像都包含单个神经元。 因此，这里我们仅讨论如何从图像中重构单个神经元的形态。

在许多以前的研究中，使用树图G描述神经元的3D重建形态.G具有对应于重建的种子位置的根节点，其在许多情况下也对应于神经元的神经元。 G也可以包含许多叶节点，分支节点和其他节点间节点。最近的APP方法（Peng et al。，2011）中的一个重要思想是首先从图像中产生过度重建，以捕获神经元的所有可能的信号/像素，然后使用最优修剪过程去除大部分刺在这种过度重建中产生神经元的最终简洁表示，具有所有神经元信号的最大覆盖。修剪过程从过度重建的叶节点开始。迭代地进行“覆盖”测试，以检查其他节点是否已被“覆盖”（即具有显着信号重叠）。被其他人覆盖的节点将被删除;否则将被保留。类似的过程也适用于所有节间。整个过程被重复，直到产生了使信号覆盖最大化的最简洁的表示。尽管APP中的首要搜索方法是有效的，但是它需要多次迭代，这仍然是耗时的，即使算法本身对重构节点的数量具有线性时间复杂度。此外，APP不考虑如何最佳地预处理输入图像以优化跟踪结果。

APP2遵循APP的基本框架。 然而，它增强了原始方法的关键组件。 简而言之，APP2由图1B，C和D三个步骤组成：（i）GWDT（2.3节）; （ii）构建追踪神经元的初始，过度重建（第2.4节）; 和（iii）按层次顺序修剪过度重建（第2.5节）。 APP2的细节描述如下。

基本上是区域生长方案的快速行进（FM）算法（Sethian等，1999）在APP2中起着重要的作用。 在FM框架中实现GWDT和初始神经元重建。 在FM中，我们将图像建模为图形，其中每个图形顶点对应于图像像素（体素）。 在每对直接相邻的像素顶点之间存在边缘。 FM将图像图从一组预定义的种子顶点增长到所有剩余的图像像素顶点，并以距离递增的顺序。 所有图像像素分为三组，标记为ALIVE，TRIAL和FAR。

FM有两个主要步骤：初始化和递归。 首先，所有种子顶点都被初始化为ALIVE; 种子的邻居被初始化为TRIAL; 其余的设置为FAR。 然后，从一组TRIAL顶点，我们将提取一个与ALIVE集合具有最小距离值的顶点x。 提取的顶点x然后从TRIAL转换为ALIVE。

FM的一个重要实现技巧是在斐波纳契堆中维护TRIAL顶点，以便有效地获得所需的最小值.

为了增强产生初始神经元重建的步骤，在APP2中，我们将距离变换（DT）应用于输入图像。 在具有相对“平坦”强度的图像区域的情况下，DT能够产生图像强度的梯度：靠近该区域的中心，图像强度大并且接近边界，强度很小。 我们把这个ICDB的原则称为“增加中心强度，降低边界附近的强度”。 这将有助于通过迫使最短的路径穿过神经元的骨架来建立高质量的初始重建。

然而，传统的距离变换仅适用于通过对灰度图像进行阈值化而产生的二进制图像。 不合适的阈值可能在图像下或之上。 在这里，我们直接提出了用于灰度图像的GWDT方法。 在常规距离变换中，每个图像像素的距离值被定义为与背景像素的最小欧几里德距离。 在GWDT中，每个像素的距离值被定义为沿着最短路径到背景的图像像素的强度之和。 GWDT最初由Rutovitz（1968）提出。 然而，大多数以前的GWDT研究和相应的实现（如最近发布的Matlab工具箱功能）仅限于2D情况，而我们的方法和实现对于N维数据是一般的（n = 2，3，...））。 在下文中，我们描述了我们在FM框架内的快速实现。

GWDT中定义的距离值符合ICDB原则。 为了使用GWDT，我们经常使用低阈值（例如整个图像的平均强度）。 具有不大于该阈值的强度值的任何图像像素称为“背景像素”。 我们首先将所有图像背景像素设置为“种子”，然后计算从其他像素到这些种子像素的距离。 这个过程与区域增长相似，因此在第2.2节的FM框架内制定。 这里，连续图像像素顶点之间的边缘距离被定义为

: We find that GWDT method provides a way to detect the soma position of a neuron. Normally, the soma has the maximum distance-transformed value

在APP中，初始神经元重建本质上是通过找到所有剩余的前景图像像素的单源（通常从soma）最短路径产生的。 APP使用Dijkstra的算法，它需要首先构建所有前景像素的图形，然后找到从每个像素到种子的最短路径。 对于大型3D图像堆栈，这种方法可能需要大量的计算机内存来保存图形。 在APP2中，我们提出了一种使用FM构建初始重建的新方法（图2），而不需要创建如此大的图形。

在我们的实现中，我们在FM中添加一个父谱图，如第2.2节所述，从单个源生成最短路径树。 最初，每个图像像素x的母体被设置为自身，即parxð¼x。 然后，对于s的每个邻居像素y，我们将它们设置为具有标签“TRIAL”，并且同时参数。 在递归步骤中，对于最小像素x和其邻居y的每个，我们设置

当FM完成后，我们可以从父母图中构建初始重建。 除了需要的小的工作空间外，用于产生初始重构的FM的另一个有用属性是可以根据需要容易地停止。 我们在APP2中考虑两种方法。 首先，当任何背景像素变为ALIVE时，递归步骤将停止。 这种方法可以防止行军过程发展到任何不相关的领域。 第二，用户可以选择性地预先指定一些位置（例如神经元的一些特殊终端）作为附加先验; 当所有这些特殊位置都标记为ALIVE时，FM停止。 第二种方法迫使FM到达这些指定位置。 在实践中使用这两种方法来产生尽可能覆盖信号的完整的初始重构，从而使得更容易追踪图像中的碎片或间隙。

图2显示了神经元初始重建的例子。 它们是树形图，每个都有一些刺激。 下一步是通过删除不必要或冗余的杂散来找到树的主要骨架。 在这里，我们提出了一种分层修剪方法，它包含两个步骤：分层段构造步骤和递归修剪步骤。

为了简单起见，我们在本节中将初始重建称为“树”。 我们在树中定义一个段，作为连接树中两个分支节点的路径。 在层次分段构造步骤中，我们将树中的所有段从最重要到最不重要，从而生成它们的层次结构。 段的“重要性”是根据其长度来定义的。 段越长越重要。 显然，任何一段都没有重叠。 为了获得重要性分数，我们首先找到从最远端叶节点到源节点（种子）的最长路径。 然后，我们从树中删除该段，并从原始树中的其余部分中找到第二个最长段。 我们迭代此过程，直到所有段都被排序。

我们进一步提高算法的效率如下。 我们观察到，在我们对树的分解中，每个分层段从叶节点开始。 此外，每个分层段与叶节点之间存在一对一的映射关系。 因此，在精细的算法中，我们以自下而上的顺序构建层次的段。 首先，我们检测从每个叶节点到其最近的分支节点（不包括分支节点）的分段。 每个分支节点连接至少两个这样的分段。 然后，我们将分支节点合并到最长的分段（称为“联合分段”）。 接下来，原来连接到分支节点的其他段被设置为联合段的子段。 我们迭代这个联合段合并过程，直到达到种子节点。

在修剪步骤中，从未删除的层次结构段中，我们按照重要性得分按顺序选择一个段。 然后，我们将APP中的信号覆盖概念扩展到此段的覆盖率（见文中的后面）。 如果覆盖率大于阈值（通常为75％），我们删除此细分及其所有子细分。 否则，我们保留这个选择的段并掩盖段的覆盖区域。 我们重复此过程，直到不能删除段。

段的覆盖区域定义为段中所有节点的合并覆盖区域。 节点的覆盖区域被定义为以节点为中心的球面区域，具有估计的节点半径，其使用Peng等人描述的方法计算。 （2010b，2011）。 段的覆盖率被定义为所有屏蔽节点的数量与段中的节点总数的比率。 此外，我们认为具有不同强度的图像像素应具有不同的权重。 因此，在我们的方案中，我们使用图像像素强度加权覆盖率，其被定义为所有屏蔽节点的强度之和除以该段中所有节点的强度之和。