Algorithm for GPS Navigation, Adapted for Visually Impaired People

Rosen S. Ivanov

Abstract — The paper presents an algorithm for speech enabled GPS navigation in Bulgarian. The algorithm is part of the Java mobile application for GPS tracking and navigation. Application is adapted for people with visual disabilities. The proposed algorithm allows: navigation through a trace of a route in the presence and in the absence of an electronic compass, embedded in mobile terminals; reducing errors in GPS data using Kalman filtering; adapting navigation to the current accuracy of GPS receiver; dynamic change of the course; SOS mode - the opportunity to send information about the current user location via SMS and voice message by MMS. User is informed by Text To Speech (TTS) module for: a need to change the course; distance and time to reach the end of the route; reaching the target point and a change of status of the GPS receiver.

Index Terms — GPS outdoor navigation, J2ME applications for blind navigation

1 Introduction

The number of people with visual disabilities is around 180 million, of which 45 million are totally blind [1]. Navigation of people with visual impairment in urban and suburban environment, necessary for their normal way of life, are very serious problem and create social and professional difficulties. These limitations are partially overcome by the use of dogs, white canes and by adapting the environment.

Current level of mobile communications, satellite navigation systems and functional characteristics of mobile terminals allow the creation of mass available applications for GPS outdoor navigation, which can be used by people with visual disabilities. Major problem in such systems is the insufficient accuracy of GPS receivers, mainly due to noise in GPS data. In the absence of support for the differential GPS, the noise can be reduced through the use of information from inertial sensors and filtering techniques [2], [3].

One of the most commercial mobile applications for the past years is based on customer location - Location Based Services (LBS) [4], [5]. The majority of applications for mobile GPS navigation, which can be used by people with visual disabilities, are developed in C++ for operating systems such as Symbian, Windows Mobile and Linux. This implies the use of mobile terminal such as smartphones, PDA or Pocket PC, which still have a high price. There are existing navigation applications for visually impaired people, such: Drishti [6], Wayfinder Access [7], Brunel navigation system for blind [8], Street Talk [9], Mobile Geo [10], and etc., but a mobile application is not present in Bulgarian yet.

Mass availability of such application can be guaranteed when using Java. J2ME is platform independent technology, that allows the application to be installed on any mobile terminal with built-in JVM, which supports the necessary Java API.

Considering the trend for hardware interpretation of the Java bytecode, such as technology Jazelle Direct Bytecode eXecution (DBX) [11], it is not a problem the creation of applications requiring high JVM performance.

2 Algorithm Design

It is proposed to use an external GPS receiver with Bluetooth™ interface. Such a decision has the following advantages: still small number of mobile terminals of the medium price segment have integrated GPS receiver; the user has the option to choose a GPS receiver, taking into account parameters such as price, sensitivity and accuracy.

Data necessary for the operation of the navigation algorithm are: GPS status, longitude, latitude, speed and hdop. The sequence of their obtaining is shown in figure 1.

For communication with the GPS receiver class GPSProvider is used. It implements search and communication with any GPS receiver with Bluetooth™ interface. Search and connect to GPS receiver are realized without user interaction.

▪ *R.S. Ivanov is with the Department of Computer Systems and Technologies, Technical University of Gabrovo, BULGARIA. E-mail: rs-soft@ieee.org.*

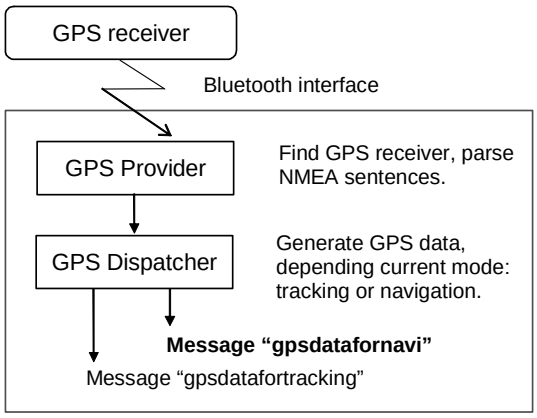

Fig. 1. Obtaining the necessary GPS data

Parsing GPRMC, GPGGA, GPGSV and GPGSA NMEA-0183 sentences, following GPS data are obtained: status, longitude, latitude, altitude, speed, direction, hdop and vdop. Access to the GPS information is possible through the interface GPSListener. Class GPSDispatcher, which implements interface GPSListener, filters GPS position (adaptive Kalman filter) and speed (1st order IIR filter) and notify Tracking and Navigation modules for new data availability. Communication between classes is realized through the mailbox. GPSDispatcher class generates message "gpsdatafornavi", when there are new data for Navigator module. This message is generated in the 1.5 to 10 seconds, depending on the trend of the filtered speed. Each program module, that should receive messages, must define method newMessage.

3 ALGORITHM DESCRIPTION

Whenever Navigator module receives "gspdatafornavi" message, method NavigationHandler is called (see figure 2).

```
1.  Algorithm NavigationHandler()
2.  if (loadRouteFlag)
3.     if (navigate)
4.        if (navigationStatusFlag) Navigate(lon,lat,hdop)
5.        else StartNavigation(lon,lat)
6.        endif
7.     endif
8.  endif
```

Fig. 2. Algorithm NavigationHandler

Its task is to call the method Navigate or StartNavigation, depending on whether the navigation is started or not. The methods are called only if the track is loaded (flag loadRouteFlag is true) and the user has enable navigation (navigate flag is true).

3.1 Start Navigation

Navigation mode is started if the user current position is less than a preset distance from the route. The method StartNavigation uses the following variables and constants:

direction - defines the direction of the route: (1) – go to the last route waypoint, (-1) – go to the first waypoint;
lastPoint - last waypoint number;
lastDistance – last traversed distance in meters;
lastAlpha – last heading error in degrees;
MAX_DIST - initial value of lastDistance;
MAX_ALPHA - initial value of lastAlpha.
Navigation can be started if the user approaches a track *MINDIST_TO_TRACK* meters.

Figure 3 shows the coordinate system that was used for computation of heading error. In this coordinate system North is 0° and positive angles are measured clockwise.

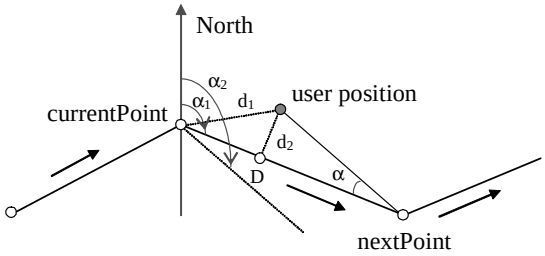

Fig. 3. User Position

To verify the condition for the start of navigation, nearest track waypoint (currentPoint) is searched.

The distance to this waypoint (d_1 on figure 3 and minDist on figure 4) and the nearest distance to track (d_2 on figure 3 and d on figure 4) if available are calculated. If $\min(d_1,d_2) <$ MIN_DIST_TO_TRACK (line 13) assumes that navigation can start (startNavigationFlag is set true). Otherwise the user is informed that it is too far from the track (line 32).

The initial values of variables lastDistance and lastAlpha are set (lines 22-23) and next waypoint of the track (nextPoint), depending on the chosen direction, is obtained (lines 19-20).

If the mobile terminal has a built-in electronic compass (line 24), the direction that the user must follow to reach nextPoint is obtained through getAzimuth method (line 25). The method NavigateToAzimuth notifies the user to turn round until hear "stop", as shown on figure 5.

```
1.    Algorithm StartNavigation(lon,lat)
2.    index = 0, pos = 0
3.    if (direction = 1) lastPoint=numberOfPoints-1
4.    else lastpoint=0
5.    endif
6.    [index,minDist] = FindNearestTrackPoint(
                                    lon,lat,pathLon,pathLat)
7.    d = FindNearestDistToTrackSegment(index-1)
8.    if (d < minDist) minDist=d, pos=-1
9.    endif
10.   d = FindNearestDistToTrackSegment(index+1)
11.   if (d < minDist) minDist=d, pos=1
12.   endif
13.   if (minDist < MIN_DIST_TO_TRACK)
14.      if (index = lastPoint)
15.         TTS.say("Last waypoint is reached")
16.         stopNavigation()
17.      else
18.         startNavigationFlag = true
19.         if (pos=-direction) nextPointIndex=index
20.         else nextPointIndex=index+direction
21.         endif
22.         lastDistance = MAX_DIST
23.         lastAlpha = MAX_ALPHA
24.         if (compassFlag)
25.            azimuth = Compass.getAzimuth()
26.            navigateToAzimuth(azimuth)
27.         else
28.            TTS.say("Compass missing")
29.         endif
30.      endif
31.   else
32.      TTS.say("Too far from the track"+minDist)
33.   endif
```

Fig. 4. Algorithm StartNavigation

```
1.    Algorithm NavigateToAzimuth(destAzimuth)
2.    currAzimuth = -1
3.    TTS.say("Turn round until you hear stop")
4.    while(currAzimut != ±5%(destAzimuth))
5.       currAzimuth = Compass.getAzimuth()
6.       wait(2sec.)
7.    endwhile
8.    TTS.say("Follow this direction")
```

Fig. 5. Algorithm NavigateToAzimut

Depending on the distance d_2 user navigates to next waypoint or point D. In the absence of electronic compass the user is necessary to walk certain distance to determine the direction. The value of the direction, obtained from GPS receiver can not be used, because if the speed is less than 10km/h the error is too large.

3.2 Navigation

After setting the flag navigationStartedFlag method NavigationHandler starts to call method Navigate (see figure 6). It is used to perform the waypoint following task and to inform the user with voice for necessary direction adjustment. The heading error is the difference between the heading to the goal waypoint (angle α_1) and the user's current heading (angle α_2), $\alpha=\alpha_2-\alpha_1$. If $\alpha>0$, the correction of the course should be $|\alpha|$ degrees in right, and if $\alpha<0$ - $|\alpha|$ degrees in left. If $\alpha\rightarrow 0$ is assumed that the user follows the correct direction.

```
1.    Algorithm Navigate(lon,lat,hdop)
2.    if (hdop > 5.0) return
3.    endif
4.    MIN_DIST_POINT_TO_POINT = 10*hdop+15
5.    elapsedDistance = GPS.distance(lon,lat,
         pathLon[nextPointIndex],pathLat[nextPointIndex]
6.    if
      (elapsedDistance>MIN_DIST_POINT_TO_POINT)
7.       ALPHA_MIN = 15
8.    else
9.       ALPHA_MIN = 35
10.   endif
11.   if (elapsedDistance<MIN_DIST_TO_POINT)
12.      if (nextPointIndex=lastPoint)
13.         TTS.say("Last point reached")
14.         stopNavigation()
15.         return
16.      else
17.         currPointIndex=nextPointIndex
18.         nextPointIndex=findNextPoint()
19.         elapsedDist=GPS.dist(lon,lat,
          pathLon[nextPointIndex],pathLat[nextPointIndex]
20.         alpha=GPS.bearingToNextPoint(currPoint,
                                          nextPointIndex)
21.         if (alpha > 0)
22.            TTS.say("Turn right"+abs(alpha)+"degree")
23.         elseif (alpha < 0)
24.            TTS.say("Turn left"+abs(alpha)+"degree")
25.         else
26.            TTS.say("Go ahead")
27.      endif
28.   else
29.      alpha=GPS.bearingToNextPoint([lon,lat],
                                       nextPointIndex)
30.      if (alpha < ALPHA_MIN) alpha = 0
31.      endif
32.      if (elapsedDistance > (lastDistance +
                           MIN_DIST_POINT_TO_POINT))
33.         if (abs(alpha - lastAlpha) < 15)
34.            TTS.say("Go back")
35.         else
36.            TTS.say("Stray from the route")
37.         endif
38.      else
39.         if (alpha > 90)
40.            if (nextPointIndex = lastPoint)
41.               TTS.say("Last point reached")
42.               stopNavigation()
43.               return
44.            else
45.               TTS.say("You pass point"+nextPointIndex)
46.               nextPointIndex=nextPointIndex+direction
47.            endif
48.         else
49.            if (alpha > 0)
50.               TTS("Turn in right"+abs(alpha),
                     "degree, remain",
                     elapsedDistance+"meters")
51.            elseif (alpha < 0)
52.               TTS("Turn in left"+abs(alpha),
                        "degree, remain",
                        elapsedDistance+"meters")
53.            else
54.                  TTS("Go ahead",
      elapsedDistance+"meters")
55.            endif
56.         endif
57.      endif
58.   endif
59.   lastDistance = elapsedDistance
60.   lastAlpha = abs(alpha)
```

Fig. 6. Algorithm Navigate

Navigate method returns when current GPS accuracy is too low (line 2). The user is informed of this situation by TTS module.

The value of parameter MIN_DIST_POINT_TO_POINT is obtained adaptively, depending on the value of hdop (line 4). The parameter is used to set minimum value of α - ALPHA_MIN (lines 6-10), and by method findNextPoint.

If the distance to next waypoint (nextPoint) is less than MIN_DIST_TO_POINT is assumed that waypoint is reached (line 11). In this case algorithm checks (line 12) if next waypoint is last waypoint (lines 13-15) or not (lines 17-26). If this waypoint is not last waypoint next waypoint is obtained (line 18) and heading error is calculated (line 20).

If the next waypoint is not reached (lines 29-57) direction to follow to reach nextPoint is calculated (line 29). The algorithm informs user if it is moving in the opposite direction (line 34) or if the heading error is too big (line 36). The user is informed if next waypoint if passed (lines 45-46), otherwise voice navigation is realized (lines 49-55).

When last waypoint is reached (lines 13 and 41) navigation is stopped by calling method StopNavigation (see figure 7).

```
1.  Algorithm StopNavigation()
2.  navigate = false
3.  navigationStatusFlag = false
```

Fig. 7. Algorithm StopNavigation

4 EXPERIMENTAL RESULTS

The proposed algorithm is part of J2ME application for GPS outdoor navigation, adapted for people with visual disabilities.

Application can be installed on any mobile terminal with JVM and: profile MIDP 2.0, configuration CLDC 1.1, Bluetooth API (JSR-82), Mobile Media API (JSR-135), Wireless Messaging API (JSR-120) and File Connection API (JSR-75). The package Wireless Messaging API ver.2.0 (JSR-205), which is used to send MMS messages, is optional.

Since the application makes access to protected resources (Bluetooth interface, file system, and etc.) it is needed to be signed. This prevents the need for confirmation that the user is agrees with access to any protectted resource.

The user can access the most important information in Navigation mode with the keys of the mobile terminal (see Table 1).

TABLE 1
KEYS USED IN NAVIGATION MODE

Key	Description
0	GPS data and status
1	Send SOS (SMS or MMS)
3	Distance to next waypoint
5	Last navigational information
7	Change the direction
9	Distance and time to reach target waypoint
*	Start/Stop TTS
#	Record audio landmark

The results, obtained when testing the application in navigation mode, are shown in figure 8 (mobile terminal Nokia N95 is used).

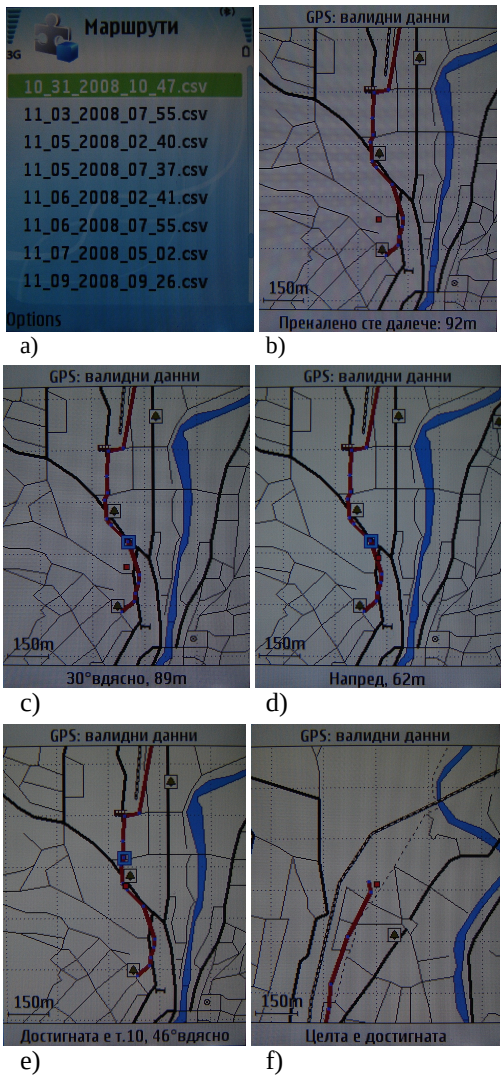

Fig. 8. Experimental results: a) select a track from list; b) navigation can not be started, because the user is so far from the track (92m); c) direction correction - 30° in right; d) go ahead (62m); e) waypoint 10 has reached – turn 46° in right; f) last waypoint is reached.

5 CONCLUSION

In the paper a speech enabled GPS navigation algorithm in Bulgarian is presented. The algorithm is part of the Java mobile application for GPS outdoor navigation, adapted for people with visual disabilities.

The proposed algorithm has the following advantages:

1. Ability to work without GPS maps. In Tracking mode visually impaired user walks through a route with additional person. During this walk, GPS information for track waypoints a stored on flash disk of mobile terminal.

2. Navigation is adaptive to the current accuracy of GPS receiver.

3. Changing the direction can be realized at any time.

4. User can pass track's waypoints.

5. User is informed for the necessary adjustments of the direction of movement in degrees.

6. User can send SOS messages in the form of SMS (username, GPS status, longitude, latitude, altitude, date and time) or MMS (the information from the SMS and voice message).

7. User can record voice landmarks.

8. Information from an electronic compass, if it is available, can be used.

REFERENCES

[1] http://laico.org/v2020resource/files/vision2020_jul-sep01.pdf. 2008.

[2] M.H. Bruch, et al., *Accurate Waypoint Navigation Using Non-differential GPS*, AUVSI Unmanned Systems, 2002.

[3] M.H. Grewal, L.R. Weill, A.P. Andrews, *Global Possitioning Systems, Inertial Navigation, and Integration*, John Wiley & Sons, NY, 2001.

[4] http://www.mobilein.com/location_based_services.htm. 2009.

[5] http://www.forum.nokia.com/Resources_and_Information/Explore/Mobile_Technologies/Location-Based_Services/. 2008.

[6] L. Ran, S. Helal, S. Moore, "Drishti: an Integrated Indoor/Outdoor Blind Navigation System and Service", *Proc. of the 2nd IEEE Annual Conference Pervasive Computing and Communications*, pp.23-30, 2004.

[7] http://www.mywayfinder.com/manual/access/en/main.html. 2008.

[8] http://dea.brunel.ac.uk. 2009.

[9] http://www.freedomscientific.com/products/fs/streettalk-gps-product-page.asp. 2009.

[10] http://www.codefactory.es/en/. 2008.

[11] http://www.arm.com/products/multimedia/java/jazelle.html. 2008.

Rosen S. Ivanov received the Electronics Engineering degree from the University of Gabrovo, Bulgaria. He received the Ph.D. degree from the University of Sofia, Bulgaria, in 2000. He is currently Associate Professor of Computer Systems and Technologies at University of Gabrovo, Bulgaria. He is the author of five books and over than 40 technical papers. His research interest include mobile communications and digital signal processing.