PEKING UNIVERSITY

高等实验流体力学大作业

College of Engineering 2001111690 袁磊祺

June 21, 2021

代码和文档可以在 https://github.com/circlelq/Advanced-Experiment-Fluid-Mechanics 查看.

PIV 课程设计

课程设计目标

完成一个二维 PIV 技术的基本系统设计, 对图像处理、基本算法有初步、较为清晰的理解.

课程设计内容

设计内容主要涉及以下内容:PIV 原始图像的获取、PIV 图像的计算框架、PIV 图像相关算法、二维 PIV 数据可视化

课程设计中的若干问题说明

1. 原始图像的获取: 推荐三种建立原始图像的路径:
   (a) 图像仿真方法: 随机建立粒子像的中心坐标, 形成高斯光斑, 确定光斑光强, 按照 3*3 像素或 5*5 离散, 确认粒子数密度 (建议达到 0.1 ∼ 0.2PPP(Particle Per Pixel), 按照某种流动规律, 生成第二幅曝光粒子图像. 可以考虑加入随机噪声.
   (b) 简易 PIV 系统: 利用手机摄像机, 对一个慢速运动成像, 如: 水盆中漂浮的缓慢运动的泡沫颗粒.
   (c) 实验室装置: 可以自行设计一个简易模型, 利用实验室 PIV 设备进

行测试, 获取图像.

2. PIV 图像处理和计算: 了解图像格式、图像前置处理、相关算法、相关峰值的亚像素拟合计算、偶然误差处理.

3. 二维 PIV 数据可视化: 利用 TECPLOT 形成速度矢量场、梯度场等.

数据获取

我和王博源同学一起去圆明园校区进行了实验（之后的报告和数据处理是独立完成的）, 实验装置是一个水槽, 水槽内的水均匀流动. 水槽下方有一束激光向上照射, 打在水中的示踪粒子上形成点状图像, 并用相机从水平方向上进行拍摄, 如图 2.1 所示, 为拍摄样片, 可以发现白色的粒子点非常清晰, 背景没有杂色, 所以这里省去了图片预处理.

图 2.1. 拍摄样片.

采样频率为 2000Hz. 通过尺子进行标定可得 $(219, 28)$, $(201, 633)$ 两个像素点间的距离是 20mm. 所以, 两个相邻像素点的距离是

$$h = \frac{20\text{mm}}{\sqrt{(219-201)^2 + (28-633)^2}} = 0.0330\text{mm}. \tag{2.1}$$

数据处理

图 2.1 的格式是`.tif`, 标签图像文件格式（Tagged Image File Format, 简写为 TIFF）是一种灵活的位图格式, 主要用来存储包括照片和艺术图在内的图

像. 通过 Matlab 的 `imread()` 读取图片.

将 interrogation areas 的宽度设为 20, 在上下左右 10 个像素点的距离计算相邻两帧图片的二维相关函数

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \bar{A}\right)\left(B_{mn} - \bar{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \bar{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \bar{B}\right)^2\right)}}, \tag{3.1}$$

画图可得图 3.1. 可以发现相关性较大的点非常明显, 所以之间将相关系数最大的点作为粒子团新的坐标点, 并计算移动的位置, 记录在 `vec` 数组里.

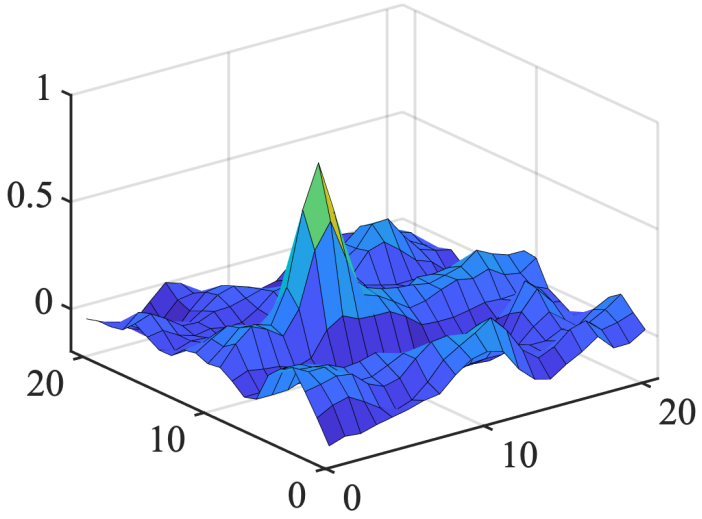


图 3.1. 附近点的相关函数.

如图 3.2 所示, 流场为均匀场, 经过一个时间点后移动的位置为三个像素点. 流速为

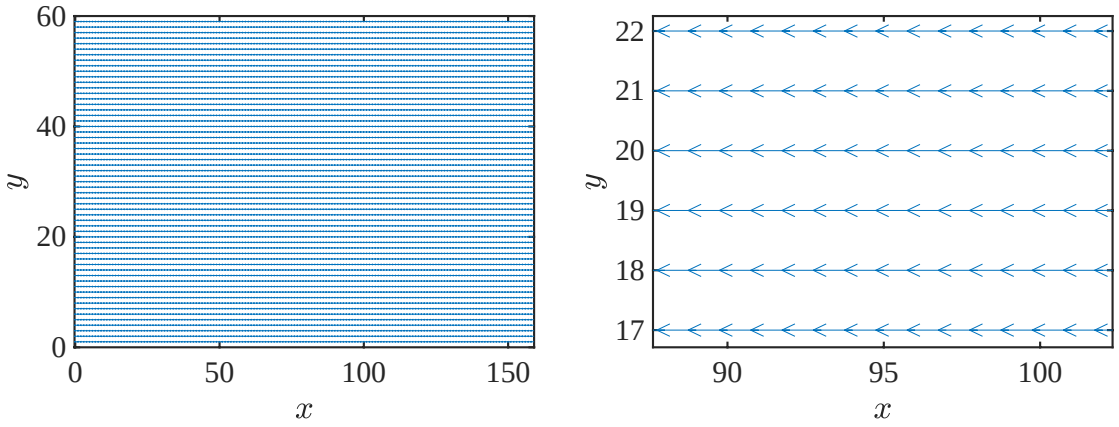
$$u = \frac{3h}{\tau} = \frac{3 \times 0.0330\text{mm}}{1/2000\text{s}} = 20\text{cm/s}. \tag{3.2}$$

代码

```
I1 = imread('001.tif');
I2 = imread('002.tif');

I1 = I1(1:100, 1:200);
I2 = I2(1:100, 1:200);

width = 20; % the width of interrogation areas
```



(a) 流场向量场.　　(b) 流场向量场放大.

图 3.2. 流场向量场.

```
d = ceil(width/2);
l = 10; % search interrogation areas no longer than l

[m, n] = size(I1);

vec = zeros(m-2*l-2*d-2, n--2*l-2*d-2, 2);
corre = zeros(2*l+1, 2*l+1);

for i = l+d+1:m-d-l-1
   for j = l+d+1:n-d-l-1
      for p = -l:l
         for q = -l:l
            % calculate correlation
            corre(p+l+1, q+l+1) = corr2(I1(i-d:i+d, j-d:j+d
                ), I2(i+p-d:i+p+d, j+q-d:j+q+d));
         end
      end
      % surf(corre)
      [M, index] = max(corre);
      [M1, index_j] = max(M);
      index_i = index(index_j);
      vec(i-l-d, j-l-d, 2) = index_i-l-1;
      vec(i-l-d, j-l-d, 1) = index_j-l-1;
   end
end
```