# MATLAB codes

Some of the codes used to process and to analyse the signals acquired during the measurements on the treadmill, are reported. In particular:

- The codes used for the IMUs calibration
- The codes for the cycle events identification:
  - Gold standard identification (based on the poles force signals)
  - Acceleration norm-based method
  - Angular velocity norm-based method
  - Fusion method

## Main calibration

```
% main_calibration legge i dati dagli IMUs, sincronizza i sensori,
% effettua la calibrazione secondo Ferraris 1995 -->
% - 6 misure statiche in cui ogni asse del sensore è messo una volta
%   concorede ed una discorde a g. le 6 misure statiche devono essere
%   effettuate prima e dopo l'esperimento.
% - ottengo i coefficienti correttivi che tengono conto dei vari errori
% di misura (bias di acc e gyr, acc sensitivity del gyr, acc scale factor e
% oriantation matrix).

% leggo i segnali dai sensori per ciascuna posizione statica sia per le misure
% fatte prima che dopo l'esperimento.
% per identificare la posizione statica -->
% - la prima lettera mi dice quale asse è allineato con g
% - se questa lettera è seguita da un underscore ( _ ) allora l'asse punta
%   verso il basso altrimenti punta verso l'alto
% - la seconda lettera, che è seguita da 'eo' mi dice quale asse è
%   all'incirca in direzione est-ovest, info utile per la definizione
%   dell acc sensitivity del gyr.
% - esempio1: ZYeo = asse Z verso l'alto e Y in direzione est-ovest
% - esempio2: X_Yeo = asse X punta verso il basso, Y in direzione est-ovest

% dopo aver effettuato le 6 misure prima e dopo, organizzo le prime 6 in una
% cartella CalibBefore ed una cartella CalibAfter. Ognuna delle 6 misure
% deve poi contenere nel suo nome l'identificativo della posizione assunta dal
% sensore come specificato prima, questo è necessario per il riconoscimento
% automatico della cartella esatta per ogni posizione da parte del codice
% seguente. Le sei posizioni e quindi i 6 nomi identificativi sono ZYeo,
% XYeo, Z_Yeo, X_Yeo, YXeo, Y_Zeo

function [IMUsSignals_calib,calibParameters]=main_calibration(IMUsSignals)

% valore di g. Inserire il valore esatto di g del posto in cui ci si trova.
% si puo usare local gravity calculator online inserendo latitudine e altezza.
g = 9.82172;

%% data Reading

elaborazione_Path = cd('.');           % Identify the script path
cd ..                                 % It moves one level up

disp( 'select folder mem-IMUsCalib');
```

```
subjects_Path = uigetdir('C:\Users\alexa\Documents'); % mi apre un box dove ho la
cartella dei dati salvati
cd(subjects_Path) % mi muovo in quella cartella

subjectsLIST=dir('*subject*');                              % leggo i folders dei
diversi subjects in una unica cartella
for h=1:length(subjectsLIST)

    subjectname=string(subjectsLIST(h).name);
    cd(subjectname)
    before_after_Path = cd('.');

    % salvo i nomi delle misure Before e After
    list=dir('*Before*');
    before_after(1)=string(list.name);
    list=dir('*After*');
    before_after(2)=string(list.name);

    for j=1:length(before_after)
        data_Path= fullfile(before_after_Path,char(before_after(j)));
        cd (data_Path)

         % salvo i nomi di ciascuna cartella corrispondente a ciascuna delle 6
         % misure statiche
         list=dir('*ZYeo');
         PositionsFolders(1)=string(list.name); % nome Folder ZYeo
         list=dir('*XYeo');
         PositionsFolders(2)=string(list.name); % nome Folder XYeo
         list=dir('*Z_Yeo');
         PositionsFolders(3)=string(list.name); % nome Folder Z_Yeo
         list=dir('*X_Yeo');
         PositionsFolders(4)=string(list.name); % nome Folder X_Yeo
         list=dir('*YXeo');
         PositionsFolders(5)=string(list.name); % nome Folder YXeo
         list=dir('*Y_Zeo');
         PositionsFolders(6)=string(list.name); % nome Folder Y_Zeo

         %ciclo for che itera sulle 6 misure statiche
         for k=1:6

             FolderPath= fullfile(data_Path,char(PositionsFolders(k)));
             cd (FolderPath)

            %  faccio una lista, assegno un nome a ogni file
            LIST=dir('*.csv');                              % leggo i file csv in
un'unica cartella
            for i=1:length(LIST)

                sensorname(i)=string(LIST(i).name);    % assegno nome a ogni
sensore

            end
            cd(elaborazione_Path)

            %leggo i dati dai sensori con la function readsensor, per ogni
sensore
```

```
            %ottengo una matrice con [time acc_x acc_y acc_z omega_x omega_y
omega_z]
            % notare che l'ordine dei sensori in sensorsraw corrisponde a quello
in
            % sensorname --> controllare in sensorname per sapere esattamente
quale
            % sensore si sta guardando in sensorsraw.
            for i=1:length(LIST)


[sensorsraw(i).sensor]=readSensor(g,fullfile(FolderPath,char(sensorname(i))));

            end
            % sincronizzazione dei sensori
            [calib(j).measurement(k).position]= sincronizeSensors(sensorsraw);

         end

        % vettore di due elementi contenete l'ora della prima e della seconda
        % misura di calibrazione
        t(j) = sensorsraw(1).sensor(1,1);
    end

    % time paseed between two measurements in ms
    tpassed = (t(2)-t(1));

    % riorganizzo i dati in una struttura più comprensibile
    for j=1:2
        calibPositions(j).ZYeo=calib(j).measurement(1).position;
        calibPositions(j).XYeo=calib(j).measurement(2).position;
        calibPositions(j).Z_Yeo=calib(j).measurement(3).position;
        calibPositions(j).X_Yeo=calib(j).measurement(4).position;
        calibPositions(j).YXeo=calib(j).measurement(5).position;
        calibPositions(j).Y_Zeo=calib(j).measurement(6).position;
    end

    %% controllo visivo segnali

    % plot accelerazioni e vel ang di un solo sensore per la prima e la seconda
misura
    % per essere sicuri di aver eseguito le misure statiche correttamente.

    % for i=1:2
    %     figure();
    %     subplot(3,2,1)
    %     plot(calibPositions(i).ZYeo(1).sensor.acc); ylabel({'acc (g)'});ylim([-
1.5 1.5]);legend('x','y','z');title (['ZYeo misura',num2str(i)])
    %     subplot(3,2,2)
    %     plot(calibPositions(i).XYeo(1).sensor.acc); ylabel({'acc (g)'});ylim([-
1.5 1.5]);legend('x','y','z');title (['XYeo misura',num2str(i)])
    %     subplot(3,2,3)
    %     plot(calibPositions(i).Z_Yeo(1).sensor.acc); ylabel({'acc
(g)'});ylim([-1.5 1.5]);legend('x','y','z');title (['-ZYeo misura',num2str(i)])
    %     subplot(3,2,4)
    %     plot(calibPositions(i).X_Yeo(1).sensor.acc); ylabel({'acc
(g)'});ylim([-1.5 1.5]);legend('x','y','z');title (['-XYeo misura',num2str(i)])
    %     subplot(3,2,5)
```

```
    %     plot(calibPositions(i).YXeo(1).sensor.acc); ylabel({'acc (g)'});ylim([-1.5 1.5]);legend('x','y','z');title (['YXeo misura',num2str(i)])
    %     subplot(3,2,6)
    %     plot(calibPositions(i).Y_Zeo(1).sensor.acc); ylabel({'acc (g)'});ylim([-1.5 1.5]);legend('x','y','z');title (['-YZeo misura',num2str(i)])
    % end
    %
    % for i=1:2
    %     figure();
    %     subplot(3,2,1)
    %     plot(calibPositions(i).ZYeo(1).sensor.gyr); ylabel({'vel ang (dps)'});ylim([-1.5 1.5]);legend('x','y','z');title (['ZYeo misura',num2str(i)])
    %     subplot(3,2,2)
    %     plot(calibPositions(i).XYeo(1).sensor.gyr); ylabel({'vel ang (dps)'});ylim([-1.5 1.5]);legend('x','y','z');title (['XYeo misura',num2str(i)])
    %     subplot(3,2,3)
    %     plot(calibPositions(i).Z_Yeo(1).sensor.gyr); ylabel({'vel ang (dps)'});ylim([-1.5 1.5]);legend('x','y','z');title (['-ZYeo misura',num2str(i)])
    %     subplot(3,2,4)
    %     plot(calibPositions(i).X_Yeo(1).sensor.gyr); ylabel({'vel ang (dps)'});ylim([-1.5 1.5]);legend('x','y','z');title (['-XYeo misura',num2str(i)])
    %     subplot(3,2,5)
    %     plot(calibPositions(i).YXeo(1).sensor.gyr); ylabel({'vel ang (dps)'});ylim([-1.5 1.5]);legend('x','y','z');title (['YXeo misura',num2str(i)])
    %     subplot(3,2,6)
    %     plot(calibPositions(i).Y_Zeo(1).sensor.gyr); ylabel({'vel ang (dps)'});ylim([-1.5 1.5]);legend('x','y','z');title (['-YZeo misura',num2str(i)])
    % end

    %% calibration parameters
    for i=1:length(LIST)
        [calibParameters(h).subject(i).sensor]=calibration(g,tpassed,calibPositions(1).ZYeo(i).sensor,calibPositions(1).XYeo(i).sensor,calibPositions(1).Z_Yeo(i).sensor,calibPositions(1).X_Yeo(i).sensor,calibPositions(1).YXeo(i).sensor,calibPositions(1).Y_Zeo(i).sensor,calibPositions(2).ZYeo(i).sensor,calibPositions(2).XYeo(i).sensor,calibPositions(2).Z_Yeo(i).sensor,calibPositions(2).X_Yeo(i).sensor,calibPositions(2).YXeo(i).sensor,calibPositions(2).Y_Zeo(i).sensor);
    end
    IMUsSignals_calib_tmp=[];
    for i=1:length(IMUsSignals(h).subject)
        for j=1:length(IMUsSignals(h).subject(i).trial)
            % riscrivo il vettore dei tempi in relazione al tempo della prima calibrazione
            IMUsSignals_calib_tmp(i).trial(j).sensor.time=IMUsSignals(h).subject(i).trial(j).sensor.time-t(1);
            for k=1:length(IMUsSignals(h).subject(i).trial(j).sensor.acc)

                IMUsSignals_calib_tmp(i).trial(j).sensor.acc(k,:) = inv(calibParameters(h).subject(j).sensor.accOrient)*inv(calibParameters(h).subject(j).sensor.accScale)*(IMUsSignals(h).subject(i).trial(j).sensor.acc(k,:)' - diag(calibParameters(h).subject(j).sensor.accBias));
                IMUsSignals_calib_tmp(i).trial(j).sensor.gyr(k,:) = IMUsSignals(h).subject(i).trial(j).sensor.gyr(k,:)' - calibParameters(h).subject(j).sensor.gyroAccSens * IMUsSignals(h).subject(i).trial(j).sensor.acc(k,:)' -
```

```
calibParameters(h).subject(j).sensor.gyroBias0' +
calibParameters(h).subject(j).sensor.gyroBias1'
*IMUsSignals(h).subject(i).trial(j).sensor.time(k);

                %usare queste due righe sotto al posto di quelle sopra se non
                %si vuole correggere acc misalligment e gyro acc sensitivity
    %             IMUsSignals_calib_tmp(i).trial(j).sensor.acc(k,:) =
inv(calibParameters(h).subject(j).sensor.accScale)*(IMUsSignals(h).subject(i).tri
al(j).sensor.acc(k,:)' - diag(calibParameters(h).subject(j).sensor.accBias));
    %             IMUsSignals_calib_tmp(i).trial(j).sensor.gyr(k,:) =
IMUsSignals(h).subject(i).trial(j).sensor.gyr(k,:)' -
calibParameters(h).subject(j).sensor.gyroBias0' +
calibParameters(h).subject(j).sensor.gyroBias1'
*IMUsSignals(h).subject(i).trial(j).sensor.time(k);

            end
        end
    end

    % calcolo norma dell'acc e della vel ang per ogni sensore

    for j=1:length(IMUsSignals_calib_tmp)
        for k=1:length(IMUsSignals_calib_tmp(j).trial)

            for i=1:length(IMUsSignals_calib_tmp(j).trial(k).sensor.acc(:,1))

                IMUsSignals_calib_tmp(j).trial(k).sensor.accNorm(i,1) =
norm(IMUsSignals_calib_tmp(j).trial(k).sensor.acc(i,:));
                IMUsSignals_calib_tmp(j).trial(k).sensor.gyrNorm(i,1) =
norm(IMUsSignals_calib_tmp(j).trial(k).sensor.gyr(i,:));

            end

        end

    end
    IMUsSignals_calib(h).subject=IMUsSignals_calib_tmp;
    cd(subjects_Path)
end
 cd(elaborazione_Path)
end
```

## Function called in main calibration

```
% calibration---> calculation of the parameters for the correction of the
%                 accelerometer offset, sensitivity and misallaignment
%                 and of the gyro offset and acceleration sensitivity
% input-->  - g = valore locale dell'acc gravitazionale
%           - dt = tempo trascorso tra la prima e la seconda misura di
%             calibrazione ( quella fatta prima e quella dopo l'esperimento)
%           - le 6 misure statiche fatte prima dell'esperimento
%           - le 6 misure statiche fatte dopo l'esperimento
% output--> - calibParameters = struttura contenete tutti i parametri utili
%             alla correzione degli errori
```

```matlab
function [calibParameters] =
calibration(g,dt,ZYeo,XYeo,Z_Yeo,X_Yeo,YXeo,Y_Zeo,ZYeoAfter,XYeoAfter,Z_YeoAfter,
X_YeoAfter,YXeoAfter,Y_ZeoAfter)

g=1; %perchè gli sto passando gia i segnali espressi in g!! altrimenti mi viene
tutto moltiplicato di un fattore g
     % nel caso si passino i segnali non espressi in g allora togliere questa
riga

% calcolo la media nel tempo per ogni posizione e per ogni canale di acc e gyr
% ottengo dei vettori riga, ogni colonna rapp. output x,y,z.
ZYeo.acc = mean(ZYeo.acc);
XYeo.acc = mean(XYeo.acc);
Z_Yeo.acc = mean(Z_Yeo.acc);
X_Yeo.acc = mean(X_Yeo.acc);
YXeo.acc = mean(YXeo.acc);
Y_Zeo.acc = mean(Y_Zeo.acc);
ZYeo.gyr = mean(ZYeo.gyr);
XYeo.gyr = mean(XYeo.gyr);
Z_Yeo.gyr = mean(Z_Yeo.gyr);
X_Yeo.gyr = mean(X_Yeo.gyr);
YXeo.gyr = mean(YXeo.gyr);
Y_Zeo.gyr = mean(Y_Zeo.gyr);
ZYeoAfter.acc = mean(ZYeoAfter.acc);
XYeoAfter.acc = mean(XYeoAfter.acc);
Z_YeoAfter.acc = mean(Z_YeoAfter.acc);
X_YeoAfter.acc = mean(X_YeoAfter.acc);
YXeoAfter.acc = mean(YXeoAfter.acc);
Y_ZeoAfter.acc = mean(Y_ZeoAfter.acc);
ZYeoAfter.gyr = mean(ZYeoAfter.gyr);
XYeoAfter.gyr = mean(XYeoAfter.gyr);
Z_YeoAfter.gyr = mean(Z_YeoAfter.gyr);
X_YeoAfter.gyr = mean(X_YeoAfter.gyr);
YXeoAfter.gyr = mean(YXeoAfter.gyr);
Y_ZeoAfter.gyr = mean(Y_ZeoAfter.gyr);

%% acc BIAS

Ua = cat(2, XYeo.acc', YXeo.acc', ZYeo.acc');
Ua_ = cat(2, X_Yeo.acc', Y_Zeo.acc', Z_Yeo.acc');
Uas = (Ua + Ua_);
Ba = Uas./2;

calibParameters.accBias = Ba;
%% acc SCALE FACTORS and ORIENTATION MATRIX

Uad = (Ua-Ua_);
Ka = diag(sqrt(diag(Uad*Uad')./(4*(g)^2)));
Ra = (inv(Ka).*(1/(2*g))*Uad);

calibParameters.accScale = Ka;
calibParameters.accOrient = Ra;
%% gyro BIAS

Bg0 = [YXeo.gyr(1) XYeo.gyr(2) Y_Zeo.gyr(3)];
BgAfter = [YXeoAfter.gyr(1) XYeoAfter.gyr(2) Y_ZeoAfter.gyr(3)];
Bg1 = (BgAfter-Bg0)/dt;
```

```
calibParameters.gyroBias0 = Bg0;
calibParameters.gyroBias1 = Bg1;

%% gyro ACCELERATION SENSITIVITY

Ugp = [XYeo.gyr(1) YXeo.gyr(1) ZYeo.gyr(1)];
Uga = [X_Yeo.gyr(1) Y_Zeo.gyr(1) Z_Yeo.gyr(1)];
Kga = (Ugp-Uga)/(2*g);
Kga = diag(Kga);
calibParameters.gyroAccSens = Kga;
end
```

## Cycle events identification from the force signals (gold standard)

```
% identification Hits and Leaves from Coachtech

% identifica gli istanti nel segnali di forza da coachtech in cui i
% bastoncini colpiscono e lasciano il terreno all'interno di una finestra
% definita.

% input: -CoachtechSignals_sinc --> segnaloi da coachtech, sincronizzati
%        -sartSample --> campione che definisce l'inizio dell'intervallo di
%                        osservazione

% output: -Coach_pole_events --> struttura che contiene gli istanti di
%          contatto e distacco tra bastoncini e terreno per tutti i trial
%          considerati, caclolati a partire dal segnale di F di coachtech

function [Coach_pole_events] =
Hits_Leaves_Coachtech(CoachtechSignals_sinc,startSample)

    % identificazione poleLeaves
    for i=1:length(CoachtechSignals_sinc)

        [pks,locs,w,p] =
findpeaks(ribalta(CoachtechSignals_sinc(i).forceR(startSample(i):end)),'MinPeakHe
ight',0.90*max(ribalta(CoachtechSignals_sinc(i).forceR(startSample(i):end))),'Min
PeakDistance',340,'MinPeakProminence',4);
%          figure()
%
findpeaks(ribalta(CoachtechSignals_sinc(i).forceR(startSample(i):end)),'MinPeakHe
ight',0.90*max(ribalta(CoachtechSignals_sinc(i).forceR(startSample(i):end))),'Min
PeakDistance',340,'MinPeakProminence',4);
        Coach_pole_events(i).Leaves.sample = locs + startSample(i)-1;
        Coach_pole_events(i).Leaves.peak=
CoachtechSignals_sinc(i).forceR(Coach_pole_events(i).Leaves.sample);

    end

    % identificazione poleHits
    for i=1:length(CoachtechSignals_sinc)

        for j=1:length(Coach_pole_events(i).Leaves.sample)-1

                [pks,locs,w,p] =
findpeaks(CoachtechSignals_sinc(i).forceR(Coach_pole_events(i).Leaves.sample(j,1)
```

```
:Coach_pole_events(i).Leaves.sample(j+1,1)),'MinPeakHeight',0.2*max(CoachtechSign
als_sinc(i).forceR(Coach_pole_events(i).Leaves.sample(j,1):Coach_pole_events(i).L
eaves.sample(j+1,1))));
    %            figure()
    %
findpeaks(CoachtechSignals_sinc(i).forceR(Coach_pole_events(i).Leaves.sample(j,1)
:Coach_pole_events(i).Leaves.sample(j+1,1)),'MinPeakHeight',0.2*max(CoachtechSign
als_sinc(i).forceR(Coach_pole_events(i).Leaves.sample(j,1):Coach_pole_events(i).L
eaves.sample(j+1,1))));
                Coach_pole_events(i).Hits.sample(j,1)=
locs(1)+Coach_pole_events(i).Leaves.sample(j,1)-1;

        end

        Coach_pole_events(i).Hits.peak=
CoachtechSignals_sinc(i).forceR(Coach_pole_events(i).Hits.sample);
%         figure()
%         ax1=subplot(3,1,1);
%         plot(ax1,CoachtechSignals_sinc(i).forceR)
%         hold on
%
plot(ax1,Coach_pole_events(i).Hits.sample,Coach_pole_events(i).Hits.peak,'o',Coac
h_pole_events(i).Leaves.sample,Coach_pole_events(i).Leaves.peak,'square' )
%         grid on
%         ax2=subplot(3,1,2);
%         plot(ax2,polse_sensor_sinc(i).trial.accNorm)
%         hold on
%
plot(ax2,poleHits(i).trial,peaksHits(i).trial,'o',poleLeaves(i).trial,peaksLeaves
(i).trial,'square' )
%         grid on
%         ax3=subplot(3,1,3);
%         plot(ax3,CoachtechSignals_sinc(i).speed); ylabel({'Speed (km/h)'});
%         hold on
%         plot(ax3,smoothdata(CoachtechSignals_sinc(i).speed,'movmean',30));
%         grid on
%         linkaxes([ax1 ax2 ax3],'x')
    end

end
```

## Function to refine the cycle events identification from the force signals

```
% identification Hits and Leaves from Coachtech

% identifica gli istanti nel segnali di forza da coachtech in cui i
% bastoncini colpiscono e lasciano il terreno all'interno di una finestra
% definita.

% input: -CoachtechSignals_sinc --> segnaloi da coachtech, sincronizzati
%        -sartSample --> campione che definisce l'inizio dell'intervallo di
%                        osservazione

% output: -Coach_pole_events --> struttura che contiene gli istanti di
%          contatto e distacco tra bastoncini e terreno per tutti i trial
%          considerati, caclolati a partire dal segnale di F di coachtech
```

```matlab
function [Coach_pole_events_refined,CoachtechSignals_sinc_unbiased] =
Hits_Leaves_Coachtech_refine(CoachtechSignals_sinc,Coach_pole_events)

   CoachtechSignals_sinc_unbiased=CoachtechSignals_sinc;

   window1=85; %campioni di cui mi sposto da hits vecchi
   window2=10;% capioni di cui  mi sposto da hits nuovi e vecchi

   % smooth  di force signal cosi le parti in cui i bastoncini non sono a
contatto e
   % dovrebbero perciò essere zero rimangono più costanti

   for i=1:length(CoachtechSignals_sinc)


CoachtechSignals_smoothed(i).forceR=smooth(CoachtechSignals_sinc(i).forceR,12);
%         figure()
%         plot(CoachtechSignals_sinc(i).forceR)
%         hold on
%         plot(CoachtechSignals_smoothed(i).forceR)
   end

    for i=1:length(CoachtechSignals_smoothed)
        asintoto=[];
        for j=1:length(Coach_pole_events(i).Hits.sample)

            dev(i).samples(j).cycle=Coach_pole_events(i).Hits.sample(j,1)-
window1:Coach_pole_events(i).Hits.sample(j,1)-1;

dev(i).values(j).cycle=diff(CoachtechSignals_smoothed(i).forceR(Coach_pole_events
(i).Hits.sample(j,1)-window1:Coach_pole_events(i).Hits.sample(j,1)));

outliers=dev(i).samples(j).cycle(isoutlier(dev(i).values(j).cycle,'percentiles',[
0,99]));
            Coach_pole_events_refined(i).Hits.sample(j,1)=outliers(1);

            asintoto= [asintoto;
CoachtechSignals_smoothed(i).forceR(dev(i).samples(j).cycle(1):dev(i).samples(j).
cycle(end-window2))];
        end

        bias(i)=mean(asintoto);

        CoachtechSignals_sinc_unbiased(i).forceR=CoachtechSignals_sinc(i).forceR-
bias(i);

        for j=1:length(Coach_pole_events_refined(i).Hits.sample)
            tmp=
find(CoachtechSignals_sinc_unbiased(i).forceR(Coach_pole_events_refined(i).Hits.s
ample(j):Coach_pole_events(i).Leaves.sample(j)) < 0.6) ;

Coach_pole_events_refined(i).Leaves.sample(j,1)=Coach_pole_events_refined(i).Hits
.sample(j,1)+tmp(1)-1;
        end
```

```
Coach_pole_events_refined(i).Leaves.peak=CoachtechSignals_sinc_unbiased(i).forceR
(Coach_pole_events_refined(i).Leaves.sample);

Coach_pole_events_refined(i).Hits.peak=CoachtechSignals_sinc_unbiased(i).forceR(C
oach_pole_events_refined(i).Hits.sample);

    end

%      for i=1:length(CoachtechSignals_sinc_unbiased)
%
%          figure()
%          ax1=subplot(3,1,1);
%          plot(ax1,CoachtechSignals_sinc_unbiased(i).forceR)
%          hold on
%
plot(ax1,Coach_pole_events(i).Hits.sample,Coach_pole_events(i).Hits.peak,'o',Coac
h_pole_events(i).Leaves.sample,Coach_pole_events(i).Leaves.peak,'square' )
%          for j=1:length(Coach_pole_events(i).Hits.sample)-1
%              hold on
%              plot(ax1,dev(i).samples(j).cycle,dev(i).values(j).cycle,'r')
%          end
%          hold on
%
plot(ax1,Coach_pole_events_refined(i).Hits.sample,Coach_pole_events_refined(i).Hi
ts.peak,'*')
%          hold on
%
plot(ax1,Coach_pole_events_refined(i).Leaves.sample,Coach_pole_events_refined(i).
Leaves.peak,'X')
%
%          grid on
%          ax2=subplot(3,1,2);
%          plot(ax2,polse_sensor_sinc(i).trial.accNorm)
%          hold on
%
plot(ax2,poleHits(i).trial,peaksHits(i).trial,'o',poleLeaves(i).trial,peaksLeaves
(i).trial,'square' )
%          grid on
%          ax3=subplot(3,1,3);
%          plot(ax3,CoachtechSignals_sinc_unbiased(i).speed); ylabel({'Speed
(km/h)'});
%          hold on
%
plot(ax3,smoothdata(CoachtechSignals_sinc_unbiased(i).speed,'movmean',30));
%          grid on
%          linkaxes([ax1 ax2 ax3],'x')
%      end


end
```

## Cycle events identification: acceleration norm-based method

```
% identification Hits and Leaves from linear acceleration norm signal

function [pole_events] =
Hits_Leaves_IMUs_improved_new(wrist_sensor_sinc,wrist_sensor_filt,...
                         wrist_sensor_filt_high,startSample,Fs)

%% questa parte commentata usarla se si passa alla funzione la norma non filtrata
ma no se si passa la norma high e low
%     %filtraggio high
%     fcH = 40; %cutting freq
%
%     [bH,aH] = butter(2,fcH/(Fs/2),'high');
% %         freqz(bH,aH,Fs,Fs)
%
%     fc = 4; %cutting freq
%
%     [b,a] = butter(2,fc/(Fs/2));
% %     freqz(b,a,Fs,Fs)
%
%     for i=1:length(wrist_sensor_sinc)
%
%         wrist_sensor_filt(i).trial=filtfilt(b,a,wrist_sensor_sinc(i).trial);
%
wrist_sensor_filt_high(i).trial=filtfilt(bH,aH,wrist_sensor_sinc(i).trial);
% %
%         figure()
%         time=(0:length(wrist_sensor_sinc(i).trial)-1).*1/Fs;
%         plot(time,wrist_sensor_sinc(i).trial,'LineWidth',1.5,'Color','k')
%         hold on
%         plot(time,wrist_sensor_filt(i).trial,'LineWidth',1.5,'Color','r')
%         title ('accNorm raw signal VS filtered signal')
%         xlabel('time (s)')
%         ylabel('acceleration norm (g)')
%
%         figure()
%         plot(time,wrist_sensor_sinc(i).trial,'LineWidth',1.5,'Color','k')
%         hold on
%         plot(time,wrist_sensor_filt_high(i).trial,'LineWidth',1.5,'Color','r')
%         title ('accNorm raw signal VS filtered signal')
%         xlabel('time (s)')
%         ylabel('acceleration norm (g)')
%
%     end

%% voglio usare il segnale raw per identificazione lifts -- > migliori
prestazioni
% se si volesse provare a farla con il segnale filtrato basta commentare la
% riga qui sotto e andare a fare un filtraggio appropriato nella funzione
% spectral analysis
wrist_sensor_filt=wrist_sensor_sinc;
%%

    % identificazione polehits
    for i=1:length(wrist_sensor_sinc)
```

```
        [pks,locs,w,p] =
findpeaks(wrist_sensor_filt_high(i).trial(startSample(i):end),'MinPeakHeight',0.1
0*max(wrist_sensor_filt_high(i).trial(startSample(i):end)),'MinPeakDistance',340)
;
%         figure()
%         t=(0:length(wrist_sensor_filt_high(i).trial(startSample(i):end))-
1).*1/Fs;
%
findpeaks(wrist_sensor_filt_high(i).trial(startSample(i):end),t,'MinPeakHeight',0
.10*max(wrist_sensor_filt_high(i).trial(startSample(i):end)),'MinPeakDistance',34
0*1/Fs);
%         title ('find peaks in accNorm filtered signal')
%         xlabel('time (s)')
%         ylabel('acceleration norm (g)')
        pole_events(i).Hits.sample = locs + startSample(i)-1;
        pole_events(i).Hits.peak=
wrist_sensor_sinc(i).trial(pole_events(i).Hits.sample);
    end

    % identificazione poleleaves
    for i=1:length(wrist_sensor_filt)
         windows=round(diff(pole_events(i).Hits.sample)./10);

        for j=1:length(pole_events(i).Hits.sample)-1

            [pks,locs,w,p] =
findpeaks(wrist_sensor_filt(i).trial(pole_events(i).Hits.sample(j,1)+windows(j):p
ole_events(i).Hits.sample(j+1,1)-
windows(j)),'MinPeakHeight',0.90*max(wrist_sensor_filt(i).trial(pole_events(i).Hi
ts.sample(j,1)+windows(j):pole_events(i).Hits.sample(j+1,1)-
windows(j))),'MinPeakDistance',220);
%             figure()
%
t=(0:length(wrist_sensor_filt(i).trial(pole_events(i).Hits.sample(j,1)+60:pole_ev
ents(i).Hits.sample(j+1,1)-60))-1).*(1/Fs).*1000;
%
findpeaks(wrist_sensor_filt(i).trial(pole_events(i).Hits.sample(j,1)+60:pole_even
ts(i).Hits.sample(j+1,1)-
60),t,'MinPeakHeight',0.90*max(wrist_sensor_filt(i).trial(pole_events(i).Hits.sam
ple(j,1)+60:pole_events(i).Hits.sample(j+1,1)-
60)),'MinPeakDistance',220*(1/Fs)*1000);
%             title ('window between two hits (filtered signal)')
%             xlabel('time (ms)')
%             ylabel('acceleration norm (g)')
%             if length(locs) > 1
%                pole_events(i).Leaves.sample(j,1)= locs(end-
1)+pole_events(i).Hits.sample(j,1)+60-1;
%             elseif isempty(locs)==1
%                 pole_events(i).Leaves.sample(j,1)=
pole_events(i).Hits.sample(j)+round((pole_events(i).Hits.sample(j+1)-
pole_events(i).Hits.sample(j))./3);
%             else
%                 pole_events(i).Leaves.sample(j,1)=
locs(end)+pole_events(i).Hits.sample(j,1)+60-1;
%             end

             if isempty(locs)==1
```

```
                 pole_events(i).Leaves.sample(j,1)=
pole_events(i).Hits.sample(j)+round((pole_events(i).Hits.sample(j+1)-
pole_events(i).Hits.sample(j))./3);
             else
                 pole_events(i).Leaves.sample(j,1)=
locs(1)+pole_events(i).Hits.sample(j,1)+windows(j)-1;

             end


        end

        pole_events(i).Leaves.peak=
wrist_sensor_sinc(i).trial(pole_events(i).Leaves.sample);

    %     figure()
    %     ax1=subplot(2,1,1);
    %     plot(ax1,wrist_sensor_sinc(i).trial)
    %     hold on
    %
plot(ax1,pole_events(i).Hits.sample,pole_events(i).Hits.peak,'o',pole_events(i).L
eaves.sample,pole_events(i).Leaves.peak,'square' )
    %     grid on
    %     ax2=subplot(2,1,2);
    %     plot(ax2,Coachtechwrist_sensor_sinc_sinc(i).speed); ylabel({'Speed
(km/h)'});
    %     hold on
    %
plot(ax2,smoothdata(Coachtechwrist_sensor_sinc_sinc(i).speed,'movmean',30));
    %     grid on
    %     linkaxes([ax1 ax2],'x')

    end

    %% pole hits refine
    window1=5; %campioni di cui mi sposto dall'hit
    window2=10; % finestra in campioni in cui identifico la variazione della
pendenza curva

   % smooth  di signal

   for i=1:length(wrist_sensor_sinc)

       wrist_sensor_sinc_smoothed(i).trial=smooth(wrist_sensor_sinc(i).trial,5);
%         figure()
%         plot(wrist_sensor_sinc(i).trial)
%         hold on
%         plot(wrist_sensor_sinc_smoothed(i).trial)
   end

    for i=1:length(wrist_sensor_sinc_smoothed)

        for j=1:length(pole_events(i).Hits.sample)

            dev(i).samples(j).cycle=pole_events(i).Hits.sample(j,1)-
window2:pole_events(i).Hits.sample(j,1)-window1-1;
```

```
dev(i).values(j).cycle=diff(wrist_sensor_sinc_smoothed(i).trial(pole_events(i).Hits.sample(j,1)-window2:pole_events(i).Hits.sample(j,1)-window1));

outliers=dev(i).samples(j).cycle(isoutlier(dev(i).values(j).cycle,'percentiles',[0,88]));
            pole_events_refined(i).Hits.sample(j,1)=outliers(1);
        end
        pole_events_refined(i).Leaves.sample=pole_events(i).Leaves.sample;


pole_events_refined(i).Leaves.peak=wrist_sensor_sinc(i).trial(pole_events_refined(i).Leaves.sample);

pole_events_refined(i).Hits.peak=wrist_sensor_sinc(i).trial(pole_events_refined(i).Hits.sample);

    end
    pole_events= pole_events_refined;
%     for i=1:length(wrist_sensor_sinc)
%
%         figure()
%         plot(wrist_sensor_sinc(i).trial)
%         hold on
%
plot(pole_events(i).Hits.sample,pole_events(i).Hits.peak,'o',pole_events(i).Leaves.sample,pole_events(i).Leaves.peak,'square' )
%         for j=1:length(pole_events(i).Hits.sample)-1
%             hold on
%             plot(dev(i).samples(j).cycle,dev(i).values(j).cycle,'r')
%         end
%         hold on
%
plot(pole_events_refined(i).Hits.sample,pole_events_refined(i).Hits.peak,'*')
%         hold on
%
plot(pole_events_refined(i).Leaves.sample,pole_events_refined(i).Leaves.peak,'X')
%
%     end

end
```

## Cycle events identification: angular velocity norm-based method

```
% identification Hits and Leaves from angular velocity norm signal

function
[pole_events_gyr]=Hits_Leaves_IMUs_gyr00(gyrNorm,gyrNorm_filt,gyrNorm_filt_high,startSample,Fs)
%% questa parte commentata usarla se si passa alla funzione la norma non filtrata ma no se si passa la norma high e low
%filtraggio angular vel high
% fcH = 60; %cutting freq
%
% [bH,aH] = butter(2,fcH/(Fs/2),'high');
% %     freqz(bH,aH,Fs,Fs)
```

```
%
% % filtraggio angular vel
%
% fc = 20; %cutting freq
%
% [b,a] = butter(2,fc/(Fs/2));
% % freqz(b,a,Fs,Fs)
%
% for i=1:length(gyrNorm)
%     gyrNorm_filt(i).trial=filtfilt(b,a,gyrNorm(i).trial);
%     gyrNorm_filt_high(i).trial=filtfilt(bH,aH,gyrNorm(i).trial);
%
%     figure()
%     time=(0:length(gyrNorm(i).trial)-1).*(1/Fs);
%     plot(time,gyrNorm(i).trial,'LineWidth',1.5,'Color','k')
%     hold on
%     plot(time,gyrNorm_filt(i).trial,'LineWidth',1.5,'Color','r')
%     title ('gyrNorm raw signal VS filtered signal')
%     xlabel('time (s)')
%     ylabel('angular velocity norm (dps)')
%
%     figure()
%     plot(time,gyrNorm(i).trial,'LineWidth',1.5,'Color','k')
%     hold on
%     plot(time,gyrNorm_filt_high(i).trial,'LineWidth',1.5,'Color','r')
%     title ('gyrNorm raw signal VS filtered signal')
%     xlabel('time (s)')
%      ylabel('angular velocity norm (dps)')
%
% end
%% voglio usare il segnale raw per identificazione lifts -- > migliori
prestazioni
% se si volesse provare a farla con il segnale filtrato basta commentare la
% riga qui sotto e andare a fare un filtraggio appropriato nella funzione
% spectral analysis
gyrNorm_filt=gyrNorm;
%%
% identificazione polehits
for i=1:length(gyrNorm)

    [pks,locs,w,p] =
findpeaks(gyrNorm_filt_high(i).trial(startSample(i):end),'MinPeakHeight',0.10*max
(gyrNorm_filt_high(i).trial(startSample(i):end)),'MinPeakDistance',350);
%         figure()
%         t=(0:length(gyrNorm_filt_high(i).trial(startSample(i):end))-1).*(1/Fs);
%
findpeaks(gyrNorm_filt_high(i).trial(startSample(i):end),t,'MinPeakHeight',0.10*m
ax(gyrNorm_filt_high(i).trial(startSample(i):end)),'MinPeakDistance',350*(1/Fs));
%         title ('find peaks in gyrNorm filt signal')
%         xlabel('time (s)')
%         ylabel('angular vel (dps)')
    pole_events_gyr(i).Hits.sample = locs + startSample(i)-1;
    pole_events_gyr(i).Hits.peak=
gyrNorm(i).trial(pole_events_gyr(i).Hits.sample);
end

% identificazione poleLeaves
```

```
for i=1:length(gyrNorm)

    windows=round(diff(pole_events_gyr(i).Hits.sample)./10);

    for j=1:length(pole_events_gyr(i).Hits.sample)-1


        [pks,locs,w,p] =
findpeaks(gyrNorm_filt(i).trial(pole_events_gyr(i).Hits.sample(j,1)+windows(j):po
le_events_gyr(i).Hits.sample(j+1,1)-
windows(j)),'MinPeakProminence',100,'MinPeakDistance',220);
%           figure()
%
t=(0:length(gyrNorm_filt(i).trial(pole_events_gyr(i).Hits.sample(j,1)+windows(j):
pole_events_gyr(i).Hits.sample(j+1,1)-windows(j)))-1).*(1/Fs).*1000;
%
findpeaks(gyrNorm_filt(i).trial(pole_events_gyr(i).Hits.sample(j,1)+windows(j):po
le_events_gyr(i).Hits.sample(j+1,1)-
windows(j)),t,'MinPeakProminence',100*(1/Fs)*1000);
%           title ('window between two hits (filtered signal)')
%           xlabel('time (ms)')
%           ylabel('angular velocity norm (dps)')
        if isempty(locs)==1

pole_events_gyr(i).Leaves.sample(j,1)=pole_events_gyr(i).Hits.sample(j)+round((po
le_events_gyr(i).Hits.sample(j+1)-pole_events_gyr(i).Hits.sample(j)).*(2/3));
        else
         pole_events_gyr(i).Leaves.sample(j,1)=
locs(1)+pole_events_gyr(i).Hits.sample(j,1)+windows(j)-1;
        end

    end

    pole_events_gyr(i).Leaves.peak=
gyrNorm(i).trial(pole_events_gyr(i).Leaves.sample);

end



end
```

## Cycle events identification: fusion method

```
% identification Hits and Leaves from IMUs
% metodo fusion:
%   -Hits from accNorm
%   -Lifts from gyrNorm


function
[pole_events]=Hits_Leaves_IMUs_fusion(gyrNorm,accNorm,gyrNorm_filt,accNorm_filt,a
ccNorm_filt_high,startSample,Fs)
%% questa parte commentata usarla se si passa alla funzione la norma non filtrata
ma no se si passa la norma high e low
 %filtraggio high acceleration
```

```
%     fcH = 40; %cutting freq
%
%     [bH,aH] = butter(2,fcH/(Fs/2),'high');
% %         freqz(bH,aH,Fs,Fs)
%
% %filtraggio low acceleration
%     fc = 4; %cutting freq
%
%     [b,a] = butter(2,fc/(Fs/2));
% %     freqz(b,a,Fs,Fs)
%
% % filtraggio angular vel
% fcg = 20; %cutting freq
%
% [bg,ag] = butter(2,fcg/(Fs/2));
% % freqz(bg,ag,Fs,Fs)
%     for i=1:length(accNorm)
%
%         accNorm_filt(i).trial=filtfilt(b,a,accNorm(i).trial);
%         accNorm_filt_high(i).trial=filtfilt(bH,aH,accNorm(i).trial);
%         gyrNorm_filt(i).trial=filtfilt(bg,ag,gyrNorm(i).trial);
% %
% %         figure()
% %         plot(accNorm(i).trial,'LineWidth',1)
% %         hold on
% %         plot(accNorm_filt(i).trial,'LineWidth',1.5)
% %         title ('accNorm raw signal VS filtered signal')
% %         xlabel('samples')
% %         ylabel('acceleration norm (g)')
%
% %         figure()
% %         plot(accNorm(i).trial,'LineWidth',1)
% %         hold on
% %         plot(accNorm_filt_high(i).trial,'LineWidth',1.5)
% %         title ('accNorm raw signal VS filtered signal')
% %         xlabel('samples')
% %         ylabel('acceleration norm (g)')
%
%     end
%% voglio usare il segnale raw per identificazione lifts -- > migliori
prestazioni
% se si volesse provare a farla con il segnale filtrato basta commentare la
% riga qui sotto e andare a fare un filtraggio appropriato nella funzione
% spectral analysis
gyrNorm_filt=gyrNorm;
accNorm_filt=accNorm;
%%
    % identificazione polehits
    for i=1:length(accNorm)

        [pks,locs,w,p] =
findpeaks(accNorm_filt_high(i).trial(startSample(i):end),'MinPeakHeight',0.10*max
(accNorm_filt_high(i).trial(startSample(i):end)),'MinPeakDistance',340);
%         figure()
%
findpeaks(accNorm_filt_high(i).trial(startSample(i):end),'MinPeakHeight',0.10*max
(accNorm_filt_high(i).trial(startSample(i):end)),'MinPeakDistance',340);
```

```
%          title ('find peaks in accNorm filtered signal')
%          xlabel('samples')
%          ylabel('acceleration norm (g)')
        pole_events(i).Hits.sample = locs + startSample(i)-1;
        pole_events(i).Hits.peak= accNorm(i).trial(pole_events(i).Hits.sample);
    end

    %% identificazione poleleaves accNorm
    for i=1:length(accNorm_filt)
       windows=round(diff(pole_events(i).Hits.sample)./10);
        for j=1:length(pole_events(i).Hits.sample)-1

            [pks,locs,w,p] =
findpeaks(accNorm_filt(i).trial(pole_events(i).Hits.sample(j,1)+windows(j):pole_e
vents(i).Hits.sample(j+1,1)-
windows(j)),'MinPeakHeight',0.90*max(accNorm_filt(i).trial(pole_events(i).Hits.sa
mple(j,1)+windows(j):pole_events(i).Hits.sample(j+1,1)-
windows(j))),'MinPeakDistance',220);
%               figure()
%
findpeaks(accNorm_filt(i).trial(pole_events(i).Hits.sample(j,1)+windows:pole_even
ts(i).Hits.sample(j+1,1)-
windows),'MinPeakHeight',0.80*max(accNorm_filt(i).trial(pole_events(i).Hits.sampl
e(j,1)+windows:pole_events(i).Hits.sample(j+1,1)-
windows)),'MinPeakDistance',220);
%               title ('window between two hits (filtered signal)')
%               xlabel('samples')
%               ylabel('acceleration norm (g)')
%               if length(locs) > 1
%                  pole_events(i).Leaves.sample(j,1)= locs(end-
1)+pole_events(i).Hits.sample(j,1)+60-1;
%               elseif isempty(locs)==1
%                   pole_events(i).Leaves.sample(j,1)=
pole_events(i).Hits.sample(j)+round((pole_events(i).Hits.sample(j+1)-
pole_events(i).Hits.sample(j))./3);
%               else
%                   pole_events(i).Leaves.sample(j,1)=
locs(end)+pole_events(i).Hits.sample(j,1)+60-1;
%               end

             if isempty(locs)==1
                 pole_events(i).Leaves_acc.sample(j,1)=
pole_events(i).Hits.sample(j)+round((pole_events(i).Hits.sample(j+1)-
pole_events(i).Hits.sample(j))./3);
             else
                 pole_events(i).Leaves_acc.sample(j,1)=
locs(1)+pole_events(i).Hits.sample(j,1)+windows(j)-1;
             end


        end

        pole_events(i).Leaves_acc.peak=
accNorm(i).trial(pole_events(i).Leaves_acc.sample);

    %     figure()
    %     ax1=subplot(2,1,1);
```

```
    %     plot(ax1,accNorm(i).trial)
    %     hold on
    %
plot(ax1,pole_events(i).Hits.sample,pole_events(i).Hits.peak,'o',pole_events(i).L
eaves.sample,pole_events(i).Leaves.peak,'square' )
    %     grid on
    %     ax2=subplot(2,1,2);
    %     plot(ax2,CoachtechaccNorm_sinc(i).speed); ylabel({'Speed (km/h)'});
    %     hold on
    %     plot(ax2,smoothdata(CoachtechaccNorm_sinc(i).speed,'movmean',30));
    %     grid on
    %     linkaxes([ax1 ax2],'x')

    end

   %% identificazione poleLeaves gyr
for i=1:length(gyrNorm)

    window=round(diff(pole_events(i).Hits.sample)./10);

    for j=1:length(pole_events(i).Hits.sample)-1


        [pks,locs,w,p] =
findpeaks(gyrNorm_filt(i).trial(pole_events(i).Hits.sample(j,1)+window(j):pole_ev
ents(i).Hits.sample(j+1,1)-
window(j)),'MinPeakProminence',100,'MinPeakDistance',220);
%           figure()
%
findpeaks(gyrNorm_filt(i).trial(pole_events(i).Hits.sample(j,1)+window(j):pole_ev
ents(i).Hits.sample(j+1,1)-
window(j)),'MinPeakProminence',100,'MinPeakDistance',220);
%           title ('window between two leaves (raw signal)')
%           xlabel('samples')
%           ylabel('angular velocity norm (dps)')
       if isempty(locs)==1

pole_events(i).Leaves_gyr.sample(j,1)=pole_events(i).Hits.sample(j)+round((pole_e
vents(i).Hits.sample(j+1)-pole_events(i).Hits.sample(j)).*(2/3));
       else
        pole_events(i).Leaves_gyr.sample(j,1)=
locs(1)+pole_events(i).Hits.sample(j,1)+window(j)-1;
       end
       %versione fusion che calcola la media identificazione acc e gyr per
       %i lifts. Commentare se si usa l'altra versione

pole_events(i).Leaves.sample(j,1)=round((pole_events(i).Leaves_gyr.sample(j,1)+
pole_events(i).Leaves_acc.sample(j,1))/2);
 % versione fusion  che identifica i lifts come il metodo gyr.
% pole_events(i).Leaves.sample(j,1)=pole_events(i).Leaves_gyr.sample(j,1);
    end
    pole_events(i).Leaves_gyr.peak=
gyrNorm(i).trial(pole_events(i).Leaves_gyr.sample);
    pole_events(i).Leaves.peak= gyrNorm(i).trial(pole_events(i).Leaves.sample);

end
```

```
    %% pole hits refine
    window1=5; %campioni di cui mi sposto dall'hit
    window2=10; % finestra in campioni in cui identifico la variazione della
pendenza curva

   % smooth  di signal

   for i=1:length(accNorm)

       accNorm_smoothed(i).trial=smooth(accNorm(i).trial,5);
%         figure()
%         plot(accNorm(i).trial)
%         hold on
%         plot(accNorm_smoothed(i).trial)
   end

    for i=1:length(accNorm_smoothed)

        for j=1:length(pole_events(i).Hits.sample)

            dev(i).samples(j).cycle=pole_events(i).Hits.sample(j,1)-
window2:pole_events(i).Hits.sample(j,1)-window1-1;

dev(i).values(j).cycle=diff(accNorm_smoothed(i).trial(pole_events(i).Hits.sample(
j,1)-window2:pole_events(i).Hits.sample(j,1)-window1));

outliers=dev(i).samples(j).cycle(isoutlier(dev(i).values(j).cycle,'percentiles',[
0,88]));
            pole_events_refined(i).Hits.sample(j,1)=outliers(1);
        end
        pole_events_refined(i).Leaves.sample=pole_events(i).Leaves.sample;


pole_events_refined(i).Leaves.peak=accNorm(i).trial(pole_events_refined(i).Leaves
.sample);

pole_events_refined(i).Hits.peak=accNorm(i).trial(pole_events_refined(i).Hits.sam
ple);

    end
    pole_events= pole_events_refined;


end
```