Routines For Indoor Localization

EE712 Course Project

April 3, 2019

1 ⟨*boilerplate* 1⟩≡

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// -----------------------------------------------------------------
// This package defines:
//
//    fnVectorNorm  : Calculates norm of a vector
//    fnDetectStill : Detect if the sensor is in still phase by
//                    comparing with threshold value
//    fnZuptVelocity: Estimate velocity by integrating the accelerometer
//                    readings
//
// -----------------------------------------------------------------

⟨global constants 2b⟩

⟨function fnVectorNorm 2a⟩

⟨function fnDetectStill 3a⟩

⟨function fnZuptVelocity 3c⟩
```

Routines to implement an indoor localization technique which use Inertial Sensor (accelro-gyro-magneto). Adaptive Zero Velocity Update is the preferred technique to correct the error which accumulates when integrating the accelerometer output to compute velocity.

To apply adaptive zero velocity update, it is important to correctly detect the still-phase of the foot and this is done by carefully setting the threshold on the norm value of the gyro to detect zero-swing. This means that the device has to be placed where there is some swing

Further, this threshold needs updating for different walking/running speeds. The update algorithm is beyond the scope of this project. Further as our device is designed for patients with limited mobility, it will be sufficient to assume a swing on the lower end of walking speed (¡ 4kmph)

The steps to compute the velocity using ZUPT consists of the following:

1. Calculate norm (gyro)

2. Detect still-phase by comparing norm with threshold value

3. If not in still-phase, integrate accelerator output to get velocity

4. In in still-phase, velocity = 0

5. Integrate velocity to get position. This also includes figuring out the direction using the gyro and magneto.

The `fnVectorNorm` function computes the norm of a vector. For example,

$$\sqrt{\omega_{x,k}^2 + \omega_{y,k}^2 + \omega_{z,k}^2}$$

,

where $\omega_{x,k}$ indicates the angular velocity along the x coordinate at the k^{th} sampling instance.

2a ⟨*function fnVectorNorm* 2a⟩≡ (1)

```
// Computes the norm of a vector
float fnVectorNorm (float x, float y, float z) {
   float xSq = x * x;
   float ySq = y * y;
   float zSq = z * z;

   return (sqrtf (xSq+ySq+zSq));
}
```

The function `fnDetectStill` detects the still-phase by comparing the norm of the angular velocity vector against a pre-determined threshold. The value of this threshold depends on the walking/running speed of the subject.

2b ⟨*global constants* 2b⟩≡ (1) 3b▷

```
// threshold value to detect still phase (deg/sec)
float glStillPhaseThreshold = 0.01;
```

3a ⟨*function fnDetectStill* 3a⟩≡ (1)

```
// Detects still phase by comparing the gyro-norm against a threshold
int fnDetectStill (float gyroNorm) {
   if (gyroNorm < glStillPhaseThreshold) return (1);
   else return (0);

}
```

The function `fnZuptVelocity` implements the zero-update algorithm to correct the velocity obtained from the accelerometer readings by using the still-phase to zero the velocity. It returns a new velocity value.

- `ax, ay, az` are the accelerometer readings after gravity corrections, and low-pass filtering
- `gx, gy, gz` are the gyroscope readings after high-pass filtering
- `vt` is the current velocity

3b ⟨*global constants* 2b⟩+≡ (1) ◁2b

```
// Accelerator sampling duration (seconds)
float glASamplingDuration = 0.001;
```

3c ⟨*function fnZuptVelocity* 3c⟩≡ (1)

```
// Function to implement zero-update correction of the velocity values
// obtained by integrating the accelro output
float fnZuptVelocity (
      float ax, float ay, float az, float gx, float gy, float gz, float vt) {

   float gyroNormValue = fnVectorNorm (gx, gy, gz);
   float vnew = vt;
   if (fnDetectStill (gyroNormValue) == 1) return (0.0);
   else {
      vnew += glASamplingDuration * fnVectorNorm (ax, ay, az);
   }

   return (vnew);
}
```

Calculation of orientation of the body as it moves cannot be done by the gyro readings alone as over time the error due to bias accumulates. This variation in bias is due to the changing temperature of the gyroscope as it operates (this is low frequency noise).