A bio-inspired model for robust navigation assistive devices

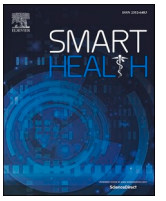
Simon L Gay, Edwige Pissaloux, Jean-Paul Jamont

Smart Health 33 (2024) 100484

Contents lists available at ScienceDirect

Smart Health

journal homepage: www.elsevier.com/locate/smhl



A bio-inspired model for robust navigation assistive devices

Simon L. Gay [a,*], Edwige Pissaloux [b], Jean-Paul Jamont [a]

[a] LCIS, Univ. Grenoble Alpes, 26000 Valence, France
[b] LITIS, Univ. Rouen Normandie, 76800 Saint-Étienne-du-Rouvray, France

ARTICLE INFO

Keywords:
Bio-inspired navigation
Artificial place cells
Artificial grid cells
Artificial head direction cells
Visual localization
Navigation assistive devices for visually impaired mobility

ABSTRACT

This paper proposes a new implementation and evaluation in a real-world environment of a bio-inspired predictive navigation model for mobility control, suitable especially for assistance of visually impaired people and autonomous mobile systems. This bio-inspired model relies on the interactions between formal models of three types of neurons identified in the mammals' brain implied in navigation tasks, namely place cells, grid cells, and head direction cells, to construct a topological model of the environment under the form of a decentralized navigation graph. Previously tested in virtual environments, this model demonstrated a high tolerance to motion drift, making possible to map large environments without the need to correct it to handle such drifts, and robustness to environment changes. The presented implementation is based on a stereoscopic camera, and is evaluated on its possibilities to map and guide a person or an autonomous mobile robot in an unknown real environment. The evaluation results confirm the effectiveness of the proposed bio-inspired navigation model to build a path map, localize and guide a person through this path. The model predictions remain robust to environment changes, and allow to estimate traveled distances with an error rate below 3% over test paths, up to 100m. The tests performed on a robotic platform also demonstrated the pertinence of navigation data produced by this navigation model to guide an autonomous system. These results open the way toward efficient wearable assistive devices for visually impaired people independent navigation.

1. Introduction

The autonomous navigation, indoor and outdoor, is a challenging task of human interaction with space. This task is significantly complex in the case of assistive navigation, especially in the frame of an autonomous navigation of visually impaired people (VIP).

Currently, human guide, white cane and dog are most frequently used for assist the VIP navigation. More recently, several GPS-based applications have been proposed to assist the VIP independent mobility and orientation through spoken turn-by-turn applications; they use also maps, landmarks and signage (e.g. Pissaloux & Velazquez, 2018). For example, the Nearby Explorer (APH, online) offers bacon-based and complete mapping navigation assistance. Seeing Eye GPS (online) provides a 3-layered approach: route, Point-Of-Interest (POI) and location. BlindSquare (online) and Smart Cap (Hengle et al. 2020) provide also a kind of socio-economic audio description of the VIP immediate vicinity and near space (nearest places, POI, their distance and direction and street intersections, all when well pointed). Google Map (online) offers voice guidance based on GPS. However, the precision of GPS localization may be considerably limited (e.g. on streets where tall buildings are present); moreover, the GPS does not work properly in indoor environments.

* Corresponding author.
E-mail address: simon.gay@lcis.grenoble-inp.fr (S.L. Gay).

https://doi.org/10.1016/j.smhl.2024.100484
Received 10 June 2023; Accepted 15 April 2024
Available online 17 April 2024

Current research tries to solve the problem of indoor assistance of VIP independent navigation. The ICT (Information and Communication Technologies) aim to provide an efficient support for wayfinding by the VIP. A comprehensive overview of existing indoor navigation systems can be found in Fallah et al. 2013; Flores & Manduchi, 2018. The most popular solutions are based on indoor infrastructures such as WI-FI networks, on IR/RFID/Bluetooth beacons, on QR-codes (in buildings such as airports, shopping malls, hospitals, schools, museums, etc.), or on a magnetic signature (e.g. Liu et al. 2016; Liu et al. 2017). Such solutions require the redeployment of specific sensor networks and their maintenance.

Wearable technologies, mainly computer vision (CV, e.g., Fusco & Coughlan, 2020), inertial sensing (IS), and visual-inertial odometry (VIO, combining CV and IS, e.g., Orb-SLAM3, Campos et al. 2021) avoid this additional equipment and their spatially limited redeployment. The CV may allow SLAM–based approaches (Simultaneous Localization And Mapping), useful for robot autonomous navigation. However, constructed model of the environment may have discontinuities due to motion drift, whose resolution is a major challenge in SLAM approaches. The IS and VIO drift over time, especially if we consider the irregular gait of a pedestrian (Flores & Manduchi, 2018), or unexpected obstacles. The VIO estimates the user's movements in the space (Fusco & Coughlan, 2020). Associated with a map, VIO allows to follow the user's displacement.

Several ICT navigation models propose to overcome the above-mentioned issues by drawing inspiration from mammals' navigation, especially after the discovery of place cells (O'Keefe & Nadel, 1978), grid cells (Hafting et al. 2005) and head direction cells (Taube et al. 1990) in rats' brain. These ICT vision-based navigation models try to mimic the mammals (mainly rats) abilities to map, localize and define navigation strategies. They integrate more or less accurately the properties of navigation-specialized neurons to construct robust internal and topological representations of the environment, but also to validate biological hypotheses on cognitive mobility.

Several neurobiological inspired models of place cells (PC) and grid cells (GC) have been experimentally tested with a camera of the field of view larger than human vision system (e.g. omnidirectional or pan cameras). These models can map and track position in initially unknown environments, and some of them can maintain a stable and exploitable model of the environment over long periods (Chen & Mo, 2019; Gaussier et al. 2019; Karaouzene et al. 2013; Zhou et al. 2018; Zhou et al. 2019). However, omnidirectional or pan cameras are not practical for use as a portable or wearable device.

RatSLAM (Milford & Wyeth, 2010), SeqSLAM (Milford & Wyeth, 2012; Tang et al. 2018 models encode position and orientation using 3D attractor network of *pose cells*, mimicking both grid cells and head direction cells. Pose cells recognize visited places, and estimate position and orientation data using images memorized during the navigation. These systems are robust to environmental changes, and able to map very large environments and track positions. However, these models are constrained to forward-motion only, therefore they cannot suit for human mobility assistance.

Our bio-inspired navigation model (Gay et al., 2021a) constructs a navigation graph using a model of Place Cells allowing to map and navigate in an environment (global navigation), and a model of Grid Cells and Head Direction Cells (HDC) to localize around a PC and navigate toward neighbor PCs (local navigation). Considering the environment as a graph, with PCs as its vertices, makes the model tolerant to motion drift, as two adjacent PCs are connected in a local model of the environment, independently of their positions in the global navigation graph. Consequently, such a model is suitable for assistive navigation in dynamic spaces such as a museum, shopping malls, airports, etc., according to the binary space partitioning principle (Pissaloux et al., 2017). So far, this model was tested in simulated environments only (proof of concept). This paper proposes its implementation using a stereo camera demonstrating its feasibility and relevance for navigation in a real environment.

The paper is organized as follows: Section 2 describes the proposed navigation model, while Section 3 presents an implementation of this model for real environments. Section 4 defines the test environment and analyses the collected results. Section 5 provides an evaluation of this model on an embedded autonomous platform, demonstrating the possibility to design small wearable assistive devices and evaluating the pertinence and usability of navigation and guidance data provided by the system. Finally, Section 6 concludes the paper and proposes future extensions of the navigation model.

2. A bio-inspired navigation model

This section summarizes the navigation model proposed in Gay et al. (2021a) and describes its underlying operation mechanisms: the data context used as input, the three used types of cells (Place Cells, Grid Cells and Head Direction Cells) and the environment model construction steps.

2.1. The environment context

The navigation model requires a sensory system able to provide a visual context of the surrounding environment under the form of a

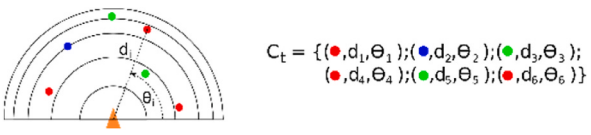


Fig. 1. Example of environment context. The contexts is a set of points of interests (here 6 POIs) characterized by a type (colors in this figure) and their positions (here defined as a couple (d,θ)) in egocentric polar reference frame, centered on the sensory system (orange triangle).

list of points defined by a type and a position in egocentric (sensory system) reference frame. The context C^t is thus a set of couples *(type of element, position)*, noted *(e,p)*, characterizing the environment at the instant *t*. Fig. 1 illustrates an environmental context.

Several sensors with distance measure capability can be used to obtain such contexts (e.g. LIDAR, stereo camera, optic flow/odometry measure). The model can use sensors with a limited field of perception but a minimum of 90° is recommended to better differentiate rotations from lateral translations. As we aim at wearable assistive devices, we use passive sensors only, as several active sensors (e.g. LIDAR) may be dangerous for people in the immediate vicinity. We thus propose an implementation relying on a stereovision camera (see Section 3).

2.2. The place cell model

Place cells (PC) (O'Keefe & Nadel, 1978) are neurons of the hippocampus of mammals' brain associated with a specific position in the environment. A PC becomes active when the animal is near the position associated with this PC.

In our model, a PC *P* is a structure that records the context observed from a specific position in space, and recognizes this place when moving in its vicinity, by comparing the recorded context with the currently observed context C^t. Formally, a place cell P_i is a structure that records the context C^{ti} observed when creating P_i at time t_i. We note C_{Pi} the context recorded by the PC P_i (with $C_{Pi} = C^{ti}$). The PC similarity function is a function $P: \{C\}^1 \rightarrow \Re$ defining the similarity or dissimilarity of two contexts. This function is defined by Eq. (1):

$$P(C_{Pi}, C^t) = \sum_{a \in C_{Pi}, b \in C^t} id(a,b) \times f^P(d(a,b)) \quad (1)$$

where $id(a,b) = 1$ when *a* and *b* are points of interests of the same type, and -1 otherwise, $d(a,b)$ is the distance between *a* and *b*, and f^P is a decreasing and positive function, where $f^P(0) = 1$, defining the size of PCs' receptive field. In Gay et al. (2021a), a bounded linear function was used, although other functions (e.g. Gaussian) were tested with similar results.

Neighbor PCs are connected to form a navigation graph, allowing to navigate in the environment by moving from a PC to the next one in the graph. The ability to move from a PC to a neighbor requires to know the system's position around a PC. Moreover, the navigation graph construction requires to define when to create a new PC. To answer these questions, we drew inspiration from other types of neuron implied in navigation, the *Grid Cells* and *Head Direction Cells*.

2.3. Grid cells and head direction cells models

Grid cells (GC) are neurons involved in displacement estimation and path integration (Hafting et al. 2005), mainly found in the entorhinal cortex of mammals' brain. A GC has a receptive field forming the lattice of an infinite hexagonal grid, making a GC becoming active periodically when the animal is moving in straight line. GCs gather into discrete modules (Stensola et al. 2012): GCs of the same module have receptive fields forming the lattices of grids with the same orientation and same spacing (distance between lattices), but with different translation offsets, making receptive fields of these cells of the same module covering the entire environment. A module thus forms a toroidal support that can measure movements in the environment by repeating itself indefinitely, making possible to integrate the movement of the animal by observing the activation sequence of GC of a module.

Our model draws inspiration from this module operation principle, and uses, in the current version, a unique module of GC. The module defines a relative position p_{ji} between each pair of GC (G_i, G_j) that it contains. This position is equivalent to the shortest linear movement allowing to move from G_i to G_j in the module's toroidal reference.

The operation principle is the following: the first created PC of the navigation graph is associated to a GC G_c of the module. As the module forms a toroidal surface, any GC can initially be selected, and becomes the *center* of the module. Then, the PC sends its context C_P to the module, and each GC G_j will generate a predicted context C_{Gj} that should be observed if moving according to relative position $p_{jc}=(G_c, G_j)$. Thus, each GC G_j generates a context $C_{Gj} = \{ (e, p') \mid (e,p) \in C_{Gc}, p' = p + p_{jc}\}$. These contexts are then compared to observed context C^t. This comparison uses the similarity function given in Eq. (2):

$$G(C_{Gi}, C^t) = \sum_{a \in C_{Gi}, b \in C^t} id'(a,b) \times f^G(d(a,b)) \quad (2)$$

where $id'(a,b) = 1$ when *a* and *b* are points of interests of the same type and 0 otherwise (i.e. dissimilarity is not considered), $d(a,b)$ is the distance between *a* and *b*, and f^G is a decreasing and positive function with $f^G(0) = 1$. The function f^G defines a narrower receptive field for GCs than for PCs (an example of implementation is given in Section 4.1). The GCs with the greatest value provide the position of the agent in the PC's reference. Fig. 2 summarizes this principle.

To handle rotations, the model draws inspiration from *Head direction cells* (HDC) (Taube et al. 1990), that are specific neurons that activate when the animal's head is in a predefined orientation in the environment (in allocentric reference). The model proposes to use a set of HDC and associates each HDC with a predefined angle φ, the set of HDC covering the interval $[0;2\pi[$ uniformly. Each HDC H_φ generates a "rotated" version of observed context C^t, by applying a rotation of angle φ to positions of elements of C^t. These contexts, noted $C_{H\varphi}$, are then used instead of C^t to be compared to GCs' generated contexts C_G. The couple HDC-GC providing the greatest

[1] https://osoyoo.com/2019/11/08/omni-direction-mecanum-wheel-robotic-kit-v1/.

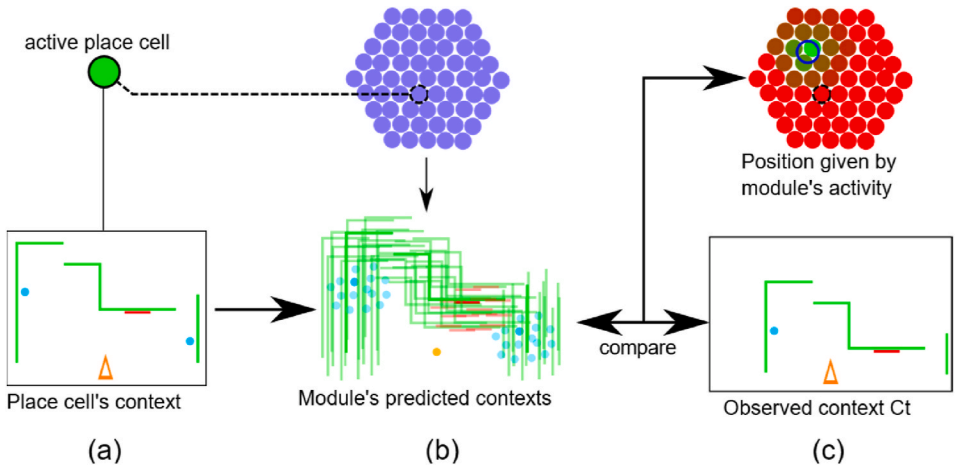


Fig. 2. Principle of local localization. a) A newly created place cell records the current observed environmental context C^t, and associates itself with a grid cell of the module, which becomes its *center*. The POIs represent walls (green) and several recognizable elements (blue and red), in the sensory system (orange triangle) reference frame. b) Using the recorded context of the PC, each grid cell predicts the context that should be observed at its associated position (from center Grid Cell). c) The predicted contexts are compared to currently observed environmental context C^t. The GCs with greatest similarities (here in green) define the spatial position in the PC reference.

correspondence (i.e. maximum $G(C_{H\varphi}, C_{Gi})$) between their contexts provide the position and orientation in the PC's reference. Note that these contexts $C_{H\varphi}$ are also used to recognize places with PCs irrespective of the agent's orientation.

As the position and orientation of the system, relative to PC's position, can be defined, it is possible to compute the position of points of interest of current context C^t in the reference of the currently active PC. This makes possible to update and complete the contexts C_P of the current PC while moving around its position.

2.4. Construction and exploitation of the navigation model

The local tracking principle described in Section 2.3 is however limited to the GC module's coverage. To allow tracking beyond this limit, the model uses the following principle: as we defined a center of the toroidal module, we can also define a border using GCs of the module that are the most distant from the center. Then, when reaching one of these "border GC", a new PC is created. This new PC records the current environmental context, connect itself to the reached GC and is connected to the previous PC in the navigation graph. The border GC becomes the new center of the module, and GCs' contexts are updated using the new PC's context (c.f. Fig. 3). Thus, when moving in the environment, the model regularly creates new PCs, enabling the construction of a navigation graph reflecting the topology of the environment.

Before creating a new PC, other PCs of the graph compare the current context C^t with their own recorded contexts. Thus, when reaching a "border" GC of the module centered on currently active PC P_i, other PCs P_k compute the similarities $P(C_{Pk}, C^t)$ between their own contexts and environmental context. If a PC P_j "recognizes" the current context through a high similarity (i.e. $P(C_{Pi}, C^t) = \max_k P(C_{Pk}, C^t)$ and $P(C_{Pi}, C^t)$ is greater than a predefined threshold), implying that a sufficient amount of points of interests are recognized and at the expected position, then P_j becomes the new active PC and is connected to P_i. This principle allows to close a loop in the navigation graph.

The navigation graph can be exploited to join a specific position in the environment: a pathfinding algorithm defines a sequence $S = (P_0, \ldots, P_k, \ldots, P_n)$ of PCs between current and target positions. Then, by considering the GC connected to the next PC P_{k+1}, and the current position on the GC module (in the currently active PC P_k reference), it is possible to define the movement required to reach P_{k+1}. When approaching the position of P_{k+1}, it becomes the new active PC, and the process is repeated to join P_{k+2}, until reaching P_n.

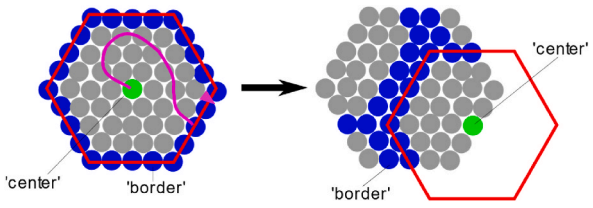


Fig. 3. Creation of a new Place Cell. As the current PC defines a center in the module, border GCs can be defined as GC that are the furthest from the center in the toroidal module surface (here in blue). It is then possible to track the movements (pink trajectory) within the module coverage. When reaching such a "border GC", a new PC is created, recording the current context C^t, and associates itself with the reached border GC. This new PC redefines the center of the module (and also the borders), allowing to continue the position tracking.

The whole navigation model is summarized in Fig. 4.

3. Adaptation of the navigation model for a real-world environment

Using the presented above navigation model in a real-world environment requires to develop a sensory-system able to generate and provide a context of points of interest that can be recognized and localized in space. Thus, any sensor that can measure distances, such as a LIDAR or a stereo camera, can be used. As we aim at developing wearable assistive navigation devices, we use a stereo camera, as it is a passive sensor (no laser to probe the environment), and its small size allows discrete devices. The use of passive sensors is also an advantage for groups of autonomous systems, such as robots, to avoid mutual interferences between sensors.

To validate our model in real environment, we propose a first implementation using a stereo camera. We used a *Sony PlayStation 4 stereo camera* (Fig. 5) modified to be used on a PC through standard USB3 port (Sieuwe Elfering github repository, online). This device is equipped with two cameras with a field of view of 85°, and has an exploitable depth perception with a field of view of 70°. Despite being lower than recommended 90°, this field of view is sufficient for our tests. Each camera provides a high-resolution image of 1280x800 pixels. The cameras are however too sensitive to sunlight, limiting the tests to indoor environments.

This section describes the vision system used to generate the environmental context, and adaptations of the navigation model and its parameters. Details of the implementation are provided in parallel with additional explanations.

3.1. Visual cues

We propose to use vertical lines in visual scenes as cues. Such elements have multiple advantages: they are omnipresent in the environment (especially in indoor environments), and easy to detect. They also facilitate their localization through stereo-vision, and can be projected as punctual element on the navigation plane. Moreover, the vertical nature of these element makes the feature invariant to camera height changes. As we detect vertical lines through high changes in light intensity level, we propose to define two types of vertical lines based on gradient direction: *light to dark* (*l2d*) and *dark to light* (*d2l*).

3.2. Detection of vertical lines

We propose a simple and minimalist algorithm to detect vertical structures in images. First, images are converted to greyscale images, and an interest value v_i is computed for each column $i \in [1, width\text{-}2[$ of one image, both for *l2d* and *d2l* lines (Eq. (3) and (4)):

$$\mathrm{v}_{\mathrm{i}}^{l2d} = \sum_{k\in[-1,1]} \sum_{j\in[0,height[} \frac{1}{|\mathrm{k}|+1} \times \max\left(p_{i+k,j} - p_{i+k+2,j}, 0\right) \tag{3}$$

$$\mathrm{v}_{\mathrm{i}}^{d2l} = \sum_{k\in[-1,1]} \sum_{j\in[0,height[} \frac{1}{|\mathrm{k}|+1} \times \max\left(p_{i+k+2,j} - p_{i+k,j}, 0\right) \tag{4}$$

where $p_{i,j}$ is a pixel of the image at line j and column i. Note that we only consider one line in 20, as it is not necessary to use all lines (and it reduces the calculation volume). Then, the set of vertical lines is defined by columns i where v_i is greater than a threshold τ_v and is maximum over a certain range n around i (Eq. (5) and (6)). We used a threshold of τ_v=*2000* and a range of *[i-20, i+20]* around the column, offering a good compromise between number and detectability of lines).

$$L^{l2d} = \{\, i \mid \mathrm{v}_{\mathrm{i}}^{l2d} > \tau_{\mathrm{v}}, \mathrm{v}_{\mathrm{i}}^{l2d} = \max_{k\in[i-n,i+n]} \mathrm{v}_{k}^{l2d} \,\} \tag{5}$$

$$L^{d2l} = \{\, i \mid \mathrm{v}_{\mathrm{i}}^{d2l} > \tau_{\mathrm{v}}, \mathrm{v}_{\mathrm{i}}^{d2l} = \max_{k\in[i-n,i+n]} \mathrm{v}_{k}^{d2l} \,\} \tag{6}$$

3.3. Localization of vertical lines

We use the disparity between images to define the distance of a vertical line in the environment. As we compute the disparity on a limited set of pixels instead of the whole image, we proposed to use an optical flow algorithm instead of a classic disparity algorithm. An optical flow algorithm estimates the apparent movement between two consecutive frames of a video by detecting points of interest on the first frame, then detecting the position of these points on the second frame. By using left and right images of the stereo camera, it becomes possible to estimate the disparity between points from vertical lines. More precisely, we use the function *calcOpticalFlowPyrLK*[2] of OpenCV library.

The disparity is computed on points of vertical lines that offer a sufficient light gradient τ_{grad} for a detection on the second image. A point *(i,j)* on the image, that has a light value of $p_{i,j}$, is then used for disparity estimation if $i \in L^{l2d}$ and $p_{i,j}$-$p_{i+2,j} > \tau_{grad}$, or $i \in L^{d2l}$ and $p_{i+2,j}$-$p_{i,j} > \tau_{grad}$. A gradient threshold of $\tau_{grad} = 10$ offers a good compromise between number and quality of points of interest.

The camera is calibrated to define parameters allowing to convert the position *(i,j,disp)* of a point of interest on the image into

[2] https://docs.opencv.org/3.4/dc/d6b/group_video_track.html#ga473e4b886d0bcc6b65831eb88ed93323.

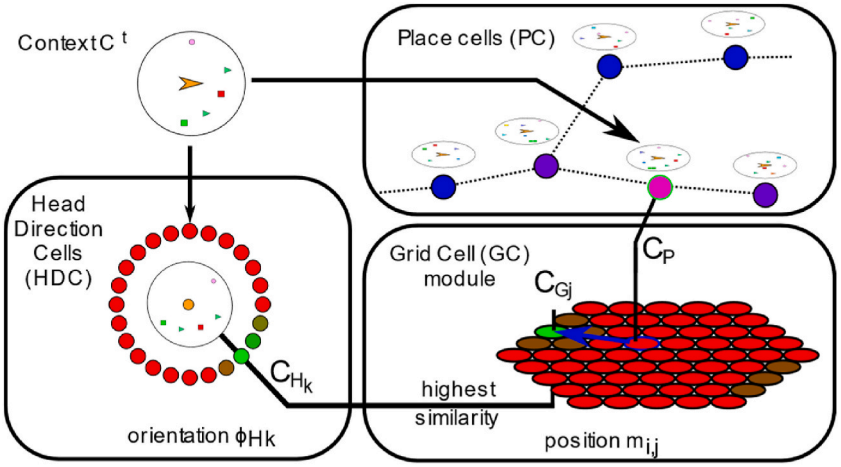


Fig. 4. Navigation model. PCs record the environmental context C^t observed at their attributed position, and are associated to GCs of the module. The currently active PC sends its contexts to the module, where each GC generate a predicted context C_{Gj}. HDCs generate "rotated" contexts C_{Hk} from observed context C^t. The rotated and predicted contexts with the greatest similarity provide the position and orientation of the system in the active PC's reference. The PCs form a navigation graph allows a global navigation by following a sequence of PC. The PCs use also rotated contexts C_{Hk} to recognize an already visited place.

Fig. 5. The used stereo camera is a *Sony PlayStation 4 camera* model 1. The camera provides two 1280x800 images. The cable was modified to be used on a standard USB3 port.

cartesian coordinates *(x,y,z)*. The used camera provides a couple of rectified images, simplifying the conversion functions, provided in Eq. (7):

$$z = \lambda \ / \ (\text{disp} \ - \text{disp}_{\text{min}}) \tag{7}$$

$$x = (i - \text{width} / 2) \times z / \mu$$

$$y = (j - \text{height} / 2) \times z / \mu$$

where *disp* is the disparity of the point between left and right image, and $disp_{min}$ is the minimum measurable disparity value obtained for points at 'infinite' distance. For a PS4 stereo camera, the calibration provides the following parameters: $disp_{min}$=20.5, λ=6761.6, μ=804.

We only keep points from a slice parallel to the ground (i.e. *y* in a predefined range) to eliminate points on ceiling or floor. Points with the same type (*l2d* or *d2l*) and column *i* (i.e. same vertical line) are gathered. The distance z of a vertical line is defined as the median value of distance of points composing it, eliminating noisy values. Coordinates *(x,z)* of lines are converted into polar coordinates. The result is a set of points $\{(e_k, \theta_k, d_k)\}_k$ (where $e \in \{l2d, d2l\}$) forming an environmental context C^t that can be exploited by the navigation mechanism. Fig. 6 shows the main steps of this process.

3.4. Adaptation of the navigation model

The "real-world" model implementation did not require important changes from model used in virtual environments (Gay et al., 2021a): indeed, the context obtained with the stereoscopic system has the same nature than contexts used in simulated environments. However, due to the noisy input, the estimated position of visual cues, and thus the estimated position in space, is sometime inconsistent. To limit such inconsistences, we proposed to only consider GCs and HDCs whose associated position and orientation are close to the current position and orientation of the system. We thus propose to only considers the current most active GC and its closest neighboring GCs, and HDC whose orientation is in a certain range around current estimated orientation. By limiting the movement possibilities around the current position at each new frame, the system avoids considering unrealistic move estimations due to noisy inputs. Another advantage of this method is that it also reduces the number of contexts to compare and thus saves computational resources. In the experiments presented in this paper, the navigation system considers the current GC and its eight neighbors, and a

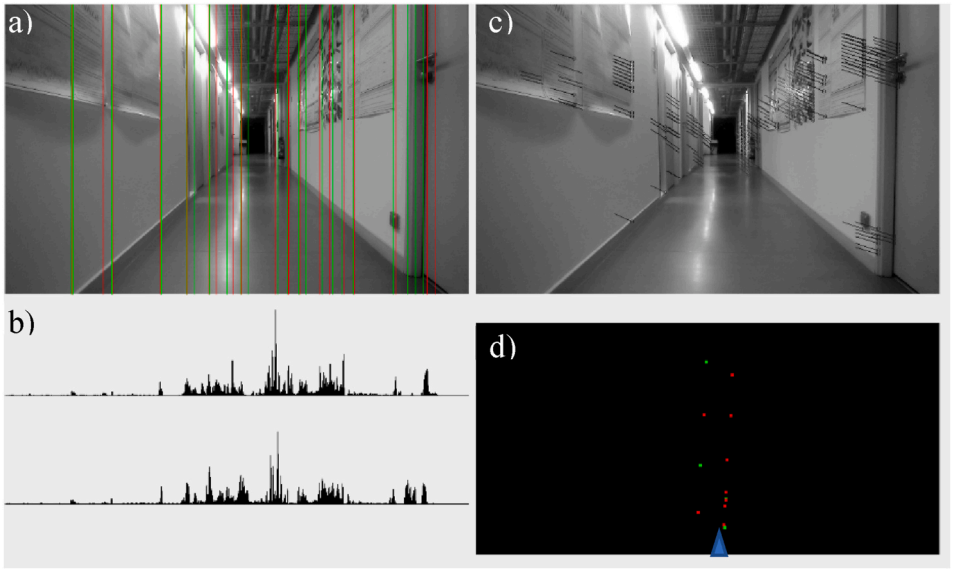


Fig. 6. Image processing. a) Image of the left camera. b) Global light gradient of each column of image (see (3) and (4)). Top histogram: positive gradients (dark to light); bottom histogram: negative gradients (light to dark). Local maximums define vertical structures in the scene: in (a) image, red lines show dark to light lines, and green lines show light to dark lines. c) Left image with points of interests and disparity with right image. The vertical offset is due to a misalignment between cameras. Filtered points are not shown. These disparities allow computing positions of vertical lines in space. d) Projections of vertical lines on navigation plane (floor), here displayed in Cartesian coordinate, defining the context C^t.

range of [−10°,10°] around current estimated orientation, limiting the linear movement to one GC spacing, and angular movement to 10° per frame.

3.5. Place cell recognition mechanism

The PCs' recognition mechanism detects when moving in the vicinity of the position where a PC was created, allowing to detect an already known place and thus close a loop in the navigation graph. This mechanism thus must be highly reliable to avoid false positive detection. To increase its trustworthiness, we propose to use ORB (Oriented FAST and Rotated BRIEF) descriptors (Rublee et al. 2011) as additional and more discriminative types of points of interest than vertical lines. A ORB descriptor defines a rotation invariant description of a point and its surrounding pixels, making it possible to record points of interests and recognize them in a scene.

When a new PC is created, a set of ORB descriptor points are detected in the camera image (using OpenCV's *ORB.detect* and *ORB.compute* functions), and localized in ego-centric space using stereovision, in the same way than for vertical line's points (see Section 3.3). The PC thus records a set of points of interests under the form of *(O, θ, d)* where *O* is the vector of values defining the descriptor, and θ and d are the orientation and distance of the descriptor in polar coordinates. As the detection function only defines a bounded number of ORB points and selects points with best properties for recognition, which can lead to a set of points covering a unique part of the image, we propose to improve the distribution of ORB points using multiple detections applied on different vertical sectors of the image. In the current implementation, we used three *detection sectors* for left, center and right parts of the image. Each sector covers a third of the image (width of 426 pixels).

The recognition mechanism is used when moving out of the GC module (i.e. reaching a border GC). In this case, ORB descriptors are detected and localized, and integrated in the current context C^t. This context is compared with each PC's descriptor set C_P. The recognition starts by comparing detected descriptors with a PC's descriptor set. We used the *BFMatcher.match* function from OpenCV to obtain a list *M* of best matching couples of descriptors, and couples with insufficient similarity are removed from the list (this filter uses a threshold of 50 on the matching values, defining the dissimilarity between descriptors, provided by the *match* function). To ensure an efficient place recognition, the recognized points must be well distributed in the scene. We thus propose that the PC can be considered as candidate if a sufficient number of descriptors is recognized in each sector, ensuring that descriptors are sufficiently numerous and distributed.

The next step detects for the coherence of position of detected points of interest: if it is possible to have multiple descriptors detected by a wrong PC due to local visual similarities, it is unlikely to have a majority of points recognized over the whole scene with a consistent position offset. We thus propose to use the average difference of distance $\Delta d = de_p - de^t$ between pairs of matching PC's descriptors (e_P) and C^t's descriptors (e^t), in each sector. Indeed, in the right PC's position, the variations of distance of points of the same sector are expected to be similar. This solution was preferred over orientation, as orientation variation of points depends of their distance. We use a histogram for each sector to estimate the coherence of distance variations. These histograms accumulate points that have the same distance variations. To make the system more tolerant to small variations due to the distribution of points on the image,

we propose to fill the histogram on a certain range k around a distance variation value, as defined in Eq. (8):

$$h_i = \sum_{(e^t,e_P)_j \in M^t, \left|de_j^t - de_{P,j} - i\right| < k} \sigma\left(de_j^t - de_{P,j} - i\right) \tag{8}$$

with σ a Gaussian function centered on 0. Candidate PCs are the filtered based on the maximums of histograms in each sector, using a threshold value, ensuring that there are both a sufficient amount of points of interests, and these points are coherent in their positions in space. This principle is illustrated in Fig. 7.

Once the list of candidate PC is obtained, equation (1) can be used to define the PCs' responses. If the PC with the greatest response is higher than a threshold, then the place's scene is recognized by this PC.

A last step is applied to validate the place recognition: the PC's context is loaded in the grid cell module, that is computed. If the estimated position is not on a border grid cell (which would imply that the system is still too far from the PC's position), then the recognized PC can be connected to current PC, closing a loop in the navigation graph. Overwise, the initial set of ORB points of interest is recorded by a newly created PC.

3.6. Adaptation of the place cell detection mechanism

The fact that each point of interest from a PC's context C_{Pi} can be paired with at most one point from the environmental context C^t brings several simplifications on the PC recognition. Indeed, the relative orientation is not required as this information is mainly used to discriminate points of interest of the same type. It is thus possible to simplify the PC recognition equation Eq. (1) as Eq. (9):

$$P(C_{Pi}, C^t) = \sum_{(a,b)_k \in M^t, a_k \in C_{P_i}, b_k \in C^t} f^P(da_k - db_k) \tag{9}$$

Moreover, the average distance variation measured in the sectors can be used to estimate roughly (but sufficiently) the position from a PC's center: the front sector provides a measure of position on Y axis while the left and right sectors provide a measure of the position on X axis (Eq. (10)). This approximated position can be used to filter candidate PCs for place recognition. We thus define three *recognition sectors*, *Sf*, *Sr* and *Sl*, corresponding to front, right and left sectors. These recognition sectors are independent from the *detections sectors* (described in Section 3.5) and can overlap: a point of interest can belong to more than one recognition sector.

$$P_Y(C_{Pi}, C^t) = \underset{(a,b)_k \in M^t, a_k \in C_{P_i}, b_k \in C^t, b_k \in S_f}{\text{average}} (da_k - db_k) \tag{10}$$

$$P_X(C_{Pi}, C^t) = \underset{(a,b)_k \in M^t, a_k \in C_{P_i}, b_k \in C^t, b_k \in S_l}{\text{average}} (da_k - db_k) \quad - \underset{(a,b)_k \in M^t, a_k \in C_{P_i}, b_k \in C^t, b_k \in S_r}{average} (da_k - db_k)$$

Values P_X and P_Y can then be used to filter candidate PCs where $|P_X|$ or $|P_Y|$ exceeds the GC's module coverage.

4. Evaluation of the navigation model in a real environment

The navigation model was tested and evaluated in an indoor environment. We selected two environments-tests: a straight corridor, and a building floor allowing to perform a complete loop. The former allows to evaluate the visual odometry generated by the navigation system, the robustness and the performances of the model. The later allows to estimate the motion drift when returning to the

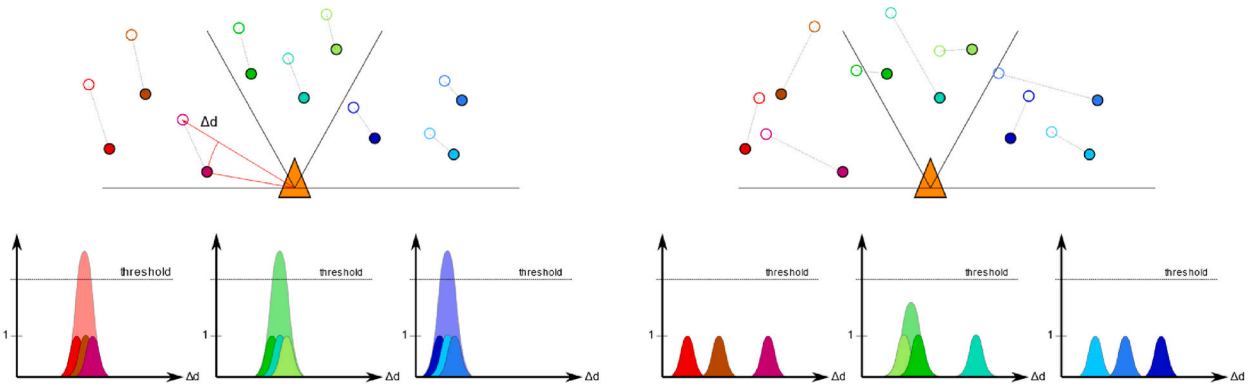


Fig. 7. Detection of candidate Place Cells. Left: nine points of interests (colored discs) are recognized from a PC's context (colored circles). In each of the three sectors, the variations in distance differences between observed and recorded point is low, leading to high values in the cumulative histograms. As the value is greater than the detection threshold in each sector, the PC can be considered as a candidate for recognition. Right: despite also having 9 recognized points of interest, the variations in distance differences does not allow a sufficient accumulation in histograms. The PC is thus filtered from candidate PCs.

starting position and test the PC recognition mechanism.

4.1. Test system and model parameters

The camera and laptop were mounted on a trolley, allowing to monitor the system while maintaining the camera at constant pan and tilt. The system was tested on a i5-10210U CPU, and runs in mono-thread configuration. The framerate is limited to 10 frames per second to reduce battery consumption (see Section 4.4 for more details on performances).

The GC module is composed of a square matrix of 11 × 11 GCs, with a spacing set to 20 cm, making the module covering a square of 220 cm by 220 cm. The system contains a set of 360 HDC, defining a resolution of 1°. Estimated position and orientation are interpolated from output values of most active GC and HDC to increase the precision.

Cartesian coordinates of points of interest in contexts are converted into polar coordinates. As precision of position decreases with distance, the distances are modified using a hyperbolic tangent function, making close points more discriminative than distant elements, and bounding the maximum distance value. The distance of a point of interest p_i (from camera) is then defined as Eq. (11):

$$\mathrm{d}_{\mathrm{i}}^{'} = \mathrm{d}_{\max} \times \tanh\left(\frac{d_i}{\varphi}\right) \tag{11}$$

Where $d_{\max}$ is the bounding distance value (corresponding to an infinite distance in the environment), and φ is a coefficient defining the distribution of distances in the $[0, d_{\max}[$ interval. We use a maximum value of $d_{\max} = 200$, and a coefficient $\varphi = 800$, thus bounding the distance values to the $[0, 200[$ interval, with a value of 100 indicating a distance of about 4.4m in the environment. The GC comparison function uses a bounded linear function (Eq. 12):

$$\mathrm{G}(\mathrm{C}_{\mathrm{Gi}}, \mathrm{C}_{H_\varphi}) = \sum_{a \in \mathrm{C}_{\mathrm{G_i}}, b \in \mathrm{C}_{H_\varphi}} \mathrm{id}^{'}(a,b) \times \max\left(0, 1 - \mathrm{d}^{'}(a,b) \;/\; \beta\right) \tag{12}$$

where β defines the radius of the receptive field of GCs. We use a value of β = 11 (thus defining a receptive field with a diameter of 22 cm), making GC's receptive field slightly larger than the GC spacing (20 cm), allowing neighboring receptive fields to overlap slightly.

To analyze the graph of PC, we proposed an observation tool, called *global grid*, which consists of the module of GCs repeating itself indefinitely. This grid integrates the movements measured by the module, allowing to observe the trajectory and project the PC graph in space. Note that the navigation system cannot access the global grid. The videos of all experiments presented in this section are available at https://gaysimon.github.io/projects/navig_camera_en.html.

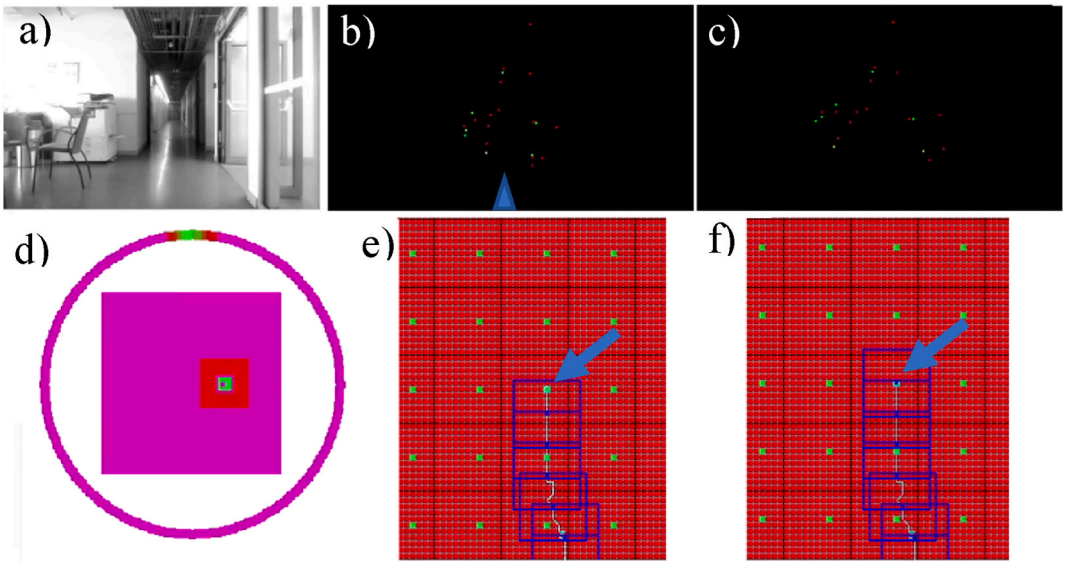


Fig. 8. Creation of a new Place Cell. a) Input image (left camera). b) Environment context of vertical lines in Cartesian reference (top-view projection). Red points indicate dark to light lines, and green points indicate light to dark lines. c) Environment context represented in polar reference (used by the navigation system). d) The module of 11 × 11 GCs (center square) and the 360 HDC (external ring). Green indicates cells with high activity, giving the position and orientation around current place cell. Magenta cells are not computed. e) The *global grid* is an observation tool that repeats the module indefinitely, and integrates movements measured by the GC module. This allows to project the PC graph (blue points) and trajectory (cyan line) in space. The blue squares show the GC module centered on a PC when active, defining the limits of the module's coverage (the navigation system cannot access this global grid). f) When reaching the "border" of the module, a new PC is created and connected to the reached GC, allowing to continue tracking the position.

4.2. Construction of a navigation model

The construction of the navigation graph starts from a preselected position in the environment. When the navigation system is initialized, a first PC is created and associated with a GC of the module (by default, the GC at the center of the module). Then, the trolley is moved along a path. Once the destination point is reached, the navigation model is recorded to test guiding possibilities (c.f. Section 4.3). The PC recognition system is here deactivated (see Section 4.4 for tests of this mechanism).

Corridor environment. The system starts on a side of the corridor. The trolley is then moved in straight line to the other end of the corridor. The covered distance is 44.9 m. While navigating, we can observe the system estimating position and orientation around the current PC using GCs and HDCs' activity. When reaching a "border" grid cell, a new PC is created, allowing to continue tracking the position (cf. Fig. 8).

While covering the path, the system created a sequence of 46 PCs. By considering the distance between the GCs associated to two consecutive PCs, the distance between starting and ending points measured by the navigation system is 45.16 m (relative error of 0.58%). Fig. 9 provides collected results. The measured distance covered by the trolley is 45.49m (slightly higher as the trolley's trajectory is not a perfect straight line). The sequence of PCs is not aligned with the GC module because the camera was not perfectly aligned with the corridor when initializing the first PC.

Floor environment. The system starts from a certain point of the building floor. Then, a complete tour of the floor is performed, and stops at the starting point. The complete tour is about 96 m long.

The system created 90 PCs during this tour (cf. Fig. 10). By considering the distance between two consecutive PCs on the GC module, the measured covered distance is 97.27m. The estimated orientation at the end is 30°, and the position estimation is 9.57m far from true position. This estimation error is mainly due to error in orientation estimation, that "open" the graph loop.

We can note that this motion drift is not important in this model: when using a PC recognition system, the recognition of an already visited place would make the system simply connecting the PCs by considering the local offset between the two PCs observed on the module when changing active PC, without requiring to change position of PCs of the graph on the GC module (Gay et al., 2021a). This principle is evaluated in Section 4.4.

4.3. Following a path in a navigation model

The navigation graph is loaded and the trolley is placed at the same starting position than for graph construction. The system's displayer indicates the estimated orientation and position around the current PC, but also the position of the next PC of the graph and its direction from current position (cf. Fig. 11). Thus, it is possible to reach the next PC by aligning the system on the provided direction and moving forward. When approaching the next PC, it becomes the new active PC, allowing to continue to follow the path by moving toward the new next PC. After following the whole sequence of PCs, the camera is at the position of the last created PC during graph construction. The guidance could be performed in both corridor and floor environment.

We can note that, despite the redundancy in the corridor environment, the system is still able to track the position, due to the sequential aspect of the PC graph. The navigation system can also handle and track lateral movements during guidance (cf. Fig. 12). It is thus possible to deviate slightly from the recorded path, while remaining at GC module range, making possible to track and guide a pedestrian even when deviating.

4.4. Test of the place cell recognition mechanism

The PC recognition mechanism was tested in the two environments: the recognition principle was evaluated in the corridor environment. Then, the whole navigation system was tested in real conditions in the floor environment.

Corridor environment. The recognition mechanism was implemented on the navigation system. The system defines three

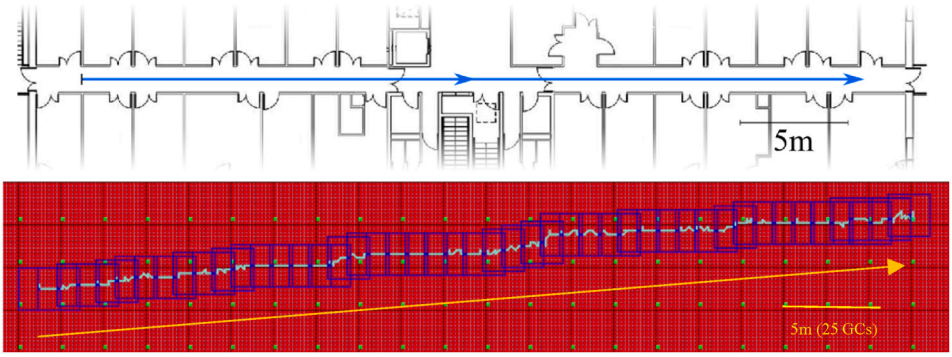


Fig. 9. Mapping of the corridor environment (top map). The path is 44.9m long (blue arrow). Bottom: PCs are represented on the global grid with blue points. The path is composed of 46 PCs, with a measured length of 45.16m (yellow arrow). The cyan line joining PCs shows the trajectory of the camera in the model.

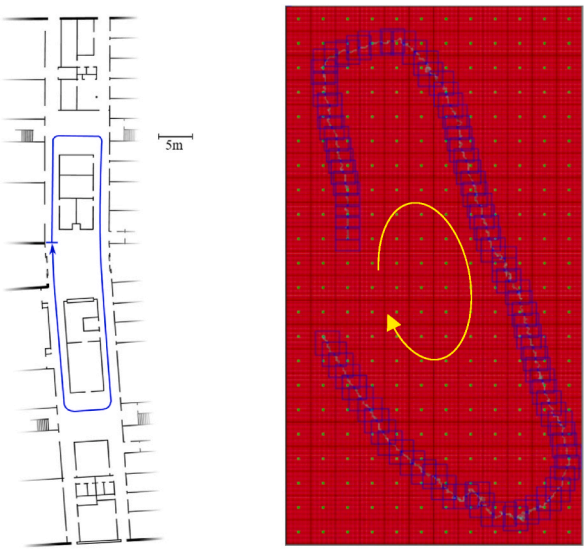


Fig. 10. Mapping of the floor environment (left map). The path (blue curve) is 96m long and returns to the starting position. Left: graph of PCs projected on the global grid. The graph is composed of 90 PCs, with a measured length of 97.27m. The offset error between first and last PCs is 9.57m.

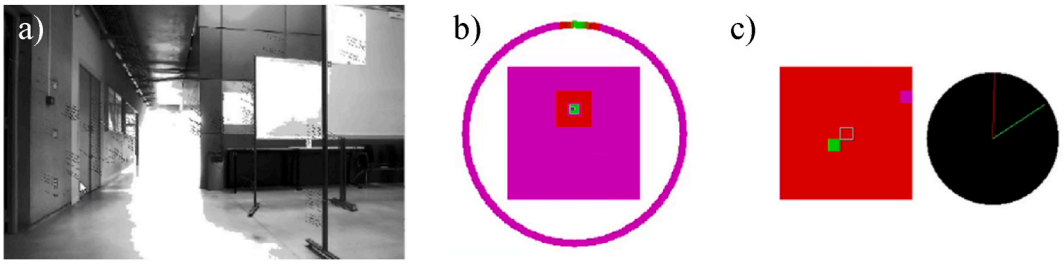


Fig. 11. Guidance and navigation data. a) Input image (position just before first turn in floor environment). b) Module of GCs and HDCs indicating the position and orientation in current PC's reference. c) Guidance display: left part is the GC module shifted to place the PC on the center GC (blue square), giving the current position (most active GC in green) in a more comprehensive way for an external observer, but also the position of the next PC in the sequence (magenta square). The compass on the right gives the orientation of the camera (red line) and the orientation of the next PC (green line). Thus, by aligning the camera orientation with the next PC orientation, and moving forward, it is possible to reach next PC, and then to cover the sequence of PCs.

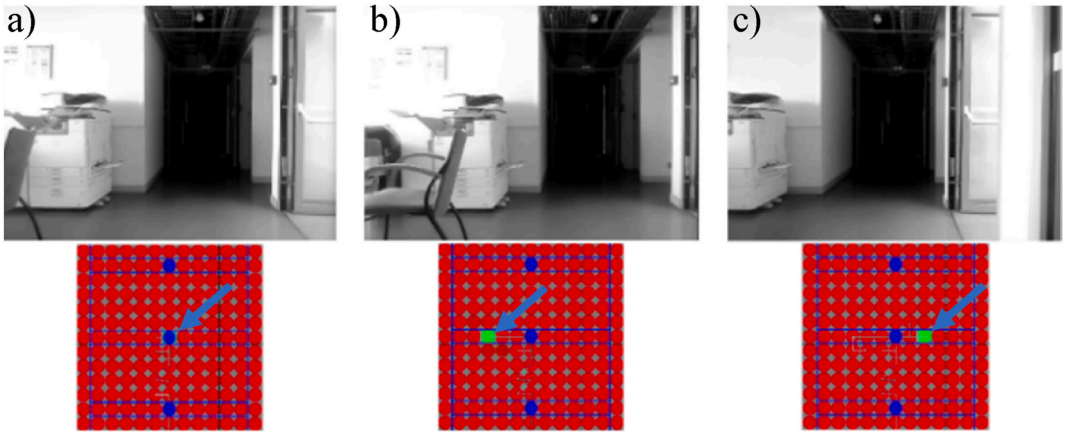


Fig. 12. Measures of lateral movements. a) Same alignment than for graph construction. b) Lateral movement of 60 cm on left (equivalent to 3 grid cells). c) Lateral movement of 40 cm on right (equivalent to 2 grid cells).

overlapping sectors: left and right sectors cover respectively the left and right half of the image (nearly 35°). The front sector covers a sector of 40° in front of the image (thus overlapping both left and right sectors). A minimum of 20 points of interests must be recognized in each sector to enable the recognition of the PC. The PCs' contexts are evaluated on each frame, which allows to observe the evolution of point of interest recognition and position coherence while moving in the vicinity of a PC. We first evaluate the point of interest coherence and the position estimation systems. A first PC is created in the center part of the environment. Then the trolley is moved laterally (without moving out of the GC module). Fig. 13 shows the evolution of histograms when moving on the left and on the right from the initial position: when moving away from the center, the histogram of sectors separate quickly, although each histogram spreads more slowly, and thus decreases slowly. This property makes possible to detect a PC when moving in its vicinity. We also evaluated the estimated position (10). These estimations are unprecise, and the movement on X axis is often underestimated, mainly due to the low visual field of the camera. However, the estimation errors are at most of one GC spacing (20 cm), making possible to use them to define candidate PCs and the nearest candidate PC from current position. In the following experiments, we define a distance threshold of 1 m from the PC's center, making the detection area slightly smaller than the GC coverage, and excluding "border" GCs.

We then test the PC recognition: the system is initialized in a side of the corridor, and the trolley is pulled forward to create a sequence of three PCs, that we note P_1, P_2 and P_3. (c.f. Fig. 14). Then, it is pulled backward to cover the position of the three PCs. From the position of P_3, while approaching P_2, we can observe the distance of points of interest converging, in each sector, to similar distance values. When reaching the GC module's border, centered on P_3, points of interest of P_2 converge the most to the same distance (in each sector), making P_2 recognized as current PC. P_3 is then associated as a neighbor of P_2, enabling a two-ways connection between these PCs. The process is then repeated when reaching the border of GC module centered P_2: P_1 is recognized, becomes the new active PC and a two-way connection is created between P_1 and P_2. The estimated position is also monitored. It appears that these estimations, despite to being imprecise for local navigation, are sufficient to confirm or infirm the proximity of a PC: while approaching the module's border, only one PC estimates a position within the module's range, confirming the trustworthiness of this estimation as filter for candidate PCs.

Floor environment. The recognition mechanism is then tested in the floor environment in order to evaluate the possibility to close a loop in the navigation graph. As the navigation graph is composed of a large number of PCs, their contexts will only be evaluated when reaching the GC module's border (i.e. before creating a new PC). In Fig. 15, the navigation system starts at the same position than previously (Section 4.2) and a complete tour of the floor is performed, while the system constructs the navigation graph. The performed path was slightly larger than in Section 4.2, and has an estimated length of around 100m. While performing the tour of the

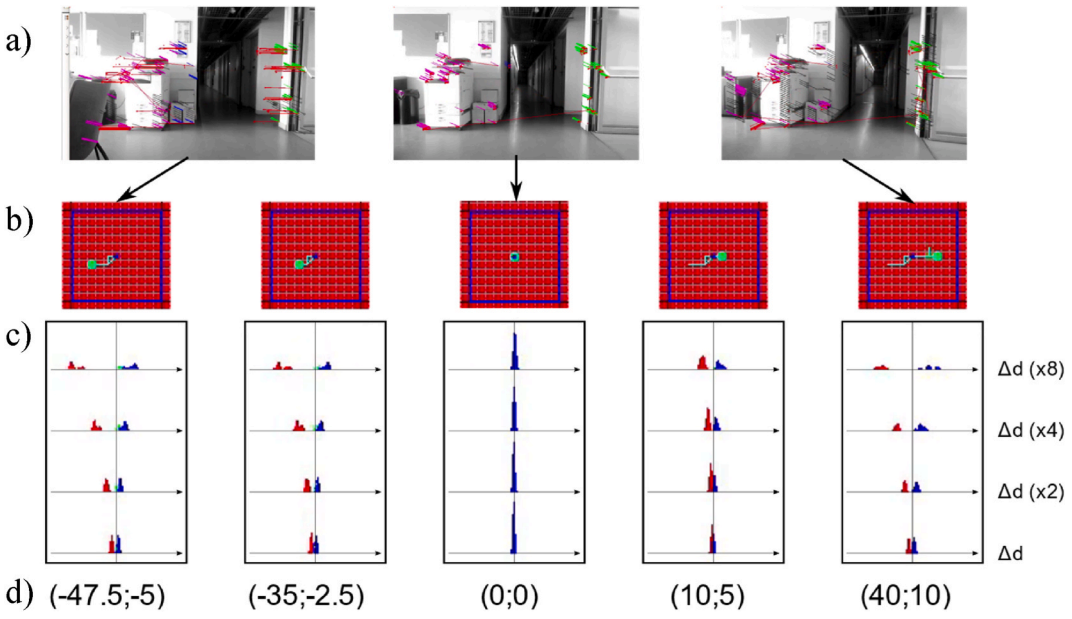


Fig. 13. Coherence of points of interests. a) Point of view of the camera. Red lines show pairs of recognized ORB points of interest from scene and PC's context. Mauve and green line show ORB points from the scene respectively from left and right sectors, and their stereo disparity. From the initial position (center), the trolley is moved to left then to right side. b) Estimated position of the camera (green point) in the GC module centered on the current PC (blue point). The GC spacing (distance between two GC) is 20 cm. c) Histograms of the three sectors (left: blue, front: green, right: red). The bottom line displays the histograms used for detection. Upper lines apply a coefficient on Δd, without changing the gaussian function (Eq. 8), to better observe the histogram dispersion. As the trolley moves left, points from left side become closer, while points in right sector move farther, and inversely when moving to right side. Thus, the left and right histograms separate quickly, while individual histograms decrease more progressively, remaining with a sufficient value to trigger the recognition of the PC. d) Estimated position (in cm) from PC's position. Although these predictions are unprecise, they are sufficient to detect whether the position is within a module coverage and which PC is the closest from current position.

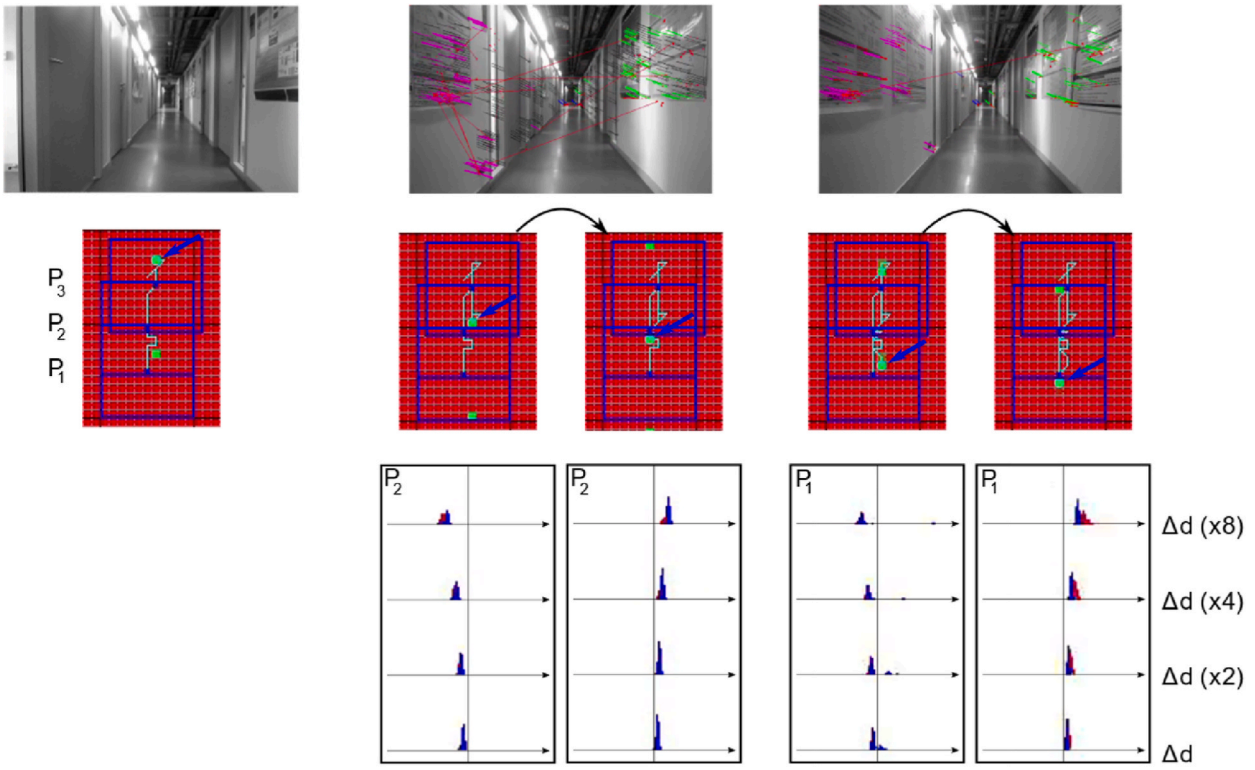


Fig. 14. Place Cell recognition mechanism. Left: the trolley is moved forward to create three PCs, named P_1, P_2 and P_3. Then, the trolley is moved backward. When approaching P_2 (middle), a sufficient number of points of interest are recognized to promote it as a candidate. The estimated position also confirms that a module centered on P_2 can cover current position. When reaching a border GC of the module centered on P_3, P_2 is the most probable candidate, and its POI coherence histograms are greater than detection threshold. Thus, instead of creating a new PC, P_2 becomes the new active PC and is connected to P_3, enabling a two-way connection. Also, the middle image shows that only a few points of interest from P1 are recognized, and with insufficient position coherence (red points and lines), preventing P1 from being considered as candidate. Then, when approaching P_1 (right), the process is repeated between P_2 and P_1: a sufficient number of POI of P_1 are recognized, and, when reaching the border of module centered on P_2, P_1 becomes the new active PC, and is connected with P_2 with a two-way connection.

environment, the system created a sequence of 99 PCs, and the measured covered distance (considering the movement from a PC to the next one) is 104.15m.

When reaching the starting point, a motion drift appears, both in position and orientation: the position estimation is 6.45m from the actual position. The orientation error remains low, although it is mainly due to errors canceling out. However, the first PC is recognized, and the last PC of the graph is connected as a predecessor of the first one. As the first PC is loaded in the GC's module, it becomes the new reference, indicating the position of the next PC in the navigation graph. Once the position and orientation are obtained in the reference of the first PC, it becomes possible to follow the sequence of PC, in a similar way than previously (c.f. Section 4.3). We can note that despite the motion drift appearing between first and last PC, the estimated orientation and position on the module remains consistent through this reference change. Moreover, the last PC records the position of first PC in its own reference (i.e. position defined by the reached "border" GC), enabling the possibility to guide the system from last to first PC. This observation confirms that the model is not affected by motion drift.

4.5. Robustness and performances

Framerate. We measured the execution time both for the vision system and for the navigation system on the test laptop (i5-10210U CPU), and the camera set to 60fps, which limits the motion blur. The vision system requires 16–30ms to operate, depending on the number of vertical lines in the scene. The navigation system requires less than 2ms to operate, thanks to the simplification proposed in Section 3.4 (the system needs 25–30ms without this simplification). The global system can thus operate in less than 32ms (with an average of 25ms in test environments), making possible to run at least at 30 frames per seconds.

The recognition mechanism has a low influence on performances, as it is only used when reaching the GC's border during the exploration of unknown part of the environment. However, while the number of PC increases, an increasing latency can be observed when creating a new PC, due to the evaluation of existing PC's contexts. Section 6 discusses on additional filter mechanisms that will be investigated to reduce the number of candidate PCs.

Robustness to tilt and roll. We tested the robustness of the system to camera movements. Due to the used points of interests, made

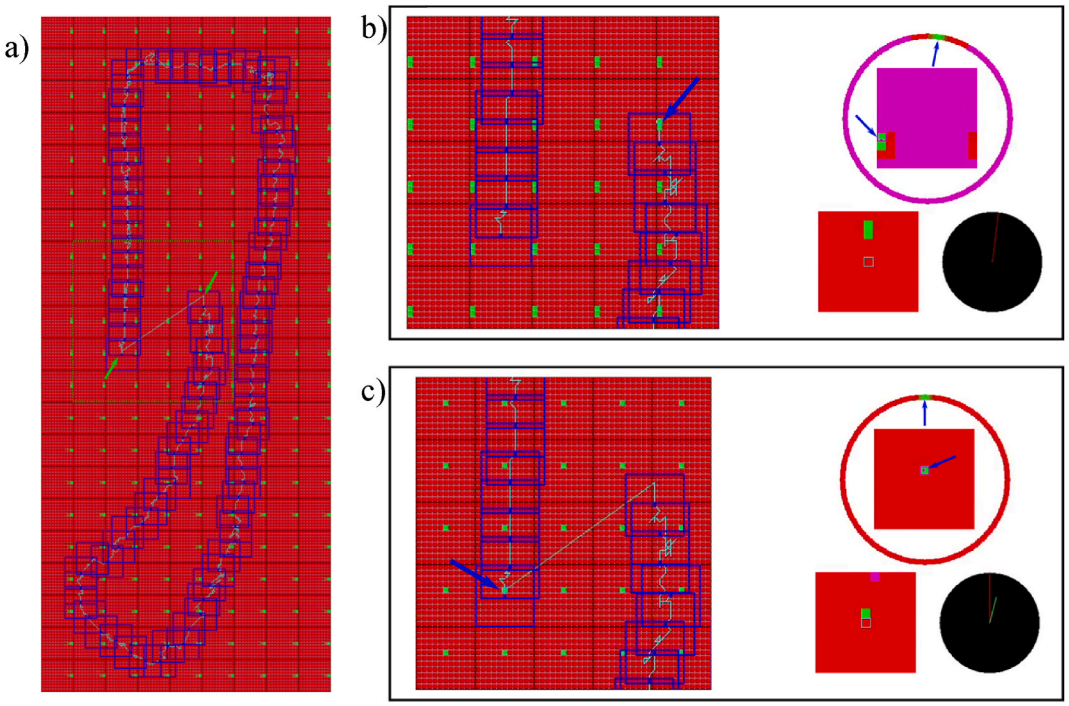


Fig. 15. Graph construction with Place Cell recognition. a) While performing a tour of the floor environment, the navigation system creates a sequence of 99 PCs, with a measured length of 104.15m and an offset error of 6.45m between first and last PCs (green arrows). b) Estimated position and orientation of the trolley in the reference of last PC (blue arrows), a frame before reaching a border GC of the module. c) When reaching a border GC of the module, the navigation system recognizes the first PC of the graph, which becomes the new active PC. Then, the position and orientation are estimated using all GCs and HDCs. Once position and orientation are obtained in the first PC's reference (blue arrows), it becomes possible to perform the tour again by following the navigation data below: pink GC indicate the position of the next PC (module is shifted to place current PC on the center). In the black compass, the green line indicates the orientation of next PC while red line indicates the current estimated orientation. Thus, by aligning the trolley towards the next PC, it is possible to perform the sequence, as shown in Section4.3. As the last PC defines the position of the first PC in its own local reference, the tour can be performed while ignoring the error offset.

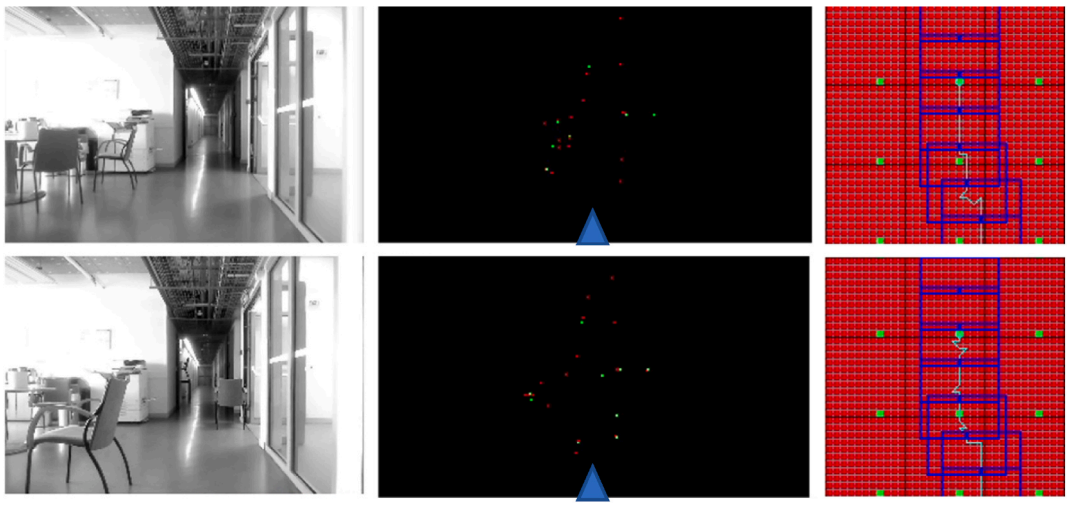


Fig. 16. Guidance mode in original (top) and modified (bottom) environment. Several furniture pieces were added or shifted, to change the configuration of the environment, and thus, the environmental context (center, here in Cartesian reference). Although a large number of points of interests are removed, added or shifted, a sufficient amount of points remains close to their expected position, allowing to track the position of the camera (right).

of vertical lines, the system is sensitive to roll movements. The system can however tolerate rolls of $\pm 10^{\circ}$. The tolerance to tilt movement depends more on the distance of points of interests, because points from ceiling and floor are eliminated based on their measured height from the navigation plane, relative to the camera tilt. In the corridor environment, the system was successfully tested with tilts up to $\pm 15^{\circ}$. This low tolerance makes the system suitable for mobile systems (e.g. robots). Additional mechanisms must be implemented to makes the model more reliable for wearable assistance devices.

Robustness to environment changes. The guidance system was tested in a modified corridor environment to test its robustness. Several furniture was added or shifted from mapping configuration (Section 4.2), as shown in Fig. 16. These changes have no significant incidences on the tracking system. Indeed, as the model looks for points of interests that are close to their expected positions, the system is not affected by these changes while at least half of visible points of interests remains unchanged.

4.6. Toward an assistive wearable device

We evaluated the possibility to use this navigation model as a wearable assistance device: the camera was attached to the front strap of a backpack (see Fig. 17), while the laptop remains on the trolley, allowing to read the direction to reach next PC in the navigation graph. Due to the low tolerance to tilt and roll, it is necessary for the person to remains to stand in straight upright position. Then, by following the direction provided by the navigation system, the corridor path could be performed.

5. Evaluation of the navigation system on an autonomous embedded system

Our navigation model aims the development of navigation systems to assists the navigation of both visually impaired people and autonomous embedded systems. These applications require 1) a system that can run on portative or embedded systems with limited computational resources, and 2) that can provide reliable and efficient navigation data to guide the person or the autonomous system. We thus propose to evaluate our model on a robotic platform, equipped with a small single-board battery-powered computer. This small computer has limited resources, but is small and light enough to be integrated in a portative distance. The robotic platform has no other sensor than the camera, and is only guided by the navigation system, allowing to evaluate the reliability of navigation instructions that it provides. Also, the single-board computer controls the platform motors through an Arduino board, and provides a minimalist web server allowing the experimenter to observe, from a distant web browser, the navigation model's properties and constructed map, and control the execution of the model running on the single-board computer. The architecture of the robotic platform is shown in Fig. 18.

5.1. Embedded experimental platform: hardware

The embedded system (Fig. 19) is built around a small single-board computer. The main constraints were the presence of a USB-3 port, allowing to connect the stereo camera, and a low voltage and low consumption, allowing to power it with light battery. We thus select the *Banana Pi M5* single-board computer, that have four USB-3 ports and runs at 5V with a maximum current of 3A. This single-board computer is equipped with a Cortex-A55 quad-core processor clocked at 2Ghz, and 4 GB of RAM. This board has similar specifications to a Raspberry PI 4 board, that can also be used instead. The computer board is equipped with a WIFI dongle to communicate with another computer during the development and tests. The system is powered with a 20A.h powerbank. An Arduino board is added as an interface with the robotic platform's actuators. The complete system weight less than 600g.

The System is mounted on a robotic platform (Mecanum Wheel chassis from Osoyoo[1]), shown in Fig. 20. The platform is equipped with an Arduino Mega board, motor driver boards, four motors, four mecanum wheels, and a battery to power the motors. The mecanum wheels allows the robot to move in any directions (including lateral translations), making possible to simulate movements of a pedestrian.

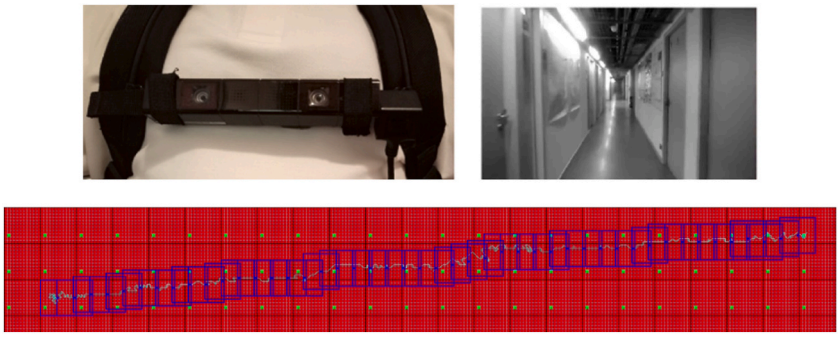


Fig. 17. The camera is attached to the front strap of a backpack. Even though the image is less stable than with the trolley, it was possible to follow the recorded path (bottom).

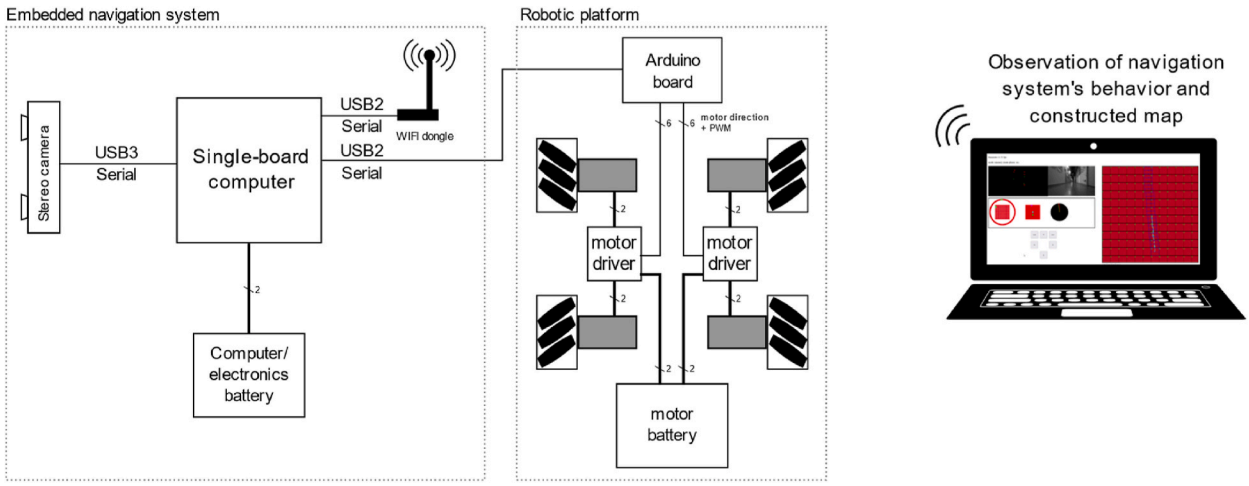


Fig. 18. Architecture of the robotic platform. The navigation system is embedded on a single-board computer, and provide a direction information to follow a previously recorded path. An Arduino board is used as interface between the computer and motors, converting motion commands into motor signals. The computer also provides a simple web server providing an interface allowing an experimenter to monitor and control the execution of the navigation model through a WIFI connection.

Fig. 19. The complete embedded system. From left to right: a Banana Pi M5 single-board computer equipped with a WIFI dongle and antenna, the stereo camera, and a 20A.h powerbank.

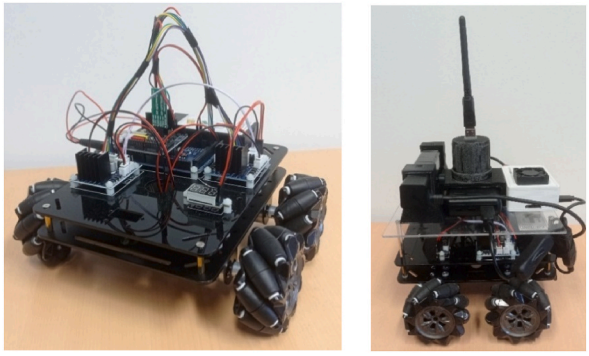


Fig. 20. Robotic mobile platform. Left, the platform base, equipped with four DC motors and mecanum wheels, two motor driver boards (each controlling two motors), and an Arduino Mega board, here equipped with a WIFI/Bluetooth shield (not used in our experiments). Right: the platform equipped with the embedded system. The elements are placed on the top level using 3D printed supports (from left to right: stereo camera, battery and WIFI dongle, and single-board computer).

5.2. Embedded experimental platform: software

The single-board computer runs under a Linux system (Raspbian, Raspberry Pi OS), allowing to directly port the system's software developed and used on the laptop (Section 4.1). Several modules must however be added to operate the robot. First, we added a communication module to communicate with the robotic platform through the Arduino over serial USB port. This module sends motor commands under the form of a triplet (v_x, v_y, r_θ) giving the translation vector and rotation to perform. On the other side, the Arduino software receives these triplets and converts them into motor control command $(V_{l1}, V_{r1}, V_{l2}, V_{r2})$, allowing to move the robotic platform in the desired direction. A web interface was developed to monitor and control the system from a remote computer: a minimal web server provides a single webpage interacting with the system using web sockets. The system receives inputs from users, and sends the observed image, content of the environmental context and values of grid cells, head direction cells and global grid, allowing Javascripts to draw these structures on the interface, shown in Fig. 21.

Finally, we added a simple control system to pilot the robot in automatic guidance mode. This control system uses the position of the next PC on the navigation graph, and compute its orientation in the robot's reference. The robot moves forward continuously. When the orientation of the next PC is greater than a threshold (a threshold of 5° was used in the following experiments), either in left or right direction, the system adds a rotation movement in the other side. As the navigation system changes the current PC when approaching the next PC, the robot follows the sequence of PC. This simple control rule makes the robot moving through a "zig-zag" movement allowing to test the localization system when following a path that is slightly different from path used to map the environment.

5.3. Test in the corridor environment

The robotic platform was evaluated in the corridor environment. We used the environment model created with the trolley. The robot starts from the same position than the trolley when testing the guidance mode. The robot is then set in automatic mode, and move through the corridor, guided by its guidance mechanism. Despite the difference in camera height, the robot successfully crossed the first section of the corridor (18 first place cells). It however stops in the middle hall, where a large amount of lines recorded by the environment model were not visible from the ground. Fig. 22 shows the robot's estimated movements. As the robot always moves toward the next place cell, its trajectory passes through the sequence of PCs, even though the zig-zag motion trajectory can still be observed.

The model thus shows that the navigation data that it provides can accurately guide an autonomous system through a predefined

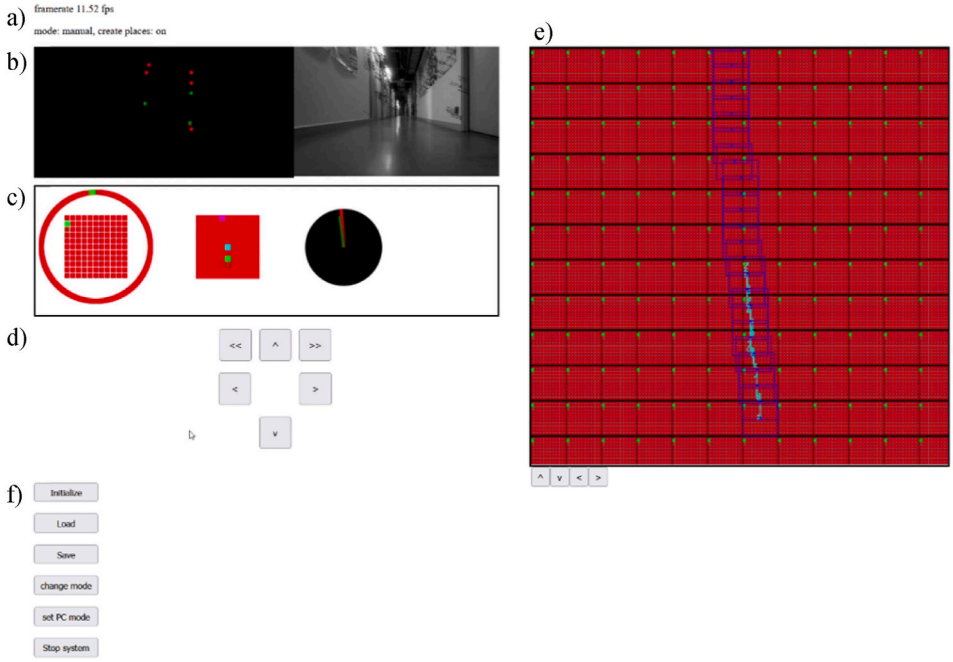


Fig. 21. The web interface, used to monitor and control the embeded system. a) Information about the system: current framerate, control mode (manual or automatic guidance), possibility to add new place cells (*on* or *off*). b) environmental context (Cartesian coordinates) and image from camera. c) Module of grid cells and head direction cells. The middle representation shifts the module to place the PC (cyan square) on the center, and display the position (green square) and connected PCs (if any) as violet squares. The right representation shows the current estimated orientation (red line) and orientation of next PC in the graph (green line). d) Buttons to control the robot (manual mode only). e) Global grid representation. Buttons allow to shift the grid. f) Buttons to control the execution of the system.

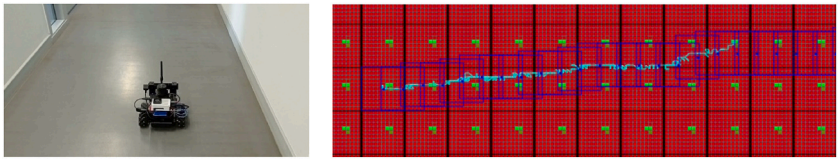


Fig. 22. Robot guidance in the corridor environment. Left: the robot traveling the corridor environment, using the navigation graph created with the trolley. The robot uses a rudimentary control system that aligns with the direction of the next PC. Right: The robot successfully navigates through the 18 first PCs, until facing a context looking too different due to the difference in camera height.

path. The tracking loss appearing when the visible points of interests showed that additional mechanisms, such as IMU, must be added to make the system more robust to environments with insufficient visible points of interests.

While moving in the corridor environment, the system can operate in 80–125ms on a Banana Pi M5, depending on the number of visible vertical lines. The system thus runs at 8 to 12 frames per seconds. We measured a consumption between 1 and 1.1A (5 to 5.5W) while running the navigation system (from an IDLE consumption of 0.45–0.480A, or 2.25–2.4W).

6. Conclusion and future work

We proposed an implementation of a bio-inspired navigation model for a real-world environment, using a stereo camera, and demonstrated the feasibility and pertinence of this model to allow the tracking and the guiding of a person or an autonomous system in an unknown environment. Our tests showed the possibility to map a path in an initially unknown environment, and to perform this path by following the constructed sequence of place cells, guided by the local navigation system Seeing Eye (grid cells and head-direction cells).

The structure of the environment model, under the form of a graph of local contexts, makes the navigation system robust to environment changes and redundancy, due to the sequential aspect of the model allowing to compensate for a low number of visual cues. The system is also highly tolerant to motion drift, as it only considers the position of neighbor place cells in a local reference instead of a global map. Moreover, the local navigation system can track movements in any direction, which is an important feature to guide a pedestrian. The obtained spatial performances allow to expect its possible integration in an assistive device after some technical improvements.

In future work, we will improve the reliability and robustness of the model, especially to handle situations with noisy images or when the number of points of interest is low, and to consider pan and tilt movements of the camera that a pedestrian would produce. We will implement the PC's ORB points of interests in GC's localization system, in addition to vertical lines, to increase the number of point of interests. Then, we will add additional sensory modalities, such as an Inertial Measurement Unit (IMU) to provide additional navigation data when the number of cues becomes insufficient. Using an IMU will also allow the detection of camera tilt and roll. Also, we will improve the reliability of PC detection mechanism to avoid false positives. We propose the use of multiple modules of GCs of different scales: by using different modules repeating with different frequencies, we expect a better robustness to environment redundancies. Several improvements will be investigated to improve the PC recognition mechanism, avoiding the need to evaluate the contexts of all PCs of the graph: we intend to exploit the topography of the navigation graph to estimate most probable candidate PCs before using visual filters. Although it is not necessary, we will also study motion drift corrections, through shifting the GC and HDC associated to a PC, in order to make the navigation graph more consistent in complex environments. We will then investigate methods to convey navigation data, through sound or vibrations, and finally test our navigation system with visually impaired and in outdoor environments.

Moreover, we will investigate the possibilities to guide autonomous systems, and specifically the possibility to split the environmental model and distribute it, as demonstrated in simulated environment (Gay et al., 2021b), to control a swarm of small robots or drone with limited computational and memory resources. Strategies for efficient distribution, and redundancies will be investigated.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:Simon Gay reports financial support was provided by French National Research Agency. Simon Gay reports financial support was provided by Normandy University. Simon Gay reports financial support was provided by European Commission.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the French National Research Agency (ANR) in the frameworks of "Investissements d'avenir" (ANR-15-IDEX-02) and "Inclusive Museum Guide" (IMG, ANR-20-CE38-0007), and by the Region of Normandy and European Commission in

the frame of "Guide Muséal" (https://guide-museal.univ-rouen.fr/).

References

APH (American Printing House), https://tech.aph.org/neios/, last accessed 2024/04/26.

Blindsquare homepage, https://www.blindsquare.com/, last accessed 2024/04/26.

Campos, C., Elvira, R., Gómez Rodríguez, J. J., Montiel, J. M. M., & Tardós, J. D. (2021). ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Transactions on Robotics, 37*(6), 1874–1890.

Chen, Q., & Mo, H. (2019). A brain-inspired goal-oriented robot navigation system. *Appl. Sci., 9*(22).

Fallah, N., Apostolopoulos, I., Bekris, K., & Folmer, E. (2013). Indoor human navigation systems: A survey. *Interacting with Computers, 25*(1), 21–33.

Flores, G., & Manduchi, R. (2018). *Easy return: An app for indoor backtracking assistance* (Vol. 18, pp. 1–12). ACM CHI'.

Fusco, G., & Coughlan, J. M. (2020). Indoor localization for visually impaired travelers using computer vision on a smartphone. *Proc. ACM Web4All conf.: Automation for accessibility*.

Gaussier, P., Banquet, J. P., Cuperlier, N., Quoy, M., Aubin, L., Jacob, P.-Y., Sargolini, F., Save, E., Krichmar, J. L., & Poucet, B. (2019). Merging information in the entorhinal cortex: What can we learn from robotics experiments and modeling? *Journal of Experimental Biology, 222*, 1–13.

Gay, S. L., Le Run, K., Pissaloux, E., Romeo, K., & Lecomte, C. (2021a). Towards a predictive bio-inspired navigation model. *Journ. Information, 12*(3), 1–19.

Gay, S. L., Truong, N.-T., Pissaloux, E., & Jamont, J.-P. (2021b). Towards predictive and decentralized bio-inspired navigation models for distributed systems. *IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8.

Google Map, https://www.google.com/map, last accessed 2024/4/26.

Hafting, T., Fyhn, M., Molden, S., Moser, M. B., & Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature, 436*(7052), 801–806.

Hengle, A., Kulkarni, A., Bavadekar, N., Kulkarni, N., & Udyawar, R. (2020). Smart cap: A deep lear-ning and iot based assistant for the visually impaired. *3rd int. Conference on Smart systems and inventive technology (ICSSIT)* (pp. 1109–1116).

Karaouzene, A., Delarboulas, P., Vidal, D., Gaussier, P., Quoy, M., & Ramesh, C. (2013). Social interaction for object recognition and tracking. In *IEEE ROMAN Workshop on Develop-mental and bio-inspired approaches for social cognitive robotics*.

Liu, K., Motta, G., Dong, J., & Abu, H. I. (2017). Wi-Fi-aided magnetic field positioning with floor estimation in indoor multi-floor navigation services. *ICIOT*.

Liu, K., Motta, G., Ma, T., & Guo, T. (2016). *Multi-floor indoor navigation with geomagnetic field positioning and ant colony optimization algorithm*. SOSE.

Milford, M., & Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired slam system. *Int. J. Rob. Res., 29*(9), 1131–1153.

Milford, M. J., & Wyeth, G. F. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. *Proc. IEEE Int. Conf. Robot. Autom.*, 1643–1649.

O'Keefe, J., & Nadel, L. (1978). *The hippocampus as a cognitive map*. Clarendon Press.

Pissaloux, E., & Velazquez, R. (Eds.). (2018). *Mobility of visually impaired people: Fundamentals and ICT assistive technologies*. Springer.

Pissaloux, E., Velazquez, R., & Maingreaud, F. (2017). A new framework for cognitive mobility of visually impaired users and associated tactile device. *IEEE T-HMS (Trans. Human-Ma-chine Systems), 47*(6), 2168–2291.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). Orb: An efficient alternative to SIFT or SURF. *International Conference on Computer Vision*, 2564–2571.

Seeing Eye GPS, Sendero Group, http://www.senderogroup.com/products/SeeingEyeGPS/, last accessed 2024/04/26.

Sieuwe Elferink github repository, https://github.com/sieuwe1/PS4-eye-camera-for-linux-with-python-and-OpenCV, last accessed 2024/04/26.

Stensola, H., Stensola, T., Solstad, T., FrØland, K., Moser, M. B., & Moser, E. I. (2012). The entorhinal grid map is discretized. *Nature, 492*(7427), 72–78.

Tang, H., Yan, R., & Tan, K. C. (2018). Cognitive navigation by neuro-inspired localization, mapping, and episodic memory. *IEEE Trans. Cogn. Dev. Syst., 10*(3), 751–761.

Taube, J., Muller, R., & Ranck, J. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *Journal of Neurosciences, 10*, 420–435.

Zhou, X., Bai, T., Gao, Y., & Han, Y. (2019). Vision-based robot navigation through combining unsupervised learning and hierarchical reinforcement learning. *Sensors, 19*(7), 1–23.