import numpy as np  
from tqdm import tqdm  
from scipy.stats import multivariate\_normal  
from MCM\_B\_model1\_main import simulate\_motion  
  
  
# 从CSV文件读取数据  
result\_front = np.loadtxt('result.csv', delimiter=',', skiprows=1)  
  
true\_position = [10000, 40000, -600]  
# 只取前 1000 个点  
result = result\_front[:10000, :]  
  
# 模型参数  
num\_iterations = 30 # 迭代次数  
search\_radius = 30 # 搜索半径  
stop\_probability\_threshold = 0.0001 # 停止概率阈值  
  
# 先验概率的参数  
std\_prior = 8  
std = 1  
window\_size = 100 # 窗口大小，每一百个点计算一次局部均值  
# 初始化一个与 result 大小相同的数组 local\_means，用于存储每一百个点的局部均值  
local\_means = np.zeros\_like(result)  
# 使用循环遍历每一百个点  
# 取result中最后十个点  
last\_thou\_points = result[-1000:, :]  
  
# 计算最后十个点的均值  
mean\_x = np.mean(last\_thou\_points[:, 0])  
mean\_y = np.mean(last\_thou\_points[:, 1])  
mean\_z = np.mean(last\_thou\_points[:, 2])  
  
  
  
def prior(x, y, z):  
 mu\_prior = np.array([mean\_x, mean\_y, mean\_z])  
 sigma\_prior = np.eye(3) \* std\_prior  
 prior\_distribution = multivariate\_normal(mu\_prior, sigma\_prior)  
 positions = np.column\_stack([x.flatten(), y.flatten(), z.flatten()])  
 prior\_values = prior\_distribution.logpdf(positions)  
 return prior\_values.reshape(x.shape)  
  
  
def likelihood(x, y, z, observed\_position, std):  
 mu = observed\_position[:3]  
 cov = np.eye(3) \* std\*\*2  
 likelihood\_distribution = multivariate\_normal(mu, cov)  
 positions = np.column\_stack([x.flatten(), y.flatten(), z.flatten()])  
 likelihood\_values = likelihood\_distribution.logpdf(positions)  
 return likelihood\_values.reshape(x.shape)  
  
  
def bayesian\_update(prior, likelihood, observed\_positions, std, x\_range, y\_range, z\_range):  
 posterior = np.zeros((len(x\_range), len(y\_range), len(z\_range)))  
  
 x\_grid, y\_grid, z\_grid = np.meshgrid(x\_range, y\_range, z\_range, indexing='ij')  
  
 positions\_grid = np.stack([x\_grid, y\_grid, z\_grid], axis=-1)  
  
 prior\_values = prior(positions\_grid[:, :, :, 0], positions\_grid[:, :, :, 1], positions\_grid[:, :, :, 2])  
  
 for observed\_position in tqdm(observed\_positions):  
 likelihood\_values = likelihood(  
 positions\_grid[:, :, :, 0], positions\_grid[:, :, :, 1], positions\_grid[:, :, :, 2],  
 observed\_position, std  
 )  
  
 posterior += np.exp(prior\_values) \* np.exp(likelihood\_values)  
  
 return posterior  
  
  
def Posibility(initial\_trajectory):  
 cumulative\_probabilities = []  
 positions\_and\_distributions = []  
 iteration = 0  
  
 while True:  
 # 获取当前位置附近的可能后验概率分布  
 current\_position = initial\_trajectory[-1][:3] # 取路径中的最后一个点作为当前位置  
 x\_range = np.linspace(current\_position[0] - search\_radius, current\_position[0] + search\_radius, num\_iterations)  
 y\_range = np.linspace(current\_position[1] - search\_radius, current\_position[1] + search\_radius, num\_iterations)  
 z\_range = np.linspace(current\_position[2] - search\_radius, current\_position[2] + search\_radius, num\_iterations)  
  
 # 更新mean，采用过去十个点的均值  
 last\_thou\_points = np.array([point[:3] for point in initial\_trajectory[-1000:]])  
 mean\_x = np.mean(last\_thou\_points[:, 0])  
 mean\_y = np.mean(last\_thou\_points[:, 1])  
 mean\_z = np.mean(last\_thou\_points[:, 2])  
  
 posterior = bayesian\_update(prior, likelihood, initial\_trajectory, std, x\_range, y\_range, z\_range)  
  
 # 找到后验概率分布中的中心点  
 max\_index = np.unravel\_index(np.argmax(posterior), posterior.shape)  
 new\_center\_relative = np.array([x\_range[max\_index[0]], y\_range[max\_index[1]], z\_range[max\_index[2]]])  
  
 # Ensure the new center is within 10 meters relative to the current center  
 movement = np.random.uniform(low=-5, high=5, size=3)  
 movement /= np.linalg.norm(movement) # 将随机移动向量归一化为单位向量  
 movement \*= 30 # 缩放为十米的移动距离  
 next\_position = current\_position + movement  
  
 probability = np.prod(posterior[max\_index[0], max\_index[1], max\_index[2]])  
  
probability\_str.replace('.', '')  
 probability = float(probability\_str\_with\_decimal) \* 5  
  
 if iteration == 0:  
 cumulative\_probabilities.append(probability)  
 else:  
 cumulative\_probabilities.append(cumulative\_probabilities[-1] \* probability \* 0.8)  
  
 print(f"Iteration {iteration + 1}: Next position: {next\_position}, Probability: {probability}")  
  
 if cumulative\_probabilities[-1] < stop\_probability\_threshold:  
 print("Stopping search: Probability below threshold.")  
 break  
 else:  
 initial\_trajectory.append(np.concatenate([next\_position, np.zeros(3)]))  
 iteration += 1  
  
  
 positions\_and\_distributions.append({  
 'iteration': iteration,  
 'position': next\_position,  
 'probability': probability  
 })  
  
 return cumulative\_probabilities, positions\_and\_distributions  
  
  
  
cumulative\_probabilities, positions\_and\_distributions = Posibility(result.tolist())  
print(positions\_and\_distributions)  
  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
# 绘制贝叶斯预测的新散点  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
  
for pos in positions\_and\_distributions:  
 ax.scatter(pos['position'][0], pos['position'][1], pos['position'][2], c='blue', label='Bayesian Prediction', alpha=0.2)  
  
ax.set\_xlabel('X')  
ax.set\_ylabel('Y')  
ax.set\_zlabel('Z')  
plt.show()