```python
import numpy as np
from numpy.linalg import inv


def wrapper(ang):
    ang -= np.pi * 2 * np.floor((ang + np.pi) / (2 * np.pi))
    return ang


class EIF:
    def __init__(self, Quad, World):
        self.ts = Quad.ts
        self.mu = np.array([[Quad.x0],
                            [Quad.y0],
                            [Quad.theta0]])
        self.SIG = np.diag([1, 1, 1])
        self.landmarks = World.Landmarks
        self.num_landmarks = World.Number_Landmarks

        # given with each function call
        self.Omega = inv(self.SIG)
        self.xi = self.Omega @ self.mu
        v = Quad.v_t
        theta = 0
        # also r and ph, both vectors containing measurements

        # needed for lines 2-7
        self.G = self.calc_G(theta, v)
        self.V = self.calc_V(theta)
        self.M = np.diag([Quad.sig_v ** 2, Quad.sig_omega ** 2])
        self.Q = np.array([[Quad.sigma_r ** 2, 0],
                           [0, Quad.sigma_theta ** 2]])
        self.Q_inv = inv(self.Q)

    def calc_G(self, theta, v):
        G = np.array([[1, 0, - v * np.sin(theta) * self.ts],
                      [0, 1, v * np.cos(theta) * self.ts],
                      [0, 0, 1]])
        return G

    def calc_V(self, theta):
        V = np.array([[np.cos(theta) * self.ts, 0],
                      [np.sin(theta) * self.ts, 0],
                      [0., self.ts]])
        return V

    def calc_g(self, v, theta, omega):
        mat = np.array([[v * np.cos(theta) * self.ts],
                        [v * np.sin(theta) * self.ts],
                        [omega * self.ts]])
        mu_bar = self.mu + mat
        return mu_bar

    def update(self, xi, Omega, v, omega, r, ph):
        self.xi = xi
        self.Omega = Omega
        self.SIG = inv(self.Omega)
        self.mu = self.SIG @ self.xi
        theta = self.mu[2][0]
        self.G = self.calc_G(theta, v)
        self.V = self.calc_V(theta)
        # lines 2-5
```

```
        self.SIG_bar = self.G @ self.SIG @ self.G.T + self.V @ self.M @ self.V.T
        self.Omega_bar = inv(self.SIG_bar)
        self.mu_bar = self.calc_g(v, theta, omega)
        self.xi_bar = self.Omega_bar @ self.mu_bar

        self.features(r, ph)

        self.Omega = self.Omega_bar
        self.xi = self.xi_bar

    def features(self, r, ph):
        for spot in range(0, self.num_landmarks):
            diff_x = self.landmarks[0][spot] - self.mu_bar[0][0]
            diff_y = self.landmarks[1][spot] - self.mu_bar[1][0]
            q = diff_x ** 2 + diff_y ** 2
            z_hat = np.array([[np.sqrt(q)],
                              [wrapper(np.arctan2(diff_y, diff_x) - self.mu_bar[2]
[0])]])
            z_diff = np.array([r[spot] - z_hat[0],
                               wrapper(ph[spot] - z_hat[1])])
            H = np.array([[-diff_x / np.sqrt(q), -diff_y / np.sqrt(q), 0],
                          [diff_y / q, -diff_x / q, -1]])
            S = H @ self.SIG_bar @ H.T + self.Q
            K = self.SIG_bar @ H.T @ np.linalg.inv(S)
            self.mu_bar = self.mu_bar + K @ z_diff
            self.SIG_bar = (np.eye(3) - K @ H) @ self.SIG_bar
            self.Omega_bar = self.Omega_bar + H.T @ self.Q_inv @ H
            self.xi_bar = self.xi_bar + H.T @ self.Q_inv @ (z_diff + H @ self.mu_bar)
```