```
function [aircraft_state_est, wind_inertial_est] = SimpleEstimator(time,
gps_sensor, inertial_sensors, sensor_params)
%
% This estimator assumes it runs at the rate of the IMU sensors, and that
% the GPS sensor runs at a slower rate. Thus, the GPS sensor is only
% updated at its rate.
%
% gps_sensor = [pn; pe; ph; Vg; chi]
%
% inertial_sensors = [y_accel; y_gyro; y_pressure; y_dyn_pressure];
%
%
%
%

persistent phat
persistent qhat
persistent rhat

persistent press_stat
persistent press_dyn

persistent pn_hat
persistent pe_hat

persistent s_x
persistent s_y
persistent s_z
persistent chi_hat


h_ground = sensor_params.h_ground;
density = stdatmo(h_ground);

Ts_imu = sensor_params.Ts_imu;
Ts_gps = sensor_params.Ts_gps;
g = sensor_params.g;

%%%%%%%%%%%%%%%%%%%%%%
%%% angular velocity
%%%%%%%%%%%%%%%%%%%%%%
a_omega = 1000;
alpha_omega = exp(-a_omega*Ts_imu);


if(isempty(phat))
    phat = inertial_sensors(4);
else
    phat = LowPassFilter(phat, inertial_sensors(4), alpha_omega);
end

if(isempty(qhat))
```

```
    qhat = inertial_sensors(5);
else
    qhat = LowPassFilter(qhat, inertial_sensors(5), alpha_omega);
end

if(isempty(rhat))
    rhat = inertial_sensors(6);
else
    rhat = LowPassFilter(rhat, inertial_sensors(6), alpha_omega);
end

%%%%%%%%%%%%%%%
%%% height
%%%%%%%%%%%%%%%

a_h = 20; % <======================STUDENT SELECT
alpha_h = exp(-a_h*Ts_imu);

if(isempty(press_stat))
    press_stat = inertial_sensors(7);
else
    press_stat = LowPassFilter(press_stat, inertial_sensors(7), alpha_h);
end
hhat = press_stat / (density * g) + h_ground;%
<======================STUDENT COMPLETE

%%%%%%%%%%%%%%%
%%% airspeed
%%%%%%%%%%%%%%%

a_Va =  1000; % <======================STUDENT SELECT
alpha_Va = exp(-a_Va*Ts_imu);

if(isempty(press_dyn))
    press_dyn = inertial_sensors(8);
else
    press_dyn = LowPassFilter(press_dyn, inertial_sensors(8), alpha_Va);
end
Va = sqrt(2/density * press_dyn);% <======================STUDENT COMPLETE


%%%%%%%%%%%%%%%%%%%
%%% position (gps)
%%%%%%%%%%%%%%%%%%%
a_gps =  10; % <======================STUDENT SELECT
alpha_gps = exp(-a_gps*Ts_gps);


if(isempty(pn_hat))
    pn_hat = gps_sensor(1);
else
    if(mod(time, Ts_gps)==0) % <=============== Only update at GPS rate
        pn_hat = LowPassFilter(pn_hat, gps_sensor(1), alpha_gps);
    end
```

```
end

if(isempty(pe_hat))
    pe_hat = gps_sensor(2);
else
    if(mod(time, Ts_gps)==0) % <============== Only update at GPS rate
        pe_hat = LowPassFilter(pe_hat, gps_sensor(2), alpha_gps);
    end
end

if(isempty(chi_hat))
    chi_hat = gps_sensor(5);
else
    if(mod(time, Ts_gps)==0) % <============== Only update at GPS rate
        chi_hat = LowPassFilter(chi_hat, gps_sensor(5), alpha_gps);
    end
end

%%%%%%%%%%%%%%%%%%%%
%%% orientation
%%%%%%%%%%%%%%%%%%%%
a_acc =  1000; % <======================STUDENT SELECT
alpha_acc = exp(-a_acc*Ts_imu);

if(isempty(s_x))
    s_x = inertial_sensors(1);
else
    s_x = LowPassFilter(s_x, inertial_sensors(1), alpha_acc);
end
if(isempty(s_y))
    s_y = inertial_sensors(2);
else
    s_y = LowPassFilter(s_y, inertial_sensors(2), alpha_acc);
end
if(isempty(s_z))
    s_z = inertial_sensors(3);
else
    s_z = LowPassFilter(s_z, inertial_sensors(3), alpha_acc);
end

roll_hat =  atan(s_y / s_z);% <======================STUDENT COMPLETE
pitch_hat =  asin(s_x / g);% <======================STUDENT COMPLETE
yaw_hat = chi_hat;%%%%%

%%%%%%%%%%%%%%%
%%% output
%%%%%%%%%%%%%%%

% Calculate the inertial velocity vector in body coordinates
alpha_hat = pitch_hat; % Assume angle of attack is equal to the pitch
Vrel_body = Va .* [cos(alpha_hat); 0; sin(alpha_hat)];

aircraft_state_est =  [pn_hat; pe_hat; -hhat;
                       roll_hat; pitch_hat; yaw_hat;
```

```
                        Vrel_body;
                        phat; qhat; rhat];% <======================STUDENT
COMPLETE

wind_inertial_est = [0; 0; 0];

end %function


%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% Low Pass Filter
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

function ynew = LowPassFilter(yold, unew, alpha)
    ynew = alpha*yold + (1-alpha)*unew;
end
```

Published with MATLAB® R2023b