人工智能与数据挖掘课程设计

表情包制作

**组员信息：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 姓名 | 学号 | 成员贡献 | 实践成绩 |
| 1 | 陈昶源 | 2021211902 | 代码实现 |  |
| 2 | 刘腾钰 | 2021211916 | 报告撰写 |  |
| 3 | 吴炫 | 2021211908 | 报告撰写 |  |
| 4 |  |  |  |  |

# **引言**

表情包是一种通过图像和文字来表达情感和态度的网络文化现象，它在社交媒体和聊天软件中广泛流行，具有传播性、趣味性和创造性。表情包的制作通常需要借助一些图像处理软件，如Photoshop、AI等，这些软件的操作较为复杂，需要一定的专业技能和美术基础。而Python是一种简单易学、功能强大的编程语言，它提供了一些图像处理库，如PIL、turtle等，可以实现一些基本的图像处理功能，如裁剪、缩放、旋转、合成、添加文字等。利用Python来制作表情包，不仅可以提高编程能力，还可以培养创造力和审美能力，同时也可以增加编程的乐趣和趣味性。此外，通过python还可以实现将人像提取出来，再以此为素材来实现表情包的制作，在本课程设计中，我们组以此为理论基础，将从百度图片网站中以不同表情的关键词来进行爬取素材，再通过对这些素材进行筛选、裁剪，得到大小一致的人像图。而后我们将对这些人像裁剪处理，而后再实现图片的融合以及文字在图片中的定位等功能，来实现特定表情包的制作。

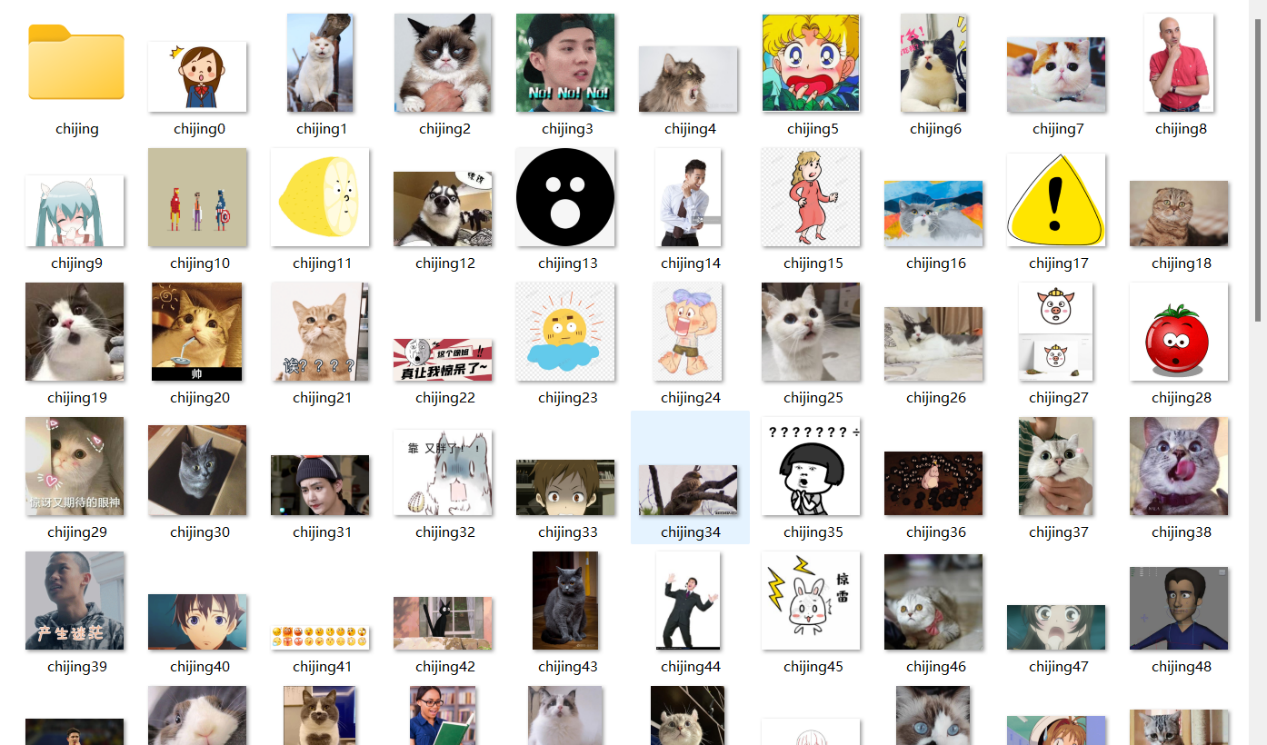
# **实验数据**

## 数据获取或预处理

## 首先在没有任何数据的情况下，该如何开始数据的获取将是一个难题，我们组考虑到了两种方法，一种是自己采集，例如用摄像头抓帧，另一种是爬虫爬取数据。而在该实验中我们组将采用爬虫爬取百度图片中的数据。然后自己设计自己需要的表情，我自己设置了10类。这里以“吃惊”表情为例，在百度图片搜索中可以看到：

## 

而我们所需要的就是将这些图片提取并下载到本地文件夹，我们组针对此问题，使用了python，借此获取了初步数据：



从百度中获得的图片大小不一，因此需要进一步进行处理

## 2.1预处理

# 爬取完人脸表情之后，我们需要裁剪处图像中的人脸，这里设置裁剪大小为128\*128。裁剪过程需要用到opencv的人脸识别工具，经过以上几步，基本可以得到人脸数据集了。而后以此为依据，我们将进行接下来的文字在图片中的定位与添加。

# **3. 方法/模型**

我们组在实现自制表情包中主要考虑了两方面的内容：其一为实现文字在图片中的自动定位与添加，其二为将两张相似图片自动拼接，实现图片的修复。下面将分点描述：

3.1 文字定位

经过前面的预处理步骤，我们得到了大小一致的人像图片，大小的一致性为我们后续的文字定位提供了便利，我们仅需在图片中选择合适的位置添加我们的文字即可，下面为我们实现该功能的步骤：

首先，导入需要用到的模块，包括cv2（用于图像处理），PIL（用于图像和文字的绘制），numpy（用于数组操作）和os（用于文件操作）。

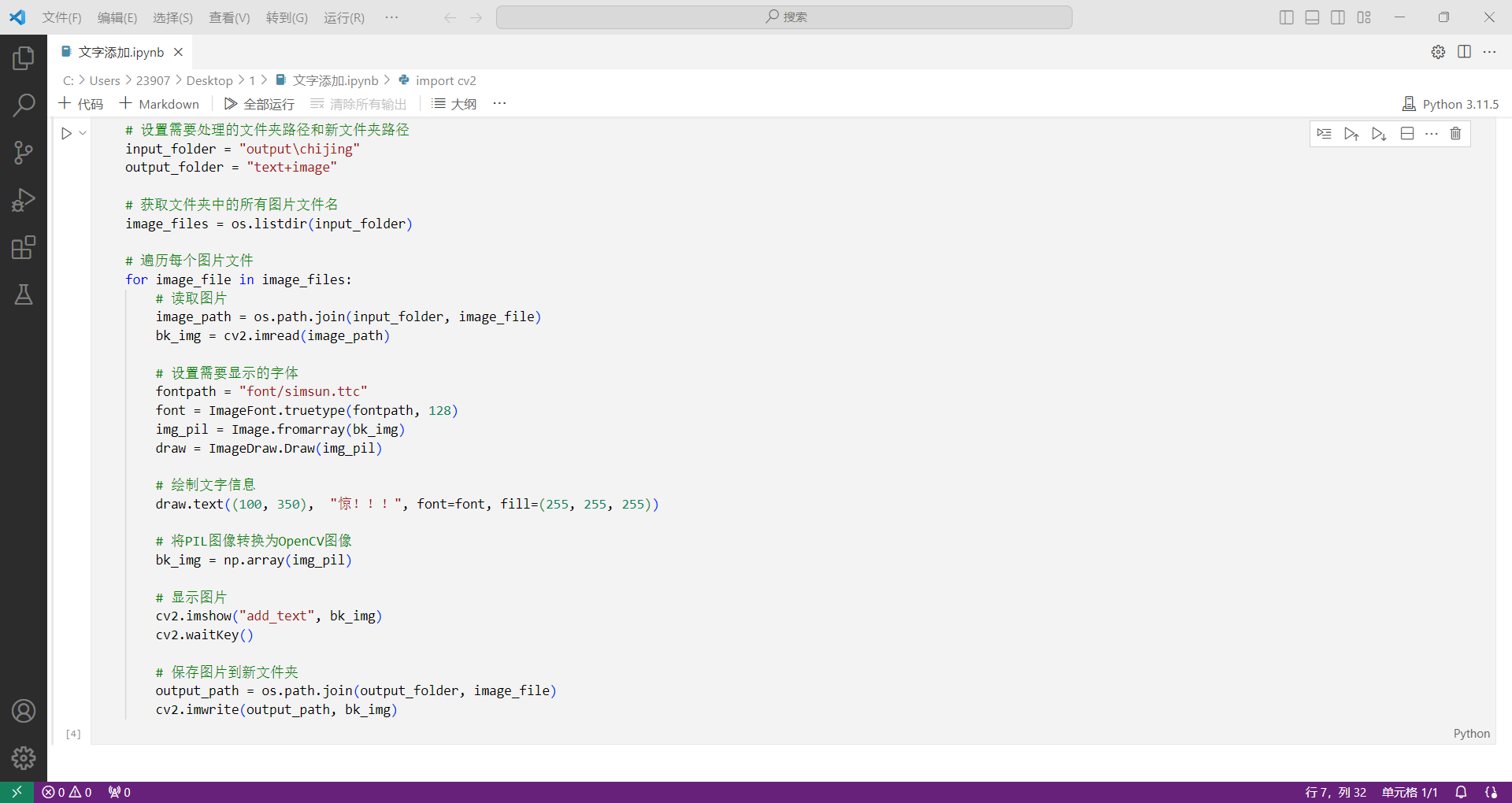
然后，设置需要处理的文件夹路径和新文件夹路径，分别为input\_folder和output\_folder。这两个文件夹应该已经存在，否则会报错。

接着，获取input\_folder中的所有图片文件名，存储在image\_files列表中。这里假设文件夹中只有图片文件，没有其他类型的文件。

然后，遍历每个图片文件，对每个图片文件执行以下操作：

1. 读取图片文件，得到一个OpenCV图像对象bk\_img。
2. 设置需要显示的字体，这里使用的是宋体（simsun.ttc），字号为128。创建一个PIL图像对象img\_pil，并将bk\_img转换为PIL图像。创建一个PIL绘图对象draw，用于在img\_pil上绘制文字。
3. 在img\_pil上绘制文字信息，这里使用的是白色的“惊！！！”字样，位置为(100, 350)。这里的位置是相对于图片左上角的坐标，单位为像素。
4. 将img\_pil转换回OpenCV图像对象bk\_img。
5. 显示bk\_img，并等待用户按任意键关闭窗口。
6. 保存bk\_img到新文件夹output\_folder中，文件名与原文件名相同。
7. 最后，结束循环，完成所有图片的处理。

代码截图如下图所示，代码文件为“文字添加.ipynb”：



3.2 图片拼接

在日常生活中，我们还可能遇见这样的问题拍到的图片并不完整，或者需要几张图片的整合以得到一个更全面的图片，为了解决这个问题便要实现图片的拼接图像拼接的过程大致分为以下几个步骤：

1. 首先，读取待拼接的两幅图像，并统一它们的大小，以便后续的处理。
2. 然后，使用SIFT特征点检测算法，在两幅图像中提取出一些具有尺度不变性和旋转不变性的特征点，以及对应的描述子，用于表示特征点的局部特征。
3. 接着，使用FLANN匹配算法，根据描述子的相似度，找出两幅图像中的最佳匹配点对，即具有最小距离的匹配点对。为了提高匹配的质量，使用一个阈值筛选出优质的匹配点对，即最近邻距离小于0.65倍的次近邻距离的匹配点对。然后，可视化特征匹配的结果，并保存到文件中。
4. 接下来，使用RANSAC算法，从优质的匹配点对中，随机选取四个点，计算它们之间的单应性矩阵，即描述两幅图像之间的几何变换关系的矩阵。然后，使用该矩阵对所有的匹配点进行变换，计算变换后的误差，统计误差小于一定阈值的内点的个数。重复这个过程若干次，选择内点个数最多的单应性矩阵作为最终的结果。
5. 然后，使用最终的单应性矩阵，对其中一幅图像进行透视变换，使其与另一幅图像对齐。这一步称为图像变形，其结果是将两幅图像映射到同一个坐标系中。
6. 最后，使用图像融合的方法，将两幅变形后的图像拼接成一幅大的图像。这一步需要处理两幅图像的重叠部分，使其在边界处能够平滑过渡，避免出现明显的接缝或者色差。这段代码使用了一种加权平均的方法，根据像素点距离重叠区域的左右边界的距离，给予不同的权重，从而实现图像的融合。最终，显示并保存拼接后的图像。

该代码文件为“图像拼接.ipynb”：

# **4. 结果评测/展示**

















可以从以下几个方面进行评测：

1. 相似性：在相似性方面，得到的的表情包能够与原始照片保持高度相似，包括面部轮廓、眼睛、鼻子等主要特征。观察表情包与原始照片的对比时，可以明显地看到生成的表情包在面部特征上具有高度一致性，呈现出与原始照片相近的外貌特征，如面部轮廓的线条、眼睛的大小和位置、鼻子的形状等。在准确性方面，得到的的表情包成功地传达了原始照片中的惊讶表情和姿势，使得它们与原始照片在情感表达上高度一致。
2. 细节保留：得到的的图片在细节保留方面表现出色，能够准确保留原始照片中的细节，包括一些关键特征，呈现出高度保真的效果。观察生成的表情包，可以清晰地看到细致的面部特征，这在一定程度上增强了表情包的真实感和质感。在惊讶表情细节的还原方面，生成的表情包也能够准确地表达出原始照片中的惊讶表情和惊讶情感。
3. 表达能力：得到的表情包能够准确地表达原始照片中的惊讶表情。能够通过脸部表情、眼神、姿态等方面准确地重现并传递出原始照片中所要表达的惊讶的意思。这使得生成的表情包具有很强的沟通力和表达能力，能够与观看者产生共鸣和情感连接。
4. 平滑度和连续性：得到的的表情包在面部特征变化的过程中保持了良好的平滑度，添加的字体能够流畅地呈现，没有出现明显的不连续或突兀的过渡，使得表情包变得突兀。这种流畅的表现使得得到的表情包在表情过程中呈现出自然、真实的效果，为个性化表情包的定制和使用提供了更加流畅和细腻的视觉体验。
5. 色彩保真度：得到的表情包能够准确还原原始照片中的色彩饱和度，无论是鲜艳的颜色、柔和的色调还是其他特定的色彩效果，都能够保持原始照片中的色彩饱和度，使得表情包的色彩表现出生动而鲜明的效果，保持原始照片中的色彩平衡。无论是皮肤的色调变化还是细微的阴影和高光，都能够准确呈现细节部分的色彩信息，让观看者能够清晰地感知到每个细节的色彩变化，增强了表情包的真实感和质感。

# **总结**

在实验初期，我们选择了用百度图片作为数据源进行爬取下载，在试验中，我们发现程序爬取图片在同一IP下有一定的时间间隔，在这个间隔中无法进行二次爬取，可以考虑使用代理IP、设置合理的延迟、使用多线程或异步IO等技术提高爬取效率；同时，可以尝试使用更高级的爬虫框架，如Scrapy，以简化代码和提高可维护性。此外，还可以对爬取到的图片进行分类和过滤，以满足特定的需求。

由于爬取的图片大小不一，所以需要处理图片设定统一格式进行处理，使用OpenCv进行人脸处理，这就完成了对于数据的获取过程。

在实验中后期，我们组在实现自制表情包中主要考虑了两方面的内容：其一为实现文字在图片中的自动定位与添加，其二为将两张相似图片自动拼接，实现图片的修复。在上方的方法模块中已经有所体现，这里不再进行详细叙述。

在实验过程中，我们遇到了很多问题。例如，如何有效地选择和调整图像的尺寸，如何进行图像的拼接，以及如何处理颜色和光照的不一致性。为了解决这些问题，我们三个查阅了很多关于Python图像处理的资料，也尝试了不同的方法，例如使用PIL库进行图像的读取、处理和保存，使用OpenCV进行更复杂的图像分析和处理。这个实验的核心原理是图像处理和合成。通过Python的图像处理库，我们可以读取、修改和合成图像。在这个过程中，我们学到了很多关于颜色理论、图像格式以及像素操作的知识。