第三章 积分公式的切分与字符识别

3.1 积分公式的分割

为了进行下一步的识别，我们必须将积分公式进行分割，将所有单个字符提取出来。积分公式的分割共有两步，分别是图像二值化与图像连通域分析。

3.1.1 图像二值化

在数字图像处理中，二值图像占有非常重要的地位，图像的二值化使图像中数据量大为减少，从而能凸显出目标的轮廓。图像二值化的方法一般分为全局固定阈值算法和局部自适应阈值算法两类，全局固定阈值，就是对整幅图像都是用一个统一的阈值来进行二值化；常用的有最大类间方差法 OTSU[16]等。局部自适应阈值则是根据像素的邻域块的像素值分布来确定该像素位置上的二值化阈值。这样做的好处在于每个像素位置处的二值化阈值不是固定不变的，而是由其周围邻域像素的分布来决定的。亮度较高的图像区域的二值化阈值通常会较高，而亮度较低的图像区域的二值化阈值则会相适应地变小。不同亮度、对比度、纹理的局部图像区域将会拥有相对应的局部二值化阈值。常用的局部自适应阈值有局部邻域块的均值、局部邻域块的高斯加权和等。

本文采用局部自适应阈值算法——基于图像累加的局部自适应快速二值化算法（Adaptive Thresholding Using the Integral Image）[17]。

图像累加：图像累加是指将图像某个点的左上方的所有点的像素值进行累加，等到的累加和即为这个点的值。例如在一幅单通道图像中，各个点的像素值如图 3-1 左。之后得到每个点的左上方所有点的像素值之和（包括该点）并可用一个二维数组暂时存储，如图 3-1 右所示，例如，第二行第二列中 9=4+0+4+1，第二行第三列中 12=4+1+2+4+1。

4	1	2	2
0	4	1	3
3	1	0	4
2	1	3	2

4	5	7	9
4	9	12	17
7	13	16	25
9	16	22	33

图 3-1 图像累加举例

一般地，所谓二值化就是给定一个阈值，当图像中某一点的像素值大于阈值时令它的像素为 255(0)，像素值小于阈值时为 0(255)。而所谓区域二值化，是把图片划分为若干的区域，每个区域有各自的阈值，再分别判定。

如图 3-2 所示，我们要判定区域 D 时，就可以利用图 3-1 右的表，设 P(x，y)为点(x,y)的像素值（已转换为图 3-1 右的值），对于区域 D 有，P(D)=P(x2，y2)-P(x2,y1)-P(x1,y2)+P(x1,y1)，得到区域 D 的总阈值，再除以区域 D 的像素点个数得到平均阈值 P=P(D)/N,对于区域中心点(i，j)若 P(i，j)>P,则重写点(i, j)的像素为 255(0)，反之为 0(255)。

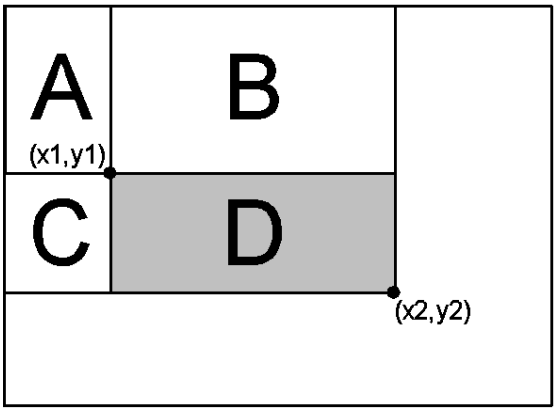

图 3-2 分区后的图

Wellner 于 1993 年提出的算法[18]的主要思想是，每个像素都与周围像素的平均值相比较。具体地说，在遍历图像时，计算出的已扫描的距离该像素点最近 s 个像素的近似平均值。如果当前像素的值比平均值低 t 个百分点，那么它就被设置为黑色，否则它将被设置为白色。这种方法之所以有效，是因为将像素与附近像素的平均值进行比较，将保留强烈的对比度，忽略平淡的渐变变化。这种方法的优点是只需要通过图像的单个传递，但是这个方法的一个问题是它依赖于像素的扫描顺序。此外，平均值在每个步骤中都不能很好地表示周围的像素，因为附近的样本并不是均匀分布在各个方向上的。

Derek Bradley 和 Gerhard Roth 提出了一个不受这些问题影响的解决方案[17]。即基于图像累加的局部自适应快速二值化算法（Adaptive Thresholding Using the Integral Image）Derek Bradley 和 Gerhard Roth 的自适应阈值技术是 Wellner 1993 年方法的一个简单扩展。他们使用了以某一像素点为中心，2s 大小的面积的平均值作为用来比较的平均值。这是一个比较好的平均值，因为它考虑了所有边的相邻像素。通过使用累加图像，平均计算是在线性时间内完成的。算法在第一个过程中通过输入图像来计算累加图像。在第二个过程中，算法用恒定时间内每个像

素的累加图像来计算某一像素点的平均值，然后进行比较。如果当前像素的值小于这个平均值，那么它就被设置为黑色，否则它将被设置为白色。下面的伪代码演示了局部自适应快速二值化算法

w 为图像宽度，h 为图像高度 h，根据论文[17]的内容，s 取宽度 w 的 1/8，t 取 15

基于图像累加的局部自适应快速二值化算法(in,out,w,h)：

```
计算 im 的图像累加和 intImg
for i = 0 to w do
    for j = 0 to h do
        x1 = i − s/2 {border checking is not shown}
        x2 = i + s/2
        y1 = j − s/2
        y2 = j + s/2
        count = (x2−x1)×(y2−y1)
        sum = intImg[x2,y2] − intImg[x2,y1−1] − intImg[x1−1,y2] +
              intImg[x1−1,y1−1]
        if (in[i, j] × count) ≤ (sum×(100−t)/100) then
            out[i, j] = 0
        else
            out[i, j] = 255
        end if
    end for
end for
```

图 3-3 一幅光照不均匀的图像，图 3-4 是使用基于图像累加的局部自适应快速二值化算法进行二值化的结果展示图片，可以看出使用这种算法对光照不均匀的图像可以得到一个较完美的结果

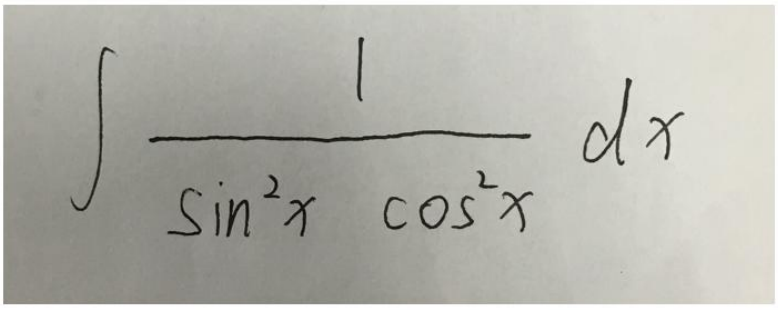

图 3-3 未进行二值化的光照不均匀的图像

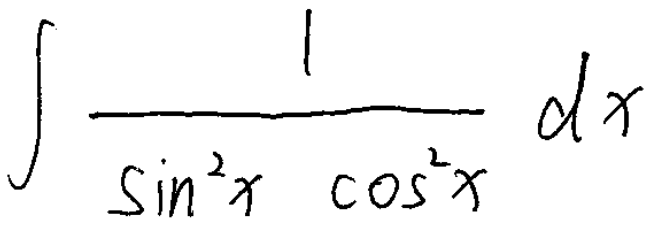

图 3-4 二值化后的图像

3.1.2 连通域分析

连通域分析一般采用四连通或者八连通如图 3-5

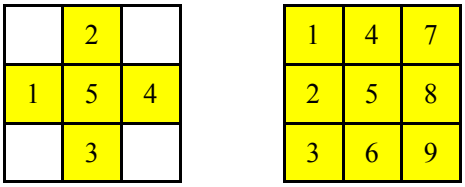

图 3-5 四连通域和八连通域

考察中心点，在四领域内它与 1234 位连通，与其他位不连通，在八领域内，它与周围 12346789 位都连通。

本文采用八连通种子填充算法，算法如下。

1. 扫描图像，直到当前像素点 B(x,y) == 1：

   a. 将 B(x,y)作为种子（像素位置），并赋予其一个 label，然后将该种子相邻的所有前景像素都压入栈中；

   b. 弹出栈顶像素，赋予其相同的 label，然后再将与该栈顶像素相邻的所有前景像素都压入栈中；

   c. 重复 b 步骤，直到栈为空；此时，便找到了图像 B 中的一个连通区域，该区域内的像素值被标记为 label；

2. 重复第 1 步，直到扫描结束；

   扫描结束后，就可以得到图像 B 中所有的连通区域；

3.2 单个字符的识别

3.2.1 字符集的建立

本文的手写字符集收集自本专业同学的手写笔迹，平均每个字符收集 100 个，经过变形、旋转、加噪等变换，最终每个字符扩大到 700 个左右。最终字符集为：三十九类，25001 个样本。其中 80%作为训练集，20%作为测试集。

3.2.2 卷积神经网络

卷积神经网络(Convolution Neural Networks，简称 CNN)是一类特殊的人工神经网络，区别于神经网络其他模型，它的最主要特点是卷积运算操作(convolution operators)。因此，CNN 在诸多领域应用特别是图像相关任务上表现优异，如图像分类(image classification)、图像语义分割(image semantic segmentation)、图像检索(image retrieval)、物体检测(object detection)等计算机视觉问题[19]。

自 2006 年 Hinton 和 Salakhutdinov 在 Science 上发表的深度学习论文点使神经网络复兴，接着，2012 年 AlexNet 在 ImageNet 上夺冠又迅速促进了深度学习在人工智能领域的发展。当下深度学习算法主宰了计算机视觉、自然语言学习等众多人工智能应用领域。

与此同时，与学术研究一起快速发展的还有层出不穷的许多深度学习开源开发框架。其中 matconvnet 是由英国牛津大学著名计算机视觉和机器学习研究组 VGG 负责开发，是一个 MATLAB 工具箱，用于实现计算机世界应用的卷积神经网络。本课题就使用了 matconvnet 框架进行研究。

3.2.3 CNN 网络及其参数

CNN 网络结构：

第一层：输入层，图片大小为 28×28。

第二层：卷积层，卷积核为 5×5×1×20，用 0 填充，步长为 1，初始值由标准正态随机数产生。

第三层：池化层，池化操作核大小 2×2，使用最大池化方法，用 0 填充，步长为 2。

第四层：卷积层，卷积核为 5×5×20×100，用 0 填充，步长为 1，初始值由标准正态随机数产生。

第五层：池化层，池化操作核大小 2×2，使用最大池化方法，用 0 填充，步长为 2。

第六层：卷积层，卷积核为 4×4×50×500，用 0 填充，步长为 1，初始值由标准正态随机数产生。

第七层：激活层，使用 RELU 函数，

第八层：卷积层（全连接层），卷积核大小为 1×1×500×1000，用 0 填充，步长为 1，初始值由标准正态随机数产生。

其余参数：

训练共 50 轮，学习率为 0.001，批处理大小为一次 100 张。

3.2.4 训练结果

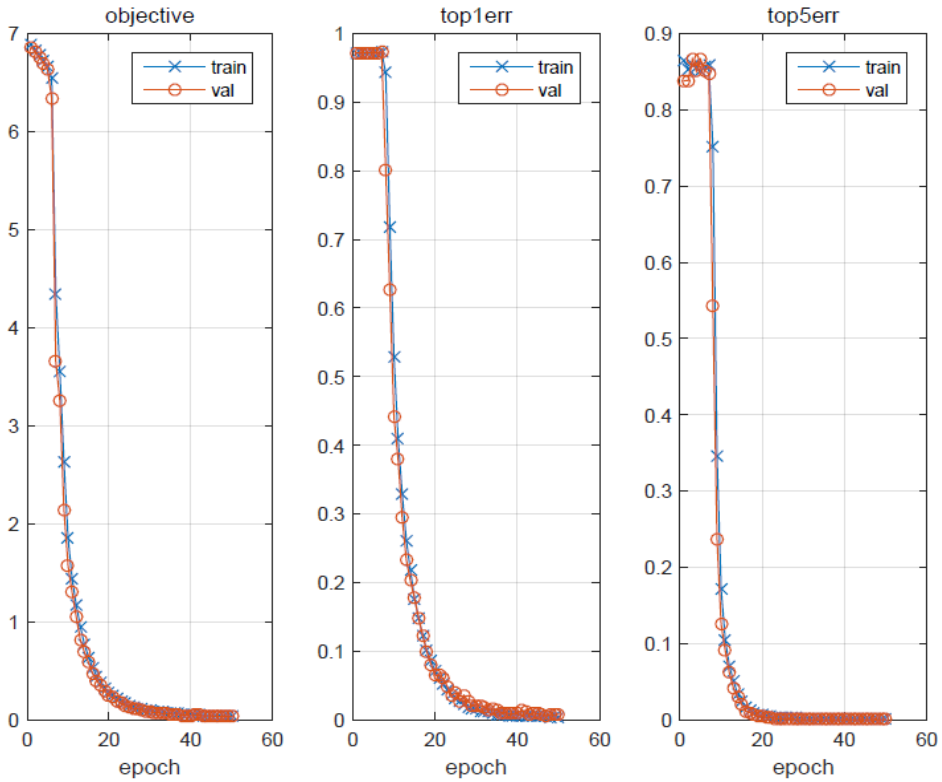

图 3-6 训练过程图

训练 50 轮结果如图 3-6，在测试集上的正确率为 99.2%

3.3 本章小结

本章主要做了图像的预处理和单个字符识别，在图像预处理方面，本章使用了基于图像累加的局部自适应快速二值化算法，对光照不均匀的图像进行了二值化，之后使用八连通种子填充算法对图像进行了连通域分割。最后本章使用了卷积神经网络对收集到的三十九类，25001 张图片进行了训练。在测试集上得到了99.2%的正确率。

第四章 积分公式的结构分析和复原

4.1 BST 算法

BST 算法[20]即公式基线结构树算法，以下都简称 BST。BST 算法的输入为所有的符号及其属性，输出为一棵公式基线结构树（Baseline Structure Tree）。

4.1.1 BST 算法的输入

BST 算法的输入是一张表，以一幅图片的左下角为原点，水平为 x 轴，垂直为 y 轴，建立每个符号的参数如下表。表中的每一行代表一个符号，每一列代表一种属性，符号的属性如表 4-1。

表 4-1 符号属性表

属性	意义
NO	符号的编号
centent	符号的识别结果
class	符号的类别
minX maxX minY maxY	符号最左、右、下、上边界
below above super subsc	符号上域、下域、右上域、右下域的边界
x-centroid	符号的水平中心点
y-centroid	符号的垂直中心点
topWall bottonWall leftWall rightWall	符号领域的最高点、最低点、最左点、最右点
child	符号的子节点，为域节点

其中：

1. NO 是每一个符号的编号，符号从左到右进行排序，编号从最左侧开始编。
2. content 来自于上一章中训练的结果。在实际处理中，因为类似于分数线、积分等符号不易于表示，因此给每个符号一个代替值 content，作为其表示。content 编号为 1-46，符号对应为：1, 2, …, 9, 0, a, b, …, z, +, -, 分数线, ∫, π , (,), 根号, 无穷, 点。
3. 符号的类别：数学符号按照其结构以及功能可以分为 7 类，分别为：Non-Scripted, Open Bracket, Root, Variable Range, Ascender, Descender, Centered. 每类的所包含的符号如表 4-2.
4. minx, maxX, minY, maxY 分别为符号最左侧最右侧最上侧最下侧的坐标。
5. below, above, super, subsc 分别代表符号的四个域的边界，域的概念在下文

中详述，域边界的值与符号的类别有关

6. x-centroid 与 y-centroid 分别代表符号的水平中心和符号的竖直中心，其中，符号的 x-centroid 属性为符号宽度的 1/2，y-centroid 与符号的坐标以及符号的类别有关，如表 4-2

表 4-2 符号类别与边界值对应表

符号类别	y-centroid	边界			
		below	above	super	subsc
Non-scripted “+, -, *, .,”	1/2H	1/2H	1/2H	-	-
Open Bracket “(”	cH	minY	maxY	-	-
Root “根号”	cH	minY	maxY	H-tH	tH
Variable Range “ ∫ ”	1/2H	tH	H-tH	H-tH	tH
Ascender “0…9, b, d f, h, i, k, l, t”	cH	tH	H-tH	H-tH	tH
Descender “g, p, q, y”	H-cH	1/2H+1/2tH	H-1/2tH	H-1/2tH	1/2H+1/2tH
Centered “a, c,e,j,m,n,o,r,s,u,v,w,x,z,., π ,), ∞”	1/2H	tH	H-tH	H-tH	tH

表 4-2 中，c 和 t 分别为中心参数和域参数。根据论文已得的结果，我们将 c 和 t 分别设置为 1/3 和 1/6

7. child 代表改符号的子节点，子节点为域节点，在下文中详述。
8. rightwall, leftwall, topwall, bottonwall 符号领域的最右点、最左点、最高点、最低点。rightwall, leftwall, topwall, bottonwall 属性的定义如下：
   1. 设 S 为一个域节点 R 中在基线上的符号的集合
   2. For i = 1 : n-1, n 代表 S 中元素的个数
      设置 Si (rightWall) = Si+1(minX);
      设置 Si (leftWall) = Si (minx);
      设置 Si (topWall) =R (RmaxY);
      设置 Si (bottonWall) = R(RminY);
      其中域节点的定义在下文中详述

4.1.2 积分公式中的符号布局

在积分表达式中，数学符号的意义以及数学符号布局联合起来传达了表达式的含义。比如’3^2+1’中，四个符号的意义分别表示了这是三个数字及其数值大小以及

一个操作符要进行加法运算，而其布局则表达了这个四个数字之间如何操作。比如根据 3 和 2 的布局我们可以知道加号要加的是 3 和 1，而不是 2 和 1.

4.1.2.1 符号的域、分类、属性和基线

基线的定义：基线是符号的水平排列，两个符号水平相邻即认为在一条基线上。例如：$a^{x+y} - b^{x-y}$ 中，(a, -, b)在一条基线上，(x, +, y)在一条基线上，(x, -, y）在一条基线上。

主基线：在一个域中，主基线包含的符号不包含在其他符号的域中。

支基线：在一个域中，支基线上所包含的符号包含在其他符号的域中。

例如上例中，(a, -, b)为整个域的主基线，(x, +, y)和(x, -, y)为支基线。

域的定义：域是指每个字符的嵌套字符所在的区域。 域可以分为 ABOVE（上标区域）、BELOW（下标区域）、SUPER（右上区域）、SUBSC（右下区域）、CONTAINS（包含区域）、HOR（平行区域）共 6 个区域，每个字符有全部或部分的域。例如图 4-1 中字母’d’的域。

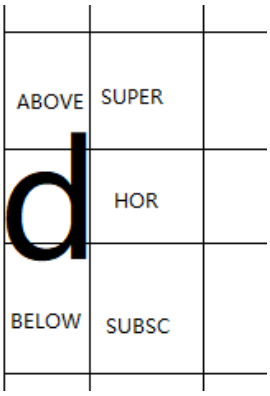

图 4-1 域的示例

域的范围定义如表 4-3，其中，RminX, RmaxX, RminY, RmaxY 分别代表域的左、右、下、上边界。

表 4-3 域的范围

域	RminX	RmaxX	RminY	RmaxY
SUPER	maxX	rightWall	super	topWall
SUBSC	maxX	rightWall	bottonWall	subsc
ABOVE	minX	maxX	above	topWall
BELOW	minX	maxX	bottonWall	below
CONTAINS	minX	maxX	minY	maxY
HOR	maxX	rightWall	subsc	super

其中有三条限定如下：

- 只有 root 类有 CONTAINS 域
- Non-scripted 和 Open Bracket 类没有 SUPER 域
- Non-scripted 和 Open Bracket 类没有 SUBSC 域

4.1.2.2 基线结构树

基线结构树表示了积分表达式中基线的层次结构。基线结构树明确地捕捉了符号布局的重要方面，而不需要任何特定的句法或语义解释。例如，基线结构树可以用来表示“2+”的符号布局，尽管这个表达式在语法上和语义上都是无效的。类似地，基线结构树代表“f(x)”的符号布局，不管函数是关于 x 的乘法，或是一个关于 x 的三角函数。

基线结构树包括两种节点：符号节点和域节点（下方定义）。整个公式的结构就通过这些节点层层组织起来。树的根节点“EXPRESSION”是一个域节点，它就代表了整棵树。

符号节点：符号节点代表了一个数学符号。每个符号节点存储该符号的属性。符号节点的子节点为域节点，一个符号节点的种类决定了该符号有哪些域节点。

域节点：一个域节点是由确定该域的符号定义的，每个域节点包括了一条基线。域节点的属性有域标签，域的范围(RminY, RmaxY, RminX, RmaxX)。每个域节点的子节点就是该域中位于主基线上的符号节点。

域标签：域标签共有 8 种，EXPRESSION（即树的根）, Non-Scripted, Open Bracket, Root, Variable Range, Ascender, Descender, Centered.

以 $x^2+\frac{y}{3}$ 为例，EXPRESSION 包含 3 个子节点（x,+,-），x 包含一个子节点 super，“-”包含两个子节点 above 和 below，super 包含一个子节点（2），above 包含一个子节点（y），below 包含一个子节点（3）如图 4-2。

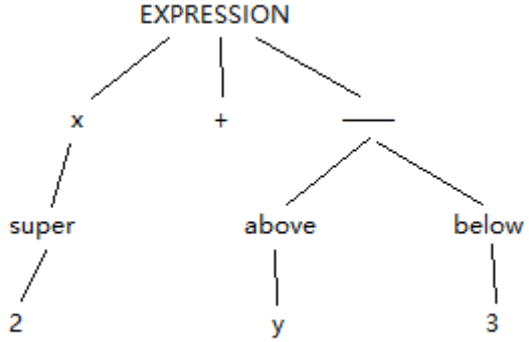

图 4-2 一个 BST 公式基线结构树

4.1.3 BST 算法

BST 的输入为一串字符以及其属性，输出是一个类似于图 4-2 的公式基线结构树。

4.1.3.1 一级函数算法

BST = BuildBST(snode_list)：从输入 snode_list（输入的字符及其属性）建立一个基线结构树。

1. 设置树根 R 的 label 为 EXPRESSION, if | snode_list | = 0, return R
2. snode_list1 = SortSymbolsByMinX(snode_list)
3. 设置每个在 snode_list1 中的符号为 R 的孩子
4. 返回 ExtractBaseline (R)

RNODE1 = ExtractBaseline (RNODE)：找到域 RNODE 中的主基线，同时更新以 RNODE 为根的基线结构树。

1. 使 snode_list = Symbols (rnode). if | snode_list | ≤ 1, return rnode
2. 使 S_start = Start (snode_list)
3. 使 baseline_symbols = Hor (S_start, snode_list).
4. 用 updated_baseline 中的符号节点替换 rnode 的孩子
5. 对于 updated_baseline 中的符号节点的每一个子节点 childrnode，ExtractBaseline(childrnode)
6. return rnode.

SNODE1 = Start (snode_list)：找到 snode_list 中的主基线的起始节点

如果 1. Overlaps (Sn, Sn-1)

或者 2. Contains (Sn, Sn-1)

或者 3. class (Sn) = Variable Range 并且 !IsAdjacent (Sn-1, Sn)

那么 Sn 支配 Sn-1, 否则 Sn-1 支配 Sn。

SNODE_LIST3 = Hor (SNODE_LIST1, SNODE_LIST2): 找到由 SNODE_LIST1 中的符号开始，由 SNODE_LIST2 中符号继续的基线。基线中的符号返回存储在 SNODE_LIST3。SNODE_LIST2 中的符号如果没有在基线上，则被划分到 ABOVE, BELOW, SUPER, SUCSC, CONTAINS 中。

1. if | snode_list2| = 0, return snode_list1.
2. 使 current_symbol = snode_list1 中的最后一个符号节点
3. 使 (remaining_symbols, current_symbol1) = Partition (snode_list2, current_symbol).

4. 在 snode_list1 中， 用 current_symbol1 替换 current_symbol。
5. if | remaining_symbol| = 0, return snode_list1.
6. if class (current_symbol1) = Nonscripted then return Hor (ConcatLists (snode_list1, (Start (remaining_symbol))), remaining_symbols).
7. 使 SL = remaining_symbols.
8. while | SL | ≠ 0
   a. 使 L = SL 中的第一个符号节点
   b. 如果 IsRegularHor(current_symbol1, L), 那么就 return Hor (ConcatLists (snode_list1, (CheckOverlap (L,remaining_symbols))), remaining_symbols).
   c. 使 SL = SL –L.
9. 使 current_symbol1 = PartitionFinal (remaining_symbols, current_symbol1).
10. return ConcatLists(snode_list1, (current_symbol1)).

4.1.3.2 二级函数算法

下方的二级函数是在上方的一级函数中调用过的。

snode1 = AddToRegion(label, snode_list, snode)：函数作用是使 snode_list 中的符号节点成为 snode 的孙子节点，label 是指 snode 子节点的标签，比如如果 label = ABOVE, 则 snode_list 中的节点成为 snode 的 above 子节点的孩子。

snode1 = CheckOverlap(snode, snode_list)：通过 Overlaps 函数来查找 snode_list 中水平覆盖 snode 节点的 Noscripted 类型节点。如果存在这样的节点，就返回最宽的节点；如果不存在，就返回 snode

snode_list3 = ConcatLists ((snode_list1, snode_list2))：将 snode_list1 和 snode_list2 的并集放入 snode_list3 中

Boolean = Contain((snode1,snode2))：判断 snode1 是否为 root 类型，并且包含 snode2.

如果 snode1 ≠ snode2 并且 class(snode1) = Root 并且 minx(snode1) < controidX(snode2) ≤ max(snode1) 并且 minY(snode1) ≤ centroidY(snode2) < maxY(snode1), return true

否则 return false

Boolean = HasNonEmptyRegion((snode, region_label)：判断 snode 的 region_label 域是否为空

如果 snode 有子节点的 label 为 region_label，return true，否则 return false

boollean = IsAdjacent ((snode1, snode2)：测试 snode1 与 snode2 是否水平相邻，

无论是左侧或是右侧。

如果 class (snode2) ≠ Nonscripted 并且 snode1 ≠ snode2 并且 subsc (snode2) ≤ centroidY(snode1) < super (snode2)

return true

boolean = IsRegularHor ((snode1, snode2))：用来判断 snode1 和 snode2 是否是普通的平行关系，即判断 snode1 和 snode2 是否在一条 baseline 上

如果 1. IsAdjacent(snode2, snode1), 或者

2. maxY(snode1) <=maxY(snode2) 并且 minY(snode1)>=minY(snode2) 或者

3. class(snode2) = open Bracket 或者 close Bracket 并且 minY(snode2) <= centroidY(snode1) < maxY(snode2)

boolean = Overlaps((snode1, snode2))：测试 snode1 是否是 Nonscripted 类型,并且垂直覆盖 snode2.

如果 1. snode1 != snode2 并且

2. class(snode1) =nonscripted 并且

3. minx(snode1) <=centroidX (snode2) <maxX(snode1) 并且

4. !Contain(snode2,snode1)

5. i 和 ii 都为假

i．class(snode2) = openbracket 或者 close bracket，

并且 minY(snode2）<= centroidY(snode1) < maxY(snode2)

并且 minx(snode2) <= minx(snode1)

ii. class(snode2)= nonscripted 或者 varialble range

并且 maxX(snode2) – minx(snode2) > maxX(snode1) – minx (snode1)

snode_list2, snode2 = Partition((snode_list1, snode1)：测试 snode_list1 中的符号节点是否属于 snode1 中的域，不属于的符号节点将被放入 snode_list2，属于的符号节点就放置在 snode1 相应的孩子域节点中，之后将 snode1 赋值给 snode2

snode1 = PartitionFinal (snode_list，snode)：snode 是一条基线的最后一个符号节点。snode_list 中的符号节点将放置在 snode 节点的 subsc 或 super 子节点中。将 snode 赋值给 snode1

snode_list2 = SortSymbolsByMinX (snode_list1)： 按照 snode_list1 中的符号节点的 minX 属性，来从小到大进行排序，排好序后存入 snode_list2 中。

snode_list = Symbols (rnode)：返回 rnode 的孩子

4.1.4 简化版 BST 算法

在研究 BST 算法时，我想到了一个更为简化的算法，称为简化版 BST 算法，简化版 BST 算法不需要 Wall 属性，相对于 BST 算法，简化版 BST 算法更易于理解，易于编程，更轻量化。其中 BST = [BST snode1.NO]是 matlab 语句，意思是在 BST 后直接拼接 snode1.NO，之后新的数组放入 BST 中

主函数算法如下：

BST = Construct(rnode)：找到域节点 rnode 中位于基线上的符号，放入 BST 中。其余符号如果没有在基线上，则被划分到相应符号的 ABOVE, BELOW, SUPER, SUCSC, CONTAINS 中。

1. 如果 rnode 为空，直接返回
2. 使 snode 为 rnode 中的第一个符号，snode1 为下一个符号，将 snode 放入 BST
3. 如果 snode 类型为'non_scripted'， [boollean,label] = Cover(snode, snode1); 如果 boollean 为 true，snode = AddToRegion(label, snode1, snode);否则将 snode1 放入 BST
4. 如果 snode 类型为'root'，boollean = includes(snode, snode1)，如果 boollean 为 true，snode = AddToRegion('contains', snode1, snode)；否则[boollean1, label] = IsNext(snode1, snode)，如果 boollean1 为 true，snode1 放入 BST，否则 snode1 放入 snode 相应的域中。
5. 如果 snode 类型为'ascender'或'descender'或'centerd'或 'variable_range'，则 [boollean,label] = IsNext(snode1, snode)，如果 boollean 为 true，snode1 放入 BST，否则 snode1 放入 snode 相应的域中。
6. 如果 snode1 和 snode 是水平相邻，snode=snode1，snode1 为下一个符号，跳转到第 3 步。
7. 对每个字符的子域节点 Rnode 执行 Construct(Rnode)

子函数算法如下：

bool = includes（snode1, snode2）：includes 函数判断 snode1 是否包含符号节点 snode2.如果 snode1.minX <= snode2.x_centroid 并且 snode2.x_centroid < snode1.max，返回 bool 为 true，否则返回 bool 为 false.

[bool, label] = Cover(snode1, snode2)：cover 函数判断 snode1 是否上下覆盖 snode2，用来判断某个符号是否在分数线的上下域。如果 if snode1.minX <= snode2.x_centroid 并且 snode2.x_centroid <= snode1.maxX，返回值为 true，否则为 false。如果 snode2.y_centroid >= snode1.maxY， label = 'above'; 否则如果 snode2.y_centroid <= snode1.minY，label = 'below';否则 label = 'null' 。

[bool, label] = IsNext(snode1, snode2)：判断 snode1 和 snode2 是否水平相邻，即两个符号节点都不在任意一方的域内。如果 snode2.subsc <= snode1.y_centroid 并且 snode1.y_centroid <= snode2.super，则返回 bool 值为 true，label 为空。否则 bool 为 false，同时如果 snode2.subsc > snode1.y_centroid，则 label 为”subsc”，如果 snode1.y_centroid > snode2.super，label 为”super”。

4.1.5 结果展示

以图 4-3 为例进行展示，图 4-3 为原图。

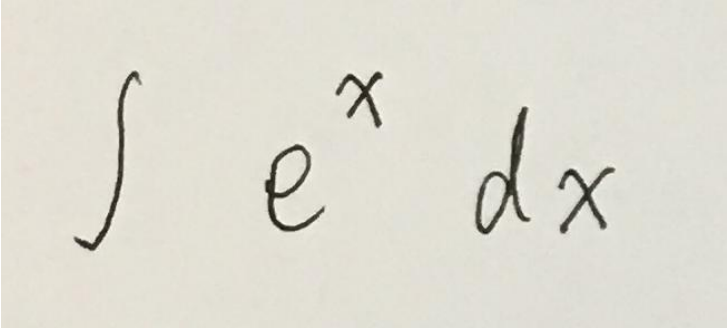

图 4-3 结果展示

在进行公式基线结构树建立算法之后得到的中间结果是一个字符节点表格，如表 4-4 所示

表 4-4 建立公式基线结构树所得表格

NO	1	2	3	4	5
content	41	15	34	14	34
maxX	57	208	255	376	445
minX	2	159	217	331	399
maxY	148	88	148	119	70
minY	2	17	102	15	16
class	variable_range	centerd	centerd	ascender	Centerd
y_centroid	75.5	53	125.5	50	43.5
x_centroid	30	184	236.5	354	422.5
below	26.5	29	109.8333	32.5	25.16667
above	124.5	77	141.1667	102.5	61.83333
subsc	26.5	29	109.8333	32.5	25.16667
super	124.5	77	141.1667	102.5	61.83333
child	struct	struct	struct	struct	struct

上面表格中没有 number 属性是因为在图 4-3 中，并没有数字符号，因此在数字合并步骤中不需要进行数字合并，于是就没有产生 number 属性。child 为子域节点，每个域节点都是一个 struct，域节点举例使用 NO 为 2 的符号节点的 super 域节点如

表 4-5。

表 4-5 NO 为 5 的符号节点的 super 域节点

children	3
RmaxY	inf
RminY	77
RminX	208
RmaxX	331

其中 children 为 3 代表这个域节点的子节点为 NO 为 3 的符号节点。RmaxY 是 inf 代表无限，RminY 是符号节点的 super 属性，RminX 是符号节点的 maxX 属性等等，这些属性的赋值都在前文详述，在此不再赘述，

4.2 BST 分析算法

在前文中，已完成了每个字符的识别以及积分公式的结构分析建立了 BST 公式基线结构树，得到了一张记录所有符号节点属性的表格。本节中，将要利用这张表格作为输入，要完成对 BST 公式基线结构树的分析，输出一个符合 matlab 语法，可以被 matlab 执行的语句。

4.2.1 字符合并算法

在上一节中，每个字符都是分开独立的，但是诸如一些函数，比如 cos 和 sin 等，以及数字都是需要进行合并的。在进行公式基线结构树语义分析之前，首先要进行预处理，将这些字符进行合并。

4.2.1.1 函数的合并

对于 cos, sin, cot 等函数要对其进行合并，合并算法如下：

1. 取主基线上的所有字符的内容，从左到右排列为一个基线字符串
2. 所有函数集合为 S = {'lg'，'ln', 'log', 'sin', 'tan', 'cos', 'arcsin', 'arctan', 'arccos','sec','cot','csc'}共 12 个元素 i=1
3. 首先用字符串 S(i)对基线字符串进行匹配，看是否存在 S(i)，若不存在则令 i = i+1 匹配下一字符串，若存在则记录基线字符串中出现 S(i)的位置为 P1, P2 … Pn, j=1，
4. len 为 S(i)的长度，将 Pj 的 content 修改为 S(i)，修改 maxX 为 Pj+len 的 maxX，minY 修改为几个字符节点中最小值，maxY 修改为几个字符节点中最大值，同时修改其领域，不做赘述。删除 Pj+1 到 Pj+len 字符节点，保留 Pj 节点。
5. j = j + 1，若 j <= n 则返回 4 步

6. i = i + 1，若 i <= 12 则返回 3 步
7. 分别对基线字符串的每个字符的子节点执行上述步骤
8. 返回更新的表

注意：函数的匹配是有顺序的，一定要先匹配完 cos, sin 和 tan 再匹配 arccos, arcsin 以及 arctan。如果先匹配 arccos, arcsin 以及 arctan，在匹配 cos, sin, tan 时就会将反函数中的字符串再次匹配。

4.2.1.2 数字的合并

在基线上的字符串中，数字还只是一串字符串，如 3.14 现在还是{‘3’, ‘.’ ,’1’,’4’}，下述算法就是实现了将字符串转换为数字。在字符表的 content 属性中，72 代表是数字的意思。

1. 取主基线上的所有字符的内容，从左到右排列为一个基线字符串。
2. 查找基线字符串中内容为 0-9 的节点，记录所有位置。
3. 记录连续的一串数字的起始位置 S1…Sn 和长度 L1…Ln，其中 S1 代表一串连续的数字的起始位置，S2 代表另一串连续的数字的起始位置……设 i=1，pos=1，count=0，设 num 为空字符串。
4. 将位置为 S（i）+count 的字符节点的内容赋值给 num(pos）。
5. 检查字符节点 S（i）中的 subsc 子节点是否有‘.’小数点。如果有，pos 增加一，将小数点放在 num（pos）中，如果没有，则不进行操作。
6. count=count+1，如果 count<=L(i)，则 pos 增加一，转至第 4 步，否则进行下一步。
7. 利用 matlab 已有的函数将 num 从字符串型转化为数字型，将位置为 S（i）的字符的 content 改为数字标识 72，之后多加一个属性 number，存放转化为数字型的 num，修改位置为 s（i）的字符的上下左右域，删除位于 s(i+ 1) 至 S(i+count)的字符节点。
8. i=i+1, 如果 i<=n, 则 count = 0 ，pos = 1，num 赋值为空字符串，转至第 4 步，否则结束，返回更新的表。

4.2.2 BST 公式基线结构树分析算法

这一步中，要将所得到并且进行过更新的公式基线结构树 BST 进行语法分析，得到 matlab 可以执行的语句。

首先要将所有字符根据其意义进行分类，每类的内容以 content 表示，共分 11 个类分别为：除了积分变量之外的变量；积分变量；37, 38, 39；41；42；43；44；

60-71；72；15。

公式基线结构树的语法分析算法，str = Analyse(snodelist)：核心思想主要是向前扫描一个字符，根据前一字符的类型，以及此字符的类型，判断此时的字符应当如何操作。以下只列出核心算法思想，完整算法参考附录一。

1. snode 为当前符号节点，snode1 为前一个符号节点。
2. 根据 snode 类型，递归调用 Analyse 函数分析 snode 的子域。比如对于 e^x，有 str1=Analyse(snode.super)。而 $\sqrt{x}$，有 str1=Analyse(snode.contains)
3. 根据 snode1 类型，判断如何进行字符串拼接操作。比如对于”1+x^2”，snode1 为 ‘+’，snode 为 ‘x’，已得的字符串 str=”1+”，则拼接方式为 str=[str ‘x’ ‘^’ str1]。而对于”2x^2”，snode1 为 ‘2’，snode 为 ‘x’，已得的字符串 str=”2”，在拼接方式为 str=[str ‘*’ ‘x’ ‘^’ str1]。
4. snode1 赋值为当前节点，snode 赋值为下一节点，返回步骤 2。

下面，以积分公式∫(e^x+1)dx 为例，对此算法做一个说明。假设经过前边的操作，已经得到了一个公式基线结构树如图 4-4

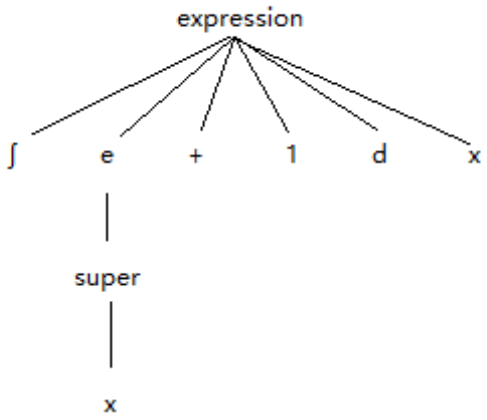

图 4-4 公式基线结构树 BST

1. 删除积分符号和积分变量，只保存积分函数 e^x+1。
2. 令 s 为首个字符节点，首先判断 s 的类型为 11，之后分析 s 的 super 域即 str1= Analyse (s.super)，得到字符串 str1 = ’x^(1)’. str = [‘exp(’ str1 ‘)’]即 str = ’exp(x^(1))’。
3. 令 s 为基线上的下一字符节点，判断 s 的类型为 3，则 str = [str s.content]即 str = ’exp(x^(1))+’。
4. 令 s 为基线上的下一字符节点，判断 s 的类型为 10，则分析 s 的 above 域，即 str1 = Analyse (s.super)得到字符串 str1 = ’1^(1)’，分析 s 的上一字符，上一

字符类型为 3，则 str = [str str1]，即 str=’exp(x^(1))+1^(1)’，这就是一个可以被 matlab 理解的字符串。

4.3 本章小结

在本章中，实现了两种公式基线结构树 BST 的建立算法，其中一种来自于文献[20]，另一种由我自己提出。在构建公式基线结构树之后，本章提出了公式基线结构树的语法分析算法，完成了公式基线结构树到可执行 matlab 语句的转换。

第五章 系统设计与实现

5.1 需求分析

5.1.1 需求概述

本系统是一个积分公式的识别和求解系统。本系统以识别一幅图像中的积分公式并且求解该积分公式为目的，包括了图像的浏览、选择、识别、报错等功能。图5-1 是系统环境图

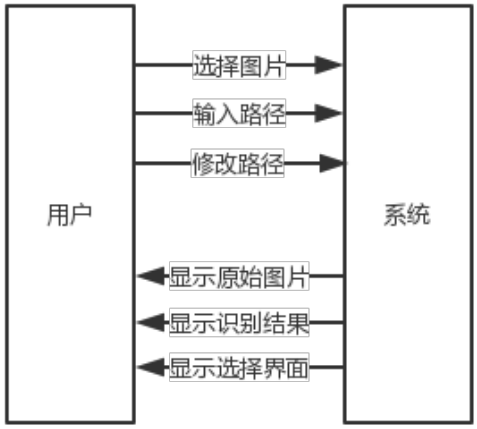

图 5-1 系统环境图

5.1.2 系统设计工具

本课题核心算法由 matlab 实现，软件界面由 matlab 自带的 GUI 设计实现。其中 matlab 使用 2014(b)版本，CNN 架构使用 matconvnet 的 beta22 版本，c 语言编译器使用 Microsoft Visual Studio2010.

5.1.3 系统功能需求

本系统功能描述如下：

图像选择：

用户进入页面后，点击浏览按钮，弹出文件选择框，用户选择其中的一个图像，点击确定。之后在指定位置显示用户所选择的图片，并且在编辑框中显示该图片的路径。

图像识别：

用户选择了图片之后，点击识别按钮，进行图像中的积分的识别与求解，之后将求解结果显示在指定位置。

图像选择：

用户选择所需要识别的图片的功能，具体描述如下：

用例描述：用户选择图像

执行者：用户

前置条件：无

后置条件：用户选择图片之后方可进行图片识别

基本路径：

1. 用户打开软件，进入主界面
2. 用户填写图片路径或者点击浏览按钮选择图片
3. 用户等待图片路径的显示以及所选图片的显示

图像识别：

识别和求解图像中的积分的功能，具体描述如下：

用来描述：图片的识别和求解

执行者：用户

前置条件：用户选择了图片并且点击识别按钮

后置条件：无

基本路径：

1. 用户点击识别按钮
2. 用户等待结果展示

5.1.4 系统非功能性需求

界面需求：

1. 页面内容：主题突出，术语和行文格式统一、规范、明确，栏目、菜单设置和布局合理，传递的信息准确、及时。内容丰富，文字准确，语句通顺；专用术语规范，行文格式统一规范。
2. 技术环境：页面大小适当，无错按钮或空按钮,采用 matlab 自带的 GUI 控制字体大小和版面布局。
3. 艺术风格：界面、版面形象清新悦目、布局合理,字号大小适宜、字体选择合理，前后一致，美观大方；色彩和谐自然,与主题内容相协调。

可靠性/可用性需求：

1. 用户输入为空或所属入图片不存在时，应进行报错并且停止本次求解。
2. 用户输入图片无法正常识别时，应当进行提示并且停止本次求解。

硬件需求：

1. 最小内存需求：128M
2. 最小存储空间需求：20G

软件需求：

1. win10 系统（64 位）
2. MATLAB R2014(a)
3. matconvnet-1.0-beta22
4. Visual Studio 2010

5.2 概要设计

5.2.1 界面设计

本系统共一个界面，界面中的控件设计如下：

1. 文本编辑控件：ID 为 edit2，文本编辑控件用来直接输入或者选择后显示文档路径，初始显示字符为“请输入图片路径”。

2. 按钮控件：ID 为 bushbutton1，该按钮用来选择要识别图片的路径，按钮上的显示字符为“浏览”。

3. 按钮控件：ID 为 bushbutton2，该按钮用来执行图像识别函数，进行积分图像的识别，按钮上的字符为“识别”。

4. 坐标轴控件：ID 为 axes1，当点击按钮 bushbutton1 时坐标轴控件用来显示所选择的要识别图片，点击按钮 bushbotton2 时坐标轴控件用来显示识别结果。

控件操作流程图如图 5-2

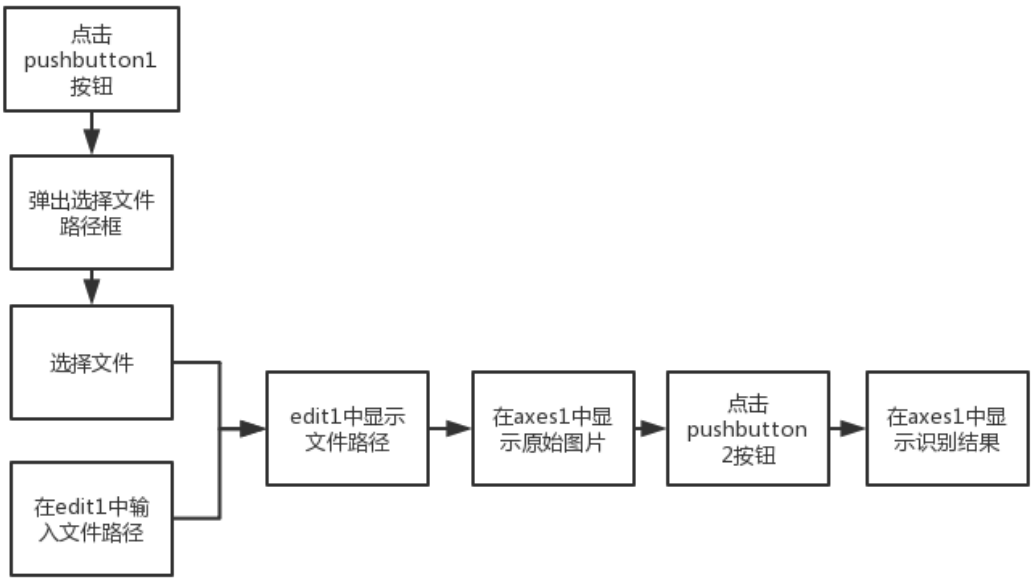

图 5-2 控件操作流程图

5.2.2 模块设计

按照功能分解，本系统共设计四个模块，分别为：图像预处理模块、图像识别模块、BST 建立模块、BST 分析模块。各模块之间关系如图 5-3。

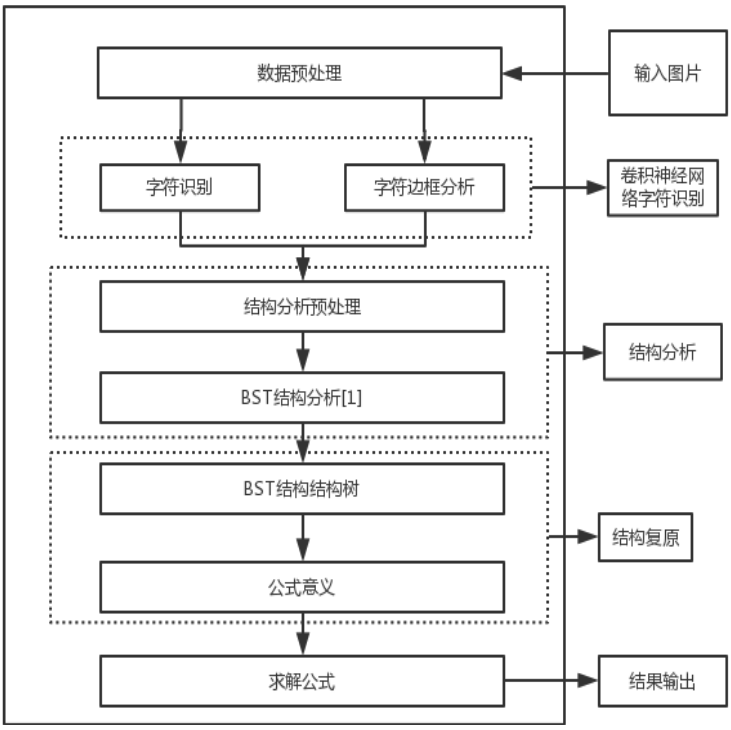

图 5-3 系统各个模块关系

本系统共涉及到三十余函数，其中各模块用到的主要函数如下，其余辅助函数以及 matlab 系统函数在详细设计中列出。

图像预处理模块：

AdaptiveThreshold：图像二值化

findinter：寻找连通域，得到连通域标记图像

图像识别模块：

cnn_plate_init：设计 CNN 网络及其参数

cnn_plate_setup_data：图像数据处理

cnn_plate：CNN 训练主函数

建立 BST 模块：

preparelist：建立字符节点表格

Construct：基线结构树分析，建立基线结构树

BST 分析模块：

merge_function：合并字符节点表格的函数

merge_number：合并字符节点表格的数字

analyse：语法分析函数

5.3 详细设计

5.3.1 图像预处理模块详细设计

图像预处理模块主要包括：图像的灰度化、图像的二值化、图像连通域分析等。具体函数如下：

im = imresize(im, ratio)

输入：原图像 im，缩小比例 ratio

输出：缩放后的图像 im

功能：将图像 im 缩小到高为 270 左右，宽缩小到相应比例 ratio，该函数主要是将过大的图片缩小，加快处理速度。

im2 = rgb2gray(im1)

输入：彩色图像 im1

输出：灰度图像 im2

功能：该函数主要是将彩色 RGB 图像 im1 变为灰度图像 im2，以便于下一步处理。

im2 = AdaptiveThreshold(im1)

输入：灰度图像 im1

输出：二值图像 im2

功能：该函数实现了局部自适应快速二值化方法，将输入的灰度图像 im1 变为二值化图像 im2。

im2 = cut_small(im1)

输入：原图片 im1

输出：裁剪后图片 im2

功能：裁去图像 im1 白边，只保留黑字部分 im2，提高下一步连通域分析的效率。

con = findinter(im)

输入：原图像 im

输出：标记了连通域的图像 con

功能：标记连通域算法，将输入图像 im 的每个连通域标记为 1,2,3…得到输出的标记好的图像 con。

list = part(con)

输入：标记好连通域的图像 con

输出：每个连通域的标号及其上下左右边界的表格 list

功能：连通域分割算法，逐步提取图像 con 标记的各个连通域，同时得到个连通域的上、下、左、右边界，最后输出表 list。

该模块输入为一幅图像，输出为一张表格包括了各个连通域的标记和上、下、左、右边界。本模块流程图如图 5-4。

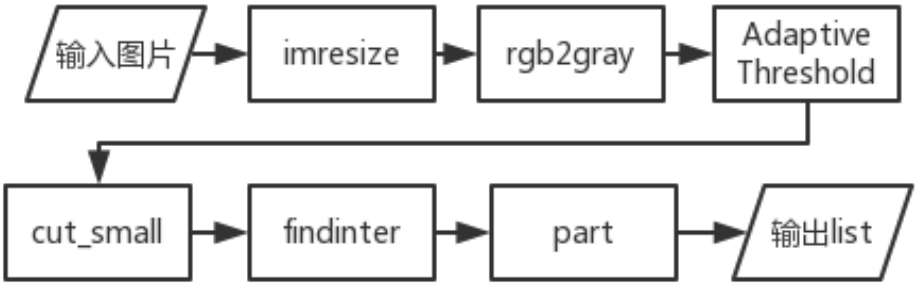

图 5-4 图像预处理模块流程图

5.3.2 图像识别模块详细设计

该模块主要是卷积神经网络的设计，其中包括要训练图像的预处理，网络结构的初始化，以及训练过程。具体设计如下：

cnn_plate_init()

功能：设计 CNN 网络及其参数，网络各个层以及参数参考 3.2.3 内容。

cnn_plate_setup_data()

功能：图像数据处理，将所有图像放入 28*28*1*n 的数组之内，n 为图像张数。

cnn_plate()

功能：CNN 训练过程。

5.3.3 BST 建立模块

本模块的输入是一张字符节点属性表，输出为一棵 BST 树。

list2 = preparelist(list1,con)

输入：函数 part 输出的表格 list1，标记了连通域的图像 con

输出：包括了每个连通域所有属性的表格 list2

功能：将预处理模块得出的 lis1t 以及图像识别模块得出的识别结果，建立一张新的表 list2，表中包括了表 4-1 中的所有属性。

list2 = correct(list1)

输入：preparelist 输出的表格 list1

输出：修正后的表格 list2

功能：修正 list1，识别不是完全正确的，有一定的差错，比如‘cos’有可能识别为‘(os’，于是在发现‘(os’就会修正为‘cos’，其余修正不做赘述。最后得到修正的表 list2。

buildBST()

功能：建立公式基线结构树的预处理以及调用 BST 的主函数 Construct。

list2 = SortSymbolByMinX(list1)

输入：correct 函数输出的表格 list1

输出：排序后的表格 list2

功能：将 list1 每个符号从左到右进行排序与重新编号得到新的表格 list2。

bst = Construct(rnode)

输入：域节点 rnode

输出：位于该域基线上的符号节点 bst

功能：输入域节点 rnode，构建公式基线结构树 bst。

list = Symbols(rnode)

输入：域节点 rnode

输出：该域中的符号节点表 list

功能：根据一个域节点 rnode 得到一张表 list，表中包括了域节点的所有子节点及其属性，并且按照从左到右的顺序编号。

[boollean,label] = Cover(snode,snode1)

输入：两个符号节点 snode 与 snode1

输出：布尔值 boollean 和域标签 label

功能：判断 snode1 是否属于分数线符号 snode 的上下域，返回结果 boollean，同时得到域标签 label。

snode = AddToRegion(label,snode1,snode)

输入：两个符号节点 snode 与 snode1，域标签 label

输出：更新的符号节点 snode

功能：将符号节点 snode1 添加到特定符号节点 snode 的特定域 label 中，更新 snode。

boollean = includes(snode, snode1)

输入：两个符号节点 snode 与 snode1

输出：布尔值 boollean

功能：判断某个符号 snode1 是否属于根号 snode 的 contains 域中。

[boollean, label] = IsNext(snode1,snode)

输入：两个符号节点 snode 和 snode1

输出：布尔值 boollean，域标签 label

功能：判断某两个符号 snode1 和 snode 是否水平相邻，如果不是水平相邻，则将 snode1 放入 snode 相应的域节点中，返回域标签 label。

5.3.4 BST 分析模块

str = analyse(list)

输入：符号节点表 list

输出：matlab 可识别的字符串 str

功能：语法分析主函数，输入 list，得到 matlab 可识别的字符串 str。

[list2, bst2] = refineBST(bst1,list1)

输入：原公式基线结构树 bst1，符号节点表 list1

输出：修正后公式基线结构树 bst2，符号节点表 list2

功能：修正已建立的 bst1 和 list1，包括合并函数，合并数字，删除多余的子节点，输出为 list2 和 bst2。

list2 = clear_toomuch_children(list1)

输入：原符号节点表 list1

输出：清理后的符号节点表 list2

功能：在上一模块中得到的符号节点表中，对于 a^{b^c}，c 和 b 都是 a 的孙子节点，此函数是为了使 c 只成为 b 的孙子节点，a 的孙子节点中不出现 c。输出为更新的符号节点表 list2。

list2 = merge_function(list1)

输入：原符号节点表 list1

输出：合并函数后的符号节点表 list2

功能：将类似于 cos、sin 等函数由单个字符合并为整个函数。输入 list1 更新后得到输出 list2。

list2 = merge_number(list1)

输入：原符号节点表 list1

输出：合并数字后的符号节点表 list2

功能：将单个数字字符如’1’、’5’合并为’1.5’, 输入 list1 更新后得到输出 list2。

本模块输入为一棵 bst 树，输出为可以被 matlab 识别的字符串 str，具体流程图见图 5-5。

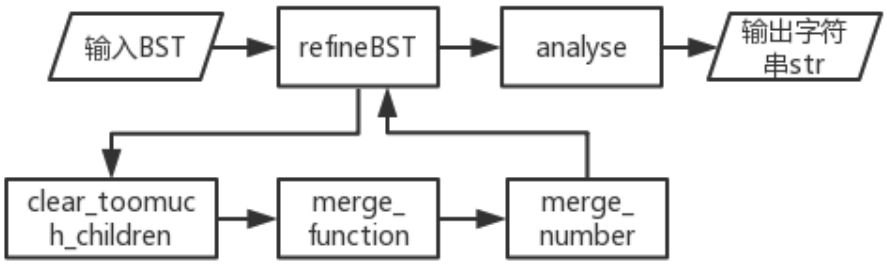

图 5-5 BST 分析模块流程图

5.4 本章小结

本章内容为积分图像识别系统的设计与实现，分别讲述了系统的需求分析、概要设计以及详细设计。在概要设计中展示了核心的 10 个函数，在详细设计中，详细展示了各个模块中的函数，以及部分模块流程图。