Dynamic Initialization algorithm

Goal: The proposed algorithm intends to predict the initial User position on the map prior to navigation flow.

Sources of input information: IMU readings, Beacon readings.

Multithreading: Algorithm requires at least three threads to be implemented.

1) IMU thread
2) BLE thread
3) Dynamic initialization thread (DI thread)

Algorithm Description:

<u>Thread 1 (already implemented):</u>

1) Detect current accelerometer and angle readings.
2) Transform angle readings to an appropriate format.
3) Transform accelerometer readings + filter them with an awfWS filter.
4) Correct angles according to map orientation angle.
5) Convert accelerometer readings from local CS to global CS.
6) Filter accelerometer reading in global CS with Butterworth filter.
7) Calculate the magnitude of filtered global acceleration.
8) Run step detector and form isStationary indicator, which is an array of binaries (0 – motion, 1 – no motion).

The output of IMU thread sent to DI thread:

isStatonary = 1 or 0 – the binary indicator of motion;

angle = [pitch roll yaw] – AH angles of the device in global CS;

t = current timestamp of the reading;

<u>Thread 2 (already implemented):</u>

1) Detect the RSSIs from all available beacons.
2) Use RSSI Kalman to fill in the gaps in the RSSI map and smooth the readings;
3) Get triplets for the trilateration
4) Run trilateration
5) Check the point presence on the map. If the point is outside the map, put it on the map

The output of BLE thread sent to DI thread:

beacs = [x y] – User position according to beacon readings.

Thread 3 DI thread

INPUTS: dx – the distance between the adjacent points along the X axis

dy – the distance between the adjacent points along the Y axis

cloud – the half-length of a side of the sawed area

$V_{average}$ – average velocity of the User

$t_{Dynamic\ Init}$ – time for dynamic initialization

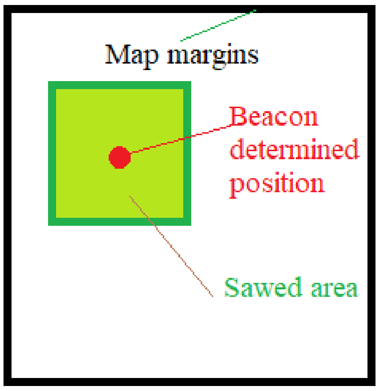
1. Initial Sawing (implemented in particle filter):

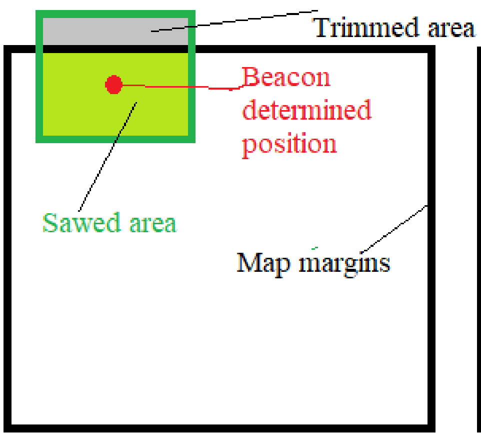
   Options:

   1) If the first beacon determined coordinates are available

      Saw the particles (use distances dx and dy) in the square surrounding the User position determined by the beacon readings (see the first sketch). If the rectangle violates the map bounds then trim it to fit map sizes (see the second sketch).

   2) If no beacon reading is available, then saw the particles covering all map





NO BEACON DETERMINED POSITION

2. Accumulating the time of a step vs Running the extinction(implemented in particle filter):

```
IF no step has been detected yet and the step is not in progress
        then do nothing.
ELSEIF the step is in progress
        then accumulate the time of a step for the following step length estimation
        $dist = V_{average} * dt;$
        $incrX = incrX + dist * \sin(angle)$
        $incrY = incrY + dist * \cos(angle)$
        Reperat step 2 of current algorithm
ELSEIF the step has been detected (isStationary(i) = 0 && dist != 0)
```

then go to the step 3.

END

3. Run model of displacement for each particle, which is (implemented in particle filter):

   $X_i = X_{i-1} + incrX_i + randn * disp$

   $Y_i = Y_{i-1} + incrY_i + randn * disp$

4. Run the extinction (implemented in particle filter):

   Extinction rules:

   1) If the particle got outside the map = > its weight turns zero (died);
   2) If the particle is in the black region => its weight turns zero (died);
   3) If the particle crossed the black region => its weight turns zero (died);

   All died particles are excluded from next iterations. No resampling is called. Number of particles gets down.

5. Calculate parameters of the circle:

   1) Calculate the centre of particles as an average of all particles.
   2) While calculating the average, find the minimum and maximum distance to the centre.
   3) Calculate the radius of the circle as

$$rad = \sqrt{\frac{partNO_i}{partNO_{initial}}} \cdot (d_{max} - d_{min}) + d_{min}$$

6. Plot the circle
7. Check the escape condition, which is

   $partNO_i < 0.05\ partNO_{init}$ or $t > t_{Dynamic\ Init}$

   If the condition is met, then start clustering the remained particles on the map starting from the first point.

   If the distance between current and previous particles is below the limit, then average these particles. If the distance is greater than ignore this point and go to the next up till the end of the array of particles.

   Then, switch to the next point and repeat the described sequence.

   Do this until the number of particles in the array stops changing (in the ideal case, all points will form one final position, in the real cases – some positions (less than 8). A number of final positions depend on the map and obstacles on the map.

8. Select the only final point from the array of final positions according to an actual position based on beacon readings at current time moment