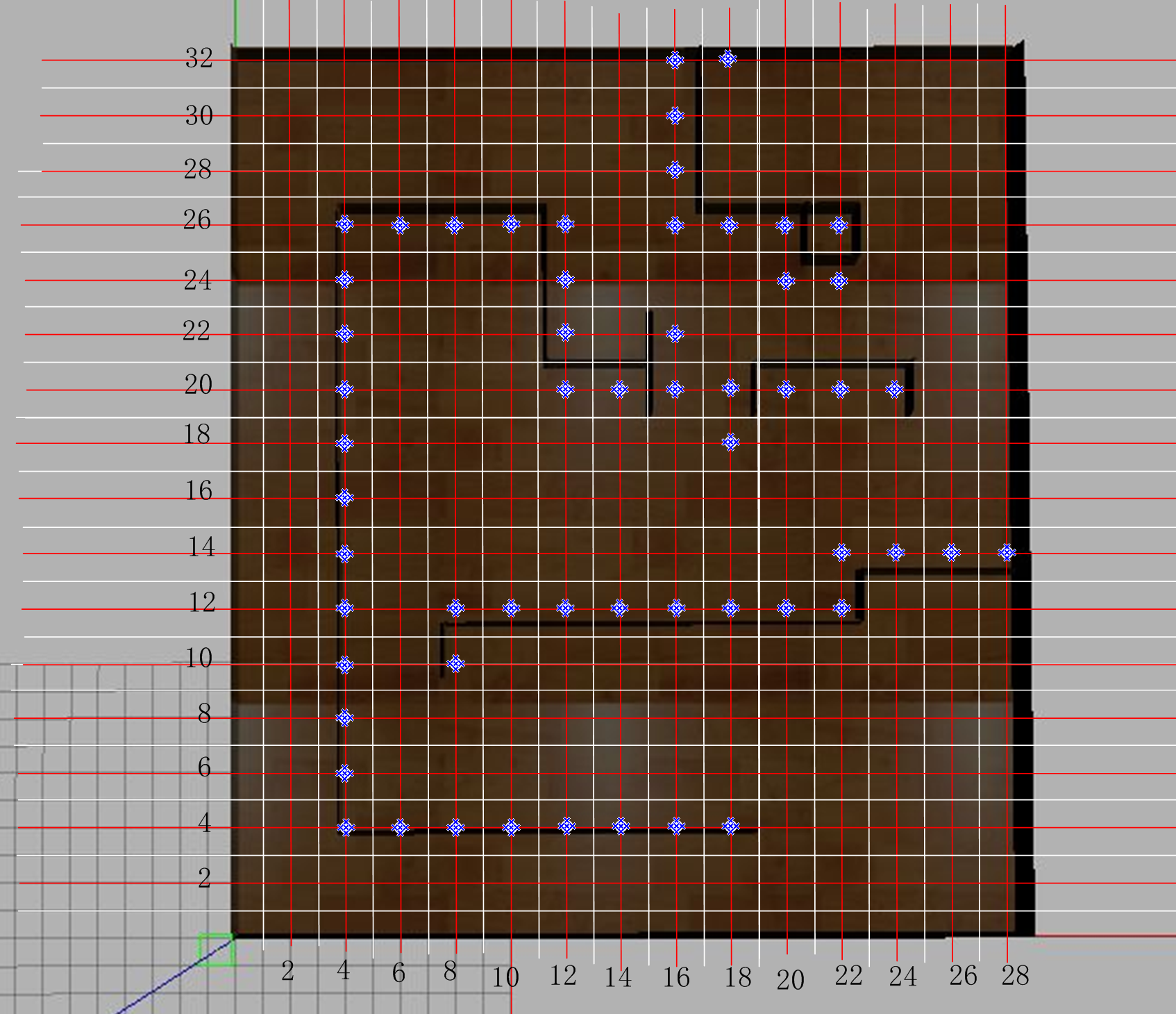
利用网格分解法将图1(a)所示的三维地图分解为图1(b)所示的二维网格地图，按照gazebo中网格大小划分，该网格地图大致为28×32大小。从红线交织出的顶点中去掉墙体占据的坐标点（包括图1(b)中圆点示出的坐标点），选择剩下的坐标点（称为自由坐标点）作为小车的目标位置。需要说明的是，因为gazebo中网格较小，因此红色网格才是算法中使用的最小单位网格，仿真中的坐标除以2为算法中的坐标，仿真使用的坐标与算法中计算的坐标是二倍关系。



(a)三维地图 (b)二维网格地图

图1

算法原理如下：

假设小车载有深度相机（仿真中为激光雷达但不影响算法），该相机的视野范围为正前方8个红色网格，注意是红色网格。如图2所示，三角形表示小车，编号①~⑧的网格为视野范围。



图2

仅使用每个小车在自由坐标点时的位置。每个自由坐标点的相邻自由坐标点为1-4个，根据一系列规则选择一个相邻自由坐标点作为小车的目标位置。对每个小车而言，记录它所经过的自由坐标点，根据它的当前位置和所记录的上一个自由坐标点，可以得到小车的运动朝向，据此可以知道小车附近哪个方向的网格被“看到了”。而知道小车经过的自由坐标点，就可以知道地图中哪些网格逐渐被看到。故，需要获取小车经过自由坐标点的时间点，这也是当时附近那些网格被看到的时间点。网格没被看到的这个状态称为空闲状态，为每个网格赋空闲时间这个变量，表示网格自上次被看到以来所经过的时间，网格一旦被看到其空闲时间置0。仿真开始时每个网格的空闲时间为1，小车看到的网格空闲时间为0，后面小车每到达一个自由坐标点就更新所有网格的空闲时间并获取下一个目标位置，这个空闲时间即需求的数据。

算法伪代码如下：

**算法4.1** 基于感知能力可变的目标分配算法

**输入：**大小为h\_1×v\_1的地图map，大小为*n*×2的机器人初始位置*rob\_pos*，大小为h\_1×v\_1的网格信息素*P*h，大小为h\_1×v\_1的顶点信息素*V*a，总分配次数w

**输出：***rob\_pos*

1: Action ← map中自由顶点的位置

2: *Visited* ← 机器人在初始位置扫描过的网格坐标点集

3: **for** *i* ← 0 to h\_1 – 1 **do**

4: **for** *j* ← 0 to v\_1 - 1 **do**

5: *P*h[*i*][*j*] ← 1

6: *V*a[*i*][*j*] ← -10000

7: **for** *i* ← 1 to h\_1 – 1 **do**

8: **for** *j* ← 1 to v\_1 – 1 **do**

9: **if** [*i*, *j*] in Action **then**

10: *V*a[*i*][*j*] ← [*i*, *j*]周围网格信息素的平均值

11: *Visited*中网格的信息素按照距离机器人的远近衰减 //初始化

12: *tt* ← 0

13: **while**(*tt* != w) **do**

14: 获取机器人的相邻顶点

15: **if** 邻近 **then**

16: 遵循远离对方原则

17: **else** **do**

18: 遵循禁止回头原则

19: **if** (*tt* > 10) and发生自循环 **do**

20: 跳出自循环

21: **if** *tt* > 5 and 发生两点振荡 **do**

22: 遵循两点振荡原则

23: **if** *tt* > 11 and 发生L形振荡 **do**

24: 遵循L形振荡原则 //选出相邻顶点*next\_pos*

25: 机器人移动至*next\_pos*

26: 更新*rob\_pos*

27: 根据感知模型更新*Visited*以及*Visited*中网格的信息素

28: **for** *i* ← 1 to h\_1 – 1 **do**

29: **for** *j* ← 1 to v\_1 – 1 **do**

30: **if** [*i*, *j*] in Action **then**

31: *V*a[*i*][*j*] ← [*i*, *j*]周围网格信息素的平均值

32: *tt* ← *tt* + 1