Chapter 1

万花尺

1.1 参数方程

1.1.1 万花尺方程

Figure 1.1a 展示了类似万花尺运动的数学模型。

在 Figure 1.1a 中，C 是较小的圆的圆心，P 是笔尖。较大的圆半径为 R，较小的圆半径为 r。半径之比表示如下：

$$k = \frac{r}{R}$$

将线段 PC 与小圆半径 r 之比作为变量 l（$l = PC/r$），它决定了笔尖离小圆圆心有多远。然后，组合这些变量来表示 P 的运动，得到如下的参数方程：

$$\begin{aligned} x &= R\left((1-k)\cos(\theta) + lk\cos\left(\frac{1-k}{k}\theta\right)\right) \\ y &= R\left((1-k)\sin(\theta) + lk\sin\left(\frac{1-k}{k}\theta\right)\right) \end{aligned} \tag{1.1}$$

将曲线绘制为一系列点之间的线段。如果这些点足够接近，图看起来就像平滑的曲线。

要确定何时停止绘图，就要利用万花尺的周期性（即万花尺图案多久开始重复），研究内外圆的半径之比：

$$\frac{r}{R}$$

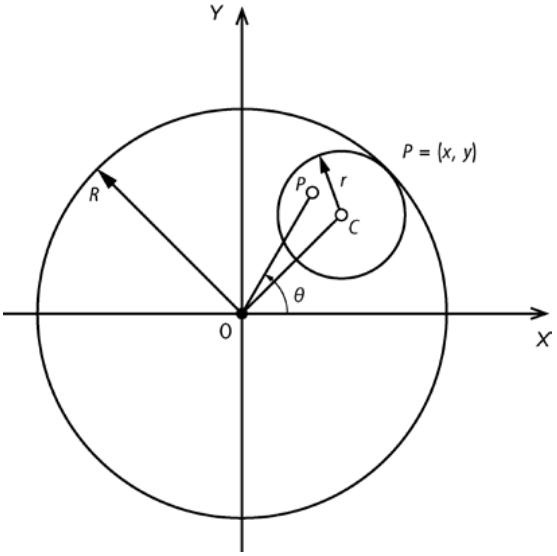
分子分母除以它们的最大公约数（GCD），化简该分数，分子就告诉我们需要多少圈才能完成曲线。例如，在 Figure 1.1b 中，(r,R) 的 GCD 是 5。

$$\frac{r}{R} = \frac{65}{220}$$

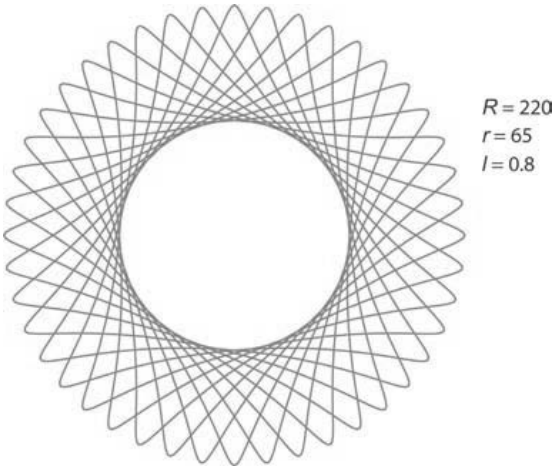
下面是该分数化简后的形式：

$$\frac{(65/5)}{(220/5)} = \frac{13}{44}$$

这告诉我们，13 圈后，曲线将开始重复。44 告诉我们小圆围绕其中心旋转的圈数，它提示了曲线的形状。在 Figure 1.1b 中数一下，会看到图形中花瓣或叶的数目恰好是 44!



(a) 万花尺数学模型



(b) 示例曲线，$R = 220$, $r = 65$, $l = 0.8$

Figure 1.1: 万花尺和某个示例

一旦用简化形式表示了半径比 r/R，画出螺线的参数 θ 范围就是 $[0, 2\pi r]$。这告诉我们何时停止绘制特定的螺线。不知道该角度的结束范围，就会循环不止，不必要地重复该曲线。

Chapter 2

ASCII 文本图形

2.1 工作原理

该项目利用了这样一个事实：从远处看，我们将灰度图像看成是它们亮度的平均值。

ASCII 文本图形的生成方法是，将图像分割成小块，并用 ASCII 字符替换一小块的平均 RGB 值。从远处看，因为眼睛的分辨率有限，我们大致会丢失细节，看到 ASCII 文本图形中的“平均”值，否则文本图形看起来就不那么真实。

该程序将给定的图像先转换为 8 位的灰度，让每个像素有一个灰度值，范围在 $[0,255]$（8 位整数的范围）。将这个 8 位值看成是亮度，0 表示黑色，255 表示白色，中间值是不同程度的灰色。

接着，将该图像分割成 $M \times N$ 个小块构成的网格（其中，M 和 N 是 ASCII 文本图形中的行和列编号）。然后程序计算网格中每个小块的平均亮度值，通过预定义的一些有梯度的 ASCII 字符（一组不断增加的值）来表示 $[0,255]$ 范围的灰度值，与适当的 ASCII 字符匹配。它将用这些值作为亮度值的查找表。

完成的 ASCII 文本图形只是一些文本行。要显示文本，就要用到 Courier 这样的等宽字体，因为如果每个文本字符宽度不相同，图像中字符将无法正确地按网格排列，会得到间隔不均和失真的输出。

所用字体的“横纵比”（宽度与高度之比）也会影响最终图像。如果一个字符所占空间的横纵比与该字符取代的图像小块的横纵比不同，则最终的 ASCII 字符图形会出现失真。实际上，你试图用一个 ASCII 字符来替换图像小块，所以它们的形状要匹配。例如，如果将图像分割成正方形小块，然后用一种高度拉伸的字体替换每个小块，最终的结果将出现垂直拉伸。

为了解决这个问题，需要缩放网格中的行数，以匹配 Courier 的长宽比。（可以向程序发送命令行参数，修改缩放，以匹配其他字体。）

总之，下面是程序生成 ASCII 文本图形的步骤：

1. 将输入图像转成灰度；
2. 将图像分成 $M \times N$ 个小块；
3. 修正 M（行数），以匹配图像和字体的横纵比；
4. 计算每个小块图像的平均亮度，然后为每个小块查找合适的 ASCII 字符；

5. 汇集各行 ASCII 字符串，将它们打印到文件，形成最终图像。