暨南大学计算机科学系

《数字图像处理》课程设计

成绩：

题　　目：	空间域图像处理
姓　　名：	杨博
学　　号：	2021101228
专　　业：	网络工程
课程类别：	专业选修课
任课老师：	彭玉青
提交日期：	2023 年 6 月 16 日

1. 选题和内容

1.1 选题和内容

选题：几何变换程序的设计

内容：设计一个几何变换程序，可在旋转角度较小（比方说θ＜ 6°）时，按照输入的参数进 行旋转、平移和比例缩放。

1.2 基本原理

1.2.1 图像平移

图像的平移：

$$x' = x + t_x$$
$$y' = y + t_y$$

在齐次坐标系中：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

1.2.2 图像缩放

图像的缩放：

$$x' = ax$$
$$y' = by$$

矩阵形式：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

齐次坐标系中：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

1.2.3 图像旋转

图像的旋转：

$$x' = x\cos(\theta) - y\sin(\theta)$$
$$y' = x\sin(\theta) + y\cos(\theta)$$

矩阵形式：

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

在齐次坐标系中：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

1.2.4 仿射变换

仿射变换：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

性质：

- 直线经过仿射变换之后，依然是直线
- 平行线经过仿射变换之后，依然是平行线
- 保持线段之间的比例关系不变

$$x' = a_1 x + a_2 y + a_3$$
$$y' = a_4 x + a_5 y + a_6$$

2. 开发环境与库函数说明

2.1 开发环境

编程语言的选择：编程语言选择使用的是 python。Python 是一种高层次的结合了解释性、编 译性、互动性和面向对象的脚本语言。

在此次实验中使用的 Python 解释器为 base：3.8

2.2 库函数说明

在程序设计中使用到的库函数有：OpenCV、numpy、time

1.Opencv：

OpenCV 是一个基于 Apache2.0 许可（开源）发行的跨平台计算机视觉和机器学习软件库，可以运行在 Linux、Windows、Android 和 Mac OS 操作系统上。 它轻量级而且高效——由一系列 C 函数和少量 C++ 类构成，同时提供了 Python、Ruby、MATLAB 等语言的接口，实现了图像处理和计算机视觉方面的很多通用算法。

2.numpy：

NumPy（Numerical Python）是 Python 的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表（nested list structure）结构要高效的多（该结构也可以用来表示矩阵（matrix）），支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

3.Time:

- time 库是 Python 中处理时间的标准库
- 提供获取系统时间并格式化输出功能
- 提供系统级精确计时功能，用于程序性能分析

3. 课程设计编程思想介绍

3.1 面向对象编程

面向对象编程（Object-Oriented Programming，OOP）是一种编程思想，它将数据和操作数据的方法封装在一起，以对象的形式表示出来。这种编程思想是基于现实世界中的对象和它们之间的相互作用来构建程序的。面向对象编程中，每个对象都有自己的属性和方法，而且可以与其他对象进行交互。面向对象编程有三个主要特征：封装、继承和多态。其中，封装是指将数据和方法封装在一起，以便于控制访问；继承是指一个类可以继承另一个类的属性和方法；多态是指同一个方法可以在不同的类中有不同的实现方式。

4. 程序设计过程

4.1 程序设计思路

4.1.1 设计思路

（1）设计一个函数实现对于图像的旋转，先对图像进行一次小角度（例如 6°）的旋转，再对得到的旋转后的图像进行旋转，缩放，平移操作。

（2）对于旋转操作的设计:

- 求出旋转中心
- 构建旋转矩阵

- 计算旋转后图像的大小和边界框
- 对图像进行仿射变换，使旋转后的图像在中心位置

（2）对平移的操作的设计：

- 生成平移矩阵
- 对图像进行平移
- 对图像进行连接操作生成对比图

（3）对缩放进行操作的设计：

- 按比例缩放的设计
- 按尺寸缩放的设计

4.1.2 流程图

（1）主函数流程图

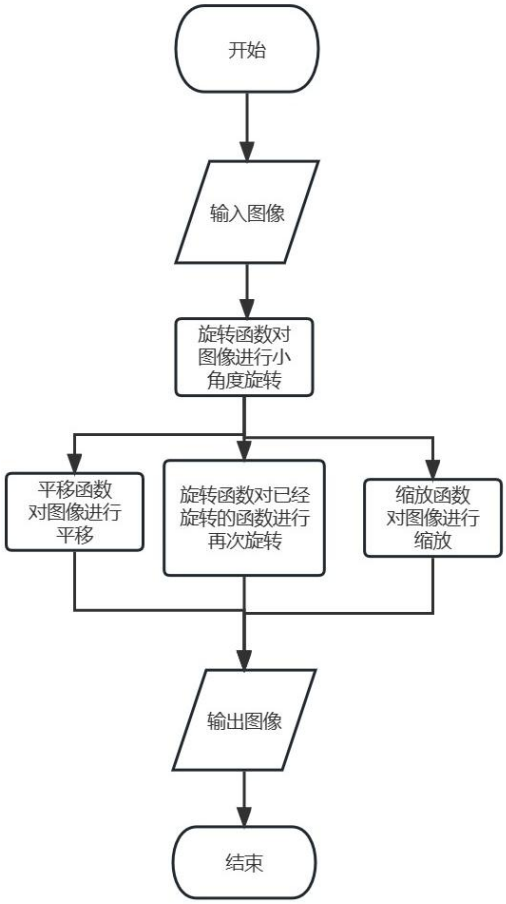


图1　主函数流程图

（2）旋转函数流程图：

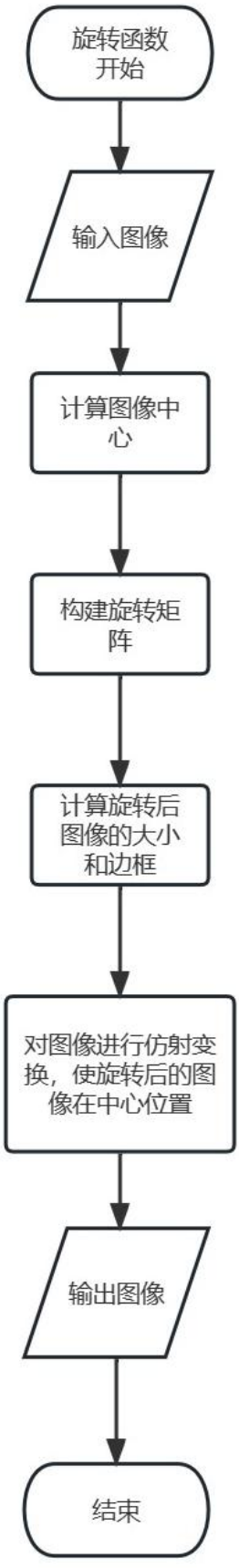


图2　旋转函数流程图

（3）平移函数流程图：

图3 平移函数流程图

4.2 探究与改进

4.2.1 改进思路

（1）提出使用近似的方法：

在进行图像旋转时，需要对每一个像素点进行坐标变换，计算出它在旋转后的位置，并通过插值算法计算出旋转后该像素点的灰度值。如果直接使用精确的插值方法进行计算，会导致计算量极大，严重影响程序运行效率。

因此，在实际应用中，一般会采用采样间隔较大、计算量较小的近似算法进行插值计算。这样可以在一定程度上平衡程序性能和图像质量之间的矛盾，从而实现高效的图像旋转操作。

（2）提出先对图像进行缩小再进行旋转的方法：

先对图像进行缩小再进行旋转操作可以加快计算速度，降低计算复杂度。缩小图像可以减少旋转操作所需的计算量和内存占用，从而加快程序运行速度。在缩小后的图像上进行旋转，计算量会更小，从而更容易实现较快的计算速度。

但需要注意的是，在缩小图像时，可能会丢失一些细节信息。因此，如果需要保留图像中的所有细节信息，则应该避免过度缩小图像，并尽量选择比较精确的插值算法进行图像缩放和旋转操作。

4.2.2 几种近似方法

（1）INTER_AREA： INTER_AREA 是一种计算较为简单的近似算法，它会根据像素区域关系进行插值计算，从而得到缩小后的图像。相比于其他更为精确的插值算法，INTER_AREA 在缩小图像时能够更好地保留图像的细节信息，并且能够减少锯齿状的伪影。此外，近似算法还具有一定的鲁棒性，能够在处理一些异常数据时保持较好的表现。INTER_AREA 算法能够更好地保留图像中的主要信息，并且对于缩小倍数较大的情况下，能够避免产生锯齿状或马赛克状的伪影。

（2）INTER_LINEAR：双线性插值算法。该算法计算速度较快，但对于像素间距较大或旋转角度较大的情况下，可能会导致图像变形。此外 INTER_LINEAR 双线性插值算法在放大图像时表现较好，因为该算法能够在保留图像细节信息的同时，平滑地插值出新图像的像素值。

（3）INTER_CUBIC：双立方插值算法。该算法能够更准确地保留图像的细节信息，但计算量较大，运行速度相对较慢。

（4）INTER_LANCZOS4：Lanczos 插值算法。该算法能够在保留较好图像细节信息的同时，产生较少的伪影，但计算量较大，运行速度相对较慢。

4.2.3 近似方法的选择

（1）对图像缩小选用的近似方法：

INTER_AREA 算法能够更好地保留图像中的主要信息，并且对于缩小倍数较大的情况下，能够避免产生锯齿状或马赛克状的伪影。

（2）对图像放大选用的近似方法

INTER_LINEAR 双线性插值算法在放大图像时表现较好，因为该算法能够在保留图像细节信息的同时，平滑地插值出新图像的像素值。

4.2.4 对算法消耗时间的监测

利用 python 标准库函数中的 time 类对算法的运行时间进行监测，检测改进

方式对于计算效率的提高程度。

记录函数开始时间，再记录函数结束时间，用结束时间-开始时间得到函数的运行时间。

4.2.5 具体改进步骤

对图像旋转进行改进的展开：

（1）对图像使用近似和不近似的方法进行探究比较

（2）在对图像进行旋转之前对图像进行缩小，为提升计算速度，缩小时使用近似的方法（INTER_AREA ），对图像进行旋转同样也使用近似的方法（INTER_AREA ），最后再将图片放大也使用近似的方法（INTER_LINEAR）

5. 实验结果与分析

5.1 实验结果

5.1.1 未改进前

（由于图像尺寸原因无法按原图片尺寸放入报告，查看各个原始图像请进入实验结果文件夹查看）

（1）原始图像

图3 原始图像

（2）图像旋转 6°

图4 旋转6°图像

（3）基于图像旋转后进行操作的结果：

① 旋转（180°）

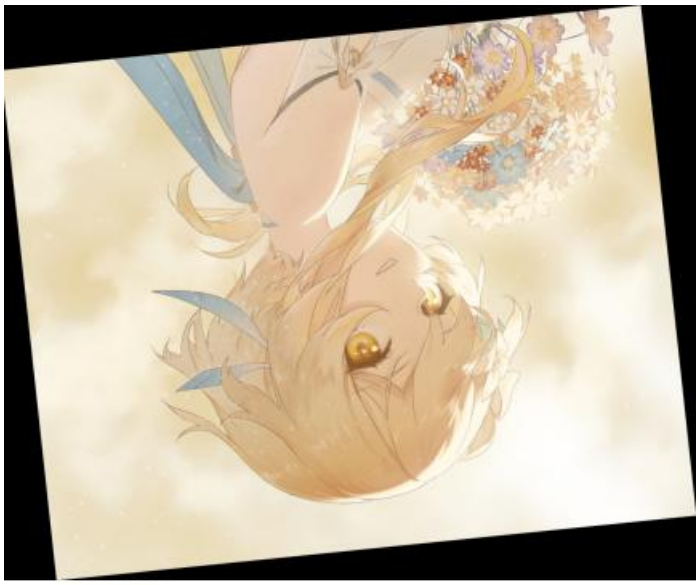


图5 旋转180°图像

② 平移

（移动距离 x=50，y=50）

图6 平移效果图

③ 缩放

- 按比例缩放

图7 按比例缩小50%

- 按尺寸缩放

图8 按尺寸缩放

5.1.2 改进后

（1）对图像进行近似方法

- 细节处对比：

未使用近似方法

使用近似方法

图9 近似方法细节对比图

- 不使用近似图片：

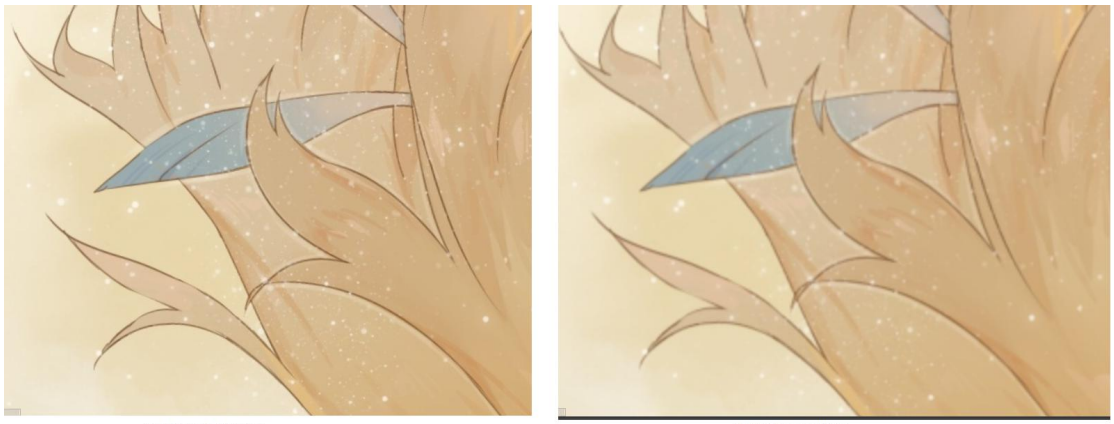
图 10 未使用近似图像

- 使用近似图片：

图 11 使用近似图像

（2）对图像进行缩放，同时使用近似方法

细节处对比：



未使用近似和缩放处理　　使用缩放和近似处理

图12 细节对比图

5.2 实验结果分析

5.2.1 未进行改进结果分析

（1）对原始图像进行一个小角度 6° 的旋转后，我们再对图像进行了 180° 的旋转，对图像进行了平移操作向右平移 50，向下平移 50，以及缩放的操作，按比例进行缩放和按尺寸进行缩放，按照预期得到了我们想要的结果。

（2）由于图像并不是很大，同时并没有使用近似的方法，图像并没有过多的损失，图像与原图像差异很小。

5.2.2 改进后结果分析和讨论

（1）使用近似后的效果分析

在细节对比图中，我们可以发现使用近似和不使用近似的效果基本一致，图像的细节被保留的较为良好。于此同时从我们对函数运行时间的检测，我们发现近似能够很好的加快计算，提升图像处理的速率。

```
The function took 0.081943s to run.
The function took 0.071572s to run.
```

图13 运行时间图

第一条为未使用近似对图像进行旋转 6° 时的时间，第二条为使用近似后的运行时间

（2）使用近似同时使用缩放情况讨论

在对图像进行缩小进行旋转，再进行放大恢复图像的大小的操作后，我们发现图像的部分细节有所丢失，图像的边缘部分出现了模糊。同时我们发现，图像总的处理时间并没有减少，因为还要对图像放大缩小进行计算，程序总的运算效率并没有得到提高。

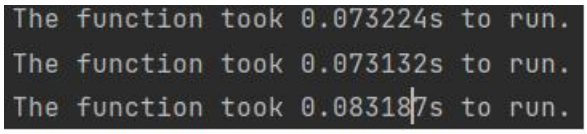


图 14 时间运行图

第一个运行时间为未使用近似和缩放操作，第二个为使用了近似，第三个为使用了近似和缩放。

6. 参考

(1) 百度百科

(2) 维基百科

(3) CMU 计算摄影课

(4) https://baike.baidu.com/item/opencv

(5) https://baike.baidu.com/item/numpy

(6) http://graphics.cs.cmu.edu/courses/15-463/

(7) 《Computer Vision Algorithms and Applications》

7. 课程总结

数字图像处理是计算机科学中的一个重要领域，它研究如何使用计算机对数字图像进行处理、分析和识别。通过对数字图像课程的学习，我们学习到了如何可以使用多种算法和工具，如滤波、边缘检测等，同时也学习到了其背后的原理与逻辑。

数字图像处理涵盖了很多领域，如医学影像分析、计算机视觉、计算机图形学等，因此具有广泛的应用价值。比如，在医学领域，数字图像处理技术可以被用于 X 光图像的诊断和治疗；在安防领域，数字图像处理技术可以被用于人脸识别和视频监控等。

通过学习数字图像处理，我们掌握了数字图像处理的基本方法和技术，如空间域和频域的处理、图像增强和去噪、形态学处理和特征提取等；还需要学会如何使用编程语言和图像处理库进行实际操作。

数字图像处理为我们提供了一种强大的工具，可以帮助我们更好地理解和处理图像信息，从而应用于各种实际场景中。