Biologically Inspired Cognitive Architectures xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Biologically Inspired Cognitive Architectures

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

Research article

A novel neurophysiological based navigation system

Peter J. Zeno*, Sarosh Patel, Tarek M. Sobh

Robotics, Intelligent Sensing and Control (RISC) Lab, School of Engineering, University of Bridgeport, 221 University Avenue, Bridgeport, CT 06604, USA

ARTICLE INFO

Keywords:
Cognitive map
Hippocampus
Path integration
Path planning
FPGA

ABSTRACT

This paper introduces a novel neurophysiologically based mobile robot navigation system, which emulates the dynamics of a rodent's navigation and spatial awareness cells found in the hippocampus and entorhinal cortex. The model presented here replicates the functionality of these neurons in their hardware and software counterparts. By using data structures and computational logic that best utilizes currently available processing architectures, a cognitive map is created using a unique multimodal source model for place cell activation. Path planning is performed by using a combination of Euclidean distance path checking, goal memory, and the A* algorithm. Localization is accomplished using simple, low power sensors, such as a camera, ultrasonic sensors, motor encoders and a gyroscope. The place code data structures are initialized as the mobile robot finds goal locations and other unique locations, and are then linked as paths between goal locations, as goals are found during exploration. The place code creates a hybrid cognitive map of metric and topological data. In doing so, much less memory is needed to represent the robot's roaming environment, as compared to traditional mapping methods, such as occupancy grids. A comparison of the memory and processing savings are presented, as well as to the functional similarities of our design to the rodent's specialized navigation cells.

Introduction

Autonomous mobile robotics have many diverse applications and domains (i.e., indoor, outdoor, underwater, and airborne). For instance, indoor applications include: security, rescue, and service mobile robots, while outdoor applications include driverless automobiles. Underwater and airborne robot systems include ocean and space exploration robots, respectively. The success of any autonomous mobile robot is based on its ability to reliably navigate in its environment. This is especially true for animals and other living creatures, whose survivability is dependent on their ability to navigate effectively in their environment. They would perish if they were unable to relocate food and cache locations, their home, as well as shelter spots from predators. Navigation, for both biological creatures and machines, can be defined as the ability to maintain a course when going from one location to another (Franz & Mallot, 2000; Trullier, Wiener, Berthoz, & Meyer, 1997).

The basic tasks and capabilities required for accomplishing navigation are localization and mapping. In robotics, the combination of these two tasks is referred to as the simultaneous localization and mapping (SLAM) problem (Bailey & Durrant-Whyte, 2006; Durrant-Whyte & Bailey, 2006; Wallgrün, 2010, chap. 2). Mobile robots using a SLAM algorithm to both map its environment and localize itself within that map, do so at a level of adequacy that is based on the fidelity of their sensory input data. Because autonomous mobile robots have sensors, actuators and navigation algorithms that cater to their application and working environment (Gonzalez-Arjona, Sanchez, López-Colino, de Castro, & Garrido, 2013; Sariff & Buniyamin, 2006), these robots can still be very rigid and short coming in their navigation capabilities. The problem areas that arise in navigation include dealing with dynamic environments, as well as the need for high precision localization data for mapping and path planning.

Animals, on the other hand, are masters at navigating in their environments. For central to biological based navigation is the ability to travel from one place to another without getting lost (Tolman, 1948). It was suggested by Tolman in 1948 that for rats and humans to be able to accomplish various navigation tasks, they must have a cognitive map of their environment in their head (Redish, 1999; Tolman, 1948). In 1971, O'Keefe and Dostrovsky (1971) discovered a special type of neuron in the rodent's hippocampus that fired only when the rodent was in a specific location and was aptly named the place cell. It became evident that place cells were part of the suspected cognitive map and it has been heavily researched from that point on. Since the discovery of the place cell, the head direction cell, and the boundary cell were discovered in the rodent's hippocampus and its surrounding area, and the grid cell in the neighboring entorhinal cortex. These specialized brain cells are believed to play a vital role in the navigation abilities of the rodent. The hippocampus is also believed to be involved in the storage of new episodic memory (Burgess, Maguire, & O'Keefe, 2002; Fyhn, Molden, Witter, Moser, & Moser, 2004).

* Corresponding author.
E-mail address: pzeno@my.bridgeport.edu (P.J. Zeno).

http://dx.doi.org/10.1016/j.bica.2017.09.002
Received 26 April 2017; Received in revised form 17 July 2017; Accepted 20 September 2017

In addition to cognitive maps used by rodents, many species, such as spiders, crustaceans, insects, birds, and many mammals are capable of homing. That is, they continually update an internal vector trajectory with respect to their previous location to be able to return directly home (Müller & Wehner, 1988; Redish, 1999). These creatures can do so even after wandering in its environment for some time. Homing can also be accomplished in the dark and through unknown areas, and despite having traveled a circuitous route. This is accomplished through dead reckoning, which is also known as path integration (PI), as originally proposed by Darwin (1873). For the rodent, the neural circuitry involved is speculated to take place in or around the hippocampus and its surrounding area.

The remainder of this paper outlines the navigation system designed around the functional concepts of these specialized rodent's navigation and spatial awareness cells. First, the firing and functional characteristics of these cells are reviewed.

Rodent's specialized navigation cells

The rodent brain has been studied greatly, particularly the hippocampus and its surrounding area for its navigation related cells (Bush, Barry, & Burgess, 2014; Redish, 1999). These cells include: place cells, boundary cells, head direction cells (in the subiculum), and grid cells (in the neighboring entorhinal cortex). The rodent is not the only mammal with these special brain cells. Mice, rats, and bats have been found to also have place cells and grid cells (Burgess, Recce, & O'Keefe, 1994; Moser et al., 2014). However, this list is most likely broader. A brief description of the firing characteristics of these navigation related brain cells follow and can also be found in (Zeno, 2015; Zeno, Patel, & Sobh, 2016). Fig. 1a illustrates the location and size of the rodent hippocampus (left and right), while Fig. 1b illustrates the major components of the hippocampus, via a cross section horizontal slice of the ventral portion of the hippocampus. The locations of the specialized navigation cells with respect to the areas shown in Fig. 1b, as well as their basic behavior are covered next.

Place cells

A place cell (PC) fires maximally when the rodent is in a particular location of its environment (Bush et al., 2014). A place cell is usually limited to a single firing field (FF), unless the environment is large. Thus, many PCs are utilized to map a rodent's environment. Additionally, the firing of a PC is only dependent on location and not direction (in rodents), unless the place field is in a constrained location, such as a maze corridor. PCs are found mainly in CA3 and CA1 of the hippocampus (pyramidal cells), and to a lesser degree in the dentate gyrus (DG) with smaller place fields (Redish, 1999). As described in the introduction, PCs play an important role in the mapping of the rodent's environment, which is typically identified by the term place code. A PC's FF size is dependent on its type and location in the hippocampus. As presented in (Kjelstrup et al., 2008), a place field can be defined as the area between the points in an environment where the theta phase precession begins and terminates.

Head direction cells

The head direction (HD) cell fires at a preferred direction (± a few degrees) of the rodent's head direction in the horizontal plane, and has no relation to the rodent's body position. HD cells are aligned to the rodent's allocentric cues found in its environment, but are informed of motion through vestibular signals. HD cells are found primarily in the rodent's postsubiculum (PoS), the anterior thalamic nuclei (ATN) and the lateral mammillary nuclei (LMN) (Redish, 1999; Taube, 2007).

Boundary cells

The boundary cell (BC) is direction invariant and location specific in its firing. The BC typically has a single FF, which is dedicated to a particular boundary or border in the rodent's roaming environment. BCs can be found in the medial entorhinal cortex (MEC), parasubiculum (PaS) and subiculum (Bush et al., 2014). Additionally, it is believed that there are boundary vector cells (BVCs) in the subiculum which fire according to a fixed distance and direction to a boundary (Derdikman, 2009; Lever, Burton, Jeewajee, O'Keefe, & Burgess, 2009). From here on, we will use BVCs in our system description and will simply designate them as BCs.

Grid cells

The grid cell (GC) is a unique spatial awareness brain cell found in the entorhinal cortex (EC) of a rodent. GCs are predominantly found in layer II of the medial entorhinal cortex (mEC), which is located one synapse upstream of the PCs in the hippocampus (Fyhn, Hafting, Treves, Moser, & Moser, 2007; Hafting, Fyhn, Molden, Moser, & Moser, 2005). The GC differs from the PC and BC such that it has many spatial FFs. Each GC's FF maps over the rodent's entire roaming environment in a hexagonal lattice formation. At the node of each equilateral triangle in the lattice is the location of a single FF of a GC. The FFs of a GC,

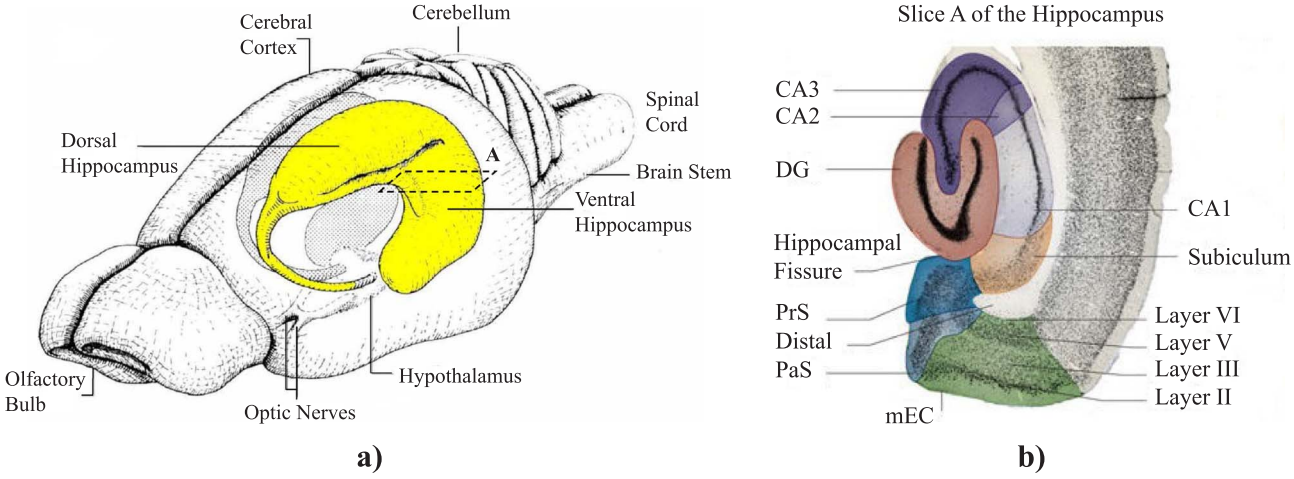

Fig. 1. The rodent brain. (a) In yellow is the left hemisphere hippocampus. (b) Anatomy of hippocampal formation and parahippocampal region (horizontal slice A in part a). Abbreviations: Carnu amonis (CA), dentate gyrus (DG), lateral entorhinal cortex (lEC), medial entorhinal cortex (mEC), parasubiculum (PaS), and presubiculum (PrS). Picture adaptions: Figure (a) from (Little, 2007), and (b) from (Moser et al., 2014).

which creates a hexagonal lattice across the environment, is defined shortly after a rodent is introduced to a novel area (Barry & Burgess, 2014). It is suggested that the lattice is anchored in orientation and phase to external landmarks and geometric boundaries (Hafting et al., 2005; McNaughton, Battaglia, Jensen, Moser, & Moser, 2006; Moser & Moser, 2008). Additionally, each FF of a GC is direction independent. Although, there do exist conjunctive grid cells in the middle and deeper layers of the entorhinal cortex, which fire only on a given absolute direction (Moser & Moser, 2008; Sargolini et al., 2006; Wyeth & Milford, 2009). It should also be noted that the FFs of GCs can differ in three different ways: size, orientation, and phase. The GCs FF's size increases monotonically from its dorsal to ventral location in the mEC (Moser, Kropff, & Moser, 2008; Moser & Moser, 2008).

Rodent navigation system computational models

There are currently two prevailing computational model classes for describing the stimuli configuration required for the grid cell firing pattern. The first is the continuous attractor network (CAN) model, which in simplest terms, is a neural network based model that describes the stabilization or convergence of a multistate system to a single state over time, by way of synaptic interaction between excitatory and inhibitory neurons (Boucheny, Brunel, & Arleo, 2005; Shipston-Sharman, Solanka, & Nolan, 2016). An example CAN model that describes the role of PI related cells in the firing of GCs is outlined in (Burak & Fiete, 2009). The second computational model, is the oscillatory interference model (Burgess, 2008). This model is typically simulated using spiking neural networks on non-robotic systems (Erdem & Hasselmo, 2012, 2014; Giocomo & Hasselmo, 2008). Both working models have strong pros and cons to their validity. However, we chose the interpreted relationship between the various specialized navigation cells previously covered using the oscillatory interference model presented in (Erdem & Hasselmo, 2014). Our choice was driven by how well this model fits our functional interpretation of how these special cells interact. Fig. 2 illustrates the basic connectivity of the chosen oscillatory interference model.

Oscillatory interference model

The oscillatory interference model relies on HD cells, based on their preferred direction and the current head direction of the rat, to modulate persistent spiking cells (oscillators) who's frequency is a function of the distance traveled by the rodent over a delta time-period. Each oscillator has a given offset phase and frequency scaler. Each GC is fed by the same input network of oscillators in a neural network layer configuration, then the output of these GCs feed PCs. Thus, the HD cells and oscillators act as a PI system, which feeds the GCs. The GCs will fire when their FFs overlap the physical location any PC is tuned to. The oscillatory interference computational model for the implementation presented in (Erdem & Hasselmo, 2014) is as follows:

$$\phi_{i,j}(t) = 2\pi(ft + b_j \int_0^t d_i(\tau)d\tau) \quad (1)$$

$$s_{i,j}(t) = H(\cos(\phi_{i,j}(t) + \psi_{i,j}) - s_{thr}) \quad (2)$$

$$g_j(t) = \prod_{s \in S_j} s(t) \quad (3)$$

where $\phi_{i,j}$ is the persistent spiking cell's phase modulated by the ith head direction cell and projecting to the jth grid cell, f is the frequency, b_j is the scaling factor for all persistent spiking cells projecting to the jth grid cell, $s_{i,j}$ is the persistent spiking cell signal, ψ is the phase offset, s_{thr} is the threshold, H is the Heaviside function with $H(0) = 0$, g is the grid cell signal, and S_j is the set of persistent spiking cells projecting to the jth grid cell.

Proposed simplified path integration-grid cell firing model

Removing the neurophysiological implementation details, as described in the CAN and oscillatory interference model results in a simplified functional module that can replace the GCs and the path integrator neuron circuitry. As previously described, the PI related inputs to the persistent spiking model are the HD cells and distance traveled information. From a hardware implementation view point, heading information from a microelectromechanical systems (MEMS) gyro (vestibular data) can replace the HD cells, and distance traveled information in delta time ($\int_0^t d_i(\tau)\, d\tau$) can be replaced by distance data gathered from motor encoders (proprioceptive data). The distance traveled, is assumed to be straight segment movements. Therefore, the travel vector that emerges from the combined path integrator and GCs module, in moving from one point in the environment to another, is of magnitude d and at heading θ, or vector $\mathbf{d} = (d, \theta)$. The travel vectors of the mobile robot presented in this paper are acquired and transformed into Cartesian coordinates by a microcontroller. Our model maps the functional firing characteristics of a GC to a Cartesian coordinate system. Being that GCs are neural network based, this might not be a perfect one-to-one comparison, however, from a top-level view, there are many functional similarities.

In our system, the x, y coordinates for an internally stored Cartesian coordinate based map, are found by using the sine and cosine functions on the mobile robot's tracked allocentric heading θ. This is similar to the cybernetic models of PI found in (Strösslin, Chavarriaga, Sheynikhovich, & Gerstner, 2005), and also presented by Mittelstaedt as described in (Redish, 1999). The travel vector to coordinate equation used is as follows:

$$x_k = d_k \sin(\theta_k) + x_{k-1} \quad (4)$$

$$y_k = d_k \cos(\theta_k) + y_{k-1} \quad (5)$$

The terms x_{k-1} and y_{k-1} represent the Cartesian coordinates of the robot's last stop or turn. For $k = 0$, the values of these terms, (x_{-1}, y_{-1}), are defined as (0, 0). This represents the initial starting location (home) of the robot. Fig. 3 shows the graph assignment with respect to "home" and an initial allocentric bearing of 90° (θ = 0° for the robot's internal calculations).

Mobile robot sensory input

Before going into detail of our proposed neurophysiological based navigation system, we will cover the sensors used by our robot, known here after as *ratbot*, to obtain information about its environment.

Idiothetic sensors for path integration

Heading sensor

As previously described, the *ratbot* uses the InvenSense MPU6050, MEMS – 6 axis, accelerometer and gyroscope for heading data, which is used in place of the HD cells. Specifically, the yaw rotational axis of the gyroscope is used to determine the robot's heading. With MEMS based gyroscopes, however, there is a constant drift. To compensate for this drift, the gyroscope measurement data is sampled in a loop at the beginning of the robot's main program, from which an average drift rate is derived. This drift rate is subtracted from all future reads of the MEMS gyro. However, since the drift rate is not perfectly constant, this value will slowly drift as well. The graph in Fig. 4 shows how the measured heading still drifts when the gyroscope is stationary over a 12-min interval. The drift is approximately 15 degrees in this time frame, which works out to be approximately 0.02 degrees/s. The initial, uncompensated drift rate was measured at 0.47 degrees/s. This, of course is just the static error. There are three phases of movement during the turning of the robot in which additional errors can occur: (1) initial acceleration, (2) constant velocity, and (3) deceleration. Since the turn

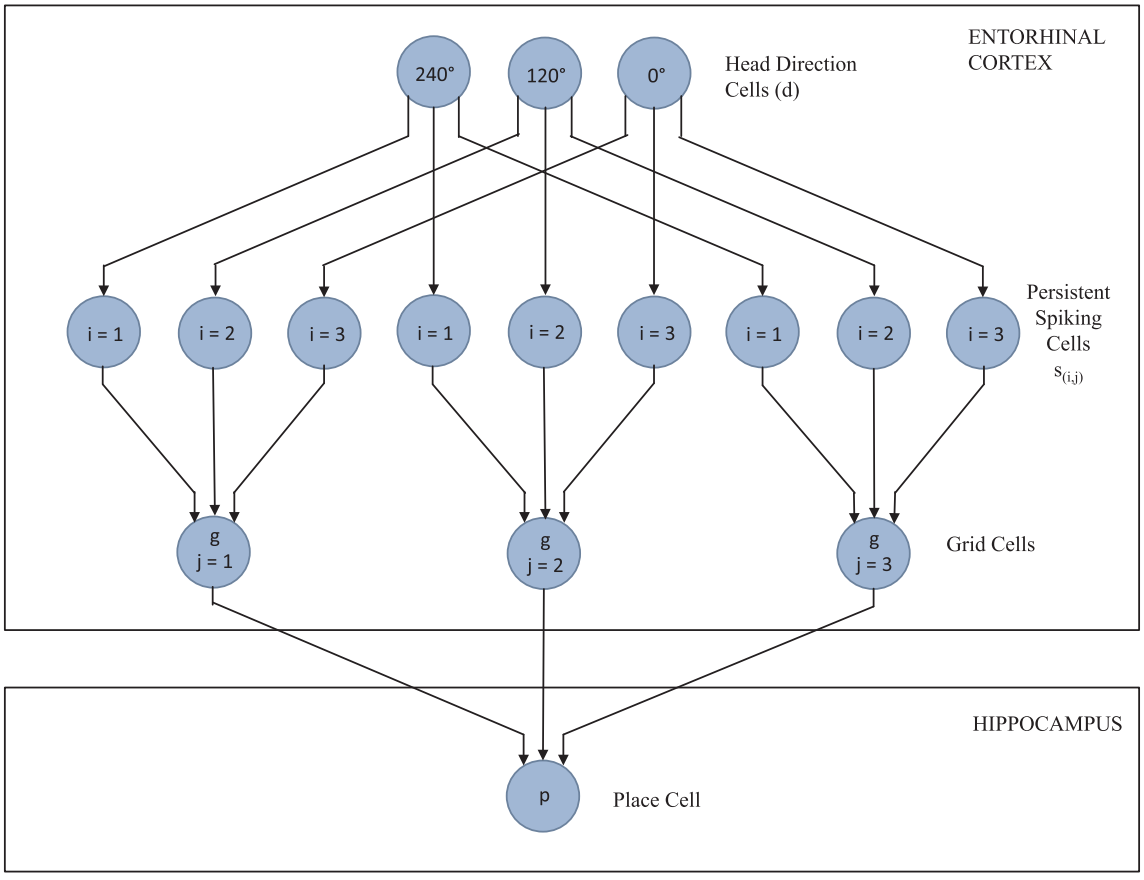

Fig. 2. Architecture representation of the persistent spiking computational model which drives the selection of GCs and PCs as presented in (Erdem & Hasselmo, 2014).

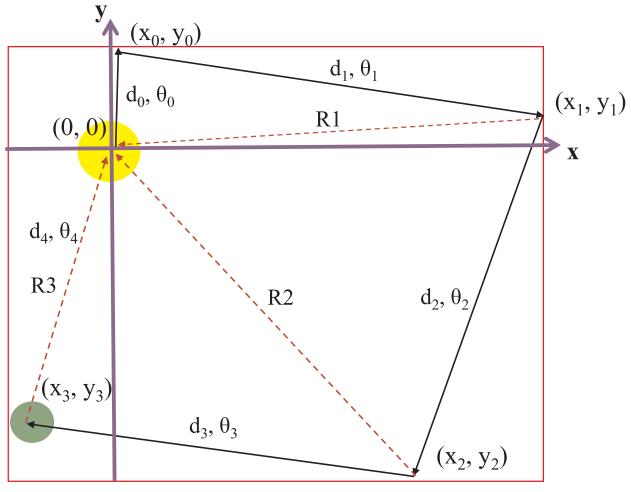

Fig. 3. A conceptual overlay of an internal Cartesian graph representation of the robot's navigation environment. The yellow circle represents the robot's home and starting location, while the green circle is a goal location (e.g., food or water). The PI algorithm always assumes the starting position to be at the origin (0, 0) of an imaginary graph. The black vectors represent the robot's path, while the red vectors (Rn) represent calculated homing vectors. θ_i is the robot's allocentric heading.

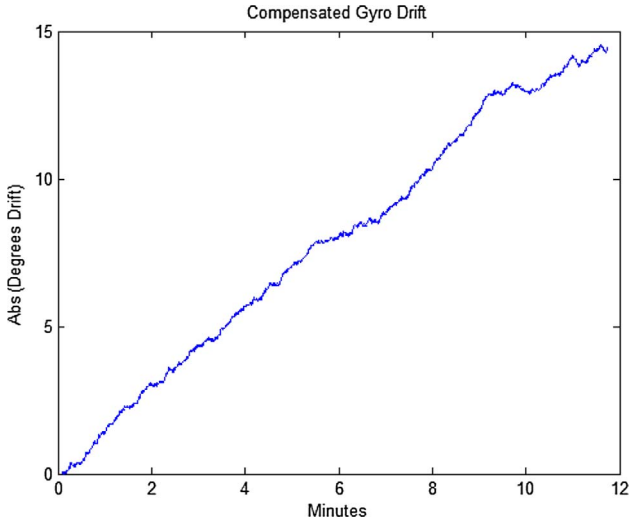

Fig. 4. MPU6050 post drift compensated gyroscope data.

rate is a rotational velocity measurement, there will be rate averaging occurring over these three phases.

The *ratbot's* heading θ via use of the gyroscope is calculated as follows:

$$\theta = \theta_{prev} + (\omega - \omega_d) * \Delta t \quad (6)$$

where θ is the current heading, θ_{prev} is the previous heading, ω is the measured gyro's angle rate of change (16 bit A/D value) at Δt microseconds after the previous gyro rate measurement, and ω_d is the drift rate of the sensor (measured average at startup).

As with rodents and other animals, PI error is reset by observing known external distal cues, which allows them to become certain again of their local or global location (Hafting et al., 2005; Hardcastle, Ganguli, & Giocomo, 2015; Hayman & Burgess, 2015). Autonomous systems have found that using sensors that capture allothetic stimuli, such as visual recognition hardware and software, greatly helps with this area (Hafner, 2005; Strösslin, Sheynikhovich, Chavarriaga, & Gerstner, 2005; Wyeth & Milford, 2009). Therefore, the use of some form of allothetic sensor on the mobile robot is imperative

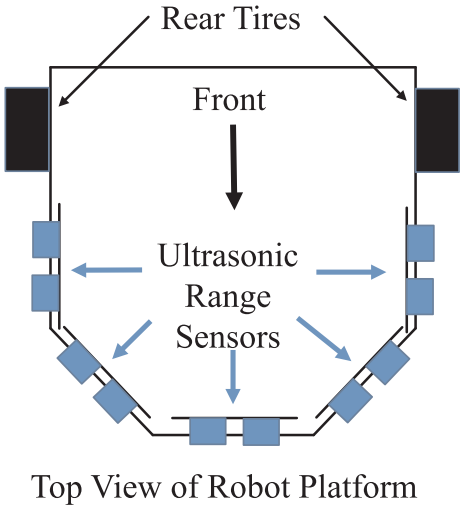

Fig. 5. Ultrasonic range sensors covering the front of the *ratbot*. Five sensors giving a forward looking 180° field of view coverage.

to its autonomous capabilities. The allothetic sensors used on the *ratbot* are covered shortly.

Motor encoders

The *ratbot* uses two Devantech 12 V, 30:1 gear motors with encoders. A Daventech MD25 motor controller board is connected to these motors for digital control of H-bridge motor drivers, as well as data acquisition (i.e., motor encoder values, supply voltage level, etc.) via a PIC microcontroller and a serial interface. The main controller of the *ratbot* sends data to, and receives data from, the MD25 to regulate the movement of the *ratbot* over a serial communication interface, and collect encoder values to derive distance traveled. The encoder values are summed by the PIC microcontroller over time, and can be zeroed out at any time. The encoder values collected are in degrees of the wheel's rotation at a resolution of 2 degrees.

Allothetic sensors

Ultrasonic range sensors

The *ratbot* is equipped with five ultrasonic (sonar) sensors to achieve an object detection coverage of approximately 180°, as shown in Fig. 5. These sensors are located around the front of the *ratbot*: one forward, a pair of left and right angled "whiskers", and a pair of left and right side facing ultrasonic sensors.

The ultrasonic sensors used on the *ratbot* are the HC-SR04 Ultrasonic Range Sensor. The ultrasonic sensor is obviously not as fast or responsive as light based systems, nor as accurate as an optic range finder. Additionally, the beam width of the ultrasonic sensor is much wider and doesn't have the same range capabilities as an optic range finder. However, the ultrasonic sensor does have the advantage of not being affected by the color and texture of the target. For object detection at relatively short distances, the target offset error due to the larger beam width can be greatly reduced. The ultrasonic sensor's beam angle is defined as the total angle, where the sound pressure level of the main beam has been reduced by 3 dB (half power) on both parts of the center axis, represented here by θ. This angle is obtained using acoustic design charts based on the results of the following equations:

$$\lambda = v/f = (343\ \text{m/s})/(40\ \text{k cycles/s}) = 8.6\ \text{mm} \quad (7)$$

$$D/\lambda = 13\ \text{mm}/8.6\ \text{mm} = 1.5 \quad (8)$$

where, wavelength of the sound pulse λ is equal to the speed of sound v divided by the pulse frequency f. The ratio of the diameter of the round transmitter (infinite planar baffle) to the wavelength (D/λ), determines the sound beam's width or angle θ of 20° at 3 dB.

The ultrasonic sensors are used for object detection and avoidance. The data collected from these range sensors, along with pose data, are used for BC FF activation and initialization, which becomes part of the navigation system's cognitive map.

As described, the ultrasonic sensor is unable to collect the level of detail needed to replace a visual system. Particularly, the details required to identify landmarks and goals, and thus perform a reset of the PI error. For this, the *ratbot* uses a visual system and a slightly engineered environment.

Visual system

The *ratbot* uses the Pixy Cam (CMUcam5) from Charmed Labs for landmark and goal location recognition. The Pixy can swivel on a two degrees of freedom platform, via two mini servos. One servo rotates the camera along the horizontal axis, while another servo rotates the camera along the vertical axis. Currently, the camera is used in a stationary position, pointing directly forward and downwards at a 40° angle with the parallel plane of the *ratbot's* platform. The camera lens field of view (FOV) is 75° horizontal and 47° vertical. Fig. 6 illustrates

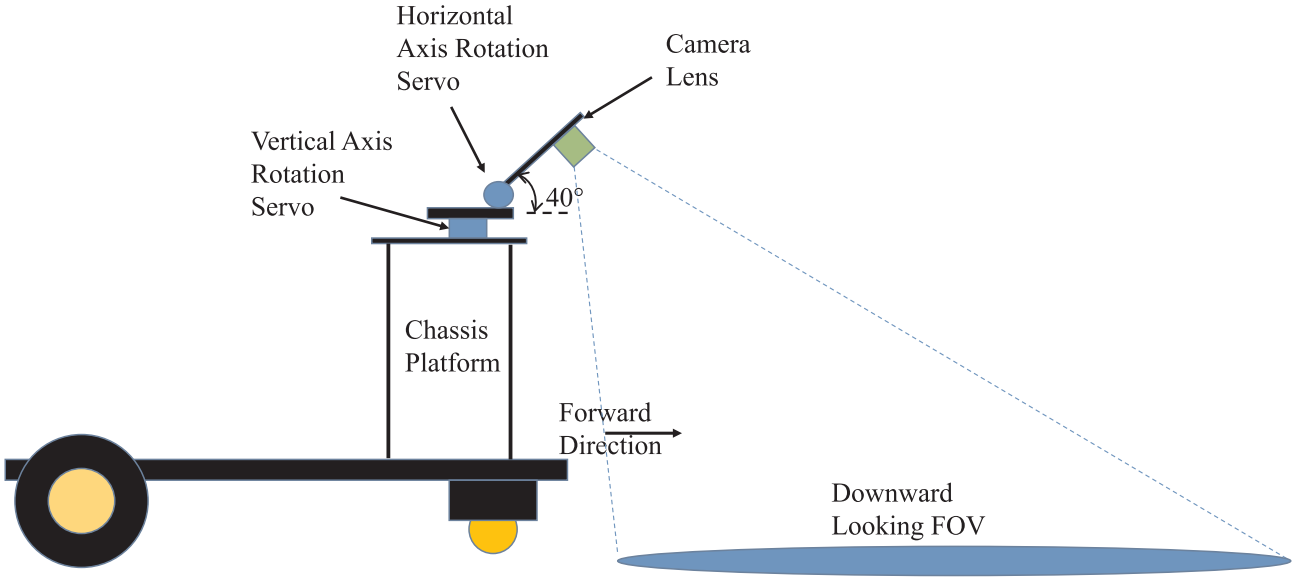

Fig. 6. The *ratbot's* Pixy Cam downward looking FOV. Illustration is a view from the *ratbot's* right side.

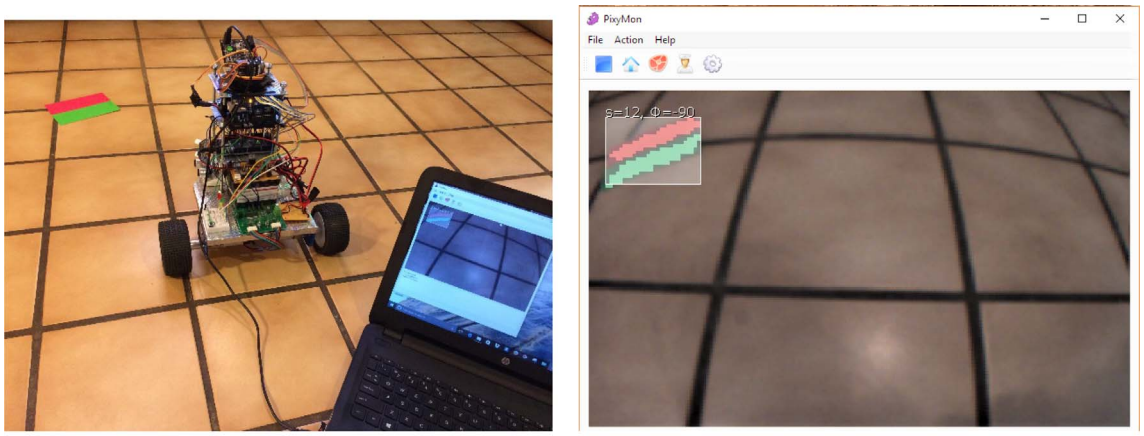

Fig. 7. Demonstration of the Pixy camera. (a) The *ratbot's* Pixy camera is connected to a laptop to demonstrate what the camera sees. The color code card (red and green) represents a preprogrammed goal or landmark that has been recognized. (b) A screen shot of the PixyMon program, which is used offline to program the color codes and display what the Pixy camera sees, such as color code identification (e.g., s = 12), and its relative angle (e.g., $\phi = -90°$).

this configuration of the *ratbot's* camera.

Instead of using neural networks and vision data compression algorithms to record and compare gathered visual data to, as is done in reviewed systems (Zeno et al., 2016), the Pixy Cam identifies objects by color, using a connected components algorithm to determine where one object begins and another ends. Additionally, using more than one color placed next to each other (color code), allows for many more objects to be uniquely identified. For the *ratbot's* environment, goal places (i.e., home, water and food), and unique landmark locations are marked by color coded cards. An example of this is shown in Fig. 7a and b. The identities of these color codes are pre-programmed into the Pixy Cam's flash memory using the PixyMon application.

Therefore, a tradeoff is made between having an ANN based visual recognition system, which doesn't require an engineered environment, versus using a simple color-code based system, which requires very little processing power and resources, but a slightly engineered environment. Since the aim of this paper is to test the core of the navigation system, the actual vision system used is of little consequence. However, the processor onboard the Pixy Cam does calculate relative X, Y position data with respect to the object's location in the camera's field of view. Additionally, the angle of the color-code image, with respect to the axis running between the two or more colors, is calculated and is available for use. This is displayed as ϕ in Fig. 7b. Therefore, the color-coded object's pose can be translated into an allocentric pose, based on the robot's current pose data.

Rodent inspired mobile robot navigation system

It can be inferred from the described firing characteristic of the rodent's navigation specific cells that simplification of their connectivity from a neural network based system to a computational architecture results in a loss of low level understanding of their true interactions. However, it can be argued that much of the connectivity of this part (or any) of the rodent brain is not well understood in the first place. Thus, this paper presents a higher level functional framework to characterize what is known or interpreted to be true about these brain cells. This allows for testing the understanding at a top level, and helping to direct further studies at the lower levels (i.e., neural networks). Additionally, what is found to be more optimal for autonomous navigation can be utilized in mobile robots. The following describes the mapping of the rodent's navigation specific cells to their hardware and software counterparts.

Path integration

PI systems, whether natural or manmade, requires the integration of some form of compass (sense of direction) and distance cues (Goldschmidt, Dasgupta, Wörgötter, & Manoonpong, 2015; Wolf, 2011). Thus, in a rodent, PI is achieved through the integration of the HD cells with self-motion cues (Stackman, Clark, & Taube, 2002), as illustrated in the previously described oscillatory interference model. Non-external self-motion information comes from proprioceptive stimuli (e.g., muscle movement signals), and vestibular stimuli (e.g., inner ear angular head velocity signals). Self-motion information can also come from external sources, such as visual and tactile information. For the *ratbot's* navigation system PI is achieved using a MEMs based gyroscope for the angular velocity data, and wheel encoders from the two rear motors for distance measurements. Thus, HD cell firing data has been directly replaced by a gyroscope and the distance covered in Δt by the motor encoders.

New multimodal navigation model

As proposed in (Bush et al., 2014), grid cells and place cells are speculated to not be successive stages of a processing hierarchy, but complementary. This is contrary to the oscillatory interference model presented earlier, but clearly an optional addition to the model. Additionally, BCs have a great influence on the creation of PC fields (Lever, Burton, Jeewajee, O'Keefe, & Burgess, 2009; Stewart, Jeewajee, Wills, Burgess, & Lever, 2014). Factoring in that taxon navigation takes place by visual input primarily, works into a newly derived model by this paper as to how PCs are activated. As illustrated in Fig. 8, in the new multimodal model, there are three parallel sources which feed the input to the PCs. Firstly, as illustrated and described in (Bush et al., 2014), active BCs for a particular environment can source a PC to fire near the intersection of two adjacent boundaries. Similarly, our model produces place fields at the ends of boundaries. Secondly, unique locations, such as landmarks and goals locations, can be learned from the visual data, thus creating place fields which can be used to help reduce PI error. Thirdly, the metric, coordinate based system used by the *ratbot* to map out its environment is similar to the function of GCs in the rodent's brain, which source the activation of PCs for cognitive map generation, (i.e., place code), as well as allow for PI in complete darkness (no allothetic stimuli).

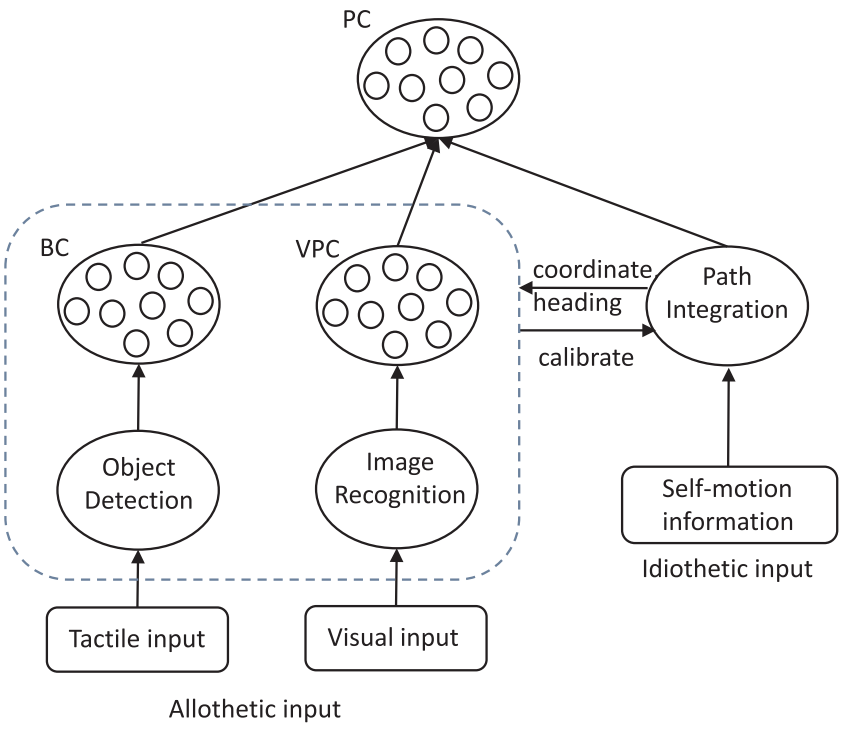

Fig. 8. The proposed multimodal model of the PC firing field sources. The cognitive map located in the central processor and FPGA will possess BCs, visual place cells (VPCs) and PCs. The output from the PI source to the PCs represents the interaction of PI data with GCs (pseudo coordinate data), which in turn enable the place fields.

Navigation system implementation

To better understand how the new multimodal navigation system works, we will describe the hardware connectivity and data flow of the total system. Then we present the cognitive map and spatial awareness created through data structures on the central processor and a field programmable gate array (FPGA) silicon chip.

Hardware system design

Central processor

The main agent of the *ratbot* is the central processor board, an Arduino Mega 2560, which uses an Atmel® ATmega2560 microcontroller, and is integrated to the external sensors and actuators previously covered. The ATmega2560 microcontroller is limited to 256 Kbytes of program memory and operates at 16 MHz. Additionally, the central processor board uses many of its 54-digital input/output pins and four serial ports to gather data from sensors, communicate with another microcontroller boards, interface with the Pixy Cam (discrete signals), and communicate with the motor controller board and gyroscope, see Fig. 9 and the pictures of the *ratbot* in Fig. 10. Thus, the central processor gathers data about the environment through the ultrasonic range sensors, camera, motor encoder data, and MEMs based gyroscope (via I2C bus), and makes decisions on the next action to take, based on the sensor data and its current motivation state.

The basic decision making of the core multimodal model, illustrated in Fig. 8, is carried out in the central processor. Possible new BC, visual place cell (VPC) and PC FFs are identified in the central processor's main loop program. The pseudo code for the main loop program is listed in Fig. 11. The data from these newly identified VPC and PC FFs are stored in the central processor's memory. The data structures that represent these navigation specific cells of the rodent will be covered shortly.

FPGA

The direction and coordinate data for the BC FFs are sent to the FPGA from the central processor through a serial interface. The FPGA checks in parallel the activated BC modules as to whether the current identified BC FF is new or not. The FPGA will then either activate a new cell module and save the relevant data in it, add to a currently activated BC, or simply ignore the redundant data sent. The FPGA also performs a connected components algorithm that is specific to the BC. This allows for a single BC to fire over a continuously connected boundary, as is performed in the rodent's brain. An illustration of the BC module architecture, which is designed into the FPGA is given in Fig. 12.

Software based representation of navigation and spatial awareness cells

As can be seen from the main loop pseudo code outlined in Fig. 11, the action that takes place by the actuators (motors) of the *ratbot* is a function of the agent's motivation state. Thus, the motivation state integrates with the core multimodal navigation model, and is influenced by the current state of the environment. The motivation states of the *ratbot's* navigation model includes: hunger, thirst, tired, lost, fight or flight (predator threat), and curious (specifically: in a new area and goes into explore mode). The explore mode is the initial navigation state resulting from the curious motivation state of the *ratbot*. In this mode the *ratbot* randomly navigates its environment, while mapping the area by creating a place code using BCs, VPCs and PCs as shown by the multimodal model in Fig. 8. This phase continues until the *ratbot* has discovered the food and water goal locations, and saved the path information between the goals and home. How these specialized neural cells relate to each other logically and are stored is presented next. The BC data structures that are managed by the FPGA has already been described, and its overall importance will be covered next as well.

Place cell data structures and the cognitive map

As with the PCs in the rodent's hippocampus, the *ratbot's* PC data structures records key locations in its environment. Additionally, they are linked together to represent paths found or typically taken. Thus, the *ratbot* constructs the place code or cognitive map in the form of a topological graph representation with stored allocentric location information (Konolige, Marder-Eppstein, & Marthi, 2011; Thrun, 2002). The *ratbot's* PC data structure stores the following information: ID number, type, Cartesian coordinates, several pointers that can be initialized to point to other PC structures, and the Euclidean distance

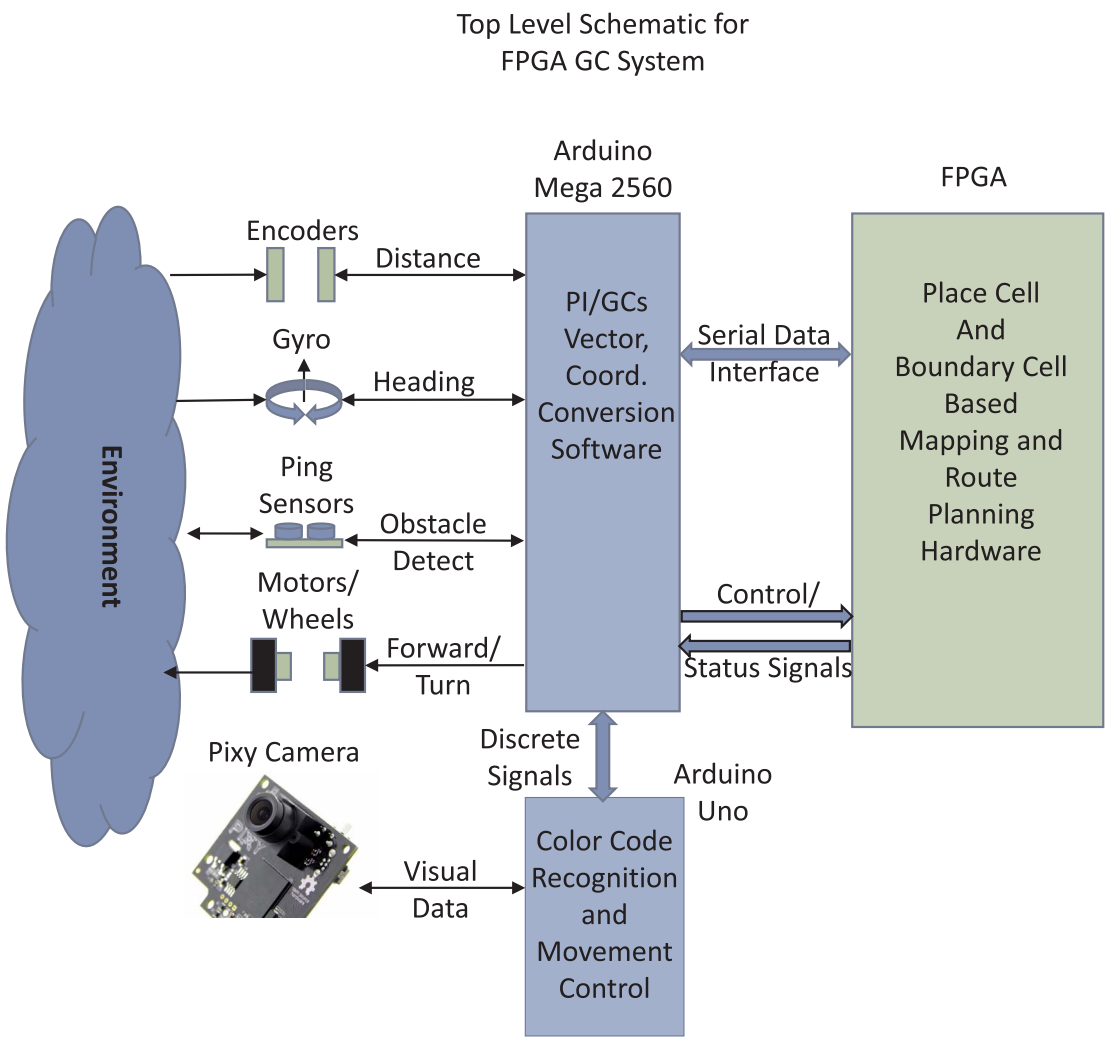

Fig. 9. Top-level block diagram of the *ratbot's* neurobiological based navigation system. Shown are the *ratbot's* sensors, actuators and computational resources.

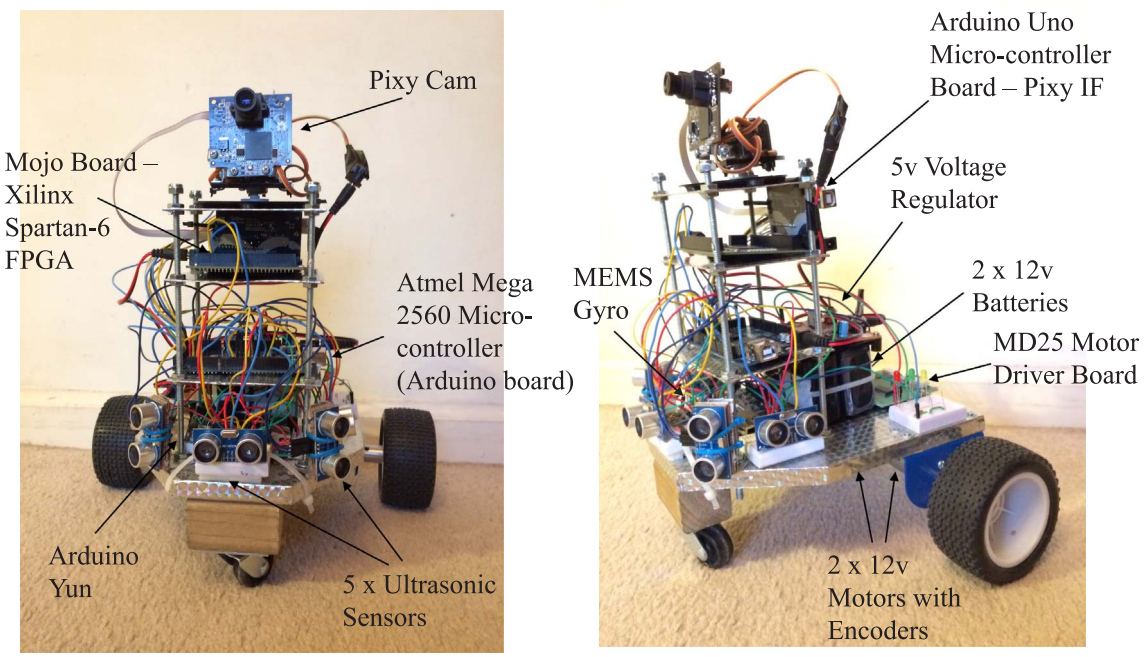

Fig. 10. The *ratbot* and its hardware.

```
1.  While MotivationState ≠ done:
    a.  Check visual data gathered from camera for objects recognized.
    b.  If (Identified Object[k] AND Searching for Object[k]) then
        i.   Go to Object[k];
        ii.  Record and Verify Object[k] (VPC, PC) in FPGA;
        iii. Take action based on MotivationState AND ObjectType;
    c.  Get sonar data (distances of objects) from all five sensors.
    d.  Based on MotivationState and barrier(s) distance(s)/location(s):
        i.   Record and Verify BC or PC in FPGA
        ii.  Take action.          /* e.g., stop, turn, go forward … */
    e.  Check MotivationState for change.
```

Fig. 11. Central processor's main loop pseudo code.

between connected PCs (to be used for the A* path planning algorithm). The ID is merely a sequential number given to the PCs as they are found. The type is a location descriptor, such as: a goal (color-coded marker on floor), a turn, landmark or some unique location, or boundary corner. A goal place cell (G) is the location of a particular goal (e.g., home, water, or food). A turn cell (TC) is the name given to a PC which marks the turn location for the robot at the end of a boundary. Boundary corner cells indicate a dead end that is created by two intersecting boundaries. This type of PC is very much like a unique landmark location.

With the current visual capability of the *ratbot*, the VPCs are currently pre-programmed into the microcontroller that collects visual

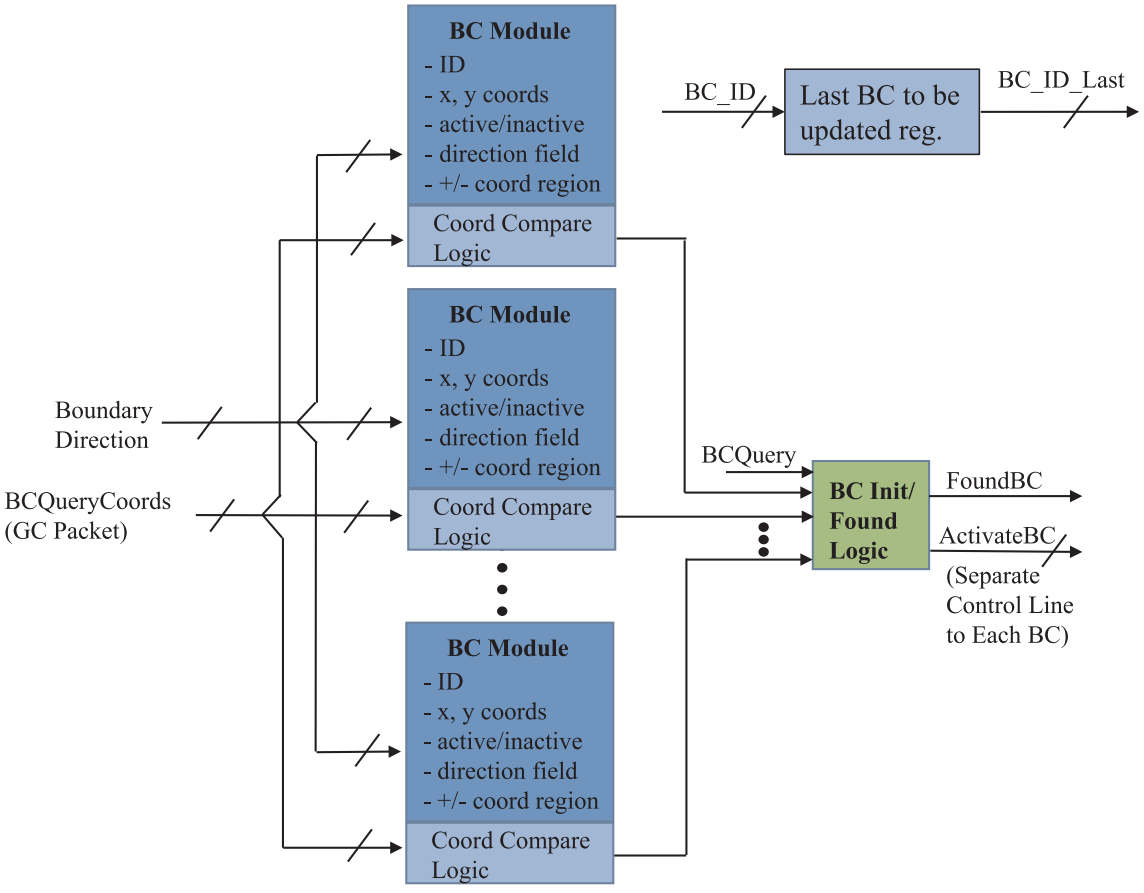

Fig. 12. BC implementation in the FPGA.

data from the Pixy Cam. Thus, the identity of each color-coded marker represents the VPC dedicated to that goal or landmark. Future versions of the proposed navigation system will use an artificial neural network (ANN) to learn and remember these unique locations.

Goal memory

As defined in (Redish, 1999), goal memory in a rodent plays a role in route planning to goal locations, and is based on the position of the animal and its current needs or motivations. Additionally, goal memory requires input from the place code to accomplish route planning. Our implementation of goal memory is a linked list of PC structures (place code) defined above. During the exploration phase, the path from a goal location found by the *ratbot* (e.g., food or water) to the home location is recorded in a linked list. The steps that occur to find the path from a found goal to home during exploration is as follows:

(a) Random exploration until a goal location is found (water or food).
(b) Dedicate a place cell for this goal location (store coordinates).
(c) Calculate vector home from goal location and home coordinates.
(d) Turn towards home based on direction calculated in step (c).
(e) Head towards home, checking visual data for home recognition, distance traveled and barrier detected.
(f) If barrier is detected, go into wall-follow mode, else go to step (h). Record boundary vector cells as robot moves along barrier. Use scan and backtrack algorithm if dead end is found, else search for opening towards home.
(g) If path was found blocked in step (f) and opening is found, go to step (b) and use TC for transition point around barrier in place of goal location.
(h) If home is recognized, go to home location and stop, else go to step (e).
(i) The path from the goal found to home should now be stored in linked list (e.g., G1 → TC1 → G0). Save length of path as edge value for this path (the saved path represents goal memory).
(j) Return to the goal just found by reverse traversal of the linked list path just stored in the place code (e.g., G0 → TC1 → G1).
(k) Go back to step (a), unless all goal locations have been found.

Software system design

Our software block diagram is designed after the computational spatial cognitive model used in (Barrera & Weitzenfeld, 2007, 2008). We broke our software functional blocks into similar logical blocks in that literature. Fig. 13 illustrates the block diagram of the *ratbot's* navigation software system.

Localization and path planning

Localization

The *ratbot* initially localizes itself to its environment based on its starting location and heading at its home position, as is illustrated in Fig. 3. The initial anchoring of spatial orientation of the *ratbot's* coordinate system is similar to the way a rodent's grid network (GC FFs) becomes aligned with respect to external landmarks shortly after being introduced to a new environment (McNaughton et al., 2006; Moser & Moser, 2008). The use of both allothetic and idiothetic data is essential in adding spatial information to memorized visual information (Filliat & Meyer, 2003). The occupancy grid is an example of a metric or grid based traditional (non-biological based) fine-grained localization and mapping technique which shares some similarities to our neurophysiological based system. However, occupancy grids rely on highly accurate pose data with respect to a single global coordinate system. The pose system is typically a combination of both odometry and external (environment sensing) based sensors. The robot's area is divided up into equally sized squares. Each internally represented square, in the robot's memory, is given a probability of being occupied. This value is based on sensor readings, such as sonar sensors, which is typically represented by a two-dimensional Guassian equation (Gonzalez-Arjona et al., 2013; Grisettiyz, Stachniss, & Burgard, 2005; Meyer-Delius, Beinhofer, & Burgard, 2012; Thrun, 2003). Due to the amount of detail collected for the map of the occupancy grid, and the increasing number of squares for large areas, the memory requirement becomes unbounded, as well as the time to map the area. The *ratbot*, however, only creates a place code with a select group of salient entities (i.e., goals, turns, and landmarks) and BC mapped areas for internal boundaries. Comparisons between the occupancy grid localization and mapping method and our neurobiologically based system occur throughout the remainder of this section due to their many similarities.

Since idiothetic data is cumulative, so is the error. Thus, after time, the accrued PI error becomes too great for a robot to rely on its internal position estimate. Additionally, allothetic information can be misleading due to different locations having similar views or representations (perceptual aliasing) (Arleo, Smeraldi, & Gerstner, 2004; Hafner, 2008; Wyeth & Milford, 2009). The *ratbot* uses a level of confidence function to keep the PI error bounded.

Level of confidence calculation

The *ratbot* minimizes PI error by performing timely resets based on a calculated level of confidence (LoC). The LoC can be as simple as an allotted amount of time before a robot needs to return home to recalibrate (Arleo & Gerstner, 2000), to being a function of the location delta between an observed and previously recorded PC FF location to current PI data, or a similar PC based comparison algorithm (Jauffret, Cuperlier, & Gaussier, 2015). The *ratbot's* LoC is calculated using a combination of these two methods.

Due to the constant drift over time of the *ratbot's* gyroscope, as discussed in the sensors section, the LoC requires a time element to its calculation. Additionally, there will be an assumed level of systematic and non-systematic errors occurring with each turn the *ratbot* makes. A threshold is set such that if the LoC decreases to a certain point, BC, PC, and VPC FFs will no longer be assigned to the cognitive map. Since the true accuracy of the PI system is not known, the *ratbot* performs verifications of already learned place fields (i.e., goals and landmarks) it comes across during any of its navigation modes, if the LoC is above the threshold. The LoC is adjusted in such cases. If the *ratbot* becomes lost, such that it is having to be reset too often by too great of a differential in pose, then the *ratbot* will search for home to recalibrate with its home base and initial heading.

Place and boundary field initialization accuracy

The global or allocentric location of barriers detected via ultrasonic sensors, and goal locations detected via the *ratbot's* camera, are calculated by translating their relative position to, and direction from, the *ratbot's* inertial frame, as illustrated by the lines on the *ratbot* in Fig. 14a and b. Only the front and side ultrasonic sensors are used in the creation or verification of BCs. Fig. 14a illustrates how the *ratbot* reacts when the "whisker" sensor detects an object which becomes too close in range (pre-defined threshold) with respect to the *ratbot's* path. The *ratbot* will stop and rotate until a measurement from the side sensor hits a minimum, indicating a near parallel position to the object, see Fig. 14b. The data from the side sensor is then stored in the FPGA for the BC module. Boundary detection occurs in the sonar data processing and affordances portion of the navigation software, as illustrated in Fig. 13.

As previously covered in the Ultrasonic Range Sensors subsection, the large beam width of a sonar sensor affects the accuracy of a detected barrier's or object's actual location. Even with perfect PI, the actual location of a detected object is described statistically, as found with occupancy grid mapping. However, at short distances from the object and for long objects, such as walls, this is not so much of an issue, due to the integration of the locations detected into a single, continuous element by the FPGA. Additionally, where walls, boundaries or barriers

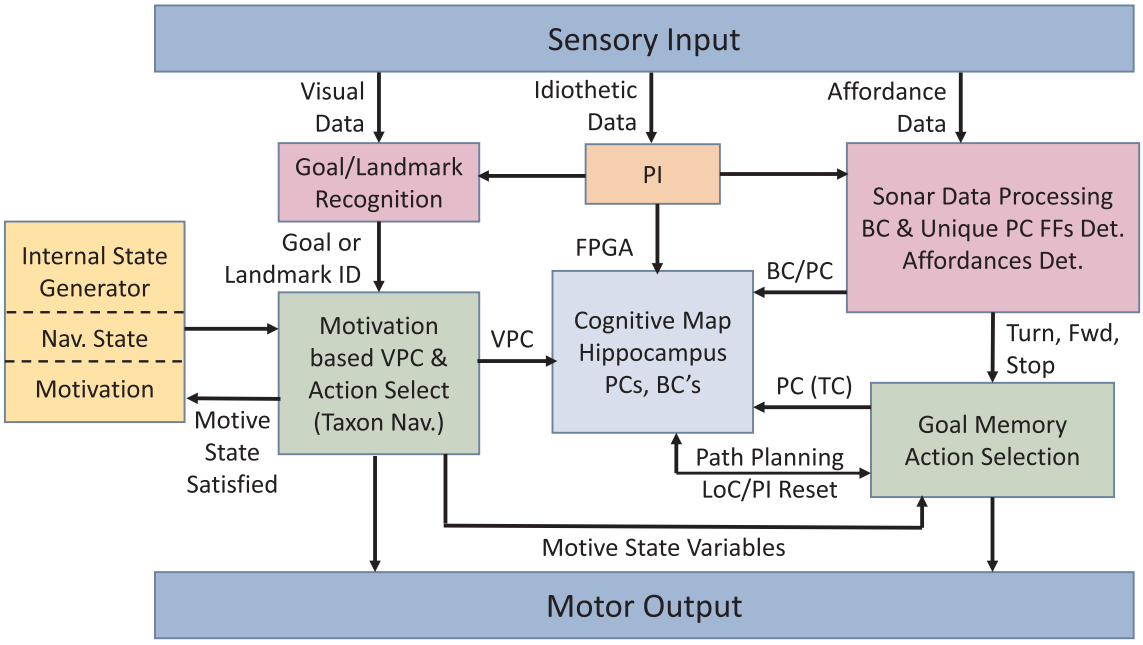

Fig. 13. Software block diagram of the *ratbot's* neurophysiological based navigation system.

end can be discerned and maneuvered around with real time sensing. Only with mapping smaller objects does accuracy become a real issue.

Route planning

There are two methods to the *ratbot's* ability to perform path planning. The first method is simply following the paths stored during exploration to go from one goal to another. These are not necessarily optimal paths, but they were found by following a set routine to return home, which is based on the rodent's use of homing (trying to be as direct as possible). This type of navigation uses goal memory and is designated as local navigation.

The second method finds a new route from a new or previously dedicated PC FF to another stored goal location is designated as way-finding. This form of navigation allows for changes to occur in the environment and gives the *ratbot* the capability to reroute on the fly. The BC FFs stored in the FPGA come into use in this route planning method. The *ratbot* performs the following sequence of events to find a new path:

(1) Create line equations for the target path and for each BC that has multiple locations assigned to it (e.g., a wall).
(2) Check to see if the target path intersects any of the recorded boundaries.
(3) If no intersection (blockage) is detected, then proceed straight towards target. Go into exploration mode if an unrecorded barrier is found in the *ratbot's* path.
(4) If an intersection is found, add the TCs associated with this BC and perform the A* algorithm on this graph.
(5) If the A* algorithm fails to find a previously found goal, then exploration is required.

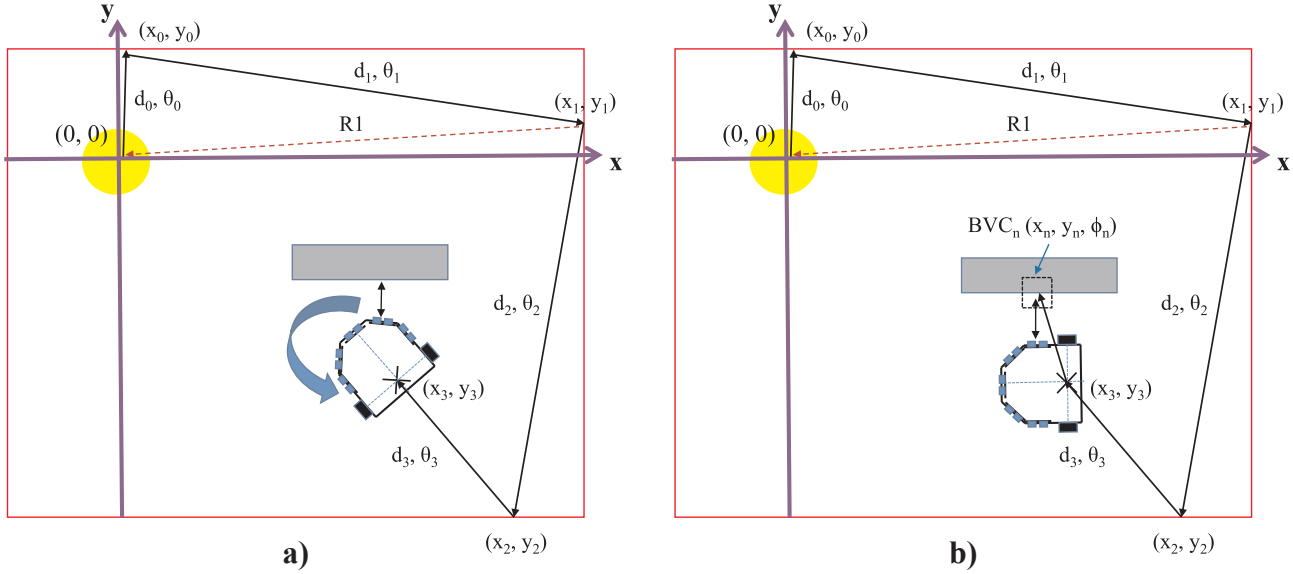

Fig. 14. Recording of BC (BVC) location and angle of incident. (a) The right "whisker" sensor detects a barrier (grey rectangle) to be too close (threshold range), so the *ratbot* stops and rotates. (b) The distance measured from the side ultrasonic sensor is translated to a global (allocentric) reference frame from the *ratbot's* inertial frame.

The A* path planning algorithm variant of the Dijkstra algorithm was chosen because it best aligns with the concept of how rodents, animals, and insects follow a straight vector (minimal Euclidean distance) during homing. For the A* heuristic function, which performs a least-cost path algorithm on the nodes (PCs), is as follows:

$$f(x) = g(x) + h(x) \quad (9)$$

where g(x) is the sum of the distance between the initial position and the current node (PC) being examined, and h(x) is the Euclidean distance (straight line calculated from stored coordinates) from the current node to the target node (PC FFs). The benefits of using A* over Dijkstra is covered next in the analysis section.

Tests and results

Path integration test

The first test we put the *ratbot* through was to test its PI capabilities without the use of any external stimuli data other than the data obtained from the ultrasonic sensors for barrier detection. The *ratbot* was programmed to record its present position in Cartesian coordinates from its travel vector at each barrier it came to in a 5 m × 5 m, open center, bordered area. At each boundary it came to, the *ratbot* would take a right turn of approximately 100°. After reaching a pre-programmed number of turns (2, 3, 4, 5 & 6), the *ratbot* calculated a vector to home, turned the appropriate number of degrees to obtain the calculated heading, then traveled the calculated distance to where it believed home to be, as illustrated by the red vectors in Fig. 3 for turns of 2 (R1), 3 (R2), and 4 (R3). A total of 12 runs were performed (a small sample size, but the outcomes were very representative of what was observed in other tests). The delta distance error from home and the stopping position of the *ratbot* ranged from 1.3 cm to 21.6 cm for an average of 8 cm.

Local path planning and goal memory demonstration

Morris water maze test

The *ratbot* was next tested in a variant of the Morris water maze (Morris, 1984; Vorhees & Williams, 2006) using all sensors (including its camera). Instead of a circular area, a 5 m × 5 m square area was used. Due to the *ratbot's* narrow, local FOV that is created with its camera facing down at a 40° incline, see Fig. 6, its ability to find the "hidden" platform can be thought of as being tactile rather than visual. Fig. 15 illustrates the dimensions of the local visibility of the *ratbot's* camera as currently situated, and as shown in Fig. 7b.

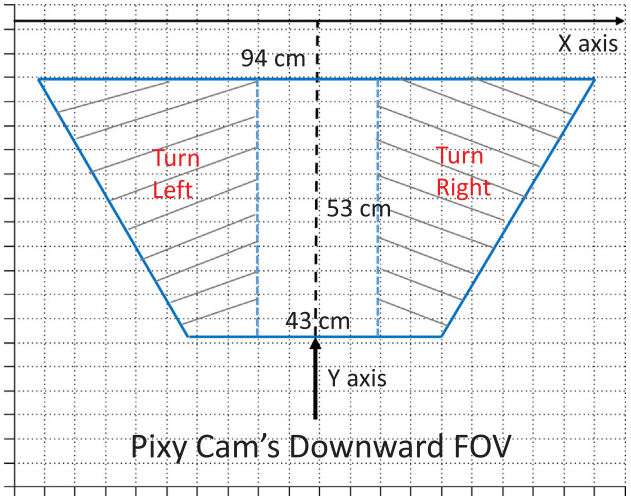

Fig. 15. Dimensions of the *ratbot's* FOV as seen by its camera in the 40° incline position.

The Morris water maze test environment was set up similar to that shown in Fig. 3. However, the "platform" or color-coded marker was placed at a diagonal to the home position. This created the greatest distance between the home and platform markers. Once in the explore mode, the *ratbot* found the platform along its search path, as it took approximately 100° turns from boundaries. After reaching the platform, the *ratbot* recorded the Cartesian coordinates for this newly defined PC FF, then calculated its return vector home. The *ratbot* then proceeded home. After reaching home, the *ratbot* calculated the vector to the platform, turned the appropriate angle and headed to, and reached, the platform again. Thus, demonstrating its quick learning and spatial awareness capabilities. This procedure is different from the original Morris maze task, such that the rodent is usually removed from the platform and placed back at the starting platform. This could be accomplished with the *ratbo*t, by using a PI reset button that is connect to the central processor. By pushing the PI reset after placing and realigning the *ratbot* to its initial pose at its home location, its initial position of (0, 0) and heading of 0° would be restored.

Single interior barrier test

The latest environment configuration the *ratbot's* navigation system has been tested in, separates a single goal location from home with an internal boundary structure, as illustrated in Fig. 16. The *ratbot* follows the exploration steps listed in the above Goal Memory subsection. Since a barrier is located between the goal and home, decision making logic is required for a wall-following mode which allows the *ratbot* to search for and find an open path home, as well as send BC FF data to the FPGA. In the process, a TC FF is created and stored in the place code, and is used by the goal memory for path planning.

This experiment exemplifies the use of the place code for a non-direct line of site path. As was observed in multiple runs of *ratbot* in this environment, the success of performing this wayfinding task, after the path has been learned, is primarily a function of the accuracy of the PI data. A rodent uses local cues, such as the visual identification of the barrier's corner to aid in its level of confidence and accuracy of following such a path. The VPCs are used by rodents to override PI data when possible (Arleo & Gerstner, 2000; Redish, 1999). This is true of the ratbot when it has a goal in its camera's FOV. Therefore, the barrier's corner acts as a signpost for rodents (Vorhees & Williams, 2014). The *ratbot's* currently does not have visual corner recognition (or other types of signposts) as part of its navigation system. Such data would tie into the navigation system's place code, as it does with rodents.

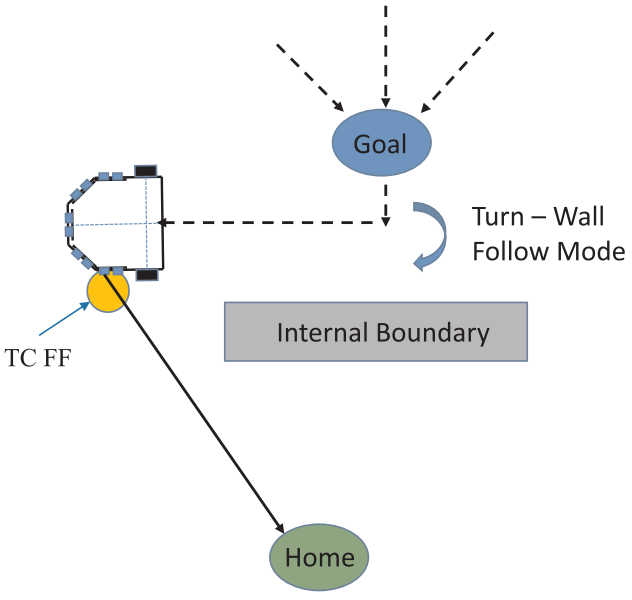

Fig. 16. Configuration of internal boundary test between goal and home locations.

Table 1
Power consumption of Ratbot's processors.

Processor type	Note	DC volts (V)	DC amperes (mA)	Power (mW)
Central Processor-Atmel ATmega2560	Plus 5 Ultrasonic Sensors & MEMS Gyro	5	128	640
Pixy Cam Dual Processor + Atmel ATmega328	Full Pixy Camera System	5	300	1500
Xilinx Spartan 6 XC6SLX9 FPGA		3.3	98.5	330
			Total power	2470

System analysis

Power consumption

One of the goals we set for this navigation system was to require relatively low power, such that the robot could easily carry the entire navigation system on a small footprint. The dimensions of the *ratbot* are approximately 21 cm long by 17 cm wide. The height of the chassis on the front of the *ratbot* is 25.5 cm. The *ratbot* currently carries two 12 V, 2000 mAh battery packs. However, only one of the batteries is currently used to power the entire system. The second battery is dedication to an Arduino Yun microprocessor board, which is used during debugging only. The Yun microcontroller board becomes a WiFi access point, which a laptop can connect to and receive debug data from. The average running voltage and currents for processors onboard the *ratbot* are tabulated in Table 1. An estimated power of 2.5 watts is being used by the processors (in addition to some sensors).

Most of the power consumed by the system is by the two 12 V motors, which can draw up to 530 mA each. That is a potential 12 W for the two motors alone. However, normal cruising speed for the *ratbot* is less than half of maximum and is used on a level, hard surface. Thus, the power used by the motors is likely to be 6 W at maximum, given the max power of the battery and the fact the *ratbot* can usually last 2 h or so, until the battery dips to approximately 6 V.

The low processing power required is attributed to the low operating frequencies of the processors. For processor power consumption is proportional to the operating voltage (squared) times the clock frequency. The Atmel microcontrollers run at 16 MHz, the Pixy Cam's onboard NXP LPC4330 dual core processor operates at 204 MHz, and the Xilinx Spartan 6 XC6SLX9 FPGA is run with a 50 MHz clock.

Navigation computations at a low frequency

The only time constraint on the main loop program's (Fig. 11) time is for the *ratbot* to stop and turn before hitting an obstacle. One can buy time by either slowing down the robot's movement, or giving enough buffer range to turn after detection. The *ratbot's* current loop time does not require much of an obstacle range detect buffer (20 cm) and the *ratbot's* current speed is moderate. The current main loop time is 325 ms. This value gives good reaction time and is the baseline for future modifications.

The vision system of the Pixy Cam can update at a rate of 50 frames per second, and the visual x, y coordinate data are obtained in a tight software loop in the interfacing microcontroller, which runs at 16 MHz. Performing the visual recognition task and turn control signals external to the central processor also decreases its impact on the main program's loop time.

The FPGA additionally removes processing time from the main loop program by managing the BC FFs. This would be very time consuming for the central processor in areas with many internal boundaries. Since this task is performed in parallel on the FPGA, without the need for feedback to the central controller, it is seamlessly out of the loop for the management task. However, when non-goal memory path planning is required, it would have been best if there was a processor core in the FPGA, along with external RAM. This would allow for line intersection calculations to be performed in the FPGA's CPU and not require the transmission of data back and forth between the FPGA and the central processor for this task. The need for a core processor in the FPGA is based on the fact that the FPGA doesn't configure well to using floating point numbers, trigonometric operations or the square root operation.

Computational complexity

As previously described, the occupancy grid method of environment mapping requires a set amount of memory to hold its map data. Depending on the amount of data used to record the occupancy of a cell (e.g., binary value, statistical data, or three-dimensional data), greatly affects the navigation system's processing requirements and capabilities. This is based on the need for full environment mapping, while many samples (multi-view, multi-sensor) of each cell location are taken, and Bayesian statistical analysis on that data. This does result in high accuracy maps which helps greatly with path planning. Therefore, occupancy grids are used in many mobile robot applications, such as driverless cars (Li & Ruichek, 2014) and the Mars rover (Volpe, Estlin, Laubach, Olson, & Balaram, 2000). However, high location accuracy is the cornerstone to making and using these maps. Path planning for the occupancy grid application is typically accomplished using Dijkstra's algorithm or its heuristic variant, the A* algorithm. The run complexity of these two algorithms are $O(|E| + |V| \log|V|)$ (minimum priority cue) and $O(|E|)$ respectively, where V is the number of nodes and E is the number of edges in the graph. The amount of memory required to perform these searches are equally large. Thus, searching for paths in occupancy grids that cover large areas is very demanding time and memory wise.

Our neurophysiologically based model trades detailed accuracy (high precision mapping of every inch of the environment) with a less exact and detailed method. However, that is how nature works. Thus, as detailed in our approach to assigning PC FFs to key locations only, the number of nodes (PC FFs) that need to be stored to memory are minimal. Additionally, checking for straight vectors to key locations allows for the possibility of finding an optimum path in very low searching time: O(number of extended BC FFs). Large environments, such as buildings can be subdivided into a connection of local maps, as long as their coordinate systems can be aligned at key entry and exit locations, or through the use of omnidirectional camera systems that use ANN for visual alignment to salient distal cues. This reduces the search time and memory required using the A* algorithm.

Discussion

We have demonstrated through the navigation tests performed above that the *ratbot* can successfully emulate the integration of HD cell (gyro) data with self-motion (wheel encoder) data to produce PI data. When the *ratbo*t approached previously found goal locations, any error in PI data was overridden with visual data. This was visually observed as the *ratbot* realigned itself from its calculated path, to a direct path to the goal marker. This is normal with many animal species and insects performing navigation tasks, as previously discussed. Additionally, a place code (linked list of PC structures) is generated using VPC, BC and PI data, as presented in our multimodal source model, and used by goal memory for path planning. Thus, the high level functional model of the rodent's navigation related brain cells, as presented in this paper, shows

promising results and serves as a good template for lower level detailed models.

The next step is to test out the *ratbot* in a larger, dynamic environment, where its planned path becomes blocked, causing for it to search for a new path starting from its last departed PC FF to its target location, while removing the blocked node (PC FF) from its search. Another important step is to test out the *ratbot's* ability to navigate in various environments, such as a maze and a building with many rooms. This will require an FPGA module with a built-in core processor and external memory to better handle the *ratbot's* path planning algorithm.

As our multimodal model shows, specialized navigation and spatial awareness cells of a rodent are dependent to some degree on visual cues (Burgess, Donnett, Jeffery, & John, 1997; Bush et al., 2014; Knierim & Hamilton, 2011; Redish, 1999; Winter & Taube, 2014). Of course, the caveat with using visual data in a mobile robot system, is the ability to process this data fast enough to be used in real-time. Additionally, information extraction requires deep learning neural networks (DNN), or similar, for image recognition. One possible solution is to perform general purpose processing on a graphics processor unit (GPGPU), such as DNN or convolutional neural networks (CNN). A relatively low power (5–15 W) GPGPU solution that might be powerful enough to perform these task is the NVIDIA® Jetson™ TX1 Module GPU with 256 light weight parallel processor (CUDA®) cores. They can be programmed using CUDA or cuDNN. Thus, the technology to perform lower power ANNs are becoming more available.

Possible future directions in model computation

Stanford University and Sandia National Laboratories have been working on creating a non-volatile organic electrochemical artificial synapse for neuromorphic computing (van de Burgt et al., 2017). This low-voltage, artificial synapse mimics the way neurons are connected in the brain. Thus, the neural inspired system could theoretically learn and keep its memory through the artificial synapse connectivity. Perhaps a neuromorphic computing machine, which more closely mimics the functionality of the brain than current processing systems, will be realized in the future. A system with elements that more closely resembles the dynamic learning structure of the human brain, and is similar with respect to processing capabilities, power requirements and size of the human brain.

Conclusions

We have presented and produced an efficient mobile navigation model that is based on the properties of the rodents' specialized navigation and spatial awareness cells. This was accomplished at a high level of abstraction, which allows for quick computations and a lower memory requirement as compared to similar neural network based systems and traditional mobile robot navigation systems previously discussed. Additionally, we implemented two path planning capabilities that best represent the speculated path planning capabilities of a rodent. The first uses goal memory, while the second combines PI, boundary memory, and the A* algorithm.

The use of data structures to represent PCs and BCs, and a gyroscope to represent HD cells worked out well for the small environment the robot was tested in. Not using ANNs for these rodent specialized navigation and spatial awareness cells did not affect the accuracy of how these cells naturally perform. However, VPCs need to be ANN based to give the robot a more natural way of identifying objects, classifying them, and learning them, without the need for engineering the environment. This is required to remove the PC and BC location accuracy dependency from purely PI and egocentric landmark recognition data, to more of an allocentric distal cue recognition system for relative positioning.

References

Arleo, A., & Gerstner, W. (2000). Spatial cognition and neuro-mimetic navigation: A model of hippocampal place cell activity. *Biological Cybernetics, 83*(3), 287–299.
Arleo, A., Smeraldi, F., & Gerstner, W. (2004). Cognitive navigation based on nonuniform Gabor space sampling, unsupervised growing networks, and reinforcement learning. *Neural Networks, IEEE Transactions on, 15*(3), 639–652.
Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine, 13*(3), 108–117.
Barrera, A., & Weitzenfeld, A. (2007). Rat-inspired model of robot target learning and place recognition. Paper presented at the Control & Automation, 2007. MED'07. Mediterranean Conference on.
Barrera, A., & Weitzenfeld, A. (2008). Computational modeling of spatial cognition in rats and robotic experimentation: Goal-oriented navigation and place recognition in multiple directions. Paper presented at the Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS & EMBS International Conference on.
Barry, C., & Burgess, N. (2014). Neural mechanisms of self-location. *Current Biology, 24*(8), R330–R339.
Boucheny, C., Brunel, N., & Arleo, A. (2005). A continuous attractor network model without recurrent excitation: Maintenance and integration in the head direction cell system. *Journal of Computational Neuroscience, 18*(2), 205–227.
Burak, Y., & Fiete, I. R. (2009). Accurate path integration in continuous attractor network models of grid cells. *PLoS Computational Biology, 5*(2), e1000291.
Burgess, N. (2008). Grid cells and theta as oscillatory interference: Theory and predictions. *Hippocampus, 18*(12), 1157–1174.
Burgess, N., Donnett, J. G., Jeffery, K. J., & John, O. (1997). Robotic and neuronal simulation of the hippocampus and rat navigation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences, 352*(1360), 1535–1543.
Burgess, N., Maguire, E. A., & O'Keefe, J. (2002). The human hippocampus and spatial and episodic memory. *Neuron, 35*(4), 625–641.
Burgess, N., Recce, M., & O'Keefe, J. (1994). A model of hippocampal function. *Neural Networks, 7*(6), 1065–1081.
Bush, D., Barry, C., & Burgess, N. (2014). What do grid cells contribute to place cell firing? *Trends in Neurosciences, 37*(3), 136–145.
Darwin, C. (1873). Origin of certain instincts. *Nature, 7*, 417–418.
Derdikman, D. (2009). Are the boundary-related cells in the subiculum boundary-vector cells? *The Journal of Neuroscience, 29*(43), 13429–13431.
Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. *Robotics & Automation Magazine, IEEE, 13*(2), 99–110.
Erdem, U. M., & Hasselmo, M. (2012). A goal-directed spatial navigation model using forward trajectory planning based on grid cells. *European Journal of Neuroscience, 35*(6), 916–931.
Erdem, U. M., & Hasselmo, M. E. (2014). A biologically inspired hierarchical goal directed navigation model. *Journal of Physiology-Paris, 108*(1), 28–37.
Filliat, D., & Meyer, J.-A. (2003). Map-based navigation in mobile robots: I. A review of localization strategies. *Cognitive Systems Research, 4*(4), 243–282.
Franz, M. O., & Mallot, H. A. (2000). Biomimetic robot navigation. *Robotics and Autonomous Systems, 30*(1–2), 133–153. http://dx.doi.org/10.1016/S0921-8890(99)00069-X.
Fyhn, M., Hafting, T., Treves, A., Moser, M.-B., & Moser, E. I. (2007). Hippocampal remapping and grid realignment in entorhinal cortex. *Nature, 446*(7132), 190–194.
Fyhn, M., Molden, S., Witter, M. P., Moser, E. I., & Moser, M.-B. (2004). Spatial representation in the entorhinal cortex. *Science, 305*(5688), 1258–1264.
Giocomo, L. M., & Hasselmo, M. E. (2008). Computation by oscillations: Implications of experimental data for theoretical models of grid cells. *Hippocampus, 18*(12), 1186–1199.
Goldschmidt, D., Dasgupta, S., Wörgötter, F., & Manoonpong, P. (2015). A neural path integration mechanism for adaptive vector navigation in autonomous agents. Paper presented at the Neural Networks (IJCNN), 2015 International Joint Conference on.
Gonzalez-Arjona, D., Sanchez, A., López-Colino, F., de Castro, A., & Garrido, J. (2013). Simplified occupancy grid indoor mapping optimized for low-cost robots. *ISPRS International Journal of Geo-Information, 2*(4), 959.
Grisettiyz, G., Stachniss, C., & Burgard, W. (2005). Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. Paper presented at the Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on.
Hafner, V. V. (2005). Cognitive maps in rats and robots. *Adaptive Behavior, 13*(2), 87–96.
Hafner, V. V. (2008). Robots as Tools for Modelling Navigation Skills – A Neural Cognitive Map Approach Robotics and cognitive approaches to spatial mapping (pp. 315–324). Springer.
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. http://dx.doi.org/10.1038/nature03721.
Hardcastle, K., Ganguli, S., & Giocomo, L. M. (2015). Environmental boundaries as an error correction mechanism for grid cells. *Neuron, 86*(3), 827–839.
Hayman, R., & Burgess, N. (2015). How cumulative error in grid cell firing is literally bounded by the environment. *Neuron, 86*(3), 607–609.
Jauffret, A., Cuperlier, N., & Gaussier, P. (2015). From grid cells and visual place cells to multimodal place cell: A new robotic architecture. *Frontiers in Neurorobotics*, 9.
Kjelstrup, K. B., Solstad, T., Brun, V. H., Hafting, T., Leutgeb, S., Witter, M. P., ... Moser, M.-B. (2008). Finite scale of spatial representation in the hippocampus. *Science, 321*(5885), 140–143.
Knierim, J. J., & Hamilton, D. A. (2011). Framing spatial cognition: Neural representations of proximal and distal frames of reference and their roles in navigation.

Physiological Reviews, 91(4), 1245–1279. http://dx.doi.org/10.1152/physrev.00021.2010.

Konolige, K., Marder-Eppstein, E., & Marthi, B. (2011). Navigation in hybrid metric-topological maps. Paper presented at the Robotics and Automation (ICRA), 2011 IEEE International Conference on.

Lever, C., Burton, S., Jeewajee, A., O'Keefe, J., & Burgess, N. (2009). Boundary vector cells in the subiculum of the hippocampal formation. *The Journal of Neuroscience, 29*(31), 9771–9777.

Li, Y., & Ruichek, Y. (2014). Occupancy grid mapping in urban environments from a moving on-board stereo-vision system. *Sensors, 14*(6), 10454–10478.

Little, K. C. (2007). A rat model of systemic chemotherapy for breast cancer to evaluate and treat chemobrain. Retrieved from.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., & Moser, M.-B. (2006). Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience, 7*(8), 663–678.

Meyer-Delius, D., Beinhofer, M., & Burgard, W. (2012). Occupancy grid models for robot mapping in changing environments. Paper presented at the AAAI.

Morris, R. (1984). Developments of a water-maze procedure for studying spatial learning in the rat. *Journal of Neuroscience Methods, 11*(1), 47–60.

Moser, E. I., Kropff, E., & Moser, M.-B. (2008). Place cells, grid cells, and the brain's spatial representation system. *Neuroscience, 31*(1), 69.

Moser, E. I., & Moser, M. B. (2008). A metric for space. *Hippocampus, 18*(12), 1142–1156.

Moser, E. I., Roudi, Y., Witter, M. P., Kentros, C., Bonhoeffer, T., & Moser, M.-B. (2014). Grid cells and cortical representation. *Nature Reviews Neuroscience, 15*(7), 466–481.

Müller, M., & Wehner, R. (1988). Path integration in desert ants, Cataglyphis fortis. *Proceedings of the National Academy of Sciences, 85*(14), 5287–5290.

O'Keefe, J., & Dostrovsky, J. (1971). The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain Research, 34*(1), 171–175.

Redish, A. D. (1999). Beyond the cognitive map: From place cells to episodic memory. MIT Press Cambridge, MA.

Sargolini, F., Fyhn, M., Hafting, T., McNaughton, B. L., Witter, M. P., Moser, M.-B., & Moser, E. I. (2006). Conjunctive representation of position, direction, and velocity in entorhinal cortex. *Science, 312*(5774), 758–762.

Sariff, N., & Buniyamin, N. (2006, 27–28 June 2006). An overview of autonomous mobile robot path planning algorithms. Paper presented at the Research and Development, 2006. SCOReD 2006. 4th Student Conference on.

Shipston-Sharman, O., Solanka, L., & Nolan, M. F. (2016). Continuous attractor network models of grid cell firing based on excitatory–inhibitory interactions. *The Journal of Physiology*.

Stackman, R. W., Clark, A. S., & Taube, J. S. (2002). Hippocampal spatial representations require vestibular input. *Hippocampus, 12*(3), 291–303.

Stewart, S., Jeewajee, A., Wills, T. J., Burgess, N., & Lever, C. (2014). Boundary coding in the rat subiculum. *Philosophical Transactions of the Royal Society B: Biological Sciences, 369*(1635), 20120514.

Strösslin, T., Chavarriaga, R., Sheynikhovich, D., & Gerstner, W. (2005). Modelling path integrator recalibration using hippocampal place cells. Paper presented at the International Conference on Artificial Neural Networks.

Strösslin, T., Sheynikhovich, D., Chavarriaga, R., & Gerstner, W. (2005). Robust self-localisation and navigation based on hippocampal place cells. *Neural Networks, 18*(9), 1125–1140.

Taube, J. S. (2007). The head direction signal: Origins and sensory-motor integration. *Annual Review of Neuroscience, 30*, 181–207.

Thrun, S. (2002). Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millennium, 1*, 1–35.

Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous Robots, 15*(2), 111–127.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review, 55*(4), 189.

Trullier, O., Wiener, S. I., Berthoz, A., & Meyer, J.-A. (1997). Biologically based artificial navigation systems: Review and prospects. *Progress in Neurobiology, 51*(5), 483–544.

van de Burgt, Y., Lubberman, E., Fuller, E. J., Keene, S. T., Faria, G. C., Agarwal, S., ... Salleo, A. (2017). A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nature Materials*.

Volpe, R., Estlin, T., Laubach, S., Olson, C., & Balaram, J. (2000). Enhanced mars rover navigation techniques. Paper presented at the Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on.

Vorhees, C. V., & Williams, M. T. (2006). Morris water maze: Procedures for assessing spatial and related forms of learning and memory. *Nature Protocols, 1*(2), 848.

Vorhees, C. V., & Williams, M. T. (2014). Assessing spatial learning and memory in rodents. *ILAR Journal, 55*(2), 310–332.

Wallgrün, I. J. O. (2010). Robot mapping hierarchical Voronoi graphs (pp. 11–43). Springer.

Winter, S. S., & Taube, J. S. (2014). Head direction cells: From generation to integration. In Space, time and memory in the hippocampal formation (pp. 83–106). Springer.

Wolf, H. (2011). Odometry and insect navigation. *Journal of Experimental Biology, 214*(10), 1629–1641.

Wyeth, G., & Milford, M. (2009). Spatial cognition for robots. *Robotics & Automation Magazine, IEEE, 16*(3), 24–32. http://dx.doi.org/10.1109/mra.2009.933620.

Zeno, P. J. (2015). Emulating the functionality of rodents' neurobiological navigation and spatial cognition cells in a mobile robot. *International Journal of Computing, 14*(2), 77–85.

Zeno, P. J., Patel, S., & Sobh, T. M. (2016). Review of neurobiologically based mobile robot navigation system research performed since 2000. *Journal of Robotics, 2016*, 17. http://dx.doi.org/10.1155/2016/8637251.