Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 27 (2014) 28 – 37

Procedia Computer Science

Proceedings of the 5th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2013)

Navigation framework using visual landmarks and a GIS

M. Serrão[b], J.M.F. Rodrigues[a,b], J.M.H. du Buf[a]

[a]*Vision Laboratory, LARSyS, University of the Algarve, 8005-139 Faro, Portugal*
[b]*Instituto Superior de Engenharia, University of the Algarve, 8005-139 Faro, Portugal*

Abstract

In an unfamiliar environment we spot and explore all available information which might guide us to a desired location. This largely unconscious processing is done by our trained sensory and cognitive systems. These recognise and memorise sets of landmarks which allow us to create a mental map of the environment, and this map enables us to navigate by exploiting very few but the most important landmarks stored in our memory. In this paper we present a route planning, localisation and navigation system which works in real time. It integrates a geographic information system of a building with visual landmarks for localising the user and for validating the navigation route. Although designed for visually impaired persons, the system can also be employed to assist or transport persons with reduced mobility in way finding in a complex building.

Keywords: Navigation; Accessibility; Visually impaired persons; Geographic information system; Vision.

1. Introduction

Finding an office or a room in a large public building, such as a university, a city hall or a shopping centre, is often difficult for normal persons. For blind persons it is far more difficult because they must completely rely on people passing by to ask for information. When a building is becoming more familiar, they can memorise all locations for moving about.

There exist systems which try to alleviate the problems of blind and visually impaired persons. For the state-of-the-art of existing systems we refer to [1-5]. Navigation is one of the fundamental problems to be solved in creating completely autonomous mobility systems, mainly for blind persons or mobile robots. Navigation models can be classified into local and global ones. Local models navigate by exploring information in the environment immediately around the actual position. They are completely blind to everything beyond the reach of the system's sensors. In contrast, global navigation takes into account the environment as a whole, far beyond the reach of the system's sensors. They enable auto-localisation and optimisation of navigation routes.

There are several systems which deal with simultaneous localisation and mapping (SLAM). Morioka et al. [6] proposed a model that allows navigation in crowded environments such as shopping centres and transport

doi:10.1016/j.procs.2014.02.005

service stations. Their method is based on 3D points extracted from image sequences and odometry calculations. A completely autonomous SLAM algorithm based only on image sequences was presented by Saleiro et al. [7], although it has only been tested with a small robot in a restricted area.

Within the models based on pre-built maps there are several solutions which are based on topological relations or statistical methods for the determination of the location. Arras et al. [8] developed a method based on statistical Kalman filters. Position estimation is based on the average and variance of the information calculated at a previous location, data from odometry sensors, and the geometric observation of the environment. This method has the disadvantage that it cannot automatically recover when the location is lost. Seitz et al. [9] presented a pedestrian navigation algorithm optimised for smart mobile platforms with low-cost sensors and limited processing power. The algorithm is based on a Hidden Markov Model that combines WiFi positioning and dead reckoning. The hidden states are the positions of WiFi fingerprints in the database. State transitions include dead reckoning based on step-length estimation from acceleration measurements and compass heading. Positioning is achieved by correlating the actual WiFi signal strength with stored fingerprints.

Schmitz et al. [10] developed a navigation system that seamlessly integrates static maps with dynamic location-based textual information from a variety of sources. Each information source requires a different kind of acquisition technique. All acquired information is combined by a context management platform, and then presented to the user as a tactile or acoustic map depending on the sources available at the current position. Positioning is achieved by a combination of an inertial tracking system, RFID technology and GPS, and the user is guided to a desired destination by speech output and a haptic cane.

Almost all systems depend on different sensors to accomplish navigation. In this paper we present a navigation system which integrates data of a GIS of a building with only visual landmarks. Any normal camera can be used together with a small, portable computer like a netbook. GIS/vision-based localisation is complemented by navigation: at any time, the system traces and validates a route from the current position to a given destination. Although designed for being integrated in the Blavigator prototype [5], this system can be integrated in a robotic system that needs to navigate in a complex building. Summarising, we present a new navigation algorithm that works in real time, integrating existing GIS data with existing visual landmarks like objects and signs. We emphasise that, in contrast to other approaches, we do not distribute and employ special tags with location codes.

2. The System

For the validation of the concept, the prototype was developed on a netbook with a standard WiFi webcam. It was tested at the Institute of Engineering (ISE) of the University of the Algarve (ISE/UAlg). The user holds the camera in his hand. In the case of the SmartVision [2] and Blavigator prototypes [5], the camera is worn at the chest. However, these prototypes are also being implemented on a Smartphone, such that its camera can be easily pointed to different directions. In case of an uncertain user location due to missing or ambiguous visual landmarks, the user can point the camera to different directions by rotating it 180º horizontally and about 45º vertically. Any detected landmarks are matched against those in the GIS for each space. By combining the detected landmark positions with the traced route, the user is informed about the current location by a speech module. Finally, since the GIS/landmark system must be integrated in the Blavigator prototype, all visual functions have been optimised for small CPU time and memory usage, and most visual functions are based on a few basic algorithms which are employed only once for each video frame.

The system's component which handles spatial data was developed by using the *OGR Simple Features Library*. OGR is a part of GDAL - the Geospatial Data Abstraction Library [11]. For details about the GIS of ISE/UAlg (testing site) see [5]. Figure 1 shows a map of an ISE/UAlg GIS (left) and respective data table (middle) with the attributes of the selected region highlighted in yellow.

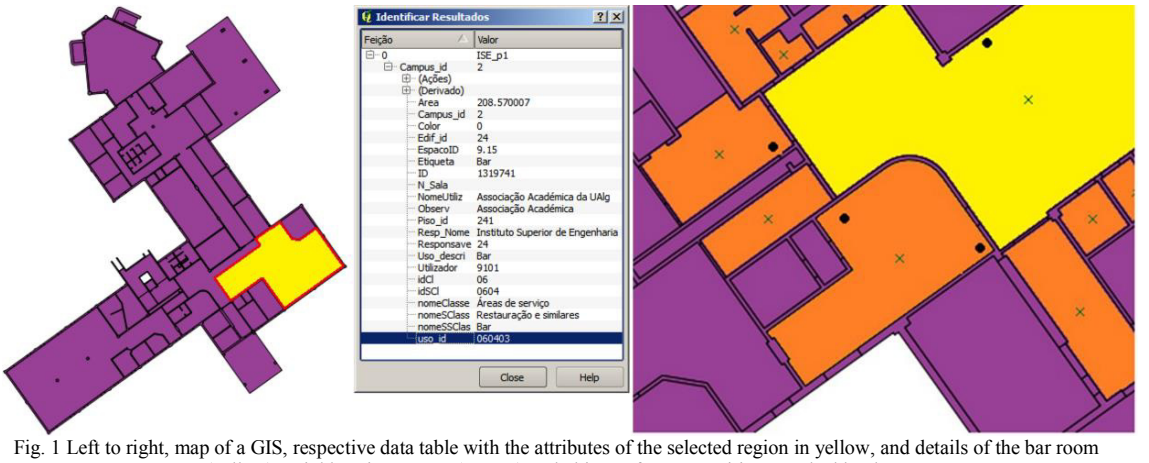

Fig. 1 Left to right, map of a GIS, respective data table with the attributes of the selected region in yellow, and details of the bar room (yellow), neighbouring spaces (orange) and object reference positions marked by dots.

The system comprises a *Training Module* for creating object libraries. These are formed by keypoints drawn from a set of images with different views of each object. Each keypoint is characterised by its neighbourhood by means of features extracted by the OpenSURF library [12]. After training, these libraries are then used to recognise the objects in the images captured by the camera (see Section 3 or [5]). The system is controlled by a *Main Module* for real-time localisation and navigation.

The main module executes a set of sequential processes. In the *initialisation process,* all parameters and libraries for object recognition are loaded. Also the maps files containing geographic information to be extracted and handled through the GDAL library [11], and the auxiliary applications used for character recognition, Tesseract-OCR [13], and the text-to-speech module Jampal for generating audible information for the user [14].

After initialisation, the *object recognition process* is responsible for analysing all image frames acquired by the camera, and it contextualises objects in the environment. Depending on the similarities between captured frames, it can assign unique identifiers for distinguishing visual landmarks. This serves to invalidate a same object (at a same position) when detected several times in consecutive frames. The *localisation and navigation process* is in charge of estimating the position of the user through the detected landmarks and of determining the shortest path between the current position and the intended destination. The *notification process* informs the user through vocal commands (other interfaces can also be used, see [2]). It reports identified objects, location information, and instructions to reach the destination. Below we present very briefly the object recognition process; for more detail see [5,15,16]. We will mainly focus on localisation and navigation. For the notification process we refer to [2,5].

3. Object Recognition Using Vision

As mentioned above, the data acquisition model is only based on vision by means of a normal camera. The goal is to extract reference or landmark objects such as doors, windows, lockers, text and figural signs, and fire extinguishers. The objects are contextualised through a frame identifier, the type of object and position coordinates (in the image), and they are stored in a first-in-first-out (FIFO) buffer. This buffer, also called the "navigation history," is later used for location estimation and navigation.

The process of object recognition starts by (a) capturing an image, followed by (b) a process that aims to contextualise detected objects by checking if the objects are identical in previous images. Contextualisation is

performed by recognising (c) doors, (d) stairs, (e) rectangles, (f) texts - OCR and (g) general objects using SURF [12].

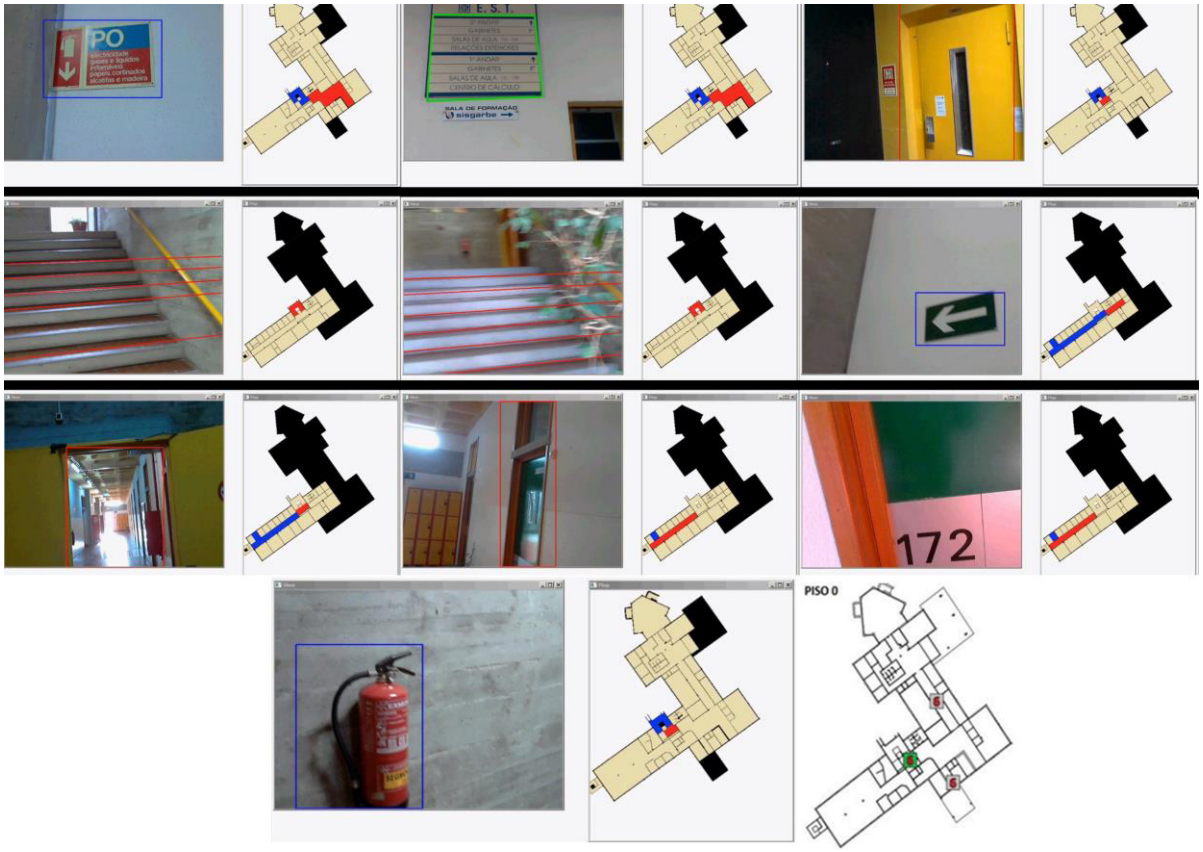

Fig. 2 Top three rows show detected visual landmarks (at left) and their detected positions in the map (at right); in red the position of the landmark, in blue the next room along the route. The bottom row shows the same, plus all the positions of extinguishers in nearby rooms.

Steps a) and c) to g) are due to lack of space not explained here. Detailed explanation is available in [5,15,16]. Examples of detected landmarks are shown in Fig. 2. In order to understand the next section, about route planning, localisation and navigation, we must keep in mind that object templates are captured at a distance of about 2 metres, and objects are usually detected at a distance between 1 to 3 metres.

Step (b), visual contextualisation, needs to be explained in more detail, because of problems due to artificial lighting, reflections, and limitations caused by detection algorithms like SURF. It is impossible to capture all information available in the visual scene in a single image (frame). Usually several frames are necessary to capture various objects, even in combination with movement of the user (camera) to detect the objects. The information acquired, although captured at different stages (frames and positions), can be related, because they belong to a same visual scene. Likewise, a same object can be successively acquired in a sequence of frames. This can result in incorrect interpretations if it is assumed that the captured object represents different objects, when in reality it is the same one. Therefore it is necessary to contextualise the information obtained with the visual scenes to which the objects belong.

Before objects are added to the navigation history, newly detected objects are compared to those already present in the FIFO buffer: the image identifier from which they were extracted, object type, and area in which

they are located within the image. Objects can be discarded if they were already recognised in the same image and at the same position.

This evaluation is done by a very simple and fast process: (a) the centre and area of any new object are compared with those of already detected objects (with the same frame identifier). If there is significant overlap, the object already recognised is removed; otherwise it is added to the history as a new object. (b) Also, to create and update the context of the objects it is necessary to test the similarity of a sequence of frames, in order to verify whether or not there is a transition to a new visual scene. There are various techniques to test the correlation between images and check their similarity: subtraction of images, colour histogram correlation [17], Hilbert transform [18], motion flow analysis [19] and correlation of key points from SIFT or SURF [20]. Here we use colour histogram correlation based on the work of Gargi et al. [17].

4. Route Planning, Localisation and Navigation

Using GDAL (see Section 2 or [5]) it is possible to extract the geographical information of the spaces, reference points and structures such as walls, doors and windows. This information is stored in memory by creating a database which includes all information necessary for navigation, such as the type of structure (room, wall, window, door) or reference object (fire extinguisher, exit sign, locker, etc.). Their geographical locations through the geometric centroids and neighbour relations are extracted. Neighbour relations are extremely important for navigation, route planning and determination of the current location.

Reference points of objects were introduced in the GIS vector map, using the Quantum GIS editing tool [21]: their global positions in geographical coordinates, and their region (space) identification. The regions are spatially grouped, forming neighbour relations with each other through walls, windows, doors and stairs. These relations allow us to determine the neighbourhoods of all regions by analysing the contiguity of access structures such as doors or stairs.

4.1 Route Planning

Figure 1 (right) shows an example of a room: a bar (marked in yellow) and nearby rooms (marked in orange) that can be accessed through a door, as well as their geometric centroids indicated by crosses. Also shown are reference object points in the regions, marked by black dots. These are referenced by a common identifier which relates the room with the object by the attribute “EspacoID” (Fig. 1 middle). Figure 2 (bottom right) shows for example positions of fire extinguishers on the ground floor near the bar.

Spatial relations allow us to build a matrix map, in binary format, indicating the neighbourhoods of all regions, with common access structures and reference objects. The matrix map enables efficient and fast examinations for determining the location and for planning the navigation route, i.e., the navigation path between a *start* and a *destination* point on the map (see Fig. 3). We implemented the lowest-cost path between these two points (rooms/divisions), taking into account access points between rooms like doors and stairs.

Calculating the navigation route is an iterative process that computes at every step the shortest path to the destination: (a) In the first step, the algorithm checks which access structure (door or stairs) neighbouring to the start region (room), is in the same direction and has the shortest distance to the destination. (b) Once this access structure has been determined, the room to which it gives access is determined. This yields a new starting point and (c) the direction to the destination is (re-)computed and the next access point (door or stairs) that is closest in the indicated direction is found (Fig. 3 left). (d) This process is repeated until the destination is reached. This process yields the most appropriate route in vector format. (e) Every time when a room is found without further accesses, like a room with only one door, the corresponding neighbourhood map (including the access structures) are updated and marked as invalid for the desired route (see yellow dots in Fig. 3 left and right). (f) The algorithm returns to its initial stage, and from the starting point a new route is calculated with the updated

neighbourhood maps. (g) If the start and destination points are on different floors (example of Fig. 3), then partial destination points (and partial starting points) are common stairs or elevators, depending on the user's preference, on each floor. The path is computed as explained before for each floor, taking into consideration the partial points. The final path is a single vector from the start to the destination that merges all partial start and destination points on each floor. Figure 3, from left to right, exemplifies route planning from floor 0 to an office on floor 2, using the stairs, with all vectors linking the geometric centroids of all selected rooms.

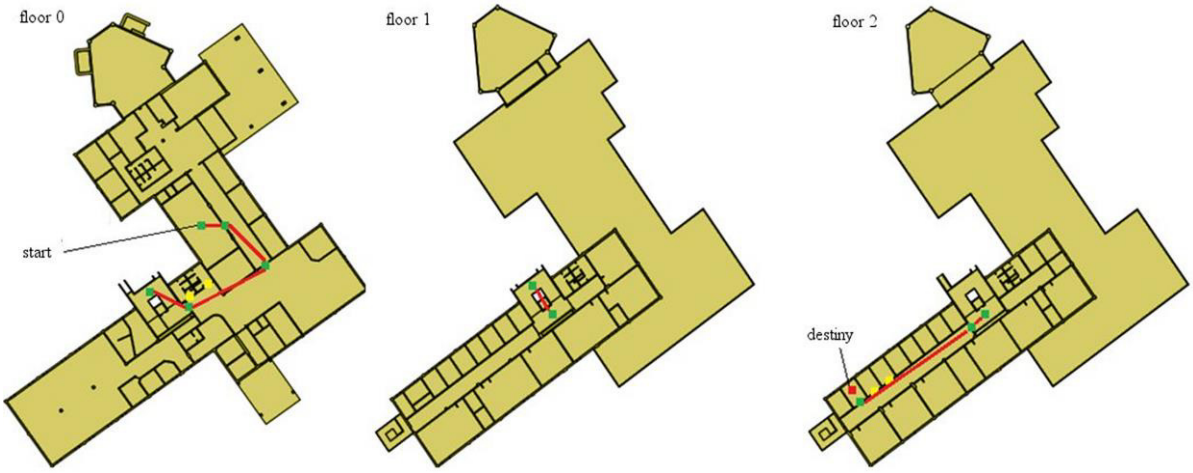

Fig. 3 Route planning, taking as starting point a room on floor 0, passing via a staircase to floor 2, and arriving at an office on floor 2.

4.2 Localisation and Navigation

As mentioned, a planned route is stored in a sequentially organised vector format, which includes pointers to the GIS of each room and access structures (doors and stairs). To guide the user along the route it is essential to determine the actual location, such that the system can generate the correct instructions to follow the path or to correct the position of the user.

The overall algorithm is shown in a block diagram in Fig. 4. The determination of the user location is supported by FIFO buffer data with the navigation history of the last four detected objects. The number of four was experimentally chosen for not compromising the performance of the algorithm. Using less objects results in a greater degree of uncertainty. To validate or determine the location, the localisation algorithm is executed every time when a reference object has been detected.

In the first step (a) weight values are assigned to all reference objects based on the route vector and floor transitions, emphasising those to be expected according to the estimated direction and last known floor. (b) Then, for each of the regions of the SIG (hallways, rooms, bar, bathrooms, etc.) a voting value is computed based on the quantitative distribution of objects stored in the navigation history, both for the regions/rooms and for their neighbours. The distribution is weighted by a value in function of the reference object in the region and identical to the last detected one with the highest weight (assigned in (a)), last known approximate position, navigation route, and relationship in the region between previous objects that were detected in frames with the same likelihood (by means of contextualisation; see Section 4).

The next step (c) consists of verifying which region has the largest voting value. This region/room is chosen as the new user position. (d) Once the possible location is chosen, it is necessary to check whether it is possible to find the exact position within the room by analysing the reference object (identical to the last detected) with a greater weight, assuming a new location with the same object but with a larger weight value, that is, if its

value is unique among similar objects within the region. Having a new position, (e) the data for the current position is updated, and (f) the navigation path and the vector for the next displacement are (re-)computed.

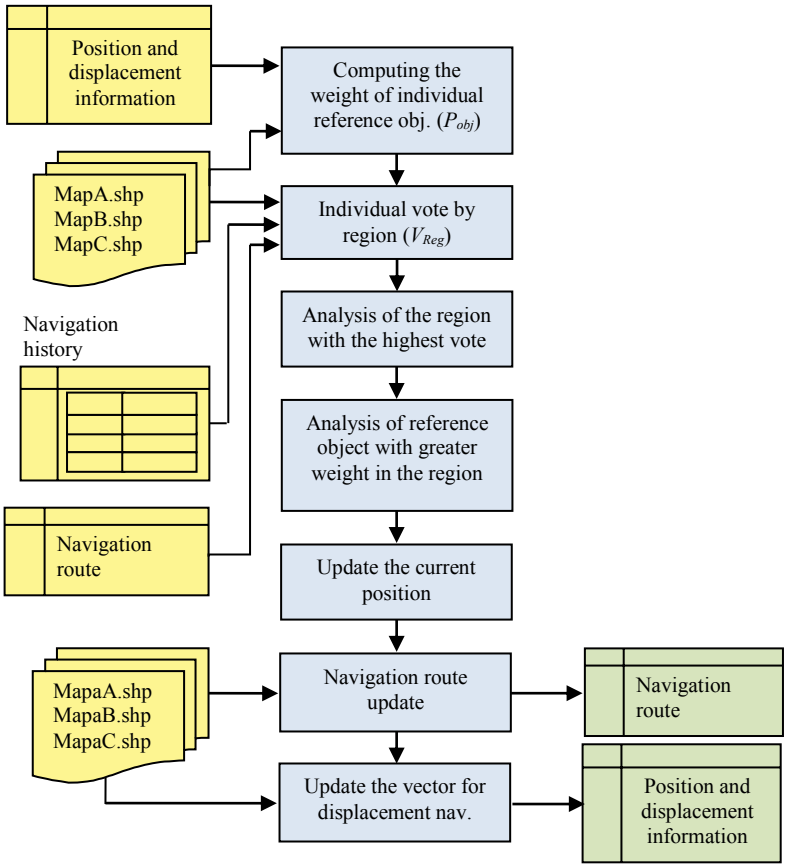

Fig. 4 Block diagram of localisation and navigation.

Steps (a) and (b) of the algorithm need to be more detailed. The first step (a) serves to highlight the GIS objects that are most likely to be detected by weighting them. The calculation of the weight of the object (P_{obj}) is applied to each of the GIS objects, taking into consideration a relative weight of the displacement vector (P_{dir}) and a relative weight of a floor transition (P_f), i.e., the presence of a staircase or elevator: $P_{obj} = P_{dir} + P_f$.

The weight factor for displacement (P_{dir}) allows to assign a greater weight to objects which are in the estimated direction (displacement) of the user relative to the last known position. The 2D displacement vector is composed of components according to the GIS south-north (e_y) and west-east (e_x) axes, see Fig. 5 (left), with the displacement vector in green (dir_{disp}) and the object vector in red (dir_{obj}). In the same figure, the 2nd column illustrates the values of the components e_x and e_y for each of the quadrants. To facilitate the calculation, these components can only be plus one, minus one, or zero. In Fig. 5 (2nd column) $dir_{obj} = (-e_x, +e_y)$ and $dir_{disp} = (-e_x, -e_y)$.

Now we can assign P_{dir} a weight by checking similarities found between the vertical and horizontal directions of dir_{disp} and dir_{obj}. The maximum weight is assigned to objects in the direction of the displacement: $P_{dir} = 3$ if $dir_{disp}\ (e_x) = dir_{obj}\ (e_x)$ and $dir_{disp}\ (e_y) = dir_{obj}\ (e_y)$. Objects that agree with a single component (vertical or horizontal) of the displacement vector are given a smaller value: $P_{dir} = 2$ if $dir_{disp}\ (e_x) = dir_{obj}\ (e_x)$ or dir_{disp}

(e_y) = dir_{obj} (e_y). A lower weight is assigned to objects that are in the opposite direction of the displacement vector. The value, however, should be more than 0 such that these objects are not totally discarded: P_{dir} = 1 if dir_{disp} (e_x) ≠ dir_{obj} (e_x) and dir_{disp} (e_y) ≠ dir_{obj} (e_y).

In the case of a building with multiple floors, we emphasise GIS objects that are on the user's current floor, by assigning a weight P_f (in the case of a single-floor building P_f =0). The weight is zero (P_f =0) during the initial phase where the position of the user is unknown, or in case when the system loses completely the position of the user (when the user chose a path different from the one suggested by the system), or every time there is a staircase (in order not to inhibit a possible floor transition). This way identical weights P_f are given to objects, regardless of the floor to which they belong. The user position then only influences the region vote; see step (b) below. However, when the position of the user is far from a staircase, it is likely that the user will remain on the same floor, and it is therefore necessary to assign a greater weight to objects situated on the same floor. For this reason we use P_f =3 for objects on the same floor and P_f =0 for objects on the other floors.

Step (b) consists of computing the vote for the region (room) were the user is. For each region the vote is calculated by $V_{reg} = P_{reg} \times R_p + (P_{reg} - 1) \times R_v$, with R_p the region under analysis and R_v the neighbour region. Here $R_p = N_p / N_{obj}$ and $R_v = N_v / N_{obj}$, with N_p the number of objects from the navigation history found in the region in question, and N_v the number of remaining navigation history objects not found in the region but which belong to the neighbour region. The total number of navigation history objects is N_{obj}, and P_{reg} is a factor to highlight the regions with objects detected when compared with other neighbour regions: $P_{reg} = Pmax_{obj} + P_{pos} + P_{perc} + P_{ID}$.

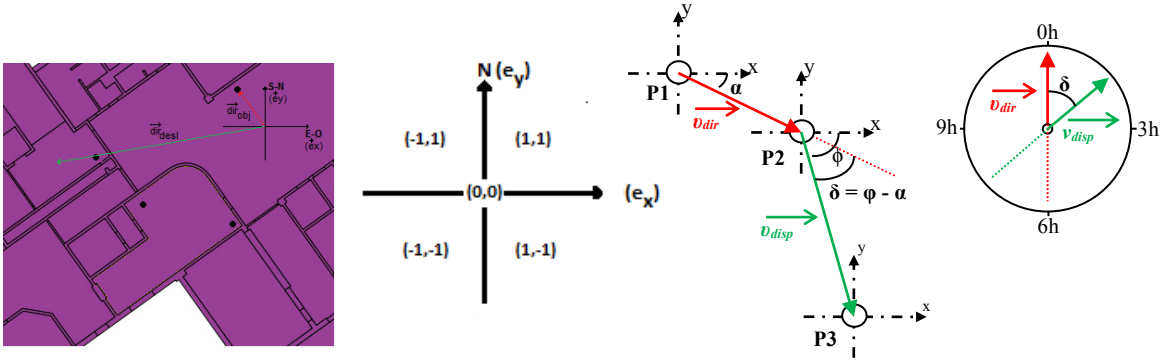

Fig. 5 Left to right, the representation of the vector displacement (dir_{disp}) and the object's direction (dir_{obj}) when the user is at the centre of the region, the values for each of the vertical and horizontal components e_x and e_y, the computation, and navigation instructions for the user given in clock hours (see text).

$Pmax_{obj}$ is the maximum value of the objects identical to the last detected ones in the region (room), i.e. $Pmax_{obj}$ = MAX $_i$ ($P_{obj,i}$), with i the index of the detected identical objects within the region. P_{pos} represents the relationship of the neighbourhood with the region were the user is assumed to be. It serves to emphasise regions which are close to the region were the user was last, indicating them as more favourable to the new user location when compared to other, more distant regions. We assign P_{pos} = 2 to neighbour regions of the last known user position, and 0 to other regions.

P_{perc} is the weight assigned to regions on the planned route. Its purpose is to highlight the regions belonging to the navigation route relative to other regions. When these regions have identical expected objects, we use P_{perc} = 1.5, otherwise 0. The last parameter, P_{ID}, represents the proximity of objects in the navigation history that were detected in frames with the same likelihood (belonging to the same GIS regions; contextualisation process), considering distances of up to 1 metre. For regions that validate this criterium, P_{ID} = 2, otherwise P_{ID} = 0. Figure 2 shows (in the right columns) in red the region of the detected landmark or reference objects (shown in the left columns) and in blue the next room on the route.

4.3 Navigation Notifications

Recognition and navigation notifications are intended to inform the user of detected objects, text information (written on signs), and to guide the user on the path to the destination. Here we used vocal commands by text-to-speech [14]. Any other interface can be used for the guidance commands [2]. Whenever a recognition of a new object occurs, this information is provided to the user, indicating the type of object (extinguisher, door, stair, text, etc.) and the detection zone in the captured image, considering the image being split into 3×3 regions, i.e. *left*, *centre*, *right* and *up*, *centre*, *down*. For example, in case of Fig. 2 bottom-left, the system says "extinguisher-left-down." The user can select between receiving only direction notifications, or direction plus object and/or text notifications.

User guidance is complex, since orientation using the cardinal points available via the GIS is very difficult for most if not all users. Due to this we use "hours" (see Fig. 5 right). To determine in which direction is the next destination, it is necessary to use the direction vector (v_{dir}) and the displacement vector (v_{disp}) in order to verify the angle between the two. The direction vector characterises the movement performed by the user from the last position (P1) to the current position (P2), while the displacement vector characterises the expected movement from the current position (P2) to the next destination point, door or staircase (P3), proposed in the planned route; see Fig. 5, 3rd column.

If δ is the angle between the displacement and direction vectors, and these vectors have angles φ and α with the x-axis, then $\delta = \varphi - \alpha$. Knowing δ, it is quite easy to convert to hours: $h = (360 - \delta)/30$ if $\delta > 0$ and $h = \delta /30$ if $\delta \leq 0$ (see Fig. 5 right). For example, in the case of Fig. 2, 1st row 3rd column, the system says "stair-three o'clock." This method, based on the combination of direction and displacement vectors, is only applicable if the system knows the user location, through an object (or group of objects) in the GIS, and the user is really following the instructions along the planned route.

5. Discussion

We presented a route planning, localisation and navigation system which integrates an indoor GIS of a building with visual landmarks detected by a normal, hand-held camera. The system was designed such that it can easily be integrated in the Blavigator prototype. The latter is already able to detect valid paths and any obstacles in local navigation. The system presented here complements local navigation with global navigation, although only indoor. The system works in real time on a netbook computer (Intel core i3, 1st series with 2GB RAM) and it was tested successfully in the ISE/UAlg building. The performance time is on average 0.5s for the localisation and 0.3s for the route planning.

The developed navigation framework offers robust localisation, even in cases where it is difficult to differentiate regions on the basis of reference objects that are present. The navigation history enables auto-localisation, by selecting the region that combines the largest number of nearby objects in the history.

However, the system supported in the estimated direction information to the next move (vector for displacement). Localisation usually fails when the user deviates from the assumed or expected path. The correct location is then restored as soon as sufficient navigation history (objects) or a single significant object exists. False positive and negative detections may also interfere negatively. In the future a verification technique that can rule out false detections by analysing the proximity relations with the other navigation history objects. Nevertheless, most planned routes could be followed from the starting to the destination, even when these were on different floors.

On-going work concerns including more landmarks in the database such that all spaces can be covered more densely, and detecting doors and stairs from oblique viewing angles, such that the user can limit the pointing angles of the camera to ±45º from the front. The final goal is a system which only employs a SmartPhone with a built-in camera, worn by a strap around the neck.

Acknowledgements

This work was partly supported by the Portuguese Foundation for Science and Technology (FCT), LARSyS project PEst-OE/EEI/LA0009/2013 and project Blavigator RIPD/ADA/109690/2009.

References

[1] CASBliP, Final activity report of the EU-funded CASBliP project, http://casblipdif.webs.upv.es (2009).
[2] J. du Buf, J. Barroso, J. Rodrigues, H. Paredes, M. Farrajota, H. Fernandes, J. José, V. Teixiera, M. Saleiro, The SmartVision navigation prototype for blind users. JDCTA: Int. J. of Digital Content Technology and its Applications 5(5) (2011) 351-361.
[3] O. Foong, N. Razali. Signage recognition framework for visually impaired people. Proc. Int. Conf. on Computer Communication and Management (2011) 488-492.
[4] J. Feng, Y. Liu. Wifi-based indoor navigation with mobile GIS and speech recognition. Int. J. of Computer Science Issues 9(6) (2012) 256-263.
[5] M. Serrão, S. Shahrabadi, M. Moreno, J. José, J.I. Rodrigues, J.M.F. Rodrigues, J.M.H. du Buf. Computer vision and GIS for the navigation of blind persons in buildings. Accepted for Int. J. Universal Access in the Information Society (2013).
[6] H. Morioka, S. Yi, O. Hasegawa. Vision-based mobile robot's SLAM and navigation in crowded enviroments. Proc. Int. Conf. on Intelligent Robots and Systems (2011) 3988-4005.
[7] M. Saleiro, J. Rodrigues, J. du Buf. Minimalistic vision-based cognitive SLAM. Proc. 4th Int. Conf. on Agents and Artificial Intelligence, Special Session Intelligent Robotics (2012) 614-623.
[8] K. Arras, N. Tomaris, B. Jensen, R. Siegwart. Precision and reliability for applications: Multisensor on-the-fly localization. Robotics and Autonomous Systems 34 (2001) 2-3.
[9] J. Seitz, T. Vaupel, S. Meyer, J. Gutiérrez Boronat, J. Thielecke. A Hidden Markov Model for pedestrian navigation. Proc. 7th Workshop on Positioning Navigation and Communication (2010) 120-127.
[10] B. Schmitz, S. Becker, A. Blessing, G. Matthias. Acquisition and presentation of diverse spatial context data for blind navigation. Proc. 12th IEEE Int. Conf. on Mobile Data Management 1 (2011) 276-284.
[11] GDAL-geospatial data abstraction library, open source geospatial foundation, http://www.gdal.org (2011).
[12] C. Evans. Notes on the OpenSURF Library. Tech. Rep. CSTR-09-001, Univ. of Bristol, http://www.chrisevansdev.com (2009).
[13] Tesseract-ocr. http://code.google.com/p/tesseract-ocr/, accessed on 15 Dec. 2012.
[14] Bennett: Jampal, http://www.jampal.sourceforge.net, accessed on 15 Dec. 2012.
[15] J. José, J. Rodrigues, J. du Buf. Visual navigation for the blind: path and obstacle detection. Proc. Int. Conf. on Pattern Recognition Applications and Methods 2 (2012) 515-519.
[16] S. Shahrabadi, J.M.F. Rodrigues, J.M.H. du Buf, Detection of indoor and outdoor stairs. Proc. 6th Iberian Conf. on Pattern Recognition and Image Analysis, Springer LNCS 7887 (2013) 847-854.
[17] U. Gargi, R. Kasturi, S. Strayer. Performance characterization of video-shot-change detection methods. IEEE Trans. on Circuits and Systems for Video Technology 10 (2002) 1-13.
[18] G. Priya, S. Domnic. Transition detection using Hilbert transform and texture features. American J. of Signal Proc. 10 (2012) 35-40.
[19] L. Krulikovska, J. Polec. An efficient method of shot cut detection. Int. J. of Computer and Information Engineering (2012) 63-79.
[20] C. Huang, H. Lee, C. Chen. Shot change detection via local keypoint matching. IEEE Trans. on Multimedia 10(6) (2008) 1097-1108.
[21] QGIS: Quantum geographic information system, http://www.qgis.org/, accessed on 15 Dec. 2012.