# 2023 集成电路 EDA 设计精英挑战赛

# 作品设计报告

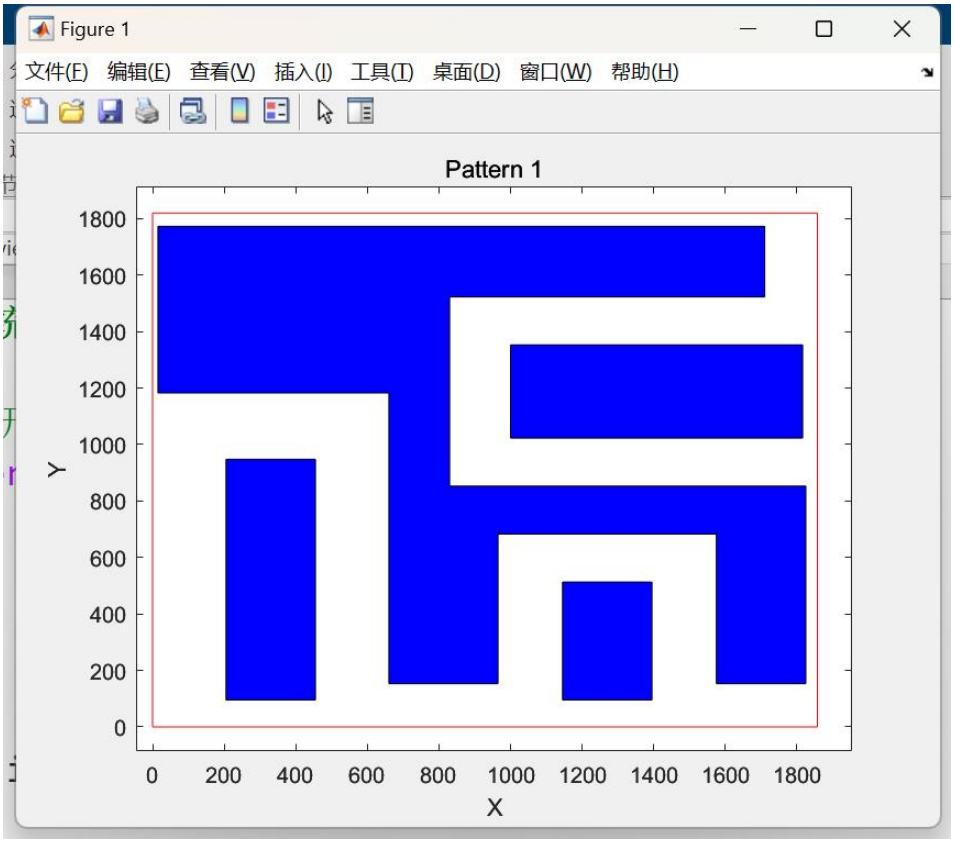
## 赛题十：超大规模版图图形匹配算法

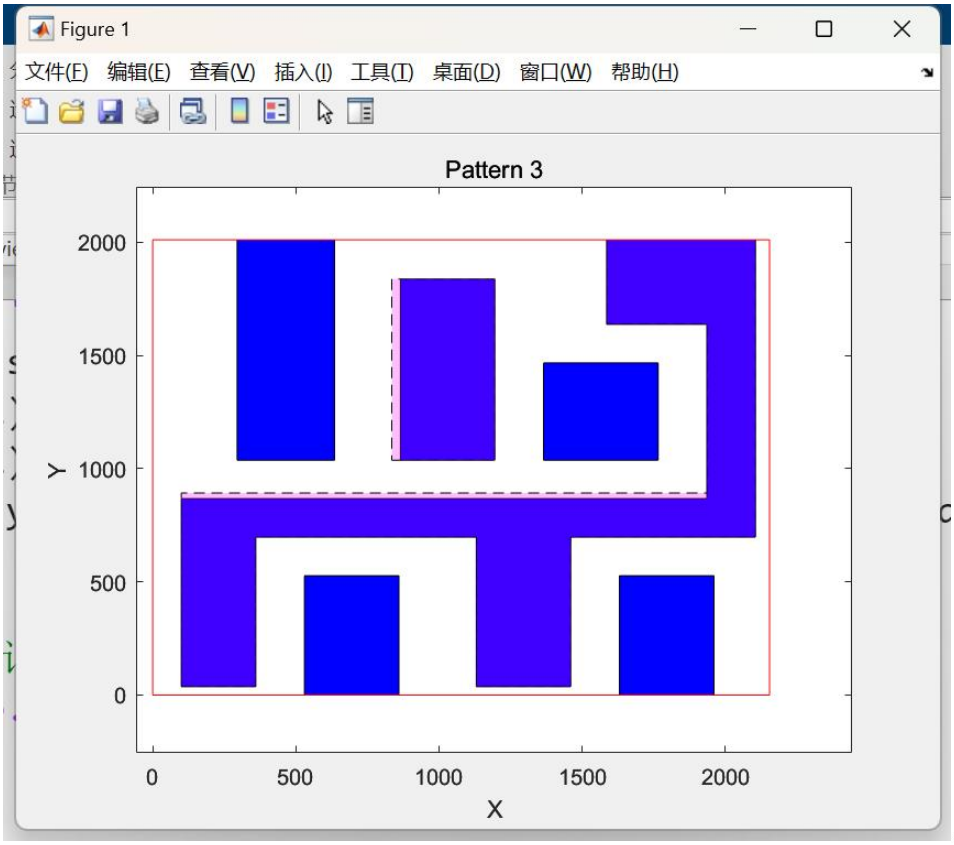
## 参赛队 ID：eda231019

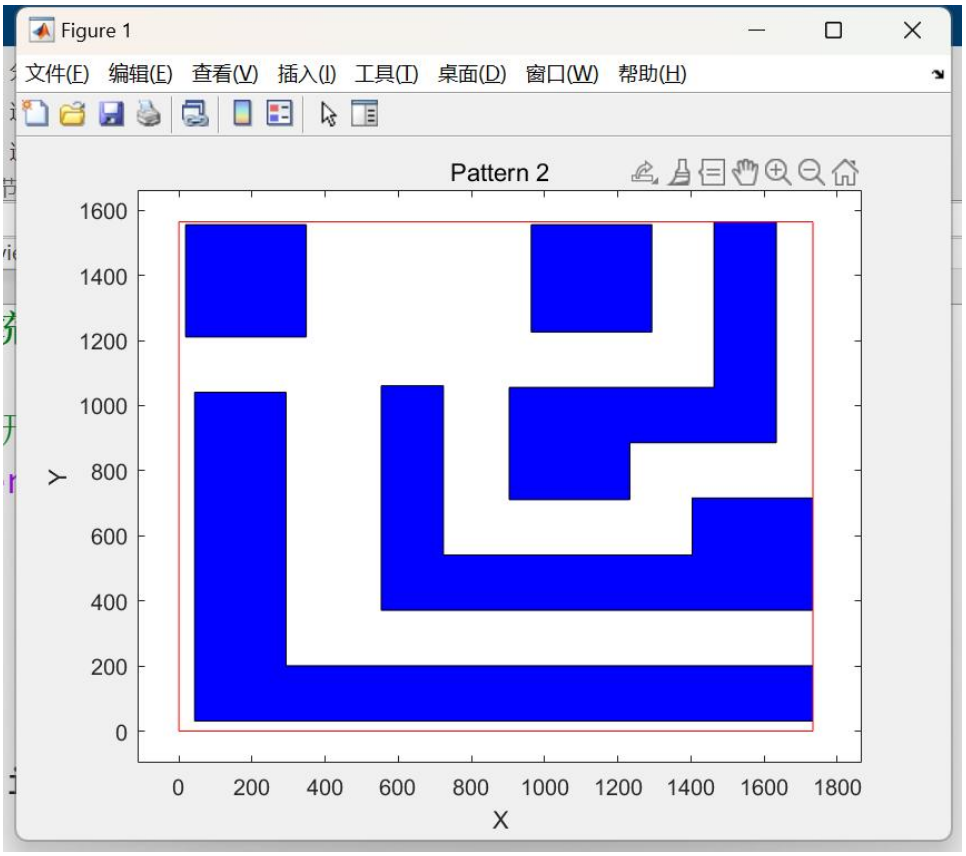
二〇二三年十一月

# 一、自测报告

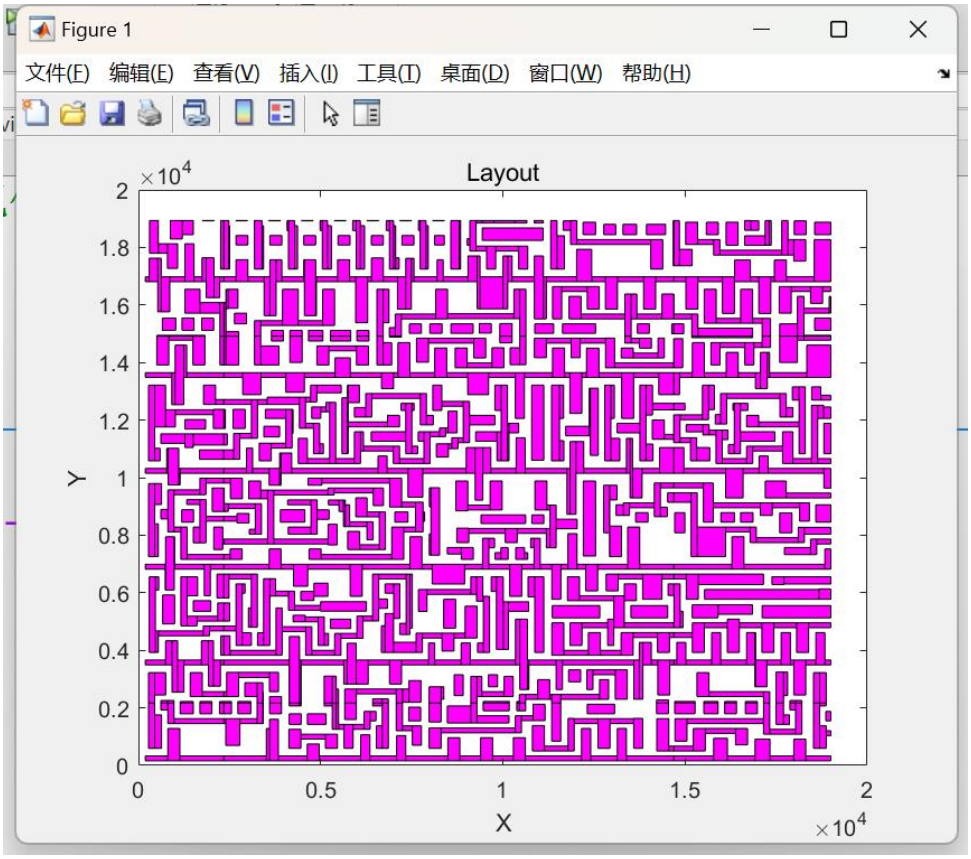
## a、模板样例的可视化



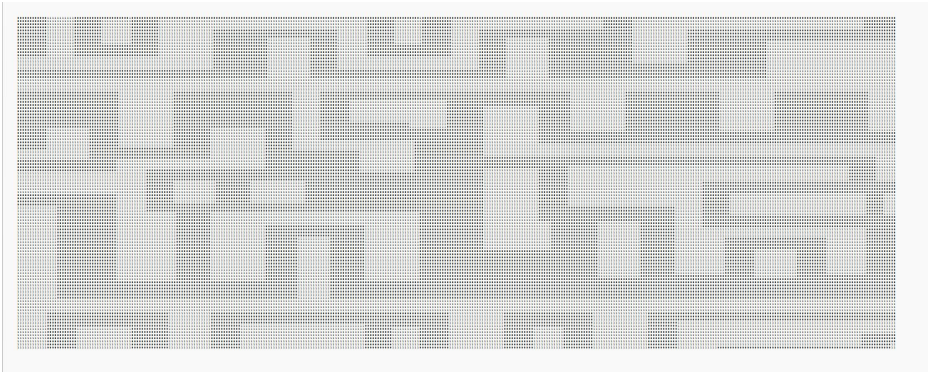





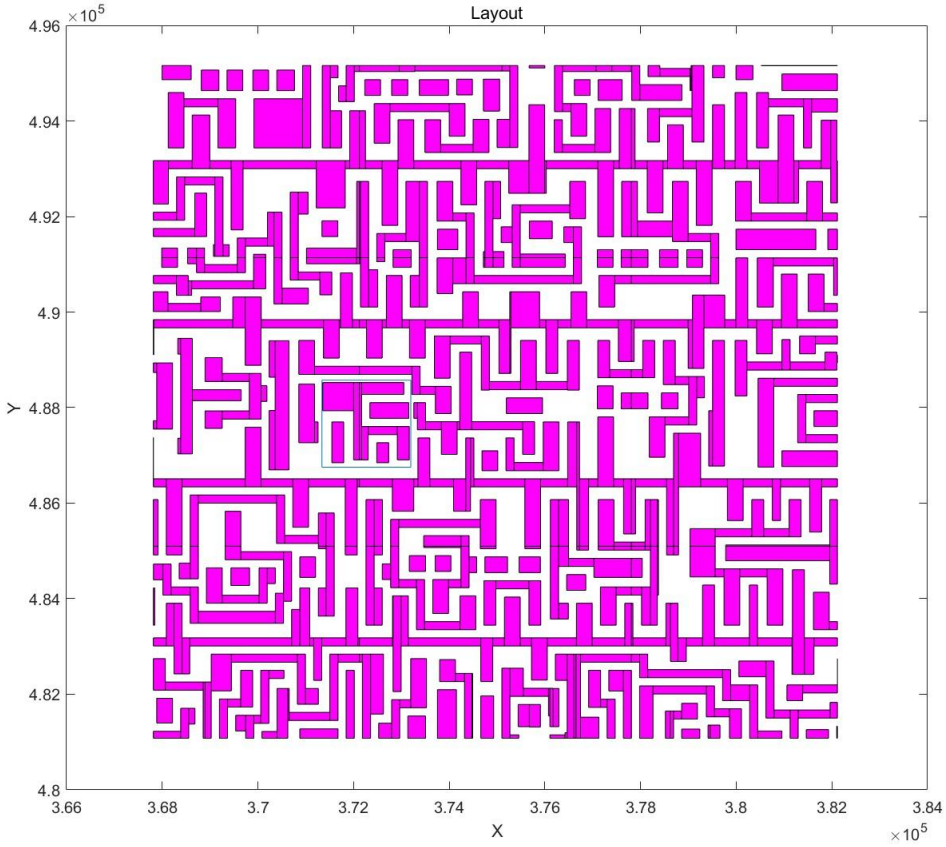

**b、版图分割以及曼哈顿多边形矩形分割**



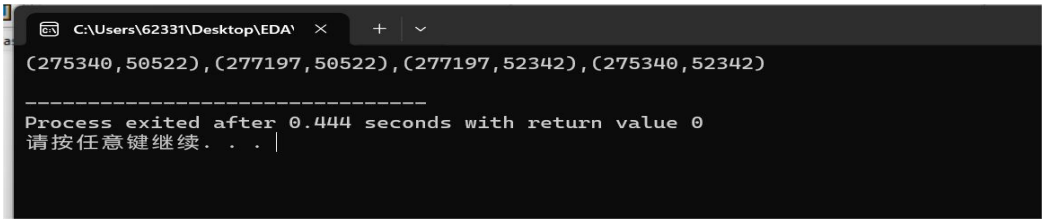
**c、生成矩阵（部分）**



**d、成功匹配**




**结果耗时：**



# 二、采用的算法和思路

## 1、遇到的主要问题及解决方法；

### a、程序设计的流程；

在我们看到赛题之后，我们对超大规模版图图形匹配算法的程序设计流程进行了初步构想，以确保我们后续程序设计方向的正确和连贯性。为此我们小组将此次赛题实现的程序设计流程大致分为：大版图分割；曼哈顿多边形矩形切割；打网格生成矩阵；匹配样例模板。四个大致过程。具体的程序说明在后续的“程序设计”部分和相关附件及服务器文件中呈现。

### b、曼哈顿多边形的矩形分割

版图中图形都为曼哈顿多边形，然而直接对曼哈顿多边形进行处理有着很大难度，为此我们决定先对曼哈顿多边形进行切割。然而我们一开始认为曼哈顿多边形分割是一项简单的工作，但真正自己写代码竟然困难重重，要考虑的细节和特殊情况太多了，为此我们不得不开始查阅大量资料。最终找到利用C++的Boost库中的工具切割曼哈顿多边形的方法，该库提供了强大的几何算法和数据结构，适用于曼哈顿多边形的切割任务。Boost工具库提供的强大功能和高效性使得曼哈顿多边形的处理更加灵活和可行，这一方法的成功应用将有助于提高图形匹配算法在复杂场景下的准确性和效率，使我们实现了高效地对曼哈顿多边形进行矩形分割

### c、数据结构的选择

在实现了对曼哈顿多边形地矩形分割后，用什么样的数据结构存储并进行后续的处理又成了新的问题。再查看相关资料后，我们决定利用R-tree的数据结构对从切割曼哈顿多边形后得到的矩形的左下和右上顶点进行存储及后续的处理。超大规模图形匹配面临着处理海量图形数据的挑战，曼哈顿多边形和R-tree数据结构的结合为此提供了一种有效的解决方案。最终选择这一数据结构，也是因为R-tree具有减小搜索空间，高效索引结构，适应超大规模数据的顶点。显著提高了匹配的效率和准确性，具有广泛的应用潜力，在处理超大规模图形数据中为解决实际问题提供了一种可行而有效的解决方案。

### d、大版图的分割

看到服务器上的大版图后，我们又遇到了新的问题。直接对大版图进行处理和匹配，程序是完全跑不动的，对次我们想到用R-tree进行大版图分割，再对划分后的小版图进行匹配的方法。实验结果证明该方法可行。R-tree作为一种多维索引结构，适用于组织和检索空间数据，在超大规模图形匹配中，我们利用R-tree对大版图进行划分，以建立高效的索引结构。将大版图分解为小块，使得匹配时能够更快速地定位到热点区域，从而提高匹配的效率。

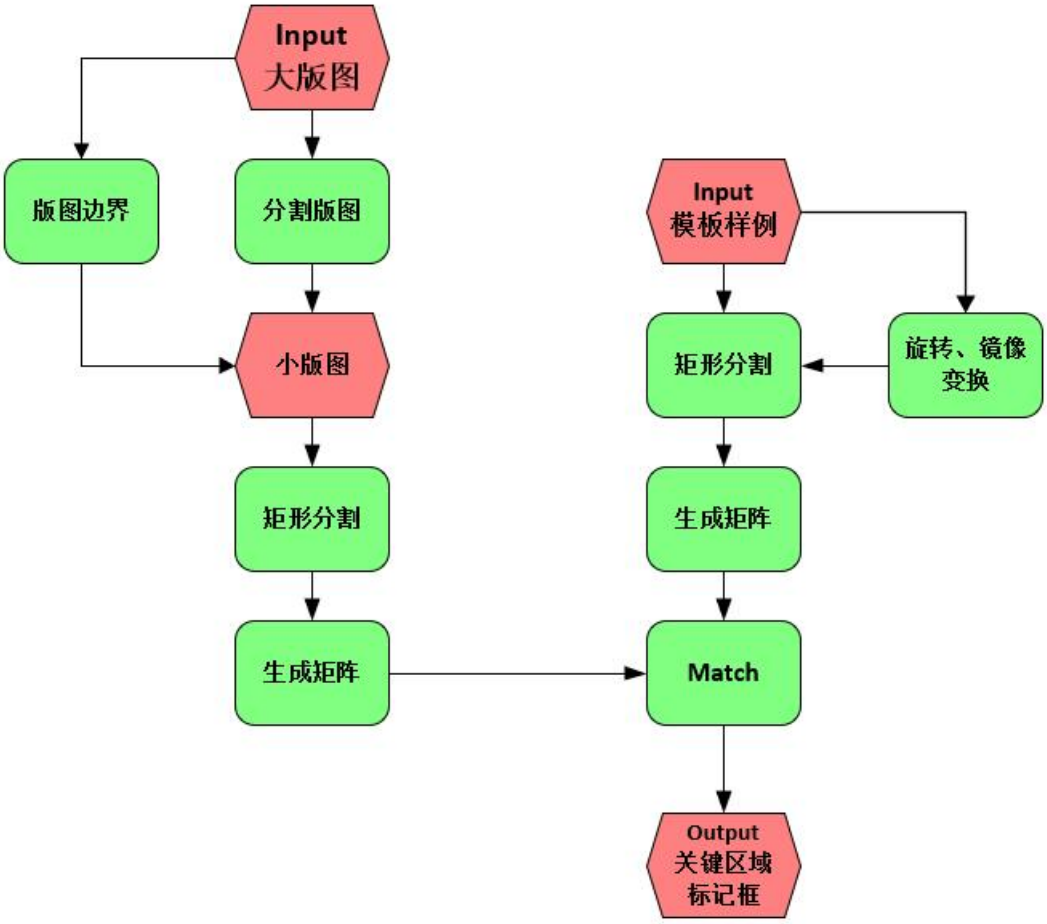
### e、匹配算法的实现

此次赛题的核心匹配算法，我们小组分为三步实现：打网格、生成矩阵、匹配。首先将得到的小版图，选取合适的距离打网格，再将多边形内部的交点设置为“1”，外部的设置为“0”，从而生成对应的矩阵。最后利用KMP与RK匹配算法对小版图与模板样例进行匹配，得到关键区域标记框的位置。

## 2、主体架构和核心算法；

**分割版图、生成矩阵、利用KMP与RK匹配的match算法为我们的核心算法**

**详细代码在我们的附件中呈现**




## 3、算法的时间空间复杂度分析；

### a、利用KMP与RK匹配算法

在对最终矩阵匹配时通过对横向与纵向的匹配，得到关键区域标记框

其时间空间复杂度为O（$n^2$）

### b、分割版图

通过R-tree对大版图进行分割的算法复杂度为O(log n)

### c、生成矩阵

对版图生成矩阵算法的时间空间复杂度为O（$n^2$）

## 4、算法主要参考文献

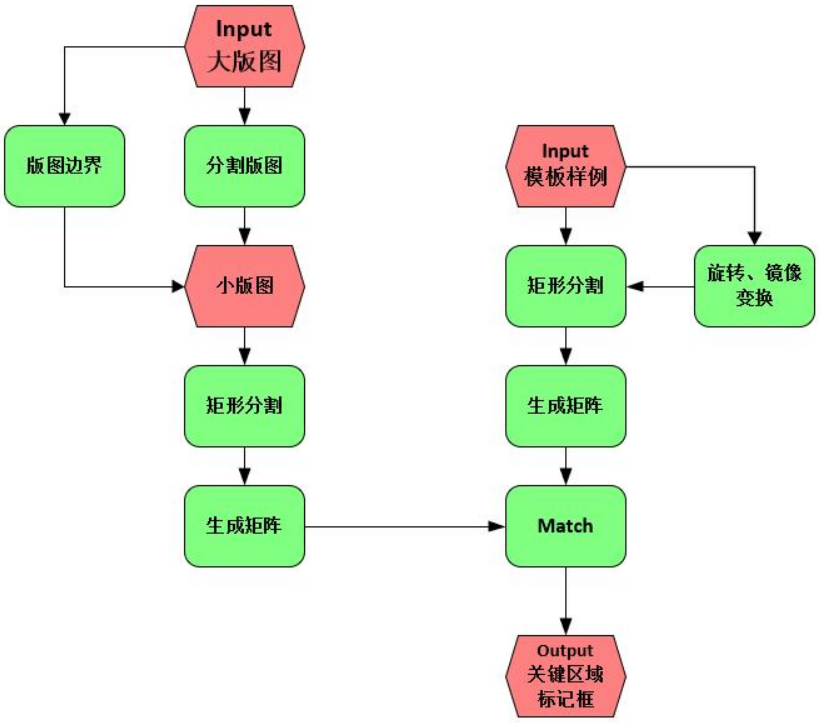
[1] Yao H, Sinha S, Chiang C, et al. Efficient process-hotspot detection using range pattern matching[C]//Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design. 2006: 625-632.

[2] Knuth D E, Morris, Jr J H, Pratt V R. Fast pattern matching in strings[J]. SIAM journal on computing, 1977, 6(2): 323-350.

[3] Zhu R F, Takaoka T. A technique for two-dimensional pattern matching[J]. Communications of the ACM, 1989, 32(9): 1110-1120.

# 三、程序设计

### a、程序设计架构




### b、主要模块说明（详细代码在附件中）

分割版图：通过R-tree将大版图分割成50*50的小版图，从而实现高效、快速的版图分割；

版图边界：通过确定版图边界，使得版图分割尽可能地合理，从而避免因分割而造成的区域不完整；

矩形分割：通过使用C++boost库工具将曼哈顿多边形分割为长方形，并通过链式结构存入R-tree中，但在我们后期的讨论和实验中发现，将输入R-tree的数据结构转换为图，可以实现更高效的矩形分割，然而因为时间与近期课程任务紧张的因素，这一想法目前很可惜还未实现，我们仍采用链式结构将对曼哈顿多边形分割后的矩形存入R-tree中；

生成矩阵：通过选取样例中的最短间距作为打网格的间距，在分割后的小版图打网格后，更具交点的一侧的边数奇偶数判断其在多边形的内部还是外部，内部标记为“1”,外部标记为“0”，用布尔的数据减少程序开销，从而进行接下来的矩阵匹配。

Match：通过纵向的KMP算法和横向的RK算法进行小版图与模板样例的矩阵的匹配，两种高效的匹配算法能使程序更加精准高效，同时为了进一步提高效率、增加精度我们通过打网格间距的不同，先进行粗匹配找到与目标矩阵相似的部分，再通过细匹配确定是否正确，从而实现高效、快速、精准的匹配算法。

### c、第三方库

我们程序中主要用到的第三方库是boost库，用其实现曼哈顿多边形的矩形切割

相关链接：https://www.boost.org/