网页工商信息图片文字提取

开发文档

目录

1 引言

1.1 编写目的

根据国家工商总局《网络交易管理办法》要求对网店营业执照信息进行公示，天猫网店经营者营业执照信息会在天猫店铺上以图片形式进行公示，但一方面图片存储占用空间巨大，并且不方便查找等操作，另一方面图片信息不能进行结构化处理，所以需要提取出图片中的企业注册号、企业名称形成结构化文档。

编程人员可以利用这份开发文档进行程序的编制工作，同时也是测试人员进行测试的依据，也可以供客户参考使用。

1.2 背景

本应用系统是针对网页工商信息图片文字提取而研发的系统。内容涉及图片文字提取、预处理、识别等等。使用本应用系统可以将一系列工商信息图片形成结构化文档。

1.3 定义

名词	定义
网页工商信息图片	指天猫网店经营者营业执照
标准字库	指非图片截取，而是通过程序生成的微软雅黑字体的所有中文、数字和大写字母
图片二值化	指把图片从彩色图转成二值图
图片边框检测	指把图片四周的白边去掉
图片规格化	指把图片从原来的尺寸变成 64*64 尺寸
特征序列	指通过求解文字、数字和字母得出的一系列特征

1.4 参考资料

1. 李润之. 基于 Linux 平台的图片文字识别系统[D]. 吉林大学, 2016.
2. 倪桂博. 印刷体文字识别的研究[D]. 华北电力大学(河北), 2008.
3. 任金昌, 赵荣椿, 张炜. 一种快速有效的印刷体文字识别算法[J]. 中国图象图形学报, 2001, 6(10):1011-1015.
4. 陈义文, 范平. 批量图片的文字识别系统设计与实现[J]. 湖北科技学院学报, 2016, 36(s1):141-143.
5. 梁涌. 印刷体汉字识别系统的研究与实现[D]. 西北工业大学, 2006.
6. 李俊. 印刷体文字识别系统的研究与实现[D]. 电子科技大学, 2011.

2 总体设计

2.1 需求概述

根据比赛详细要求和实际情况考虑，本应用系统要实现的需求主要有以下几点：

- 程序能够自动读取企业工商信息图片所在的文件夹路径。
- 从图片文件夹路径中顺序取出图片进行识别。
- 因为天猫平台公示的图片内容没有固定格式，所以需要程序能匹配不同格式的图片内容提取信息。
- 能够提取出图片中的企业注册号、企业名称数据项，并保存进 Excel 中，因为企业注册号、企业名称数据项要进行分析处理，所以需要保证提取信息的准确性，识别准确率需要保证在 95%以上。
- 最终的识别结果以一份汇总的 Excel 交付。

与此同时，系统需要最大限度地实现易维护性，易操作性，运行稳定和安全可靠。

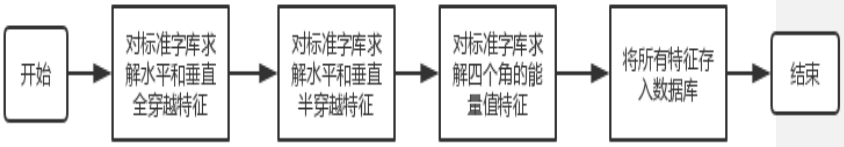
2.2 总体处理流程设计

本系统的的总体设计主要分为两部分，一是实现流程设计，二是数据流程设计。在实现流程设计中，主要分两部分，分别是用生成标准字库，并用标准字库求解出特征序列，将其存入数据库。另一部分是从工商信息图片中提取出企业注册号和企业名称，然后求解每一个单元图片的特征，在数据库中匹配找出对应文字。最终，将对应信息填入表格中。

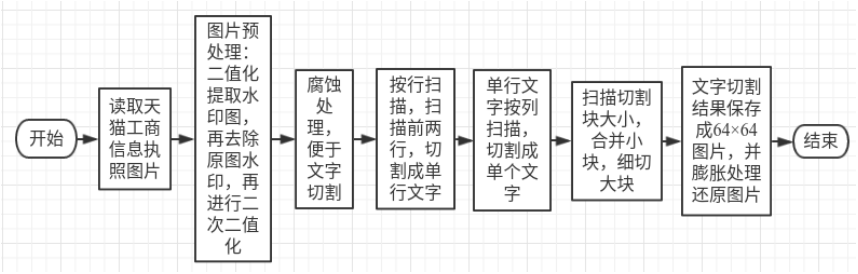
总体处理流程设计主要分为以下四步：

- 生成标准字库

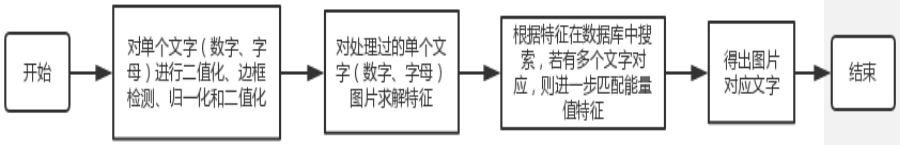
- 对标准字库求解特征



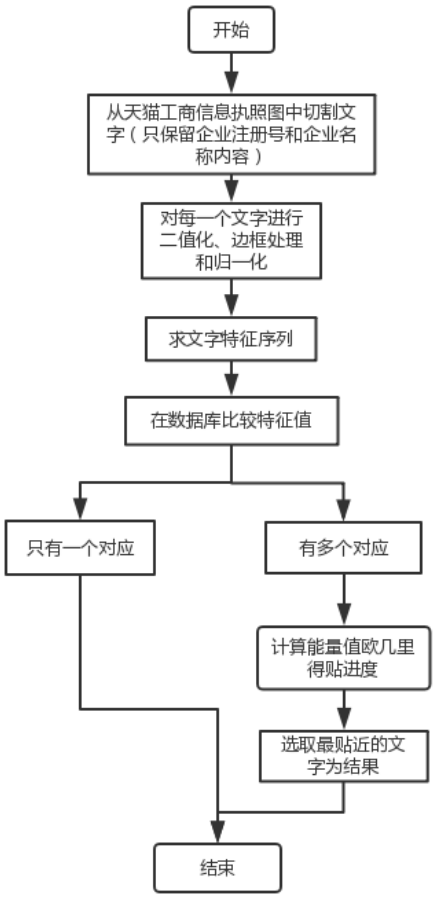
- 从图片中提取注册号和名称



- 图片文字识别



- 总体图片文字识别流程



3 算法设计

3.1 求文字特征值算法

笔画密度特征是从分析汉字本身的拓扑结构入手，对不同的汉字，它们的拓扑结构是不同的。如从不同方向扫描汉字图像，笔画相交的次数随着汉字的不同而有所区别。对已归一化的 64x64 汉字点阵图像，向不同方向扫描该图像，对各方向扫描线横切文字笔画的次数做累加计算，形成汉字笔画直方图，从而得到特

征向量。本设计从水平、垂直两个扫描方向来获取汉字点阵图像的比划密度特征。除此以外，还计算了文字的四个角的能量值密度，用于进一步校对。

a) 水平全穿越

图 1 所示：分别从水平 1/4、1/2、3/4 处扫描图像，计算越笔画的次数。由图 1 可见，穿越的次数从左到右分别为 2、4、3。存入数组 HA 中，记为：HA=(2,4,3)。

图 1：水平全穿越

b) 垂直全穿越

同理，分别从垂直方向的 1/4、1/2、1/3 处扫描图像，计算穿越笔画的次数。把穿越次数存入到数组 VA 中，记为 VA=(3,3,3)。如图 2 所示：

图 2：垂直全穿越

c) 水平半穿越

HA_H=(HUL,HDR,HUR,HDL)，HUL:左上的穿越次数，HDR：右下的穿越次数，HUR:右上的穿越次数，HDL：右下的穿越次数。如图 3 所示。

图 3：水平半穿越

d) 垂直半穿越

VA_H=(VUL,VDR,VUR,VUL), VUL：左上的穿越次数,VDR：右下的穿越次数,VUR:右上的穿越次数, VUL：右下的穿越次数。如图 4 所示。

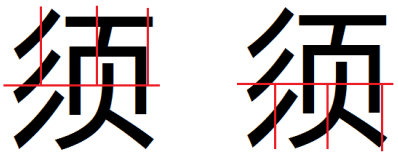


图 4：垂直半穿越

e) 能量值密度

能量值密度是指在 0-1 矩阵中 1 的值在各自的区域内所占的比例。即：

$$T = (t_{ij}),\ t_{ij} = 0\ or\ 1,\ i,j = 1...64,\ T_1 = (t_{ij}),\ t_{ij} = 0\ or\ 1,\ i,j = 1...16$$

A 为 T_1 中 1 的个数，B 为 T_1 中总的像素个数，那么 x=A/B 就为左上角的能量值密度。如图 5 所示：

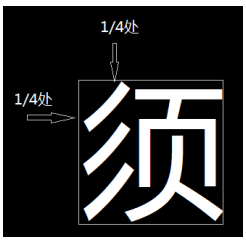


图 5：左上角能量值密度

3.2 双线性插值算法

假设源图像大小为 mxn，目标图像为 axb。那么两幅图像的边长比分别为：m/a 和 n/b。注意，通常这个比例不是整数，编程存储的时候要用浮点型。目标图像的第（i,j）个像素点（i 行 j 列）可以通过边长比对应回源图像。其对应坐标为（i*m/a,j*n/b）。

显然，这个对应坐标一般来说不是整数，而非整数的坐标是无法在图像这种离散数据上使用的。双线性插值通过寻找距离这个对应坐标最近的四个像素点，来计算该点的值（灰度值或者 RGB 值）。如果你的对应坐标是（2.5,4.5），那么最近的四个像素是（2，4）、（2，5）、（3，4），（3，5）。

若图像为灰度图像，那么（i，j）点的灰度值可以通过一下公式计算：

$$f(i,j)=w1*p1+w2*p2+w3*p3+w4*p4$$

其中，pi(i=1,2,3,4)为最近的四个像素点，wi(i=1,2,3,4)为各点相应权值。

4 数据结构设计

4.1 逻辑结构设计

本项目共有两个数据表，一个用来存储汉字的特征值，另一个用来存储数字和字母的特征值，两个数据表间没有逻辑结构关系：

汉字表：

fonts(char_chi,feature1,feature2,feature3,feature4,feature5)

数字字母表：

num_fonts(char_chi,feature1,feature2,feature3,feature4,feature5)

4.2 物理结构设计

汉字表 fonts：

字段名	字段含义	类型	长度
char_chi	具体某一汉字	VARCHAR	5
feature1	汉字对应的穿越值组成的字符串	VARCHAR	40
feature2	汉字对应的左上角能量值密度	VARCHAR	20
feature3	汉字对应的右上角能量值密度	VARCHAR	20
feature4	汉字对应的左下角能量值密度	VARCHAR	20
feature5	汉字对应的右下角能量值密度	VARCHAR	20

数字字母表 num_fonts:

字段名	字段含义	类型	长度
char_chi	具体某一数字或字母	VARCHAR	5
feature1	对应的穿越值组成的字符串	VARCHAR	40
feature2	对应的左上角能量值密度	VARCHAR	20
feature3	对应的右上角能量值密度	VARCHAR	20
feature4	对应的左下角能量值密度	VARCHAR	20
feature5	对应的右下角能量值密度	VARCHAR	20

5 接口设计

5.1 外部接口

5.1.1 文件资源管理器接口

导入待识别的图片。用户将需要识别的图片放入一个文件夹，程序调用资源管理器，用户选择文件夹，导入图片。执行后续的识别程序。

保存最终交付的 excel 文件。调用资源管理器，用户选择保存位置。

5.1.2 导出识别结果至 excel

将从图片中提取的信息格式化，生成 excel 文件交付。

5.2 内部接口

5.2.1 数据库接口

对识别的文字特征进行数据库特征匹配，调用数据库接口进行遍历搜索匹配。

5.2.2 Opencv2 接口

安装 Opencv2，python 中调用接口。

6 环境配置

开发所在系统：LINUX、WINDOWS
开发环境：python 2.7、cv2 3.1、MySQL 5.6

7 程序描述

7.1 生成标准字库模块

7.1.1 功能描述

这一模块主要是用程序生成所有汉子、字母和数字的微软雅黑格式的图片，用以求解标准字库的特征进入存入数据库中。

7.1.2 代码实现

```
def gen_chinese():
    pygame.init()
    start, end = (0x0041, 0x005a)  # 大写字母编码范围
    for codepoint in range(int(start), int(end)):
        word = unichr(codepoint)
        font = pygame.font.Font("微软 vista 雅黑.ttf", 64)
        #  当前目录下要有微软雅黑的字体文件 msyh.ttc,或者去 c:\Windows\Fonts 目录
下找
        # 64 是生成汉字的字体大小
        rtext = font.render(word, True, (0, 0, 0), (255, 255, 255))
        pygame.image.save(rtext, os.path.join('english', word + ".png"))
```

7.2 对文字（数字、字母）图片预处理模块

7.2.1 功能描述

这一模块主要功能是对标准字库（中文、数字和字母）进行预处理。预处理的步骤包括对图片进行二值化，对图片进行边框检测，对图片进行归一化。其中，边框检测的做法是搜索最左、最右、最上和最下的像素，然后截取图片；归一化的做法是采用双线性插值算法。整体实现调用了 python 的 cv2 库，借助调用其方法帮助我们实现功能。

7.2.2 代码实现

（1） 对图片进行二值化

```
def bin_chinese(pathin, pathout):
```

```
    '''
    对图片进行二值化
    :return:
    '''
    pictures = os.listdir(pathin)
    for picture in pictures:
        pic_path = os.path.join(pathin, picture)
        # 以彩色图方式读取图片
        im = cv2.imread(pic_path)
        # 将图片转成灰度图
        im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        # 二值化
        retval, im_at_fixed = cv2.threshold(im_gray, 127, 255, cv2.THRESH_BINARY)
        # 保存图片
        pic_path = os.path.join(pathout, picture)
        cv2.imwrite(pic_path, im_at_fixed)
```

（2） 对图片进行边框检测

```
def bor_chinese(pathin, pathout):
    '''
    对二值化后的图片进行边框提取
    :return:
    '''
    posup = 0
    posdown = 0
    posleft = 0
    posright = 0
    flag = 0

    pictures = os.listdir(pathin)
    for picture in pictures:
        pic_path = os.path.join(pathin, picture)
        # 以灰度图方式读取图片
        im = cv2.imread(pic_path, 0)
        width = len(im[0])
        height = len(im)
        # 找 posup
        flag = 0
        for i in range(height):
            for j in range(width):
                if im[i][j] == 0:
                    posup = i
                    flag = 1
                    break
```

```
            if flag == 1:
                break

        # 找 posdown
        flag = 0
        for i in range(height - 1, 0, -1):
            for j in range(width):
                if im[i][j] == 0:
                    posdown = i
                    flag = 1
                    break
            if flag == 1:
                break

        # 找 posleft
        flag = 0
        for i in range(width):
            for j in range(height):
                if im[j][i] == 0:
                    posleft = i
                    flag = 1
                    break
            if flag == 1:
                break

        # 找 posright
        flag = 0
        for i in range(width - 1, 0, -1):
            for j in range(height):
                if im[j][i] == 0:
                    posright = i
                    flag = 1
                    break
            if flag == 1:
                break

        # 裁剪图片
        im2 = im[posup:posdown, posleft:posright]

        # 保存图片
        pic_path = os.path.join(pathout, picture)
        cv2.imwrite(pic_path, im2)

    print '边框检测成功!'
```

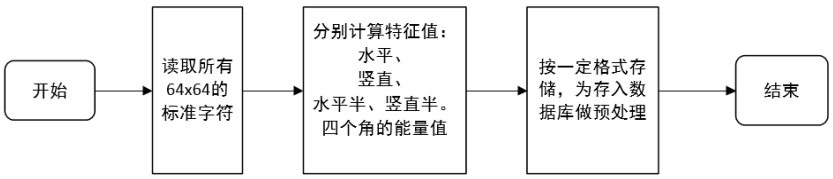
（3） 对图片进行归一化

```
def nor_chinese(pathin, pathout):
    '''
    对边框检测后的文字进行归一化
    :param pathin:
    :param pathout:
    :return:
    '''
    # 先调整文字比例
    # 高度填充
    pictures = os.listdir(pathin)
    for picture in pictures:
        pic_path = os.path.join(pathin, picture)
        # 以灰度图方式读取图片
        im = cv2.imread(pic_path, 0)
        width = len(im[0])
        height = len(im)
        h_ratio = width / height * 1.0
        w_ratio = height / width * 1.0

        # 高度填充
        if h_ratio > 1.25:
            Y = (width - height) / 2
            im = cv2.copyMakeBorder(im, Y, Y, 0, 0, cv2.BORDER_CONSTANT,
value=[255, 255, 255])

        # 宽度填充
        if w_ratio > 1.25:
            U = (height - width) / 2
            im = cv2.copyMakeBorder(im, 0, 0, U, U, cv2.BORDER_CONSTANT,
value=[255, 255, 255])

        # 采用双线性插值进行大小归一化
        im = cv2.resize(im, (64, 64))

        # 保存图片
        pic_path = os.path.join(pathout, picture)
        cv2.imwrite(pic_path, im)

    print '归一化成功!'
```

7.2.3 效果展示

7.3 对文字（数字、字母）图片求特征模块

7.3.1 功能流程图



7.3.2 功能描述

使用预处理得到的结果，计算所有标准字符的特征值和角能量值，按一定的格式存储，为存入数据库作准备。具体操作为读取预处理后的所有标准字符的图片，遍历字符图片，按 2.2 提到的求解特征方法，提取标准字符的特征值和能量值。整体实现调用了 python 的 cv2 库，借助调用其方法帮助我们实现功能。

7.3.3 代码实现

(1) 所有标准字符特征提取

```
def caculate_all(pathin):
    # 对 64x64 的字体进行特征提取
    pictures = os.listdir(pathin)
    for picture in pictures:
        HA = []   # 水平穿越的特征值
        VA = []   # 竖直穿越的特征值
        HA_H = []   # 水平半穿越的特征值
```

```
VA_H = []    # 竖直半穿越的特征值
cor_feature = []    # 四个角的能量值密度，左上，左下，右上，右下

pic_path = os.path.join(pathin, picture)
#    print pic_path
im = cv.imread(pic_path, 0)
# 水平全穿越
HA.append(caculate_line(im[16]))
HA.append(caculate_line(im[32]))
HA.append(caculate_line(im[48]))

# 竖直全穿越
VA.append(caculate_line(im[16]))
VA.append(caculate_line(im[32]))
VA.append(caculate_line(im[48]))
# 水平半穿越
x = im[16]
x1 = x[0:32]
HA_H.append(caculate_line(x1))
x2 = x[32:64]
HA_H.append(caculate_line(x2))

x = im[48]
x1 = x[0:32]
HA_H.append(caculate_line(x1))
x2 = x[32:64]
HA_H.append(caculate_line(x2))

# 竖直半穿越
y = [y[16] for y in im]    # 获取列

y1 = y[0:32]
VA_H.append(caculate_line(y1))
y2 = y[32:64]
VA_H.append(caculate_line(y2))

y = [y[48] for y in im]

y1 = y[0:32]
VA_H.append(caculate_line(y1))
y2 = y[32:64]
VA_H.append(caculate_line(y2))

# 测试打印特征值
```

```
            print pic_path
            print HA
            print VA
            print HA_H
            print VA_H
            #
            # 计算左上角的能量值
            cor_feature.append(cor_caculate(im, 0, 16, 0, 16))
            # 计算左下角的能量值
            cor_feature.append(cor_caculate(im, 48, 64, 0, 16))
            # 计算右上角的能量值
            cor_feature.append(cor_caculate(im, 0, 16, 48, 64))
            # 计算右下角的能量值
            cor_feature.append(cor_caculate(im, 48, 64, 48, 64))
            print cor_feature

            # 按格式写入文件
            name, ext = os.path.splitext(pic_path)
            name = os.path.basename(name)
        write_to_txt(name, HA, VA, HA_H, VA_H, cor_feature)
```

(2) 计算单个字符的某行/列特征值

```
def caculate_line(line):
    # 计算该行或者列的特征值
    size = len(line)
    limit = 0    # 有无的界限
    k = 0    # 该行的特征值
    tag = 1
    for i in range(size):
        if line[i] == limit and tag == 1:
            k += 1
            tag = 0
        elif line[i] > limit:
            tag = 1
return k
```

(3) 计算能量值

```
def cor_caculate(image, t_row, d_row, t_col, d_col):
    '''
    :param image: 输入图片
    :param t_row: 开始行
    :param d_row: 结束行
    :param t_col: 开始列
    :param d_col: 结束列
```

```
    :return: 能量值
    '''
    count = 0
    for i in range(t_row, d_row):
        for j in range(t_col, d_col):
            if image[i][j] == 0:    # 黑
                count += 1
    # print count * 1.0 / 256
    return count * 1.0 / 256
```

(4) 按格式将特征存入文件

```
def write_to_txt(ch_name, HA, VA, HA_H, VA_H, cor_feature):
    try:
        fp = open("eng_value.txt", "a+")
        fp.write(ch_name)
        fp.write(",")

        for item in HA:
            fp.write(str(item) + " ")
        fp.write(",")

        for item in VA:
            fp.write(str(item) + " ")
        fp.write(",")

        for item in HA_H:
            fp.write(str(item) + " ")
        fp.write(",")

        for item in VA_H:
            fp.write(str(item) + " ")
        fp.write(",")

        for item in cor_feature:
            fp.write(str(item) + " ")

        fp.write("\n")
        fp.close()

    except IOError:
        print("fail to open value.txt")
```

7.4 对营业执照图片处理模块

7.4.1 功能流程及描述

（1）文件流读取根目录下的天猫工商信息执照图片，并转换为灰度图；
（2）对灰度图二值化处理，分离出水印图，并对水印图做二值化处理；
（3）去水印处理：按水印图值为 0（即为黑色）的点将对应灰度图的点值设为 0，即完成去水印处理；
（4）腐蚀处理：由于原图字体间的间隔较小，大部分字是连在一起的，中间没有间隔。腐蚀处理可以尽可能将各个字分离开来，利于切割单字；
（5）按行扫描：如果某一行全为 255（即为白色），则该行为某行文字的起始或结束位置；
（6）按列扫描：同理，如果某列全为 255（即为白色），则该列为某个文字的起始或结束位置；
（7）由于部分文字间没有间隔，或者某个字是有两部分组成，会使文字切割出现几个文字切成一块，或者单个文字被切成若干块的情况，因此需要进行细切或合并处理。将切割块的宽度大小进行排序，选择中间块作为参照标准，大于该块的细切，小于该块的则寻找附近的小块进行合并，若小块的附近都不符合小块的条件，则该块独立（可能是标点符号等）；
（8）合并细切结果保存，进行膨胀操作还原图片，并放大为 64×64 大小，便于后续处理。

7.4.2 代码实现

（1）读取图片

```
for fileName in os.listdir(pic_dir):
        src = cv2.imread(pic_dir+fileName,cv2.CV_LOAD_IMAGE_UNCHANGED)
```

（2）获取水印图+去水印+腐蚀处理

```
def Qushuiyin(grayimage,height,width):
    #获取水印图
    ret,thresh=cv2.threshold(grayimage,220,255,cv2.THRESH_BINARY_INV)
    #去水印
    for i in range(0,height):
```

```
            for j in range(0,width):
                if thresh[i,j]<255:
                    grayimage[i,j]=0
    #腐蚀
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(2, 2))
    grayimage = cv2.erode(grayimage,kernel)
    return grayimage
```

（3）按行扫描

```
def cutlines(thresh,linebegin,height):
    fh=lh=0
    first=False
    last=False
    for i in range(linebegin,height):
        if first and last:
            break;
        for j in range(0,100):
            if thresh[i,j]==0:
                if not first:
                    fh=i
                    first=True
                    break;
                elif first and not last:
                    break;
            elif first and j==99:
                lh=i
                last=True
    return (fh,lh)
```

（4）按列扫描

```
def cutfonts(line,fh,lh,width):
    lines=[]
    length=[]
    lensum=0
    lineh=lh-fh+2
    first=False
    last=False
    #font=0

    #print(lineh)
    #print("the length of cut result:")
    for i in range(0,width):
```

```
        for j in range(0,lineh):
            if line[j,i]==0:
                if not first:
                    fw=i
                    first=True
                    break
                elif first and not last:
                    break
            elif first and j==lineh-1:
                lw=i
                last=True
        if first and last:
            first=False
            last=False
            if fw==lw:
                continue
            lines.append((fw,lw))
            length.append(lw-fw)
            lensum=lensum+(lw-fw)
            #print(fw,lw)
    return (lines,length,lensum)
```

（5）合并细切

```
def hebing(lines,length,lensum,divide):
    subsum=0
    for l in range(0,divide):
        subsum=subsum+length[l]
    #print("subsum",subsum,lensum,(lensum-subsum)/(len(length)-divide))

    divide=divide-1

    sublength=length[:divide]
    sublength.sort()

    #前后两部分分别切分或合并
    result=[]
    l=0
    while l<len(length):
        if l==0:
            #avglen=subsum/(divide+1)
            avglen=sublength[divide/2]
            #print(avglen)
        elif l==divide:
```

```
            #avglen=(lensum-subsum)/(len(length)-divide-1)
            sublength=length[divide+1:]
            sublength.sort()
            print(len(length),len(sublength),divide)
            avglen=sublength[len(sublength)-1-len(sublength)/4]
            (positionb,positione)=lines[l]
            #print(avglen)
            l=l+1
            continue
        #块比较小
        if abs(length[l]-avglen)>avglen/3 and length[l]<avglen:
            begin=l
            sublen=length[l]
            #while l<len(length) and abs(length[l]-avglen)>avglen/3 and length[l]<avglen:
            #while l<len(length)-1 and abs(sublen-avglen)>avglen/3:
            while l<len(length)-1 and sublen+length[l+1]<avglen:
                l=l+1
                sublen=sublen+length[l]

            #后面有小的块，合并起来
            if l-begin>0:
                (fa,la)=lines[begin]
                (fb,lb)=lines[l]
                result.append((fa,lb))
            #块小但也没有很小，附近也没有小的块，保存
            else:
                result.append(lines[l])
        #块比较大
        elif abs(length[l]-avglen)>avglen/3 and length[l]>avglen:
            subcount=math.floor(float(length[l])/float(avglen))
            if subcount>0:
                sublen=int(math.floor(length[l]/subcount))
            #print("cut",sublen)
            (fa,la)=lines[l]
            for sub in range(0,int(subcount)):
                result.append((fa,fa+sublen))
                fa=fa+sublen
        #块不大不小直接保存
        else:
            result.append(lines[l])
        l=l+1
    return result,positionb
```

（6）结果处理

```
for l in range(len(result)):
    (fa,fb)=result[l]
    if fa<pb:
        continue
    subline=line[0:lh-fh,fa:fb]
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(2, 2))
    subline=cv2.erode(subline,kernel)
    subline=bor_chinese(subline)
    if linenum==1 and fb-fa<10:
        cv2.imwrite(save_dir+str(font)+".png",subline)
    else:
        subline=cv2.resize(subline,(64,64),interpolation=cv2.INTER_AREA)
        cv2.imwrite(save_dir+str(font)+".png",subline)
    font=font+1
```

7.5 文字匹配模块

7.5.1 功能描述

批注 [p1]: 燕芝负责

7.5.2 代码实现

8 测试设计

8.1 测试环境和方法

开发所在系统：LINUX、WINDOWS
开发环境：python 2.7、cv2 3.1、MySQL 5.6
测试方法：黑盒测试

8.2 测试用例

7.2.1 总体功能测试：提取工商信息图片的信息，并交付 excel 文档
7.2.2 模块功能测试：
（1） 生成标准字库
（2） 由标准字库求解出特征序列，并存入数据库。

（3） 提取工商图片的文字特征，在数据库中匹配找出对应文字。
（4） 将对应信息生成 excel 文档。