

基于 C 语言的样条曲线拟合及 Yin 集布尔代数实现

徐圣泽

2021 年 9 月 7 日

1 前言

本篇文章基于样条曲线和 Yin 集的理论，通过编写软件包实现了曲线的样条拟合、Yin 集及其布尔代数的有效表示。我认为此次大作业的重点在于将两部分的理论联系在一起写出有实际数学和物理意义的程序，事实证明每个部分的程序都是彼此关联着的，且都具有一定的实际价值。

我在题中要求的三个编程部分给出了相应的算法，并且在本文展现了算法的思路、测试过程以及最终结果，最终利用算法写出程序，实现结果的呈现。

第一部分样条曲线拟合的部分，我基于《计算方法引论》一书中的数学理论编写程序，并且实现了偏序关系的判断。对于第二部分布尔代数的实现，我采用了一种比较朴素的呈现方法，但严格来讲并未实现对 Yin 集合这一类的封闭运算，详细的实现过程会在后文进行阐述。

2 卡通图案的表示

2.1 选定黑白卡通图

根据题目要求，选取了一个至少包含五个层次的黑白卡通图案。



图 1: 黑白卡通图案

2.2 选取特征点

选取的黑白卡通图案共有 13 条边界曲线, 利用 MATLAB 中 `grabit` 功能, 逐条选取特征点, 存入文本文件中。

2.3 拟合样条曲线

2.3.1 数学理论

设在区间 $[a, b]$ 上取 $n + 1$ 个节点, 给定节点上的函数值 $f(x_i) = y_i$, 现在构造三次样条插值函数 $s(x)$ 满足下列条件: (1) $s(x_i) = y_i$. (2) 在每个小区间 $[x_i, x_{i+1}]$ 上是一个不高于三次的多项式. (3) $s(x) \in C^2_{[a,b]}$. 假设在区间 $[a, b]$ 上三次样条插值函数 $s(x)$ 存在, 并用 m_i 来表示 $s(x)$ 在点 x_i 处的微商值, 由于曲线通过点 (x_i, y_i) , 并且在每一个小区间 $[x_i, x_{i+1}]$ 上满足条件:

$$\begin{aligned} s(x_i) &= y_i, s(x_{i+1}) = y_{i+1} \\ s'(x_i) &= m_i, s'(x_{i+1}) = m_{i+1} \end{aligned}$$

故根据 *Hermite* 插值公式写出小区间 $[x_i, x_{i+1}]$ 上的三次样条插值函数 $s(x)$ 的计算公式:

$$\begin{aligned} s(x) &= (1 + 2 \frac{x - x_i}{x_{i+1} - x_i}) (\frac{x - x_{i+1}}{x_i - x_{i+1}})^2 y_i + (1 + 2 \frac{x - x_{i+1}}{x_i - x_{i+1}}) (\frac{x - x_i}{x_{i+1} - x_i})^2 y_{i+1} \\ &\quad + (x - x_i) (\frac{x - x_{i+1}}{x_i - x_{i+1}})^2 m_i + (x - x_{i+1}) (\frac{x - x_i}{x_{i+1} - x_i})^2 m_{i+1} \end{aligned}$$

需要设法求出节点 x_i 处的微商值 m_i , 利用函数 $s(x)$ 在节点 x_i 上二阶微商连续的性质, 对 x 求微商并令 $h_i = x_{i+1} - x_i$, 不难得到:

$$\begin{aligned} s''(x) &= (\frac{6}{h_i^2} - \frac{12}{h_i^3}(x_{i+1} - x))y_i + (\frac{6}{h_i^2} - \frac{12}{h_i^3}(x - x_i))y_{i+1} + \\ &\quad (\frac{2}{h_i} - \frac{6}{h_i^2}(x_{i+1} - x))m_i - (\frac{2}{h_i} - \frac{6}{h_i^2}(x - x_i))m_{i+1} \end{aligned}$$

因此可以得到区间 $[x_i, x_{i+1}]$ 上点 x_i 的右微商和左微商:

$$\begin{aligned} s''(x_i^+) &= -\frac{6}{h_i^2}y_i + \frac{6}{h_i^2}y_{i+1} - \frac{4}{h_i}m_i - \frac{2}{h_i}m_{i+1} \\ s''(x_i^-) &= \frac{6}{h_{i-1}^2}y_{i-1} - \frac{6}{h_{i-1}^2}y_i + \frac{2}{h_{i-1}}m_{i-1} + \frac{4}{h_{i-1}}m_i \end{aligned}$$

利用左微商等于右微商, 可整理得到:

$$\begin{cases} \alpha_i = \frac{h_{i-1}}{h_{i-1} + h_i} \\ \beta_i = 3(\frac{1 - \alpha_i}{h_{i-1}}(y_i - y_{i-1}) + \frac{\alpha_i}{h_i}(y_{i+1} - y_i)) \end{cases}$$

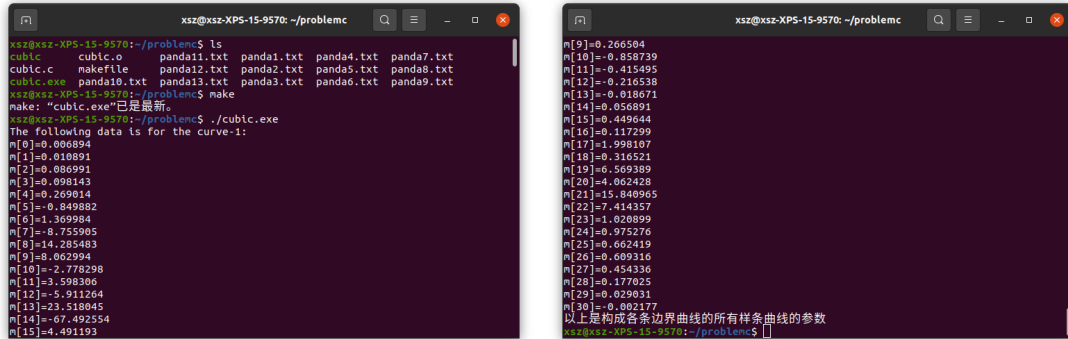
同时得到方程组:

$$(1 - \alpha_i)m_{i-1} + 2m_i + \alpha_i m_{i+1} = \beta_i$$

这是关于 $n + 1$ 个未知量的 $n - 1$ 个线性方程组, 因此为了得到特定的一个解, 还需要边界条件。下面每个算法都提供了一种边界条件的情形, 补充了两个附加条件。

2.3.2 程序及结果

本程序定义了两个函数，分别用于处理边界条件为端点处一阶导已知和二阶导已知的情形，读取存放于文本文件的特征点坐标后，分别计算得到每条边界曲线每一小段样条曲线的参数，此处的参数即上面提到的未知数微商 $m[i]$ 。



```
xsz@xsz-XPS-15-9570: ~/problemc
xsz@xsz-XPS-15-9570:~/problemc$ ls
cubic.o  panda11.txt  panda1.txt  panda4.txt  panda7.txt
cubic.c  makefile    panda12.txt  panda2.txt  panda8.txt
cubic.exe panda10.txt  panda13.txt  panda3.txt  panda9.txt
xsz@xsz-XPS-15-9570:~/problemc$ make
make: "cubic.exe"已是最新。
xsz@xsz-XPS-15-9570:~/problemc$ ./cubic.exe
The following data is for the curve-1:
m[0]=0.006894
m[1]=0.010891
m[2]=0.086991
m[3]=0.098143
m[4]=0.269014
m[5]=-0.849882
m[6]=1.369984
m[7]=-8.755905
m[8]=-14.285403
m[9]=0.062994
m[10]=-2.778298
m[11]=3.598306
m[12]=-5.911264
m[13]=-23.518045
m[14]=-67.492554
m[15]=4.491193

xsz@xsz-XPS-15-9570:~/problemc
m[9]=0.266504
m[10]=-0.858739
m[11]=-0.415495
m[12]=-0.216538
m[13]=-0.018671
m[14]=0.056891
m[15]=0.449644
m[16]=0.117299
m[17]=1.998107
m[18]=0.316521
m[19]=0.569389
m[20]=4.062428
m[21]=15.048965
m[22]=7.414357
m[23]=1.020899
m[24]=0.975276
m[25]=0.662419
m[26]=0.609316
m[27]=0.454336
m[28]=0.177625
m[29]=0.029031
m[30]=-0.002177
以上是构成各边界曲线的所有样条曲线的参数
xsz@xsz-XPS-15-9570:~/problemc$
```

(a) 1

(b) 2

图 2: 程序运行结果

需要说明的是，上述结果采取的是自然边界条件，且首尾点坐标一致（即开始点也作为结束点，以此实现闭曲线）。利用 MATLAB 绘制相应曲线，并利用自带 *spline* 和 *csape* 函数检验程序输出参数的正确性（通过检验一些坐标根据程序输出表达式对应的函数值是否符合 MATLAB 的计算结果，事实证明程序是正确的）。

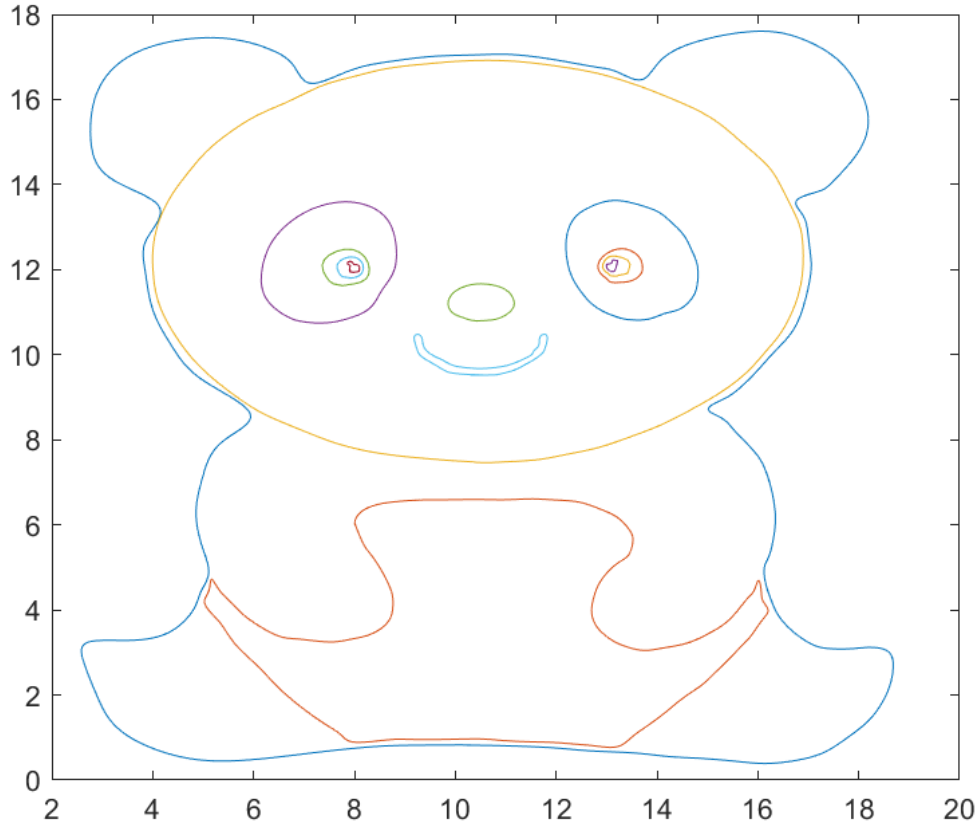


图 3: 样条拟合后的曲线

2.4 判断偏序关系

2.4.1 模型简化

在本部分我们需要判断几乎不相交的闭样条的偏序关系。根据相关理论，我们可以将问题化归为判断各闭合曲线之间的包含关系。

首先，由题意知任意两条闭曲线是不相交的，因此只有包含、被包含、非包含三种关系。为判断两条曲线是否具有包含关系，只要判断一条曲线上的某点是否落在另一条曲线上或曲线包围成的区域内。

在本题的求解过程中，我们可以简化问题模型，将所有闭曲线包围成的区域视作由特征点连线成的线性多边形。我们有两个原因足够说明此操作的合理性：1、在第二步我们为足够接近原图的样条曲线取了足够多的特征点；2、即使不视作线性多边形，我们也需要设置步长（后文会详细解释算法），很难做到逐点遍历。

2.4.2 射线法

假设此时已经有 γ_1 和 γ_2 两条边界。我们只需要检验两条曲线上的所有特征点是否同时满足上述条件。

首先设计了第一种算法，分别遍历得到两条曲线横纵坐标的最大值和最小值，如果一条曲线的横、纵坐标的最大值比另一条曲线的横、纵坐标的最大值大，而最小值小，同时满足这四个条件，则此曲线必然包含于另一条曲线。

此种算法已经能检验大多数情况，如下图左侧情形，但仍有少数情况无法被此算法正确判断，如右侧情形。

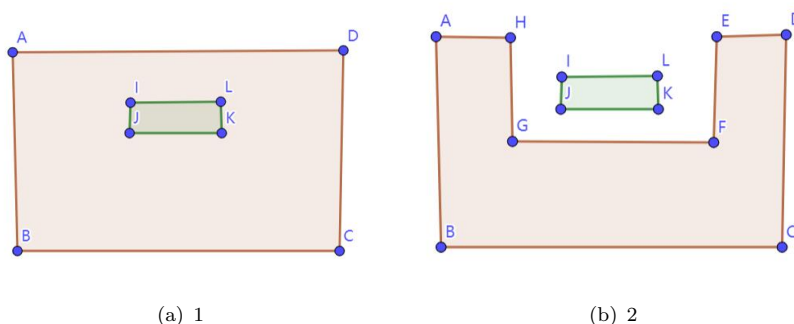


图 4: 两种相似情形

为此我们必须设计出一种更合理的算法，能够适用于任何情况，下面我们采用射线法。考虑以下情形，已给定一闭合曲线，此时取定任意一个点，向任意方向作一射线，若此射线与曲线交点数为奇，则该点在区域内部，若交点数为偶，在该点在区域外部。下图中 O 和 J 对应上述两种情形。

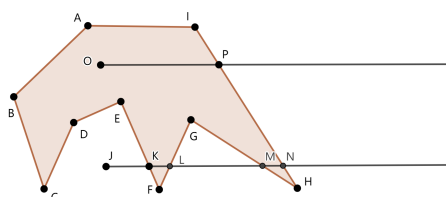


图 5: 射线法的两种情形

2.4.3 程序及结果

原卡通图案的边界若尔当曲线标注如下。

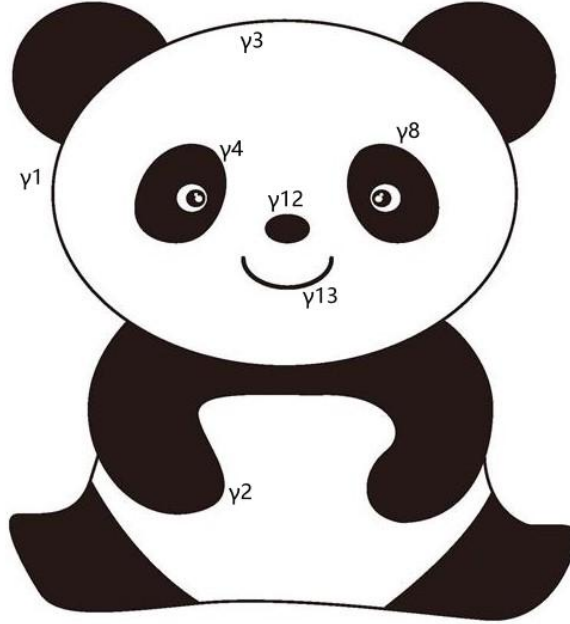


图 6: 卡通图案的若尔当曲线

我们的程序输出的是一个矩阵，矩阵第 i 行第 j 列 ($i \neq j$) 的元素代表第 i 条曲线和第 j 条曲线的包含关系，1 代表前者包含后者 (\succeq)，-1 代表被包含 (\preceq)，0 代表无包含关系。程序的运行结果如下：

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & 0 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

举个例子， $a[1][3] = 1$ ，表示 $\gamma_1 \succeq \gamma_3$ ， $a[7][3] = -1$ ，表示 $\gamma_7 \preceq \gamma_3$ 。

2.5 哈斯图

根据论文中的思路，我们给出本文中卡通图案对应的哈斯图。

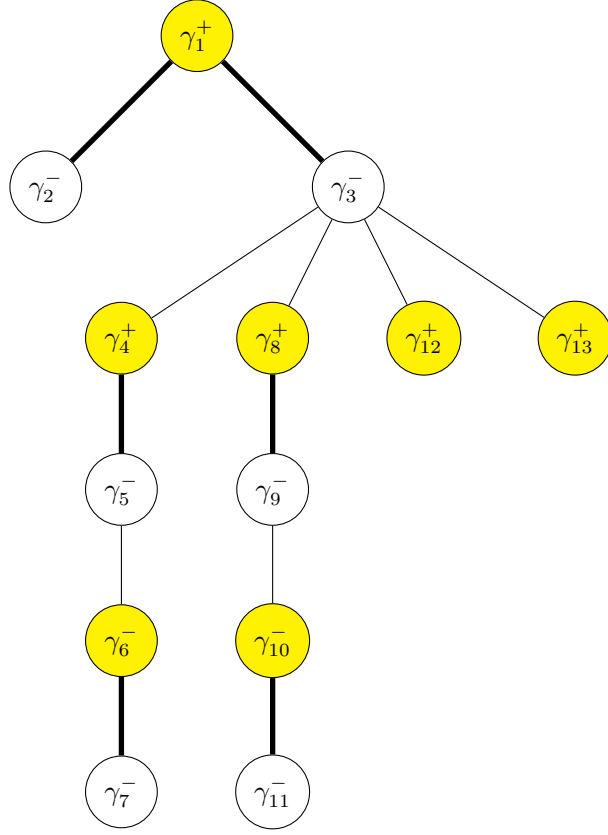


图 7: Hasse diagram

2.6 Spadjor 表示

根据论文中的思路，我们给出本文中卡通图案对应的 Spadjor 表示法。

$$\mathcal{J} = \{\gamma_1^+, \gamma_2^-, \gamma_3^-, \gamma_4^+, \gamma_8^+, \gamma_{12}^+, \gamma_{13}^+, \gamma_5^-, \gamma_6^+, \gamma_7^-, \gamma_9^-, \gamma_{10}^+, \gamma_{11}^-\}$$

$$\mathcal{J} = \cup_{k=1}^7 \mathcal{J}_k$$

$$\mathcal{J}_1 = \{\gamma_1, \gamma_2, \gamma_3\}, \mathcal{J}_2 = \{\gamma_4, \gamma_5\}, \mathcal{J}_3 = \{\gamma_6, \gamma_7\}, \mathcal{J}_4 = \{\gamma_8, \gamma_9\}, \mathcal{J}_5 = \{\gamma_{10}, \gamma_{11}\}, \mathcal{J}_6 = \{\gamma_{12}\}, \mathcal{J}_7 = \{\gamma_{13}\}.$$

$$\mathcal{Y} = \cup_{k=1}^7 \mathcal{Y}_k = \rho(\mathcal{J}), \mathcal{Y}_k = \rho(\mathcal{J}_k)$$

$$\mathcal{Y} = \cup_{k=1}^7 \mathcal{Y}_k = [\cap_{i=1}^3 int(\gamma_i)] \cup [\cap_{i=4}^5 int(\gamma_i)] \cup [\cap_{i=6}^7 int(\gamma_i)] \cup [\cap_{i=8}^9 int(\gamma_i)] \cup [\cap_{i=10}^{11} int(\gamma_i)] \cup int(\gamma_{12}) \cup int(\gamma_{13})$$

3 Yin 集合上的布尔代数

3.1 一种朴素的实现方法

3.1.1 多边形填充——扫描线算法

此部分基于Visual Studio 2019平台，使用的图形库是EasyX，主要用到的算法是扫描线算法，用于多边形填充。一般来讲，本算法实现过程中会用到两个数据结构：边表、活动边表。给定一扫描线，其对应的 ET 表，即边表，主要由四部分构成：某边的纵坐标最大值、最小值，最小值对应的横坐标、某边的斜率。

在建立 ET 表后，开始扫描转换。不同的扫描线与之相交的边线也不同，当扫描线进行转换时，我们需要考虑其对应的相交的边线，由此我们需要建立一个活动边表 AET 。

在对图案进行遍历扫描时，我们对每个 y 值都设置了一个数组，用以存放该扫描线与边线相交的横坐标值，并在得到所有横坐标值后对其进行排列。

进行上述操作后我们对每一个 y 都得到了存放交点坐标值的数组 $x[stack]$ ，此时只需要设置步长为 2 进行遍历，将坐标点用特定颜色两两连线，全部操作结束即完成对多边形区域染色。需要说明的是，一般来讲交点数均为偶数，即使有一些特殊情况，例如与极值点相交、扫描线经过某边等情况，也都已经在算法具体代码实现过程中给出了相应的解决方案。

以上便是程序中函数 `draw_Polygons1` 和 `draw_Polygons3` 的主要思路，而对于其余几个函数，也只是在染色这一步骤中的具体实现方法有细微差别，具体的思路为：在 $x[stack]$ 两两配对后的序列中对 x 进行逐点遍历，利用 `getpixel` 函数获取该点颜色信息，然后根据不同情况进行区域填充（不再对线条染色而直接对区域染色），遇到边界即停止染色。需要说明的一点是，图案的边界曲线已提前绘制，只需利用传入各函数的坐标参数连线绘制即可。

3.1.2 程序及结果

在本题中，我们首先将两幅图用 \mathcal{J} 的形式表示。对于熊猫，可表示为 $\mathcal{J} = \cup_{k=1}^6 \mathcal{J}_k$ ，其中 $\mathcal{J}_1 = \{\gamma_1^+, \gamma_2^-, \gamma_3^-\}$ ， $\mathcal{J}_2 = \{\gamma_4^+, \gamma_5^-\}$ ， $\mathcal{J}_3 = \{\gamma_6^+, \gamma_6^-\}$ ， $\mathcal{J}_4 = \{\gamma_8^+\}$ ， $\mathcal{J}_5 = \{\gamma_9^+\}$ ， $\mathcal{J}_6 = \{\gamma_{10}^+\}$ 。对于米老鼠，可表示为 $\mathcal{J} = \cup_{k=1}^2 \mathcal{J}_k$ ，其中 $\mathcal{J}_1 = \{\gamma_1^+, \gamma_2^-, \gamma_3^-, \gamma_4^-\}$ ， $\mathcal{J}_2 = \{\gamma_5^+, \gamma_6^-\}$ 。

若要得到某单个图案，我们只需要重复利用定义的函数 `draw_Polygons1` 和 `draw_Polygons3` 反复对正向曲线和反向曲线围成的区域染色即可。

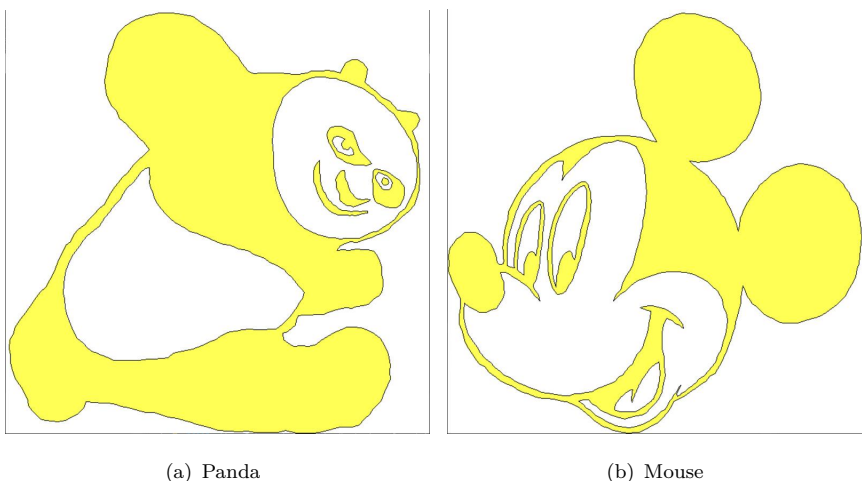


图 8: 原图案

若要得到单个图案进行补运算后的图，只需对调两函数即可。

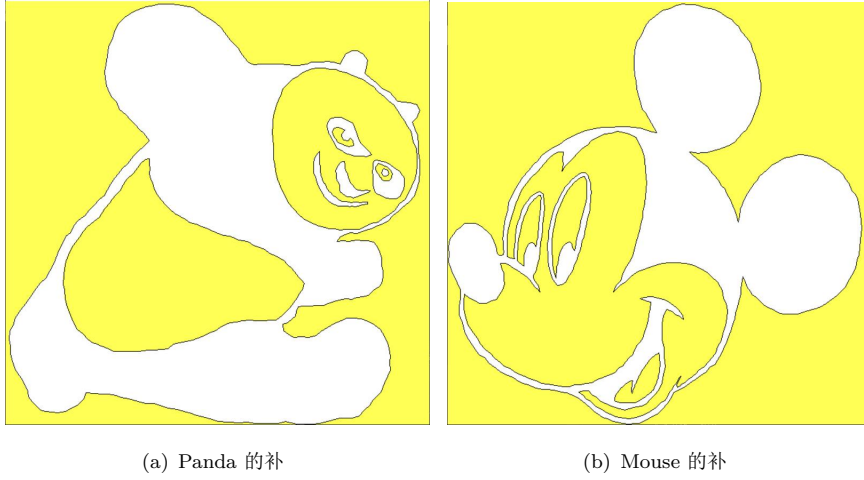


图 9: 原图的补运算

若要得到两图的交或并，我们首先确定一块主区域。假定我们选定熊猫作为主区域，先重复上述操作，此时我们在过程上得到了熊猫的图案。此时，我们只需要对构成米老鼠的每一个 \mathcal{J}_k 调用相应的函数，函数原理与此前提到的两个函数相似，不过输入参数增加至多条曲线的特征点， $x[stack]$ 的计算过程更为复杂。

在此部分的函数中，我们加入了检测该区域颜色的操作，便于交并运算的实现。若某区域属于熊猫图而不属于米老鼠图，则保持原色不变；若不属于熊猫图而属于米老鼠图，则染为其他一种颜色；若某块区域既属于熊猫图又属于米老鼠图，则染为另一种颜色。

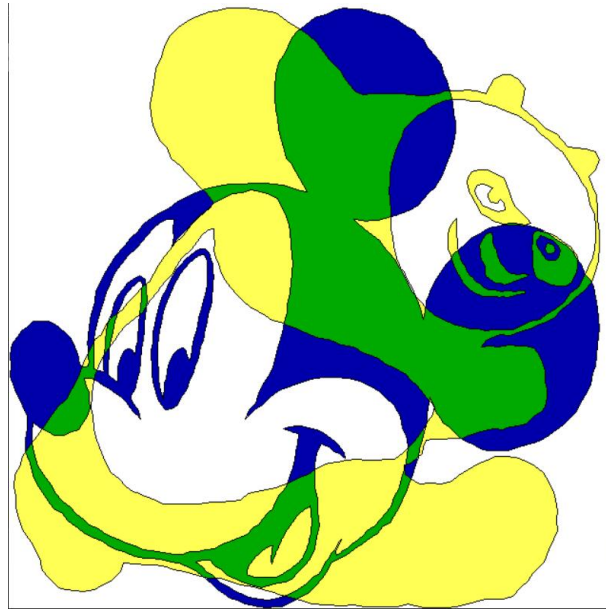


图 10: 交并运算的中间过程图

在进行上述操作后，我们得到了四个不同颜色的区域： $(P \cup M)^\perp$ 、 $P \cap M$ 、 $P \cap M^\perp$ 、 $M \cap P^\perp$ 。若要得到两图交运算后的图，我们只需要保留两图共有的区域，即 $P \cap M$ ；若要得到两图并运算后的图，我们只需要将 $P \cap M$ 、 $P \cap M^\perp$ 、 $M \cap P^\perp$ 三部分染为一种颜色。

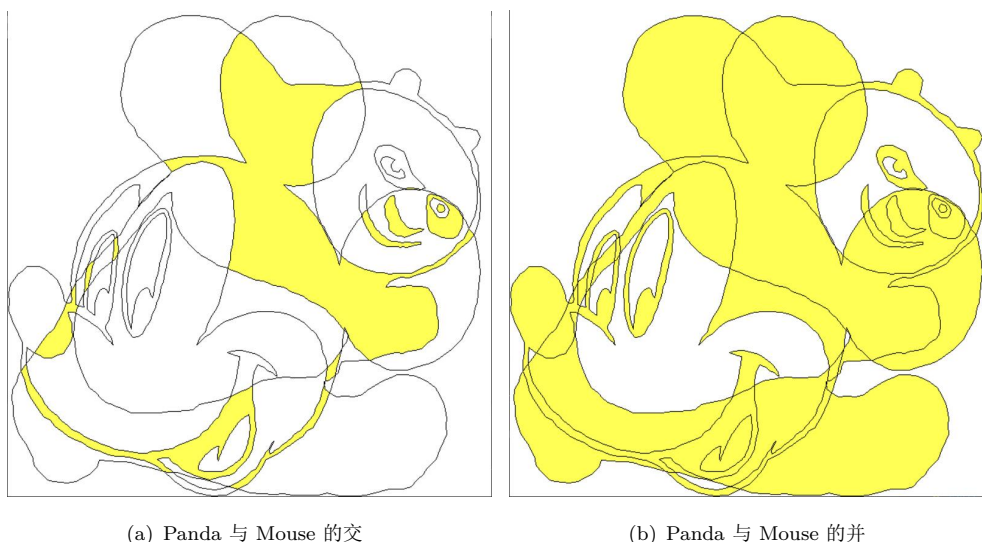


图 11: P 和 M 的交与并

需要说明的一点是，在本部分的程序中浮点数坐标读入后全部转换为整型参与运算，这是因为我们坐标系取得足够大，浮点数带来的误差几乎可以忽略。

3.2 论文的实现思路

上述的算法是一种比较朴素的图在进行交并补运算后呈现结果的实现方法。严格来讲，上述算法并没有实现对 Yin 集合的布尔代数运算。在询问邱助教后，我明白题目所希望的布尔代数是利用算法对 Yin 集合实现各种运算，其输入和输出的形式应该是近乎相同的，这种运算的封闭性是论文的价值所在，也是大作业的难点所在。

虽然没有完整地实现这种想法，但也有了一定的解决思路。首先是语言方面，我在本次大作业的编程语言上选择了 C 语言，但我认为采用 C++ 对于本部分会更方便便捷。要实现 Yin 集合的封闭运算，自然而然地想到定义封装一些类，然后对类进行相应操作。

我认为本文中上述提到的多边形扫描线算法，在此种方法的实现里也是能利用的，并且也是一个非常关键的部分，利用这种方法在两个多边形运算操作后得到新的多边形，这里多边形均通过边或者点的信息来记录和表示。不过很遗憾，我意识到这一点比较晚，且在思考后认为这种方法超出了目前我的能力范围，因此最终没有完成这种算法。

4 结语

本次大作业很好地锻炼了我的编程能力，也在一定程度上提升了我面对某个陌生问题时举一反三、自主设计算法进行解决的能力。在完成本次作业后，对暑期课程中提到的一些重难点有了切身的认识和体会，也更明白了论文的价值所在。总体而言，虽然尚有缺憾之处，但仍十分有成就感，且收获颇丰。