2D 凸多边形机器人

最短路径运动规划问题的实现

问题的描述

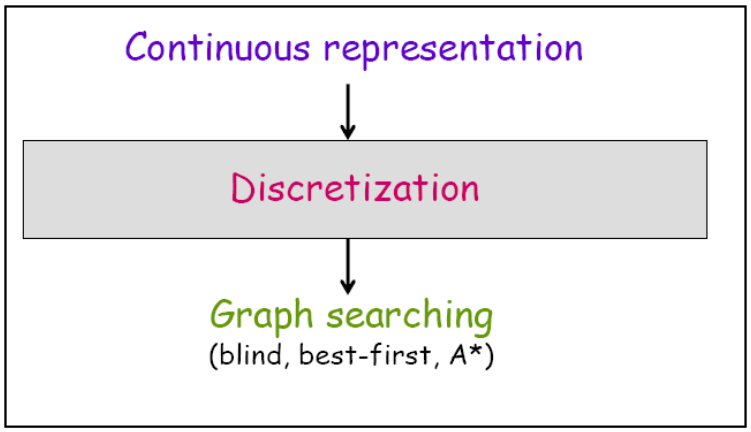
机器人学（robotics）的最终目标之一，就是设计出独立自主的机器人。在指挥这种机器人时，我们只需告诉它去做什么，而不必告诉它如何去做。其中尤为重要的一个方面就是，机器人应该懂得如何为自己的运动进行规划。只要这样的路径存在，我们所给出的算法总是能够把它找出来。而这条路径可能要兜一个大圈，或者要转很多不必要的弯。在实际的环境中我们总希望找到一条最短的路径(按照欧氏空间的距离度量)来实现我们的运动规划问题，而不仅仅只是其中任意的一条可行路径(一条路径是可行的定义为按照这条路径进行运动的机器人不会与多边形障碍物的边界发生碰撞）。

本次实验主要解决的是二维空间上的运动规划问题。具体说来，我们要实现的算法将会解决二维空间上的凸多边形机器人的运动规划问题，而对于障碍物，只要是简单多边形即可，没有凹凸的限制。对于这样一个约束下的问题，我们可以形式化为如下的表述：

- *给定平面上的一组简单多边形障碍物集合$P = \{p_1\ p_2 ... p_n\}$和一个凸多边形机器人R的起点和终点，计算出从起点到终点的最短可行路径s。*

解决方案总体描述

由于给定的平面空间是连续的，我们无法处理。因此先要对于给定的连续型的问题进行某种意义上的离散化。离散化的方法有很多，我们将在后面的章节里详述离散化的算法。离散化之后的空间变成了一张图G，我们利用图论里面的单元最短路径算法就可以求解，从而得到原问题的解。解决方案的总体流程如下图所示。



算法介绍

预处理算法

在求解面积不等于0的凸多边形机器人穿越由简单多边形构成的障碍空间的寻径问题时，需把此类问题转化为点机器人穿越障碍空间的寻径问题。通过对障碍物空间与对应机器人凸多边形进行Minkowski和，形成新的障碍物空间此状态空间即为参照点的障碍物空间。求解参照点在新障碍物空间的寻径问题即可。

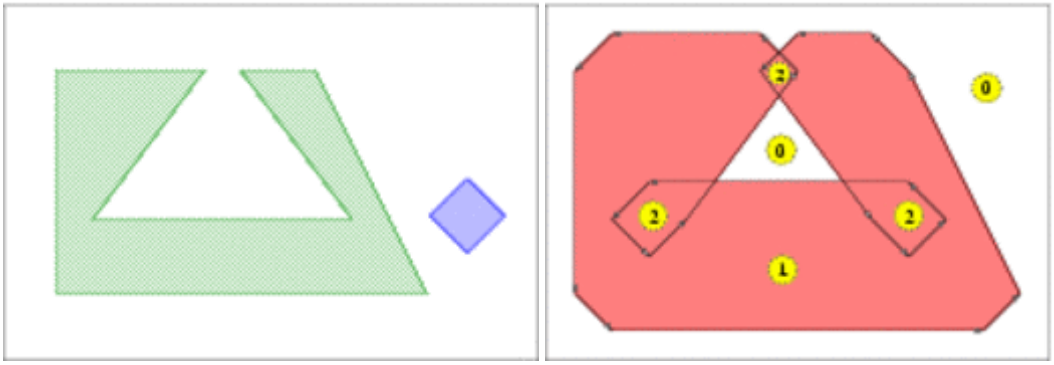
有如下定理：(引用cgaa)：

设R为沿平面做平移运动的一个机器人，P为任意障碍物。则P所对应的新障碍物空间为

$$P+(-R(0,0))$$

其中R(0,0)为R将参照点平移原点形成的新多边形。

所以做过Minkowski和后障碍物多边形会“膨胀”。如下图：



Minkowski和的数学表达式如下：（摘自cgal）

$$A + B = \{\mathbf{a} + \mathbf{b} \,|\, \mathbf{a} \in A,\ \mathbf{b} \in B\}$$

具体算法不在详述请参照cgaa。注：对于凹多边形与凸多边形的Minkowski和时首先将凹多边形进行凸剖分，分别Minkowski和后求并下面介绍凸剖分方法。

求Minkowski首先是要将一个任意的简单多边形进行凸剖分。我们采用的是去耳算法。根据双耳定理，对于一个任意多边形来说至少存在两个耳朵。因此可以循环的切掉多边形的耳朵，直到它所有的凹内角都被破坏成为一个凸多边形为止。算法流程如下。

ToConvexPolygon(SimplePolygon P)
While P is not convex polygon

```
Do if P is triangle
    then return
endif
    else do
        CutEar(P);
    end
end
```

离散化方法

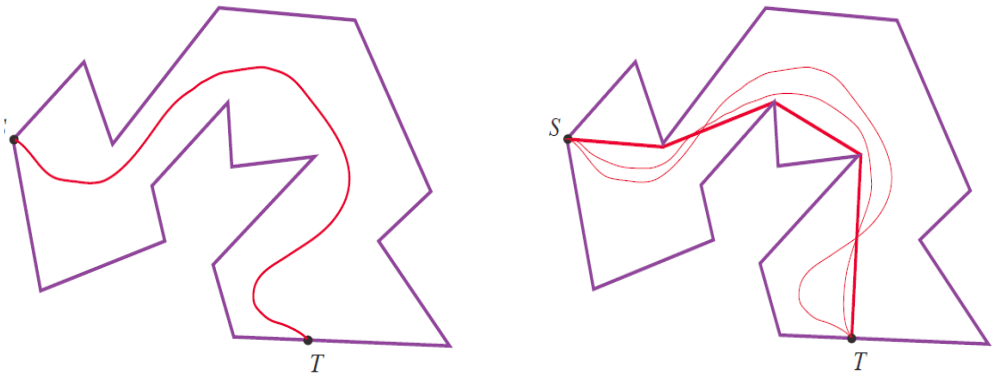
为了实现二维的机器人的运动规划，我们调研了很多关于运动规划算法的文献。现将我们调研的结果总结如下。

正如前文提到的，运动规划是人工智能和计算几何领域里面广泛研究的问题。非正式的，运动规划问题可以描述成为：给定一个几何空间以及这个空间里的一些禁止空间，给定起点和终点，求出起点和终点之间的一条路径。按照这条路径是否是最短路径，可以将问题细化为任意运动规划和最短路径运动规划，此次实验的主要目的是实现一个二维版本的最短路径运动规划；按照运动物体的形状运动规划问题又可以分为点机器人运动规划，凸多边形机器人的运动规划以及任意形状机器人的运动规划，此次实验的主要目的是实现一个二维凸多边形机器人的运动规划。

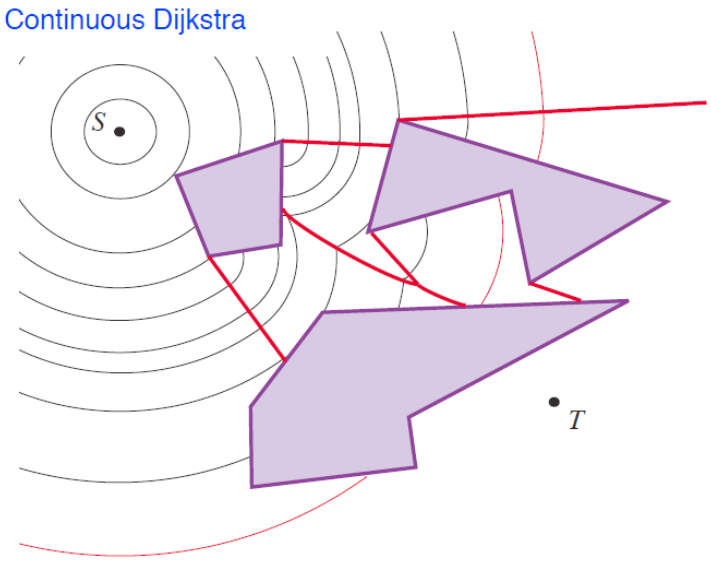
二维凸多边形的运动规划问题可以按照如下的算法框架进行求解：第一步，先利用Minkowski和方法将运动空间进行变换，把凸多边形的运动规划问题转化为点的运动规划；第二步，利用点的最短路径运动规划算法求得问题的解。下面的部分主要阐述如何解决点机器人的最短路径规划问题。

二维点机器人的最短路径规划算法是计算几何领域被广泛研究的一个问题。按照算法的思路可以分为如下两类：

- 可见图法。首先给出如下定理：起点和终点之间的任意一条路径，如果是一条最短路径，那么这条路径上的拐点都是障碍物的顶点。简单的证明入下图所示：设想如果不是这样的话，把这条路径看作一条可以自由伸缩的橡皮筋，那么当它收缩的时候，必然会遇到障碍物的顶点或边而这样一条收缩后的路径长度势必小于原来的松弛的路径长度。因此我们的问题就可以转化为：计算给定的障碍物的顶点之间可见性图，如果两个顶点之间可见，那么在一个图G中这两个顶点之间就有一条边，权重为两顶点之间的距离。得到可见性图之后，我们可以用图论中的 Dijkstra 算法得到一条最短路径，这条最短路径也就是源问题的解。



- 最短路径图方法。这种方法又被成为连续 Dijkstra 方法。我们将整个空间进行一个区域划分，使得属于同一个区域的所有点到原点的最短路径上的顶点相同。具体说来，我们可以将构造最短路径图的过程看成是点波源发出的波的传播过程。惠更斯原理告诉我们，波的传播永远走的是最短路径。因此我们可以模拟一个从源点发出的波，用它对二维有障碍物的空间进行平面扫描，只不过我们这里用的不是线扫描二是用一系列逐渐扩大的圈，波前的形状决定了最短路径图的区域划分。如下图所示。



算法复杂度。已经证明了二维凸多边形的运动规划问题的复杂度的下界是O(nlogn)，而且这是一个紧的下界。上面提到的最短路径图方法就可以有一个O(nlogn)复杂度的实现。但是由于实现难度太大，本次试验中我们采取的是一个复杂度为$O(n^2 logn)$的算法。算法的具体过程见文献[1]，此处不再赘述。

解决问题中涉及的其他算法

多边形求并

经过计算障碍物多边形和机器人的minkowski和，障碍物会“膨胀”，从而产生重叠的区

域，此时，需要对多边形进行求并操作。

在这里，我们简化了两个多边形的求并的含义。根据应用特性，多边形内部的空洞肯定不会在机器人循径时找到，因此，在求并时，我们简单的将两个简单多边形并集内部的空洞也作为并的一部分，这样，最终求并的结果是不包含空洞的简单多边形。

两个简单多边形A，B求并算法的思路如下：

- 首先保证两个多边形按逆时针旋转；
- 遍历多边形 A 的所有顶点，找到其中位于多边形 B 外的顶点 p；
- 从 p 出发沿着多边形 A 逆时针旋转，如果旋转的边与 B 的边有交点，则从交点出发沿着 B 物体旋转。如此反复，直到重新遍历到点 p；
- 将经过的多边形的顶点和交点加入到新生成的并集当中；
- 在循径过程中对在边界重合以及在端点处相交的情况进行特殊处理；
- 如果没有产生相交，说明两个多边形的边界没有交点。此时，两个多边形可能没有交集，或者存在重叠情况。只要判断一个多边形顶点是否在另一个多边形内部，即可以判断两个多边形的关系，如果多边形存在重叠情况，则返回较大的多边形。

多边形裁剪算法

经过求并后，形成的多边形可能超出了我们限制的边界，此时，需要对多边形进行剪裁操作，裁掉多边形位于边界范围内的部分。

多边形裁剪算法的主要思路如下：

- 保证两个多边形按逆时针旋转
- 找到多边形存在于边界范围内部的一点 p。
- 从 p 点出发逆时针旋转，如果经过的边和边界相交，说明多边形存在超出边界的区域，此时按照边界逆时针旋转，找到下一个边界和多边形的交点，并从该点出发沿着多边形逆时针旋转。重复该过程直到回到起点 p。
- 在循径过程中加入多边形的顶点，多边形和边界的交点，以及边界的拐点。这些点沿逆时针方向旋转形成裁剪后的多边形。

实现与测试

实验环境

操作系统：Windows XP　　开发环境：Visual Studio.NET 2005　　界面：MFC

程序总体架构

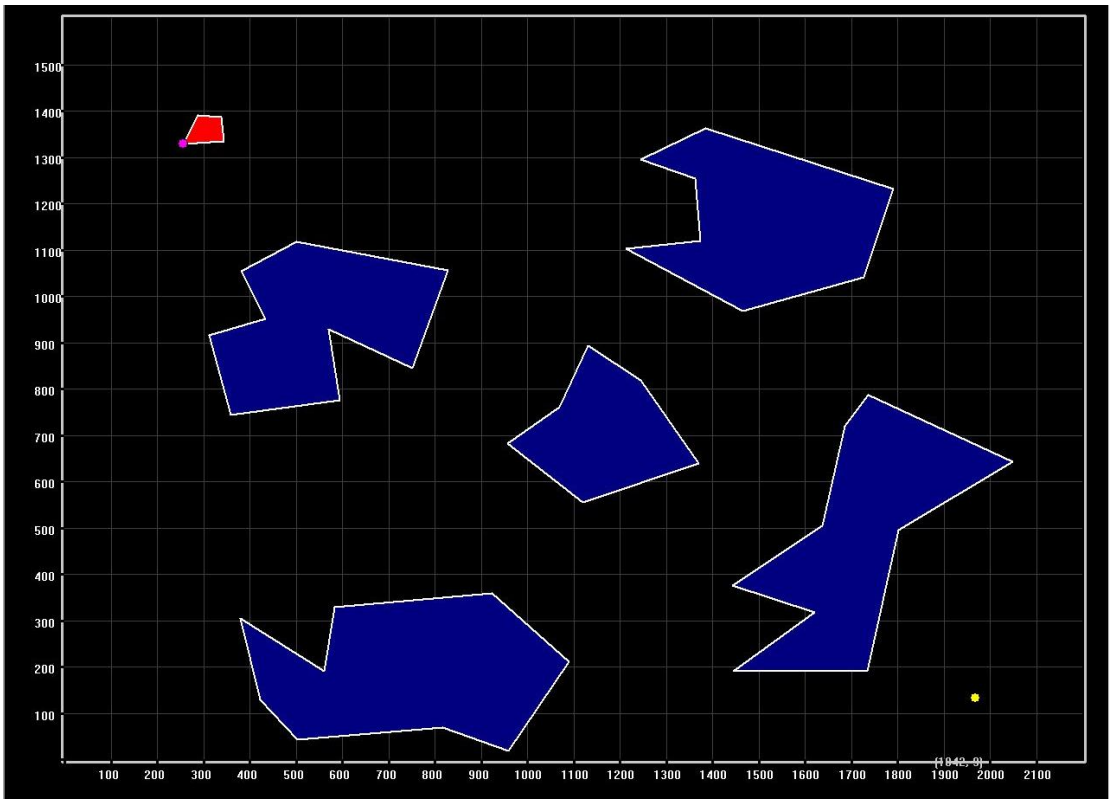
程序自上而下定义了数据结构来实现平面内的运动规划问题，程序框架主要分为以下几

个层次。

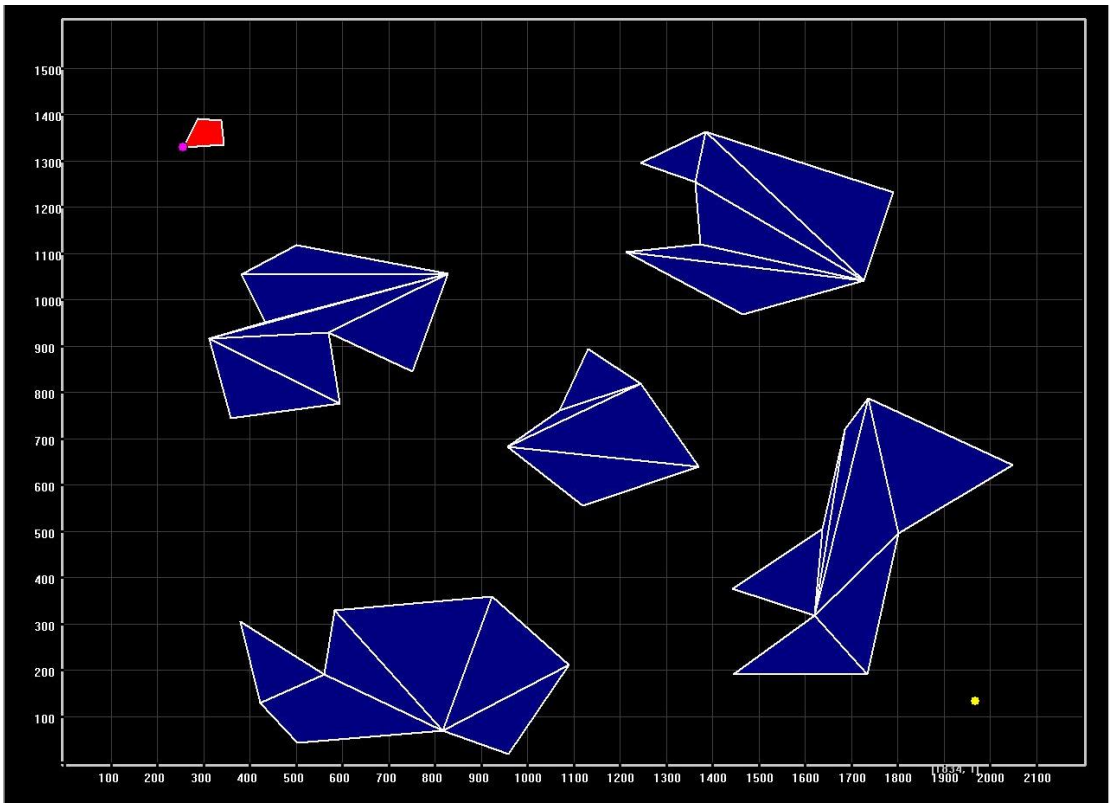
- 基本数据：包括二维点 Point2d，二维向量 Vector2d，二维直线 Line2d，并定义了计算顶点之间位置关系的算法。
- 基本结构：包括简单多边形 SimplePolygon，多边形集合 PolygonSet，可见图 VisibityGraph 等，以及这些数据结构上常用的算法。
- 算法数据：包括在编辑状态下存储数据的类 EditData 和运动规划过程中存储数据的类 MotionPlanning。
- 算法：包括多边形求并，简单多边形凸划分，求 minkowski 和，生成数据以及实现动画的算法函数
- 显示控制：包括绘制基本界面元素，以及算法中的数据。
- MFC 框架：组合数据，算法和控制元素。

算法执行过程

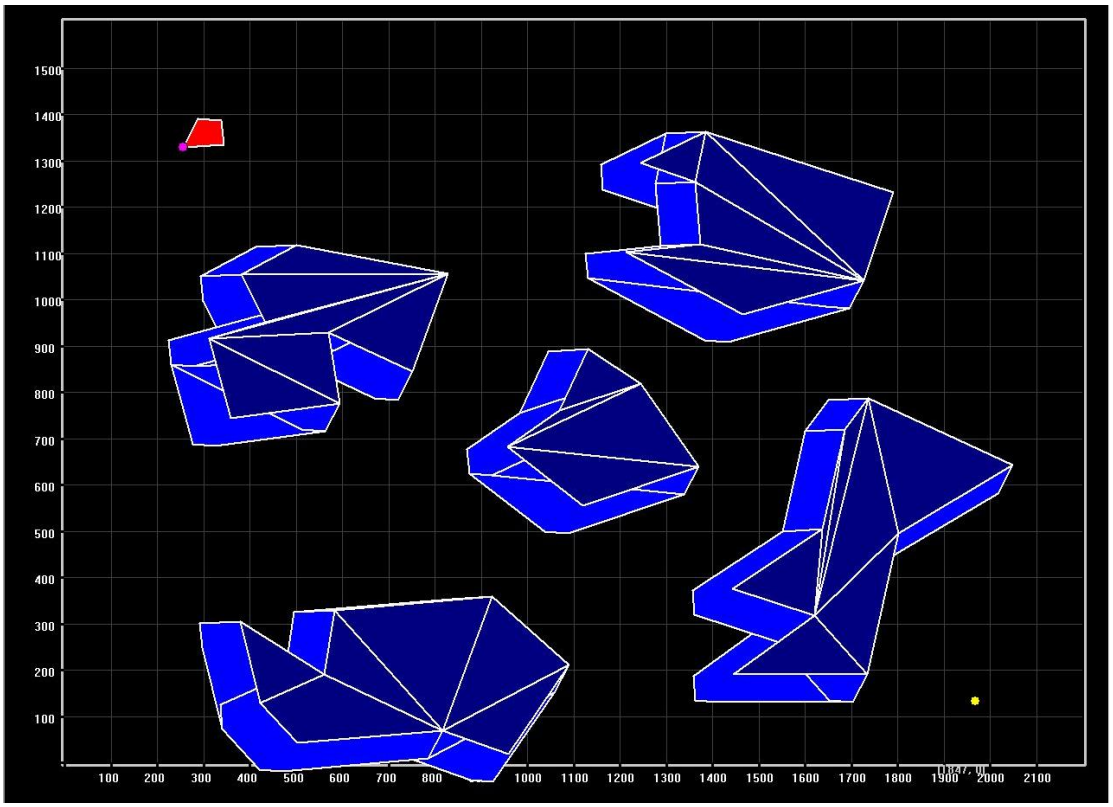
首先，用于通过手工编辑或者导入文件进行输入机器人，障碍物，以及目的点。然后算法按照以下步骤构造路径。



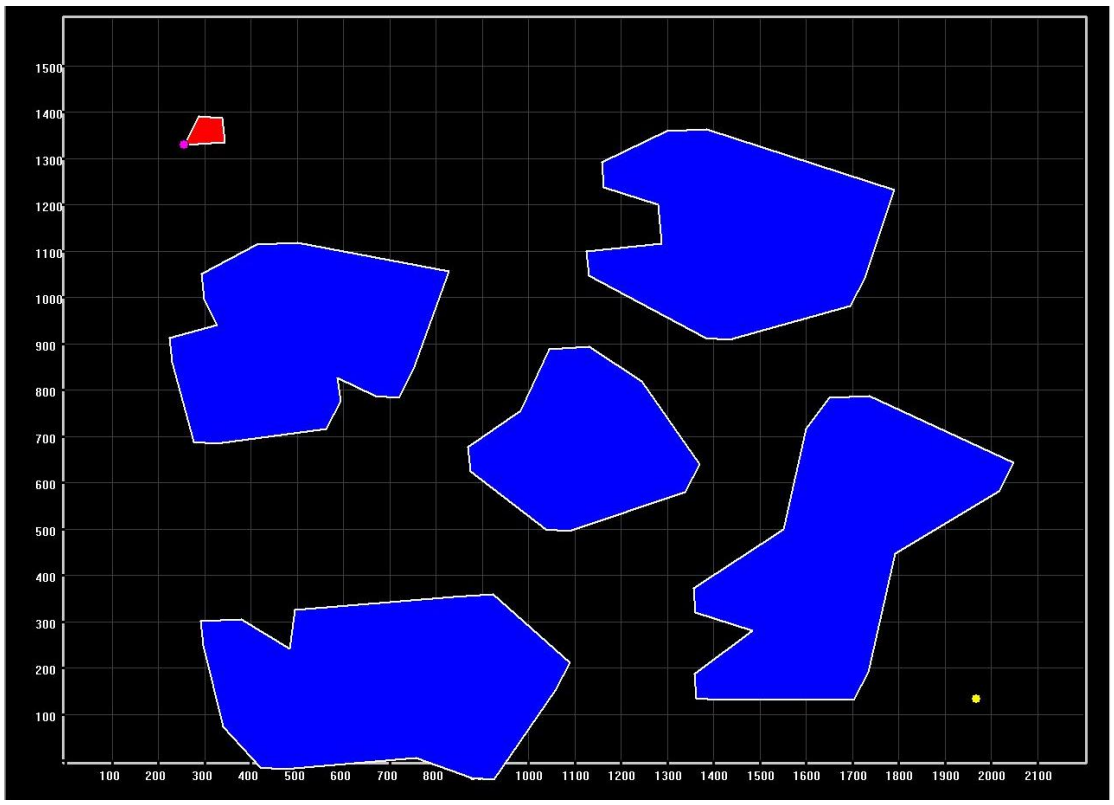
Convex Subdivide ：凸划分。对于一般的简单多边形障碍物，我们需要先对其进行凸划分，才能够进一步求解 minkowski 和。这一步的目的就是对凹多边形急性凸划分，划分得到的每个部分都是凸多边形。



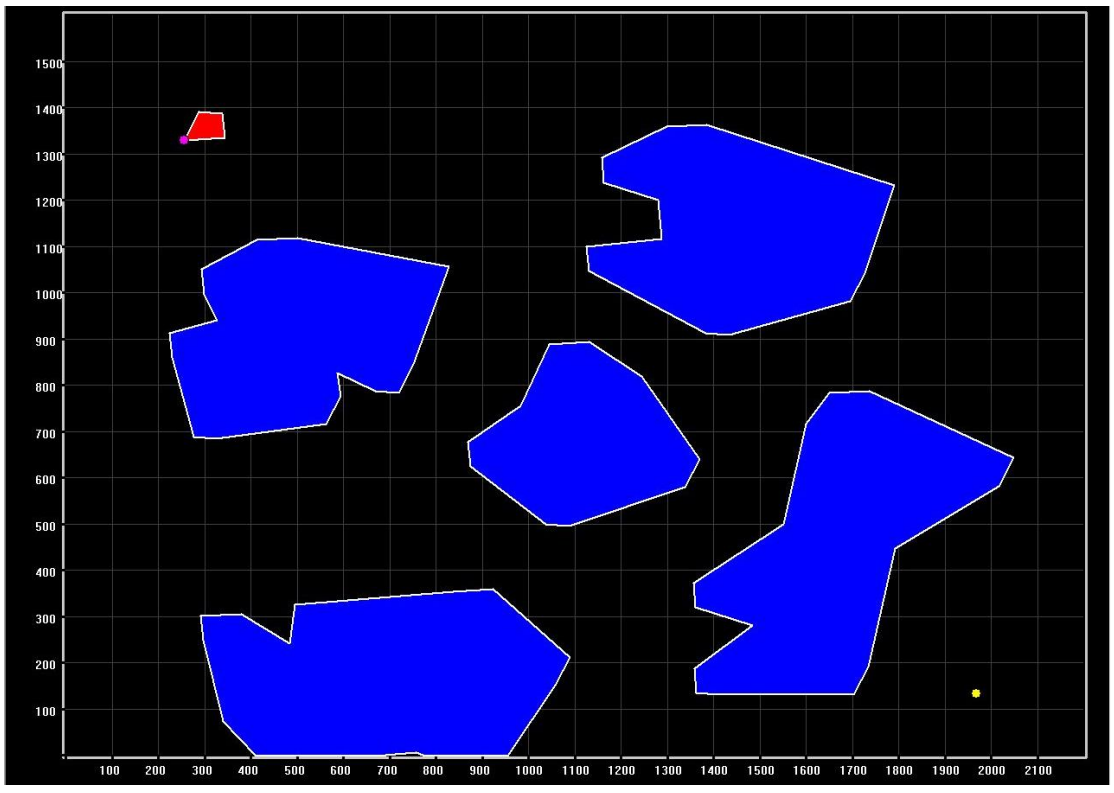
Minkowski Sum：为了求解凸多边形的运动规划，我们需要先求解障碍物多边形和机器人多边形的 Minkowski 和。在这一步，计算上一步划分得到的每个凸多边形的 Minkowski 和，结果是一个新的凸多边形集合。



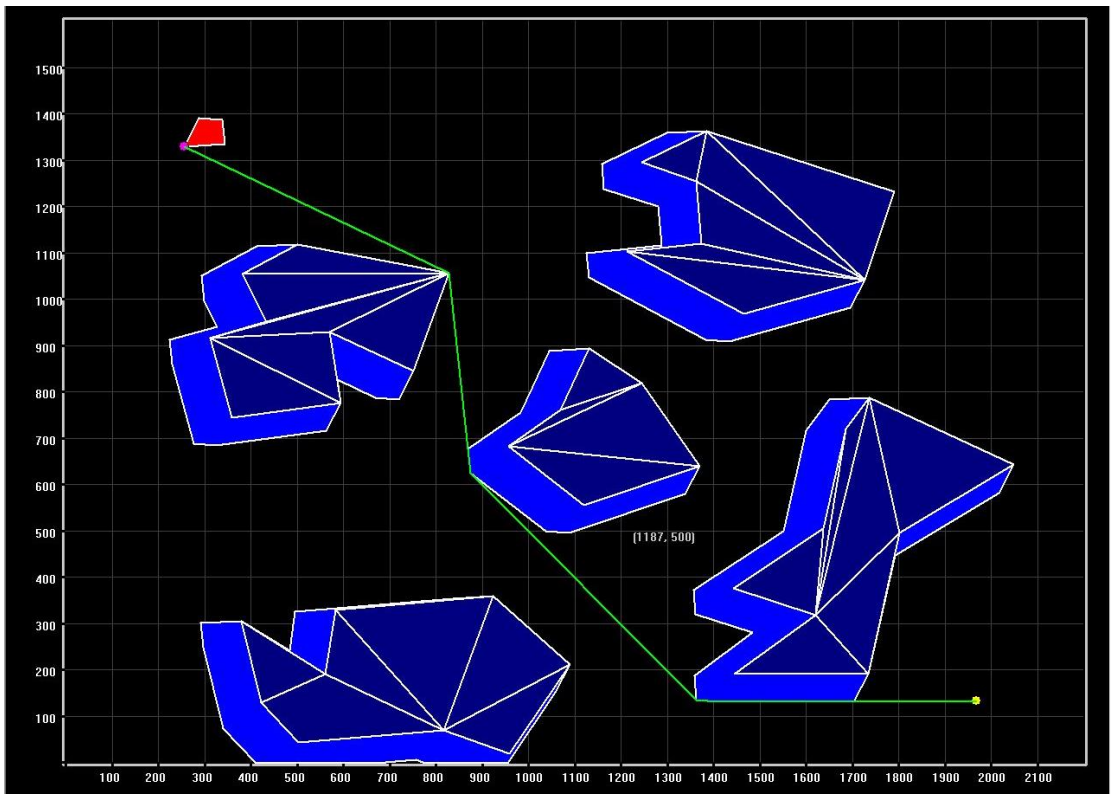
Union：上一步得到的凸多边形会产生重叠和交叉，这一步对生成的 Minkowski 和多边形求并，得到没有交集的凸多边形区域。



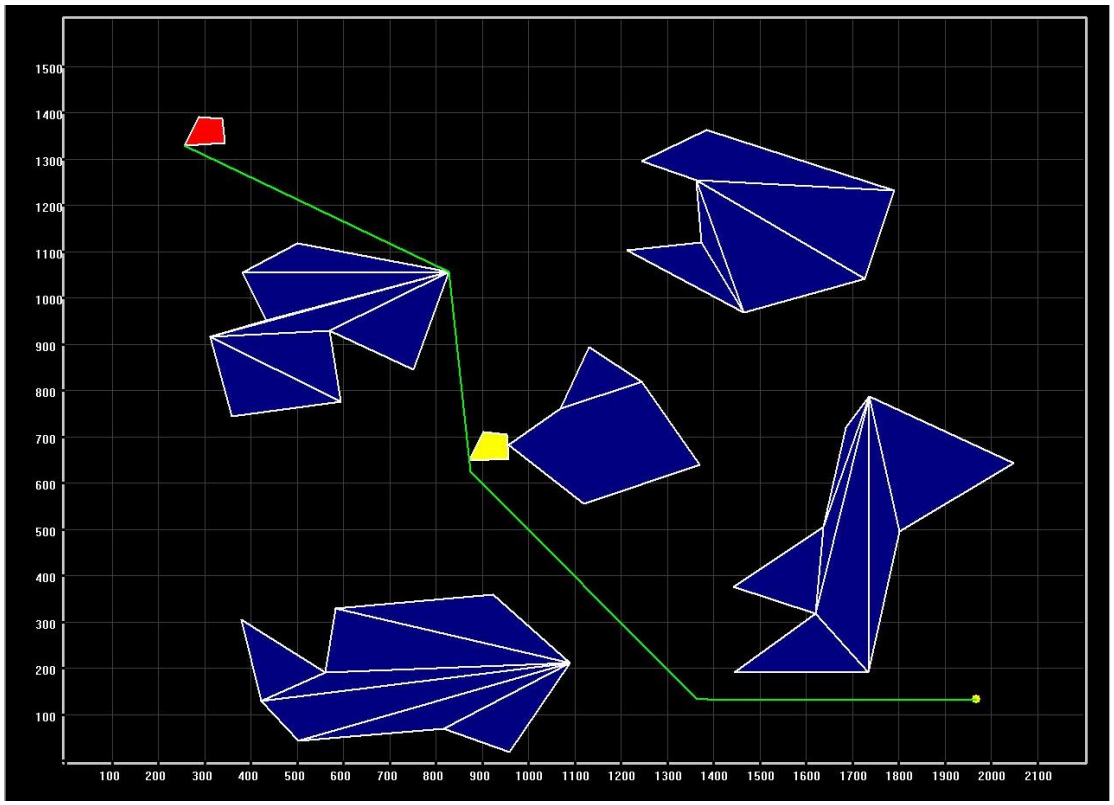
Cut in Window：如果生成的凸多边形区域部分超出了我们限定的边界，则对超出边界的部分进行裁剪。



Construct Path：利用裁剪的到的最终的凸多边形区域构造可见图，并从可见图中找出最短路径。



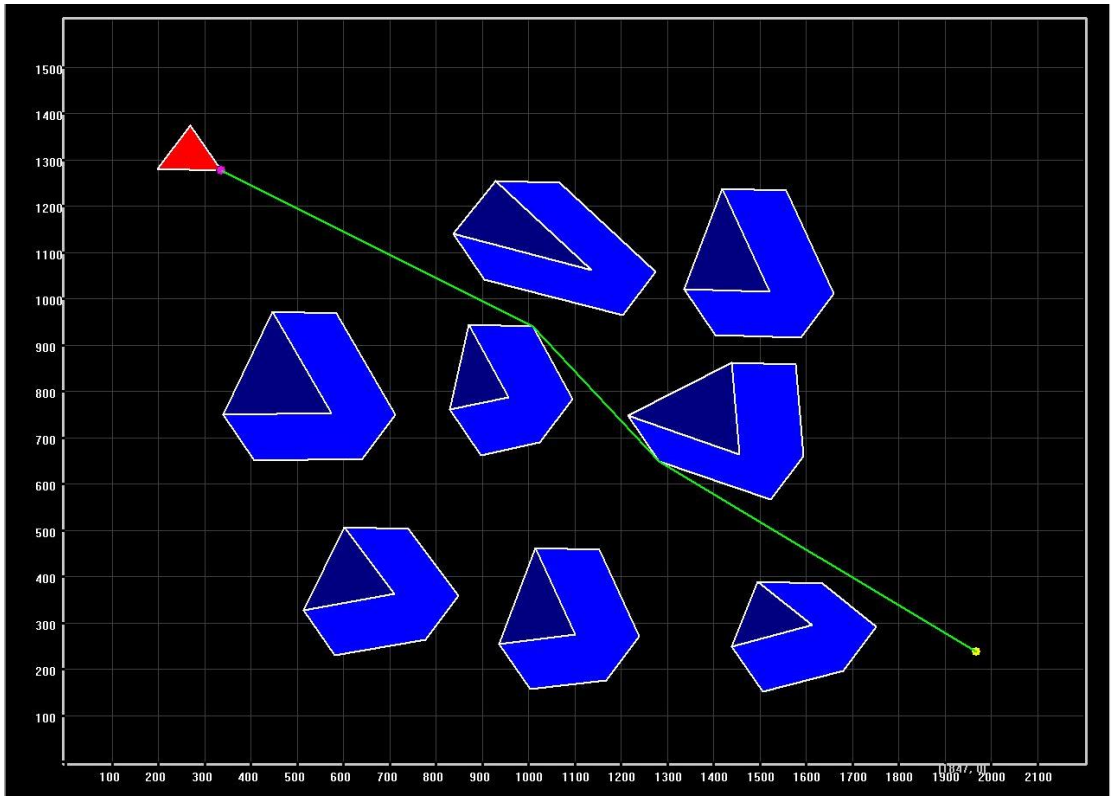
动画效果：

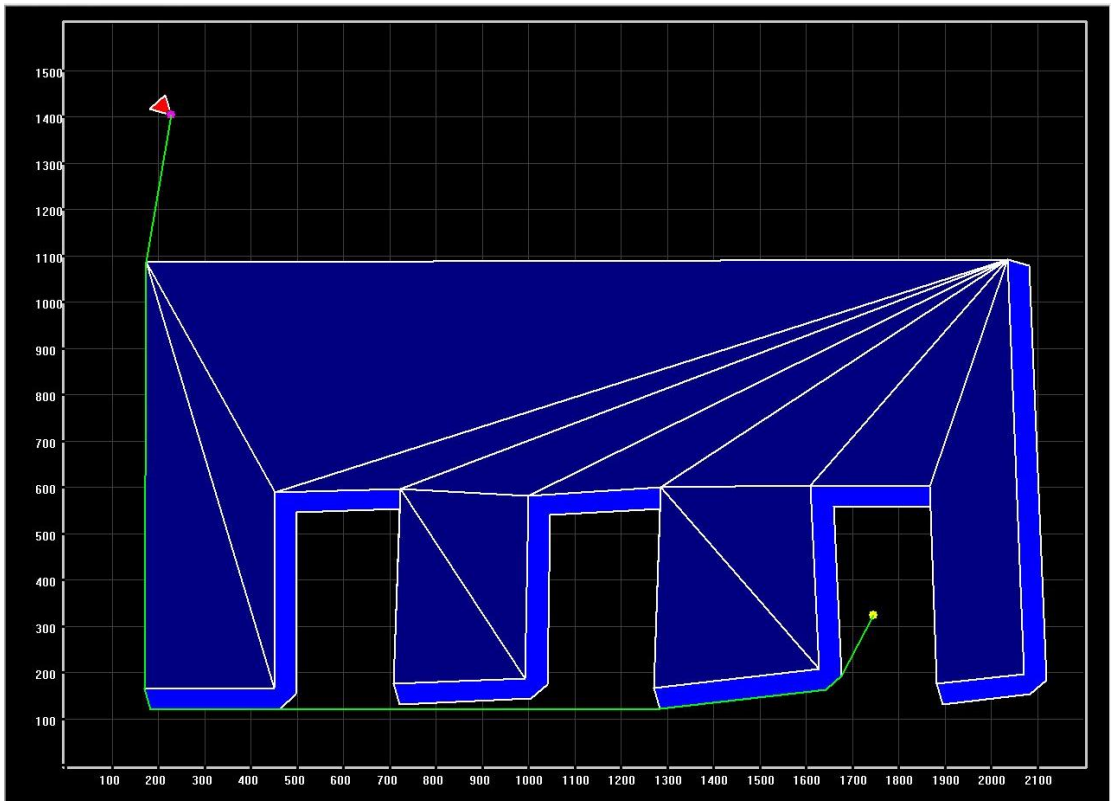


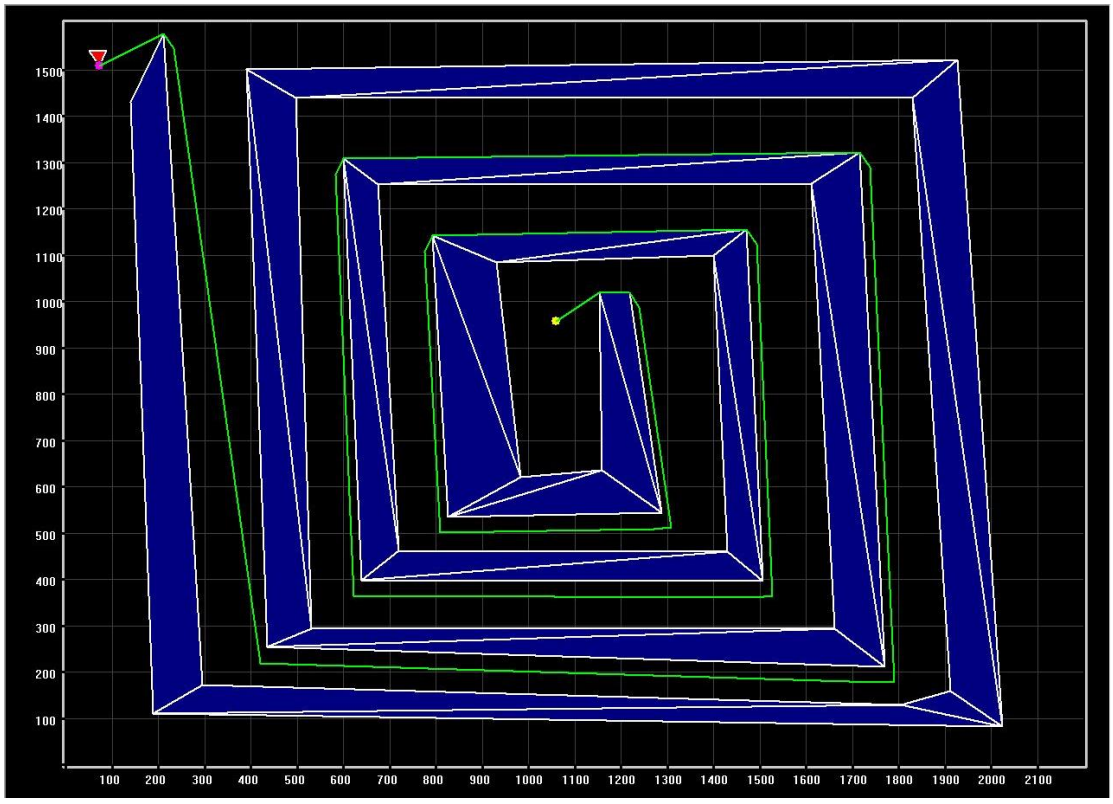
实验结果：

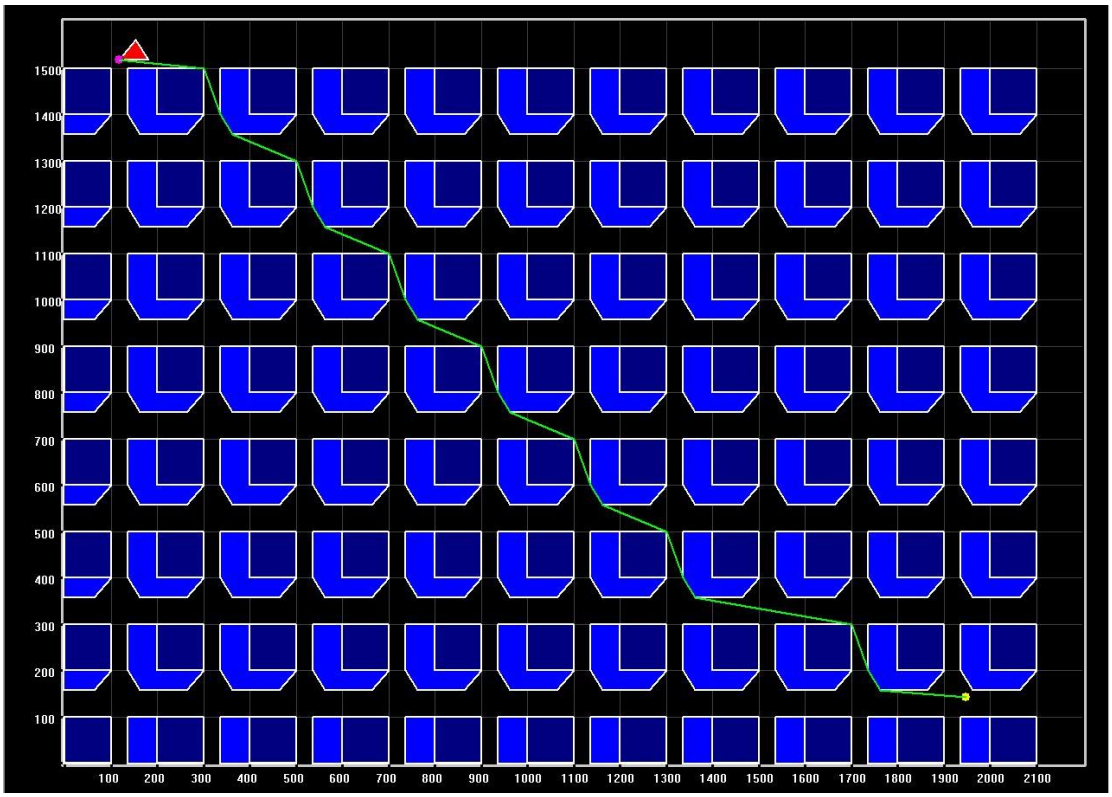
我们通过人工交互编辑和程序自动生成方式生成了各种情况的物体和障碍物，算法可以分布计算算法各个要素生成的过程，也可以动画演示，并显示机器人循径过程。

下面我们展示一些数据算法运行的效果。




1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000 2100


1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000 2100



使用说明：

平台提供了对数据进行编辑和演示的功能，主要的使用功能如下：

数据输入：

平台提供了两种主要的数据生成方式：

文件导入：导入自定义格式的文件，我们提供了十几个各种形状的数据文件，文件中指定了机器人，障碍物的形状以及终点位置。单击【File】菜单【Load File】读入文件。

手工数据：在编辑菜单中可以对机器人、障碍物和结束位置进行编辑。单击【Edit】菜单，选择【Edit Robot】【Edit Obstacles】【Edit End Position】，分别进行编辑。在编辑障碍物时，如果出现边交叉，程序会进行提示。在编辑机器人是，程序会保证得到的是一个凸多边形。

界面显示：

界面提供了丰富的现实和操作功能，单击【view】菜单，菜单下提供了界面元素和数据的显示开关。

在状态栏中，给出了每个菜单项的功能以及当前算法操作的提示。

算法运行：

用户编辑完毕或者导入数据完毕后，单击【Run】菜单下【Finish Edit】，可以进行算法演示。

算法分为五个步骤，需要依次进行。分别对应【Run】菜单下的选项。

- 【Convex Subdivide】:将非凸的简单多边形障碍物分割为凸多边形；
- 【Minkowski Sum】:求障碍物和机器人的 Minkowski 和；
- 【Union】:合并各个凸多边形的 Minkowski 和；
- 【Cut In Window】:将合并后的 Minkowski 和限制在指定的窗格中；
- 【Construct Path】:根据 Minkowski 和构造机器人运动规划的最短路。

当编辑完成后，可以直接单击【Run】菜单【Animation】动画显示算法运行的效果。

评价与总结

通过本次实验我们对于运动规划问题从理论到实际都有了很深入的了解。首先从理论上，我们了解了运动规划问题的理论下界和最新的解法，其中一些方法已经达到了理论上的最优值。但是注意到这些算法的实现难度问题，我们放弃了实现这些非常复杂的算法。因此从总体上讲，我们的算法的复杂度比较高。因此对于输入的规模有所限制。一般说来输入规模应该控制在1000个节点之内。