Final report

組員:詹竣智　蔡輝益

Download code

Download code and put it into your workspace

https://drive.google.com/drive/folders/137Jed499OjqH3O0pIZMja7GlqETIi0Fz?usp=drive_link

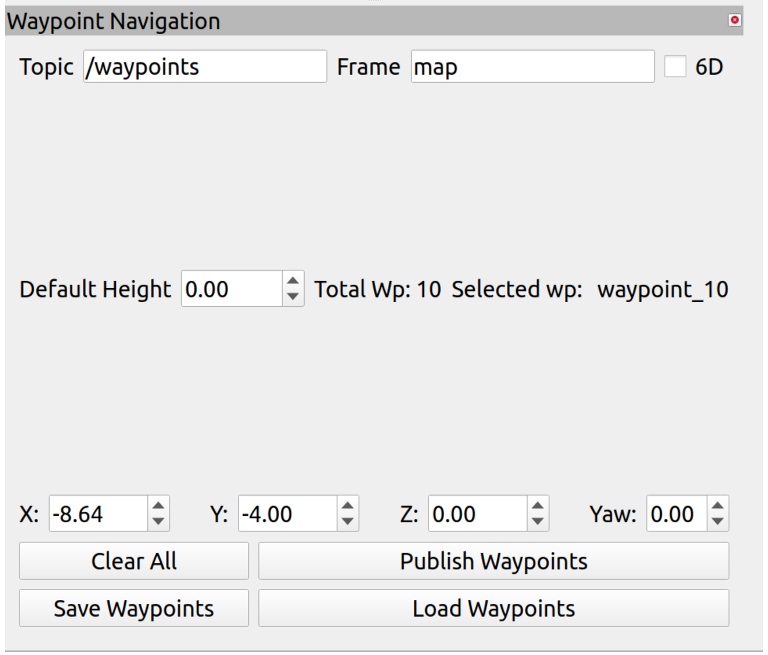
執行流程

1. 開啟軌跡可視化 Rviz 套件
2. 開啟 final.py 將publish的10個座標點位置儲存為txt file
3. 開啟 traj_gen.py 讀取上方的10個座標點位置並以最小二成法擬合出軌跡方程式。最後產生在此軌跡上的181個座標點並存於另一 txt file
4. 開啟 trajectory.py 讀取上一步驟產生的位置資料並將其發布於leader的 topic 且計算其 yaw angle
5. 開啟 formation.py 使 follower 和 leader 進行 formation

Rviz 可視化

1. 開啟軌跡可視化套件後輸入10個點並將其發布於 /waypoints topic

2. 執行 final.py 將發布的點 儲存於 waypoints.txt

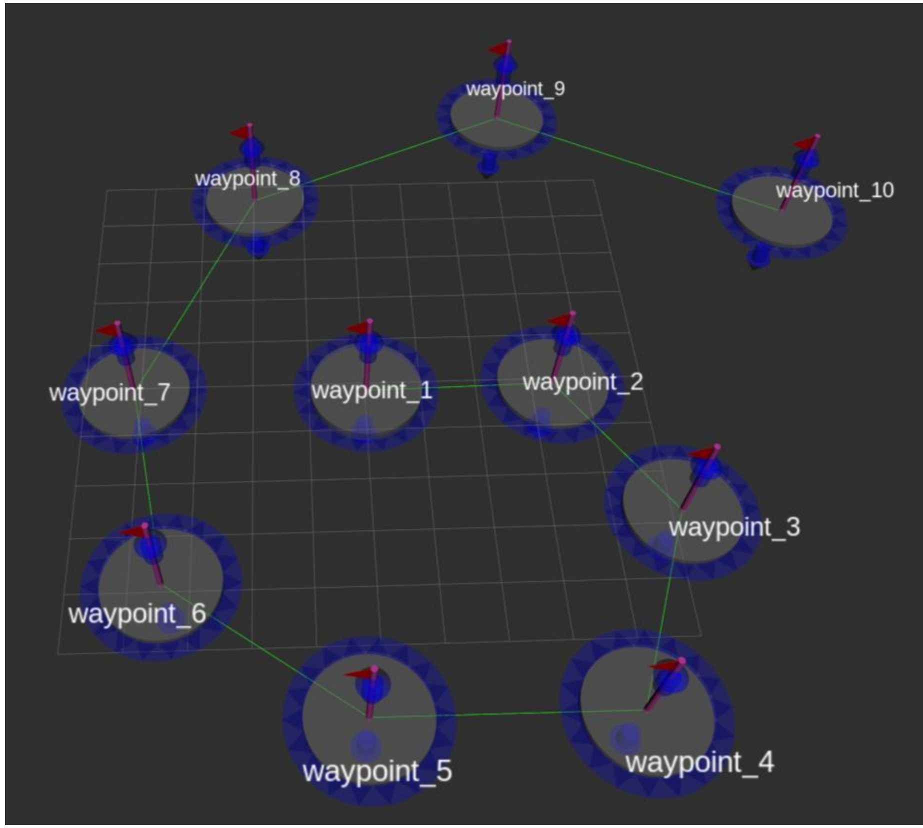


Waypoint Navigation

Topic /waypoints Frame map □ 6D

Default Height 0.00 Total Wp: 10 Selected wp: waypoint_10

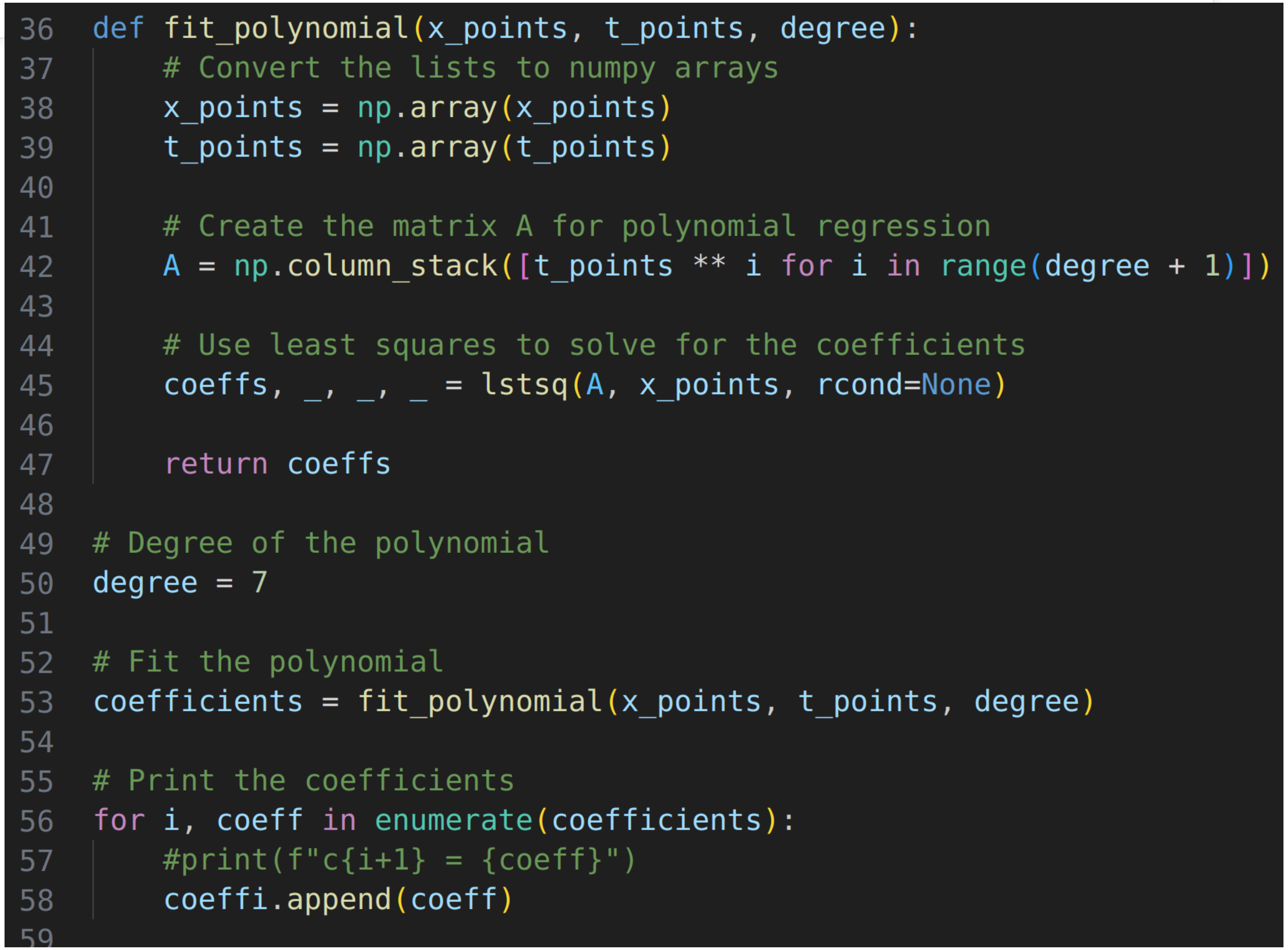
X: -8.64 Y: -4.00 Z: 0.00 Yaw: 0.00



生成軌跡

1. 讀取 waypoints.txt 之位置資料

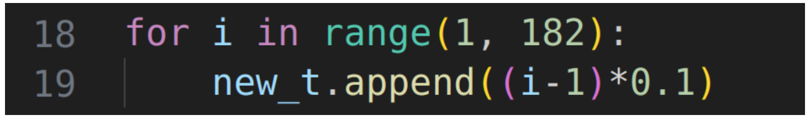
2. 以7階多項式進行擬合並使用最小二成方法求解係數



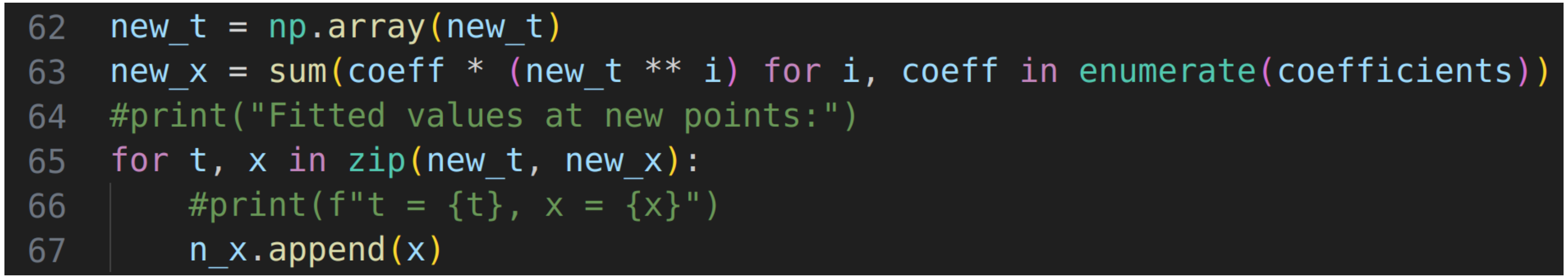
```python
def fit_polynomial(x_points, t_points, degree):
    # Convert the lists to numpy arrays
    x_points = np.array(x_points)
    t_points = np.array(t_points)

    # Create the matrix A for polynomial regression
    A = np.column_stack([t_points ** i for i in range(degree + 1)])

    # Use least squares to solve for the coefficients
    coeffs, _, _, _ = lstsq(A, x_points, rcond=None)

    return coeffs

# Degree of the polynomial
degree = 7

# Fit the polynomial
coefficients = fit_polynomial(x_points, t_points, degree)

# Print the coefficients
for i, coeff in enumerate(coefficients):
    #print(f"c{i+1} = {coeff}")
    coeffi.append(coeff)
```

生成更多在軌跡上的點

產生更多在此軌跡上的位置點並將其存於 trajectory.txt



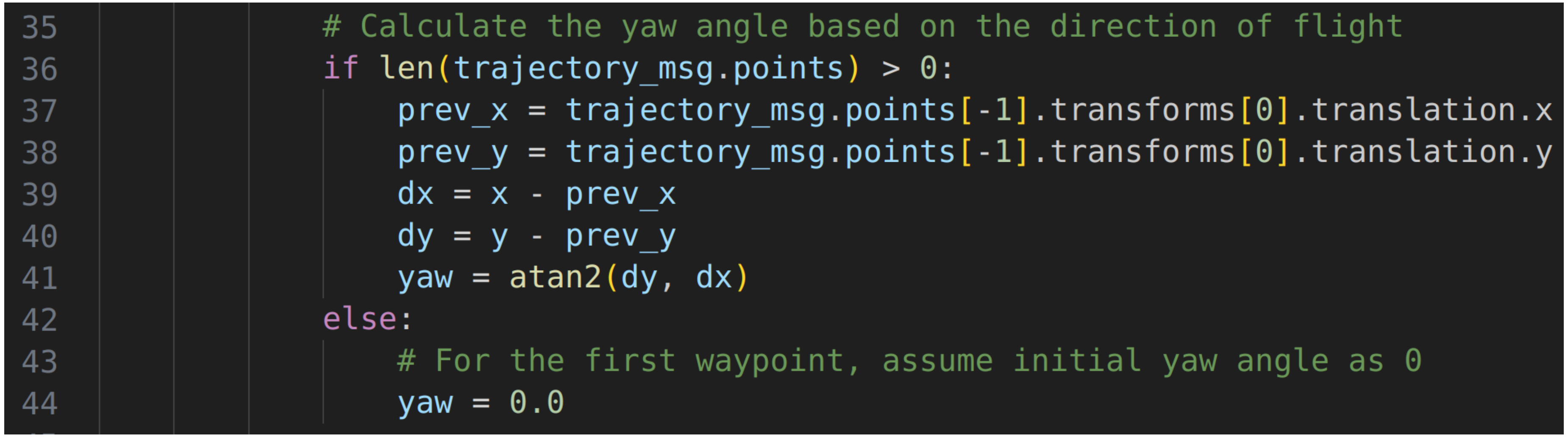
```python
for i in range(1, 182):
    new_t.append((i-1)*0.1)
```



```python
new_t = np.array(new_t)
new_x = sum(coeff * (new_t ** i) for i, coeff in enumerate(coefficients))
#print("Fitted values at new points:")
for t, x in zip(new_t, new_x):
    #print(f"t = {t}, x = {x}")
    n_x.append(x)
```

發布軌跡給 leader

1. 執行 trajectory.py
2. 計算 yaw angle 並發布位置和 yaw 資訊給 leader



```python
            # Calculate the yaw angle based on the direction of flight
            if len(trajectory_msg.points) > 0:
                prev_x = trajectory_msg.points[-1].transforms[0].translation.x
                prev_y = trajectory_msg.points[-1].transforms[0].translation.y
                dx = x - prev_x
                dy = y - prev_y
                yaw = atan2(dy, dx)
            else:
                # For the first waypoint, assume initial yaw angle as 0
                yaw = 0.0
```

Formation

1. formation.py 已加入 final.launch 因此執行final.launch 即自動執行 formation
2. formation.py 會參考 leader 的位置及 yaw 並將計算後的位置及 yaw angle 發布給 follower