Navigation Aid for Mobility Assistants

Bernd Krieg-Brückner*, Bernd Gersdorf*, Christian Mandel*, Christoph Budelmann†, Marcus-Sebastian Schröder‡

*German Research Center for Artificial Intelligence (DFKI), Bremen, Germany
†Budelmann Elektronik, Münster, Germany
mail@budelmann-elektronik.com
‡neusta mobile solutions, Bremen, Germany
m.schroeder@neusta.de

Abstract—This paper introduces Navigation Aid as the first development of the European project Assistants for Safe Mobility (ASSAM). By fusing odometry, GPS and map information with the help of a Monte Carlo particle filter, we provide precise outdoor localization for assistive devices such as a walker. This allows for the application of a navigation system that considers device-specific environments including sidewalks, cycleways, etc. Target group needs are addressed by a simple and intuitive user interface running on Android-powered devices.

I. INTRODUCTION

The ASSAM project (cf. [2], [10], [11]) includes 10 partners in Germany, Spain, and The Netherlands from research, industry, and end-user organizations; the latter ensure user-centered development including every-day usability assessment cycles in field trials. ASSAM aims to compensate for declining physical and cognitive capabilities of elderly persons by the development of modular navigation assistants for various mobility platforms, such as walker, wheelchair, and tricycle. The assistance systems shall provide sustained everyday mobility and autonomy with seamless transition from indoors to outdoors in environments such as residential complexes or the neighborhood quarter:

- Physical assistance for declining walking capabilities;
- Safety assistance by obstacle avoidance;
- Cognitive assistance for declining visual and mental capabilities by navigational aid; and
- Security assistance by a care center connection in case of emergency situations.

Based on more than 15 years of experience in building smart assistive systems for the electric wheelchair Rolland (cf. [12], [13], [16]), the iWalker demonstrator was previously developed in the EU-project SHARE-it [6]. In ASSAM we envisage several variants in configuring hardware platforms, additional devices and navigational software.

In this paper we concentrate on cognitive assistance, primarily for navigation, and security assistance. For these, only minimal hardware extension is required for existing (non-electric) vehicles, and market introduction is within reach. Indeed, apart from a state-of-the-art smart phone or tablet PC with GPS, only two OdoWheel (patent pending) smart sensor devices (cf. [3]) are required, attached to two front or back wheels. Navigation Aid provides basic outdoor navigation abilities, like a car navigation system, but tailored to the mobility platforms.

The remainder of this paper is structured as follows. Sec. II reviews existing state-of-the-art (sub-)systems that contribute to navigational aid for the target mobility platforms. Section III introduces the core components of Navigation Aid that is presented throughout this work. Its constituents are detailed in Sec. IV (OpenStreetMap as environmental representation), Sec. V (the self-localization approach used), Sec. VI (planning of suitable paths and navigational aids), and Sec. VII (user interface). After a presentation of first experimental results in Sec. VIII, we conclude in Sec. IX with an assessment of the results achieved, and an outlook on future work to do.

Fig. 1. The Navigation Aid user interface runs on an Android Nexus-7 tablet, and is mounted on a commercially available TOPRO TROYA walker.
II. RELATED WORK

Accessibility [23], and routing information [19] for mobility platforms such as walkers or wheelchairs are generally available on dedicated internet-community platforms. Building on already available OpenStreetMap data (cf. Sec. IV), users of such systems may annotate obstacles, ramps, fences, bollards, and many other items that constitute barriers or facilitate entry to houses, shops, parks, etc. Systems that provide path-planning features rely on these annotated datasets.

Menkens et al. extend the idea of web-based navigation support with their system EasyWheel [18]. The authors’ key idea is to involve an even larger fraction of the target group of wheelchair users via social communities such as Facebook, and integrate mobile devices such as smart phones. Kasemsuppakorn and Karimi present a wheelchair routing system that calculates impedance scores for possible paths [9], additionally considering the slope and the surface condition of a path. Sumida et al. further pursue this idea by including force measurements into the routing graph [24]: the cost of a potential path corresponds to the accumulation of muscle or motor forces (effort) needed to reach the desired goal.

III. SYSTEM OVERVIEW: NAVIGATION AID

Navigation Aid is designed as a general support system that may easily be adapted to different vehicles like walkers, wheelchairs, or tricycles. In this paper we describe a prototypical implementation integrated into the commercially available walker platform TOPRO TROYA (cf. Fig. 1). Its core components are briefly overviewed in this section.

A. Additional Hardware Component: OdoWheel

Each OdoWheel unit as in Fig. 3 is equipped with a three-axis acceleration sensor and a gyrometer. An embedded 32bit microcontroller reads out the sensors, performs sensor fusion and sends the processed data to the smart phone or tablet PC. For the radio link between the handheld device and OdoWheel a standard Bluetooth Low Energy link has been established making additional radio dongles for the handheld device superfluous. A solar energy harvester with maximum power point tracking is used as power supply. As the availability of the solar energy source is varying, a lithium polymer backup battery has been implemented as a storage element allowing permanent operation even during the night. This makes the OdoWheel unit completely maintenance free without the need to recharge or exchange any batteries. Only the solar cell, covered by a protective glass, is visible from the outside; the electronic components are mounted in the wheels behind the solar cell. This allows the user to adapt an existing walker by simply exchanging its back wheels.

B. User Interface Device: Google Nexus 7

For development of the user interface, Google Nexus 7 [7] has been chosen as the present hardware platform. The main reason is its status as a tablet directly marketed by Google, which means that future updates to the Android OS will be available for it shortly after announcement for the foreseeable future, allowing further development to immediately benefit from new features. Moreover, its quad-core CPU, 1 GB RAM and screen quality are rarely found in the €200 price class.

C. X86 Controller for Planning & Self-Localization

As an intermediate solution for hosting the path planning and the outdoor localization algorithms, we equipped the walker with a Linux-powered X86 computer. Although we still see issues in the limited computing power and energy storage of Nexus 7, upcoming revisions of Navigation Aid shall definitely host this functionality on the user interface device itself. At this stage the controller communicates with the user interface via a Google Protocol Buffer implementation through WLAN.

IV. OSM AS ENVIRONMENT REPRESENTATION

With the launch of the collaborative project Open Street Map (OSM) (cf. [20], [21]), a continuously growing set of road network data in digital form has become freely available. Based on user-recorded GPS track logs, and donated commercial datasets (e.g. the complete road dataset for the Netherlands), OSM provides maps of arbitrarily selectable regions in XML-format. These data not only contain a vector-based representation of the road network, but also a detailed classification of road segment types, path properties such as the accessibility of sloped curbs or ramps, and obstructions.

An important operation on the road network dataset inside the sensor model of the proposed particle filter (cf. Sec. V-C) is the computation of the closest road segment for a given position hypothesis. Assuming $M = 10^5$ road segments in an exemplary scenario, and $N = 10^4$ position hypotheses, a naive implementation of the road network data structure implies $O(M \cdot N) = 10^9$ line segment to point distance queries within a single cycle of computation of the particle filter.

In order to reduce this computational payload, we sort each line segment representing a road segment from the OSM dataset into a space partitioning data structure, the so-called PMR quadtree [8]. Like a regular quadtree, this spatial data...
structure inserts its elements, in our case line segments, into buckets. Initially, the PMR quadtree consists of its root bucket, representing the whole Euclidean plane. By using a splitting rule that defines the maximal number $\delta$ of line segments to be contained within a single bucket, the line segments are inserted one-by-one into the data structure. After reaching the threshold $\delta$ for a given bucket $B$, four child-buckets are appended to $B$, representing four equal-sized quadrants of the plane represented by $B$. At this point, all line segments that have been contained by $B$ are propagated down to the new child-buckets. Note that this approach leads to redundant storage of elements, since a line segment is stored in every bucket that it intersects.

In [8] Hoel et al. present an algorithm with computational complexity $O(\delta)$ that finds the closest line segment from a given PMR quadtree w.r.t. a given point, provided the line segments are uniformly distributed over the given map region. This yields an upper bound of $O(\delta \cdot N)$ queries for closest road segment to position hypotheses, making the PMR quadtree the first choice for the spatial data structure of the road network in our particle filter.

V. OUTDOOR LOCALIZATION

The self-localization approach used throughout this work is based on a Monte Carlo outdoor locator, formerly implemented at DFKI [15]. The inherent problem to solve is to deal with disturbances in GPS signal-reception and multi-path effects. We do so by applying a particle filter, each sample of which represents a 3D pose-hypothesis of the system. During the filter’s motion update step, each sample is translated by the movement vector recently measured by the OdoWheel’s inertial measurement units. In the following sensor update step, the plausibility of each sample is assessed by its metrical distance to the actual GPS-measurement, and several plausibility metrics that evaluate OSM information. One of the latter judges whether the type of the closest OSM street segment matches the actual mobility platform; a walker, for example, is not expected to be used on primary roads. As in general particle filters, the overall estimated pose is computed as the weighted sum over all samples, before the concluding filter step resamples the hypotheses for the next cycle of execution.

A. State Estimation

The theoretical background for the state estimation realized is given by a Monte-Carlo localization approach [5], since it is able to deal with non-linear motion models (cf. Sec. V-B) as well as with the occurrence of multiple hypotheses. The vehicle’s state $X_t$ to be estimated, and thus the content of every particle $x_t^{[m]}$ within a set of $M$ particles, is a simple pose in 2D:

$$x_t^{[m]} := \langle px_t^{[m]}, py_t^{[m]}, \theta_t^{[m]} \rangle$$

with $px_t^{[m]}$ and $py_t^{[m]}$ being the x- and y-coordinate within a global coordinate system, and $\theta_t^{[m]}$ being the orientation of the sample against the axis of abscissas.

B. Motion Model

The state transition of a single particle through motion is given by

$$v_t^{[m]} := \langle d_t, h_t \rangle$$

with $d_t$ being the translational distance traveled since the last motion update, and $h_t$ being the rotational distance respectively. A three-dimensional offset $v_t^{[m]} = \langle v_x^{[m]}, v_y^{[m]}, v_\theta^{[m]} \rangle$, thereby an updated sample position, is computed as follows:

$$v_t^{[m]} = \begin{pmatrix} M_\theta^{[m]} \left( d_t + a_1 n_x^{[m]}(d_t) \right) \\ a_2 n_y^{[m]}(d_t) \\ a_3 n_\theta^{[m]}(h_t) \end{pmatrix}$$

$$px_t^{[m]} = px_{t-1}^{[m]} + v_x^{[m]}$$

$$py_t^{[m]} = py_{t-1}^{[m]} + v_y^{[m]}$$

$$\theta_t^{[m]} = \theta_{t-1}^{[m]} + v_\theta^{[m]}$$

with $M_\theta^{[m]}$ being a rotation matrix describing $\theta^{[m]}$, $n_x^{[m]}, n_y^{[m]}, n_\theta^{[m]}$ noise function sampled from a triangular distribution, and $a_1$, $a_2$, and $a_3$ being three different scalars for the generated noise.
\begin{table}[h]
\centering
\caption{Exemplary penalty values for localization hypothesis of a walker. Besides other factors, LocPenalty influences the weighting of a sample within the sensor model of the localizer (cf. Sec. V-C). The factor $\Delta$ is used within the Dijkstra path-planning framework to favour certain classes (cf. Sec. VI) during wayfinding. Classes are taken from [20].}
\begin{tabular}{|l|l|l|l|}
\hline
Class & Description & LocPenalty $[0..1]$ & $\Delta [0..10]$ \\
\hline
cycleway & designated cycleway & 0.2 & 5 \\
footpath & designated footpath & 0.5 & 5 \\
pedestrian & e.g. in shopping areas & 0 & 0 \\
path & public footpath & 0.2 & 2.5 \\
track & roads for agricultural use & 0.2 & 5 \\
living street & traffic-calmed street & 0.7 & 5 \\
service street & access roads within industrial estate, etc. & 0.5 & 5 \\
steps & flight of steps on footways & 0.99 & 0 \\
primary road & generally linking larger towns & 0.99 & 10 \\
motorway & equivalent to Autobahn, freeway, etc. & 0.999 & 0 \\
\hline
\end{tabular}
\end{table}

\section{C. Sensor Model}
To carry out the particle filter’s resampling step (which is assumed to be known, and therefore it is not described in this paper), each sample’s weighting $w_i$ needs to be computed according to the sensor model

$$z_t := \langle GPS_t, dist_t, f(GPS_t) \rangle$$

with $GPS_t$ being the position measurement acquired from a connected GPS device. The distance $dist_t$ is a virtual measurement to keep the samples along the OSM path segments. As this is an assumption about the driver’s behavior, and not a real sensor measurement, $dist_t$ is always zero. By this construct, a sample’s distance $dist_t^{[m]}$ (which is based on state variables) to the next path segment (also considering its width) can easily be incorporated into the sensor model. The value of the function $f(GPS)$ is also a virtual measurement, and thus assumed to be zero. It is used to describe even more assumptions about the behaviour of the system, i.e. a modeled preference of types of paths to be used, or the implausible situation in which a sample is located within buildings (cf. Table I for exemplary values resulting from a sample’s position). Finally, $w_i^{[m]}$ computes as follows:

$$\delta_{GPS}^{[m]} = \left| GPS_t - \left( px_t^{[m]}, py_t^{[m]} \right) \right|^T$$

$$u_i^{[m]} = \mathcal{N} \left( 0, \sigma_{GPS}^2 \right) \left( \delta_{GPS}^{[m]} \right) \cdot \mathcal{N} \left( 0, \sigma_{dist}^2 \right) \left( dist_t^{[m]} \right) \cdot \mathcal{N} \left( 0, \sigma_{f}^2 \right) \left( f_t^{[m]}(GPS_t) \right)$$

\section{D. Sensor Resetting}
Regarding self-localization, the so-called robot kidnapping is a common problem caused by transformations not covered by the robot’s motion model, e.g. by carrying the robot to a different place during normal operation. Such classical kidnapping operations do not occur in our scenario but something causing a similar effect might happen: the robot might lose its way, e.g. by letting many samples take a wrong branch, or the distribution might fall behind the real position or be ahead of it respectively.

\section{VI. Planning of Paths and Navigational Advices}
By adding new samples computed from recent measurements to the probability distributions, the so-called sensor resetting [14], the Monte-Carlo approach is able to recover quickly from kidnapping actions. Due to the lack of unique features in our scenario, these samples are randomly added on streets within a certain frame. The frame is an axis-aligned bounding box around all samples that can be scaled by a user-defined factor. In our implementation, a fixed number of samples is added after each resampling step.

\section{VII. User Interface}
The user interface design becomes a crucial component, especially when developing assistive devices for the elderly or disabled. For Navigation Aid we build on a simple and clearly designed interface running on Android-powered devices such as smart phones or tablets. It utilizes a custom hybrid of the vector-based rendering engines mapsforge [17] and osmdroid [22] for map display, leveraging the offline functionality of mapsforge to minimise the reliance on data from the internet.
Fig. 5. The Navigation Aid’s user interface in landscape mode running on a Google Nexus 7. It comprises a mapnik- and osmdroid-based rendering of the map content, the actual position of the user (yellow figure), the start-position of the path (red marker), the goal position of the path (green marker), the computed path in-between (red line), and two kinds of navigational instructions (green compass-like arrows to the left; cf. Sec. VI, and Sec. VII-A).

and the richer programming interfaces from osmdroid to provide a more accessible user interface.

A. Visual Representation

Navigation Aid superimposes the current position of the mobility platform, targets chosen via speech or finger-type, as well as points of interest and formerly selected targets on top of a map (cf. Fig. 5). There are two overlays superimposed onto this map: an indicator for the direction the user should be heading to as well as the navigation aid, i.e. an arrow-like turn advice that gives information on actions to take in order to reach the goal.

A second interface tab is intended for users with visual impairments who might benefit more from seeing the direction arrow and the navigation aid. Both elements are displayed as large as possible here. There is no map shown in the background to minimise visual clutter and to make the navigation aid as easy to see as possible.

B. Target Selection

Since the abilities of Navigation Aid’s users will vary, we have conceived several ways to select a target.

To quickly set a target location, a user may perform a long tap gesture on any point on the map. Alternatively, a text input field allows the user to input a target address. We are using the Android Geocoder API [1] to convert a textual input to a coordinate. Since this system compensates the input of partial addresses, e.g. just a street without the city or country, by using the current location to find the nearest match, this method allows for quick entry of a destination as precise as a user wants to be.

Alternatively, instead of typing into the target input field, a user may also use the voice recognition feature of the Android OS to state the desired destination. This is usually faster than typing in an address, especially for users not familiar with a keyboard layout. It is also a more accessible solution for visually impaired users, who are also assisted in the Navigation Aid app by labeling the UI elements for speech synthesis feedback. This enables the user to explore and navigate the app through gestures without sight of the screen.

VIII. Experimental Evaluation

First results in Fig. 6 reveal superior performance of the localization scheme applied, compared to pure GPS localization. During the experiment, the Topro Troya walker was hand-pushed along a path that pretty much corresponds to the particle-filter output trajectory. We initialized the localizer module with the correct starting-point, i.e. the entry to the depicted building. Since particle filters that provide localization information from scratch need a certain amount of time in order to converge to the actual position, we plan to use entries of buildings as known portals in scenarios with combined indoor- and outdoor-localization.

As an odometry sensor, we used a pair of the prototype sensors shown in Fig. 3(a) to derive distance and heading using two accelerometers and a gyroscope on each sensor. The sensor data is fused using an Extended Kalman Filter [25] based on a purpose-built process and measurement model to predict wheel angles. This ensures a maximum of robustness while using noisy sensor data especially while driving on uneven terrain (publication in preparation).

We will soon start extensive field trials in Bremen and Oldenburg (DE), Utrecht and vicinity (NL), and Barcelona (ES). The expected feedback will be essential for deciding, for example, which elements on the interface are suitable for which kind of user, and will significantly contribute to further improvement.

IX. Conclusion and Future Work

Navigation Aid is intended as a low-cost solution for existing platforms (walkers, but also (non-electric) wheelchairs, the future tricycles developed in ASSAM—in fact any kind of vehicle with two front or back wheels). When the software port to Android OS is complete, only the app on an Android tablet and two OdoWheel units for the (back) wheels are required.
Then Navigation Aid provides superior navigation accuracy, comparable to car navigation systems with internal odometry connection, but in this case for users with special needs: the user of a walker (or wheelchair, tricycle) who will safely stay on the path or sidewalk, where Navigation Aid will select a path avoiding difficult surfaces (such as gravel), circumvent obstructions, and find the lowered curb to cross a street, or the ramp instead of a staircase.

Thus the user can take the walker with Navigation Aid along on the next visit to Barcelona, Bremen or Utrecht— wherever the map data are sufficiently enriched by necessary accessibility data. We envision the community to become increasing active here, but also companies to augment the data to be sufficiently complete and consistent to get the certification ASSAM-ready.

We also expect mobility assistants to be leased, rented out for share, or provided free of charge at supermarkets, hotels, airports, touristic areas, etc. The OSM standard will be extended indoors and by annotations for specific requirements.

Feedback from field trials will show, to what extent users with slight visual or cognitive impairments will already be sufficiently assisted; for example a user with slight dementia who is otherwise afraid not to be able to find back home. In general, more compensation for progressive loss of abilities will be required. The ASSAM project will develop further variants of the mobility assistants with motorized wheels and laser-scanner support for safe environment recognition and 3D obstacle avoidance, in particular for users with visual impairments.

Some users will require additional security assistance by interacting with a real person. In emergency situations, an alarm raised by the user or the system shall automatically connect to a call center with further development of the ASSAM software. A caregiver will provide online navigation assistance. As position and direction of the platform are known, a first question like “do you see the city hall in front of you?” will establish contact and provide assurance. So even in emergencies you will not be lost on your trip to a foreign destination!

REFERENCES


