竺可桢学院
工程教育高级班

Advanced Honor Class of Engineering Education

-2023 实践面试题-

2023. 03

目录

序

数学模型是实际问题与数学理论之间的桥梁，数学模型（Mathematical Model）是针对现实世界的特定对象，为了特定目的，根据特有的内在规律，做出一些必要的简化假设，选用适当的数学工具，得到的一种数学结构，建立数学模型的过程，即为数学建模。

进入 21 世纪，以计算机为基础的信息领域快速发展，相关研究不断深入，随着计算机算力不断增强，相关基础研究不断完善，人们开始广泛采用计算机对问题进行分析讨论。在这里，我们关注现今较为火热的两个领域——数字孪生与脑网络研究。数字孪生依托于计算机而产生，是信息化的产物。而对人大脑的研究则历史已久，但随着人工智能的发展，人们对大脑结构的研究需求愈发迫切。

在这次实践面中，你们将以小组为单位，利用建模这一利器，解决一系列问题。这其中既涉及计算流体力学，也有图像处理与神经网络的相关知识。在这个过程中，你们可能会学习从未了解过的新知识，可能会运用许多全新的工具，可能需要你们进行严谨的数学推导与思维创新……但是请记住，良好的团队协作和分工是你们完成任务的最重要的保证。

良好的团队合作，规范的学术报告，完整的解答过程，严密的逻辑思维将会是本次实践面试的重点考察内容。从现在开始你和你的团队伙伴有三天（3.30-4.1）的时间阅读题目、查阅资料、解决问题、撰写报告、准备展示。**实践面试结束后，你们需要提交完整的解题报告，并展示你们的成果。**

请注意：实践面试结束后所有参加实践面试的同学将直接进入第一轮面试，但实践面试的表现将会作为总成绩的组成部分之一，是筛选的重要指标。团队可以使用他们自己发现的任何“无生命”资源，例如网页、书籍、研究报告等。但是团队不得寻求帮助，**不得从除团队成员之外的任何人处获得答案、想法或信息。**一旦发现作弊，视情况扣分，情节严重者将取消专家面试资格。

请将你们团队的成果（解题报告及其附件，如源代码、分工名单等，注意请填写提交样例中提供的贡献度表，**具体格式详见提交样例及解题报告中说明**）打包成压缩包（命名为“组号 _ 实践面”，如 A1 组的提交文件应命名为“A1_ 实践面”），并在 2023 年 4 月 1 日周六晚 23：00 前提交至邮箱：zjuacee2023@163.com(主线路),zjuacee2023_2@163.com(备用线路)，邮件主题请命名为“组号 _ 实践面”。

请注意报告和代码的规范和可读性。我们将对过时提交文件的团队进行**扣分或者取消资格**等处理。团队展示将在 2023 年 4 月 2 日周一晚，每组展示时长约为 15 分钟，具体面试时间地点待通知，展示材料（如 PPT）无需提交。

Assignment1 计算流体力学

导言

计算流体力学（Computational Fluid Dynamics，简称 CFD）是 21 世纪流体力学领域的重要技术之一，使用数值方法在计算机中对流体力学的控制方程进行求解，从而可预测流场的流动。目前在工程领域 CFD 方法已经得到广泛的应用。

使用 CFD 处理流体力学问题一般有以下几个步骤：

1. 选择一种合适的模型描述流体本身。
2. 从经典力学中的质量守恒、动量守恒、能量守恒出发，分析流体的运动状态，列出描述其运动状态的方程。这些方程被称为**流体力学的控制方程**。它们一般是偏微分方程。
3. 使用合适的数学方法将其进行离散化，将连续的微分/积分方程转化为离散的代数方程。
4. 通过编程等方式求解。

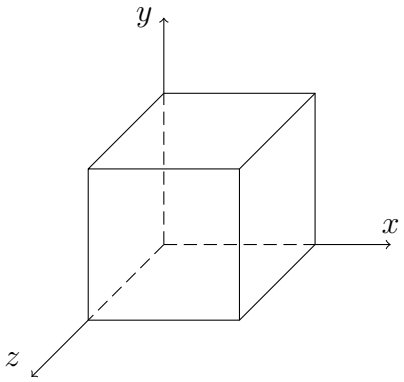
CFD 的整个过程会涉及到包括牛顿力学、微积分、线性代数、计算机科学甚至实验科学在内的庞大的知识体系。但在本题中，我们会从一些简单的模型入手，逐步建立对 CFD 方法的认识，并最终使用学习到的方法求解一个实际问题。

Part1 方程推导

在工程实际中一般有两种描述流体的方法：在流场当中使用一个闭合曲面划出封闭空间的**有限控制体法**，以及在流场当中取一个无限小体积的**无限小流体微团法**。

Question 1.1 微分形式的方程

在这个例子当中，我们考虑一个长宽高分别为 $\mathrm{d}x, \mathrm{d}y, \mathrm{d}z$ 的无穷小流体微团。如下图所示。



接下来，请你将**质量守恒、动量守恒、能量守恒**运用于上述微团，推导出如式1、2、3所示的适用于无限小流体微团的流体运动方程。

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0 \tag{1}$$

$$\begin{cases} \frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u \mathbf{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x & \text{x 方向动量守恒} \\ \frac{\partial \rho v}{\partial t} + \nabla \cdot (\rho v \mathbf{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y & \text{y 方向动量守恒} \\ \frac{\partial \rho w}{\partial t} + \nabla \cdot (\rho w \mathbf{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z & \text{z 方向动量守恒} \end{cases} \tag{2}$$

$$\begin{aligned} \rho \frac{\mathrm{D}}{\mathrm{D}t}\left(e + \frac{\mathbf{V}^2}{2}\right) = {} & \rho \dot{q} + \nabla \cdot (k \nabla T) - \nabla \cdot (p \mathbf{V}) \\ & + \frac{\partial u \tau_{xx}}{\partial x} + \frac{\partial u \tau_{yx}}{\partial y} + \frac{\partial u \tau_{zx}}{\partial z} \\ & + \frac{\partial v \tau_{xy}}{\partial x} + \frac{\partial v \tau_{yy}}{\partial y} + \frac{\partial v \tau_{zy}}{\partial z} \\ & + \frac{\partial w \tau_{xz}}{\partial x} + \frac{\partial w \tau_{yz}}{\partial y} + \frac{\partial w \tau_{zz}}{\partial z} + \rho \mathbf{f} \cdot \mathbf{V} \end{aligned} \tag{3}$$

其中

- 向量 $\mathbf{V} = (u, v, w)$ 表示流体速度
- τ_{ab} 表示粘性力，作用在垂直于 a 轴的平面上，正方向与 b 轴一致
- $\mathbf{f} = (f_x, f_y, f_z)$，$f_a$ 表示作用在单位质量流体上，指向 a 方向的体积力。
- e 表示单位质量流体的内能
- $\dot{q}$ 表示单位质量流体在单位时间内由于化学反应等原因产生/吸收的热量。
- k 表示传热系数
- $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$

上述方程被称为微分形式的 Navier-Stokes 守恒方程。

你可能需要的信息

- 质量守恒：质量的变化量 $=$ 流入质量 $-$ 流出质量
- 能量守恒：能量变化量 $=$ 内部产热 $+$ 传热 $+$ 外力做功
- 动量守恒：动量变化量 $=$ 压强产生冲量 $+$ 粘性力产生冲量
- 傅里叶传热定律：在 a 方向上，热传导为 $\Delta Q_a = -k\dfrac{\partial T}{\partial a}\Delta S\Delta t$，$k$ 为传热系数
- 高斯公式：$\displaystyle\int_V f dV = \int_{\partial V} \nabla \cdot f d\mathbf{S}$

Question 1.2 向量方程

试通过向量分析，将式2所表示的三个动量方程改写为一个向量方程，并与式1的形式进行比较，归纳出一个与具体物理表达式无关的一般形式。

你可能需要的信息

- 当 $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ 时，$\nabla \cdot A = \begin{pmatrix} \frac{\partial a}{\partial x} + \frac{\partial b}{\partial y} + \frac{\partial c}{\partial z} \\ \frac{\partial d}{\partial x} + \frac{\partial e}{\partial y} + \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} + \frac{\partial h}{\partial y} + \frac{\partial i}{\partial z} \end{pmatrix}$

Part 2 离散化

之前推导出的控制方程是连续的微分方程，为了在计算机当中进行表示与求解，必须进行离散化。

Question 2.1 有限差分法

有限差分法使用有限差分代替偏导数，将连续的微分方程转化为代数方程，可以使用计算机求解了。

先从一个简单的例子看起，对在 i 处速度 x 方向分量 u，在 x 方向上泰勒展开，有

$$u_{i+1} = u_i + \left(\frac{\partial u}{\partial x}\right)_i \Delta x + o(\Delta x) \tag{4}$$

我们可以从式4当中得到偏导数

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_i}{\Delta x} + o(\Delta x) \tag{5}$$

注意末尾的 $o(\Delta x)$。当用差分 $\frac{u_{i+1}-u_i}{\Delta x}$ 表示偏导数 $\left(\frac{\partial u}{\partial x}\right)_i$ 时，我们称它是具有**一阶精度**的。这种用 $i+1, i$ 处表达的方法称为**向前差分**，同理，用 $i, i-1$ 表达则被称为**向后差分**。

类似地，在 u_{i+1} 和 u_{i-1} 两处展开到 $o(\Delta x^2)$，用 $\frac{u_{i+1}-u_{i-1}}{2\Delta x}$ 表示时其具有**二阶精度**，并以此类推。这种表达方式是**中心差分**的一种。

现在，请你试着使用上面的方法，用 $u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}$ 表示 $\left(\frac{\partial^2 u}{\partial x^2}\right)_i$，且具有四阶精度。

Question 2.2 时间推进求解

对于一个具有时间导数的双曲型偏微分方程，时间推进求解是一种很自然的方法。其基本思想同样可以用泰勒展开式说明

$$u_i^{t+\Delta t} = u_i^t + \left(\frac{\partial u}{\partial t}\right)_i^t \Delta t + o(\Delta t) \tag{6}$$

其中对时间的偏导数可以通过待求解的偏微分方程，经过离散化后计算得到。取前两项，不断重复“推进”的过程，就可以得到偏微分方程的在不同时刻的数值解。

但是鱼与熊掌不可兼得，要想兼顾展开求解的精度与相对较小的计算量是很困难的事情：根据一节的讨论，展开高阶精度的偏导数会带来很复杂的计算，一方面这大大增加了计算量，同时也给我们编写求解程序带来很多不方便。

附件中的参考文献给出了一种兼顾效率与精度的求解方法。请阅读参考文献，用不多于 500 字解释以下几点（公式不计入字数）

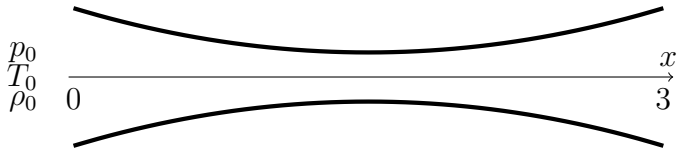
- 计算时间导数的方式
- 算法的基本求解步骤
- 算法的时间、空间精度

如果可以的话，试着证明其时间精度，并思考算法是否具有改进空间。这两个问题不是强制的，不计入字数，但请大胆将你的想法写在报告当中。

Part 3 流场求解

Question 3.1 一个计算例子

在学习了上述描述与方法之后，就可以来试着编写代码求解一个例子了。如下图所示，存在一个喷管，两侧具有压强差。



在完整情况下，这应该是一个二维问题，在这里我们简化为一个一维问题进行考虑。

我们不加推导地给出[1]这个问题的控制方程。

$$\frac{\partial \rho}{\partial t}+\rho\frac{\partial v}{\partial x}+v\frac{\partial \rho}{\partial x}+\rho v\frac{\partial(\ln A)}{\partial x}=0 \tag{7}$$

$$\frac{\partial v}{\partial t}+v\frac{\partial v}{\partial x}=-\frac{1}{\gamma}\left(\frac{\partial T}{\partial x}+\frac{T}{\rho}\frac{\partial \rho}{\partial x}\right) \tag{8}$$

$$\frac{\partial T}{\partial t}+v\frac{\partial T}{\partial x}=-(\gamma-1)T\left[\frac{\partial V}{\partial x}+v\frac{\partial(\ln A)}{\partial x}\right] \tag{9}$$

其中

- A 为喷管的截面面积，在这个例子当中取 $A=1+2.22(x-1.5)^2,(0\le x\le 3)$
- γ 为与气体性质有关的常数，这里取 1.4
- 入口处 $p_0=1,\rho_0=1,T_0=1$

你可能需要的信息

1. **边界条件**：在使用有限差分时，在出口和入口处会存在需要的网格点不存在的情况。这个问题有很多解决方式，这里介绍一种线性插值法，即认为 c_1,c_2,c_3 和 c_{n-2},c_{n-1},c_n 在一条直线上。你也可以查阅相关资料采用其他的处理方式。请务必在报告中完整介绍你的处理方式。

2. **时间步长**：根据 *CFL* 稳定性条件[2]，要获得一个收敛的结果，有 $\Delta t=C\dfrac{\Delta x}{v+\sqrt{T}}$，$C$ 为科朗数，小于 1。

3. **初始值**：理论上初始值可以任取，但为了计算方便一般取一个与实际情况较为一致的值。可以取 $\begin{cases}\rho=1-0.3146x\\ T=1-0.2314x\\ v=(1+1.09x)\sqrt{T}\end{cases}$

现在，请你使用有限差分法与上一节提到的方法编写程序，求解上述问题。请注意以下几点：

[1]如果对式子存在疑惑，我们很欢迎将你的思路与问题呈现在解题报告中

[2]可以查阅资料了解更多细节

1. 将你的算法、边界值处理、$\Delta x, \Delta t$ 的选择等细节完整地在报告中描述。
2. 将 1200 个时间步后的结果填写在附件中的 results.xlsx 当中。
3. 将源码等材料放在压缩包中提交。语言没有要求，但原则上注释不少于 20%。所有代码都将严格查重。
4. 试着使用使用多维曲线、动画等方式将结果可视化，在最后的答辩环节进行展示。

Question 3.2 控制方程

不加证明是一个令人难受的词语，那就上手推导一下方程7、8、9。

明确一下这个问题的物理性质：喷管当中的流动是无粘（不存在粘性力），等熵（不存在流体本身产热，不存在体积力）的。同时，我们将其简化为一个一维问题，因此只需要考虑 x 方向的情况。

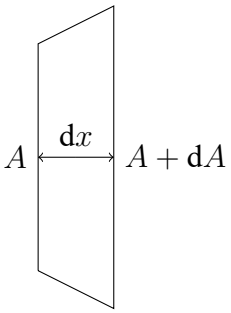
另外，注意到题目当中给出的入口边界条件均为 1，这是因为式7、8、9当中的变量均为**无量纲变量**，即参与计算的变量数值均为原始变量与入口初始值的比值。具体来讲，有

- $T = \dfrac{T_{原}}{T_0}$
- $\rho = \dfrac{\rho_{原}}{\rho_0}$
- $v = \dfrac{v_{原}}{a_0}$，其中 a 为当地声速，计算式为 $a = \sqrt{\gamma RT}$，另有 $a_0 = \sqrt{\gamma RT_0}$

另有关系 $c = R/(\gamma - 1)$，c 为气体比热容。使用无量纲变量可以在一定程度上减小量纲不同造成的数值误差，给我们的计算带来方便。

你可能需要的信息

- 必要时，尝试将中推得的方程两侧对体积积分。这种处理方式称为积分形式的方程。两者在数学上是等价的。
- 如果对问题的理解有困难，请参考下面的控制体



Question 3.3 向更实际的问题前进

之前的算例并没有限定出口处的任何条件，换言之，三个物理量都是“自由摆动”的。现在我们将出口处的压力 p 固定为 0.9，其余两个物理量依旧保持自由摆动。

可以尝试用之前的方法进行求解。如果求解失败（出现奇怪的数值等），试着查阅相关资料，从物理性质等角度解释求解失败的原因，并尝试通过其他方法求解本问题，得到流场变量的值。如果求解成功，请尝试改进。

请在报告中完整介绍你们对算例的分析、采用的新求解方式以及最终得到的结果。如果可以的话，请试着使用多维曲线、动画等方式在展示当中进行可视化

你可能需要的信息

- 无量纲条件下的理想气体方程：$p = \rho T$
- 初始条件、使用的变量形式等都可以成为改进的方面

Bonus

上面的题目介绍了从基本物理学原理到解决实际问题的一个过程。但在目前的工程实践当中，已经有很多可以使用的流体力学软件帮助我们完成计算的步骤。OpenFOAM 就是其中的一个开源工具包。

附件当中给出了与问题类似的一个算例 shockTube。请查找相关资料[1]使用虚拟机，WSL 或装有 Linux 系统的实体机完成 OpenFOAM 与 ParaView 软件的安装，学习基本操作并使用 blockMesh 建立网格，rhoPimpleFoam 对算例将进行计算。将运行过程与结果以截图的形式附在报告中。请注意：最好不要在含有中文路径的文件夹中打开，可能会出现意想不到的问题。

完成这个部分需要花费较多的精力，即便没有得到最终结果，我们也鼓励你将完成的进度与遇到的问题写在报告当中。另外，计算这个算例可能需要较长的时间。算例中已经附上了 *decomposeParDict*，如有需要可以查找资料学习 *decomposePar* 的使用方式，开启并行计算以加快计算速度。

[1]这里给出两个例子：
如何在 wsl 上优雅地使用 OpenFOAM - CC98 论坛，OpenFOAM for Windows 10 | OpenFOAM

Assignment 2 图像处理与神经网络

Part1

数字图像的量化与特征提取

Question1 数字图像的量化

下面的任务将围绕数字图像展开。俗话说“知己知彼，百战不殆”，在具体任务开始前，我们非常希望你认真阅读下方 Wiki 给出的关于图像的定义：

图像是人对视觉感知的物质再现。图像可以由光学设备获取，如照相机、镜子、望远镜及显微镜等；也可以人为创作，如手工绘画。图像可以记录、保存在纸质介质、胶片等等对光信号敏感的介质上。随着数字采集技术和信号处理理论的发展，越来越多的图像以数字形式存储。因而，有些情况下“图像”一词实际上是指数字图像。

正如图像定义中描述的那样，我们有非常多获取图像的方式，而随着移动设备的快速发展，即使没有专业的摄影设备，手机的相机也逐渐可以服务于一定的摄影需求。你一定留意过，过去手机厂商的广告语中总会用“像素大小”来标榜相机的好坏，而这一数字的进步总是飞快的，几年前就已经出现了惊人的“一亿像素”，虽然像素高并不意味着相片的质量就好，但更多像素可以保证图像有着更高的细节表现力。

但请不妨思考一下，是否所有场景我们都追求“细节”？答案是否定的，高细节意味着更大的存储和更长的传输时间，网络浏览的图片如果每一张都细节饱满，校园网似乎难以支持大家在网络中畅游。

有很多方法可以节省图片的存储空间，我们这里选用最简单的方式：减少使用颜色的种类。如果你对 RGB 颜色空间有一定了解，你的脑海中应该有下面这个立方体：

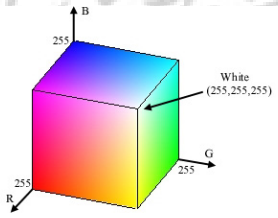


图 1: RGB 空间

图片中颜色分布似乎是连续的，但注意我们处理的是数字图像，R、G、B 三个通道的色彩值是通过离散的整数表示的。因此，在图像处理问题中，将立方体视作 $256 \times 256 \times 256$ 个小立方体的堆叠更为准确，每个小立方体代表了 RGB 空间中能够表示出的一个具体颜色。几个最基本的颜色在 RGB 空间表示为：

黑色	(0, 0, 0)
白色	(255, 255, 255)
红色	(255, 0, 0)
绿色	(0, 255, 0)
蓝色	(0, 0, 255)

不难算出，RGB 空间实现了 $256^3 = 16777216$ 种色彩，如此大量的颜色为艺术家描绘这个世界提供了丰富的资源，但在对于“细节”没有那么重视的情形下，千万级的色彩似乎显得有些“冗余”。

为了能够操作图像，我们需要再进一步明确之前提到的**像素**的概念。一幅图片通常由数以千计的像素构成，像素排列成矩阵，图像中的每一个像素代表一个特定的颜色。下面展示了一幅 8×8 像素的小图像：

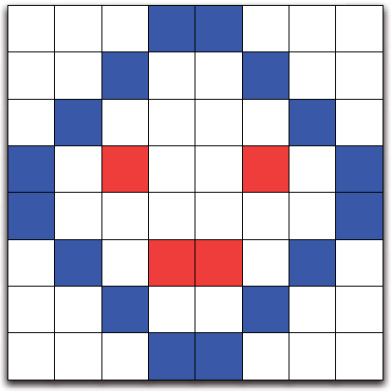


图 2: 像素图像

如果将每个像素视作一个结构体变量或对象实例，通过编程语言保存一张图片的方式有很多，在 Question1 中，使用最简单的二维数组即可（Python 可以等价的采用嵌套列表）。

- **Question 1.1**

  如果你足够敏感，应该已经快速反应出了一个像素需要占据三个字节的存储空间，我们通常称这种方式为 RGB888（每个颜色占据 8 个比特位）。但在开头我们已经明确，我们的目的是通过减少使用的色彩来节约空间，将颜色使用的数目降低的过程我们称作**量化**（这里的量化是指对于颜色数目降低采用的方式，需与图像量化或其他概念加以区分）。请合作查阅.bmp 文件格式和 RGB555 与 RGB565，分析两种 RGB 格式对于颜色空间的改变，理解两种方式可以被称作 RGB888 **简单量化**的原因以及简单所带来的弊端，并基于你们发现的弊端，设计一种改良的量化算法（需要异于第 2 小问）。最后请使用你们所熟悉的编程语言编写两段代码，其中一段用于实

现对于 A2P1Q1_01.bmp 的 RGB555 简单量化，另一段实现你们所设计的算法，将结果分别与原图片做效果与内存的对比。需要注意在实现你们设计的算法时，请将颜色数目的减少转化为实际的内存降低，为此，需要去认真了解.bmp 文件的调色板。

- **Question 1.2**

第 1 问中你们已经明确 RGB555 可以节约大致 $\frac{1}{3}$ 的存储空间，也可能已经实现了某种不错的量化算法，而在这一问中，我们会提供一种新的量化视角。在告知具体任务之前，先介绍一种在三维空间中十分常用的数据结构——**八叉树**。树状结构在计算机中十分常见，无论是 Unix 还是 Windows 操作系统，其文件目录都可以视作一棵从根目录展开的树。八叉树和最为常见的二叉树一样，是一种子节点数目固定的树状结构，不妨来看看 Wiki 给出的定义：

八叉树（英语：octree）是一种树形数据结构，每个内部节点都正好有八个子节点。八叉树常用于分割三维空间，将其递归细分为八个卦限。

百科中配的这张图可能可以帮助你直观的认识八叉树，并理解三维分割：

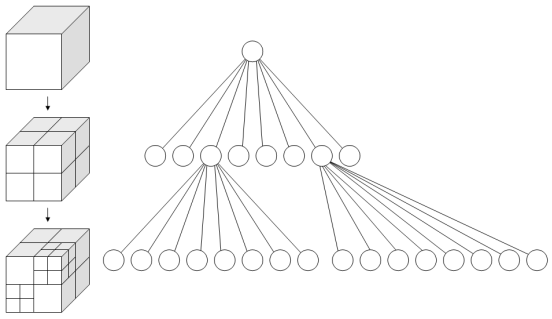


图 3: 八叉树

左图是对应的是被分割的三维空间，右图对应着相应的八叉树。有没有察觉到小方块的堆叠和之前提到的 RGB 空间的理解很接近，或许可以利用八叉树对于 RGB 这个三维空间做分割。第 2 问的任务描述十分简单，请小组成员集思广益，讨论如何用八叉树分割 RGB 空间，并设计一种算法使用尽可能少的颜色保留尽可能完整的图像信息。我们非常看重你们的创新性，小组讨论往往可以产生不错的想法。还需要注意的是利用八叉树实现时，应尽可能避免之前算法出现的弊端。当一切就绪，请仍对 A2P1Q1_01.bmp 编程实现属于你们的八叉树优化的量化算法。

- **Question 1.3**

量化图片通常存在一个后处理操作——**抖动**。抖动是指将不同的颜色靠近，让眼睛混合这些颜色，形成一张更“真实”的图片，是印刷和墨水屏通常都会采用的技术。来看看 Wiki 中给出的例子：

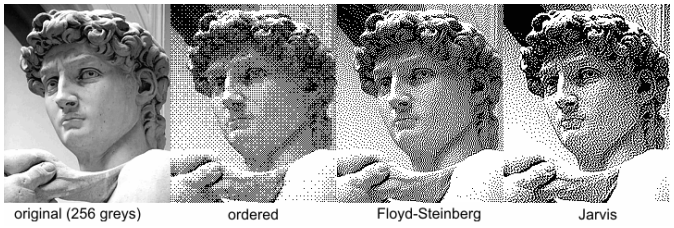


图 4: 抖动

最左侧是 256 色的灰度原始图像，右侧三张分别对应了三种不同抖动算法的结果，如果不告诉你，你也许不会相信右侧的三张图像仅用了黑白两种色彩。这就是抖动的精妙之处，它通过一个“小把戏”欺骗了你的大脑，如果心存怀疑，不妨放大附件中的图像来检查右侧三张图片是否只使用了黑白色彩。我们不去深究其中的生物学原理，但可以给出一个便于你们接受这个“神奇的小把戏”的例子：

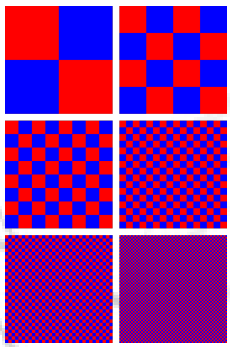


图 5: 彩色抖动

看到第三行的两幅图片靠近紫色或许就可以大致看透这个“小把戏”了，此时也应该可以理解为什么抖动会作为量化的后处理了。出于简单考虑，在这一问中，无需后处理之前量化的结果，你们只需要将 A2P1Q1_03.bmp 这张彩色图像转化为灰度图像，再通过抖动的方式转化为二值图像。为了得到灰度图像，另一个颜色空间 YUV 需要了解，并在解答中阐述其和 RGB 空间的区别以及给出如何得到 RGB 空间中的某个颜色在 YUV 空间中的表示。在得到灰度图像后，请查阅抖动的各种算法，在了解利弊后讨论出最适合给定图片的一种，或许你们还可以想出一些针对已有算法不错的优化，请介绍你们的算法，并通过编程语言实现它（代码实现包括将彩色图像转化为灰度图像）。过程中如果觉得黑白图像使用一个比特位来表示颜色较难操作，我们也允许使用只有 0 和 255 两个值的灰度图像加以代替。

提示：在 Question1 中我们非常推荐你们使用 C 语言，但其他编程语言也

是可接受的，不被接受的是调用任何现有的图像处理库和将 bmp 文件转化为其他格式的文件，换言之，操作应该“深入”bmp 文件的二进制，而不是将图像整体作为一个对象调用现成的方法。请通过至少 25% 的注释向我们展示你们对于每一步的理解，代码将被严格查重。除此之外，可视化的展示一些结果总是更受欢迎，并且我们不会限制你们展示结果的方法和工具。

Question 2 特征提取

在了解了有关数字图像量化的内容后，在这一部分你将对特征提取的有关内容进行了解。特征提取是计算机视觉和图像处理中的一个概念。它指的是使用计算机提取图像信息，决定每个图像的点是否属于一个图像特征。特征提取的结果是把图像上的点分为不同的子集，这些子集往往属于孤立的点、连续的曲线或者连续的区域。特征是一个数字图像中“有趣”的部分，它是许多计算机图像分析算法的起点，其提取效果的好坏将直接影响到后续的处理效果。在这里，你将接触三种经典的图像特征提取和表达方法，请查阅 ref 文件夹中的有关资料进行了解，并完成以下任务。对于需要通过编程解决的问题，语言不限。

- **Question 2.1**

  SIFT(Scale-Invariant Feature Transform，尺度不变特征转换) 是一种著名的视觉算法，用于检测与描述影像中的局部性特征，它在空间尺度中寻找极值点，并提取出其位置、尺度、旋转不变量，在计算机视觉领域得到了广泛应用。阅读附件中的 Distinctive Image Features from Scale-Invariant Keypoints 进行了解，并回答以下几个问题：

  1. 请用自己的语言概括 SIFT 描述子的构建过程（答案在 200 字以内）。
  2. SIFT 算法去除边缘响应的过程中用到了 Hessian 矩阵。请导出某一尺度下图像 (x, y) 处在 $\vec{V}$ 方向的二阶导数 D_{vv} 关于 Hessian 矩阵 H 的表达式，并给出 D_{vv} 的范围。

- **Question 2.2**

  SURF(Speeded Up Robust Features，加速鲁棒特征) 是在 SIFT 基础上发展出来的一种更加稳健的图像识别和描述算法。在计算过程中，SURF 对高斯二阶微分进行离散化和裁剪处理得到盒式滤波器来近似代替高斯滤波模板进行卷积计算，大大加快了计算速度。盒式滤波 (Box Filtering) 是一种简单的线性滤波技术，作为一个基础函数，其速度的快慢也直接影响着整个算法的性能。

  在本题中，我们希望你能实现一个简单的、非归一化的盒式滤波器，要求卷积核大小为 3×3，并基于积分图像对该盒式滤波器进行性能改进，从而

加速计算，同时希望你能对你所实现的各版本盒式滤波算法进行简要的原理说明与复杂度分析。附件中有一张彩色图片 test.jpg，请在完成上述任务后使用你改进后的盒式滤波器对该图片进行滤波处理并提交结果。

要求：请查阅资料对有关概念进行了解；说明你所完成的盒式滤波算法的实现原理与改进思路，同时进行算法复杂度分析，填写在解题报告的相应位置；请将基础版本与改进版本滤波器的源代码一并放在附件内提交，并在关键部分进行注释，要求注释率大于 25%（请避免直接使用对应的算法 API）；将处理后的图片命名为 box_filtering.jpg，与处理代码一同提交于附件中。

- **Question 2.3**

SIFT 与 SURF 已经足够优秀，但它们仍然是慢了点。2011 年，一篇名为“ORB：An Efficient Alternative to SIFT or SURF”的文章发表，开启了一个新的篇章。ORB(Oriented FAST and rotated BRIEF) 以计算速度快著称，因其实时性好而在工业领域得到广泛应用。ORB 之所以快速，是因为其“站在了巨人的肩膀上”：ORB 特征将 FAST 特征提取方法与 BRIEF 特征描述子结合起来，并在它们原来的基础上做了改进与优化。ORB 对 FAST 的改进使其克服了方向缺失等问题，并有效地解决了特征点数量过多的弊端。

在本题中，请你首先对 FAST 特征点检测算法进行实现；接着请你查阅资料，了解 Oriented FAST(即 ORB 中使用的 FAST 算法) 是如何对 FAST 算法进行改良以控制 FAST 特征点数量的，并据此对你所实现的 FAST 算法进行改进。附件中有一张彩色图片 test.jpg，请在完成上述任务后使用你改进后的 FAST 算法对该图片进行特征点的提取并提交结果。

要求：请查阅资料对有关概念进行了解。请说明你的改进思路，并解释改进后的算法是如何准确控制特征点数目的，填写在解题报告的相应位置；请将基础版本与改进版本的源代码一并放在附件内提交，并在关键部分进行注释，要求注释率大于 25%（请避免直接使用对应的算法 API）；将处理后的图片命名为 fast_feature.jpg，与处理代码一同提交于附件中。

- **Question 2.4**

通过完成以上题目，相信你对这几种图像特征提取算法已经有了初步的认识。而接下来，我们希望你能够从实际应用效果上对它们进行进一步的了解。你将通过完成图像的特征匹配任务来认识它们在工作中的性能差异。

特征匹配是一项底层视觉处理技术，直接对图像本身进行特征提取与配对，是许多具体大型视觉任务的首要步骤，其连接着具有相同或相似属性的两个图像目标。基本流程如下：对于两幅图片，我们在对其分别进行图像特征提取和描述后，得到了图像中具有物理意义的显著结构特征，而后在某

一准则下对两幅图片中的特征进行比对，从而确定两幅图片中特征的对应关系，并由此进行特征点的配对，获知更高层的视觉信息。以下是一个简单的例子：

我们希望大致确定 box 在 box_in_scene 中的位置，于是将两图进行特征匹配：

图 6: box

图 7: box_in_scene

通过对图中的特征点进行检测与比对，最终得到如下的结果：

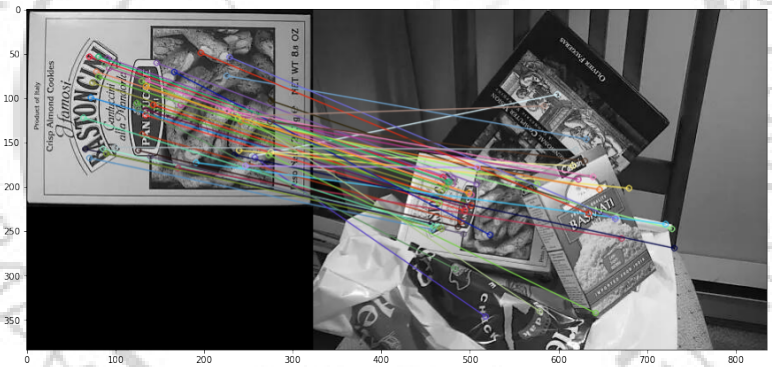


图 8: output

就特征点比对匹配过程而言，暴力法（Brute Force）与快速最近邻搜索包（FLANN，Fast Library for Approximate Nearest Neighbors）等都是可行的算法。请自行选择并完成以下任务。

请基于附件中提供的图片 src.jpg、dst.jpg，自行查询资料并设计程序，分别使用 SIFT、SURF 和 ORB 对两张图片进行图像特征提取，而后完成基于特征的图像匹配任务，并通过这一实验比较 SIFT、SURF 和 ORB 三种算法的性能差异（计算速度、鲁棒性等）。

要求：请对你所选择的特征匹配算法进行简要说明，自行选择指标对三种算法的性能进行评估，将测试方法、分析过程与实验结果填写至解题报告的相应位置；请将源代码放在附件内提交（如需要请对实验环境进行必要说

明）并在关键部分进行注释，要求注释率大于 25%；将三种算法提取特征并进行匹配后得到的结果以图片形式进行保存，分别命名为 sift.jpg、surf.jpg、orb.jpg，**与处理代码一同提交于附件中。**

注：

1. *OpenCV* 是一个成熟的开源计算机视觉库，其中对多种图像处理方法进行了封装。在本题中，你可以使用有关 *API* 完成任务。
2. 除了对附件中原有图片的特征匹配外，为了探究三种算法的性能差异，你可能需要对附件中图片进行某些处理后进行多次图像特征匹配以得到实验数据，请将这部分内容作为实验过程在实验报告中进行展示，不必将结果在附件中进行提交。

Part 2

卷积神经网络入门

卷积神经网络（Convolutional Neural Networks, CNN）在计算机视觉（Computer Vision, CV）中发挥着重要的作用。在前面的任务中，我们试着用计算机来处理图像。在本节中，我们将尝试着让计算机看懂图像。

本题将借助 Python 下的科学计算工具 NumPy 实现一个简单的 CNN 模型。

Question 1 Numpy 与神经网络

NumPy 作为 Python 中科学计算的最基础的工具包，通过 `np.ndarray` 封装数组并提供了一系列的接口，以此对 Python 中的数学计算进行向量化操作，从而加速 Python 的科学计算效率。

在本题中，你将跟随着将附件中的 `guidance.ipynb`，初步认识 NumPy 以及神经网络，将附件中 `start_code.c` 文件给出的函数改写为 Python 函数，并对代码进行向量化。最后，比较 C 函数、Python 函数以及 NumPy 向量化后的 Python 函数三者的计算速度，并尝试着解释它们速度出现差异的原因。

请将包含改写后函数的 Python 文件命名为 `A2P2Q1_solution.py`，提交于附件中。若使用代码来测试三种函数的耗费时间，请将测试所用到的相关代码（命名如 `A2P2Q1_test.py`）一并提交。

Question 2 卷积神经网络

- **Task 1**

用 $x^{[l+1]} = x^{[l]} \bigotimes w^{[l]} + b^{[l]}$ 表示第 l 层所做的卷积操作，$w^{[l]}$ 为该层的卷

积核。若已知 $x^{[l+1]}, x^{[l]}, w^{[l]}, b^{[l]}$ 与 $dx^{[l+1]}$，且其形状分别为：$x^{[l]}(C,H,W)$, $w^{[l]}(O,C,K,K)$, $x^{[l+1]}(O,H,W)$。为方便起见，假定卷积操作中步长为 1，且无 padding 操作。

请根据上述公式，推导 $dw^{[l]}, db^{[l]}, dx^{[l]}$ 在卷积层中反向传播时的计算公式。在完成数学推导后，补全附件 `start_code.py` 中卷积层的前向传播与反向传播函数的 Python 代码。

- **Task 2**

在 guidance 中，我们已经认识了针对二元分类的简单损失函数的计算以及反向过程。针对多分类问题，我们常常采用交叉熵来作为损失函数：

我们用最终输出的 K 维向量表示 K 个类别各自的预测值。因此，我们需要将模型输出的 K 维向量过一层 Softmax 函数，将输出归一化。若 $Softmax(\vec{x}) = \hat{y}$，则：

$$\hat{y}_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

通常，为了防止溢出，我们先将 $\vec{x}$ 中的每个元素都减去 $\vec{x}$ 中元素的最大值。在得到 $\hat{y}$ 即输出的预测值后，我们将其与期望输出 y 做对比，使用如下交叉熵损失函数：

$$\mathcal{L}(\hat{y}, y) = -\sum_i y_i \ln \hat{y}_i$$

若已知 K 维向量 x, y，请推导 dx 的计算公式。并由此补全附件 `start_code.py` 中计算交叉熵的函数代码。

请将所完成的 Python 文件命名为 `A2P2Q2_solution.py`，提交于附件中。

Question 3 搭建简单神经网络

在经历了以上两问后，我们完成了卷积神经网络部分基础网络层的搭建。在本题中，我们将继续搭建其余的基础网络层，并用这些网络层搭建一个用于扑克牌分类的简单神经网络。

如下图所示的是最经典的 LeNet-5 模型，其中 5 的含义是该模型中卷积层使用大小为 5×5 的卷积核。对于输入为 32×32 的图像输入，先后通过两组卷积层与池化层后，再经过三层全连接层得到输出。将输出与每组图像的标签做比较，并进行反向传播梯度下降。而在本题中，我们给出的图片大小为 $40 \times 20 \times 3$。请你参考 LeNet-5 的模型，自己搭建一个深度学习模型，对给出的图片数据进行

分类。在我们提供的数据集中，包含 20 种分类，每种分类共 3000 张图片，一共 60000 张图片。

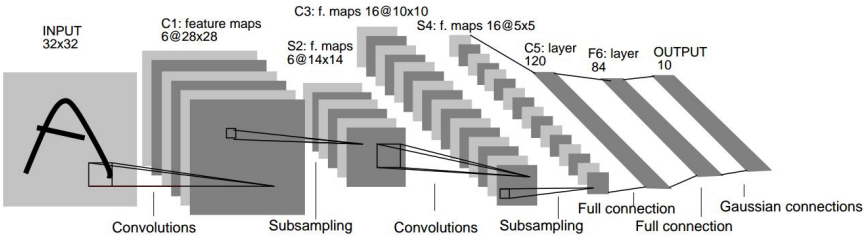


图 9: LeNet-5

具体地，你需要先完成附件中的 `nn.py` 代码，之后完成附件中给出的 `train.py` 代码，搭建自己的神经网络，并对其进行训练。请注意，附件中给出的代码及其注释仅供参考，你可以选择沿着自己的思路重新编写代码，或是对已有的代码进行修改。在进行训练时，最好使用 matplotlib 或 tensorboard 等工具来可视化损失曲线与准确率曲线。

我们将会根据模型的参数规模、代码的向量化程度以及在训练同规模数据时模型的损失来综合评估本题的分数。

请将最终完成的 `nn.py`、`train.py` 等训练相关的 Python 代码以及在本机上的训练结果截图（命名为 `A2P2Q3_result.*`）于附件中提交。

注意：

1. 您可以使用 *Tensorflow*、*PyTorch* 等封装库完成本题，但会扣除一定的分数。
2. 需要对代码进行必要的注释，注释率最好不低于 25%。
3. 本题代码将会进行严格查重，如果参考了网络上的代码请注明出处，并体现出你的理解，否则一经查出视为抄袭。

重要提示

提示：Assignment 2 所有文件请按照如下结构打包提交：

- Part1
  - Q1
    - Q1.1
      - code

        A2P1Q1_01.bmp
    - Q1.2
      - code

        A2P1Q1_02.bmp
    - Q1.3
      - code

        A2P1Q1_03.bmp
  - Q2
    - Q2.2
      - box_filtering.jpg

        code
    - Q2.3
      - fast_feature.jpg

        code
    - Q2.4
      - sift.jpg
      - surf.jpg
      - orb.jpg,code
- Part2
  - Q1
    - A2P2Q1_solution.py
    - A2P2Q1_test.py
  - Q2
    - A2P2Q2_solution.py
  - Q3
    - nn.py
    - train.py
    - A2P2Q3_result.jpg