一、　问题一的模型建立与求解

1.1 图像预处理

1.1.1 岩石轮廓识别

对于附件岩石平面分布图，由于需要设计一条合理的攀爬路线，因此问题转化为图论问题，对此，需要将不同颜色的岩石均转为图论问题中的节点，本文使用计算机视觉开源库 OpenCV 对岩石轮廓进行识别，以下是数字图像处理步骤。

表 1.1 OpenCV 轮廓识别步骤

（1）	使用 imread 读取图像并转为灰度图像
（2）	使用 Canny 边缘检测找到图像中的边缘
（3）	使用 findContours 查找边缘并遍历每个轮廓
（4）	使用 contourArea 计算轮廓面积，忽略面积过小的轮廓
（5）	使用 drawContours 在图像上绘制轮廓并编号

其中，图像灰度化的目的是为了简化矩阵，提高运算速度。彩色图像中的每个像素颜色由 R、G、B 三个分量来决定，而每个分量的取值范围都在 0-255 之间，对计算机而言，彩色图像的一个像素点就会有 256*256*256=16777216 种颜色的变化范围，而灰度图像是 R、G、B 分量相同的一种特殊彩色图像。

对计算机来说，一个像素点的变化范围只有 0-255 这 256 种。彩色图片的信息含量过大，而进行图片识别时，只需要使用灰度图像里的信息就已经足够。图像灰度化处理主要有以下几种方式：

（1）　分量法

将彩色图像中的三分量的亮度作为三个灰度图像的灰度值，可根据应用需要选取一种灰度图像，同时也是 OpenCV 库的默认方法

$$\begin{cases} Gray_1(i,j) = R(i,j) \\ Gray_2(i,j) = G(i,j) \\ Gray_3(i,j) = B(i,j) \end{cases} \tag{1-1}$$

（2）　最大值法

将彩色图像中的三分量亮度的最大值作为灰度图的灰度值

$$Gray(i,j) = \max R(i,j), G(i,j). B(i,j) \tag{1-2}$$

（3）　平均值法

将彩色图像中的三分量亮度求平均得到一个灰度值

$$Gray(i,j) = \frac{[R(i,j) + G(i,j) + B(i,j)]}{3} \tag{1-3}$$

（4）　加权平均法

由于人眼对绿色的敏感最高，对蓝色敏感最低，因此，按下式对 RGB 三分量进行加权平均能得到较合理的灰度图像

$$Gray(i,j) = 0.299 * R(i,j) + 0.578 * G(i,j) + 0.114 * B(i,j) \tag{1-4}$$

本文使用（1）分量法，将彩色图像中的三分量的亮度作为三个灰度图像的灰度值之一，得到下图 1.1 岩石平面分布图的灰度图。

图 1.1 岩石平面分布图的灰度图

得到灰度图后，需要检测灰度图的边缘，使用 Canny 边缘检测算法，它具有低错误率，检测出的边缘是真正的边缘；良好的定位，检测出的边缘像素点与真正边缘的像素点距离近；对噪声不敏感，噪声不应该标注为边缘。Canny 边缘检测算法有四个步骤：

（1） 降低对噪声的影响，对图像做高斯滤波或中值滤波，过滤噪声。

（2） 使用 Sobel 算子对图像的像素点求梯度大小和方向，以下为 Sobel 算子。

$$dx=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix}$$

$$dy=\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix}$$

（3） 使用非极大值抑制算法在一组边缘中选取最好的边缘，具体做法是检查每个像素点与附近梯度方向一致的像素点，当前像素点梯度最大，则保留，否则去除。

（4） 使用双阈值（小阈值, 大阈值）确定最终的边缘，像素点梯度高于大的阈值，则保留；像素点低于小的阈值，则忽略；介于两个阈值之间，判断像素点与边缘像素点是否相连。

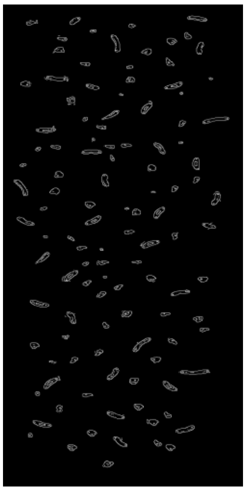

图 1.2 Canny 检测岩石轮廓图

1.1.2 岩石质心表示

由于岩石轮廓是一个形状不规则的几何体，为简化模型，避免过多的几何讨论，将每个轮廓均用其质心表示，将其当作物理学中的质点，图论中的节点，使得将问题专注于攀爬路线的设计，而非考虑岩石形状的力学分析。

几何体的质心是形状中所有点的算术平均值，假设一个形状由以下部分组成n个不同点$x_1, x_2 \dots x_n$，则质心由下式给出：

$$c = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1-5}$$

其中c是几何体的质心，x_i是点在空间中的坐标，在图像处理和计算机视觉的背景下，每个几何体都是由像素组成的，质心只是构成形状的所有像素的加权平均值。

在 OpenCV 中，进行图像操作，使用图像矩找到 blob(机器视觉中指图像中具有相似颜色，纹理等特征所组成的一块连通区域)的中心。图像矩是图像像素值的加权平均值，从而找到图像的一些特定属性，如半径，面积，质心等。为了找到图像的质心，将其二值化然后找到它的质心。质心由下式给出： -

$$C_x = \frac{M_{10}}{M_{00}} \tag{1-6}$$

$$C_y = \frac{M_{01}}{M_{00}} \tag{1-7}$$

其中C_x是质心的x坐标，C_y是质心的y坐标，M表示图像几何矩，$P(x,y)$表示图像上坐标为(x,y)上的灰度值，几何矩计算由下式给出：

$$M_{ji} = \sum_{x,y}\left(P(x,y)\cdot x^j \cdot y^i\right) \tag{1-8}$$

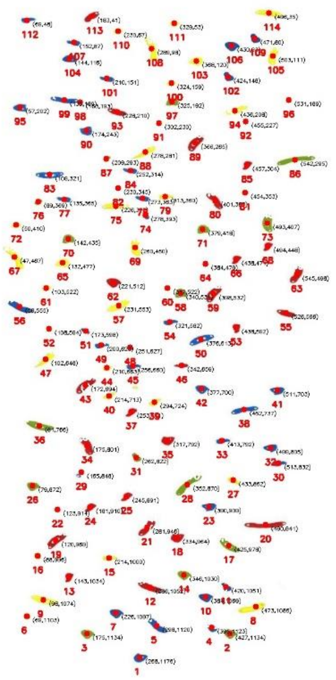

图 1.3 轮廓的质心坐标图及编号

1.1.3 轮廓识别优化算法

当一些岩石轮廓未被正确标出时，可能是由于边缘检测参数的设置不合适，或者轮廓的形状比较复杂，不易被简单的边缘检测方法捕捉到。为了更好地标出岩石轮廓，可采用以下优化步骤：

（1） 调整 Canny 边缘检测参数：Canny 边缘检测的两个阈值参数可以影响边缘检测的结果。尝试调整这两个阈值的值，以获得更好的边缘图像。

（2） 使用更高级的轮廓近似方法： 轮廓近似方法可以帮助更好地捕捉轮廓的形状，特别是当轮廓较复杂时。可以尝试 approxPolyDP 来近似轮廓。

（3） 使用颜色分割： 如果岩石的颜色非常明显，可以使用颜色分割方法，例如阈值化，以便更好地分离岩石和背景。这需要对颜色空间进行一些实验，以找到最适合的颜色通道和阈值。

（4） 形态学操作： 在边缘检测后，使用形态学操作（腐蚀和膨胀）可以消除一些小的孔洞或噪声，使得轮廓更连续。

（5） 调整轮廓面积过滤阈值： 可能需要调整轮廓面积的过滤阈值，以保留更多或更少的轮廓。

1.1.4 节点连线

通过查阅相关资料，攀爬时每次向上的高度最好不超过脚底到膝盖的距离，则会难以攀爬并保持平衡，由于图中整个场地高 10m，经过按比例缩放，分别确定了以100,90,80,70为每次上升的界限，即当节点之间的欧几里得距离只有小于该界限，才进行连边，也就是物理上的可直接从该点通过这条边跨越到另外一个点

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (1-9)$$

其中假设两点分别为$point_1, point_2$，其坐标分别为$(x_1, y_1), (x_2, y_2)$，那么其距离可由$(1-9)$式计算。

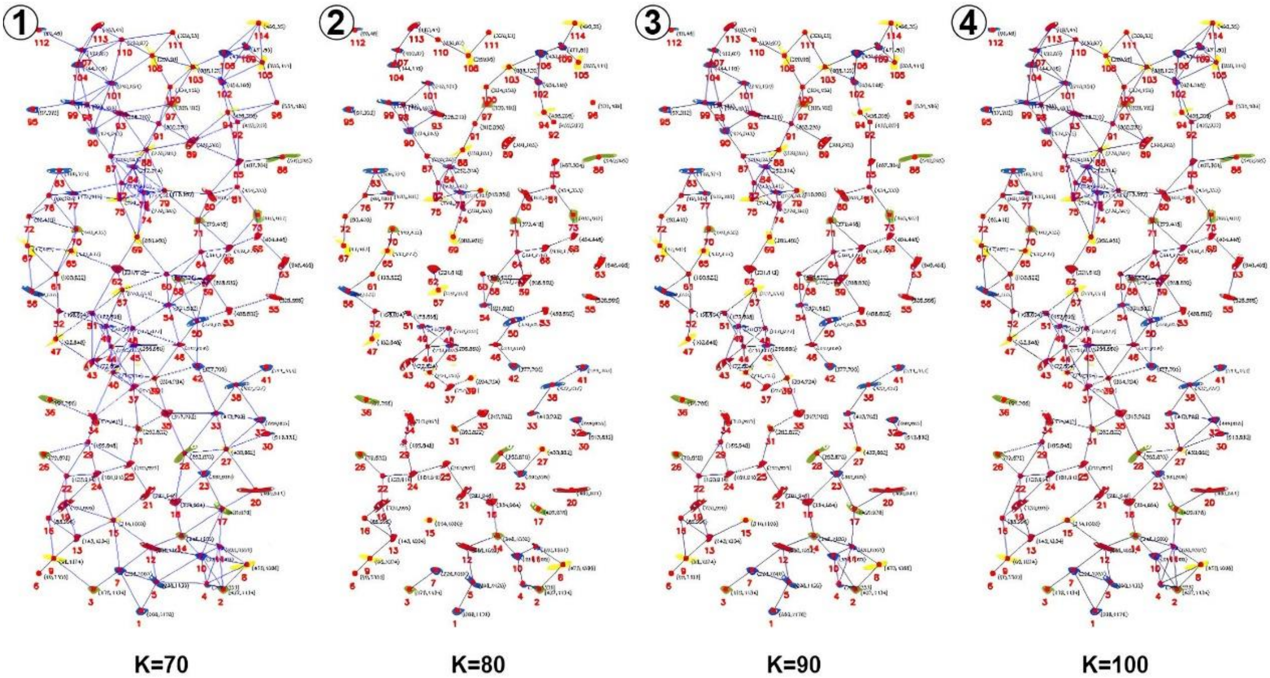

图 1.4 每次最大攀爬高度不同时的节点网络图

1.2 图论模型求解

1.2.1 设定权值

在图论问题中，需要定量地去衡量从出点到入点的价值，这条边的值称为权值，本题中，由于只需要计算一条合理的攀爬路线，故选择路径长度最小的路线，更能节省攀爬者的体力，因此，将权值设置为两点之间的路径长度，可通过式$(1-9)$进行计算。

1.2.2 设置源点终点

从图 1.4 中可以看出，编号为 1 的点为最低点，本文中选取 1 号点作为攀爬者的源点，而处于顶端的位置的共有 4 个点，分别为 111，112，113，114 号点，故分别选取这 4 个点作为终点，由于攀爬者的体型并未规定，故本文分别选取 70，80，90，100 作为攀爬者每次上升的最大高度，并以此为参数建立对应的网络图。

1.2.3 最短路模型求解

本文采用 Dijkstra 算法求解该模型，Dijkstra 算法基于贪心的思想，通过保留目前为止所找到的每个顶点$v \in V$从s到t的最短路径来运行，初始时，原点s的路径权重被赋为 0（即原点到原点的距离为 0），同时把所有其他顶点的路径长度设为无穷大，即表示不知道任何通向这些顶点的路径，当算法结束$dist[v]$中存储的便是从s到t的最短路径，如果路径不存在，则为$dist[v] = inf$。

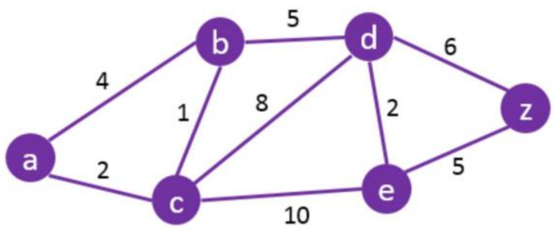

图 1.5 Dijkstra 算法示意图

松弛操作是 Dijkstra 算法的基础操作，如果存在一条从u到v的边，那么从s到v的一条新路径时将边$weight(u,v) \in E$添加到从s到u的路径尾部来拓展一条从s到v的路径，这条路径的长度是$dist[u] + weight(u,v)$，如果这个值比目前已知的$d[v]$的值都要小，那么可以用这个值来替代当前$dist[v]$的值，松弛边的操作一直执行到所有的$dist[v]$都代表从s到v的最短路径的长度值。

其伪代码为：

Function Dijkstra(G, w, s)

INITIALIZE-SINGLE-SOURCE(G, s)//将原点以外的顶点的$dist[v]$置为无穷大

$dist[s] = 0$//将原点到原点的距离设置为 0

$S \leftarrow queue$//Q是顶点V的一个优先队列

$Q \leftarrow s$//以顶点的最短路径估计排序

While($Q \in queue$)

  do $u \leftarrow EXTRACT - MIN(Q)$//选取$u$为$Q$中最短路径估计最小顶点

  $S \leftarrow S \cup u$

  for each vertex $v \in Adj[u]$

    do RELAX(u, v, w)//松弛成功的节点会被加入到队列中

1.2.4 时间复杂度分析与优化

将图中边数用$|E|$表示，顶点数用$|V|$表示，对于任何基于顶点集Q的实现，算法的运行时间是$O\left(|E| \cdot dk_Q + |V| \cdot e_{m_Q}\right)$，其中$dk_Q$和$em_Q$分别表示完成键的降序排列时间和从$Q$中提取最小键值的时间。

对于没有任何优化的戴克斯特拉算法，实际上等价于每次遍历了整个图的所有结点来找到Q中满足条件的元素（即寻找最小的顶点是$O(|V|)$的，此外实际上还需要遍历所有的边一遍，因此算法复杂度为$O(|V|^2 + |E|)$

此外，对于边数$|E|$，如果少于$|V|^2$，则称该图为稀疏图，那么可用邻接表对图进行存储，不仅节省空间，而且能更快地访问节点元素；反之则称为邻接矩阵。

对图 1.4 分别统计，可得到下表，由表中，可判断该岩石分布图为稀疏图，故使用邻接表来提高节点访问效率并节省内存。

表 1.2 不同攀爬最大高度时的点边数统计结果

每次攀爬的最大高度	顶点数	边数
70	114	113
80	114	160
90	114	225
100	114	280

1.2.5 模型求解

综上分析，首先根据 OpenCV 识别出岩石轮廓的中心点集合，并根据每次向上攀爬的最大高度，划分出 4 种不同的网络图，通过权值设置得到了边的集合，而后用邻接表建图，并分析了起点终点归属，再代入 Dijkstra 算法进行求解，最终得了$dist$数组，即起点s距离t的距离为$dist[t]$，得到以下表格。

表 1.3 问题一结果图

每次攀爬最大高度/编号	111	112	113	114
70	inf	inf	inf	inf
80	1337.43	inf	1351.51	1427.8
90	1263.63	inf	1274.58	1327.61
100	1205.84	1254.05	1216.8	1277.48

可以发现，当攀爬高度为 70 时，均无法到达终点，并且选择攀爬终点为 111 号点的距离始终小于其他点，故这里选择以 111 号点为终点，每次攀爬最大高度设为 100，则合理的攀爬路线为：

$$1 \to 7 \to 15 \to 24 \to 29 \to 34 \to 40 \to 49 \to 57 \to 62 \to 69 \to 78 \to 88 \to 91 \to 100 \to 108 \to 111$$

其完整路线图在岩石轮廓图如下图表示

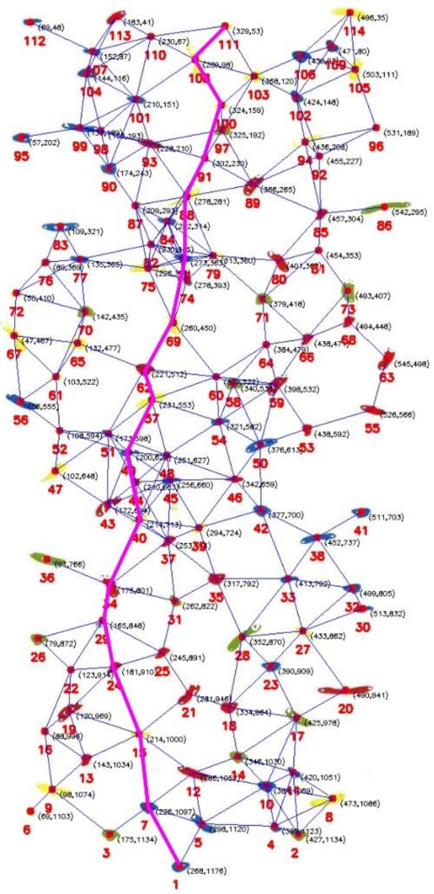

图 1.6 当每次跨越高度最大为 100 的最短路径(粉色)