基于 Tesseract 文字识别的预处理研究

章 安[1],马明栋[2]
(1. 南京邮电大学 通信与信息工程学院,江苏 南京 210003;
2. 南京邮电大学 地理与生物信息学院,江苏 南京 210003)

摘 要:针对 Tesseract 文字识别框架对输入图像的像素要求,以及图像采集过程中可能出现的歪斜、黑边等情况,基于文字识别流程,对预处理阶段的二值化、缩放、边框处理与倾斜矫正进行研究与 C++代码的实现。对文字识别 OCR(optical character recognition,光学字符识别)的流程进行了概述,重点研究图像缩放与二值化过程,利用双线性插值算法逐像素、逐行分别对横纵坐标进行线性插值,完成图像缩放;利用最大类间方差法、聚类的思想,遍历灰度值,获取最佳二值化阈值,实现图像的二值化。参考 OpenCV 库函数,提出图像边框与偏移的处理思路。在 VS2015 环境下基于 Tesseract 框架,对整个流程进行实现,介绍了 Tesseract 框架的接口与功能、输入与输出参数。图像的预处理对文字识别必不可少,有利于 Tesseract 之后的识别工作。

关键词:OCR;文字识别;预处理;Tesseract 框架;C++
中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2021)01-0073-04
doi:10.3969/j.issn.1673-629X.2021.01.013

Research on Preprocessing Based on Tesseract Text Recognition

ZHANG An[1], MA Ming-dong[2]
(1. School of Telecommunications & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China;
2. School of Geographical and Biological Information, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: According to the pixel requirements of the input image of the Tesseract text recognition framework, as well as the skew and black edges that may occur in the image acquisition process, based on the text recognition process, the binarization, scaling, border processing and tilt correction in the preprocess are researched and implemented in C++ code. The process of OCR (optical character recognition) is summarized, focusing on the process of image scaling and binarization. The bilinear interpolation algorithm is used to linearly interpolate the horizontal and vertical coordinates pixel by pixel and line by line so as to complete image scaling. According to idea of maximum inter-class variance method and clustering, the gray value is traversed to obtain the optimal binarization threshold to achieve the binarization of the image. With reference to the OpenCV library function, the image frame and offset processing ideas are proposed. Based on the Tesseract framework in VS2015, the entire process is implemented, and the interfaces and functions of the Tesseract framework, input and output parameters are introduced. Image preprocessing is essential for text recognition, which is beneficial to the recognition work after Tesseract.

Key words: OCR; text recognition; preprocessing; Tesseract framework; C++

0 引 言

1929 年德国科学家 Tausheck 提出了 OCR 的概念[1],通过对图像像素点的分割,与字符库对照,获取最近似的计算机文字。最初的 OCR 技术目的在于信息化处理大量的印刷品,例如报刊杂志、文件资料和其他文字材料,进而开始了西文 OCR 技术的研究,以便代替人工键盘输入[2]。随着时间的推移,科技公司的不懈努力以及计算机技术的飞速发展,西文 OCR 技术现已广泛应用于各个领域,实现了“电子化”信息处理。

收稿日期:2020-03-13 **修回日期**:2020-07-15
基金项目:江苏省自然科学基金-青年基金项目(BK20140868)
作者简介:章 安(1996-),男,硕士研究生,CCF 会员(B7765G),研究方向为图像处理;马明栋,博士,教授,研究方向为地理信息系统平台软件设计与开发等。

国内对于印刷体汉字识别的研究从 20 世纪 70 年代末起步[3],至今已有近四十年的发展历史。从最开始对数字、英文、符号识别的研究,到对汉字识别进行了探索,80 年代后期,大量研究人员钻研其中,汉字识别技术也趋于实用化[4]。如今市场上的 OCR 识别平台层出不穷,最常见的百度、华为和腾讯等公司都提供了文字识别云平台,在所需的环境之下,调用其提供的在线接口,便可以实现文字的识别。

Tesseract 是一个开源 OCR 库,源码存在于 github 上。基于其开源的特性,可以添加接口,丰富其功能。支持 Tesseract 的 Leptonica 组件有着优越的图像分析性能,保证了文字识别的精度。除此之外,Tesseract 还具有多平台的可移植性,拥有庞大的 Unicode 字符识别库。

识别是智能的基础[5],目前互联网技术迅速发展,在 AI(artificial intelligence,人工智能)技术的需求之下,OCR 技术成为了不可忽略的部分,通过文字的识别,提高了各行各业的生产效率。

1 文字识别流程

文字识别通常有两种形式,手写体与印刷体[6]。对于印刷体而言,研究结果相对成熟,而手写体更为复杂,该文的讨论范围仅在于印刷体的文字识别。

文字识别通常包括以下流程:

(1)预处理:预处理操作在图像识别之前。已采集的图像质量受多方面因素影响,图像的亮暗、印刷体质量、污点或阴影都是干扰识别的因素。预处理主要是对图像进行处理,将图像灰度化以及二值化,进而对图像的倾斜、边框等问题进行处理,使得文字规范,图像平滑,从而有利于接下来的处理操作[7]。

(2)版面分析:版面分析主要处理文本图像的不同部分,分析图像中不同类型文本,例如标题或是正文,插图或是表格,从而获取文章逻辑结构,包括各区域的逻辑属性、文章的层次关系和阅读顺序。最后根据版面分析和文字的结果,进行版面重构。

(3)图像切分:识别是单个文字的识别,因而必须对图像进行行列切分。印刷体文字图像行列间距、字间距大致相等,简化了处理方法。

(4)特征提取与模型训练:通过深度学习框架,对文字识别进行特征提取和模型训练,提高了识别的有效性和可靠性。

(5)识别后处理:识别后处理通常包括版面的重构和识别的校正,对原图像版面的恢复以及不同语言模型的校正[8]。

通过以上流程,计算机通过输入的图像,分析并识别文字,同时保存于硬件中。识别流程不包括图像的采集、最终图像的输出,这类工作由其他硬件实现。

2 图像预处理

预处理是文字识别的重要一环,预处理对图像质量的提升起到了关键作用。预处理通常包括灰度化、二值化、倾斜校正和噪声消除[9],该文实现了以下几个预处理方法,为 Tesseract 文字识别做好准备。

2.1 图像缩放

Tesseract 处理图像的 DPI(dots per inch,每英寸点像素数)默认为 300 DPI,因而对于 DPI 不足的图片,需要进行缩放处理。常用的缩放算法有最近邻算法(最近邻插值)、双线性插值算法以及更为复杂的三次插值算法[10],三种算法的精度不一。基于文字识别的精度要求,该文实现了双线性插值图像缩放算法。

在离散数学中,插值是指在离散数据的基础上补差连续函数[11],使得该连续曲线通过全部给定的离散数据点,关键在于恢复曲线的连续与平滑。与之类似,图像基本构成是有限像素集,在图像缩放的过程中,需要对图像进行像素的插入(放大)或是叠加(缩小)。

图像缩放的过程中,原图像与输出图像具有一定的映射关系:

$$Y(x'+u,y'+v)=X(x,y)=X(h_1(x,y),h_2(x,y)) \tag{1}$$

其中,$X(x,y)$ 表示输入图像,$Y(x'+u,y'+v)$ 表示输出图像,h_1 和 h_2 表示映射函数。转换之后可能出现小数的情况,令 x' 和 y' 作为坐标的整数部分,u 和 v 作为小数部分。

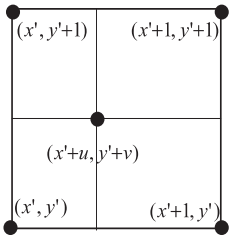

图 1 双线性插值示意

如图 1 所示,取转换点四个领域整数位置点 (x',y'),$(x',y'+1)$,$(x'+1,y')$,$(x'+1,y'+1)$。设 RGB 满足函数 $f(x,y)$,则转换之后的点坐标中的 RGB 值为:

$$f(x,y)=f(x'+u,y'+v) \tag{2}$$

$$\begin{aligned} f(x'+u,y'+v)=&f(x,y)*(1-u)*(1-v)+\\ &f(x',y'+1)*(1-u)*v+\\ &f(x'+1,y)*u*(1-v)+\\ &f(x'+1,y'+1)*u*v \end{aligned} \tag{3}$$

式(3)类比于一维坐标系下的长度比值关系,在二维坐标系下,是富含规律的面积比值关系。即所求点的 RGB 等于:

$$f(x,y) = \sum R_i * S_{对角} \tag{4}$$

其中,R_i 表示领域点的 RGB 中的某一项,$S_{对角}$ 表示对角的矩形面积值。

示例代码:

```
for (int k=0;k<3;k++)
{
change[x+k]=image[o_x+k]*(1-a_x)*(1-a_y)+
//对应f(x,y)*(1-u)*(1-v)
                image[o_b+k]*a_x*(1-a_y)+
/对应/f(x',y'+1)*(1-u)*v
                 image[o_c+k]* a_y*(1-a_x)+
//对应f(x'+1,y)*u*(1-v)
                 image[o_d+k]* a_y*a_x;
//对应f(x'+1,y'+1)*u*v
}
```

2.2　二值化

二值化的过程形如二进制中的 01 变化,图像的二值化就是将图像的像素点的灰度值设置成 0 或 255。首要工作是将图像灰度化,灰度化是将原图像各像素 R、G、B 三个值按照不同的权值,将加权和均赋值给目标图像的 RGB 值。二值化的算法的重点是获取像素点灰度的阈值,高于阈值的像素点灰度设置为 255,反之设置为 0。二值化的算法众多,常用的如利用聚类思想、根据灰度特征划分前景后景的 Otsu 算法,以及基于像素点为中心确定利用范围像素确定像素阈值的 Bernsen 算法[12]。

由于人眼对 RGB 三色的敏感度不同[13],因而灰度化时权值必然不同,该文利用心理学公式(5)中的权值实现灰度化,该公式适用于 sRGB 存储空间的图像。

$$\text{Gray} = R*0.299 + G*0.587 + B*0.114 \tag{5}$$

实现方法只需遍历所有像素点,对其 RGB 求加权和,将新值更新到源像素点中。

对于二值化过程,该文使用 Otsu 算法,Otsu 算法又称为“大津法”、最大类间方差法[14]。该算法的主要思想是根据图像的灰度特征将图像分为前景与背景两部分,而前景就是所求的目标图像[15],使用方差这一统计量,反映前后景灰度差距,其中存在概率分布,实行概率加权,最终遍历所有可能的灰度值,方差最大处为灰度阈值。

具体流程如下:

设图像灰度范围为 $[0,L-1]$,阈值 $M \in [0, L-1]$。

(1)分别计算像素灰度值在 $[0,M]$ 与 $(M,L-1]$ 出现的概率 P_1 和 P_2,其中:

$$P_1 + P_2 = 1 \tag{6}$$

(2)分别计算像素在 $[0,M]$ 和 $(M,L-1]$ 的平均值 μ_1 和 μ_2 及总体灰度平均值 μ。

(3)灰度类间方差函数为:

$$\sigma^2 = P_1(\mu_1-\mu)^2 + P_2(\mu_2-\mu)^2 \tag{7}$$

由概率以及均值关系可推出最终表达式:

$$\sigma^2 = P_1P_2(\mu_1\mu_2)^2 \tag{8}$$

(4)遍历所有灰度阈值 M,方差最大时的 M 即为最佳阈值。

示例为部分代码:

```
int OtsuAlgThreshold(const Mat){
int M;//阈值
for(int i=0;i<255;i++){
double p1=0,u1=0,p2=0,u2=0;
        double sum1=0,sum2=0;//像素点总数
         double gray_sum1=0,gray_sum2=0;//灰度值总数
for(int j=0;j<=i;j++){
sum2+=gram[j];
gray_num2+= j*gram[j];
            sum1+= gram[i+j];
gray_num1+=(i+j)*gram[i+j];
}
u2=gray_sum2/sum2;
p2=sum2/total;
u1=gray_sum1/sum1;
p1=sum1/total;
//类间方差计算
double Sq=p1*p2*(u1-u2)*(u1-u2);
if(SqMax<sq){
SqMax=sq;//更新最大方差值
M=i;
}
}
return M;//返回阈值 M
}
```

以上获取最佳方差的灰度值之后,便可通过遍历像素点进行灰度值调整。

2.3　边框与偏移

输入的图像由于采集设备的原因可能存在意想不到的边框,或是图像角度发生了偏移,因而会影响接下来的版面分析与行列切分工作,最终影响识别效果。此部分涉及到两个部分,一是发生偏移的角度,二是边框的选取。该文参考 OpenCV 库中的函数,实现了一种方法,方法的重点在于遍历查找最小外接矩形,旋转之后进行切割,从而获取最终图像。方法流程如下:

(1)获取图像最小外接矩形。

(2)获取图像偏移角度,同时进行旋转操作。

(3)对图像进行轮廓检测,循环筛选。

(4)切边,获取最终图像。

该思路之前需要对图像进行灰度化和二值化处理,以便于提高检测的精度。

3 文字识别

此前的图像预处理是为了文字识别作铺垫。预处理过程使得图像质量得到一定的改善,其 DPI 和对比度得到显著提升,文字的方向和边框也得到校正,从而改善了识别的准确性。文中的汉字识别,依赖于 Tesseract 框架,识别出 Unicode 字符。

此测试环境需要的配置如表 1 所示。

表 1 配置环境与说明

组件	版本	说明
Libtiff	4.9	Tesseract 依赖于 Libtiff 和 Leptonica 库,分别对应于图像的输入与处理,使用 VS 集成编译环境,测试代码
Leptonica	1.76.0	
Tesseract	3..05.01	
Visual studio	2015	

预先需要使用 CMake 编译工具将 Libtiff、Leptonica 编译,在 Tesseract 中添加动态链接库(后缀为. dll)文件之后方可使用。

识别代码如下:

```
char * outPut;
tesseract::TessBaseAPI * ocr = new tesseract::TessBaseAPI
();

    //初始化 ocr 识别模式
if(ocr->Init(NULL, "eng")) {
    cerr<<"Initialize Failed"<<endl;//初始化失败的情况
exit(1);//退出
}
//传入参数
    Pix * image = pixRead(Imag_name);
ocr->SetImage(image);

    //输出结果
outPut = ocr->GetUTF8Text();
    cout<<outPut <<endl;
    ocr->End();
```

上述代码中的接口功能如表 2 所示。

表 2 接口与功能

接口	功能
Init	初始化,对内部变量初始化
pixRead	读入图片,将图片载入
SetImage	处理图片,包括图像处理与识别
GetUTF8Test	获取识别结果
End	结束操作,释放程序资源

参数及说明如表 3 所示。

表 3 参数与说明

参数	说明
outPut	输出文本字符串
ocr	Tesseract 类对象,负责识别工作
image	图片读入类型变量
Imag_name	目标图像文件名

输入图像通过预先的缩放、二值化、边框处理的工作之后,输入到 Tesseract 工作环境中,实现最终屏幕控制台的输出显示,从而完成文字识别的基本流程。

4 结束语

该文概述了文字识别的基本过程,对每一个过程进行了基本的介绍。在 Tesseract 框架已有的基础上,对预处理阶段的图像缩放、二值化、边框处理、倾斜校正重点研究,相应的模块具有公式的推导以及代码的实现。最后依赖于 Tesseract 框架,先调用实现的预处理方法,实现预处理,继而调用 Tesseract 对象提供的初始化、读入、处理接口进行图像的操作,从而输出字符串变量,实现文字的识别。

该文的 OCR 文字识别研究较为普通,面向的场景也较为单一,精度要求不甚严格。除印刷体以外,还有较为复杂的手写体,证件识别,高精度识别。在 OCR 的工作流程中,除了预处理之外,还可通过版面分析、图像切分以及后续的特征匹配和模型训练的算法提升来提高识别的效率。实现的预处理方法只是领域中的一种可行的方法,除此之外也有众多优秀的算法。

Tesseract 框架的功能强大,可移植性强,识别范围广。Tesseract 支持 Linux、Windows、Mac OS,在各个领域应用广泛。Tesseract 框架只能识别印刷体,而不能识别手写体,除此之外,如上文提到的,Tesseract 依赖于图像的质量,对于图像的 DPI,文字的排布也有着要求。因此,图像的预处理工作必不可少,其中的算法研究也是研究重点。

参考文献:

[1] 张婷婷,马明栋,王得玉. OCR 文字识别技术的研究[J]. 计算机技术与发展,2020,30(4):85-88.

[2] 冯 磊. OCR,让纸质信息“上高速”[J]. 信息系统工程,2006(9):90-93.

[3] 王文华. 浅谈 OCR 技术的发展和应用[J]. 福建电脑,2012,28(6):56.

[4] 李霄霄. 基于 OCR 的字符识别的研究与实现[J]. 科技视界,2017(14):98.

[5] 隋 雪,王晓彤,任桂琴. 词汇识别中字母换位效应的研究[J]. 辽宁师范大学学报:社会科学版,2015,38(3):345-350.

(下转第 174 页)