FME 在二维视图下面要素拔高的应用

摘要

本次实验的目的是基于简单面要素在 FME 中实现二维视图下的拔高。基本思路是通过在平面横纵坐标上的平移，得到构建 2.5D 面要素的侧面线、底面线及顶面线要素。侧面线通过提取面折点并对这些折点编号后平移折点，连接相同编号的折点，从而得到侧面线。底面线通过对角线与多边形边的相交关系，对相交线打断并筛选出多边形边从而得到底面线。顶面线则由底面线平移得到。最后以侧面线、底面线及顶面线构建 2.5D 面。

1 实验及数据简介

1.1 实验简介

在本实验前，我接触了一个电子地图制图类的项目。用户向开发者提出了对场景内建筑物进行拔高的要求，并要求数据量小、显示速度快。开发者基于用户的要求对建筑物进行了简易的 2.5D 拔高，而非通过三维场景中搭建三维模型来实现，既满足了建筑物的三维即视感也满足数据量小、显示速度快的要求。作为 FME 爱好者，我设想了如何在 FME 实现上述功能，以下是对于本实验的介绍与解释。

1.2 数据简介

为了使实验过程简洁，只需要准备数个面要素即可（可直接在 ArcGIS 中随机绘制面要素）。

2 程序设计及实施

2.1 程序设计

1）将准备的面要素作为底面，根据底面的折点生成点要素。

2）将上述的点要素在水平方向及竖直方向各平移一定的距离，将平移后点要素与原点要素两两连线，构成侧面线。

3）将准备的面要素根据折点生成线要素并在水平方向及竖直方向各平移一定的距离，生成底面线要素和顶面线要素。

4）根据侧面线要素、底面线要素及顶面线要素构面，生成 2.5D 的面要素。

2.2 程序实施

2.2.1 面要素折点转点

将面要素折点转为点要素，首先需要得到面要素所有折点的坐标信息，再将坐标信息转为点要素，从而得到面的各个折点。

1）获取面要素上所有点的坐标：

使用转换器 CoordinateConcatenator，由于程序只在平面上执行，故只获取 X 坐标与 Y 坐标。如图 2-1 是该转换器的基本设置。

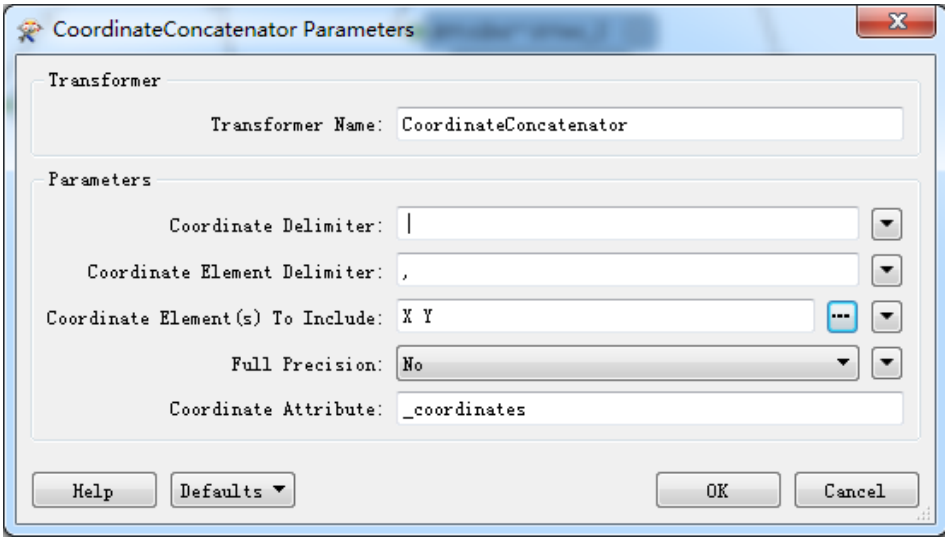

图 2-1

通过转换器 CoordinateConcatenator 得到了面要素所有折点的坐标，但是坐标是以 “,”分割 X 坐标与 Y 坐标，以“|”分割不同点之间的坐标，储存在同一字段下（字段名为_coordinates），故需要将该字段裂开并且将 X 坐标，Y 坐标分别储存到不同字段中。

	Id	图号	_coordinates
1	0	1	200001.69333672151,199993.64998730272|200008.14918296412,199995.02582338639|200007.30251460522,1
2	0	2	200020.53170773014,199998.62416391633|200027.62255524471,200000.95250190422|200029.95089323632,1
3	0	3	200008.25501650944,200005.7150114309|200012.38252476417,200005.82084497437|200011.43002285995,20

图 2-2

2）将不同点的坐标裂开：

裂开属性使用转换器 AttributeSplitter。裂开后的不同坐标仅是存储在 FME 列表（列表名为_list）中，为了方便后续操作，应将列表中“_list”中的坐标信息暴露出。暴露属性用转换器 ListExploder。

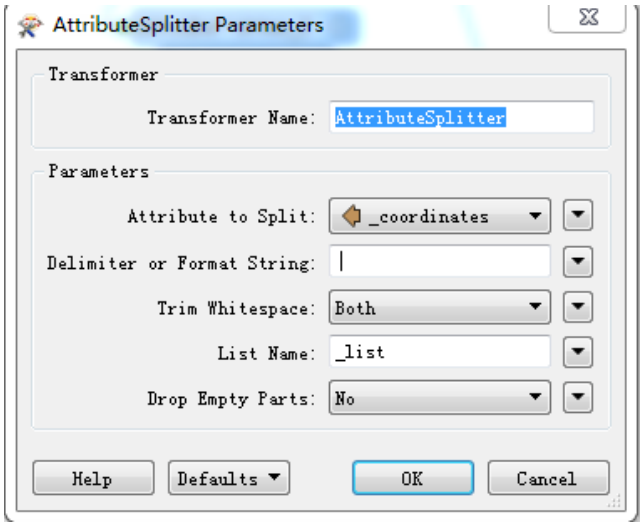

图 2-3

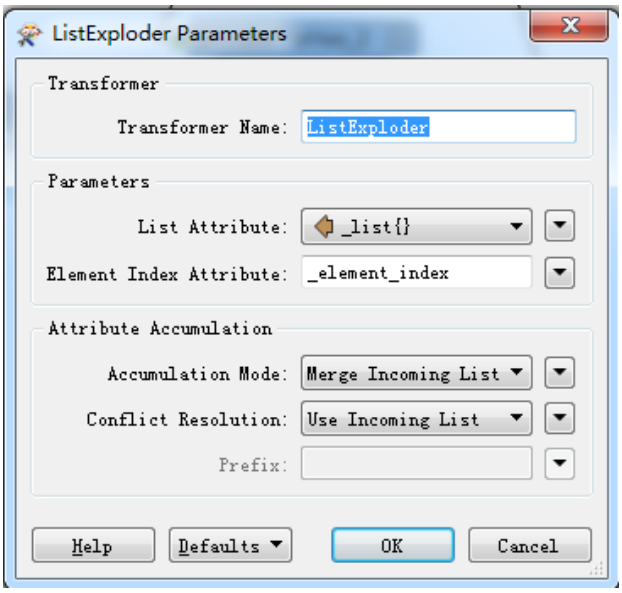

图 2-4

在“_list”列表中，面要素的点坐标信息以“_list{1}”、“_list{2}”等分别储存，而暴露出列表信息的要素中名为“_element_index”的字段正是“_list{0}”、“_list{1}”中的 0、1，可理解为同一面的折点的编号。

在暴露出“_list”列表后，生成了一些新要素，该新要素中储存有面的各个折点的坐标信息，则该新要素可视为每个折点与其原始面要素的复合要素。对这些要素编号，可视为对各折点编号，编号使用转换器 Counter，起始编号设为 1。

3）将相同点的 X、Y 坐标裂开：

与 2）中裂开不同点的坐标的思路相似，但分割符号应设置为“,”。得到的 X、Y 坐标分别储存到新列表中，列表名为别为“_list1{0}”、“_list1{1}”。那么，经过“_list1”列表暴露后生成的新要素中“_element_index”为 0 的是带有 X 坐标信息的要素，“_element_index”为 1 的是是带有 Y 坐标信息的要素。

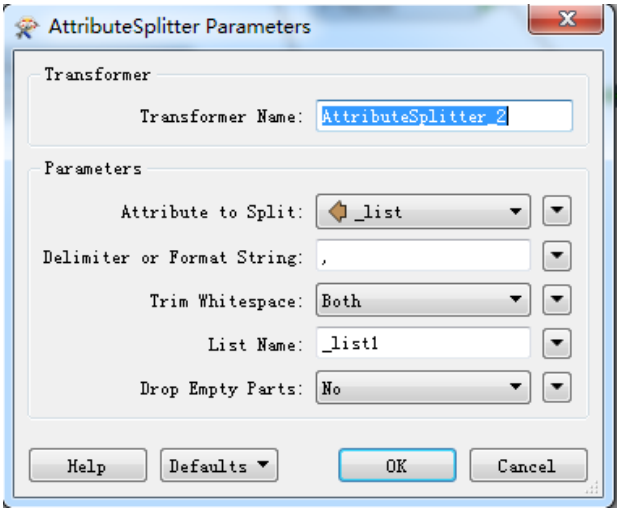

图 2-5

_element_index	_count	_list1
0	1	200001.69333672151
1	1	199993.64998730272
0	2	200008.14918296412
1	2	199995.02582338639
0	3	200007.30251460522
1	3	199990.58081449755
0	4	200000.6350012701
1	4	199989.41664550081
0	5	200001.69333672151

图 2-6

4）分别将 X、Y 坐标赋到新字段中：

如图 2-6，X、Y 坐标都储存在“list1”字段中，但通过“_element_index”可进行区分，故为了将 X、Y 坐标分别赋值到新字段中，首先要通过“_element_index”筛选 X 坐标与 Y 坐标的信息。筛选使用转换器 Tester。

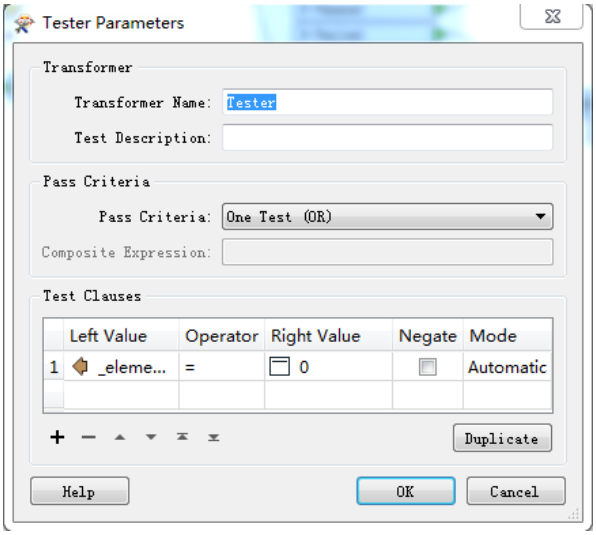

图 2-7

筛选之后的数据通过转换器 AttributeCreator，将筛选成功的要素的 list1 赋值到“x”字段，将筛选失败的要素的 list1 赋值到“y”字段

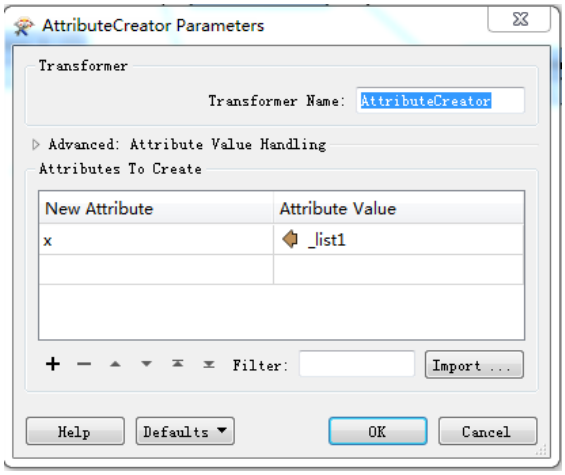

图 2-8

5）将同一点的 X、Y 坐标合并：

经过上述步骤的操作，已经将同一点的 X、Y 坐标信息存储到了两个要素中，而这两个要素的“_count”字段的值应该相同，故可以通过“_count”将要素分组，以组为单位进行合并，合并使用转换器 Dissolver。

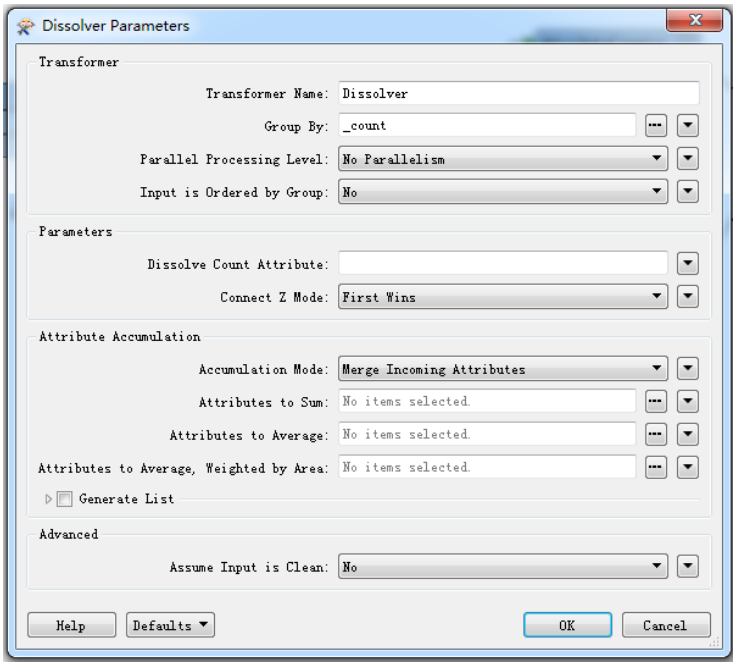

图 2-9

6）通过坐标转点和去除重复点：

坐标转点使用转换器 VertexCreator，因不需要保留面状要素，则在转换模式选择中选择以点替换原要素。

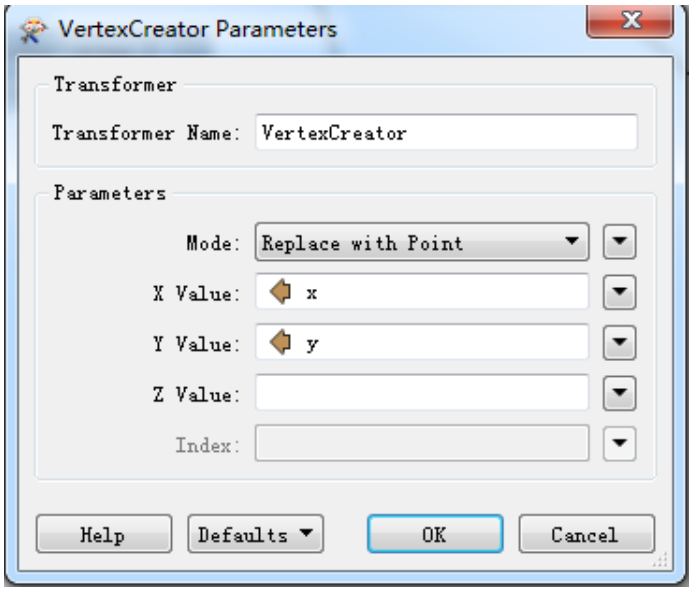

图 2-10

本次实验的基础数据是两个四边形和一个三角形，显而易见总共应该有 11 个点，但是经过上述操作生成了 14 个点，是因为在统计面要素的全部点坐标时，有一个点被算法当做起点和终点被统计了两次，故需要进行点的去重。去重使用

转换器 Matcher，输出数据的出口选择 SingleMatched 与 NotMatched。

在完成上述操作后，可以对要素的属性表进行清理，使用转换器 AttributeKeeper。

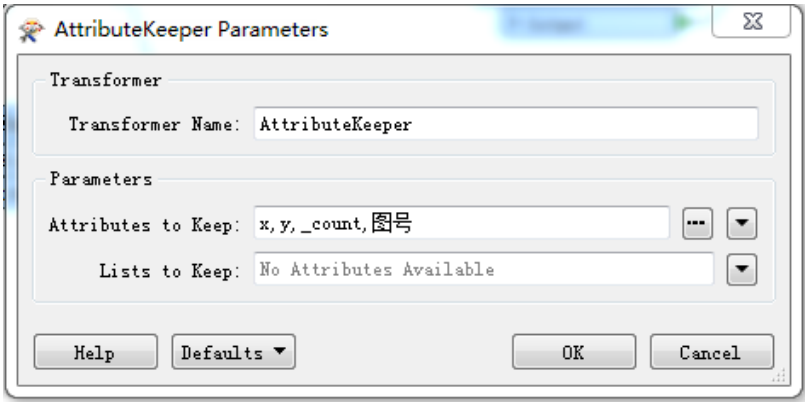

图 2-11

2.2.2 侧面线生成

1）平移点要素：

平移点要素用转换器 Offsetter。

图 2-12

2）连接点要素：

连接原点要素与平移后点要素即可构成侧面线，使用转换器 LineBuilder。选择“_count”字段作为点要素之间的识别字段。

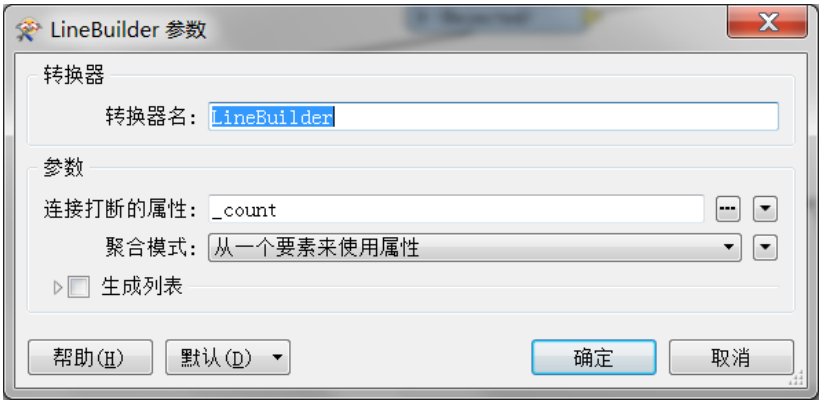

图 2-13

图 2-14 是生成侧面线要素。

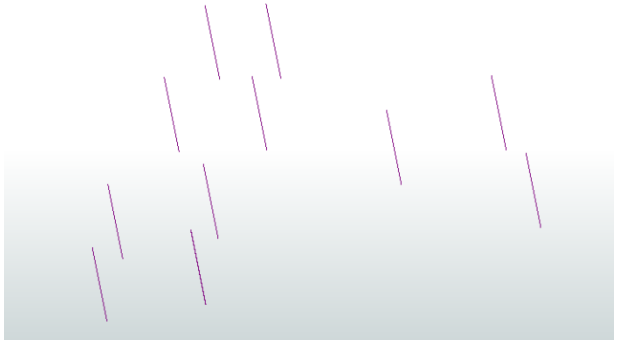

图 2-14

2.2.3 底面线与顶面线生成

1）底面线生成：

首先需要从面要素提取线要素，但是提取到线要素是闭合的连续线要素，而闭合的连续线要素并不利于后面构面的使用，故需要将提取的线要素打断为不连续的线要素。

提取线要素使用转换器 GeometryCoercer。

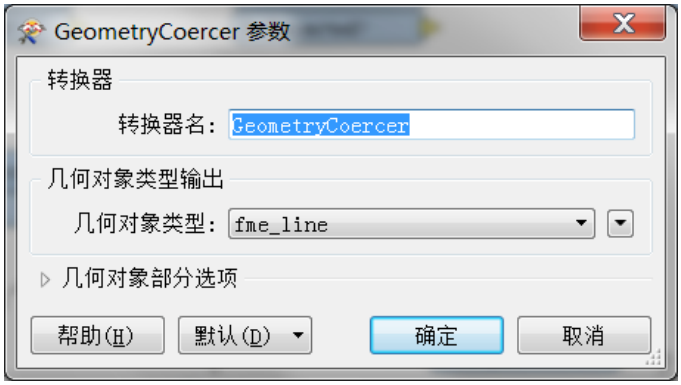

图 2-15

打断上述的线要素的思路是在闭合的连续线要素形成的面内找到与外部线相交的线，利用线相交的关系打断外部线要素，而面的对角线正好满足这个条件。在本实验中利用面的直骨架线代替对角线的功能。提取面的直骨架线使用转换器CenterLineReplacer。

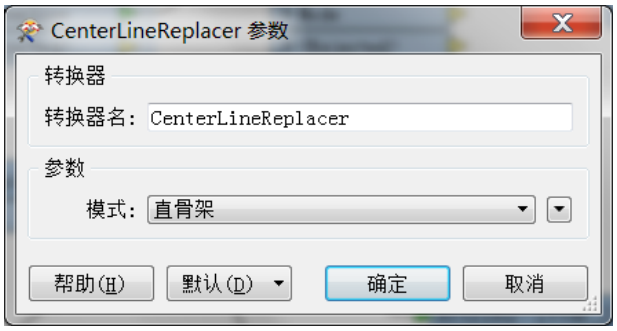

图 2-16

为了区分面的外部线与直骨架线，先给外部线赋予特有的属性，对其编号是实现这一目的最简捷的办法。

图 2-17

在完成上述步骤后，可使用转换器 Intersector 打断相交线，得到不连续的外部线与直骨架线。再使用 Tester 筛选出外部线。

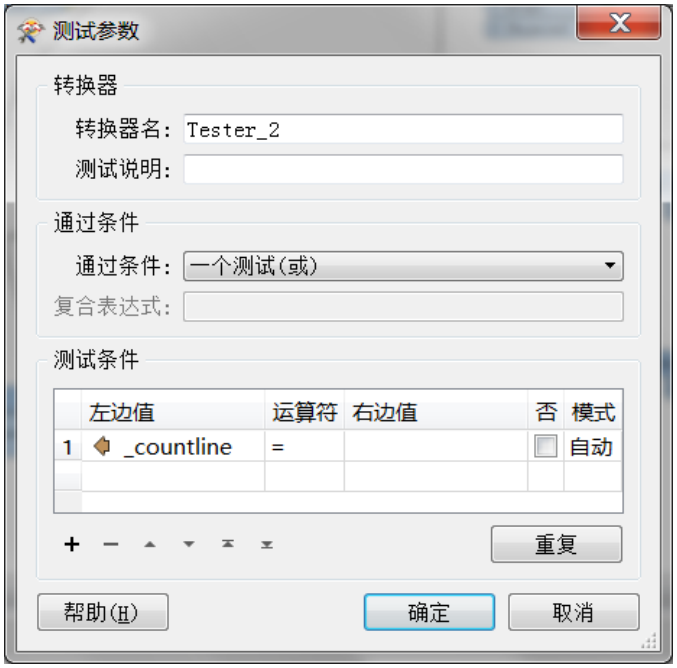

图 2-18

2）顶面线生成：

顶面线的生成通过底面线的平移即可实现。

图 2-19

图 2-20,是顶面线、底面线、侧面线的展示。

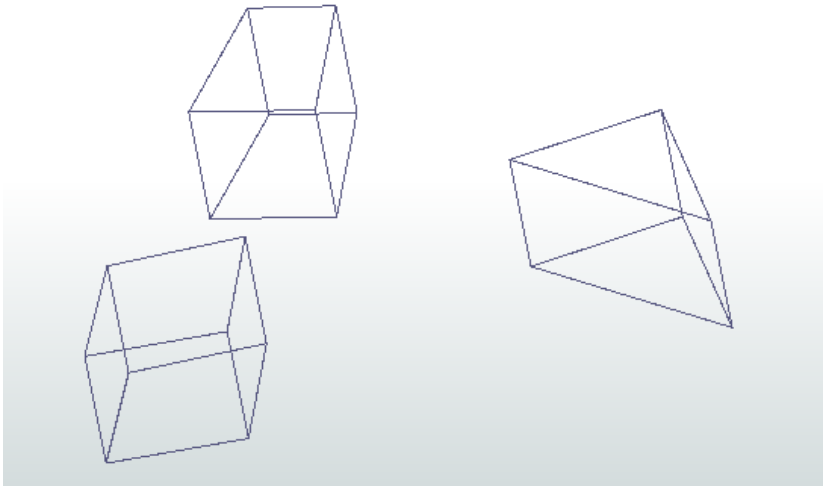

图 2-20

2.2.4　2.5D 面要素生成

由于顶面线、侧面线与底面线直接构面的效果不理想，故将三组线要素中所有的相交线打断。使用转换器 Intersector_2。

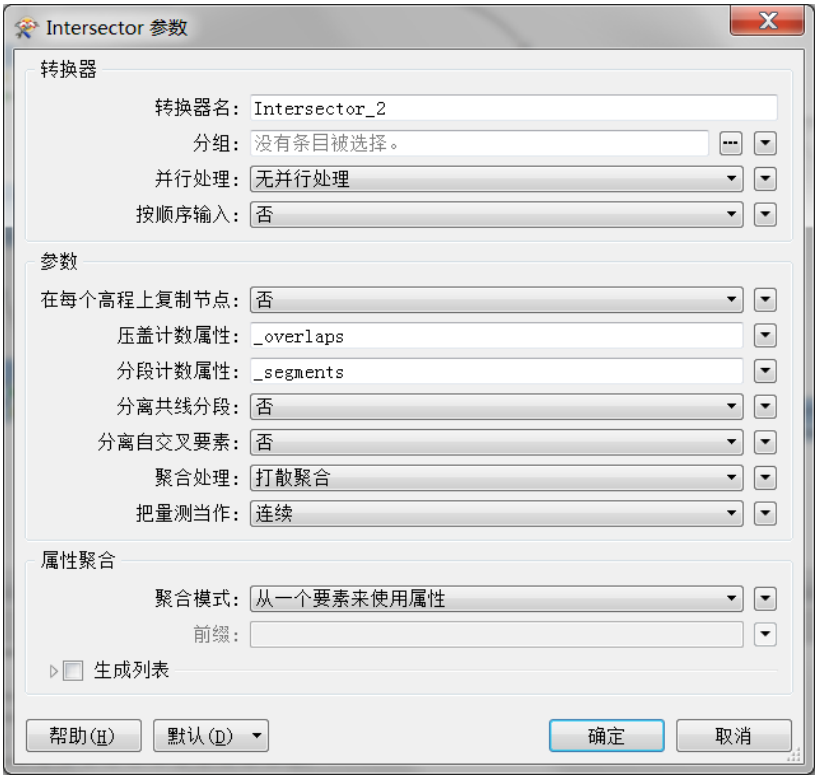

图 2-21

对打断的线要素进行构面，即可得到 2.5D 面要素。使用转换器 AreaBuilder。

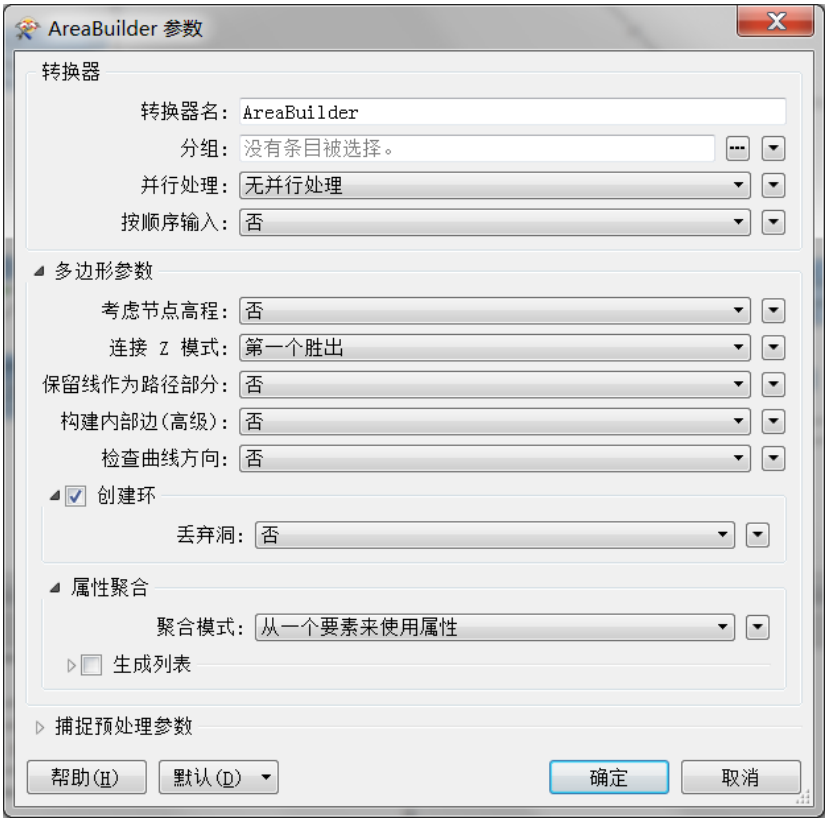

图 2-22

图 2-23 是 2.5D 面的展示：

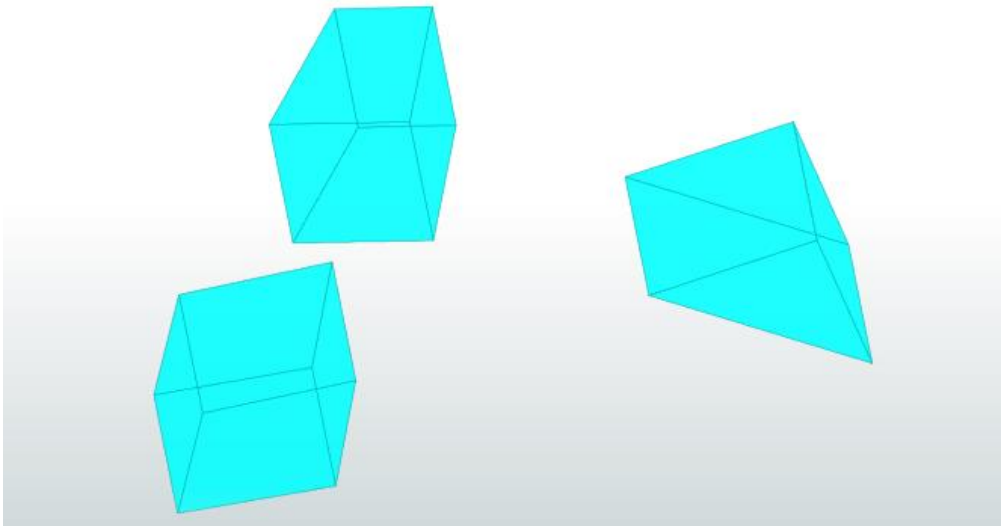

图 2-23

3 总结

本实验实现了二维视图下的面要素拔高。在实现面要素拔高的过程中实现了面折点坐标提取、面折点点要素提取、要素平移、点分组并连线、面外部线提取、面骨架线提取、相交线打断、闭合线构面等功能。二维视图下的面要素拔高主要用于电子地图中建筑物的简易拔高，这种假三维的建筑物拔高虽然不及真实三维视图下的建筑物拔高（真实三维视图下的建筑物一般具有三维分析功能），但有助于缩小电子地图的数据量以及无需三维场景。

本实验不足在于没有单独输出侧面、顶面及底面。