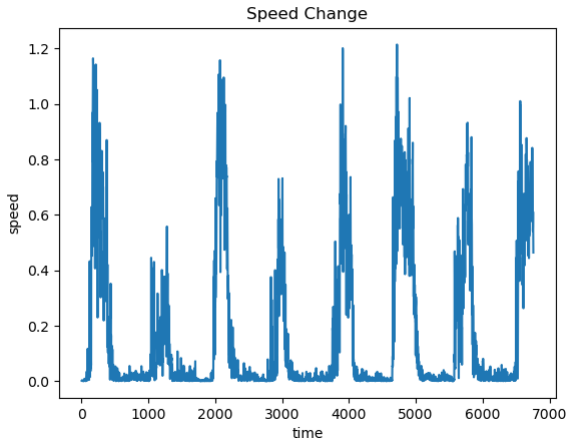
流场数据集处理

520030910182 马志强

数据清洗

首先做的就是把数据转化为向量，通过文件读取，将第7维数据抽取出来，并保存，同时选取第一个位置，绘制速度曲线，结果如图：



具体实现见代码文件DataClear.py，这里不多赘述。

小波分析

实现

首先读取我们已经获取的第一张图的数据向量，注意由于图片数据读取的坐标系和python的imshow的坐标系不同，所以要将数据向量的第一位进行翻转，便于图片的复现。

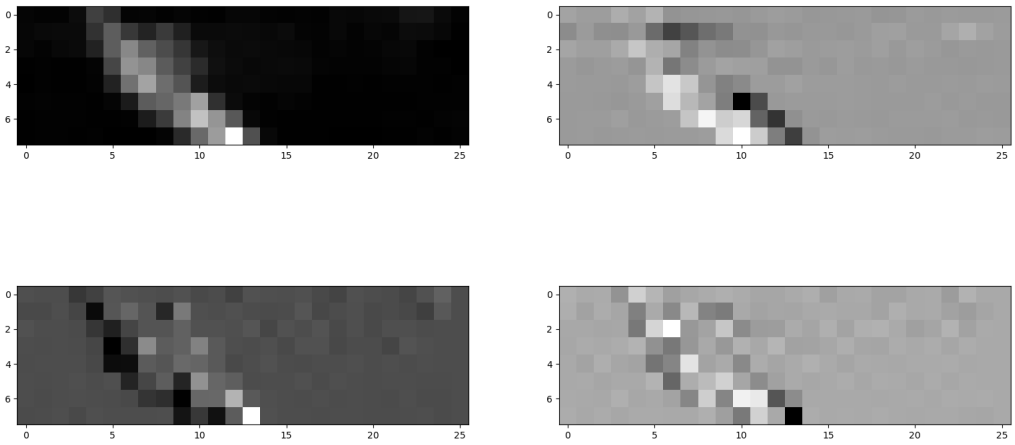
这里直接使用了python的小波分析库进行计算：

```
import matplotlib.pyplot as plt
import pywt

data = np.loadtxt("output.txt")
print(data.shape)
layer_one = np.flip(data[0].reshape(16, 51), axis=0)

coeffs = pywt.dwt2(layer_one, "haar")
cA, (cH, cV, cD) = coeffs
color = "gray"
```

```
plt.figure(figsize=(26, 10))
plt.subplot(221)
plt.imshow(cA, color)
plt.subplot(222)
plt.imshow(cH, color)
plt.subplot(223)
plt.imshow(cV, color)
plt.subplot(224)
plt.imshow(cD, color)
plt.show()
```

结果分析

结果如图所示



第一张图是小波的低频部分，可见出其保留了原图的大部分特征。其它三张是小波的高频部分，展现了原图在不同方向上的变化。

压缩感知

实现

1. 首先第一步是采样，这里从3380张图片中采出500张图片：

```
# 采样序列
sample_matrix = np.random.choice(range(3380), sample_num)
sample_matrix.sort()
# 采样
sample_data = data[sample_matrix].T
```

2. 然后对样本数据进行svd变换，分出代表空间，能量，时间的三个矩阵，并选取合适的模态数量进行重构：

```
U, S, Vt = np.linalg.svd(sample_data)
# 重构
U_new = U[:, :model_num]
S_new = np.zeros((model_num, model_num))

for i in range(model_num):
    S_new[i][i] = S[i]
Vt_new = Vt[:model_num, :]
```

3. 接着对时间维度的矩阵进行压缩感知：

```
# 压缩感知
Psi = np.linalg.inv(scipy.fft.dct(np.eye(n, n)))
Theta = Psi[sample_matrix]
Vt_re = np.zeros((model_num, n))
for i in range(model_num):
    x = cp.Variable(n)
    obj = cp.Minimize(cp.norm(x, 1))
    cons = [Theta @ x == Vt_new[i]]
    pro = cp.Problem(obj, cons)
    pro.solve()
    Vt_re[i] = scipy.ifft(x.value)
```

    - 通过对恒等变换矩阵进行FFT变换后求逆，我们可以获得FFT变换的基向量矩阵，即Φ，对Φ使用同样的采样序列进行采样，即可得到Θ矩阵。
    - 通过凸优化算法，即可解出y。
    - 对y做逆FFT变换，即可复原出对应的时间维度。

4. 最后将重构的时间矩阵代回，得到复原后的数据。

```
Data_re = np.dot(np.dot(U_new, S_new), Vt_re).T
np.savetxt("re.txt", Data_re)
```

更具体的实现可看CompressedSensing.py文件

结果展示

通过dataRe.py文件对前900张图片进行了复原，复原的图片保存在"picture/DataRe2"中，为了使结果 更加的便于展示，这里将图片合成了视频，保存在了"video"文件夹中。可以直观的看出，效果是很不错的。