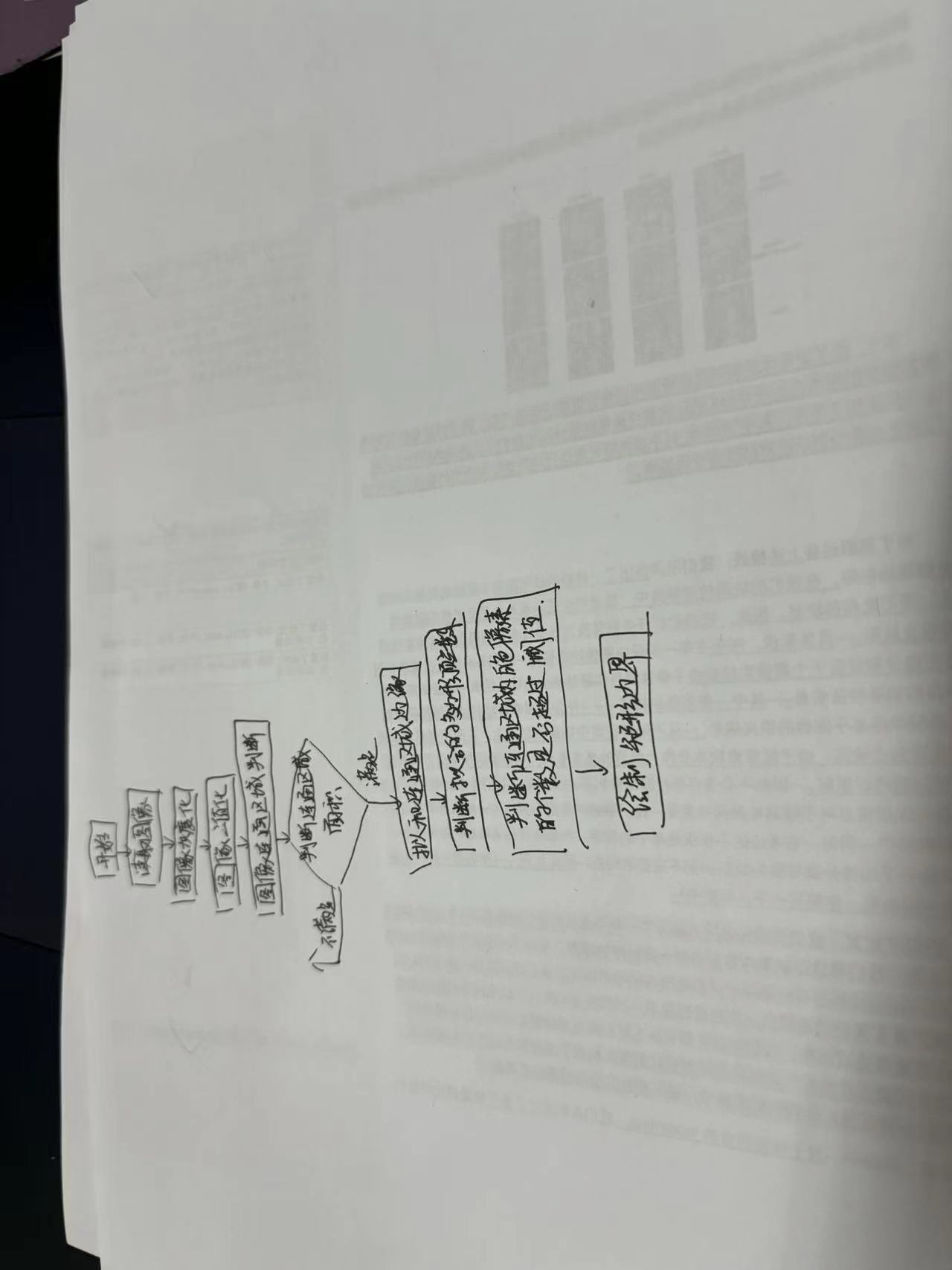
**设计方案：**

先将图像灰度化后并二值化，通过综合考虑多种情况确定不使用自适应二值化而是固定阈值二值化。提取二值化图像中所有的连通区域，计算每个连通区域的面积，小于最小阈值或大于最大阈值判定不符合要求完成初筛。由于填涂信息的形状不为固定的矩形，为了提升程序的泛化性。利用填涂信息的结构信息，将剩余的每个连通区域使用approxPolyDP拟合成多边形轮廓，并判断多边形的顶点个数，用于排除汉字和数字对于结果的部分影响；利用矩形的形态信息，使用boundingRect将连通区域拟合成矩形并计算其坐标和宽高，由于填涂信息和定位矩形框均满足宽>高且宽至少为20像素的条件，因为在此条件的约束下，继续筛出部分汉字和数字。在上述基础上仍有部分汉字或数字和矩形框混淆，我们利用“填涂矩形内均为黑色，而数字（或汉字）被标注区域不全是黑色”这一条件，判断连通区域内白色像素点的个数，当小于阈值时触发条件，判定为填涂信息。

**流程图：**

，

**算法原理：**

1. **图像预处理**：
   * 将输入图像读取为OpenCV中的图像对象。
   * 转换图像为灰度图，通过**cv.cvtColor**函数。
   * 使用自适应二值化算法，通过**cv.threshold**函数，生成二值化图像**thresh2**。
2. **轮廓检测**：
   * 使用**cv.findContours**函数寻找二值化后图像中的轮廓。
   * 对找到的轮廓进行排序，按照轮廓面积降序排列。
3. **轮廓过滤**：
   * 遍历轮廓列表，过滤掉面积小于**min\_area**或大于**max\_area**的轮廓。
   * 使用**cv.approxPolyDP**对每个轮廓进行多边形逼近，减少顶点数。
   * 进一步筛选轮廓，保留顶点数小于等于**approx\_num**的轮廓。
4. **矩形框判定**：
   * 对通过前述过滤的轮廓，使用**cv.boundingRect**获取矩形框的坐标和尺寸。
   * 进一步判断矩形框是否为横向矩形（宽度大于高度）以及宽度是否大于等于20。
   * 在符合条件的矩形框中，提取矩形框中白色像素的个数，计算白色像素占总像素的比例。
5. **结果筛选**：
   * 根据设定的阈值**pw**，筛选掉白色像素比例大于阈值的矩形框。
   * 根据设定的**top**值，筛选掉y坐标小于**top**的矩形框。
6. **结果输出**：
   * 将通过筛选的矩形框按y坐标从上到下排序，通过**contours.sort\_contours**函数。
   * 将排序后的矩形框在原图上用红色标记，并保存输出图像。