碎纸片的拼接复原

摘要

本文围绕碎纸片拼接问题，针对碎纸片纵切、横纵切、双面横纵切三种情况，建立了灰度矩阵，分别采用了相似度分析、FCM聚类算法、匹配模型的方法，并结合适当的人工干预，将碎纸片拼接还原。

针对问题一，建立了相似度模型，解决了碎纸片仅纵切情况下的还原问题。首先对图像进行数据处理，读取图片的灰度信息，构建灰度矩阵，并将灰度矩阵转化为 0-1 矩阵，从而将二维图片数值化。而又因为被切开的两边碎片边缘灰度是具有高相关性的，因此，提取出 0-1 矩阵的第一列与最后一列，存储在图片的左右边界矩阵中，通过建立两张图片的左右边界匹配度模型，探究图片的左右邻接关系。模型求解结果为：汉字图片从左到右依次为：008、014、012、 015、003、010、002、016、001、004、005、009、013、018、011、007、017、000、 006，英文的排序结果为：003、006、002、007、015、018、011、000、005、001、009、 013、010、008、012、014、017、016、004。

针对问题二，采用FCM聚类算法并结合适当的人工干预，解决了碎纸片横纵切情况下的还原问题。由于纸片进行了横纵切，因此需要处理碎纸片的灰度值应为图片四周的灰度值。对图像进行数据处理时，应当将得到的0-1 矩阵进一步转换为版式矩阵。然后将可能处于同一行的碎片用模糊C均值(FCM)聚类方法分到同一类。对于汉字或英文的缺行碎片（碎片中只有一行或者两行字），利用掩码补充模型将投影中缺失的行用掩码补齐，然后再放入分类。分类完成后，将每一类中的碎片进行横向排序拼接。每一类横向拼接完成后，从而拼接成完整图片。对于不能正确拼接的部分，采取了人工干预的策略，最终得到正确的拼接结果。

针对问题三，建立了组合匹配模型，解决了碎纸片双面横纵切情况下的还原问题。由于附件中的碎片分为正反两面，所以可以建立组合匹配模型将碎片首先将正反面碎片的灰度矩阵上下拼接，每张碎片的拼接方案有两种（灰度矩阵a放在左右镜像处理后的b上或灰度矩阵b放在左右镜像处理后的a上）。拼接过后，边缘灰度向量信息会增加为原来的两倍，这时利用问题二中的全局搜索模型用向量相关性的方法进行匹配拼接。拼接出所有行后，将每一行的下半部分矩阵切割放在上半部分的右边，又可增加横向拼接时的边缘信息，然后各行再次利用向量相关性的方法进行横向拼接，最终得到结果图，但是仍然存在较多的碎片不能正确匹配，需要人工干预。

关键词：0-1矩阵、特征因子、FCM聚类算法、掩码补充法、人工干预

1. 问题重述

破碎文件的拼接在司法物证复原、历史文献修复以及军事情报获取等领域都有着重要的应用。传统上，拼接复原工作需由人工完成，准确率较高，但效率很低。特别是当碎片数量巨大，人工拼接很难在短时间内完成任务。随着计算机技术的发展，人们试图开发碎纸片的自动拼接技术，以提高拼接复原效率。就当前情况，我们需要讨论一下问题：

1. 对于给定的来自同一页印刷文字文件的碎纸机破碎纸片（仅纵切），建立碎纸片拼接复原模型和算法，并针对附件1,2给出的中、英文各一页文件的碎片数据进行拼接复原。如果复原过程需要人工干预，说明干预方式及干预的时间节点。复原结果以图片形式及表格形式表达。

2. 对于碎纸机既纵切又横切的情形，设计碎纸片拼接复原模型和算法，并针对附件3,4给出的中、英文各一页文件的碎片数据进行拼接复原。如果复原过程需要人工干预，说明干预方式及干预的时间节点。复原结果表达要求同上。

3. 上述所给碎片数据均为单面打印文件，从现实情形出发，还可能有双面打印文件的碎纸片拼接复原问题需要解决。附件5给出的是一页英文印刷文字双面打印文件的碎片数据。尝试设计相应的碎纸片拼接复原模型与算法，并就附件5的碎片数据给出拼接复原结果，结果表达要求同上。

二、符号说明

|  |  |
| --- | --- |
| 符号 | 表示含义 |
|  | 碎纸片的灰度值矩阵 |
|  | 每个碎纸片灰度值矩阵的第一列 |
|  | 每个碎纸片灰度值矩阵的最后一列 |
|  | 代表有无字符像素 |
|  | 矩阵第列和矩阵第列相同像素值的个数 |
|  | 版式矩阵 |
|  | 聚类数目 |
|  | 聚类中心 |
|  | 初始隶属度矩阵 |

三、模型假设

1、假设需要复原的碎纸片来自同一张纸，且该张纸具有完整性；

2、假设在同一页中，每一行文字的大小、行间距和段落分布情况是相同的；

3、假设碎纸片中的文字端正，没有偏斜，并按从上到下、从左到右排列；

4、假设每一张碎纸片尺寸大小相等，边缘轮廓为规则的矩形，所有纸片没有毛边误差，裁纸破损等情况；

5、图片中的文字像素没有任何断点；

6、图像上没有任何噪声或污点造成像素干扰。

四、问题分析

4.1 对问题一的分析

问题一中，需要对于给定的来自同一页印刷文字文件的碎纸机破碎纸片（仅纵切）进行拼接复原。由于碎片文件仅为纵切所得，因此只需考虑图片纵向的吻合度。首先需要提取每个碎纸片的灰度值**[1]**，对读取出的所有碎纸片图片的灰度值，建立像素矩阵对其进行表示。由于拼接时仅仅需要考虑碎纸片两侧的文字特征，所以我们选取所有碎片灰度值矩阵的第一列和最后一列作为有效数据。同时又考虑到碎纸片中文字的特点，数据预处理时，当灰度值为255时即为白色，可令0，表无字符像素；当灰度值为0254时令1，表有字符像素。计算出所有碎片之间左右两边标记值一样的个数，以此作为碎片间吻合度参数，生成吻合度矩阵。通过对矩阵的分析，可得相似度**[2]**最高的两个元素，也代表相似度最高的两张碎纸片。最后将相似度最高的碎纸片两两拼接，人工检查图片拼接的准确性。

4.2 对问题二的分析

问题二中，对于碎纸片既纵切又横切的情形，若继续采用问题一的方法，一来需要处理碎纸片的灰度值应为图片四周的灰度值，字符的可辨识度**[3]**降低；二来不能控制所得相似度图片应该在同一行还是同一列。因此我们提出了第二种解决模型。该方法基于原图在横切时，同行字符在同一水平线上的假设，即在处于同行的被纵切的碎纸片对象，其版式具有高度的相似性。换言之，我们可以通过先将可能处于同行上的碎纸片进行完整和准确的拼接后，再对拼接后的碎纸片对象在纵向进行拼接，通过将该复杂问题转化为了若干个小问题进行解决，从而有效地提高了拼接的效率和准确率。

五、模型的建立与求解

5.1问题一的模型建立和求解

对于问题一，由于碎片文件仅为纵切所得，因此只需考虑图片纵向的吻合度。首先需要提取每个碎纸片的边沿灰度值，可以使用像素矩阵**[4,5]**对各碎纸片对象进行表示。考虑到碎纸片中文字的特点，数据预处理时，当灰度值为255时令0，表无字符像素；当灰度值为0254时令1，表有字符像素。通过对各个图片两侧相同像素点的个数进行统计，来比对得到相似度最高的两张碎纸片，之后进行两两拼接。在检验时可以根据模型求解结果，将碎纸片进行拼接，观察结果是否正确。

5.1.1模型的建立和求解

对于问题一，其简要的模型建立和求解过程可用下图表示：

读取碎片图像文字，并对像素进行预处理

将具有最佳相似度的两个碎纸片进行拼接，直至拼接完成

将所有碎片，按照所得次序进行拼接，并检验其正确性

沿一水平方向（左/右）对两个碎片对象边界的像素矩阵进行相似度比较

图5.1-1问题一的建模求解流程

首先需要提取每个碎纸片的灰度值，对读取出的所有碎纸片图片的灰度值，记为集合。对所得数据进行预处理时，因为纸片时拼接只需考虑纸片两侧的吻合度，因此第一列和最后一列的灰度值作为我们的有效数据。将）中第一列和最后一列的数据提取出来，分别存放于矩阵

和

中，以分别表示每个纸片两侧的灰度值。考虑到碎纸片中文字的特点，当灰度值为255时为白色，也即该位置没有文字显示。所以数据处理时，当灰度值为0时令0，表无字符像素；当灰度值为1255时令1，表有字符像素。

即

定义矩阵

将第个碎纸片两侧的灰度值与第个碎纸片两侧的灰度值进行比较，若，则为有效计数值，将相同值个数的结果存放于矩阵中。

计算公式为：

再对矩阵进行归一化处理，得到第行、第个元素与第行其余元素进行相似度的比较结果。

得到的关于附件一中文文件和附件二英文文件的归一化处理结果如下：



图 5.1-2 中文印刷文件的像素矩阵归一化结果



图 5.1-3英文印刷文件的像素矩阵归一化结果

取，即取相似度最高的两个元素，也即相似度最高的两张碎纸片，对其进行两两匹配。

所求结果由图表表示分别为：

表5.1-1 附件一中文印刷文件复原结果

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 14 | 12 | 15 | 3 | 10 | 2 | 16 | 1 | 4 | 5 | 9 | 13 | 18 | 11 | 7 | 17 | 0 | 6 |

表5.1-2 附件二英文印刷文件复原结果

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 6 | 2 | 7 | 15 | 18 | 11 | 0 | 5 | 1 | 9 | 13 | 10 | 8 | 12 | 14 | 17 | 16 | 4 |

5.1.2 模型的检验

根据所得的复原结果，将碎片图形拼接进行检验，观察其正确性。

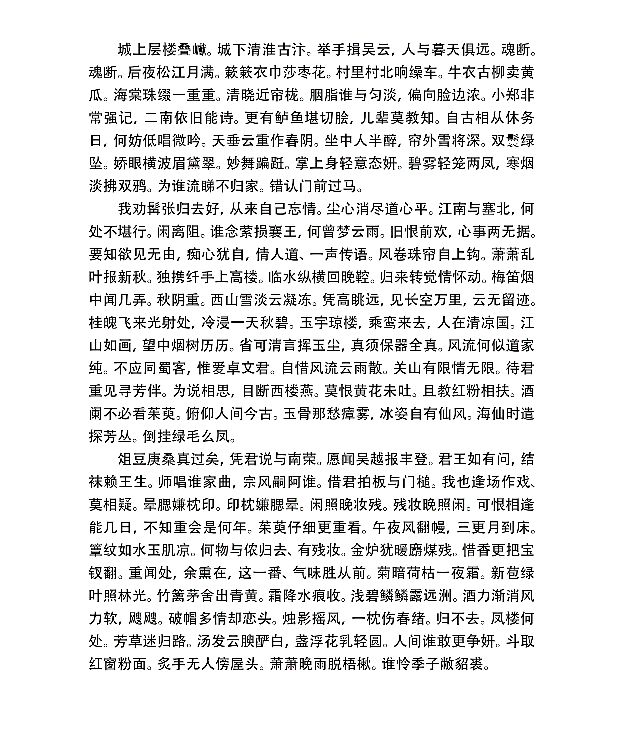
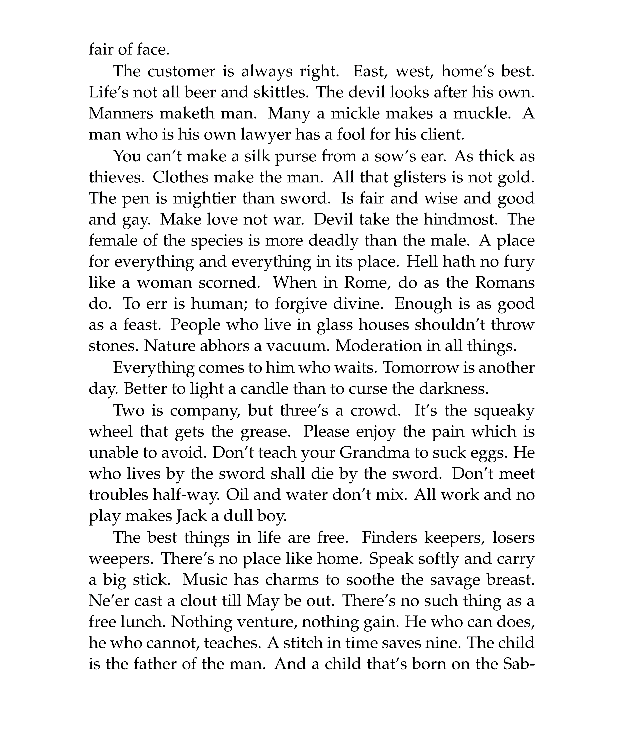
 

图5.1-4 中英文印刷文件复原图

5.1.2 问题的结论

针对问题一采取的方法，分别以0、1代表无字符像素和有字符像素，再通过比较纸片间相同像素个数来反映拟合度，具有合理性。由复原的效果图，显然该方法是正确的，且没有进行人工干预。

5.2问题二的模型建立与求解

问题二在问题一纵切的基础上，对原图进行横切，使得碎纸片尺寸变小，字符的可辨识度降低。当使用问题一提出的方法时，由于单个碎纸片的像素矩阵缩小，其边界像素矩阵也相应减小，在进行相似度比较时，容易产生较大误差，并不能准确地将两个碎纸片进行拼接复原。对数据进行预处理时建立版式矩阵，存储灰度值信息。之后采用FCM算法**[7,8]**先将可能处于同一行的碎片进行分类。对每一行的碎片进行最优拼接，考虑到特殊图片带来的误差，因此需要结合人工干预**[9]**的方法。当每一个分类都成功拼接出正确图片行后，将这11个分类用在纵向进行拼接。纵向拼接成功后，在用向量相关模型进行横向拼接，最终可以得到正确结果。检验时，将所得排列结果拼接后，通过人工观察文字语意、连贯性，检验结果的正确性。

5.2.1模型的建立和求解

问题二在纵切的基础上进行横切，使得碎纸片尺寸变小，字符的可辨识度降低，复杂度增加。在求解问题二使可遵循下列流程。

读取碎片图像文字，并对像素进行预处理

对于每一个碎纸片对象，获取其版式矩阵

根据碎纸片对象的版式矩阵间的相似度进行聚类，聚类算法采用算法

对聚在同一类中的碎纸片进行最优拼接

进行人工干预

图5.2-1 问题二建模求解流程

首先是读取碎片图像文字，在获取像素数据后，将碎纸片对象转化为像素矩阵时，可以当灰度值为255令矩阵中0，表无字符像素；当灰度值为0254时令矩阵中1，表有字符像素。在对数据进行下一步处理时，由于每页纸被切割为个碎片，而我们的首要目标是根据碎纸片间相似的版式，找到可能处于同行的碎纸片对象。因此，我们定义了碎纸片的特征向量如下：

设每页纸被切为个碎片，每个碎纸片的像素数为，将全部 张碎片进行黑白二值化处理后，若第个碎纸片的第行为纯白色行时该行标记；否则该行标记，则称向量 为第 个碎纸片的特征向量，其中

版式矩阵的转换如下所示：

将对209的碎纸片的像素矩阵进行处理后，数据存储于矩阵中，定义矩阵如下：

则矩阵作为数据预处理后的结果，存储了像素矩阵的所有预处理信息。

我们先用轮廓值对K-Means方法**[10]**得到的聚类结果进行评价，并据此来确定最佳的类别数。发现当类别数为13时，得到的轮廓数较均匀。且纸被切割为1119的碎片，取1311，具有合理性。之后对于碎纸片可能在同一行的聚类采用K-Means算法来实现。

K-Means算法的一般步骤如下：

（1） 从 n个数据对象任意选择 k 个对象作为初始聚类中心；

（2） 根据每个聚类对象的均值（中心对象），计算每个对象与这些中心对象的距离；并根据最小距离重新对相应对象进行划分；

（3） 重新计算每个（有变化）聚类的均值（中心对象）；

（4） 计算标准测度函数，当满足一定条件，如函数收敛时，则算法终止；如果条件不满足则回到步骤（2）。

这种划分使得下式最少：

其中为各类的中心。

对所得结果进行分析时发现，分类效果并不理想。

K-Means对于碎纸片的特殊情况并不能识别：

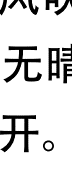
 

图5.2-2 碎片编号025 图5.2-3 碎片编号167

但实际上它们是处于同一行的，可以合成：

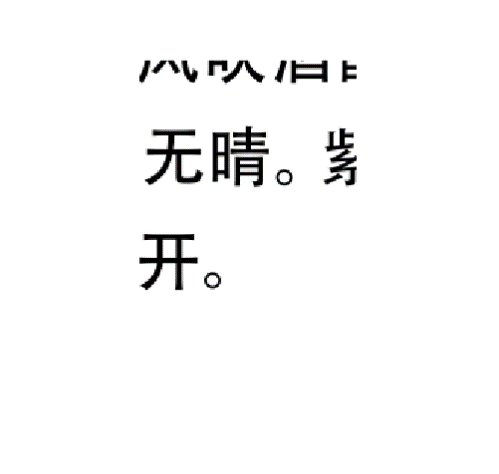


图5.2-4 两个碎片合成图

通过分析是因为在K-Means算法中，一开始的K个中心点是随机选定的，在后面的迭代中再进行重算，直到收敛，中心点会慢慢偏移。这样一来最后所生成的结果往往很大程度上取决于一开始k个中心点的位置。这样一来，也就意味着结果具备很大的随机性，每次计算结果都会因为初始随机选择的中心质点不一样而导致结果不一样。

通过进一步分析发现，大概有两种容易造成较大误差的情况。第一种情况是边缘字形相近，比如第46张与第25张碎片，因为“迎”和“晴”都切在了竖线上，所以相关系数会很大；第二种情况是碎片边缘没有任何字符，比如第74张与第9张碎片，因为第74张碎片的右端边缘没有任何中文字符，造成正确无法计算相关系数。而此种情况在碎纸片内容为英语字符时出现的次数更多。另外，英语字母高度不一致也会导致版式模型相似度计算出现误差。

综合考虑K-Means算法的缺点和碎纸片的特殊情况后，在聚类时我们采用FCM算法，它的优点是将K-Means算法的硬性划分方法推广到模糊情形。

FCM算法的基本步骤是：

（1）设定聚类数目c和参数m；   
（2）给出初始隶属度矩阵U（0）；   
（3）利用式子

计算新的聚类中心

（4）利用

计算新的隶属度矩阵   
（5）用一个矩阵范数比较两次迭代之间隶属度矩阵，如果

则停止迭代。

5.2.1.1 中文字的碎片拼接

对中文碎片资料执行FCM聚类算法后，得到按行分类结果：

表5.2-1 中文碎片资料聚类结果

|  |  |
| --- | --- |
| 中心点 | 编号 |
| 0.99561 | 35 43 44 48 59 78 85 91 95 98 113 122 125 128 137 145 150 165 184 |
| 0.013755 | 3 12 23 29 50 55 58 66 92 96 119 130 142 144 179 187 189 191 193 |
| 0.99213 | 7 20 21 37 53 62 64 68 70 73 79 80 97 100 117 132 163 164 178 |
| 0.83669 | 14 16 18 28 34 72 81 84 86 133 134 153 157 166 171 183 199 201 203 206 |
| 0.015373 | 5 41 61 90 102 103 109 114 115 118 120 124 126 141 147 152 155 156 186 195 208 |
| 0.0067661 | 2 19 24 27 31 42 51 63 77 87 88 101 121 143 148 169 180 192 196 |
| 0.99187 | 9 10 25 26 36 39 47 75 82 89 104 106 123 131 149 162 168 190 194 |
| 0.96435 | 6 11 30 38 45 49 56 60 65 76 93 99 105 112 172 173 181 202 207 |
| 0.0089549 | 4 13 15 32 40 52 74 83 108 116 129 135 136 160 161 170 177 200 204 |
| 0.0089549 | 17 22 67 107 110 111 140 146 151 158 174 182 185 188 198 205 |
| 0.99095 | 1 8 33 46 54 57 69 71 94 127 138 139 154 159 167 175 176 197 209 |

分类后C6中只有第4，5，10类中不是19张图片。

按行分类后：

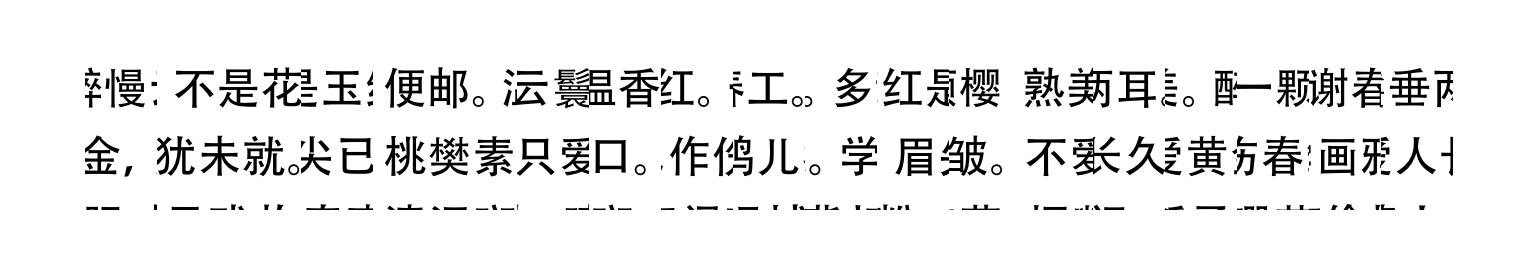


图5.2-5 某一行的碎片分类结果

在对碎纸片对象进行聚类后，我们首先对每一类中的对象进行类似于问题一中的拼接算法，即通过比较它们之间像素矩阵边界的相似度，将最有可能相邻的对象进行拼接。分类好的碎片逐类横向拼接的结果如下：

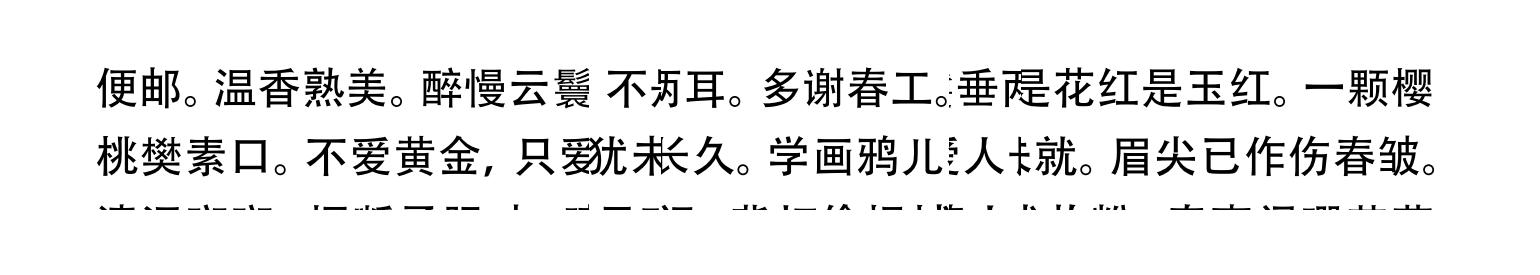


图5.2-6 某一行的碎片拼接结果

在此过程中，虽然我们将可能处于同行的碎纸片对象进行了聚类，但是聚类中的对象并不能保证是真正处于同行 的，因此会出现一个聚类中，应用拼接算法后得到若干个互相之间无法再进行拼接的碎纸片对象。产生这种情况的原因是多方面的，例如包含段落头和段落尾的碎纸片对象往往不能被准确分入对应行所在的聚类，又或者纵切位置比较特殊的碎纸片对象，因而此时需要进行人工干预。 人工干预时，根据中文字的特征，以及上下文的语义，我们进行手动调整错误的图片的位置，将此行拼成正确图片。

人工干预得到的结果：

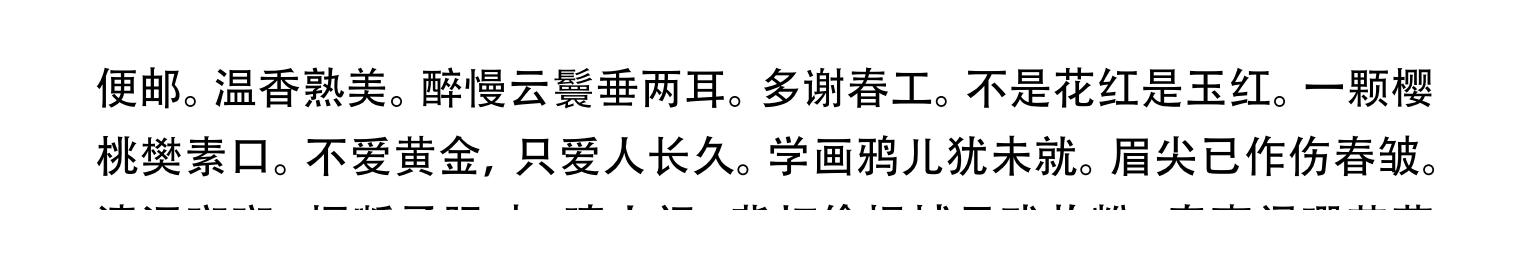


图5.2-7 某一行的碎片经人工干预后的拼接结果

当每一个分类都成功拼接出正确图片行后，将这11个分类用在纵向进行拼接。纵向拼接成功后，在用向量相关模型进行横向拼接，最终可以得到正确结果。

5.2.1.2 英文字的碎片拼接

对英文字的碎片拼接若继续采用中文字的拼接方法，则误差比较大，需要更强的人工干预。这是英文字符书写和汉字书写的差异性造成的。

针对英文字像素图片，可以英文字母“四线三格”的书写格式特点首先建立向量投影匹配分组模型。所以，利用这一特征，我们可以首先利用掩码补充法将图片中英文字的行间距找出来，利用不同行的文字中行间距在图片中的高度不通，对碎片经行预分类，将同一行的碎片分在一类总，然后利用第一问中的方法对行内小块经行排序。掩码补充法的具体做法是：将像素矩阵水平投影，设当某一像素行都为白色255时，令投影值为0，当一像素行有小于255的像素块时，此行令投影值为1。最终，209张图片生成209列的投影矩阵Shadow。

通过碎片灰度矩阵的每行灰度值的和表示成折线图，如图所示：

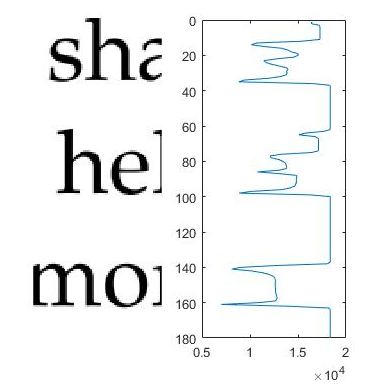


图5.2-8 英文碎片的投影矩阵

通过观察，每当那一行的值变化比较剧烈时，那一行一般情况下总是四线中的某一线。通过多次分析得出相邻两条直线之间的距离，大致分别为(12,20,15,25),之后制作模板，把每一个碎片与模板比较。通过寻找每一个碎片起始直线位置，与模板对应寻找可以匹配的个数，通过循环找出与模板匹配率最高的模式。根据匹配的模式，从起始位置对碎片根据此模式的间距向上向下添加直线，完成直线填补工作。

之后，通过模糊C均值(FCM)聚类方法进行分类，根据补齐直线投影矩阵分出11类，由于分类方法的模糊性特征，造成分出的类别误差相对较大，在之后运用相关性函数匹配时造成了很大的错误，在人工干预时大大增加了工作量。

表5.2-2 英文碎片资料聚类结果

|  |  |
| --- | --- |
| 中心点 | 编号 |
| 0.99561 | 35 43 44 48 59 78 85 91 95 98 113 122 125 128 137 145 150 165 184 |
| 0.013755 | 3 12 23 29 50 55 58 66 92 96 119 130 142 144 179 187 189 191 193 |
| 0.99213 | 7 20 21 37 53 62 64 68 70 73 79 80 97 100 117 132 163 164 178 |
| 0.83669 | 14 16 18 28 34 72 81 84 86 133 134 153 157 166 171 183 199 201 203 206 |
| 0.015373 | 5 41 61 90 102 103 109 114 115 118 120 124 126 141 147 152 155 156 186 195 208 |
| 0.0067661 | 2 19 24 27 31 42 51 63 77 87 88 101 121 143 148 169 180 192 196 |
| 0.99187 | 9 10 25 26 36 39 47 75 82 89 104 106 123 131 149 162 168 190 194 |
| 0.96435 | 6 11 30 38 45 49 56 60 65 76 93 99 105 112 172 173 181 202 207 |
| 0.0089549 | 4 13 15 32 40 52 74 83 108 116 129 135 136 160 161 170 177 200 204 |
| 0.0089549 | 17 22 67 107 110 111 140 146 151 158 174 182 185 188 198 205 |
| 0.99095 | 1 8 33 46 54 57 69 71 94 127 138 139 154 159 167 175 176 197 209 |

在对碎纸片对象进行聚类后，我们首先对每一类中的对象进行类似于问题一中的拼接算法，即通过比较它们之间像素矩阵边界的相似度，将最有可能相邻的对象进行拼接。得到同一类的碎片逐类横向拼接的结果后，还是由于将可能处于同行的碎纸片对象进行聚类后，但是聚类中的对象并不能保证是真正处于同行的。因此会出现一个聚类中，应用拼接算法后得到若干个互相之间无法再进行拼接的碎纸片对象。因而此时需要进行人工干预。 人工干预时，根据英文字的特征，以及上下文的语义，我们进行手动调整错误的图片的位置，将此行拼成正确图片。

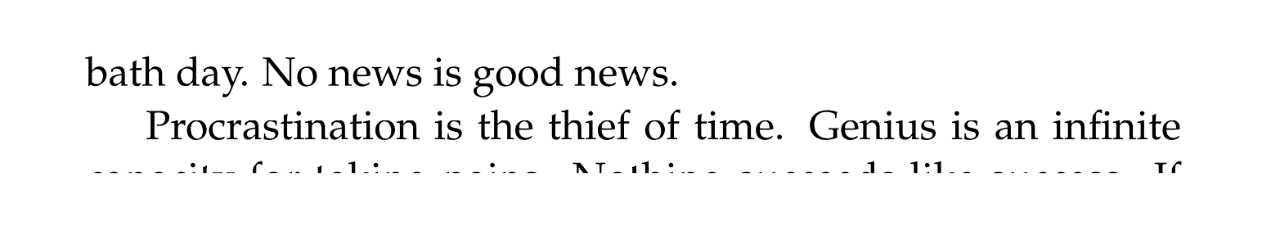


图5.2-9 某一行的碎片经人工干预后的拼接结果

当每一个分类都成功拼接出正确图片行后，将这11个分类用在纵向进行拼接。纵向拼接成功后，在用向量相关模型进行横向拼接，最终可以得到正确结果。

5.2.2 问题的结论

针对问题二采取的方法，建立版式矩阵存储灰度值信息，使用FCM算法对可能在同一行的碎片进行聚类，再结合人工干预的方法对碎纸片进行拼接。该方法具有一定的合理性。人工检验复原的效果图，显然该方法是正确的。

六、模型评价

6.1 模型的评价

（1）问题一中的连续性模型对于拼接边缘信息量较大的碎片有较高的效率以及很高的准确率，但是当单独用该模型解决例如附件三、四的小碎片时，虽然效率很高，但是准确率较低。比如，如果有两张或者多张图片边缘切割都为白色时，可能就会出现无法匹配的情况，这时就需要人工干预；向量相关模型相对于文字连续性模型，具有更高的准确度。

（2）问题二中的全局搜索模型不仅耗时多，而且由于给的信息很少，在这么多图片中匹配，会造成准确率很低的情况；向量投影分组模型，投影图片灰度向量到投影矩阵中，利用FCM聚类分析方法让打乱的横切及纵切的图片得以分类，从而分步实现了碎片的排序，简化了拼接过程，大大减少了循环匹配的次数。

6.2 模型的改进

本文提出的拼接复原算法可以继续在算法上进行优化，提升算法容错性，实现更多的由程序自动识别字体类型、自动规范文字结构以及自动完成图片的分类与拼接。也可以进一步优化分类算法，一方面细化分类准则，提高分类精度，另一方面挖掘更多同类图片间的共同信息，实现同类图片间的高效聚类。此外，如果在本文提出的模型和算法的基础上，加入对字符的识别技术，则能够有效减 少拼接的错误概率，并减少拼接过程中的人工干预。

6.3 模型的推广

这些模型针对不同的情况，利用多种模型和匹配方法，能够较好的解决题目中的问题。

我们还可以将本文的碎片拼接算法推广到拼接具有不同碎片形状、不同文字方向以及各种字体混杂出现的复杂碎片情况。如果碎片图像是彩色图片，我们还可以使用 MATLAB 软件读取图片的色彩数据，提出从色彩的连续性角度进行文字或图像拼接的模型及算法。

参考文献

[1]张欣,卜彦龙,朱良家,周宗 潭 物 证 复原 系 统 中的 碎 纸 轮 廓 提取 技 术 研究 [J],11,23(11):184，2006.

[2] Patrick Butler, Prithwish Chakra borty, Naren Ramakrishan, The Deshredder: A Visual Analytic Approach to Reconstructing Shredded Documents[J]. Department of Computer Science and Discovery Analytics Center, Virginia Tech, Blacksburg, VA 24061

[3] 罗智中，基于文字特征的文档碎纸片半自动拼接[J],计算机工程与应用，48(5)： 207-210，2013

[4]Hei Wang Chan, Evan Gillespie, Delfino Leong，Design and Implementation of a Paper De-shredder.ECE 412 Term Project Report.P.2-4 December 6, 2010

[5]杨郑，基于块匹配和特征点匹配的图象拼接算法研究[D],成都，西南交通大学，2009

[6] 李卓，邱慧娟，基于相关系数的快速图像匹配研究，北京理工大学学报， 27(11):998-1000,2007.

[7]贾海燕．碎纸自动拼接关键技术研究［ Ｊ］．国防科技大学学报，2015

[8] Gama Leitao H C，Stolfi J.A method for the reassembly of two-dimensional fragmented objects[J].IEEE Transactions on Pattern Analysis and Machine Intelligence， 2002，24（9）：1239-1251.

[9] 任澍，唐向宏，康佳伦.利用纹理和边缘特征的Criminisi 改进算法[J].中国图象图形学报，2012，17（9）：1086-1091.

[10] 姜惠兰，安敏，刘晓津，等.基于动态聚类算法径向基函数 网络的配电网线损计算[J].中国电机工程学报，2015，25 （10）：35-39.

附录

第一问附录

%第一问关系矩阵

I0=imread('C:\Users\F Vadim\Desktop\000.bmp');

I1=imread('C:\Users\F Vadim\Desktop\001.bmp');

I2=imread('C:\Users\F Vadim\Desktop\002.bmp');

I3=imread('C:\Users\F Vadim\Desktop\003.bmp');

I4=imread('C:\Users\F Vadim\Desktop\004.bmp');

I5=imread('C:\Users\F Vadim\Desktop\005.bmp');

I6=imread('C:\Users\F Vadim\Desktop\006.bmp');

I7=imread('C:\Users\F Vadim\Desktop\007.bmp');

I8=imread('C:\Users\F Vadim\Desktop\008.bmp');

I9=imread('C:\Users\F Vadim\Desktop\009.bmp');

I10=imread('C:\Users\F Vadim\Desktop\010.bmp');

I11=imread('C:\Users\F Vadim\Desktop\011.bmp');

I12=imread('C:\Users\F Vadim\Desktop\012.bmp');

I13=imread('C:\Users\F Vadim\Desktop\013.bmp');

I14=imread('C:\Users\F Vadim\Desktop\014.bmp');

I15=imread('C:\Users\F Vadim\Desktop\015.bmp');

I16=imread('C:\Users\F Vadim\Desktop\016.bmp');

I17=imread('C:\Users\F Vadim\Desktop\017.bmp');

I18=imread('C:\Users\F Vadim\Desktop\018.bmp');

T=[I0 I1 I2 I3 I4 I5 I6 I7 I8 I9 I10 I11 I12 I13 I14 I15 I16 I17 I18];

for i=1:19

L(:,i)=T(:,1+(i-1)\*72);

R(:,i)=T(:,72+(i-1)\*72);

end

for i=1:1980

for j=1:19

if(L(i,j)<255)

L(i,j)=1;

else

L(i,j)=0;

end

end

end

for i=1:1980

for j=1:19

if(R(i,j)<255)

R(i,j)=1;

else

R(i,j)=0;

end

end

end

%L

%R

count=0;

for i=1:19

for m=1:19

for j=1:1980

if((i~=m)&&(R(j,i)==L(j,m))&&(R(j,i)~=0)&&(L(j,m)~=0))

count=count+1;

end

end

M(i,m)=count;

count=0;

end

end

M=M/1980;

M

for i=1:19

max=M(i,1);

for j=1:19

if(M(i,j)>max)

max=M(i,j);

end

end

for s=1:19

if((M(i,s)==max)&&(max~=0))

disp(['第',num2str(i-1),'张图右边的图:',num2str(s-1)]);

end

end

if(max==0)

disp(['第',num2str(i-1),'张图为最后一张图']);

end

end

for j=1:19

max=M(1,j);

for i=1:19

if(M(i,j)>max)

max=M(i,j);

end

end

if(max==0)

disp(['第',num2str(j-1),'张图为第一张图']);

end

end

%第一问拼图

I0=imread('C:\Users\F Vadim\Desktop\000.bmp');

I1=imread('C:\Users\F Vadim\Desktop\001.bmp');

I2=imread('C:\Users\F Vadim\Desktop\002.bmp');

I3=imread('C:\Users\F Vadim\Desktop\003.bmp');

I4=imread('C:\Users\F Vadim\Desktop\004.bmp');

I5=imread('C:\Users\F Vadim\Desktop\005.bmp');

I6=imread('C:\Users\F Vadim\Desktop\006.bmp');

I7=imread('C:\Users\F Vadim\Desktop\007.bmp');

I8=imread('C:\Users\F Vadim\Desktop\008.bmp');

I9=imread('C:\Users\F Vadim\Desktop\009.bmp');

I10=imread('C:\Users\F Vadim\Desktop\010.bmp');

I11=imread('C:\Users\F Vadim\Desktop\011.bmp');

I12=imread('C:\Users\F Vadim\Desktop\012.bmp');

I13=imread('C:\Users\F Vadim\Desktop\013.bmp');

I14=imread('C:\Users\F Vadim\Desktop\014.bmp');

I15=imread('C:\Users\F Vadim\Desktop\015.bmp');

I16=imread('C:\Users\F Vadim\Desktop\016.bmp');

I17=imread('C:\Users\F Vadim\Desktop\017.bmp');

I18=imread('C:\Users\F Vadim\Desktop\018.bmp');

c=[I3,I6,I2,I7,I15,I18,I11,I0,I5,I1,I9,I13,I10,I8,I12,I14,I17,I16,I4];

imshow(c)

**第二问代码**

**Fcm代码**

clear

clc

file\_path ='C:\Users\F Vadim\Desktop\B\附件3\';% 图像文件夹路径

img\_path\_list = dir(strcat(file\_path,'\*.bmp'));%获取该文件夹中所有jpg格式的图像

img\_num = length(img\_path\_list);%获取图像总数量

T=[];

if img\_num > 0 %有满足条件的图像

for j = 1:img\_num %逐一读取图像

image\_name = img\_path\_list(j).name;% 图像名

i = imread(strcat(file\_path,image\_name));

T=[T,i];

end

end

M=zeros(180,209);

for m=1:180

for i=1:209

for j=(1+72\*(i-1)):(72+72\*(i-1))

if(T(m,j)<255)

M(m,i)=1;

break

end

end

end

end

%X=M';

%M

A=[];

co=0;

for j=1:209

for i=1:180

co=co+i\*M(i,j);

end

A(1,j)=co;

co=0;

end

[center,U,obj\_fcn] = fcm(M',11);

maxU = max(U);

index1 = find(U(1,:) == maxU);

index2 = find(U(2,:) == maxU);

index3 = find(U(3,:) == maxU);

index4 = find(U(4,:) == maxU);

index5 = find(U(5,:) == maxU);

index6 = find(U(6,:) == maxU);

index7 = find(U(7,:) == maxU);

index8 = find(U(8,:) == maxU);

index9 = find(U(9,:) == maxU);

index10 = find(U(10,:) == maxU);

index11 = find(U(11,:) == maxU);

disp(['第一类:','中心点:',num2str(center(1,1)),'该类编号:',num2str(index1)]);

disp(['第一类:','中心点:',num2str(center(2,1)),'该类编号:',num2str(index2)]);

disp(['第一类:','中心点:',num2str(center(3,1)),'该类编号:',num2str(index3)]);

disp(['第一类:','中心点:',num2str(center(4,1)),'该类编号:',num2str(index4)]);

disp(['第一类:','中心点:',num2str(center(5,1)),'该类编号:',num2str(index5)]);

disp(['第一类:','中心点:',num2str(center(6,1)),'该类编号:',num2str(index6)]);

disp(['第一类:','中心点:',num2str(center(7,1)),'该类编号:',num2str(index7)]);

disp(['第一类:','中心点:',num2str(center(8,1)),'该类编号:',num2str(index8)]);

disp(['第一类:','中心点:',num2str(center(9,1)),'该类编号:',num2str(index9)]);

disp(['第一类:','中心点:',num2str(center(10,1)),'该类编号:',num2str(index10)]);

disp(['第一类:','中心点:',num2str(center(11,1)),'该类编号:',num2str(index11)]);

figure

c=[];

for i=1:length(index1)

c=[c,T(:,(1+72\*(index1(1,i)-1)):(72+72\*(index1(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index2)

c=[c,T(:,(1+72\*(index2(1,i)-1)):(72+72\*(index2(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index3)

c=[c,T(:,(1+72\*(index3(1,i)-1)):(72+72\*(index3(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index4)

c=[c,T(:,(1+72\*(index4(1,i)-1)):(72+72\*(index4(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index5)

c=[c,T(:,(1+72\*(index5(1,i)-1)):(72+72\*(index5(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index6)

c=[c,T(:,(1+72\*(index6(1,i)-1)):(72+72\*(index6(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index7)

c=[c,T(:,(1+72\*(index7(1,i)-1)):(72+72\*(index7(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index8)

c=[c,T(:,(1+72\*(index8(1,i)-1)):(72+72\*(index8(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index9)

c=[c,T(:,(1+72\*(index9(1,i)-1)):(72+72\*(index9(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index10)

c=[c,T(:,(1+72\*(index10(1,i)-1)):(72+72\*(index10(1,i)-1)))];

end

imshow(c)

figure

c=[];

for i=1:length(index11)

c=[c,T(:,(1+72\*(index11(1,i)-1)):(72+72\*(index11(1,i)-1)))];

end

imshow(c)

灰度值

filename = 'C:\Users\F Vadim\Desktop\B\附件4\003.bmp';

imgData = imread(filename);

figure

imshow(imgData);

%该函数可以用来显示已经读入的数据

c=sum(imgData,2);

figure

plot(c,1:180,'-');%画出图像数据