ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA VẬT LÝ

----------🕮----------

BÁO CÁO TIỂU LUẬN

ĐỀ TÀI

Tìm hiểu về SENSOR FUSION

Hà Nội – 2023

Phạm Duy Ninh

Mã sinh viện: 21002222

Ngành: K66 Kỹ thuật Điện tử và Tin học

(Chương trình đào tạo chuẩn)

Cán bộ hướng dẫn:

TS. Phạm Văn Thành

ThS.Nguyễn Tiến Đạt

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

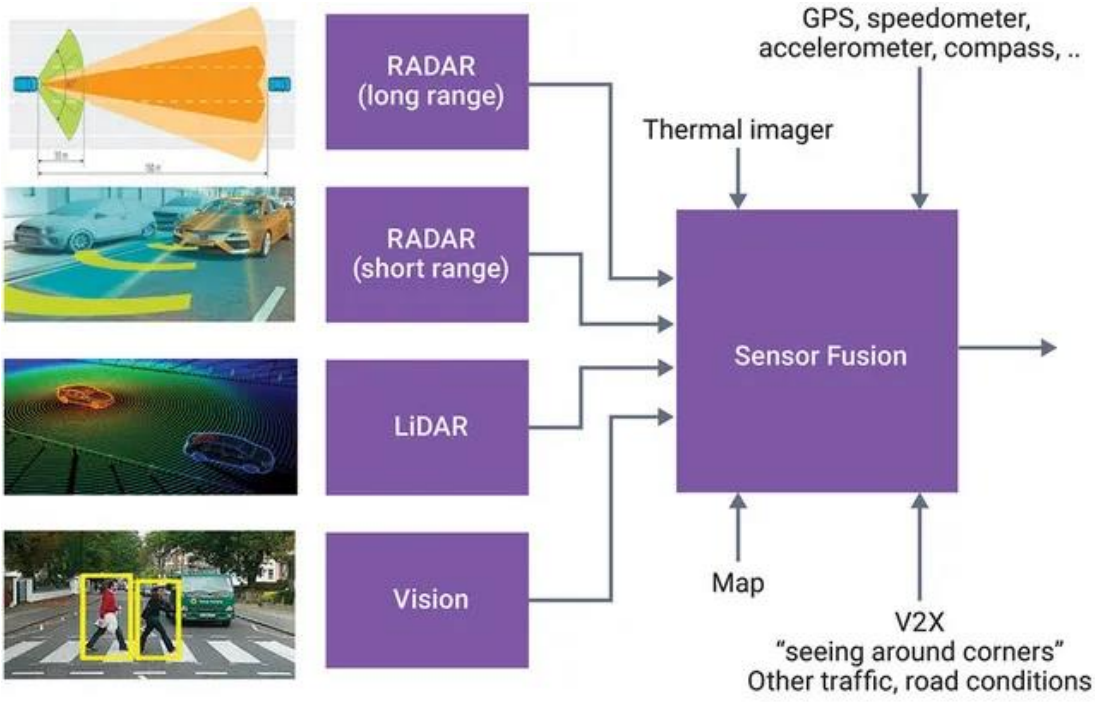
KHOA: VẬT LÝ

Mục lục

I. Giới thiệu

1. Giới thiệu chung về Sensor Fusion

Sensor Fusion, hay Sự kết hợp cảm biến, là một lĩnh vực nghiên cứu đang phát triển nhanh chóng và có ảnh hưởng lớn đến nhiều ngành công nghiệp khác nhau. Qua việc tích hợp dữ liệu từ nhiều cảm biến khác nhau, Sensor Fusion cung cấp một hiểu biết chính xác và đáng tin cậy hơn về môi trường so với việc sử dụng từng cảm biến riêng lẻ. Điều này không chỉ cải thiện độ chính xác và độ tin cậy của hệ thống mà còn mở ra khả năng ứng dụng trong các lĩnh vực như xe tự lái, robot di động, và hệ thống giám sát thông minh.



Hình 1. Minh họa cho Sensor Fusion

Lịch sử phát triển của Sensor Fusion bắt đầu từ những năm 1960 và 1970, khi các nhà nghiên cứu bắt đầu khám phá việc sử dụng các thuật toán như Kalman filter để kết hợp dữ liệu từ radar và sonar trong các ứng dụng quân sự. Từ đó, công nghệ đã phát triển mạnh mẽ, với các bước tiến quan trọng như sự ra đời của các cảm biến

MEMS giá rẻ vào những năm 1990, cho phép tích hợp Sensor Fusion vào các thiết bị tiêu dùng.

Sensor Fusion không chỉ là một công cụ kỹ thuật; nó còn là một yếu tố quan trọng trong việc đưa ra quyết định thông minh và tăng cường khả năng nhận thức của các hệ thống tự động. Với sự phát triển của công nghệ AI và IoT, vai trò của Sensor Fusion càng trở nên nổi bật, đặc biệt là trong việc đảm bảo an toàn và hiệu quả của các hệ thống phức tạp.

2. Giới thiệu Mô phỏng Matlab với Sensor Fusion and Tracking Toolbox

Matlab là một công cụ mạnh mẽ cho việc mô phỏng và phân tích dữ liệu, trong đó có quá trình Sensor Fusion. *Sensor Fusion and Tracking Toolbox* của MATLAB bao gồm các thuật toán và công cụ để thiết kế, mô phỏng, và kiểm tra các hệ thống kết hợp dữ liệu từ nhiều cảm biến để duy trì nhận thức tình huống và định vị.

Các tính năng chính của Sensor Fusion and Tracking Toolbox bao gồm:

- *Mô phỏng cảm biến và tình huống*: Định nghĩa các tình huống đa nền tảng, sau đó gán các hồ sơ chuyển động và gắn các mô hình cảm biến vào mỗi nền tảng. Mô phỏng các tình huống này và trực quan hóa các quỹ đạo nền tảng, phạm vi cảm biến, và phát hiện đối tượng.
- *Bộ lọc ước lượng*: Sử dụng các bộ lọc ước lượng khác nhau, như bộ lọc Kalman, bộ lọc nhiều mô hình, và bộ lọc hạt, để ước lượng trạng thái đối tượng2. Các bộ lọc này đã được tối ưu hóa cho các tình huống cụ thể, chẳng hạn như mô hình chuyển động tuyến tính hoặc phi tuyến, hoặc quan sát không đầy đủ2.
- *Theo dõi đa đối tượng*: Sử dụng các bộ theo dõi đa đối tượng đa cảm biến tích hợp bộ lọc, phân công dữ liệu, và quản lý theo dõi. Chọn từ một loạt các bộ theo dõi bao gồm bộ theo dõi mục tiêu đơn, nhiều giả thuyết, phân công dữ liệu xác suất chung, tập hợp hữu hạn ngẫu nhiên, hoặc theo dõi dựa trên lưới.
- *Kết hợp đa cảm biến*: Khám phá các kiến trúc theo dõi đa đối tượng trung tâm hoặc phân tán và đánh giá các quyết định thiết kế giữa việc kết hợp theo dõi, theo dõi cấp trung tâm, hoặc kiến trúc theo dõi lai cho các ứng dụng theo dõi khác nhau.

- *Đánh giá và điều chỉnh*: Phân tích và đánh giá hiệu suất của các hệ thống theo dõi so với sự thật chính xác bằng cách sử dụng các chỉ số theo dõi khác nhau.

Với Sensor Fusion and Tracking Toolbox, bạn có thể kết hợp dữ liệu từ các cảm biến thực tế, bao gồm radar chủ động và thụ động, sonar, lidar, EO/IR, IMU, và GPS. Bạn cũng có thể tạo dữ liệu tổng hợp từ các cảm biến ảo để kiểm tra các thuật toán của bạn trong các tình huống khác nhau.

II. Tìm hiểu một số Thí nghiệm mô phỏng quá trình Sensor Fusion

1. Ước lượng Hướng dựa trên Sự Kết hợp Cảm biến Quán tính và MPU-9250

MPU-9250 là một thiết bị theo dõi chuyển động 9 trục, được tích hợp trong một gói QFN nhỏ gọn với kích thước chỉ 3x3x1mm. Nó bao gồm một con quay hồi chuyển 3 trục, một gia tốc kế 3 trục và một la bàn từ trường 3 trục.

Quá trình ước lượng hướng sử dụng cảm biến quán tính và MPU-9250 dựa trên việc kết hợp dữ liệu từ ba loại cảm biến: gia tốc kế (accelerometer), con quay hồi chuyển (gyroscope), và la bàn từ trường (magnetometer). Gia tốc kế đo gia tốc, con quay hồi chuyển đo vận tốc góc, và la bàn từ trường đo trường từ trong các trục x, y và z.

Để xây dựng mô phỏng cho quá trình này, bạn cần kết nối cảm biến MPU-9250 với phần cứng như Arduino Uno và sử dụng các thuật toán kết hợp 6 trục và 9 trục để tính toán hướng của thiết bị. Bạn cũng cần phải điều chỉnh các sai số từ cảm biến từ trường bằng cách sử dụng các giá trị hiệu chỉnh có thể xác định thông qua việc quay cảm biến từ 0 đến 360 độ dọc theo mỗi trục.

Sản phẩm MathWorks® bắt buộc

- MATLAB®
- Gói hỗ trợ MATLAB cho phần cứng Arduino®
- Navigation Toolbox™ hoặc Sensor Fusion and Tracking Toolbox™

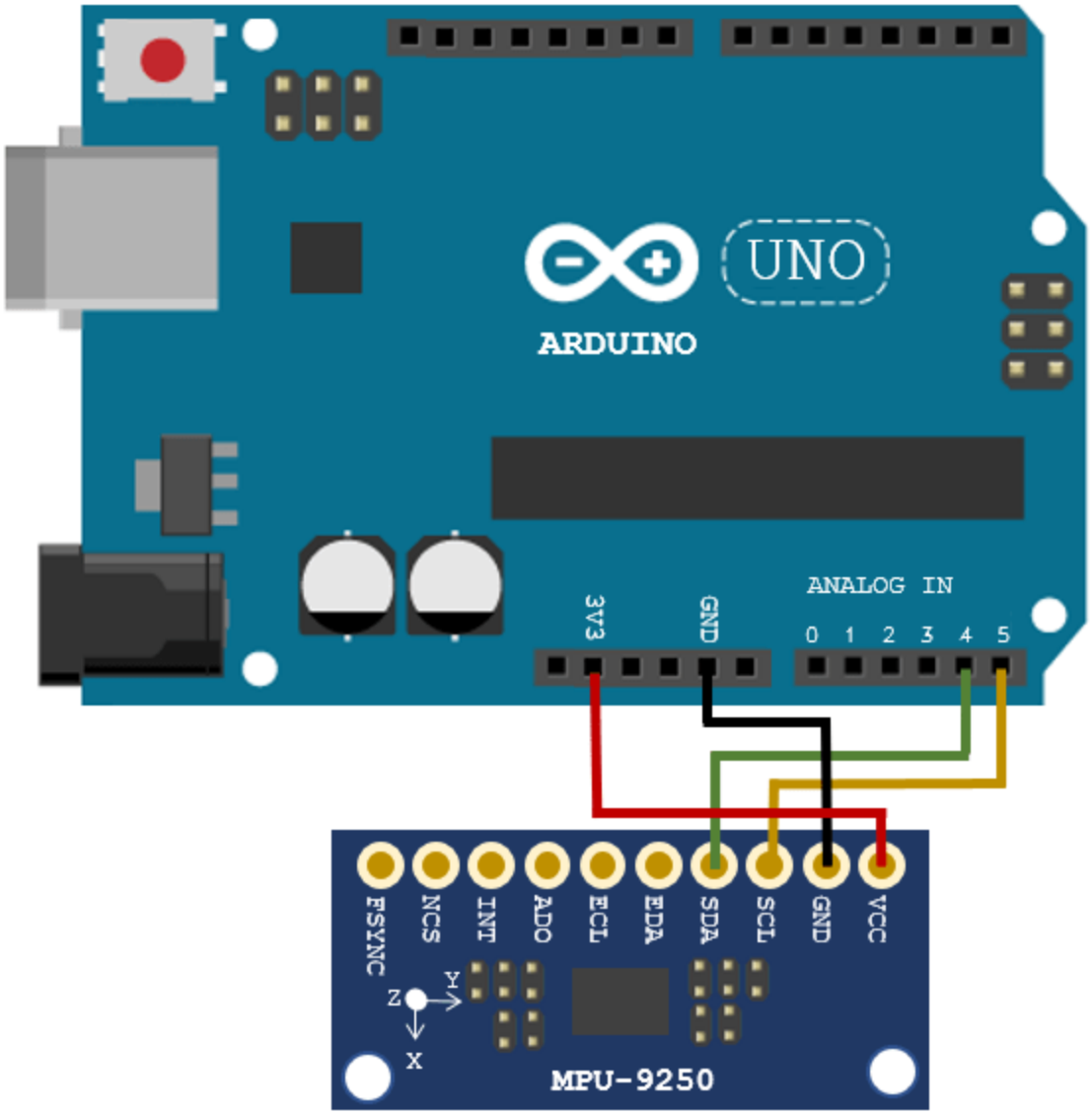
Phần cứng cần thiết

- Arduino Uno
- InvenSense MPU-9250

Kết nối phần cứng

Kết nối các chân SDA, SCL, GND và VCC của cảm biến MPU-9250 với các chân tương ứng trên Phần cứng Arduino như sau:

- SDA - A4
- SCL - A5
- VCC - +3,3V
- GND - GND



Hình 2. Minh họa kết nối phần cứng giữa Arduino và MPU-9250

Tạo đối tượng cảm biến

Tạo một đối tượng và bao gồm thư viện I2C.arduino:

```
a = arduino('COM9', 'Uno', 'Libraries', 'I2C');
```

Tạo đối tượng cảm biến MPU-9250:

```
fs = 100; % Sample Rate in Hz
imu = mpu9250(a,'SampleRate',fs,'OutputFormat','matrix');
```

Bù méo từ kế

Các thuật toán Fusion sử dụng dữ liệu Từ kế cần được bù cho các biến dạng từ do từ trường bị biến dạng từ cảm biến bảng mạch và môi trường xung quanh, các phép đo từ tính có thể bị xáo trộn theo hai loại: Biến dạng sắt cứng và biến dạng sắt mềm. Những biến dạng này có thể được sửa chữa bằng cách sử dụng các giá trị hiệu chỉnh có thể được xác định bằng các bước sau:

1. Xoay cảm biến từ 0 đến 360 độ dọc theo mỗi trục.
2. Sử dụng hàm magcal (Navigation Toolbox), như dưới đây, để có được các hệ số hiệu chỉnh:

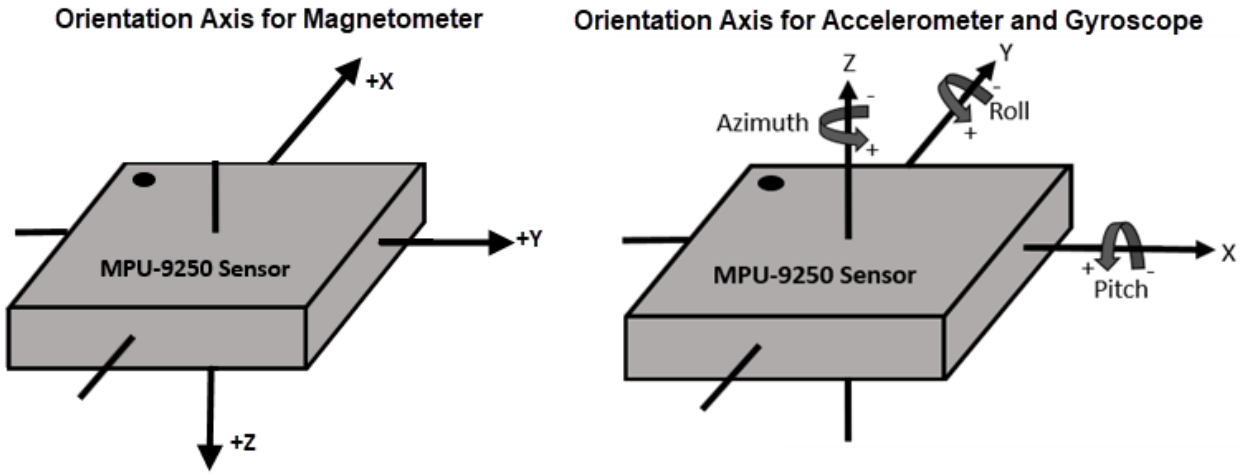
```
ts = tic;
stopTimer = 50;
magReadings=[];
while(toc(ts) < stopTimer)
    % Rotate the sensor along x axis from 0 to 360 degree.
    % Take 2-3 rotations to improve accuracy.
    % For other axes, rotate along that axes.
   [accel,gyro,mag] = read(imu);
   magReadings = [magReadings;mag];
end

[A, b] = magcal(magReadings); % A = 3x3 matrix for soft iron correction
                              % b = 3x1 vector for hard iron correction
```

Căn chỉnh trục của cảm biến MPU-9250 với tọa độ NED

Các thuật toán hợp nhất cảm biến được sử dụng trong ví dụ này sử dụng North-East-Down (NED) như một hệ tọa độ gốc cố định. Trong hệ quy chiếu NED, trục X hướng về phía bắc, trục Y hướng về phía đông và trục Z hướng xuống. Tùy thuộc vào thuật toán, phía bắc có thể là phía bắc từ tính hoặc phía bắc thực sự. Các thuật toán trong ví dụ này sử dụng hướng bắc từ tính. Các thuật toán được sử dụng ở đây mong đợi tất cả các cảm biến trong đối tượng có trục của chúng được căn chỉnh và phù hợp với quy ước NED.

MPU-9250 có hai thiết bị, từ kế và con quay hồi chuyển-gia tốc, trên cùng một bảng. Trục của các thiết bị này khác nhau. Trục từ kế được căn chỉnh với tọa độ NED. Trục của con quay hồi chuyển gia tốc khác với từ kế trong MPU-9250. Gia tốc kế và trục con quay hồi chuyển cần được hoán đổi và / hoặc đảo ngược để phù hợp với trục từ kế.



Hình 3. Định hướng trục trong biểu dữ liệu MPU-9250

Các thuật toán kết hợp sử dụng số đọc từ kế cần được bù cho các biến dạng từ

Để căn chỉnh trục gia tốc kế-con quay hồi chuyển MPU-9250 với tọa độ NED, hãy làm như sau:

1. Xác định trục thiết bị: Xác định trục tưởng tượng là trục thiết bị trên cảm biến theo hệ tọa độ NED có thể giống hoặc không giống với trục cảm biến. Đối với MPU-9250, trục từ kế có thể được coi là trục thiết bị.

2. Hoán đổi giá trị x và y của số đọc gia tốc kế và con quay hồi chuyển, sao cho trục gia tốc kế và con quay hồi chuyển thẳng hàng với trục từ kế.

3. Xác định giá trị cực tính cho gia tốc kế và con quay hồi chuyển.

- Đặt cảm biến sao cho trục X của thiết bị hướng xuống dưới, vuông góc với bề mặt giữ cảm biến. Chỉ số gia tốc kế nên đọc khoảng [9.8 0 0]. Nếu không phủ nhận các giá trị x của gia tốc kế.
- Đặt cảm biến sao cho trục Y của thiết bị hướng xuống dưới, vuông góc với bề mặt lưu giữ cảm biến. Chỉ số gia tốc kế nên đọc khoảng [0 9.8 0]. Nếu không phủ nhận các giá trị y của gia tốc kế.
- Đặt cảm biến sao cho trục Z của thiết bị hướng xuống dưới, vuông góc với bề mặt giữ cảm biến. Chỉ số gia tốc kế nên đọc khoảng [0 0 9.8]. Nếu không phủ nhận giá trị z của gia tốc kế.

Con quay hồi chuyển:

Xoay cảm biến dọc theo mỗi trục và ghi lại các bài đọc. Sử dụng quy tắc bàn tay phải để điều chỉnh cực tính của vòng quay.

Phương pháp trên được sử dụng để đặt trục của cảm biến trong ví dụ này.

Điều chỉnh thông số bộ lọc

Các thuật toán được sử dụng trong ví dụ này, khi được điều chỉnh đúng, cho phép ước tính định hướng và mạnh mẽ chống lại các nguồn nhiễu môi trường. Bạn phải xem xét các tình huống trong đó các cảm biến được sử dụng và điều chỉnh các bộ lọc cho phù hợp.

Gia tốc kế-Con quay hồi chuyển-Từ kế Fusion

Hệ thống tham chiếu AHRS bao gồm hệ thống 9 trục sử dụng gia tốc kế, con quay hồi chuyển và từ kế để tính toán hướng của thiết bị. Việc tạo ra một ước tính thay đổi trơn tru về hướng của thiết bị, đồng thời ước tính chính xác hướng bắc. Nó có khả năng loại bỏ sai lệch con quay hồi chuyển và cũng có thể phát hiện và loại bỏ nhiễu từ nhẹ.

Các đoạn mã sau đây sử dụng đối tượng hệ thống để xác định hướng của cảm biến và tạo ra một hình được cập nhật khi bạn di chuyển cảm biến. Cảm biến phải đứng yên, trước khi bắt đầu ví dụ này:

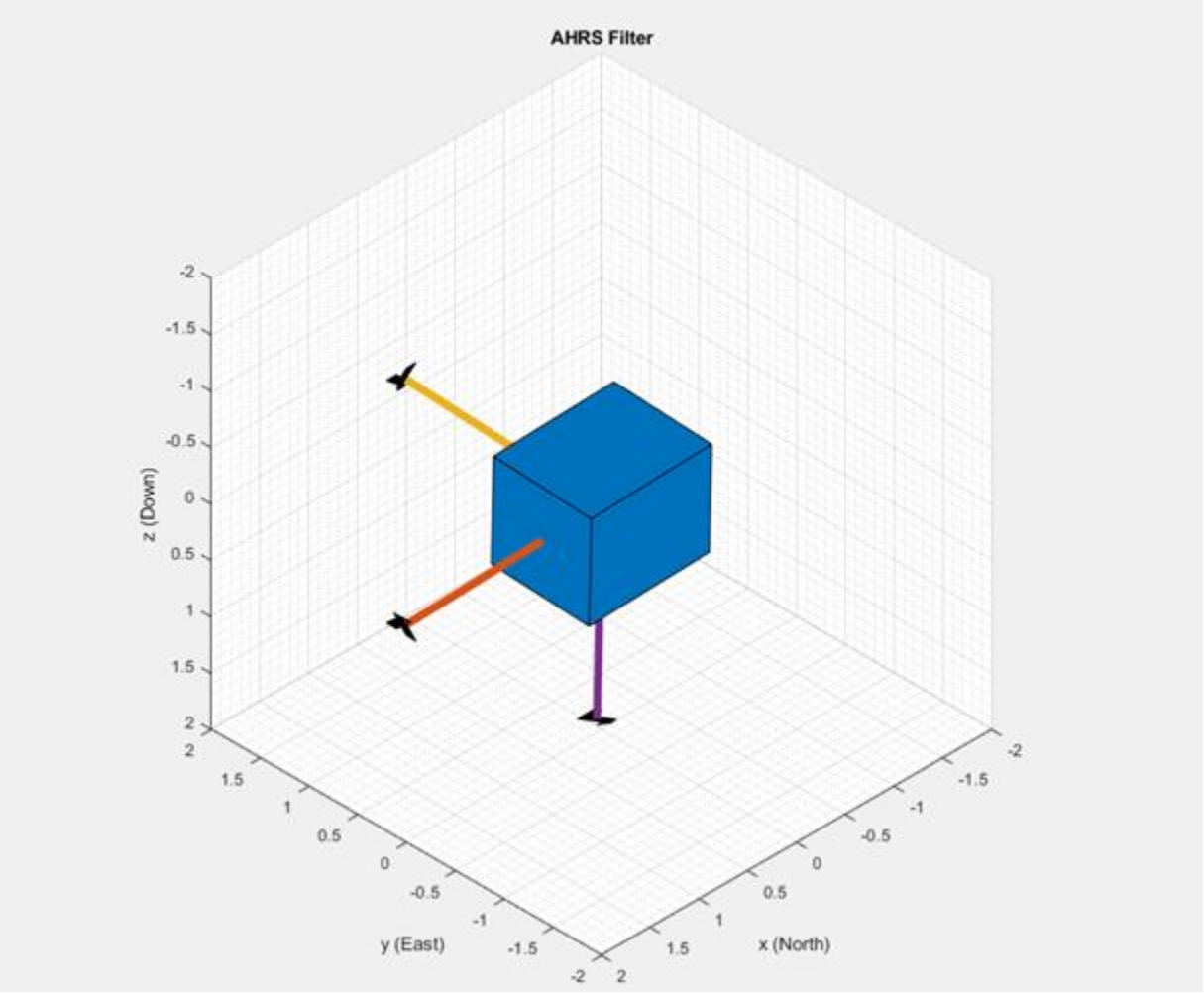
```
% GyroscopeNoise and AccelerometerNoise is determined from datasheet.
GyroscopeNoiseMPU9250 = 3.0462e-06; % GyroscopeNoise (variance value) in units of
rad/s
AccelerometerNoiseMPU9250 = 0.0061; % AccelerometerNoise(variance value)in units of
m/s^2
viewer = HelperOrientationViewer('Title',{'AHRS Filter'});
FUSE = ahrsfilter('SampleRate',imu.SampleRate,
'GyroscopeNoise',GyroscopeNoiseMPU9250,'AccelerometerNoise',AccelerometerNoiseMPU9250
);
stopTimer = 100;
```

Trong khi mã dưới đây đang được thực thi, hãy từ từ di chuyển cảm biến và kiểm tra xem chuyển động trong hình có khớp với chuyển động của cảm biến hay không:

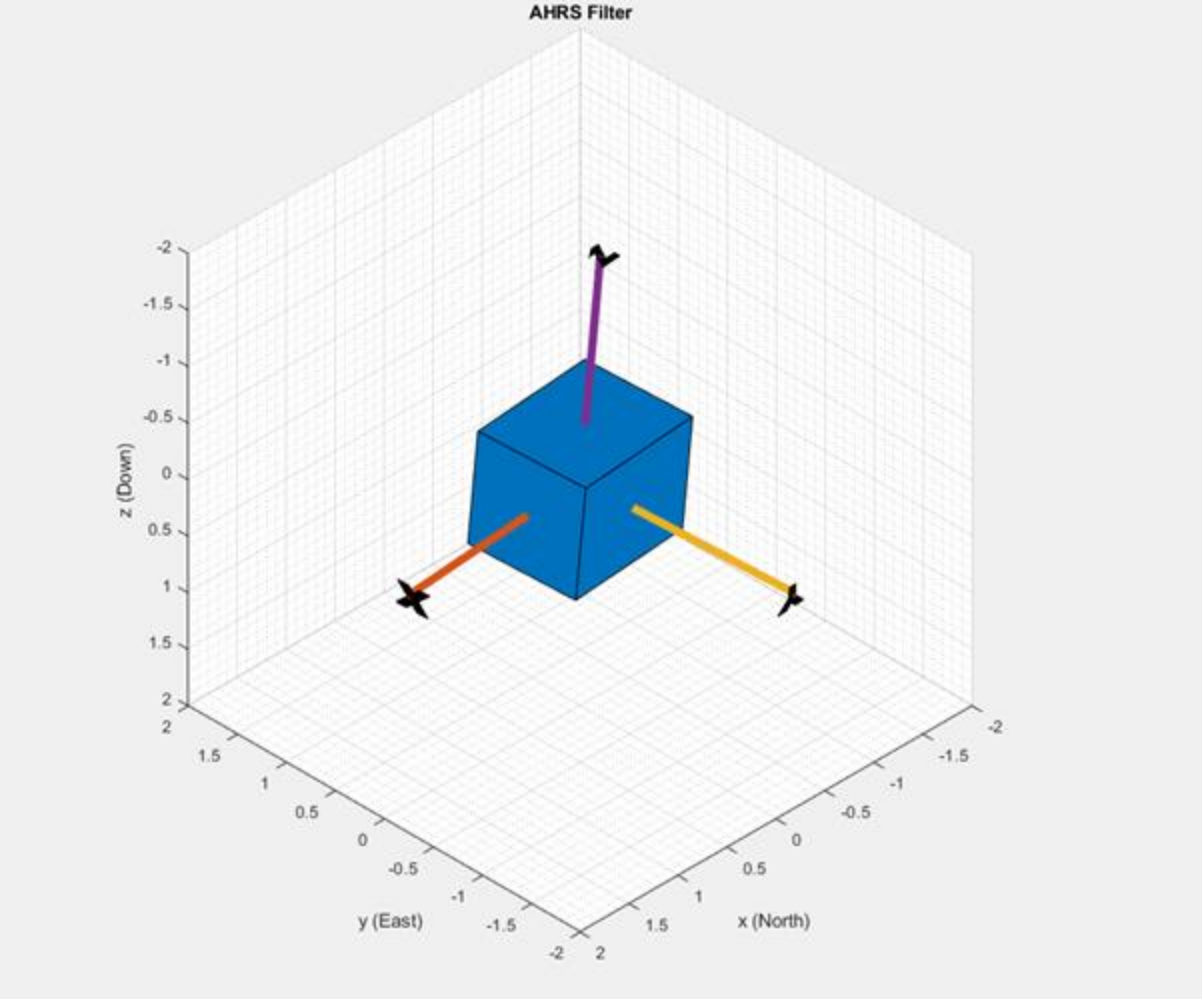
```
% The correction factors A and b are obtained using magcal function as explained in
one of the
% previous sections, 'Compensating Magnetometer Distortions'.
magx_correction = b(1);
magy_correction = b(2);
magz_correction = b(3);
ts = tic;
```

```
while(toc(ts) < stopTimer)
    [accel,gyro,mag] = read(imu);
    % Align coordinates in accordance with NED convention
    accel = [-accel(:,2), -accel(:,1), accel(:,3)];
    gyro = [gyro(:,2), gyro(:,1), -gyro(:,3)];
    mag = [mag(:,1)-magx_correction, mag(:, 2)- magy_correction, mag(:,3)-
magz_correction] * A;
    rotators = FUSE(accel,gyro,mag);
    for j = numel(rotators)
        viewer(rotators(j));
    end
end
```

Kết quả Mô phỏng:



Hình 4. Khi trục cảm biến X của thiết bị hướng về phía bắc, trục Y của thiết bị đang hướng về phía đông và trục Z của thiết bị đang hướng xuống



Hình 5. Khi trục cảm biến X của thiết bị hướng về phía bắc, trục Y của thiết bị đang hướng về phía tây và trục Z của thiết bị hướng lên trên

Gia tốc kế-Con quay hồi chuyển Fusion

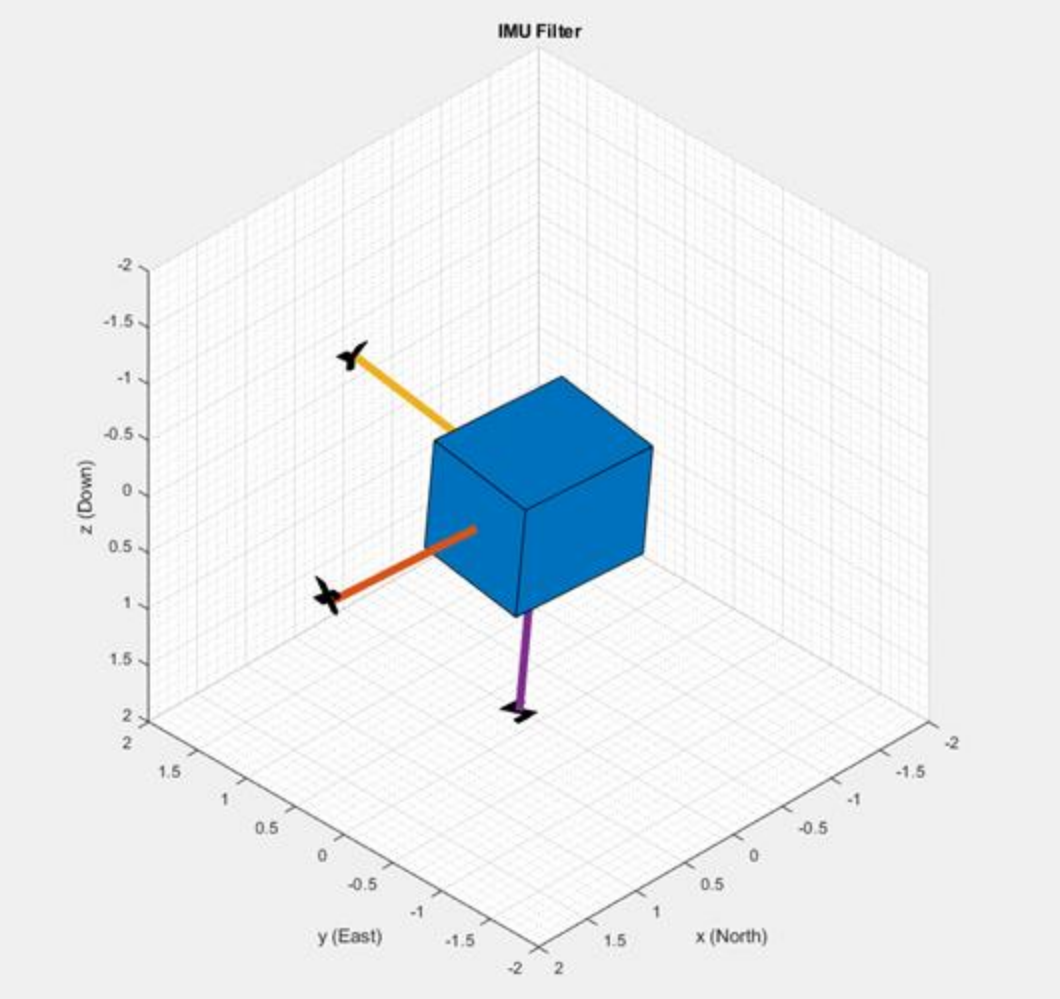
Đối tượng hệ thống hợp nhất dữ liệu gia tốc kế và con quay hồi chuyển sử dụng bộ lọc Kalman. Bộ lọc có khả năng loại bỏ nhiễu thiên vị con quay hồi chuyển trôi theo thời gian. Bộ lọc không xử lý dữ liệu từ kế, vì vậy nó không ước tính chính xác hướng bắc. Thuật toán giả định vị trí ban đầu của cảm biến theo cách mà trục X của thiết bị của cảm biến hướng về phía bắc từ tính, trục Y của thiết bị của cảm biến hướng về phía đông và trục Z của thiết bị của cảm biến hướng xuống dưới. Cảm biến phải đứng yên, trước khi bắt đầu ví dụ này.

Các đoạn mã sau đây sử dụng đối tượng để xác định hướng của cảm biến và tạo ra một hình được cập nhật khi bạn di chuyển cảm biến:

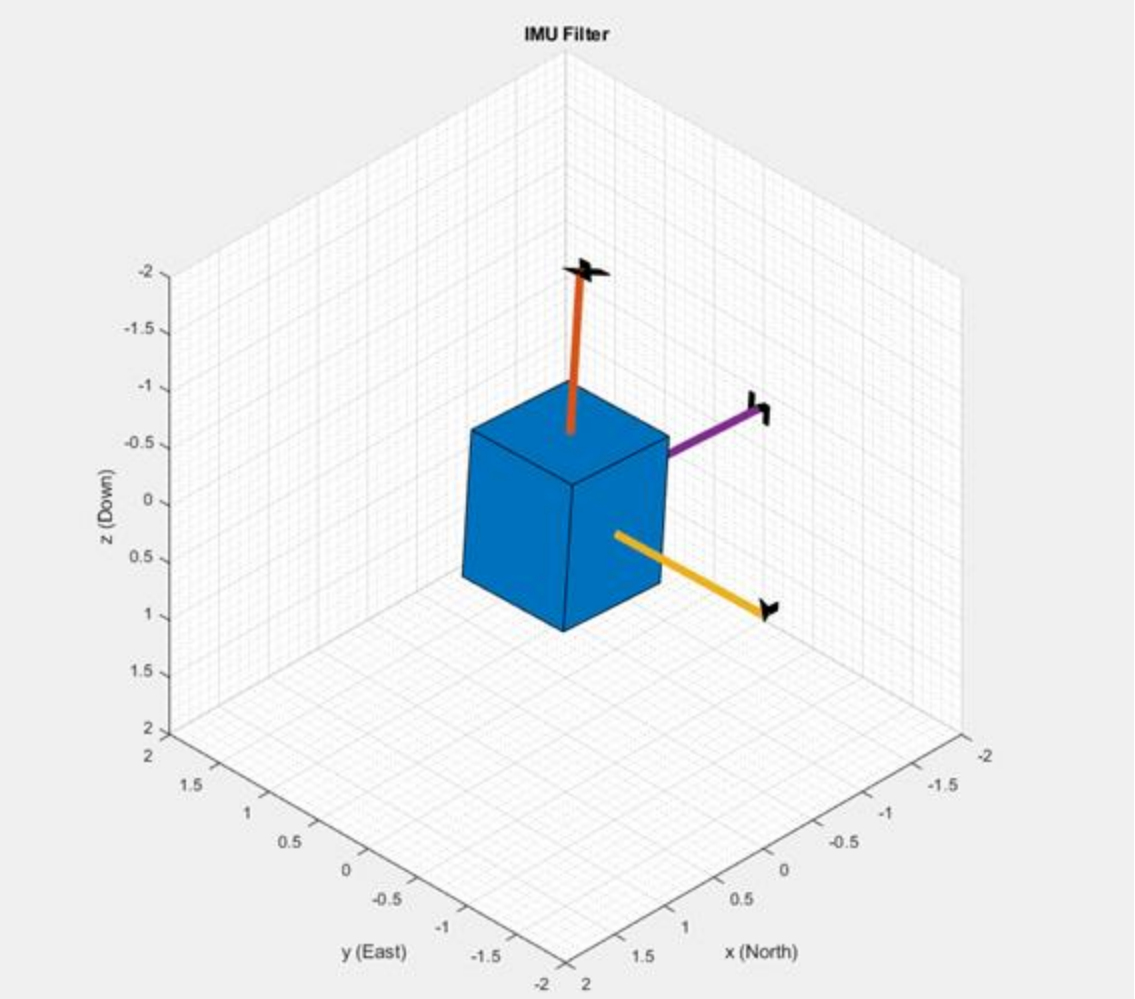
```
displayMessage(['This section uses  IMU filter to determine orientation of the sensor
by collecting live sensor data from the \slmpu9250 \rm' ...
    'system object. Move the sensor to visualize orientation of the sensor in the
figure window. Keep the sensor stationery before you'...
    'click OK'],...
    'Estimate Orientation using IMU filter and MPU-9250.')
% GyroscopeNoise and AccelerometerNoise is determined from datasheet.
GyroscopeNoiseMPU9250 = 3.0462e-06; % GyroscopeNoise (variance) in units of rad/s
AccelerometerNoiseMPU9250 = 0.0061; % AccelerometerNoise (variance) in units of m/s^2
viewer = HelperOrientationViewer('Title',{'IMU Filter'});
FUSE = imufilter('SampleRate',imu.SampleRate,
'GyroscopeNoise',GyroscopeNoiseMPU9250,'AccelerometerNoise',
AccelerometerNoiseMPU9250);
stopTimer=100;
```

Trong khi mã dưới đây đang được thực thi, hãy từ từ di chuyển cảm biến và kiểm tra xem chuyển động trong hình có khớp với chuyển động của cảm biến hay không:

```
% Use imufilter to estimate orientation and update the viewer as the sensor moves for
time specified by stopTimer
tic;
while(toc < stopTimer)
    [accel,gyro] = read(imu);
    accel = [-accel(:,2), -accel(:,1), accel(:,3)];
    gyro = [gyro(:,2), gyro(:,1), -gyro(:,3)];
    rotators = FUSE(accel,gyro);
    for j = numel(rotators)
        viewer(rotators(j));
    end
end
```



Hình 6. Khi trục cảm biến X của thiết bị hướng về phía bắc, trục Z của thiết bị hướng xuống dưới và trục Y của thiết bị hướng về phía đông.



Hình 7. Khi trục cảm biến X của thiết bị hướng lên trên, trục Y của thiết bị hướng về phía tây và trục Z của thiết bị hướng về phía nam.

Khi kết nối không còn cần thiết, hãy giải phóng và xóa các đối tượng:

```
release(imu);
clear;
```

2. Ước lượng Tư thế từ các Cảm biến Không Đồng bộ

Ví dụ này cho thấy cách bạn có thể hợp nhất các cảm biến ở các tốc độ khác nhau để ước tính tư thế. Gia tốc kế, con quay hồi chuyển, từ kế và GPS được sử dụng để xác định hướng và vị trí của một chiếc xe di chuyển dọc theo một đường tròn. Bạn có thể sử dụng các điều khiển trên cửa sổ hình để thay đổi tốc độ cảm biến và thử nghiệm với độ rơi của cảm biến trong khi xem hiệu ứng ước tính tư thế.

Bạn có thể copy lệnh sau đây vào cửa sổ Command trên Matlab để chạy ví dụ mô phỏng này(Yêu cầu Matlab phải có Sensor Fusion and Tracking Toolbox):

openExample('shared_positioning/PoseEstimationFromAsynchronousSensorsExample')

Thiết lập mô phỏng

Tải dữ liệu cảm biến được ghi sẵn. Dữ liệu cảm biến dựa trên quỹ đạo tròn được tạo bằng lớp. Các giá trị cảm biến được tạo bằng cách sử dụng các lớp và tệp có thể được tạo bằng hàm.

Bộ lọc Fusion

Tạo một phép đo IMU + GPS để hợp nhất. Bộ lọc Fusion này sử dụng bộ lọc Kalman mở rộng rời rạc liên tục (EKF) để theo dõi hướng (như một quaternion), vận tốc góc, vị trí, vận tốc, gia tốc, độ lệch cảm biến và vectơ địa từ.

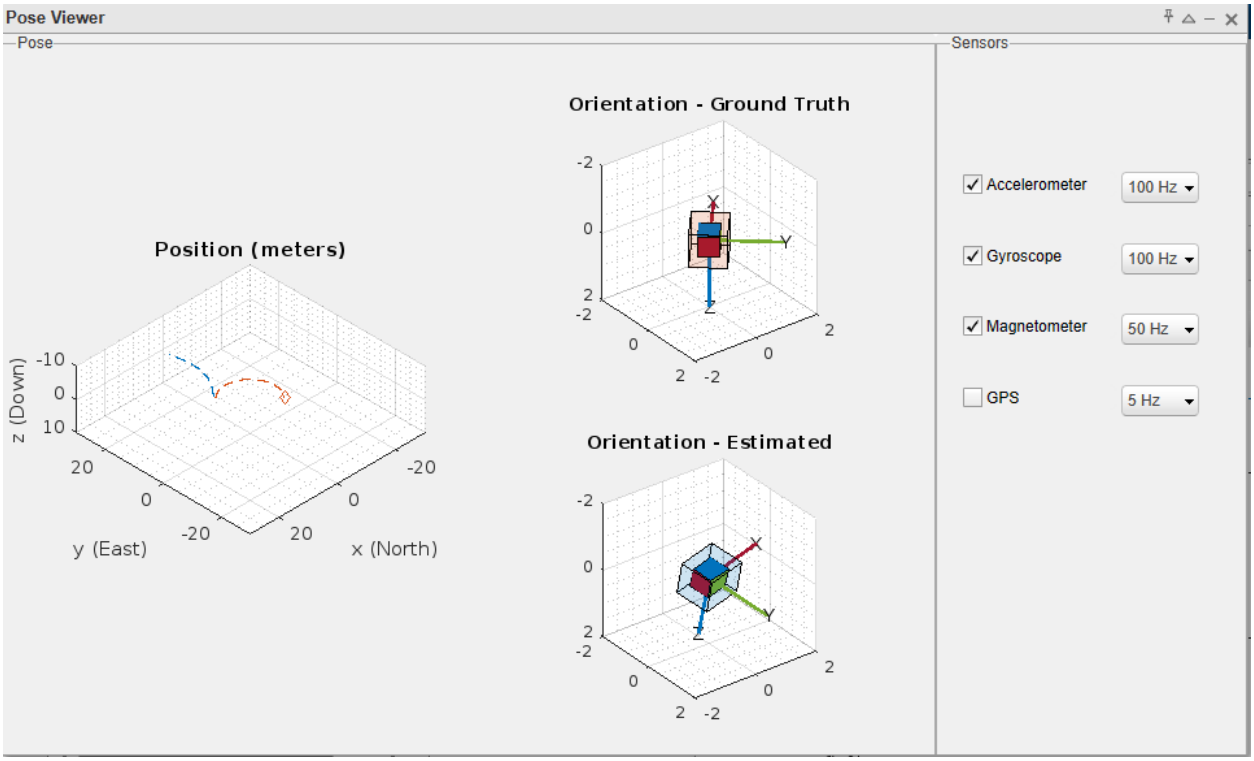
Điều này có một số phương pháp để xử lý dữ liệu cảm biến. Bởi vì sử dụng EKF rời rạc liên tục, phương pháp có thể đẩy bộ lọc về phía trước một khoảng thời gian tùy ý.

Vòng lặp mô phỏng

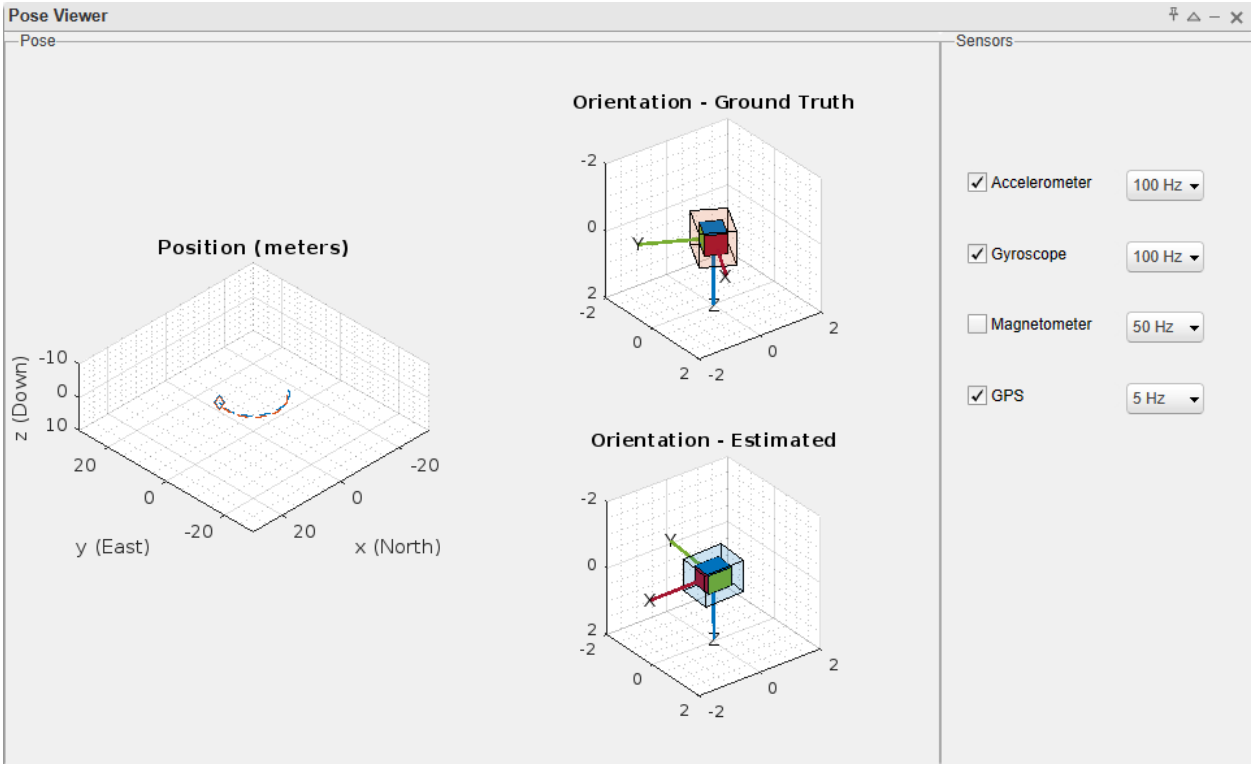
Mô phỏng thuật toán Fusion cho phép bạn kiểm tra ảnh hưởng của các tốc độ mẫu cảm biến khác nhau. Hơn nữa, sự hợp nhất của các cảm biến riêng lẻ có thể được ngăn chặn bằng cách bỏ chọn hộp kiểm tương ứng. Điều này có thể được sử dụng để mô phỏng sự sụt giảm cảm biến.

Hình 8. Hộp kiểm cảm biến trong cửa sổ mô phỏng Matlab

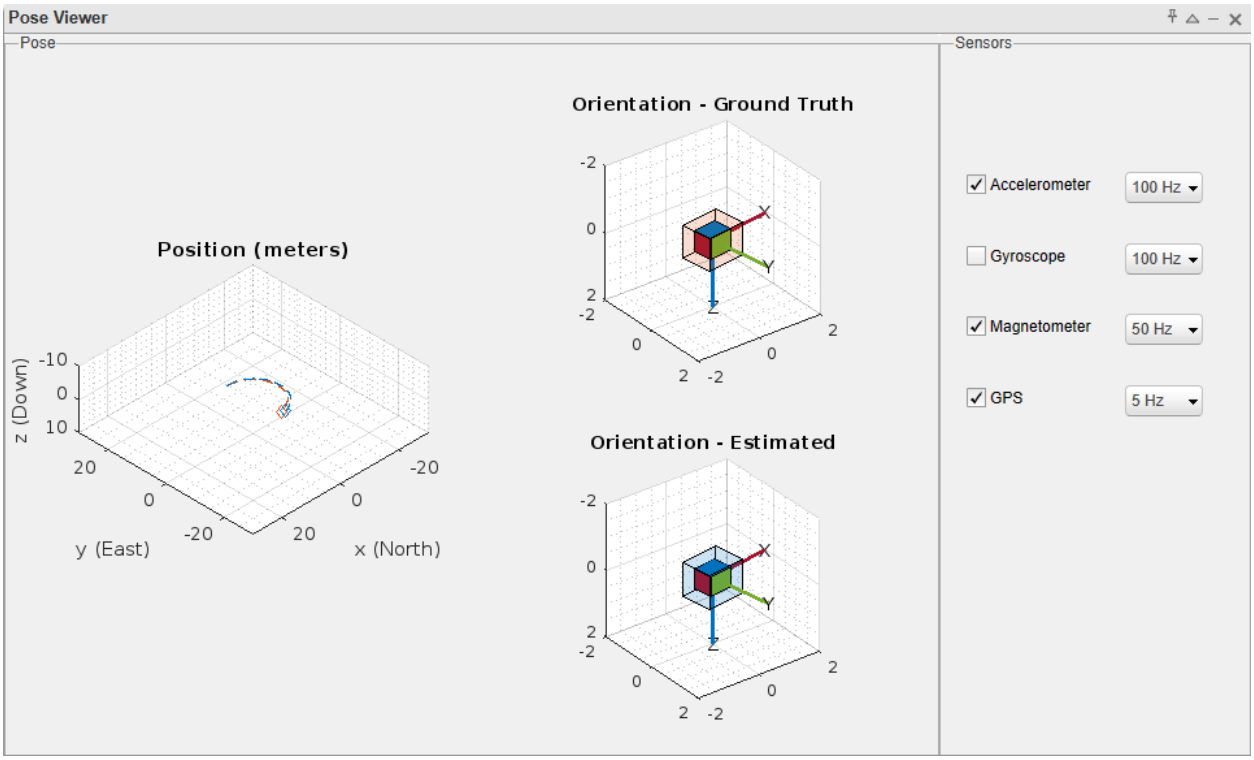
Một số cấu hình tạo ra kết quả ấn tượng. Ví dụ: tắt cảm biến GPS làm cho ước tính vị trí bị trôi nhanh. Tắt cảm biến từ kế sẽ làm cho ước tính hướng từ từ lệch khỏi mặt đất thật vì ước tính quay quá nhanh. Ngược lại, nếu con quay hồi chuyển bị tắt và từ kế được bật, hướng ước tính cho thấy sự chao đảo và thiếu độ mượt mà hiện tại nếu sử dụng cả hai cảm biến.



Hình 9. Tắt cảm biến GPS

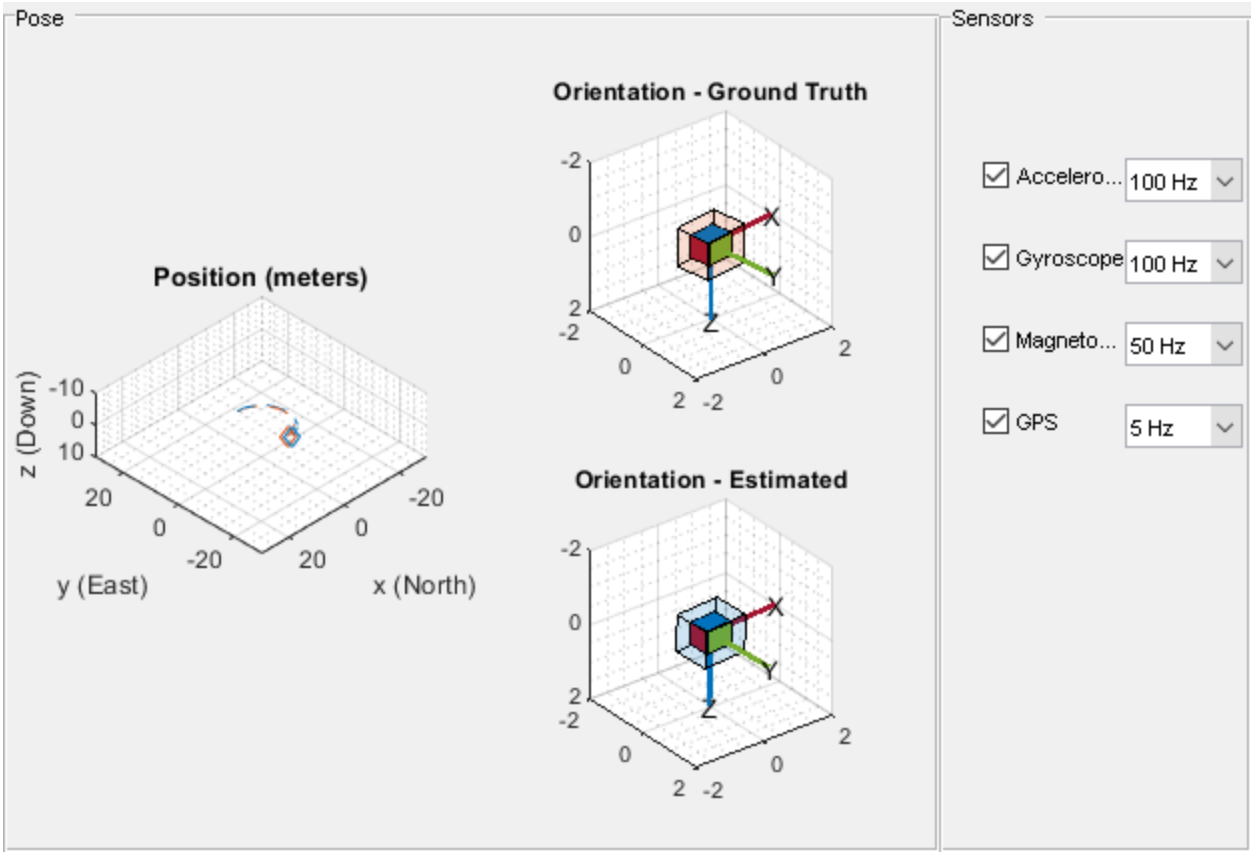


Hình 10. Tắt cảm biến Từ kế



Hình 11. Tắt cảm biến Con quay hồi chuyển

Bật tất cả các cảm biến nhưng đặt chúng chạy ở tốc độ thấp nhất sẽ tạo ra một ước tính rõ ràng sai lệch so với sự thật cơ bản và sau đó quay trở lại kết quả chính xác hơn khi các cảm biến được hợp nhất. Điều này dễ thấy nhất trong các lỗi ước tính đang chạy.



Hình 12. Bật tất cả các cảm biến

Mô phỏng chính chạy ở 100 Hz. Mỗi lần lặp lại kiểm tra các hộp kiểm trên cửa sổ hình, nếu cảm biến được bật, hợp nhất dữ liệu cho cảm biến đó ở tốc độ thích hợp.

Tóm tắt kết quả ví dụ

Các cho phép tỷ lệ mẫu khác nhau và khác nhau. Chất lượng của các đầu ra ước tính phụ thuộc rất nhiều vào tốc độ tổng hợp cảm biến riêng lẻ. Bất kỳ sự sụt giảm cảm biến nào cũng sẽ có ảnh hưởng sâu sắc đến đầu ra.

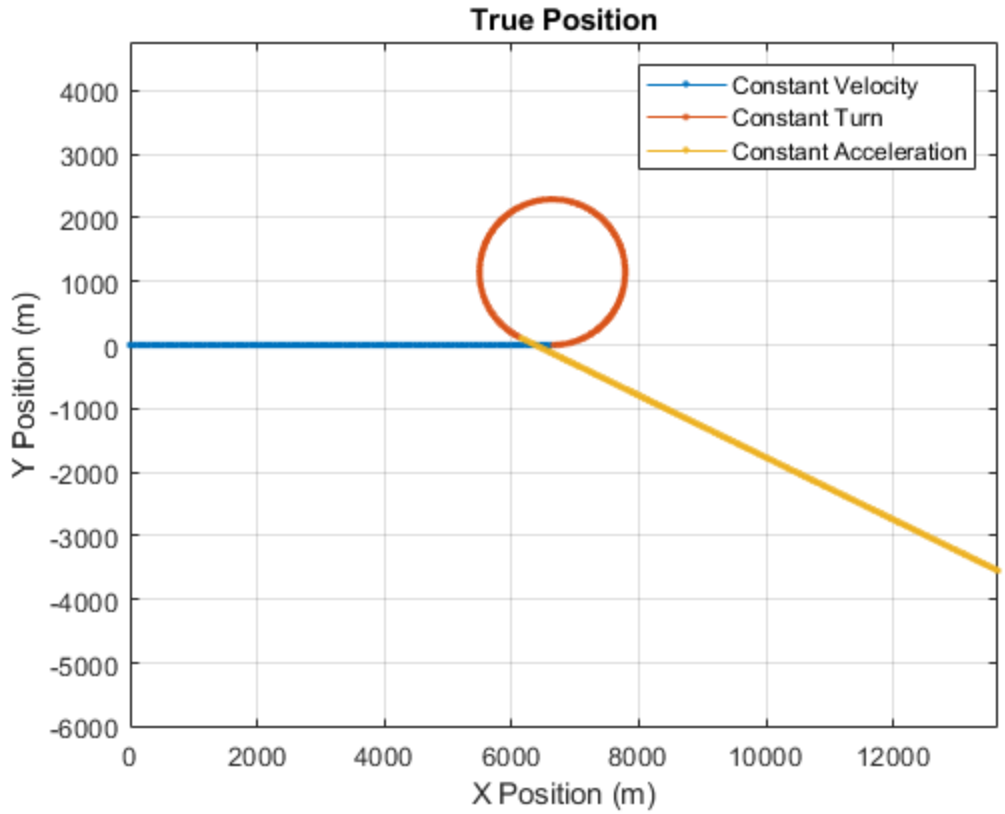
3. Theo dõi Mục tiêu Cơ động

Ví dụ này cho thấy cách theo dõi các mục tiêu cơ động bằng các bộ lọc theo dõi khác nhau. Ví dụ cho thấy sự khác biệt giữa các bộ lọc sử dụng một mô hình chuyển động đơn và nhiều mô hình chuyển động.

Sử dụng lệnh sau trên cửa sổ Command Matlab để mở code mô phỏng: openExample('fusion/TrackingManeuveringTargetsExample')

Trong ví dụ này, bạn xác định một mục tiêu duy nhất ban đầu di chuyển với vận tốc không đổi 200 m / s trong 33 giây, sau đó đi vào vòng quay không đổi 10 độ / s. Vòng quay kéo dài trong 33 giây, sau đó mục tiêu tăng tốc theo đường thẳng với tốc độ 3 m / s^2.

```
[trueState, time, fig1] = helperGenerateTruthData;
dt = diff(time(1:2));
numSteps = numel(time);
figure(fig1)
```

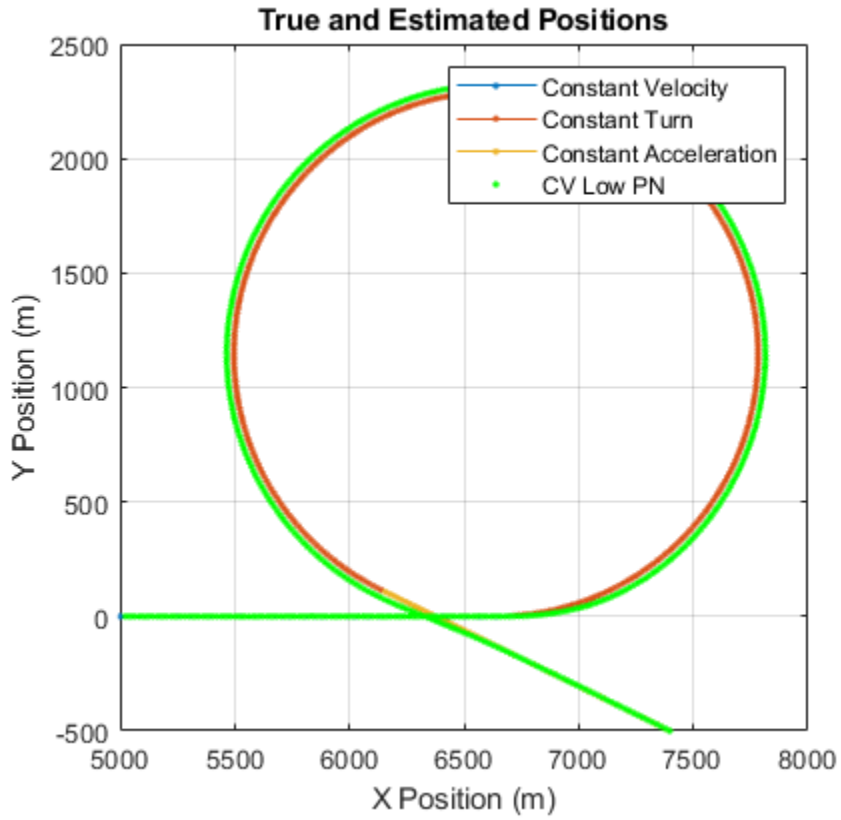


Hình 13. Vị trí thực của mục tiêu

Theo dõi bằng bộ lọc vận tốc không đổi

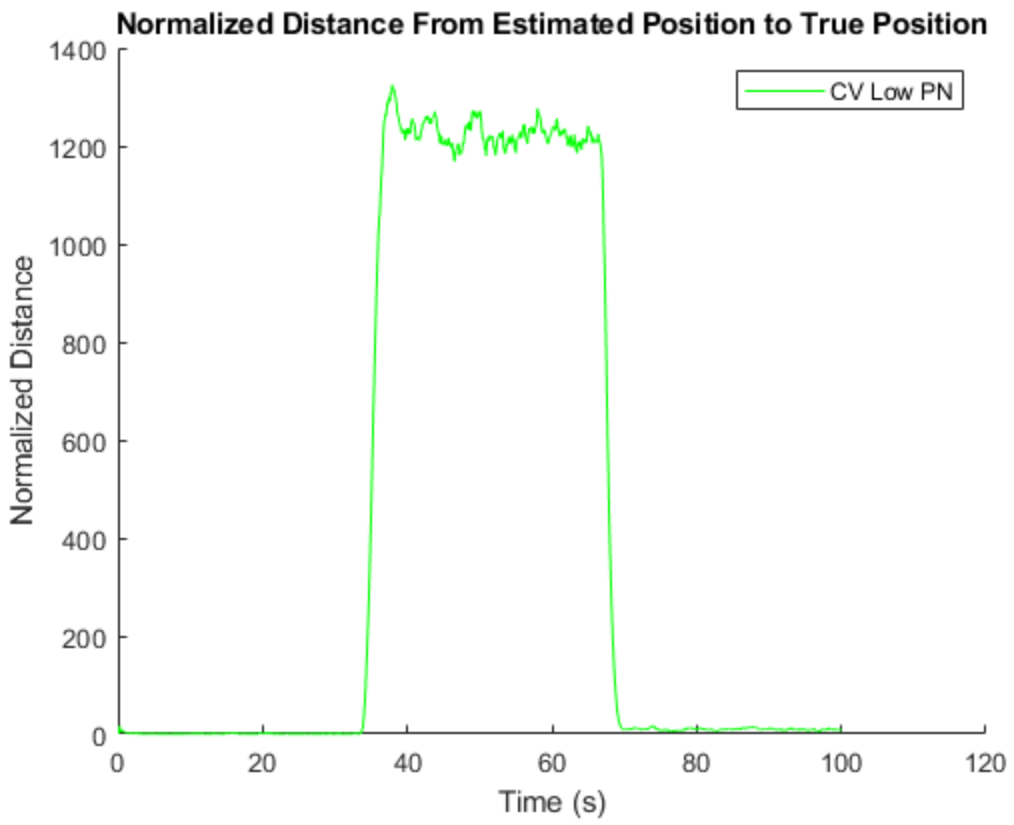
Bạn định nghĩa a với một mô hình chuyển động vận tốc không đổi. Bạn sử dụng phép đo đầu tiên để xác định hiệp phương sai trạng thái và trạng thái ban đầu và đặt nhiễu quá trình thành không phụ gia, để xác định nhiễu quá trình theo gia tốc chưa biết trong các thành phần x, y và z.

Đối với mỗi phép đo, bạn dự đoán bộ lọc, tính toán khoảng cách của trạng thái dự đoán từ vị trí thực và hiệu chỉnh bộ lọc bằng phép đo để có được ước tính đã lọc của vị trí.



Hình 14. Vị trí Thực và Ước tính

Như được mô tả trong hình, bộ lọc có thể theo dõi phần vận tốc không đổi của chuyển động rất tốt, nhưng khi mục tiêu thực hiện lượt, vị trí ước tính của bộ lọc sẽ phân kỳ khỏi vị trí thực. Bạn có thể thấy khoảng cách của ước tính từ sự thật trong kịch bản sau. Trong quá trình rẽ, ở 33-66 giây, khoảng cách chuẩn hóa nhảy lên các giá trị rất cao, điều đó có nghĩa là bộ lọc không thể theo dõi mục tiêu cơ động.



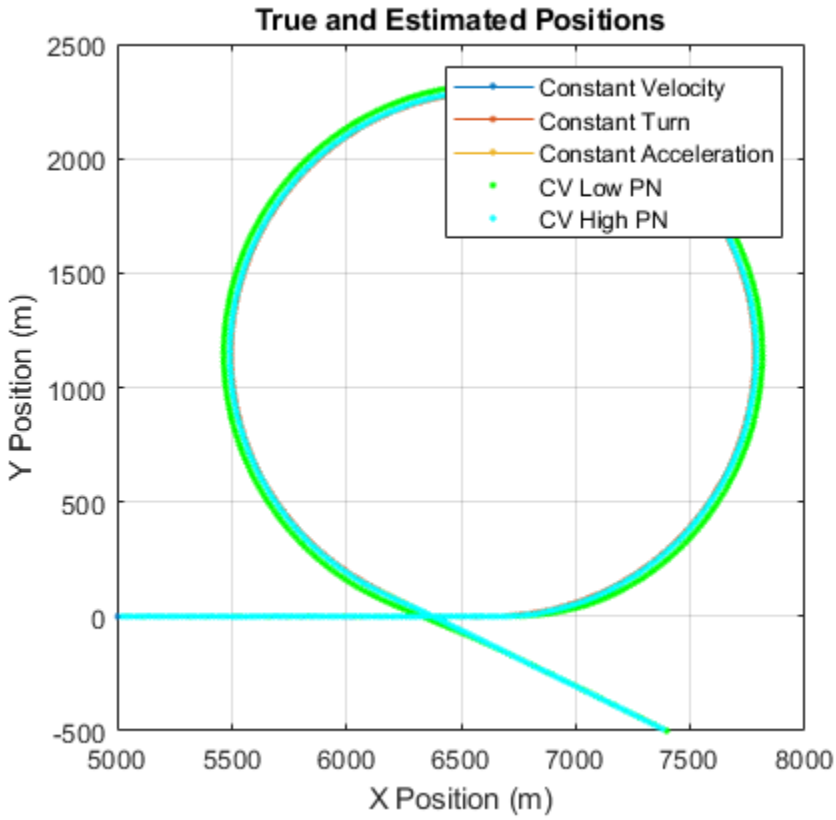
Hình 15. Khoảng cách chuẩn hóa từ vị trí ước tính đến vị trí thực

Tăng nhiễu quá trình

Một giải pháp khả thi là tăng nhiễu quá trình. Nhiễu quá trình đại diện cho các thuật ngữ không được mô hình hóa trong mô hình chuyển động. Đối với một mô hình vận tốc không đổi, đây là những thuật ngữ gia tốc chưa biết. Bằng cách tăng nhiễu quá trình, bạn cho phép độ không chắc chắn lớn hơn trong mô hình chuyển động, điều này làm cho bộ lọc dựa vào các phép đo nhiều hơn so với mô hình. Các dòng sau tạo ra một bộ lọc vận tốc không đổi, với các giá trị tiếng ồn quá trình cao tương ứng với khoảng 5 vòng quay:

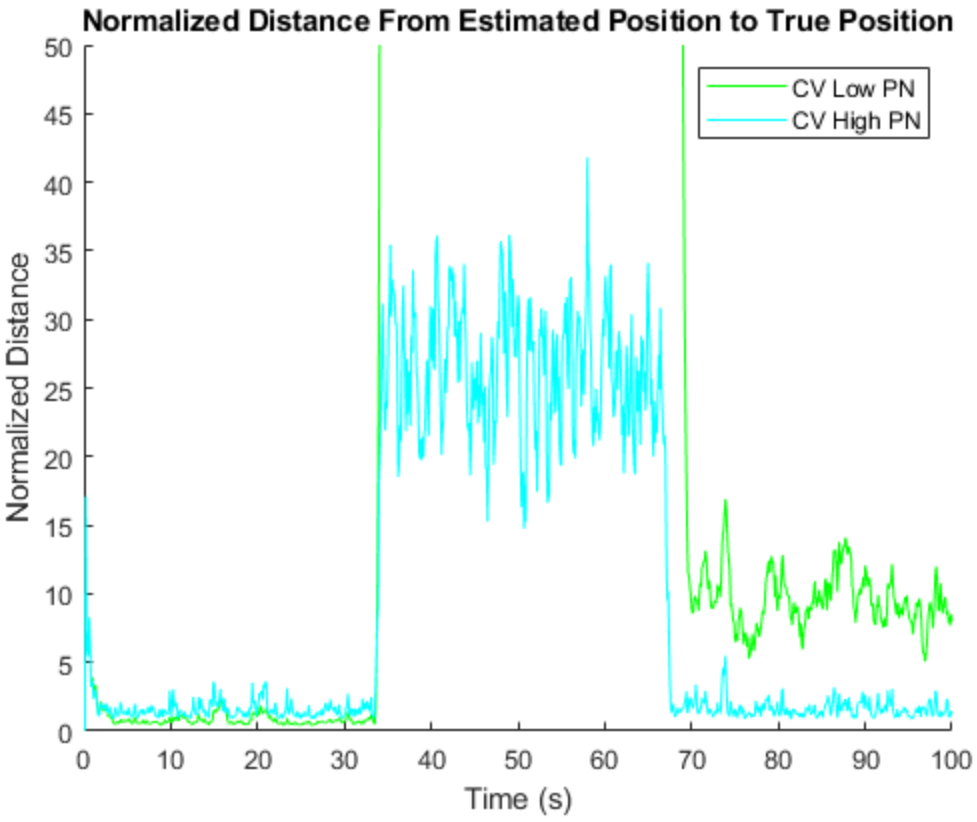
```
cvekf2 = trackingEKF(@constvel, @cvmeas, initialState, ...
    'StateTransitionJacobianFcn', @constveljac, ...
    'MeasurementJacobianFcn', @cvmeasjac, ...
    'StateCovariance', initialCovariance, ...
    'HasAdditiveProcessNoise', false, ...
    'ProcessNoise', diag([50,50,1])); % Large uncertainty in the horizontal
acceleration
dist = zeros(1,numSteps);
estPos = zeros(3,numSteps);
for i = 2:size(measPos,2)
    predict(cvekf2, dt);
```

```
    dist(i) = distance(cvekf2,truePos(:,i)); % Distance from true position
    estPos(:,i) = positionSelector * correct(cvekf2, measPos(:,i));
end
figure(fig1)
plot(estPos(1,:),estPos(2,:),'.c','DisplayName','CV High PN')
axis([5000 8000 -500 2500])
```



Hình 16. Vị trí Thực và Ước tính

Tăng nhiễu quá trình cải thiện đáng kể khả năng theo dõi mục tiêu của bộ lọc trong quá trình rẽ. Tuy nhiên, có một chi phí: bộ lọc ít có khả năng làm mịn nhiễu đo trong khoảng thời gian vận tốc không đổi của chuyển động. Mặc dù khoảng cách chuẩn hóa trong khi rẽ đã giảm đáng kể, khoảng cách chuẩn hóa tăng lên trong 33 giây đầu tiên, trong khoảng thời gian vận tốc không đổi của chuyển động.

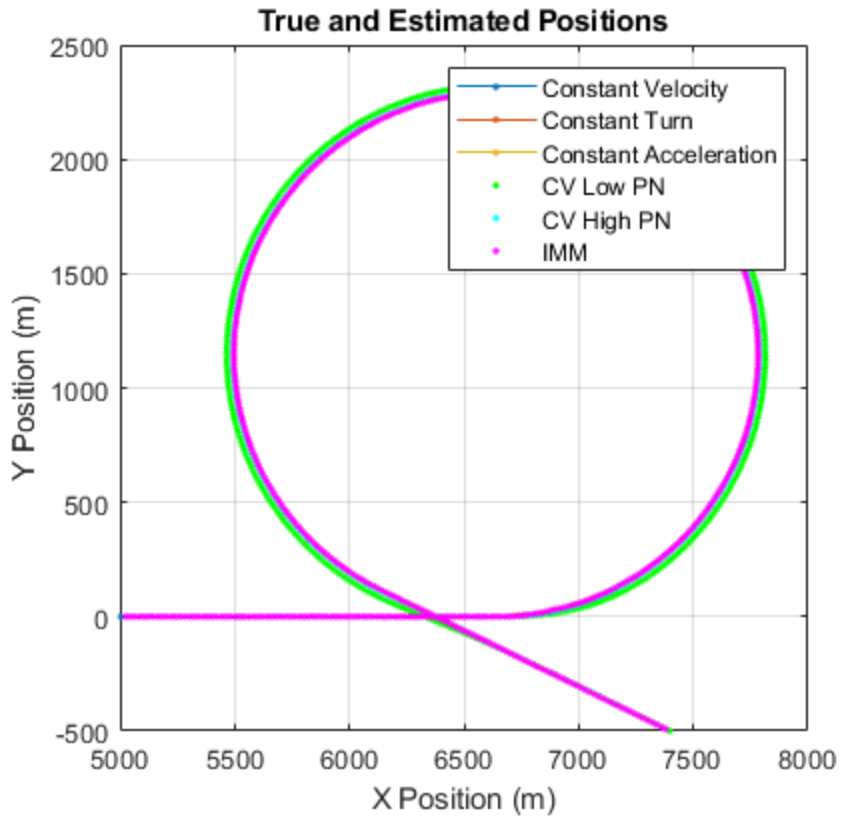


Hình 17. Khoảng cách chuẩn hóa từ vị trí ước tính đến vị trí thực

Sử dụng bộ lọc mô hình chuyển động tương tác

Một giải pháp khác là sử dụng một bộ lọc có thể xem xét tất cả các mô hình chuyển động cùng một lúc, được gọi là bộ lọc nhiều mô hình tương tác (IMM). Bộ lọc IMM có thể duy trì bao nhiêu mô hình chuyển động tùy thích, nhưng thường được sử dụng với 2-5 mô hình chuyển động. Trong ví dụ này, ba mô hình là đủ: mô hình vận tốc không đổi, mô hình quay không đổi và mô hình gia tốc không đổi.

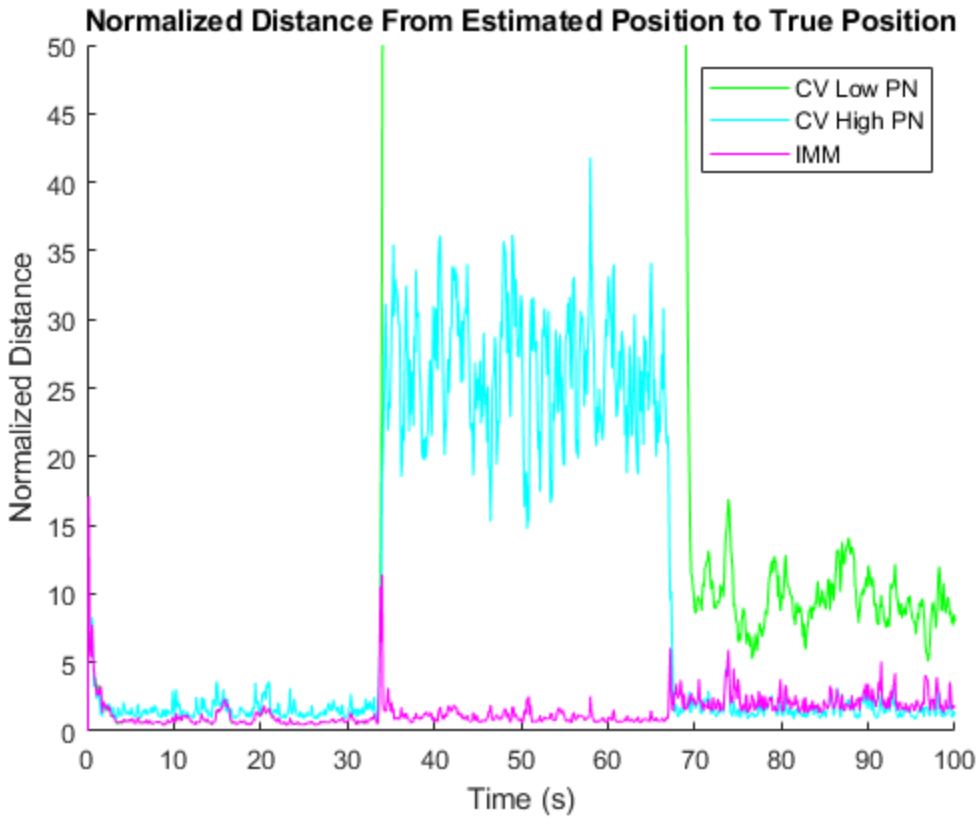
Bộ lọc IMM được sử dụng theo cách tương tự như EKF đã được sử dụng.



Hình 18. Vị trí Thực và Ước tính

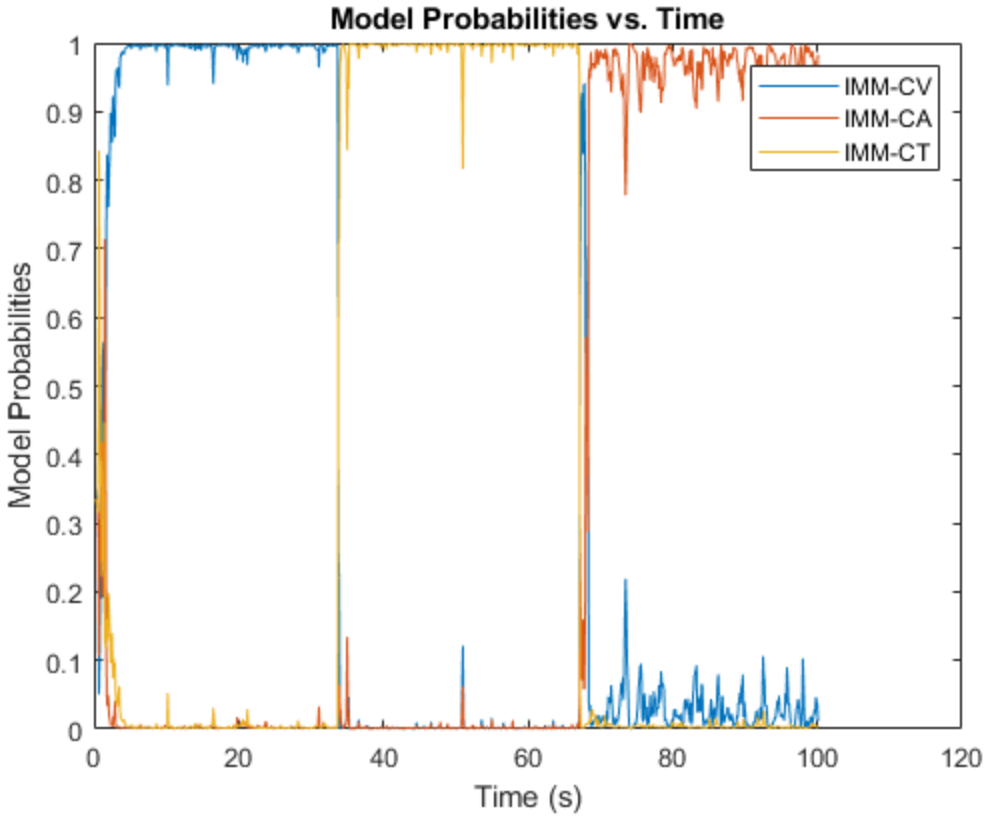
Bộ lọc có thể theo dõi mục tiêu cơ động trong cả ba phần của chuyển động.

Kiểm tra khoảng cách giữa trạng thái dự đoán của bộ lọc và vị trí thực, bạn thấy rằng bộ lọc IMM có thể giảm khoảng cách cho tất cả các phần của chuyển động. Trên thực tế, bộ lọc IMM theo dõi chuyển động tốt hơn cả hai mô hình vận tốc không đổi khác được sử dụng trước đó.



Hình 19. Khoảng cách chuẩn hóa từ vị trí ước tính đến vị trí thực

Để hiểu rõ hơn về cách thức hoạt động của bộ lọc IMM, hãy vẽ xác suất mô hình dưới dạng hàm thời gian. Hình cho thấy bộ lọc được khởi tạo với ba mô hình có cùng xác suất. Khi bộ lọc được cập nhật, nó nhanh chóng hội tụ đến xác suất rất cao rằng mô hình là một mô hình vận tốc không đổi. Sau 33 giây chuyển động, mô hình vận tốc không đổi không còn đúng và xác suất của mô hình quay không đổi trở nên rất cao trong suốt thời gian rẽ. Trong phần cuối cùng của chuyển động, trong quá trình cơ động tăng tốc liên tục, bộ lọc IMM chỉ định xác suất cao rằng chuyển động là gia tốc không đổi, nhưng bộ lọc ít chắc chắn hơn về mô hình chuyển động chính xác và có khoảng 0, 3 xác suất chuyển động vận tốc không đổi.



Hình 20. Mô hình Xác suất với Thời gian

Tóm tắt Ví dụ

Ví dụ này cho bạn thấy cách theo dõi thao tác của mục tiêu với chuyển động quay liên tục và tăng tốc liên tục. Ví dụ cho thấy cách bạn có thể tăng nhiễu quá trình để chụp thao tác không xác định với mô hình vận tốc không đổi. Bạn cũng đã thấy cách cải thiện việc theo dõi mục tiêu cơ động bằng cách sử dụng bộ lọc IMM.

III. Đánh giá kết quả mô phỏng

Điểm nổi bật của các quá trình mô phỏng trên:

- Giúp hiểu rõ hơn về các quá trình Sensor Fusion
- Tính linh hoạt: Cho phép bạn kiểm tra hiệu ứng của việc thay đổi tốc độ lấy mẫu cảm biến.
- Khả năng mô phỏng mất cảm biến: Bằng cách bỏ chọn hộp kiểm tương ứng, giúp mô phỏng tình huống mất cảm biến.
- Tăng độ chính xác: Sử dụng Sensor Fusion giúp cải thiện độ chính xác trong việc nhận biết và theo dõi mục tiêu.

- Mô phỏng môi trường thực tế: Ví dụ cho phép mô phỏng môi trường nhiễu, nơi mà các mục tiêu chỉ xuất hiện với một xác suất phát hiện nhỏ.

Điểm hạn chế của các quá trình mô phỏng trên:

- Nhiễu cảm biến: Nhiễu có thể gây ra đọc dữ liệu không chính xác.
- Hiệu chuẩn cảm biến: Đảm bảo tất cả các cảm biến được căn chỉnh đúng và tạo ra dữ liệu chính xác là một thách thức.
- Đơn vị đo lường, độ phân giải lấy mẫu và sự căn chỉnh không gian-thời gian: Các cảm biến có sự khác biệt về đơn vị đo lường vật lý, độ phân giải lấy mẫu và sự căn chỉnh không gian-thời gian.
- Giả định về độc lập: Một số thuật toán Sensor Fusion giả định rằng các cảm biến là độc lập với nhau, điều này có thể không đúng trong một số trường hợp thực tế.

IV. Thảo luận

1. Một số vấn đề khi sử dụng Sensor Fusion

- Nhiễu từ cảm biến: Các cảm biến trong IMU, bao gồm gia tốc kế, con quay hồi chuyển và cảm biến từ trường, đều chịu ảnh hưởng của nhiễu. Nhiễu này có thể làm giảm độ chính xác của ước lượng hướng.
- Giới hạn về độ chính xác của cảm biến: Các cảm biến gia tốc kế, con quay hồi chuyển và cảm biến từ trường đều có giới hạn về độ chính xác. Điều này có nghĩa là, ngay cả khi không có nhiễu, vẫn có một sai số nhất định trong ước lượng hướng.
- Khả năng chịu đựng lỗi của thuật toán Sensor Fusion: Các thuật toán Sensor Fusion cần phải có khả năng chịu đựng lỗi, tức là, chúng cần phải có khả năng xử lý được các lỗi đầu vào và vẫn cung cấp kết quả ước lượng hướng chính xác.
- Thời gian đáp ứng của thuật toán Sensor Fusion: Các thuật toán Sensor Fusion cần phải có thể xử lý dữ liệu từ cảm biến trong thời gian thực và cung cấp kết quả ước lượng hướng một cách nhanh chóng.
- Độ phức tạp của thuật toán Sensor Fusion: Các thuật toán Sensor Fusion thường rất phức tạp và đòi hỏi kiến thức sâu về cảm biến và xử lý tín hiệu. Điều này có thể gây khó khăn cho việc triển khai và tối ưu hóa các thuật toán này.

- Cần phải tối ưu hóa các tham số của thuật toán Sensor Fusion: Đối với một số thuật toán Sensor Fusion, có thể cần phải tối ưu hóa các tham số của thuật toán để đạt được kết quả tốt nhất. Việc tìm ra bộ tham số tối ưu có thể đòi hỏi thực hiện nhiều lần thử nghiệm

2. Một số cách khắc phục

- Lọc nhiễu: Sử dụng các thuật toán lọc như bộ lọc Kalman hoặc bộ lọc Complementary để giảm nhiễu từ dữ liệu cảm biến.
- Sử dụng cảm biến chất lượng cao: Cảm biến chất lượng cao có thể cung cấp dữ liệu chính xác hơn, giúp cải thiện độ chính xác của ước lượng hướng.
- Tối ưu hóa tham số của thuật toán Sensor Fusion: Một số thuật toán Sensor Fusion cho phép tối ưu hóa các tham số của thuật toán để đạt được kết quả tốt nhất. Việc tìm ra bộ tham số tối ưu có thể đòi hỏi thực hiện nhiều lần thử nghiệm.
- Sử dụng các thuật toán hiện đại: Có nhiều thuật toán Sensor Fusion hiện đại có thể giúp cải thiện độ chính xác và thời gian đáp ứng của ước lượng hướng. Ví dụ, thuật toán Madgwick và Mahony là hai thuật toán phổ biến được sử dụng trong Sensor Fusion.
- Đào tạo và kiểm tra: Đào tạo và kiểm tra thuật toán Sensor Fusion trên dữ liệu thực tế có thể giúp cải thiện hiệu suất của thuật toán.

V. Kết luận

Sensor Fusion là một quá trình quan trọng trong việc tăng cường độ chính xác và độ tin cậy của dữ liệu thu thập từ nhiều cảm biến. Nó giúp giảm thiểu lỗi và nhiễu, vượt qua các hạn chế của từng cảm biến và cung cấp một cái nhìn toàn diện hơn về môi trường.

Tuy nhiên, Sensor Fusion không phải lúc nào cũng hoàn hảo. Có những thách thức như việc đảm bảo tương thích giữa các cảm biến, hiệu chuẩn cảm biến, giả định về sự độc lập giữa các cảm biến, và sự không chắc chắn trong nguồn dữ liệu.

Dù vậy, với sự tiến bộ của công nghệ và nghiên cứu, những hạn chế này đang dần được khắc phục, mở ra triển vọng rộng lớn cho việc áp dụng Sensor Fusion trong nhiều lĩnh vực như tự động hóa, robot, xe tự lái, và nhiều hơn nữa.

Nhìn chung, Sensor Fusion là một công cụ mạnh mẽ, cho phép chúng ta tận dụng tối đa sức mạnh của các cảm biến và tạo ra hệ thống thông minh hơn, linh hoạt hơn và hiệu quả hơn.

VI. Tài Liệu Tham Khảo

1. Lecture notes on Basics of Sensor Fusion - Roland Hostettler and Simo Sarkka Uppsala University, Sweden;Aalto University, Finland, September 3, 2020

2. H. Gharavi and S. P. Kumar, Eds., Proceedings of the IEEE: Special Issue on Sensor Networks and Applications, vol. 91, IEEE, August 2003.

3. Sensor Fusion - J.Z. Sasiadek Department of Mechanical & Aerospace Engineering, Carleton University.

4. Sensor fusion: A review of methods and applications - https://ieeexplore.ieee.org/document/7979175

5. Sensor Fusion: The Ultimate Guide to Combining Data for Enhanced Perception and Decision-Making - https://www.wevolver.com/article/what-is-sensor-fusion-everything-you-need-to-know

6. Estimating Orientation Using Inertial Sensor Fusion and MPU-9250 - https://www.mathworks.com/help/fusion/ug/Estimating-Orientation-Using-Inertial-Sensor-Fusion-and-MPU-9250.html;jsessionid=7beff27f7b2758d79ef62dc0a27f?s_eid=PSM_15028

7. Pose Estimation From Asynchronous Sensors - https://www.mathworks.com/help/fusion/ug/pose-estimation-from-asynchronous-sensors.html?s_eid=PSM_15028

8. Tracking Maneuvering Targets - https://www.mathworks.com/help/fusion/ug/tracking-maneuvering-targets.html?s_eid=PSM_15028

9. Sensor Fusion and Tracking Toolbox - https://www.mathworks.com/help/fusion/index.html?s_tid=CRUX_lftnav.