Reconstitution de la position à partir de données inertielles

Cette note d'application a pour but de présenter les possibilités pour reconstruire une position à partir de données inertielles. Dans ce document, les données inertielles utilisées dans les exemples ont été obtenues via une centrale inertielle, ADIS16364.

Note d'application

Julien Monmasson

26/01/2010

Sommaire

I. Présentation

Projet P09B08

Cette étude, concernant l'odométrie ferroviaire par fusion de données accélérométriques et GPS, a été effectuée dans le cadre de notre cursus universitaire et demandée par la société Ansaldo STS, fabricant de systèmes de sécurité et de signalisation pour les transports publics : ferroviaires et métropolitains. En fonction des résultats, ce type de produit pourrait remplacer les systèmes actuels, onéreux et encombrants comparés aux matériels MEMS.

L'enjeu du projet était d'étudier les performances de matériels accélérométriques et gyrométriques à base de technologie MEMS (micro-electromecanical systems) dans le cadre de la reconstitution sécuritaire de trajectoire ferroviaire. Cette sécurité était obtenue à l'aide d'une fusion de données GPS et accélérométriques. Pour cela, une carte de test et des algorithmes de traitement ont été créés.

Chaine d'acquisition

Le but étant de reconstituer la position, il est nécessaire dans un premier temps d'acquérir les données inertielles et GPS.

Cette acquisition a été effectué par liaison série. Une interface MATLAB® permettait ensuite de stocker les données dans des vecteurs utilisés pour le traitement. Pour plus d'information concernant l'acquisition des données par interface série, voir la note d'application de Mr. Doucement, « **Acquisition de données séries binaires via une interface graphique Matlab®** ».

II. Théorie

Cette partie est un résumé de le théorie utilisée dans les algorithmes de traitement de la partie suivante. Il est expliqué les relations entre les vecteurs du mouvement et la transformation a appliquer pour changer ces vecteurs de repère.

Relation entre position, vitesse et accélération

Cette partie est un rappel des relations mathématiques entre les composantes du mouvement. Ces relations sont valables pour des transformations dans un même repère. Nous verrons par la suite que cette hypothèse n'est pas valable dans notre cas.

Notre référentiel est trois dimensions, le vecteur position s'écrira alors :

$$\overrightarrow{Pos}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

La vitesse $\vec{V}$ est obtenue en dérivant le vecteur position par le temps.

$$\vec{V}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{pmatrix} = \overrightarrow{\dot{Pos}}(t) = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{pmatrix}$$

L'accélération $\overrightarrow{Acc}$ est elle obtenue en dérivant le vecteur vitesse par le temps.

$$\overrightarrow{Acc}(t) = \begin{pmatrix} Acc_x(t) \\ Acc_y(t) \\ Acc_z(t) \end{pmatrix} = \dot{\vec{V}}(t) = \begin{pmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{pmatrix}$$

Mécanique : Principe fondamental de la dynamique

Le principe fondamental de la dynamique est la deuxième loi de Newton. Celle-ci détermine la force en fonction de la dérivée du produit de la vitesse et de la masse de l'objet. Ce qui en termes d'équation donne :

$$\vec{F} = \frac{d}{dt}(m \times \vec{V})$$

Cette loi peut être le plus souvent simplifiée car la dérivée de la masse par rapport au temps est nulle. De ce fait la formule devient :

$$\vec{F} = M \times \overrightarrow{Acc}$$

Transformation de repères

La transformation de repère permettra dans notre cas de passer du repère mobile au repère géocentrique.

Un repère peut subir toutes les transformations existantes. Dans notre cas, seul la translation et la rotation pourrait être utilisé. En faite, seule la rotation sera utilisée.

Soient deux repères R1=(O,x,y,z) et R2=(O',x',y',z') et deux vecteurs **P**=(a,b,c) et **P'**=(a',b',c') respectivement dans R1 et R2. Pour que **P'** soit la représentation de **P** dans R1 il faudra alors que

P = TP'

Avec T la matrice de transformation de R2 dans R1. Cette matrice est composée des composantes de rotation et de translation des repères.

Notre cas nous permet de n'utiliser que la rotation. En effet, le but de la transformation sera de passer des vitesse du repère mobile aux vitesse dans le repère géocentrique. Cette transformation n'a pour but que de tourné le repère mobile pour avoir les vitesses dans le repère géocentrique. Une autre hypothèse simplificatrice est exploitée, le point de référence du vecteur vitesse et le point de référence du repère mobile. Il est possible dans ce cas d'utiliser la DCM (Direction Cosine Matrix). Finalement, la relation sera :

V = RV'

Avec R, la DCM définit par :

$$R = \begin{bmatrix} \cos(\theta)\cos(\Psi) & -\cos(\varphi)\sin(\Psi) + \sin(\varphi)\sin(\theta)\cos(\Psi) & \sin(\varphi)\sin(\Psi) + \cos(\varphi)\sin(\theta)\cos(\Psi) \\ \cos(\theta)\sin(\Psi) & \cos(\varphi)\cos(\Psi) + \sin(\varphi)\sin(\theta)\sin(\Psi) & -\sin(\varphi)\cos(\Psi) + \cos(\varphi)\sin(\theta)\sin(\Psi) \\ -\sin(\theta) & \sin(\varphi)\cos(\theta) & \cos(\varphi)\cos(\theta) \end{bmatrix}$$

$\varphi, \theta \; et \; \Psi$ sont les trois angles des axes X, Y et Z. Ces angles seront obtenus en intégrant la valeur du gyromètre qui délivre des vitesses de rotation.

Transformation deux dimensions

Lors de la mise en pratique, nous nous sommes aperçu que les résultats étaient meilleurs en enlevant une dimension, la hauteur. Les relations changent peu. Seul la rotation est différente.

Désormais les vecteurs seront de ce type :

$$\overrightarrow{Acc}(t) = \begin{pmatrix} Acc_x(t) \\ Acc_y(t) \\ 0 \end{pmatrix} = \vec{V}(t) = \begin{pmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ 0 \end{pmatrix}$$

La rotation à effectuer sera uniquement celle concernant la direction. Après l'intégration de la vitesse de rotation selon l'axe des Z nous obtenons Ψ et cet angle permet de déterminer le vecteur dans un nouveau repère :

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Le choix de garder 3 composantes pour un vecteur a pour but de ne pas changer la taille des vecteurs dans nos algorithmes.

III. Mise en pratique

Les premiers résultats obtenus montrent un bruit blanc biaisé. La mise en place d'un filtre passe bas permet de lisser les valeurs et de mieux comprendre les résultats obtenus.

L'application ne fonctionne pas en temps réel, c'est-à-dire que le traitement est effectué après l'acquisition des données. Un exemple de fonction fonctionnant en temps réel est présenté en annexe

Filtrage

Par manque de temps, le filtrage fut un simple filtre à réponse impulsionnelle infinie (RII). La valeur de la fréquence de coupure normalisée était de 0.01 et d'ordre 4.

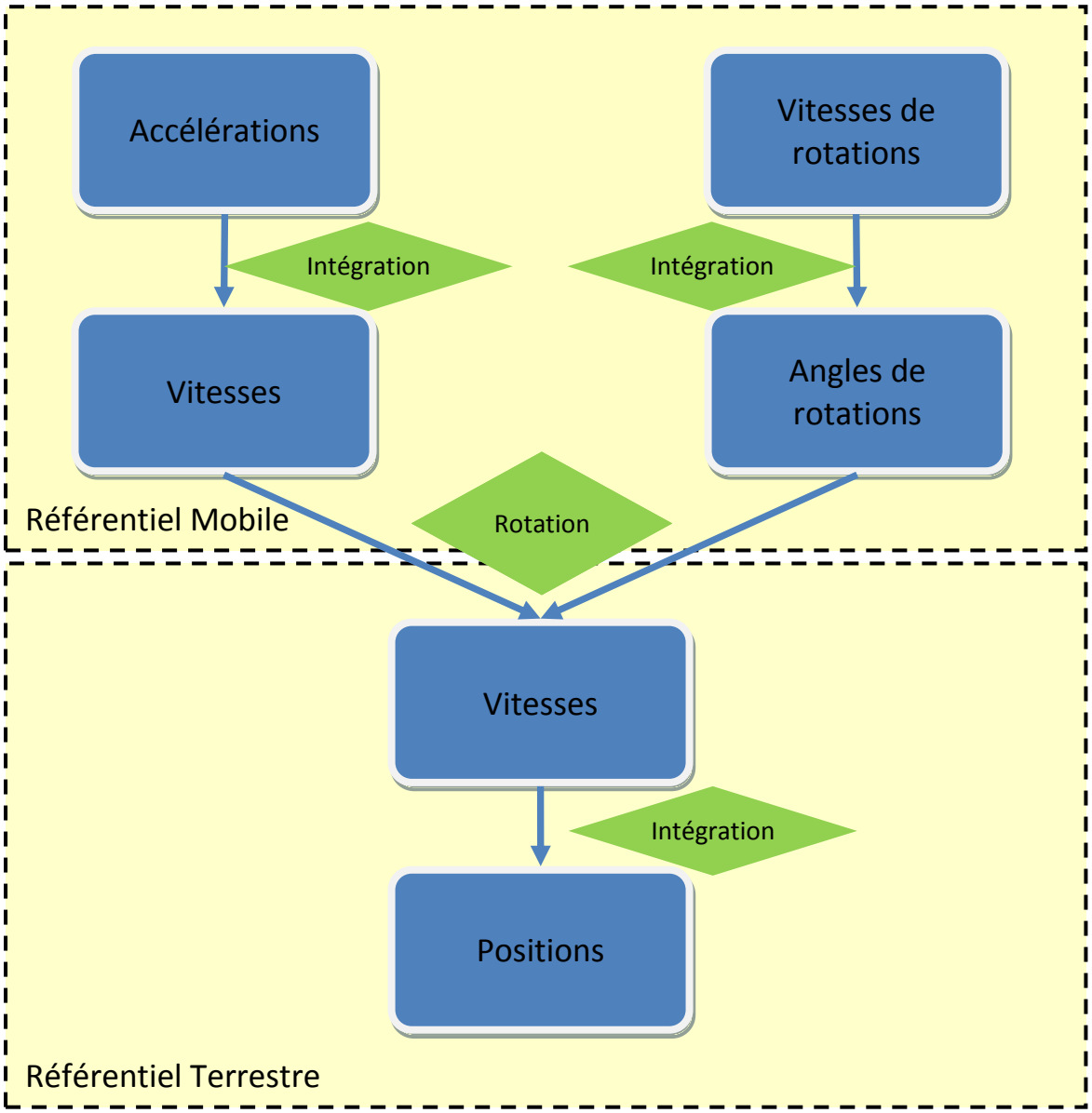
Le biais n'est pas filtré par le filtre passe bas. Cependant cette valeur, sur les accéléromètres, peut être significative. Premièrement, l'accéléromètre perçoit en continue l'effet de l'attraction terrestre, notée G. Cette valeur peut se répercuté sur toutes les composantes X, Y et Z en fonction de la topologie du terrain. Ce biais ne peut donc pas être considéré comme un défaut du système.

Il est cependant difficile de le traiter ces valeur pour obtenir des conclusions sur la topologie du terrain. Cette difficulté n'a pas pu être élucidée au terme du projet. Pour contourner le problème, nous avons décidé de travailler en 2 dimensions et de calibrer à chaque arrêt, c'est-à-dire supprimer ce biais.

Application

Dans cette section sera présenté les algorithmes utilisés pour traiter les données inertielles.

Schéma de traitement



Filtrage

Cette fonction utilise les fonctions butter et filter de MATLAB®. Ces deux fonctions permettent respectivement de déterminer les composantes du filtre RII et de l'appliquer au vecteur.

passe_bas.m

```
function [vec_sortie] = passe_bas(ordre,freq_ech,freq_coup,vec_entree)

% Fonction filtre passe bas

%       Projet Génie électrique 2008-2010 : Odométrie ferroviaire
%       Projet : P09AB08
%       Client :  Mr Clairet pour Ansaldo STS
%       Nom du fichier : passe_bas.m
%       type du fichier : fonction
%       Dernière modification : 11/12/09
%       par : Jean Doucement, Julien Monmasson

%       ordre : ordre du filtre
%       freq_ech : fréquence d'échantillonnage du système
%       freq_coup : fréquence de coupure souhaitée (0<freq_coup<freq_ech)
%       vec_entree : vecteur à filtrer
%       vec_sortie : vecteur filtré

freq_coup_norm = freq_coup./freq_ech;
[A, B] = butter(ordre, freq_coup_norm);
vec_sortie = filter(A,B,vec_entree);
```

La fonction qui suit permet de remettre à zéro les valeurs lorsque l'on détecte un arrêt. L'hypothèse faite pour détecter un arrêt consiste à comparer la valeur de la variance du signal en fonctionnement à celui de la variance du signal au repos (qui doit être calculée au préalable).

auto_calib.m

```
function [X,cpt] = auto_calib(Y,mini,Te,temp_comparaison)

% fonction de calibration à chaud

%       Projet Génie électrique 2008-2010 : Odométrie ferroviaire
%       Projet : P09AB08
%       Client :  Mr Clairet pour Ansaldo STS
%       Nom du fichier : auto_calib.m
%       type du fichier : fonction
%       Dernière modification : 20/12/09
%       par : Jean Doucement, Julien Monmasson

%       y : vecteur 6 colonne représentant [acc gyr]
%       mini : minimum de la variance sur le tmeps de comparaison
%       Te : periode d'échantillonnage
%       temp_comparaison : temps sur lequel la comparaison est faite
%       X : vecteur traité
%       cpt : nombre de calibration effectué


nb_point=temp_comparaison/Te;
X=zeros(size(Y));
```

```
i=1;
j=1;
cpt(1)=0;
while(i<= length(Y)-nb_point)
    if min(abs(var(Y(i:nb_point+i,:))) < mini)==1
      Y_att(1:length(Y)-i+1,1) = Y(i:length(Y),1)-mean(Y(i:nb_point+i,1));
      Y_att(1:length(Y)-i+1,2) = Y(i:length(Y),2)-mean(Y(i:nb_point+i,2));
      Y_att(1:length(Y)-i+1,3) = Y(i:length(Y),3)-mean(Y(i:nb_point+i,3));
      Y_att(1:length(Y)-i+1,4) = Y(i:length(Y),4)-mean(Y(i:nb_point+i,4));
      Y_att(1:length(Y)-i+1,5) = Y(i:length(Y),5)-mean(Y(i:nb_point+i,5));
      Y_att(1:length(Y)-i+1,6) = Y(i:length(Y),6)-mean(Y(i:nb_point+i,6));
      Y(i:length(Y),:) = Y_att(1:length(Y)-i+1,:);
      i=i+nb_point;
      j=j+1;
      cpt(j)=i;
    else
        X(i,:)=Y(i,:);
        i=i+1;
    end
end
X=Y;
end
```

Intégration

integration.m

```
function [ G ] = integration(U, G0, Te)
%                     fonction d'intégration                          %
%
%       Projet Génie électrique 2008-2010 : Odométrie ferroviaire
%       Projet : P09AB08
%       Client :  Mr Clairet pour Ansaldo STS
%       Nom du fichier : integration.m
%       type du fichier : fonction
%       Dernière modification : 07/10/09
%       par : Jean Doucement, Julien Monmasson
%
%       Fonctions d'intégration de deux ou trois composantes
%       d'accélération
%       en fonction du vecteur d'entrée.
%      entrée U matrice 2x2 ou 2x3, les lignes représentent les instants n
%       et n-1.
%       exemple  :
%                    |                                  |
%                    |  U_x(n-1) U_y(n-1)    U_z(n-1)   |
%                    |  U_x(n)   U_y(n)      U_z(n)     |
%                    |_                                 _|
%       G0 est la valeur initiale de l'intégration sur les différents
%       axes.
%       exemple :
%                    |_ G0_x     G0_x        G0_z    _|
%
%       Te est la période d'échantillonnage du système en secondes.

n = size(U);
if n(1)==2
         G = (U(1,:)+ U(2,:))/2*Te + G0 ;
else
    error('n doit etre égal à 2')
end
end
```

Rotation

Application de la DCM :

rot.m

```
function [v_rot] = rot(v,angle)
% Fonction rotation par DCM
%       Projet Génie électrique 2008-2010 : Odométrie ferroviaire
%       Projet : P09AB08
%       Client :  Mr Clairet pour Ansaldo STS
%       Nom du fichier : rot.m
%       type du fichier : fonction
%       Dernière modification : 02/11/09
%       par : Jean Doucement, Julien Monmasson
%
%       Les angles sont en radians
%       angle : angles d'Euler sur les trois axes
%               ( X , Y , Z )
%       v : vecteur d'entrée
%       v_rot : vecteur transformé

devers = angle(1);
pente = angle(2);
direction = angle(3);
c1 = cos(devers);
s1 = sin(devers);
c2 = cos(pente);
s2 = sin(pente);
c3 = cos(direction);
s3 = sin(direction);

M=[c3*c2     s3*c1-c3*s2*s1   s3*s1+c3*s2*c1;
   -s3*c2    c3*c1+s3*s2*s1   c3*s1-s3*s2*c1;
   -s2       -c2*s1           c2*c1];

v_rot = v*M;
```

Rotation par la direction:

rotz.m

```
function [v_rotz] = rotz(v,alpha)

%renvoie la rotation de v suivant z d'angle alpha (en radian)


c = cos(alpha);
s = sin(alpha);

v_rotz = v*[c s 0;-s c 0;0 0 1];
```

Script principal

Ce script utilise un fichier de donnée MATLAB® expliqué dans l'entête du script suivant. La fin du script affiche les résultats obtenus. L'algorithme n'utilise que trois valeur. L'accélération sur X et Y et la vitesse de rotation sur Z. Pour utiliser les 6 composantes, l'initialisation est a changer, ainsi que le traitement.

sans_gps.m

```
% Script pour un traitement uniquement des accélérations

%       Projet Génie électrique 2008-2010 : Odométrie ferroviaire
%       Projet : P09AB08
%       Client :  Mr Clairet pour Ansaldo STS
%       Nom du fichier : sans gps.m
%       type du fichier : script
%       Dernière modification : 04/01/2010
%       par : Jean Doucement, Julien Monmasson
%
%       structure du fichier d'entrée :
%           4 champs :  lignes
%                       lignes_decimal
%                       nom
%                       out
%
%       entrée du script est lignes_decimal au format :
%
%                                       7 colonnes
%                   (                                           )
%       N lignes    (ascii_gps, gyr (x , y , z) , acc (x , y , z))
%                   (                                           )




% fermeture des fenetres et supressions des variables
    close all;
    clear all;

%initialisation des variables
    %variable de conversion et de fréquence
    convert = pi/180;
    te_gps = 1 ;            % période des acquisitions GPS
    te_imu = 1/500;         % période des acquisitions IMU
    G = 9.80665;
    conversion_acc = 0.001*G;
    conversion_gyr = 0.05*pi/180;
    A =[0;0]; % vecteur colonne de zéros

    %lecture des valeurs de centrale inertielle

    % indiquer le chemin complet du fichier texte s'il ne se trouve pas
    % dans le répertoire courant
    donnees = importdata('tour_stade.mat');
    trames=donnees.lignes_decimal;
    nb = size(trames);      % Taille du fichier à traiter
```

```
    % Suppression des premières lignes fausses
    trames(1:4,:)=0;

    % séparations des accélérations et des vitesses de rotation
    acc=-trames(:,5:7)*conversion_acc;
    gyr=trames(:,2:4)*conversion_gyr;

    nb_acc=size(acc); %Tailles des vecteurs Acc et gyr

    temps= [0:te_imu:(nb_acc(1)-1)*te_imu]; %variable pour les graphes
                                            %temporels


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  INITIALISATION                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %initialisation des variables de calcul
    position_acc=zeros(nb_acc(1),3);
    vitesse_acc=zeros(nb_acc(1),3);
    vitesse_r=zeros(nb_acc(1),3);
    acc_f=zeros(nb_acc(1),3);
    gyr_f=zeros(nb_acc(1),3);
    angle= zeros(nb_acc(1),3);
    acc_r=zeros(nb_acc(1),3);

    % initialisation de l'angle
        % utilisation 3D
    angle(1,1:3) = 0;
    angle(:,3)= 0; % pour l'utilisation sans GPS direction initiale nulle

    % Suppression de la valeur moyenne des premières des premiers points

     acc_f(:,:)= [acc(:,1)-mean(acc(11:410,1)) acc(:,2)-...
         mean(acc(11:410,2)) acc(:,3)-mean(acc(11:410,3))];
     gyr_f(:,:)= [gyr(:,1)-mean(gyr(11:410,1)) gyr(:,2)-...
         mean(gyr(11:410,2)) gyr(:,3)-mean(gyr(11:410,3))];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% filtrage des données, à commenter si nécessaire %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    acc_f(:,1)=passe_bas(4,1/te_imu,1,acc_f(:,1));
    acc_f(:,2)=passe_bas(4,1/te_imu,1,acc_f(:,2));
    acc_f(:,3)=passe_bas(4,1/te_imu,1,acc_f(:,3));
    gyr_f(:,1)=passe_bas(4,1/te_imu,10,gyr_f(:,1));
    gyr_f(:,2)=passe_bas(4,1/te_imu,10,gyr_f(:,2));
    gyr_f(:,3)=passe_bas(4,1/te_imu,10,gyr_f(:,3));


    % temp = auto_calib( [acc_f gyr_f],[0.0005 0.0005 0.0005 0.0005 0.0005
    %   ...0.0005,te_imu,3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          TRAITEMENT DES DONNEES                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calcul des positions
    for i = 2 : nb_acc(1)

        % intégration des vitesses de rotation
        angle(i,:) = integration(gyr_f(i-1:i,:),angle(i-1,:),te_imu);

        % intégration des accélérations (A = [0;0])
        vitesse_acc(i,:)= integration([acc_f(i-1:i,1 :2) A],...
            vitesse_acc(i-1,:),te_imu);

        % impossibilité d'avoir une vitesse négative sur x
        if vitesse_acc(i,1)<0
            vitesse_acc(i,1)=0;
        end

        % rotation des vitesses
        vitesse_r(i,:)=rot([vitesse_acc(i,1:2) 0],[ 0 0 angle(i,3)]);

        % intégration des vitesses
        position_acc(i,:) = integration(vitesse_r(i-1:i,:),...
            position_acc(i-1,:),te_imu);
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          FIN DE TRAITEMENT                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Graphs des résultats
    % accélérations, vitesses sur le mobile
        hold on
        plot(temps,acc(:,1),'black')
        plot(temps,acc_f(:,1),'r')
        plot(temps,vitesse_acc(:,1),'g')
        xlabel('(s)');
        legend('acceleration (m.s^-^2)',...
            'acceleration filtrée(m/s^-^2)','vitesse mobile (m/s)');
        title('X axis acceleration et vitesse');

        figure
        hold on
        plot(temps,acc(:,2),'black')
        plot(temps,acc_f(:,2),'r')
        plot(temps,vitesse_acc(:,2),'g')
        xlabel('(s)');
        legend('acceleration (m.s^-^2)',...
            'acceleration filtrée(m/s^-^2)','vitesse mobile (m/s)');
        title('Y axis acceleration et vitesse');

        figure
        hold on

        plot(temps,acc(:,3),'black')
        plot(temps,acc_f(:,3),'r')
        plot(temps,vitesse_acc(:,3),'g')
        xlabel('(s)');
```

```
    legend('acceleration (m.s^-^2)',...
        'acceleration filtrée(m/s^-^2)','vitesse mobile (m/s)');
    title('Z axis acceleration et vitesse');

% Angles et vitesses de rotation

    figure
    hold on
    plot(gyr_f(:,1),'black')
    plot(angle(:,1),'r')
    legend('vitesse de rotation','angle')
    title('devers');

    figure
    hold on
    plot(gyr_f(:,2),'black')
    plot(angle(:,2),'r')
    legend('vitesse de rotation','angle')
    title('pente');

    figure
    hold on
    plot(temps,gyr_f(:,3),'black')
    plot(temps,angle(:,3),'r')
    xlabel('(s)');
    legend('vitesse de rotation (rad/s)','angle(rad)')
    title('direction');

% Graph 3D
    figure
    plot3(position_acc(:,1),position_acc(:,2),position_acc(:,3))
    title('Graph 3D : courbe')
    xlabel('X(m)')
    ylabel('Y(m)')
    zlabel('Z(m)')
    axis equal
```

Contact

Pour toutes questions à ce sujet, vous pouvez envoyer un e-mail à cette adresse :

mail Julien Monmasson

Annexe

Fonction pour le temps réel

calcul_temp_R

```
function [pos]=calcul_temp_R(ligne_decimal,duree_attente)

    convert = pi/180;
    te_gps = 1 ;% periode des acquisitions GPS
    te_imu = 1/500; %période des acquisitions IMU
    G = 9.80665;
    conversion_acc = 0.001*G;
    conversion_gyr = 0.05*pi/180;
    A=[0;0];

    persistent etat;
    persistent indice;
    persistent acc_c;
    persistent gyr_c;
    persistent nb;
    persistent gyr_f;
    persistent acc_f;
    persistent moy_acc;
    persistent moy_gyr;
    persistent angle;
    persistent vitesse;
    persistent vitesse_cart;
    persistent pos_p;

    acc=-ligne_decimal(5:7)*conversion_acc;
    gyr=ligne_decimal(2:4)*conversion_gyr;

    if (isempty (etat))
        etat = 1;
    end
    if isempty(angle)
        angle(2:4,:)=[0 0 0;
                      0 0 0;
                      0 0 0];
    end

    if isempty(vitesse)
        vitesse(2:4,:)=[0 0 0;
                        0 0 0;
                        0 0 0];
    end

    if isempty(vitesse_cart)
        vitesse_cart(2:4,:)=[0 0 0;
                        0 0 0;
                        0 0 0];
    end
    if isempty(pos_p)
        pos_p=[0 0 0];
    end
    switch etat
        case 1
            nb=duree_attente/te_imu;
            if isempty (indice)
```

```
                indice=1;
            end
            while (indice<nb)
                acc_c(indice,:)=acc;
                gyr_c(indice,:)=gyr;
                indice=indice+1;
                pos=[ 0 0 ];
                return
            end
            if (indice == nb)
                acc_c(indice,:)=acc;
                gyr_c(indice,:)=gyr;
                moy_acc=mean(acc_c);
                moy_gyr=mean(gyr_c);
                gyr_c(2,:)=  0;
                acc_c(2,:)=  0;
                gyr_f=  gyr_c;
                acc_f=  acc_c;
                indice = 2;
                etat = 2;
                pos=[0 0];
                return
            else
                error('indice violated')
            end
        case 2

            indice_moins=indice;
            indice = mod(indice-2,nb)+1;

            acc_c(indice,:)=acc-moy_acc;
            gyr_c(indice,:)=gyr-moy_gyr;

            acc_f(indice,:) = 0.05*acc_c(indice,:) + ...
                0.95*acc_f(indice_moins,:) ;

            angle(indice,:)=integration([gyr_c(indice_moins,:);...
                gyr_c(indice,:)],angle(indice_moins,:),te_imu);
            if min(abs(var(acc_f))<0.02)==1
                vitesse(indice,:)=.95*vitesse(indice_moins,:);
            else
                vitesse(indice,:)= integration([acc_c(indice_moins,1) 0 0;
                                                acc_c(indice,1)      0 0]...
                                               ,vitesse(indice_moins,:),te_imu);
            end
            vitesse_cart(indice,:)=rot([vitesse(indice,1) 0 0],...
                [ 0 0 angle(indice,3)]);
            pos_p = integration([vitesse_cart(indice,:);...
                vitesse_cart(indice_moins,:)],pos_p,te_imu);
            pos=pos_p(1:2);
            return
        otherwise
            error('unexpexted case')
    end
end
```