ABSTRACT

A Neural Inspired Grid Cell Aid for Robot Inertial Navigation

Moises Martinez, M.S.E.C.E.

Chairperson: Scott Koziol, Ph.D.

Accurate position information is needed for a robot's guidance and control systems. Therefore, the navigation system is foundational for a robot's interactions with the world. This thesis explores a biologically inspired method of navigation with the goal of improving navigational accuracy. The desire to integrate brain-inspired methods with conventional signal processing methods is based on animals' innate ability to successfully navigate through habitats. Therefore, it is reasonable to explore the way animals process brain signals to navigate and leverage it for robot navigation. This thesis uses a combinatorial model of map formation and localization with grid cells to aid dead reckoning with an accelerometer and gyroscope to show grid cells are a viable aid in idiothetic navigation. The results show that the grid cell aided navigation systems shows better performance with longer paths and higher noise values with an improvement of 57.24 cm for a 32 m path with 3 $\sigma$ noise.

A Neural Inspired Grid Cell Aid for Robot Inertial Navigation

by

Moises Martinez, B.S.E.C.E.

A Thesis

Approved by the Department of Electrical and Computer Engineering

_____

Kwang Lee, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Electrical and Computer Engineering

Approved by the Thesis Committee

_____

Scott Koziol, Ph.D., Chairperson

_____

Jonathan Hu, Ph.D.

_____

Douglas Smith, Ph.D.

Accepted by the Graduate School
May 2018

_____

J. Larry Lyon, Ph.D., Dean

*Page bearing signatures is kept on file in the Graduate School.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# CHAPTER ONE

# Introduction

## *Background and Motivations*

A robot does not know where it is without a navigation system on board. The navigation system answers the question a robot's user has of where it is in the world. The navigation system allows a robot to output where its position is in the world (in the form of an x-y-z plane coordinate location relative to the world) along with additional information such as the orientation of the robot in yaw, pitch, and roll angles.

The robot's navigation system information is necessary for higher level operations performed in a robot's guidance system and lower level operations performed in a robot's control system. These two systems, respectively, plan the robot's path from its current location to its goal and instructs the motors to move according to the path planned. Figure 1.1 pictorially describes the roles of the Guidance, Navigation, and Control systems, and 1.1 defines the systems' properties.

Accurate position information is needed for a robot's guidance and control systems. Therefore, the navigation system is a foundation for a robot's interactions with the world. This thesis explores a biologically inspired method of navigation to improve a navigation system's accuracy.

The desire to integrate brain-inspired methods with conventional signal processing methods is based on the fact that animals show an innate ability to successfully navigate through their habitats. Therefore, it is reasonable to explore the way animals and humans process the brain signals to navigate and leverage it for robot navigation.

Table 1.1 This table shows the function, question answered, and the standard method of execution of the guidance, navigation, and control systems that are used by robots as it moves around the environment [3].

| | **Guidance** | **Navigation** | **Control** |
|---|---|---|---|
| **Function** | Path Planning | Determine Robot's State | Tracks Guidance Commands and Maintains Stability |
| **Question Answered** | What roads should I take? | Where am I now? | How do I adjust acceleration and steering? |
| **Standard Method** | A* (A-star) | Kalman Filter | Proportional Integral Derivative (PID) Controller |

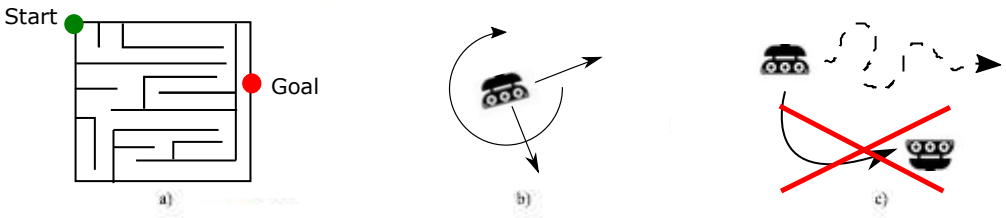



Figure 1.1 This figure describes the individual roles of a robot's guidance, navigation, and control systems. a) Representation of a guidance system that performs a high level process of planning a path from the initial position to the goal position. b) Representation of a navigation system that uses sensor inputs to determine its location. c) Representation of a control system that performs a low level process of tracking commands from the guidance system, maintains stability, and other functions to keep the robot moving [1].

### *Conventional Robot Navigation Methods*

There are three main methods of robot navigation. First, robots can navigate using only internally sensed movement. This methods is called "dead reckoning" [4] [5]. The sensors measure forces like linear acceleration or rotational velocity that act on the robot. Then they use algorithms with those read forces to calculate the position of the robot. Position can also come from sensors in the form of wheels with rotary encoders [6]. Dead reckoning navigation systems which rely on accelerometer and gyroscope measurements are called Inertial Navigation Systems (INS) [5].

A second method is navigation through the use of external information. Proximity sensors, infrared sensors, cameras, global positioning systems, maps, etc. are all technologies that rely on external stimulation to help robots know where they are [7] [8] [4]. These readings can give numbers to offset the position reading directly or through filters in order to be more accurate in navigation.

The third method deals with pattern recognition methods such as the Simultaneous Localization and Mapping (SLAM) method of navigation. [9] [10] [11] [12]. This method of navigation relies on the ability to identify landmarks, features, and other unique identifiers around the navigation space to form a map. The map corrects the position as the robot navigates through the space, and makes position estimation more accurate.

The navigation system used in this thesis is a dead reckoning system where the robot's movement is sensed using an on-board INS that has accelerometers and gyroscopes. A benefit of dead reckoning systems is that they are immune to jamming or lack of landmarks and features in the environment. This means they work in environments where Global Positioning System (GPS) signals are not available such as underwater, inside buildings, inside caves, etc. and if environments cannot be measured with external sensors.

### *Biological Navigation Methods*

Dogs, pigeons, salmon, and other animals navigate across tens to thousands of kilometers to follow migration patterns, send messages, and find their way back home [13] [14] [15] [16]. Mice navigate through mazes on their first attempt in lab environments [17]. Humans show that their position estimations can build a cognitive map for navigation similarly to SLAM systems [18] [19] [20].

Animal's hearing, touch, and other pseudo-sensors on their bodies all work together to let animals know where they are in their habitats without the need of

maps or landmarks [21] [16] [22] [23]. Rats were observed to navigate through a space without visual or auditory cues to their goals [17]. In most mammals, there is a vestibular system that acts as the pseudo-sensor. It feeds orientation and velocity to feed information to the brain [24]. Special neurons in the brain called *grid cells* receive information from the vestibular system and have been shown to be reliable for navigation without external cues [21].

There are approximately $10^5$ grid cells in a rat's brain [21]. When a grid cell fires, it provides the rat with information on a possible position. For illustrate purposes, call this "Set A" of locations. When a different grid cell fires, it creates a second set of possible locations which can be called "Set B".

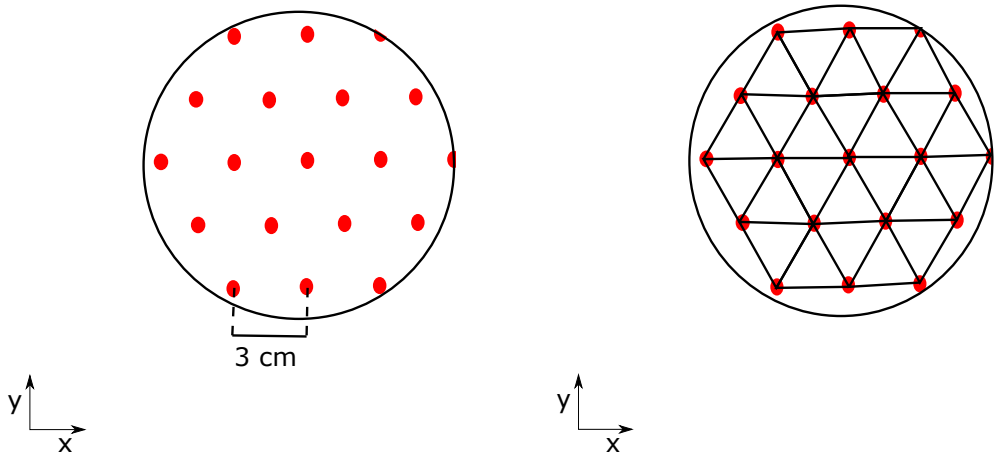
Assume more grid cells fire until you have Sets A through Z where each provides its set of location information. If you combine Set A with Set B, there will be positions that match from both sets. Those matching positions narrow down possible locations and form a separate "Set $\mu$". You can combine up to 26 of the grid cell location sets A-Z, until the set of possible locations $\mu$ narrows down to only one position. Then $\mu$ is the position of the rat.

*The key idea of grid cells* is that each set of locations from individual grid cells is not sufficient to determine a unique position. However, the sets can be combined together to determine a unique position.

The reason why grid cells can form sets is because grid cells fire at even intervals as a rat travels through space. The firing intervals have been observed to be consistent even after the rat travels 10 minutes and over 100 m [17]. The observation comes from probing a specific section of the brain called the medial entorhinal cortex (MEC).

When one records grid cells as they fire in the (MEC), they form tessellating triangles as shown in Figure 1.2 [21] [17]. A collection of the firing patterns from one grid cell across the navigational space is called the module or lattice of the grid

cell. Different grid cell modules can combine in such a way that they help an animal localize. The consistent way modules form and grid cells fire allow them to be modeled.




(a) (b)

Figure 1.2 Neuroscientists can monitor the firing patterns of a single grid cell in a rat's brain as the rat moves across a space. The dots represent a grid cell firing which can be mapped to a physical position by recording the rat's location in the environment when a firing occurs. The circles are a section of the firing pattern of the grid cell, and the red dots represents a position mapping. a) shows a grid cell that fires every 3 cm. b) shows the repeated triangles the firing from one grid cell form.

*Thesis Topic*

The consistency of grid cells allowed for algorithmic models to be formed from them. There are combinatorial models to extract positions from the grid cells as well. An INS model can be used in place of a vestibular system. This means that a grid cell navigation system can be formed for a robot. Existing grid cell applications in robotics can be seen in [25] and in a form of simultaneous localization and mapping inspired by computational models of the hippocampus of rodents (RatSLAM) [12]. While those robots use grid cells for navigation, they also use cameras to look around their environment to form a map, and adjust position while navigating through a space.

This thesis aims to answer the question:

"Does the addition of brain-inspired grid cell signal processing to a conventional dead reckoning system improve the accuracy of its localization?"

### *Content Structure of this Thesis*

This thesis is organized as follows.

*Chapter Two*

The second chapter of this thesis provides more details of navigation systems. The first part of the chapter describes more details of INS based robot navigation, and the second part describes more details of neuron based navigation.

*Chapter Three*

The third chapter of this thesis describes the algorithms used in estimating position, orientation, and location in a navigation frame. It also details the algorithmic method in which grid cells can be used to aid in the localization of the robot.

*Chapter Four*

The fourth chapter of this thesis describeshow the algorithms are applied in simulation. Then the results from the simulation are displayed and analyzed.

*Chapter Five*

The last chapter of the this thesis discusses the simulation results and how viable grid cell usage is in simulation. It also discusses possible ways to further expand on the results from this thesis.

# CHAPTER TWO

## Related Work

### *Robot Navigation*

Robots cannot navigate without external or internal sensors and systems, so mobile robots have a hierarchy of systems for movement. A path planning system guides the robot's movement. Error correction methods like filters, maps, cameras, etc. help robots stay on path. At the base of its navigation systems, a robot can use its movement's direction and acceleration to estimate its displacement from a starting point [4].

The direction and acceleration are retrieved from from an accelerometer and gyroscope that are the foundation of a robot's navigation system. Accelerometers and gyroscopes exhibit errors in their readings [5] [26] [27]. Noise is the biggest contributor to the sensor's error [4]. Noise is a random error that cannot be predicted or removed completely. The noise limits consistency when running repeated trials [4].

Another contribution to the sensor's error is its bias. The sensor's manufacturing process causes defects in the sensor. The defects make an offset that is consistent throughout the navigation process [4]. This error accrues in the navigation calculations.

The previous sensor errors need to be filtered in order to accurately estimate the robots position. If the errors are not filtered out, then the measured position differs from the true position. The difference, as the errors accrue, is called drift. The longer a robot runs, the more the drift increases as can be seen in Figure 2.1 [5] [27].

Methods of removing the errors have been used to make sensors more accurate in their measurement [28]. One method is to use more sensors within the navigation systems [27], and another is by using lasers, cameras, etc. to add "sight" and correct

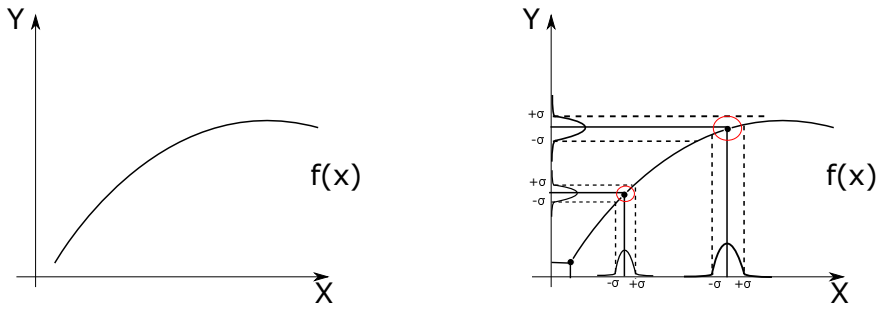



Figure 2.1 This figure illustrates an inertial navigation system's growing position uncertainty as time progresses. a) This shows the position estimation as a robot travels a curved path. b) has circles around different points of the estimation shown in a). The circle grows as time passes, giving a visual cue to the error accruing.

for deviation while navigating [25]. Different combinations of sensors have also been used to improve position estimations in navigation systems with varying levels of increased accuracy [8] [29] [30].

Numerical path integration models and other mathematical approximations for position can be used to estimate a robot's position [25] [28] [31]. For this reason, navigation has become strongly reliant on Kalman, FIR, and other filtering methods in order to maintain an accurate position estimation. Kalman filters are commonly used in robotics, and they use the readings from a GPS in order to keep drift low. However, navigation in robotics would benefit greatly if there was a localization system that could be applied to dead reckoning to make it more accurate without relying on external information like maps or landmarks. Grid cells do not need that external information, thus fitting the specifications for the goal of this thesis to improve dead reckoning without using external information.

### *Neuron Based Navigation*

Grid cell research started in 1972 when it was proposed that the hippocampus could be a spatial map [32]. It was observed that "rats with hippocampal damage are reported to be hyperactive in novel environments, ..., and poor at spatial tasks such as

mazes and tasks which require the alteration of responses on successive trials [32]." Tests were done on the hippocampus of rats to show brain activity as rats moved around their habitats and navigated through mazes.

The properties of the activity in the rats' brain were recorded through similar tests, and neurons were found to fire in another part of the brain called the medial entorhinal cortex (MEC). Research also showed that the neurons of the MEC fire consistently in a triangular pattern and are not interrupted [33]. These neurons that consistently fired were given the name "grid cells", and the hippocampal neurons were called "place cells". The grid cells persisted despite what happened with the place cells. The grid cell's triangular pattern and their behavior suggested that grid cells are "a part of a generalized, path-integration-based map of a spatial environment [34]." The development, behavior, and characteristics of place cells and grid cells continues to be studied and some other discoveries can be read in [35], [36], [37], [38], [39], and [40].

Neuron properties (size, signal transfers, chemical handling, shape, etc.) differ according to their function [41] [42]. Despite this, models have been established to show how neurons like grid cells generally work. An electronics-based model for neurons and their transfer of energy can be modeled using capacitors [43].

Neurons have a resting potential that they hold when no stimulation happens. Similarly, a capacitor will maintain a charge when isolated from a current sink or source. A neuron will charge up, like a capacitor hooked into a source, when it receives an electrical impulse. As long as the neuron continues to receive electrical impulses from other neurons, the neuron will charge up until it reaches a certain threshold voltage. Once it reaches that threshold, the neuron will fire, discharging itself, but exciting another neuron. This is similar to when a capacitor's source is removed, and it is connected to a sink. It stays at a resting potential until it receives another excitation. The similarity is visually represented in Figure 2.2.

This capacitor-like behavior is seen in the different types of neurons as animals navigate through their habitats when migrating, foraging, and their usual activities. Oculomotor neurons and head direction neurons found in vertebrates create neuron signals with the temporal deflections of the eye [44]. Neurons in the rodent's subiculum fire when its head points in a certain angular range [22]. The antenal lobe of insects identify odors with constantly evolving firing patterns [23].

Grid cells hold their own behavior and feed into place cells [34]. Grid cells are located in the dorsolateral band of the medial entorhinal cortex, and they fire when a rat is on a vertex of a virtual triangular lattice overlaid on the surface of their navigation space. These firings suggest that the grid cells encode an internal estimate of the rat's position [21] [45] [46] [47] [48]. Mice have been placed in dark rooms and navigated through mazes without the use of visual or auditory cues to prove this point [17].

Grid cells studies reveal that rats can travel up to a kilometer without getting lost in limited conditions, even through the triangular lattices cover no more than 10 m [21]. These patterns show position in a resolution as low as 25 cm after navigating long distances [47] [34]. The analysis of the grid cells firing as a rat moves across a two-dimensional plane revealed that there is a relationship between the recordings of all grid cells firing and the position of a rat. Grid cells form a neural "code that is fully combinatorial in capacity [21]." Combinatorial models have been developed to emulate the grid cell's neural code that keeps rat navigation accurate [21].

### *Related Work Summary*

In summary, combinatorial models for grid cells have been adopted into navigation systems. Path planning algorithms have been developed with grid cells [2]. Simultaneous localization and mapping inspired by computational models of the hippocampus of rodents (RatSLAM) have been developed in [12]. Grid cells have been

used alongside visual cues from cameras to form a map in an environment as the robot navigates through a certain space [25]. This "visual odometry" allows for correction while navigating and map building for more accurate navigation on repeated runs. Grid cells are used as a formation for place cells along with some sensors to navigate and map through a space simulataneously.

Unlike [12] and [25], this thesis does not use external information from cameras for navigation or combine grid cells with place cells. This thesis uses a grid cell combinatorial model of map formation and localization inspired by [21] to aid dead reckoning with an accelerometer and gyroscope in order to show that grid cells are a viable aid in idiothetic navigation.

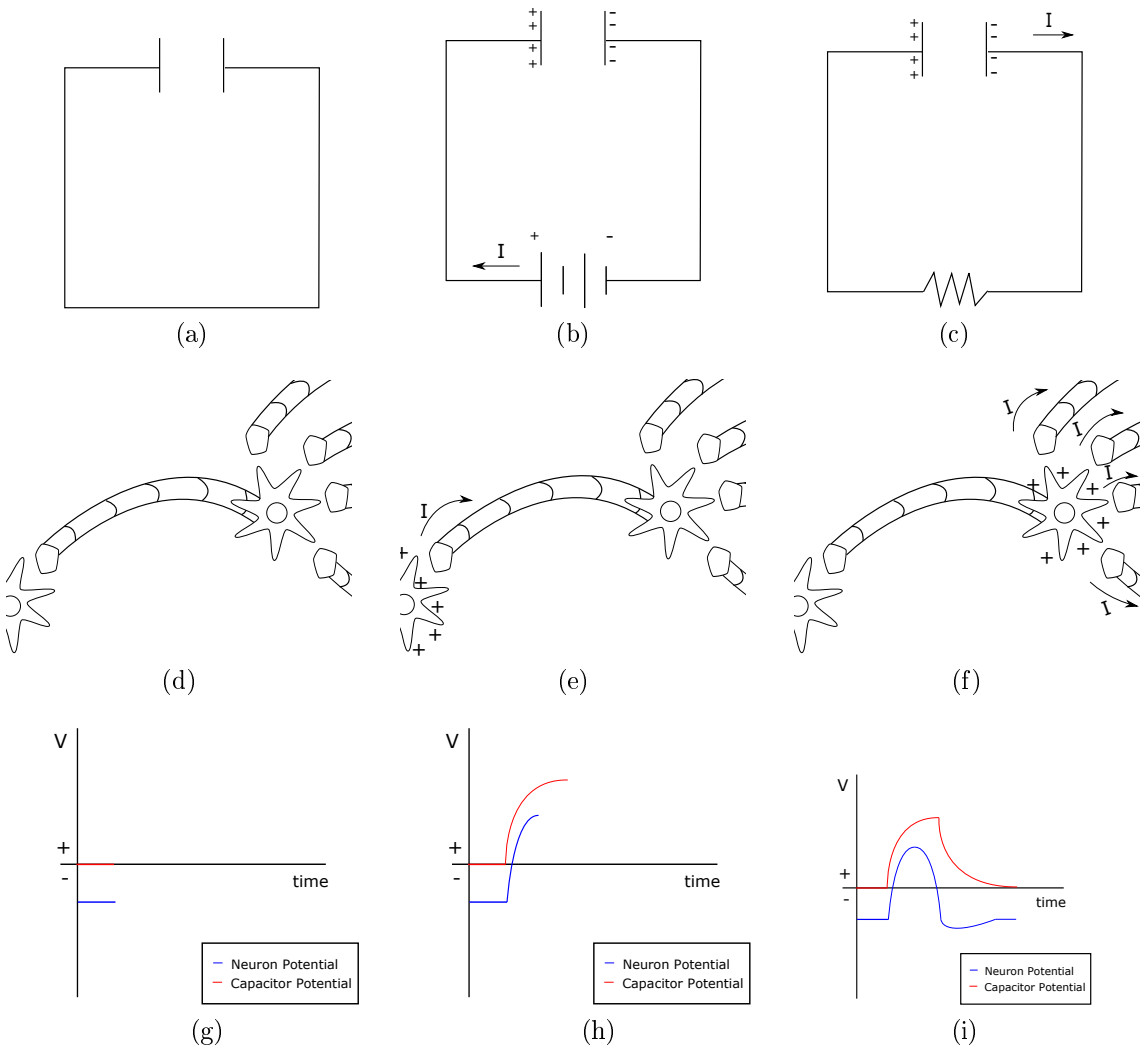



Figure 2.2 Comparing neurons with capacitors, (a), (d), and (g) represent the starting potential. (b), (e), and (h) show the neuron and the capacitor charging when excited. (c), (f), and (i) show the discharge. The figure has gone through an excitation of a neuron from start to complete discharge and is shown to behave similarly to a capacitor.

# CHAPTER THREE

## Algorithms

The methods involved in navigating with grid cells, as have been described in Chapters 1 and 2, are explained in this chapter. First, a mathematical explanation of grid cell firing is presented. This is followed by combinatorial methods in which grid cells form a spatial map and correct location through it. Finally, the mathematics of inertial navigation are presented.

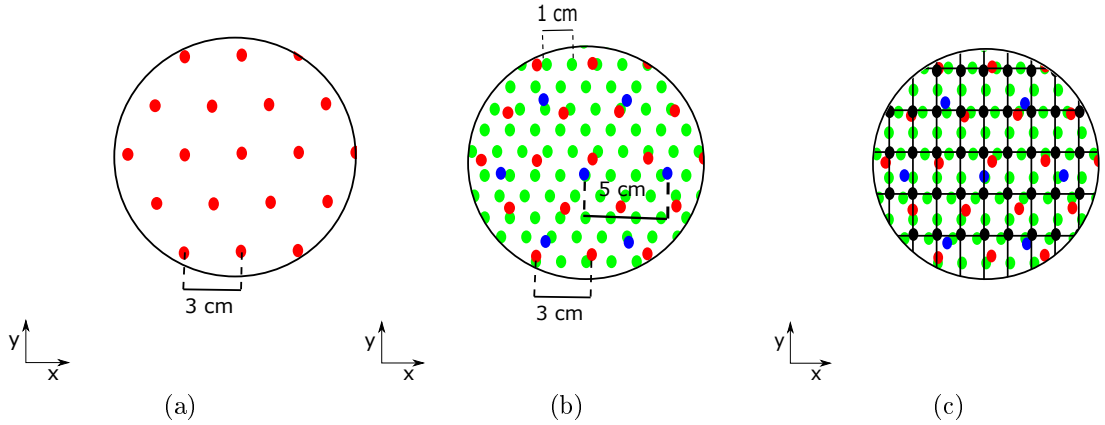
### *Grid Cell Mapping*

Grid cells form a spatial map from their firing patterns in the medial entorhinal cortex. The firing patterns form triangular lattices and combinatorially produce position estimates across the navigation space [2]. The following algorithms model the formation of spatial maps through grid cells [17].

#### *Map Position Estimation*

The medial entorhinal cortex is an idiothetic (or based off of its self motion rather than external stimuli like landmarks) path integrator [47] [17]. Grid cells also hold their signals for as long as 10 minutes [17] even when no positional information is gathered by the grid cells [21]. Each grid cell produces spikes periodically at a set amount of displacement as they move across space. The combination of all spikes observed encode space as shown in Figure (3.1) [21].

With the stability of the grid cell's firing pattern and its periodicity, grid cell's firings pattern can be transformed into the Cartesian space as shown in Figure 3.1.




(a) (b) (c)

Figure 3.1 a) shows one grid cell's firing pattern. b) shows multiple grid cell's firing patterns. c) shows an x-y grid over the firing patterns. Together, they show the progression of encoding positions through multiple grid cell's firing patterns as shown in equations (3.1) and (3.5).

*1-Dimensional Grid Cell Mapping*

Each grid cell fires off repeated patterns that are called "lattices". The interval at which each lattice forms is called "lattice periods." Lattice periods are measured in centimeter spacings, with lattice periods ranging from 20 cm to 200 cm [2]. Each grid cell's lattice and periods can be expressed algorithmically (3.1) [21].

$$\chi_\alpha(x) = \frac{mod(x, \lambda_\alpha)}{\lambda_\alpha} \cdot 2\pi \tag{3.1}$$

$\chi_\alpha$ is a phase representation of the location at which the grid cell is located. $x$ is the internal estimate of the location of the grid cell's position in space. $\lambda_\alpha$ is the lattice period of grid cell $\alpha$. $\alpha$ is an index number for the grid cell.

There are an $N$ number of grid cells throughout the medial entorhinal cortex. Each spatial position is represented by a set (3.2) of grid cell phase mappings (3.1) of its position [21].

$$X = (\chi_1(x), \chi_2(x), ..., \chi_N(x)) \tag{3.2}$$

The phase to spatial position mapping for one grid cell (3.1) are graphically represented in Figure 3.2.

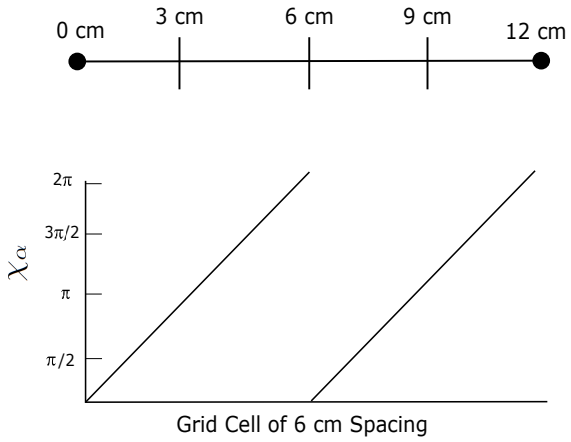



Figure 3.2 A grid cell firing every 6 cm is converted into a phase. However, pi represents the locations 3 cm and 9 cm. This means each phase represents an ambiguous location on the axis for only one grid cell as described in (3.1).

The set of spatial position mapping (3.2) is shown in Figure 3.3.

$$\vec{X}(x(t)) = (X_1, X_2, ..., X_N) \tag{3.3}$$

Equation (3.3) shows all positions represented on a one-dimensional vector by grid cells [21]. Each $X(x(t))$ is a position as represented by (3.2). Figure 3.4 shows the mapping of one axis with three grid cells.

Positions mapped will be limited by the capacity of grid cells to encode. So if $\lambda_1 = 3$ and $\lambda_2 = 4$, then 12 distinct positions can be represented. In general lattices will only have a range from 0 to $D$, where $D$ is the capacity of displacement the grid cells can encode to (3.4) [21].

$$D = LCM\{\lambda_\alpha | \alpha = 1, .., N\} - 1 \tag{3.4}$$

LCM stands for the least common multiplier operation. Figure 3.5 shows that two positions are mapped with the same phases, thus 12 cm could be read by those three grid cells as 0 cm and vice versa.

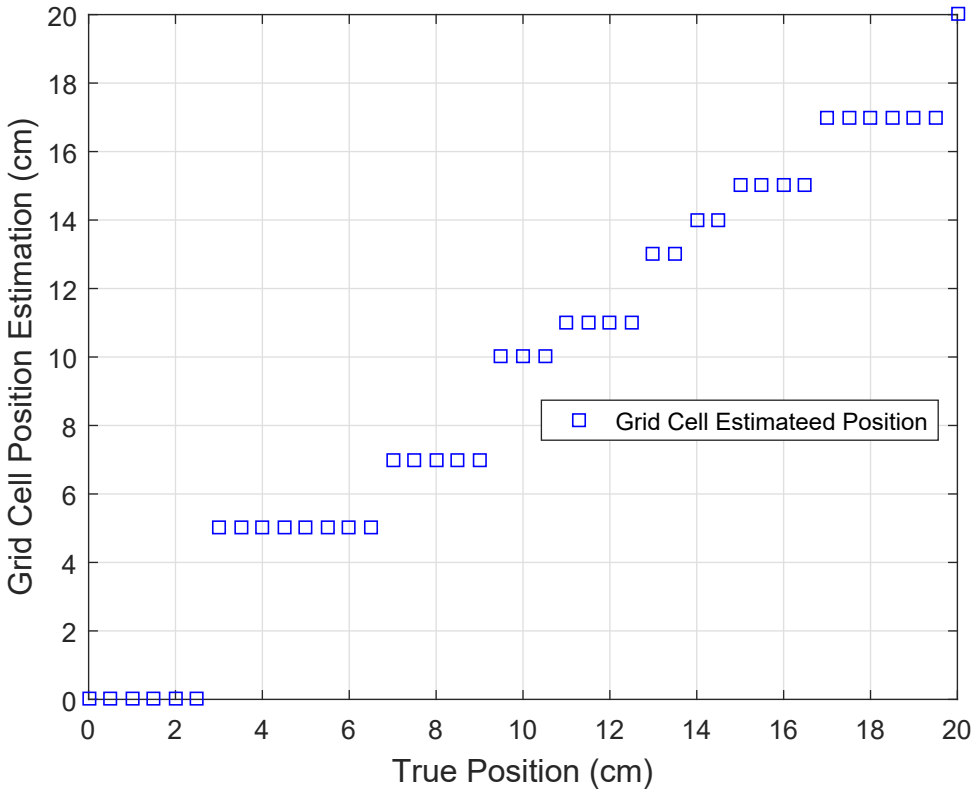



Figure 3.3 Grid Cell Estimation vs Truth: When multiple grid cells are fired at the same locations, they each provide their own phase. This means phases provided by different grid cells, when combined, represents location on the axis as described in (3.2). This figure shows what location a grid cell encodes to (on the y-axis) at every .5 cm along the axis with 6 grid cells the lattices sizes of 5, 7, 11, 13, 17, and 23 cm as used to provide the results presented in Chapter 4.

*2-Dimensional Grid Cell Mapping*

In order to make the spatial map cover a navigation plane, the math from the one-dimensional equations must be applied to an orthogonal vector. In the case of an x-y plane with an a map of x lattices, $\vec{X}$ established in (3.3), then the map of y lattices can be calculated in (3.5) similarly to (3.1) with the x axis lattices [2].

$$Y_\beta(y) = \frac{mod(y, \lambda_\beta)}{\lambda_\beta} \cdot 2\pi \tag{3.5}$$

$Y_\beta$ is location on the lattice at which the grid cell is located. $y$ is the internal estimate of the location of the grid cell's position in space. $\lambda_\beta$ is the lattice period of grid cell $\beta$. $\beta$ is an index number.

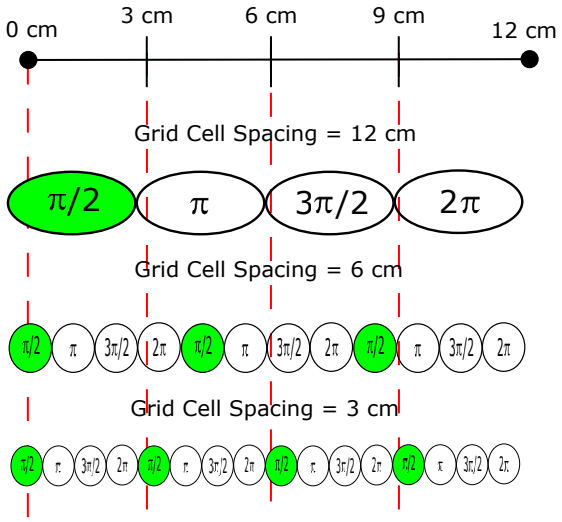



Figure 3.4 A map of positions on a one-dimensional axis is encoded with three grid cells. The red line shows the combination of phases that represent locations 0, 3, 6, and 9 [2]. The three grid cells lattice periods are of 12 cm, 6 cm, and 3 cm.

As in the case of (3.2), $N$ number of grid cells throughout the medial entorhinal cortex can be combined together for a location representation as in (3.6). The combinations of internal estimations from (3.6) build a vector of locations for the Y axis.

$$\Upsilon = (Y_1(y), Y_2(y), ..., Y_N(y)) \tag{3.6}$$

$$\vec{Y} = (\Upsilon_1, \Upsilon_2, ...\Upsilon_N) \tag{3.7}$$

With two axes mapped, there is a space formed by the X and Y vectors from (3.3) and (3.7) called $\mu$. $\mu$ from (3.8) can be visually represented in Figure 3.6.

$$\mu = \{\vec{X}, \vec{Y}\} \tag{3.8}$$

*Adding Error*

The capacity calculation (3.4) works only in an ideal case. As the grid cells continue to operate over time, the neuron's capacity is reduced and the map deteriorates. The deterioration of the grid cell's capacity [21] is approximated by (3.9).

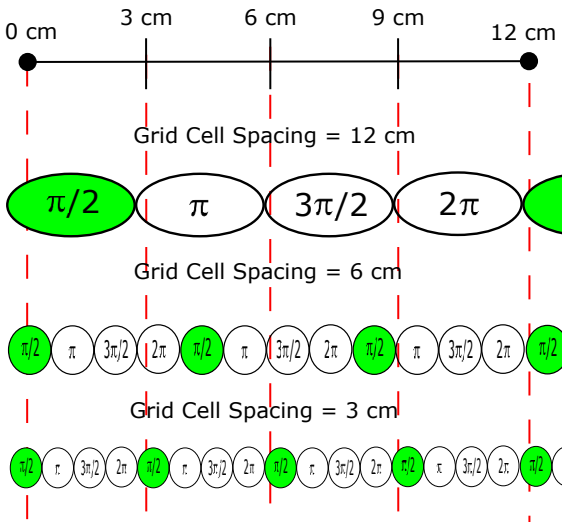



Figure 3.5 This figure shows that mapping for position 12 and position 0 are the same thus showing its capacity with grid cells with spacing 3, 6, and 12 cm will only read up to 11 cm as shown in (3.4) [2].

$$D_{est} = \frac{\lambda}{\delta\phi}^{N-1} \tag{3.9}$$

$\lambda$ is the set of grid cell lattice periods. $\frac{1}{\delta\phi}$ is the set of number of distinguishable phases for each lattice. $N$ is the total number of grid cells encoding the map.

There is a drift that occurs in the mapping as the object travels across space and localizes. The drift is calculated in (3.10) and is visually represented in Figure 3.7 and Figure 3.8 in order to visualize (3.10). Figure 3.8 shows thresholds (represented by dotted lines) have shifted upwards causing a different representation of a location.

$$\Delta x = \sqrt{D_{trans}\Delta t} \tag{3.10}$$

$$D_{trans} = CV^2/N \tag{3.11}$$

$CV$ is a coefficient of variation of the spiking intervals. $CV$ is a ratio of the standard deviation of spikes fired by the grid cells with the mean firing ratio which is set at $\frac{1}{8}$ [17]. $N$ is the neural network size. This happens to be a number of neurons recorded in [17] which is set at $10^5$ or $10^7$ in some of this thesis' experiments. The values were take directly from the analysis of a rat's medial entorhinal cortex to model grid cells appropriately [17].

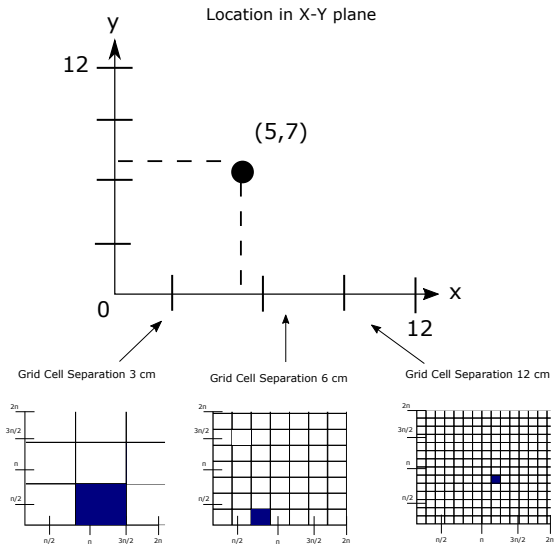



Figure 3.6 This figure shows the localization for arbitrary position (5,7) on the x-y plane through the firing of the grid cells of size 3, 6, and 12 cm. [2] Each blue square represents the phase that each grid cell takes as shown in (3.1) and (3.6). To see the phase encodings of the figure that will be used for localization see Figure 4.3.

### *Inertial Navigation and Localization*

Grid cells receive signals to fire from the inertial movement sensing system within mammals called the vestibular system [34] [49]. This vestibular system feeds inertial data to the grid cells, giving the grid cells the appropriate stimulation to spike and form lattices. This thesis does not focus on vestibular system modeling, however, because in our system the ubiquitous inertial navigation system (INS) provides inertial data to the robot. That means an INS can be used as an input to stimulate grid cells. The INS also provides an established idiothetic navigator to which to compare the grid cell's performance.

Inertial navigation systems use measurements from an accelerometer and gyroscope and apply the kinematic equations of motion to track the position and orientation of a robot. For this thesis, the inertial navigation system in the simulation mimics a rate gyroscope and a 2-axis accelerometer to measure angular velocity and linear acceleration [25] [31] [5].

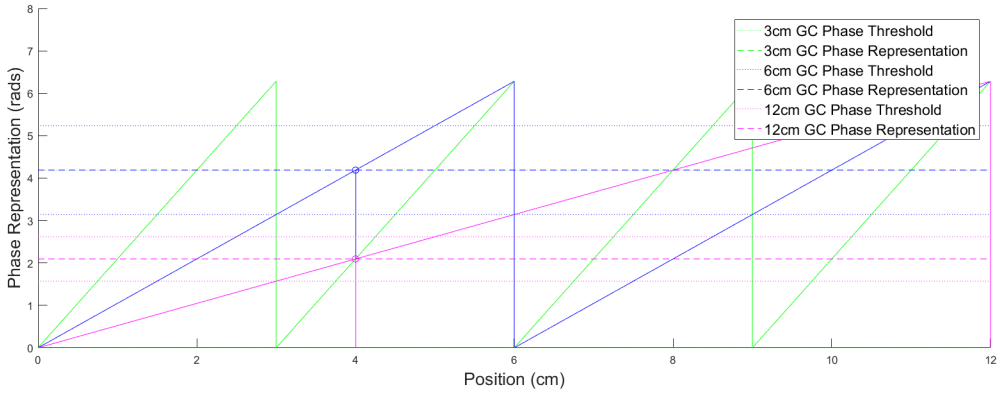



Figure 3.7 This shows the phases the map has encoded for position 5 on one axis

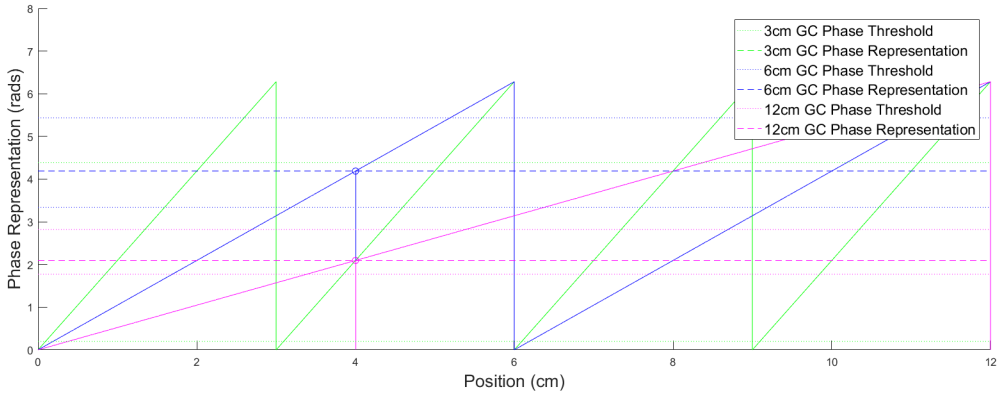


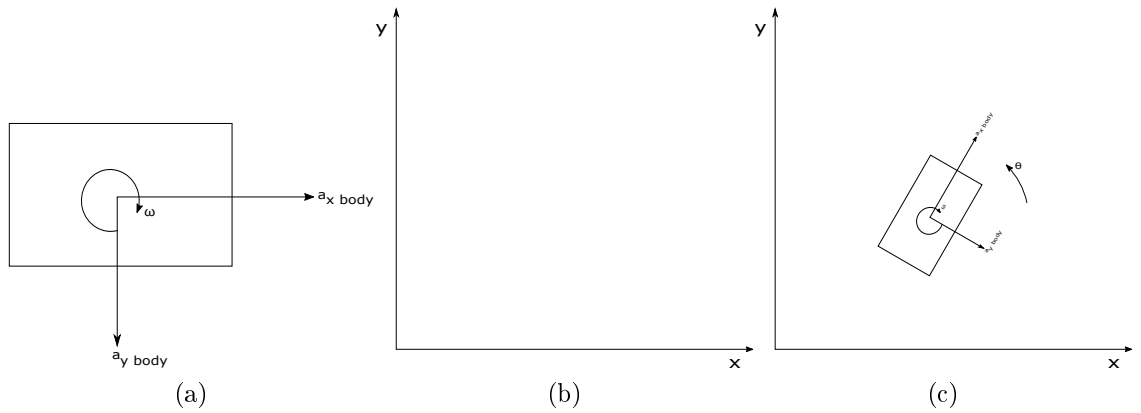

Figure 3.8 This shows (3.10) after 600 seconds (10 minutes) of navigating for position 5.

*Kinematic Equations*

A inertial navigation system uses kinematic equations to properly manipulate the gyroscope and accelerometer measurements to provide position information as an output. The kinematic equations describe the relationships among the robot, sensors, and navigation frame.

In order to get position from the acceleration readings, it is necessary to integrate both acceleration and velocity into a displacement. The measured acceleration is as read from the sensor. The estimated velocity, $v_{estimated}$, is calculated through the integration of the acceleration (3.12).

$$v_{estimated} = v_0 + \int_{t_1}^{t_2} a_{measured}\ dt \tag{3.12}$$




(a) (b) (c)

Figure 3.9 (a) shows a robot and the measurements read from the accelerometer and gyroscope (which are done relative to the body which is called the body frame). (b) shows a global frame that we can read from which is called the navigation frame. (c) shows how the robot moves through the navigation frame and its angle of change in direction.

Where $v_0$ is the initial velocity, and $a_{measured}$ is the acceleration measured and integrated from time $t_1$ to $t_2$.

The estimated position, $d_{estimated}$, is calculated is calculated through the integration of the velocity (3.13).

$$d_{estimated} = d_0 + \int_{t_1}^{t_2} v_{estimated}\ dt \tag{3.13}$$

Where $d_0$ is the initial displacement, and $v_{estimated}$ is the estimated velocity from (3.12) and integrated from time $t_1$ to $t_2$.

Equations (3.12) and (3.13) are in one-dimension and measured in relation to the body of the object that is moving (the body frame). The equations can be used to calculate a position from an orthogonal acceleration thus extending to a second dimension.

A gyroscope can measure an angular rate of change. By integrating the angular rate $\omega$ from a gyroscope over time, one can estimate the orientation (yaw angle) of the robot, $\theta_{estimated}$, (3.14).

$$\theta_{estimated} = \theta_0 + \int_{t_1}^{t_2} \omega\ dt \tag{3.14}$$

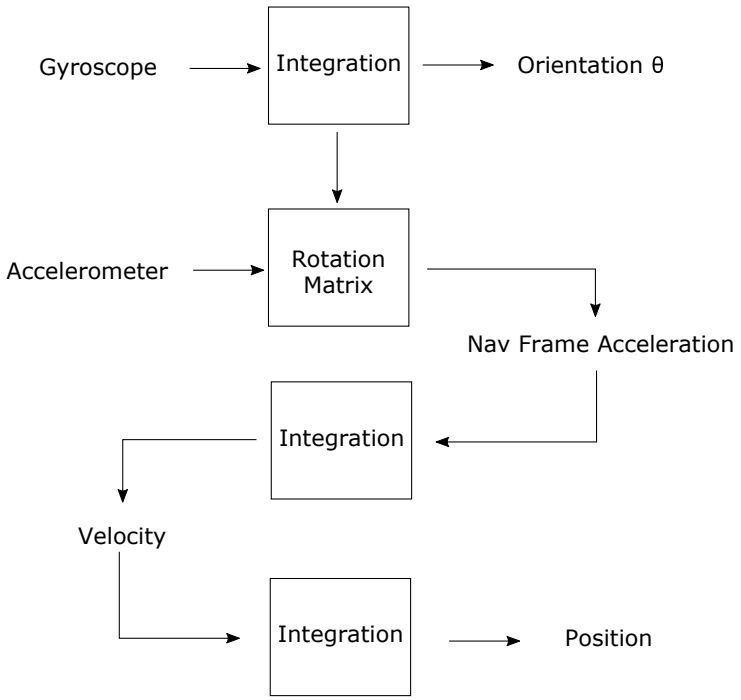



Figure 3.10 This figure shows the way that the INS readings from the gyroscope and accelerometer are manipulated through the algorithms in order to establish a position.

Where $\theta_0$ is the initial orientation of the robot, and $\omega$ is the measured angular velocity which is integrated from time $t_1$ to $t_2$.

Equation (3.14) estimates an angle which is then used to convert the acceleration and velocity from the body frame to the navigation frame.

*Position Estimation*

The angle retrieved from the gyroscope through the use of (3.14) can be used to create a rotation matrix. The rotation matrix in (3.15) transforms the measurements from the body frame to the navigation frame. In order to do this, the accelerations in the orthogonal axes ($a_{x-measured}$ and $a_{y-measured}$) is placed into a state space matrix (3.16).

$$C = \begin{pmatrix} cos(\theta_{estimated}) & sin(\theta_{estimated}) \\ -sin(\theta_{estimated}) & cos(\theta_{estimated}) \end{pmatrix} \tag{3.15}$$

$$a_{measured} = \begin{pmatrix} a_{x-measured} \\ a_{y-measured} \end{pmatrix} \tag{3.16}$$

The body frame accelerations (3.15) are multiplied by the frame transformation matrix (3.16) to express the measured accelerations in the navigation frame (3.17).

$$\begin{pmatrix} a_{x-navigation} \\ a_{y-navigation} \end{pmatrix} = C \times a_{measured} \tag{3.17}$$

A trapezoidal estimation method of numerical integration can be used to estimate position with sensor readings occurring every $\tau$ second intervals (3.18-3.20).

$$v_{navigation}(t) = v_{navigation}(t-1) + \frac{a_{navigation}(t) - a_{navigation}(t-1)}{2} * \tau \tag{3.18}$$

$$d_{navigation} = d_{navigation}(t-1) + \frac{v_{navigation}(t) - v_{navigation}(t-1)}{2} * \tau \tag{3.19}$$

$$d_{navigation} = \begin{pmatrix} x_{navigation} \\ y_{navigation} \end{pmatrix} \tag{3.20}$$

A system block diagram of the full inertial navigation system is visually represented in Figure 3.10. The position output from the inertial navigation system is the input of this grid cell mapping system. The first step of the grid cell mapping system is that the estimated position is transformed into a phase for each of the grid cells that formed the map for navigation. The position from the inertial navigation system is used as x and y in grid cell equations (3.1) and (3.5). The full interaction between the INS and the grid cell map to navigate is shown later in Figure 4.1.

*Error in Inertial Navigation*

The equations from the previous section show position accurately in an ideal environment. However, errors in sensing and calculations result in navigation systems providing position outputs that deviate from their true values. In order to model a realistic gyroscope sensor for simulation, (3.21) is used.

$$\omega_{measured} = \omega_b \times \omega_{sf} + \omega_{bias} + \omega_{sensitivity} + \omega_{noise} \tag{3.21}$$

Equation (3.21) includes errors that a real gyroscope would have included in its readings. $\omega_b$ represents the measurement without error. The $\omega_{sf}$, $\omega bias$, $\omega_{sensitivity}$, and $\omega_{noise}$ represent the scale factor error (a manufacturing tolerance error), bias (an offset that happens due to manufacturing), sensitivity (a measurement resolution factor of the gyroscope), and noise (a stochastic offset from true readings) from the sensor's specification sheet.

Accelerometers also have errors which should be modeled for more accurate position estimation. In order to properly simulate a realistic accelerometer reading, (3.22) is used.

$$a_{measured} = a_{scale_factor} \times a_b + a_{bias} + a_{noise} \tag{3.22}$$

As with (3.21) the $a_{scale_factor}$, $a_{bias}$, and the $a_{noise}$ are the scale factor, bias, and noise. The parameters are device specific, and are found on the accelerometer's specification datasheet. The $a_b$ is the acceleration without error.

# CHAPTER FOUR

## Simulation and Results

The algorithms presented in the previous chapter are implemented in this chapter. First, the combination of the INS and grid cell algorithms are described as well as the Matlab simulation code implementing the algorithms. This is followed by a detailed description mapping the theoretical mathematics to the simulation implementations. The last part of this chapter shows the results of using this code to simulate navigation over large spaces. Navigation performance is assessed when the robot performs a square movement pattern with a simple velocity and acceleration profile.

### *Grid Cell Aided Navigation System*

Grid cell navigation, when described in neuroscience literature, does not typically describe the details of how the grid cells are activated based on the rat's movement. The grid cell's firings are measured directly from rat's brains [17], and the physical (spatial) mapping is recorded as the rat moves. This thesis explores modeling the grid cells receiving their excitation from velocity estimates derived from an INS. This INS-to-Grid Cell system interaction is shown in the block diagram in Figure 4.1.

Figure 4.1 shows the interactions of the algorithms described in Chapter 3. The algorithms in the model were simulated using Matlab in order to gather results for this thesis. Matlab code implementing each of the blocks in Figure 4.1 is included in Appendix B. The correlation of Chapter 3 equations to Matlab code implementation is summarized in Table 4.1.

Table 4.1 Each one of the equations has a place in the code that is in the Appendices. The lines of code and file they're under are listed in the table above.

| Equation Number | .m File |
|---|---|
| 3.1 | posfromGCmap |
| 3.2 | gridcellmap |
| 3.3 | gridcellmap |
| 3.4 | gridcellmap |
| 3.5 | posfromGCmap |
| 3.6 | gridcellmap |
| 3.7 | gridcellmap |
| 3.8 | gridcellmap |
| 3.9 | gridcellmap |
| 3.10 | posfromGCmap |
| 3.11 | IMU |
| 3.12 | IMU |
| 3.13 | IMU |
| 3.14 | IMU |
| 3.15 | IMU |
| 3.16 | IMU |
| 3.17 | INSwGridCellAid |
| 3.18 | INSwGridCellAid |
| 3.19 | INSwGridCellAid |
| 3.20 | IMU |
| 3.21 | IMU |

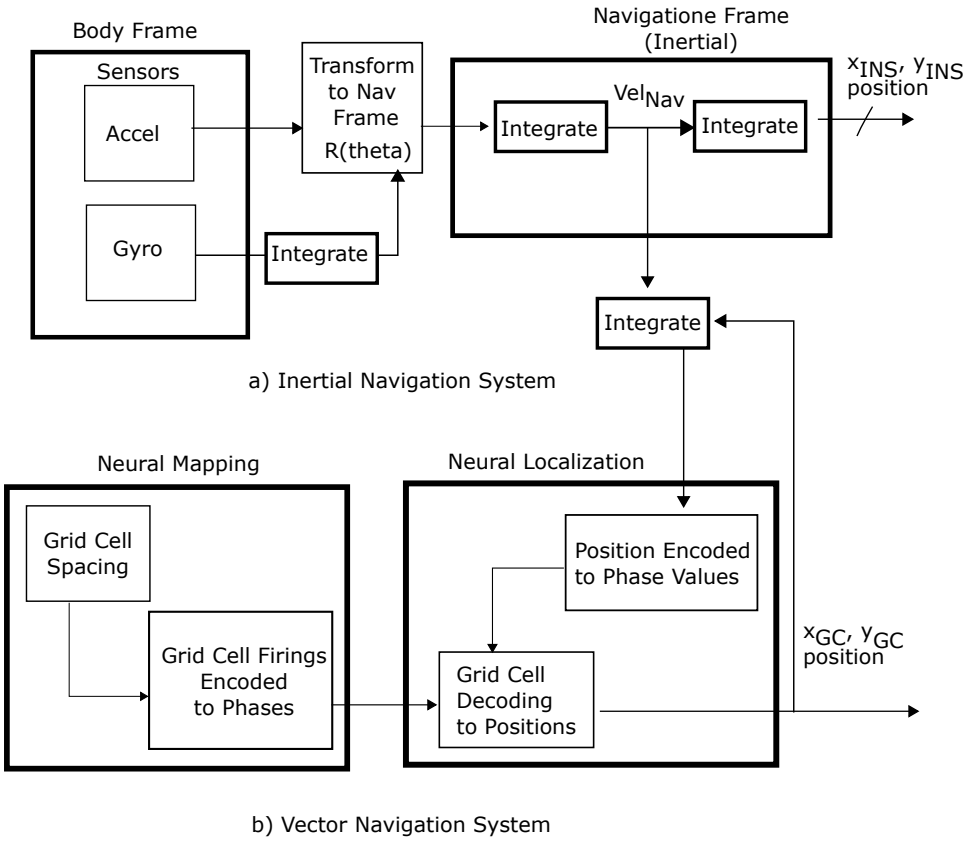



Figure 4.1 Dead Reckoning Using Grid Cells: This figure shows the interaction of the equations described in Chapter 3.

*Step 1: Neural Mapping in the Grid Cell ("Vector Navigation") System*

The model starts with a mapping of the space through grid cells. This subsystem is depicted by the Neural Mapping block in Figure 4.1b. This map's boundaries are calculated using (3.4). For example, for grid cell modules with 3, 6, and 12 spacings, the grid cells modules can map up to 12 cm in each axis (x and y).

Equations (3.2) and (3.6) represent different sets of phases for a location in space. These form the sets of locations (3.3) and (3.7) for the grid cell modules as shown in Figure 4.2. The sets will then form a map (3.8) on which to navigate.

The mapping is done in the Matlab file gridcellmap.m in Appendix B. This code is simplified in pseudocode shown in Algorithm 1. The code sets the number of grid cell modules and the interval between grid cell firings. Then it encodes the module

$$\vec{X} = \{X_1, X_2, X_3\}$$
w/ $X_1, X_2$, and $X_3$ being modules with spacing 3, 6, and 9 respectively.

$$X_1 = \{0, \frac{\pi}{15}, \frac{2\pi}{15}, \frac{\pi}{5}, \frac{4\pi}{15}, \frac{\pi}{3}, ...\}$$
$$X_2 = \{0, \frac{\pi}{30}, \frac{\pi}{15}, \frac{\pi}{10}, \frac{2\pi}{15}, \frac{\pi}{6}, ...\}$$
$$X_3 = \{0, \frac{\pi}{60}, \frac{\pi}{30}, \frac{\pi}{20}, \frac{\pi}{15}, \frac{\pi}{12}, ...\}$$

$$\vec{X} = \{0\} = \{0, 0, 0\}$$
$$\vec{X} = \{1\} = \{\frac{\pi}{3}, \frac{\pi}{6}, \frac{\pi}{12}\}$$
$$\vec{X} = \{2\} = \{\frac{2\pi}{3}, \frac{\pi}{3}, \frac{\pi}{6}\}$$

$$\mu = \{\vec{X}, \vec{\Upsilon}\}$$

Figure 4.2 This shows the phase representations of the locations being mapped with modules of interval 3, 6, and 9 and a resolution of 1 cm. These vectors are then fed into the full map in $\vec{X}$. The same process is used to map $\vec{\Upsilon}$.

into phases according to the strength of the grid cell's firing. These encoded phases get stored into a map to allow for localization to occur. This neural mapping can be seen as a pre-processing or initialization step and only needs to be performed once before the Grid Cell localization starts.

*Step 2: Neural Localization*

The Neural Localization process, shown by the box in Figure 4.1b, can begin once the neural mapping pre-processing is finished and INS data is sent to the Neural Localization algorithm.

*Preparing the Neural Localization Input Data.* As shown in Figure 4.1, estimated velocity is the input into the Neural Localization algorithm. The source of this estimated velocity, however, are the robot's gyroscope and accelerometer sensors. Before being integrated to estimate velocity, the acceleration data first needs to be expressed in the navigation frame. The gyroscope data is integrated over time to es-

**Algorithm 1** Grid Cell Mapping Code

```
1:  procedure Grid_Cells_Mapping
2:
3:      Set number of Modules
4:      Set X firing interval for all Modules
5:      Set Y firing interval for all Modules
6:
7:      for X grid cell measurements do
8:
9:          for all Modules do
10:
11:             Encode Modules with Phase Encode Equation
12:             Put Phases into Map
13:
14:         end for
15:
16:     end for
17:
18:     for Y grid cell measurements do
19:
20:         for all Modules do
21:
22:             Encode Modules with Phase Encode Equation
23:             Put Phases into Map
24:
25:         end for
26:
27:     end for
28:
29: end procedure
```

timate the robot's yaw angle, $\theta_{estimated}$ (3.14), and then $\theta_{estimated}$ is used to create the frame transformation matrix between the body frame and the navigation frame (3.15). The frame transformation matrix, C, is then used in (3.17) to express the robot's acceleration in the Navigation frame. Finally, a trapezoidal estimation integration method (3.18) is used to estimate the robot's velocity. This velocity is the input to the Neural Localization algorithm.

The simulated accelerometer and gyroscope data data for the square cases is generated using matlab file sqraccmet.m. The sqraccmet.m code is simplified using pseudocode as shown in Algorithm 2. The code sets the time step and a state space representation of the kinematics with an initial value. As acceleration is measured, the estimated velocity is updated through trapezoidal estimations. The velocity is the input to Algorithm 3 which will update position with the grid cell assistance. This is expanded upon in the next section.

**Algorithm 2** Kinematic Equations

```
1:  procedure KINEMATIC_EQUATIONS
2:
3:      Set Time Step
4:      Initialize State Space Representation
5:      Find Velocity through Trapezoidal Estimation
6:      Find Position through Trapezoidal Estimation
7:      Update State Space Representation
8:
9:      for Acceleration Readings do
10:
11:         Find Velocity through Trapezoidal Estimation
12:         Find Angle through Trapezoidal Estimation
13:         Use Rotation Matrix to find Velocity in navigation frame
14:
15:     end for
16:     Set Position from Localize Procedure
17:
18: end procedure
```

$$xmod(\lambda_3) = \frac{5mod(3)}{3} \cdot 2\pi = \frac{4\pi}{3}$$
$$xmod(\lambda_6) = \frac{5mod(6)}{6} \cdot 2\pi = \frac{5\pi}{3}$$
$$xmod(\lambda_{12}) = \frac{5mod(12)}{12} \cdot 2\pi = \frac{5\pi}{6}$$

$$ymod(\lambda_3) = \frac{7mod(3)}{3} \cdot 2\pi = \frac{\pi}{3}$$
$$ymod(\lambda_6) = \frac{7mod(6)}{6} \cdot 2\pi = \frac{\pi}{3}$$
$$ymod(\lambda_{12}) = \frac{7mod(12)}{12} \cdot 2\pi = \frac{7\pi}{6}$$

$$\mu_{est} = \{5, 7\} = \{\{\frac{4\pi}{3}, \frac{5\pi}{3}, \frac{5\pi}{6}\}, \{\frac{\pi}{3}, \frac{\pi}{3}, \frac{7\pi}{6}\}\}$$

Figure 4.3 This shows the phase representations of the position fed by the INS. These values are stored and compared to the mapping values available in the initial mapping $\mu$. On match, the grid cells will return the value from the grid cell map.

*Localization through the Vector Navigation System.* The estimated velocity from the inertial navigation system is the input to the grid cell system. Path integration is then performed using 1) the INS velocity estimate, and 2) the Neural Localization system's current estimate of position (4.1).

$$d_{estimated} = d_{gridcellestimate} + \int_{t_1}^{t_2} v_{estimated}\, dt \tag{4.1}$$

This estimated position is encoded into phases using (3.2) and (3.5). The phase converted data gets compared to the map that was formed at the beginning (3.8). This gives an estimated position through grid cells.

If the Neural Localization path integration system calculates a coordinate (5,7), then those are the inputs to phase encoding equations (3.2) and (3.5). If each one of the modules has spacings $\alpha$ and $\beta$ = 3, 6, and 12 then the calculations are shown in Figure 4.3. The provided sets $\vec{X} = \{\frac{4\pi}{3}, \frac{5\pi}{3}, \frac{5\pi}{6}\}$ and $\vec{\Upsilon} = \{\frac{\pi}{3}, \frac{\pi}{3}, \frac{7\pi}{6}\}$ are compared to map $\mu = \{\vec{X}, \vec{\Upsilon}\}$ and return $\{5, 7\}$ as a coordinate shown in Figure 4.4. A visual representation of the thresholds of phases being decoded from the map ($X_{GC}$,$Y_{GC}$) is shown in Figure 4.5

$$\mu = \{\{X_1, X_2, X_3, ...\}, \{\Upsilon_1, \Upsilon_2, \Upsilon_3, ...\}\} =$$
$$\{\{\{0,0,0\}, \{\frac{\pi}{3}, \frac{\pi}{6}, \frac{\pi}{12}\}, \{\frac{2\pi}{3}, \frac{\pi}{3}, \frac{\pi}{6}\}, ...\}, \{\{0,0,0\}, \{\frac{\pi}{3}, \frac{\pi}{6}, \frac{\pi}{12}\}, \{\frac{2\pi}{3}, \frac{\pi}{3}, \frac{\pi}{6}\}, ...\}\}$$

$$\mu_{est} = \{5, 7\} = \{\{\frac{4\pi}{3}, \frac{5\pi}{3}, \frac{5\pi}{6}\}, \{\frac{\pi}{3}, \frac{\pi}{3}, \frac{7\pi}{6}\}\}$$

$$(X_{GC}, Y_{GC}) =$$
$$\begin{cases} \begin{pmatrix} (0,0), & \mu_{est} = \{\{0 < xmod(\lambda_3) < \frac{\pi}{3}, 0 < xmod(\lambda_6) < \frac{\pi}{6}, 0 < xmod(\lambda_{12}) < \frac{\pi}{12}\}, \\ & \{0 < ymod(\lambda_3) < \frac{\pi}{3}, 0 < ymod(\lambda_3) < \frac{\pi}{6}, 0 < ymod(\lambda_{12}) < \frac{\pi}{12}\}\} \\ (0,1), & \mu_{est} = \{\{0 < xmod(\lambda_3) < \frac{\pi}{3}, 0 < xmod(\lambda_6) < \frac{\pi}{6}, 0 < xmod(\lambda_{12}) < \frac{\pi}{12}\}, \\ & \{\frac{\pi}{3} < ymod(\lambda_3) < \frac{2\pi}{3}, \frac{\pi}{6} < ymod(\lambda_6) < \frac{\pi}{3}, \frac{\pi}{12} < ymod(\lambda_{12}) < \frac{\pi}{6}\}\} \\ (1,0), & \mu_{est} = \{\{\frac{\pi}{3} < xmod(\lambda_3) < \frac{2\pi}{3}, \frac{\pi}{6} < xmod(\lambda_6) < \frac{\pi}{3}, \frac{\pi}{12} < xmod(\lambda_{12}) < \frac{\pi}{6}\}, \\ & \{0 < ymod(\lambda_3) < \frac{\pi}{3}, 0 < ymod(\lambda_6) < \frac{\pi}{6}, 0 < ymod(\lambda_{12}) < \frac{\pi}{12}\}\} \\ ... & ... \\ (5,6), & \frac{4\pi}{3} < xmod(\lambda_3) < 2\pi, \frac{5\pi}{3} < xmod(\lambda_6) < 2\pi, \frac{5\pi}{6} < xmod(\lambda_{12}) < \frac{\pi}{2}\}, \\ & \{0 < ymod(\lambda_3) < \frac{2\pi}{3}, 0 < ymod(\lambda_6) < \frac{\pi}{6}, \pi < ymod(\lambda_{12}) < \frac{7\pi}{6}\}\} \\ (5,7), & \mu_{est} = \{\{\frac{4\pi}{3} < xmod(\lambda_3) < 2\pi, \frac{5\pi}{3} < xmod(\lambda_6) < 2\pi, \frac{5\pi}{6} < xmod(\lambda_{12}) < \frac{\pi}{2}\}, \\ & \frac{2\pi}{3} < ymod(\lambda_3) < \frac{4\pi}{3}, \frac{\pi}{3} < ymod(\lambda_6) < \frac{2\pi}{3}, \frac{7\pi}{6} < ymod(\lambda_{12}) < \frac{4\pi}{3}\}\} \\ (6,7), & \mu_{est} = \{\{0 < xmod(\lambda_3) < \frac{2\pi}{3}, 0 < xmod(\lambda_6) < \frac{\pi}{6}, \pi < xmod(\lambda_{12}) < \frac{7\pi}{6}\}, \\ & \{\frac{2\pi}{3} < ymod(\lambda_3) < \frac{4\pi}{3}, \frac{\pi}{3} < ymod(\lambda_6) < \frac{2\pi}{3}, \frac{7\pi}{6} < ymod(\lambda_{12}) < \frac{4\pi}{3}\}\} \\ ... & ... \end{pmatrix} \end{cases}$$
$$(X_{GC}, Y_{GC}) = (5, 7)$$

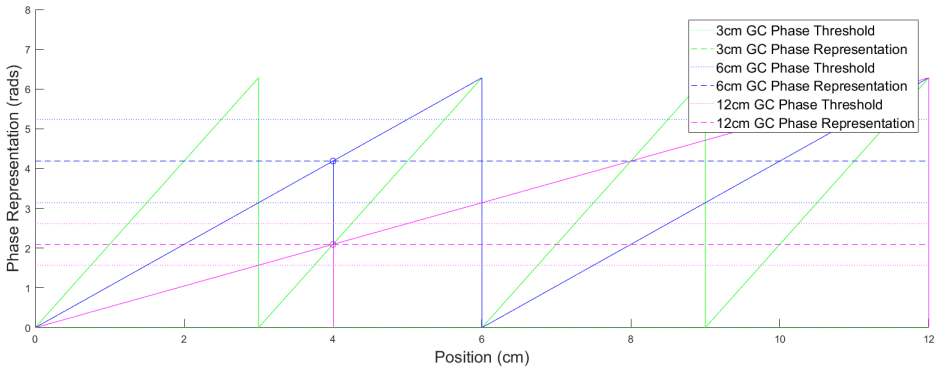
Figure 4.4 This shows how the position is found from the map once the estimated position is encoded. The values in the piecewise function are found in the mapping phases as seen are equivalent to the ones in $\mu$ above.

This updated position gets fed back into (4.1), and the localization happens at the next step. The next estimated position can be seen in Figure 4.6. As each step happens, the mapping degrades over time according to (3.10).

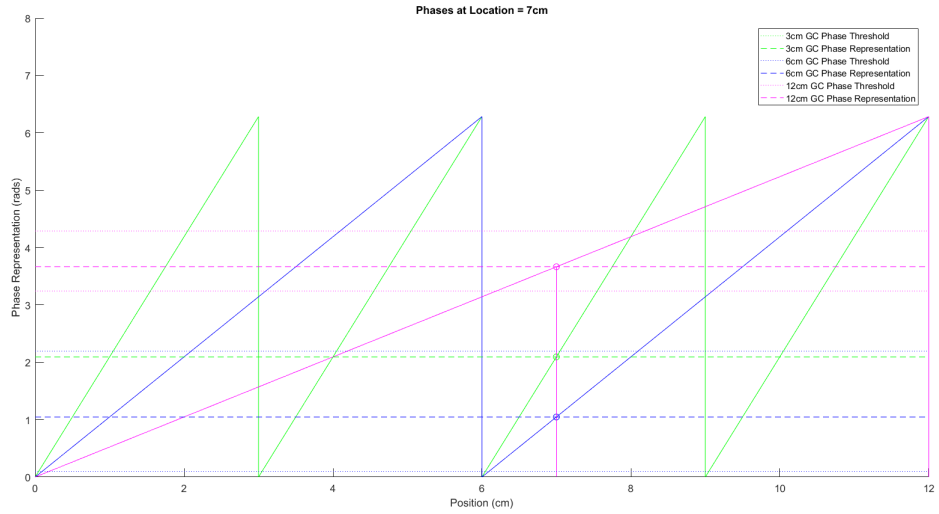
This Neural Localization code is simplified in pseudocode shown in Algorithm 3. The code gets the number of modules and map from Algorithm 1 and velocity and time step from Algorithm 2. It then sets the coefficient of variation to include the map deviation over time and phase threshold to map values appropriately. The position is then updated with a trapezoidal estimation from velocity from Algorithm 2 and the previous grid cell estimation. This position is then encoded into phases, and found in the map. As time passes, the map degrades, and the code accounts for that. The position found from the grid cell map is then the updated position estimation.

**Algorithm 3** Grid Cell Decoding Algorithm

```
1:  procedure Localize_From_Grid_Cell_Map
2:
3:      Set Number of Modules
4:      Set Time Step
5:      Set Velocity
6:      Set Map
7:      Set Coefficient of Variation
8:      Set Phase Thresholds
9:
10:     for all Modules do
11:
12:         if 1st Position Step then
13:
14:             Find Position through Trapezoidal Estimation with ...
15:             Grid Cell Position Estimate
16:
17:         end if
18:
19:         Set Phase Values for Position
20:
21:     end for
22:
23:     if 1st Position then
24:
25:         Set 1st Mapping position
26:
27:     else
28:
29:         Find Position from Map
30:
31:     end if
32:
33:     for each Mapping do
34:
35:         Add Time Step to Find total Time
36:         Add Error to Map phase values according to Time passed
37:
38:     end for
39:
40: end procedure
```




(a)




(b)

Figure 4.5 This figure is a visualization of the thresholds for the phases that represent each location. a) shows the phases that represent position 5 for the x axis and b) shows the phases that represent position 7 for the y axis as was solved for in 4.4.

### *Method Validation*

In order to confirm the validity of the methods established in the algorithms section, the simulations were run without any error on the paths created for testing. In order to do that, the error equations were not added into the estimations of position for both the inertial navigation and the grid cell aided navigation. Navigation performance is assessed when the robot performs a square movement pattern with a simple velocity and acceleration profile.

$$\mu = \{\{X_1, X_2, X_3, ...\}, \{\Upsilon_1, \Upsilon_2, \Upsilon_3, ...\}\} =$$
$$\{\{\{0,0,0\}, \{\frac{\pi}{3}, \frac{\pi}{6}, \frac{\pi}{12}\}, \{\frac{2\pi}{3}, \frac{\pi}{3}, \frac{\pi}{6}\}, ...\}, \{\{0,0,0\}, \{\frac{\pi}{3}, \frac{\pi}{6}, \frac{\pi}{12}\}, \{\frac{2\pi}{3}, \frac{\pi}{3}, \frac{\pi}{6}\}, ...\}\}$$

$$\mu_{est} = \{5.1, 7.1\} = \{\{\frac{4.2\pi}{3}, \frac{5.1\pi}{3}, \frac{5.1\pi}{6}\}, \{\frac{1.1\pi}{3}, \frac{1.1\pi}{3}, \frac{7.1\pi}{6}\}\}$$

$$(X_{GC}, Y_{GC}) =$$
$$\left\{ \begin{array}{cc} ... & ... \\ (5,6), & \mu_{est} = \{\{\frac{4\pi}{3} < xmod(\lambda_3) < 2\pi, \frac{5\pi}{3} < xmod(\lambda_6) < 2\pi, \frac{5\pi}{6} < xmod(\lambda_{12}) < \frac{\pi}{2}\}, \\ & \{0 < ymod(\lambda_3) < \frac{2\pi}{3}, 0 < ymod(\lambda_6) < \frac{\pi}{6}, \pi < ymod(\lambda_{12}) < \frac{7\pi}{6}\}\} \\ (5,7), & \mu_{est} = \{\{\frac{4\pi}{3} < xmod(\lambda_3) < 2\pi, \frac{5\pi}{3} < xmod(\lambda_6) < 2\pi, \frac{5\pi}{6} < xmod(\lambda_{12}) < \frac{\pi}{2}\}, \\ & \frac{2\pi}{3} < ymod(\lambda_3) < \frac{4\pi}{3}, \frac{\pi}{3} < ymod(\lambda_6) < \frac{2\pi}{3}, \frac{7\pi}{6} < ymod(\lambda_{12}) < \frac{4\pi}{3}\}\} \\ (6,7), & \mu_{est} = \{\{0 < xmod(\lambda_3) < \frac{2\pi}{3}, 0 < xmod(\lambda_6) < \frac{\pi}{6}, \pi < xmod(\lambda_{12}) < \frac{7\pi}{6}\}, \\ & \{\frac{2\pi}{3} < ymod(\lambda_3) < \frac{4\pi}{3}, \frac{\pi}{3} < ymod(\lambda_6) < \frac{2\pi}{3}, \frac{7\pi}{6} < ymod(\lambda_{12}) < \frac{4\pi}{3}\}\} \\ ... & ... \end{array} \right)$$
$$(X_{GC}, Y_{GC}) = (5,7)$$

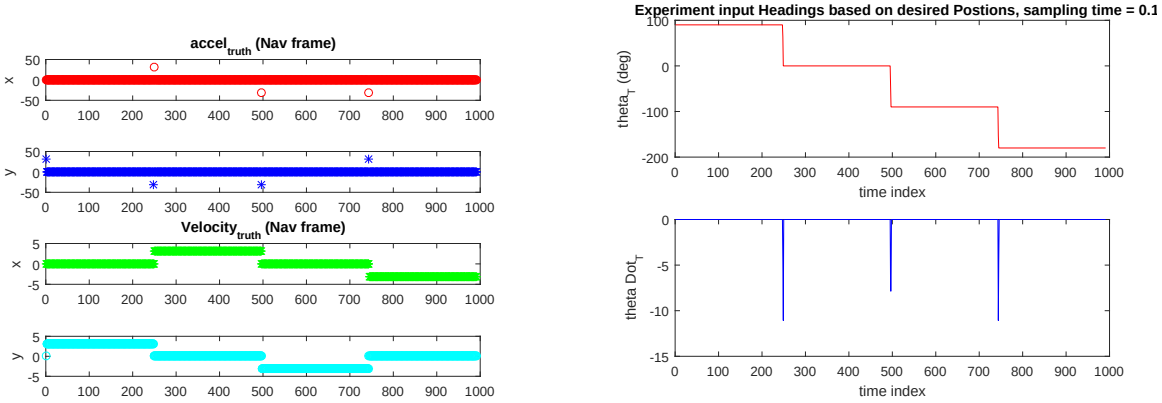
Figure 4.6 This shows localization when offset by .1 to see what happens whenever there's a difference in the estimated position. This shows the grid cell's resiliency to change.

*Square Path Test Cases*

The set paths for testing were an 80x80 cm square within $1m^2$ area, and an 8x8 m square held within a 10 $m^2$ area. These lengths and areas were chosen in order to represent different spaces that could be traveled by a robot or animal and have been observed in [21] for grid cell data from rats. The 1 $m^2$ area could represent a cage or maze, and the 10 $m^2$ area could represent an apartment or backyard.

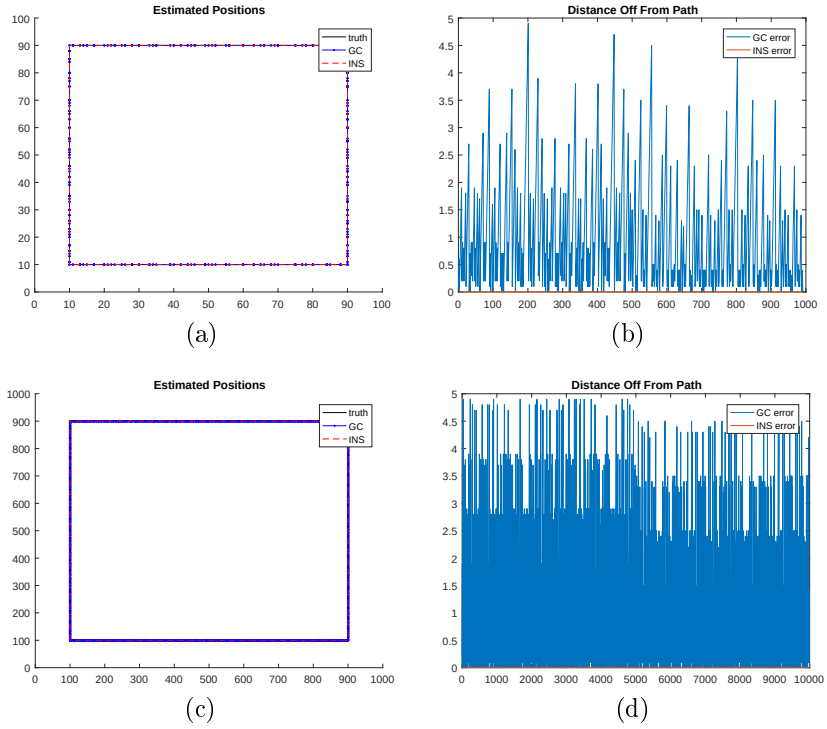
Each of these paths is traveled in 100 s and 1000 s and the paths traveled will be 320 and 3200 cm respectively thus going past the limits tested physically on rats according to [17]. The grid cells were set at 5, 23, 59, 101, 153, and 171 cm respectively to meet capacity requirements in (3.4) and (3.9) and give a diversity of grid cells.

The base-line (error free) model results shown as Figures 4.7 and 4.8 verify that the models work at following the paths. The INS without error follows the true path closely. The acceleration and angle change immediately at the corners. The velocity stays consistently at 3.2 cm/s in whatever direction of travel it takes as shown in Figure 4.7.




(a) (b)

Figure 4.7 a) shows the true acceleration at 32 $cm/s^2$ and true velocity at 3.2 $cm/s$ as the path is followed. b) shows that the turns are being taken and the simulation following the path turns at the square's 4 corners.




(a) (b)

(c) (d)

Figure 4.8 a) and b) show data for a 1 $m^2$ space, and c) and d) show data for 10 $m^2$ space. The inertial navigation with no error added shows negligible deviation in magnitude from the true path. Both navigation systems show no worsening as they travel along the path.

The grid cells follow along, increasing error as shown in the spikes of Figure 4.8b and Figure 4.8d until it meets the phase change necessary for its position reading to match up with a mapping set up in the area.

*Error Simulation*

The verification of the base-line model in the previous section allows for further studies involving the addition of error as shown in (3.10), (3.21), and (3.22) for the inertial and grid cell aided navigation.

*GC Navigation Excited by INS Dead Reckoning Position Estimates*

The basis of this thesis is grid cell navigation like in rat's brains. Thus, we want to emulate the system as closely as possible. In order to do that, the interactions that occur in a rat's brain have been modeled in Figure 4.9.

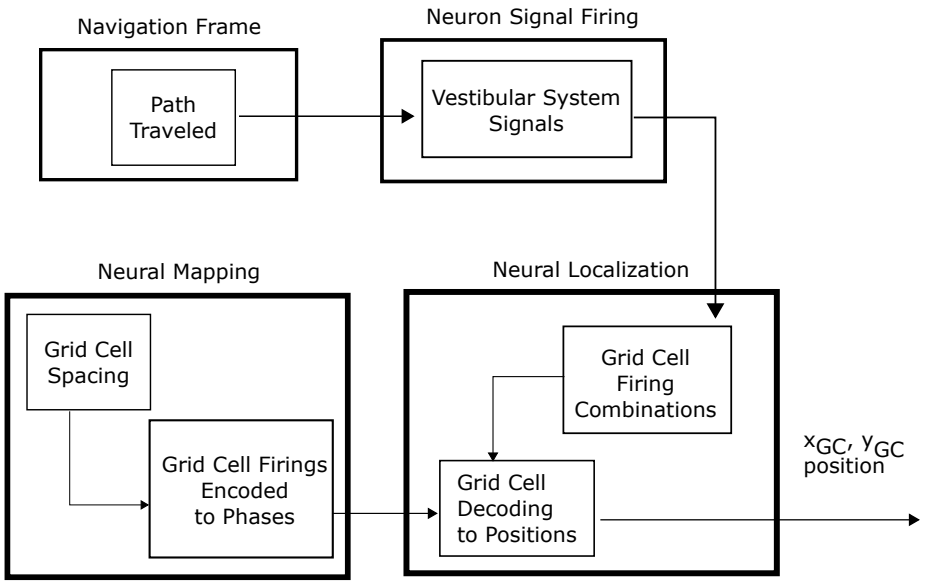


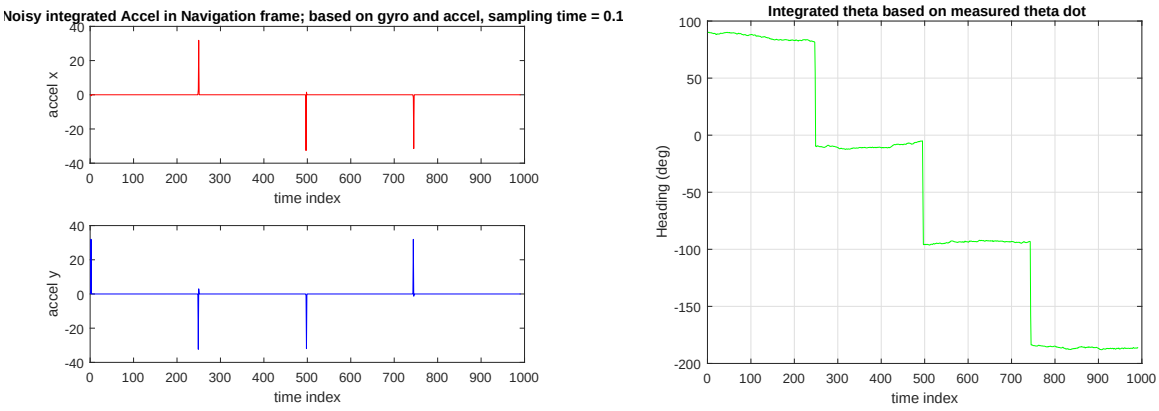

Figure 4.9 This figure shows the interaction of systems within the brain in rats.

For an initial experiment, the noise added is random and within 1 standard deviation of error as shown in Table 4.2. Because of the added noise, Dead Reckoning results in values of acceleration, integrated velocity, and integrated angles that deviate from the baseline as shown in Figure 4.10 with its effects on velocity seen in Figure 4.13. The estimated positions using only dead reckoning now stray away from the paths as shown in Figure 4.11. Figure 4.11 also shows that the grid cell navigation

Table 4.2 This table shows the values of the errors derived from the specification sheet for the accelerometer and gyroscope models used in this thesis.

| Type of Error | Magnitude |
|---|---|
| $\sigma_{accel}$ | .038259 $cm/s^2$ |
| $scalefactor_{accel}$ | 1.000382 $cm/s^2$ |
| $bias_{accel}$ | .00039 $cm/s^2$ |
| $\sigma_{gyro}$ | .038 rad/s |
| $scalefactor_{gyro}$ | 1.0069565 rad/s |
| $bias_{gyro}$ | .006891 rads/s |

alone can provide an improvement not only through the average movement away from the path, but as well as in the standard deviation of the error. A problem with this method is that the localization is not being performed with positions being updated as if the grid cells were firing like the brains in rats. Figure 4.9 shows a block diagram of thee interaction of grid cells and positioning in a rat's brain.




(a) (b)

Figure 4.10 This shows the test data for Figure (4.11) and 4.12. a) This shows some variation in the acceleration with noise. b) This shows some variation in the angle. Deviations of heading angle are evident when compared to the noise free headings in Figure 4.7b.

*GC Navigation Excited by Simulated Inertial Measurements*

The process between sensed movement and grid cell activation is not well documented in the neuroscience literature. It can be assumed that it is activated from the

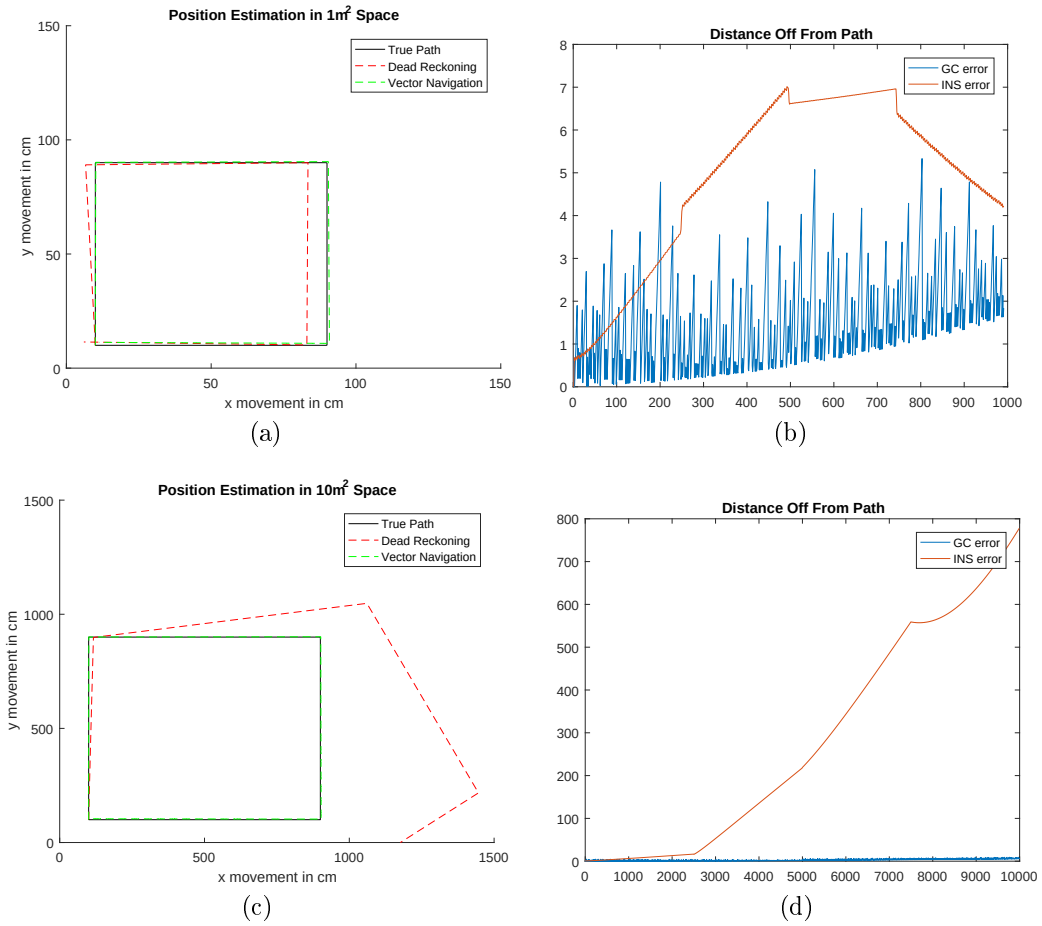



Figure 4.11 a) and b) show data for a 1 $m^2$ space, and c) and d) show data for 10 $m^2$ space. The inertial and grid cell aided navigation are shifting away from the true path.

vestibular system, but the grid cell's lattice fires whenever the rat is in a position to allow for it to happen with little correlation to other systems [21]. Until this connection is made, the grid cells must be excited by something that estimates movement like the inertial navigation system. Thus, the inertial navigation and grid cell aided navigation positions can be merged as shown in Figure 4.1. The estimated velocity from noisy INS data can be input to the grid cell algorithm. This results in Figure 4.12. For the experiment in Figure 4.12, the improvement on average is .3039 cm and has a standard deviation of 1.3267 cm for the 1 $m^2$ space and .08514 cm and 2.5339 cm for the 10 $m^2$ space. The average improvement is the mean of the improvement

at every discrete location along the path traveled, and that mean is then used to calculate the standard deviation.

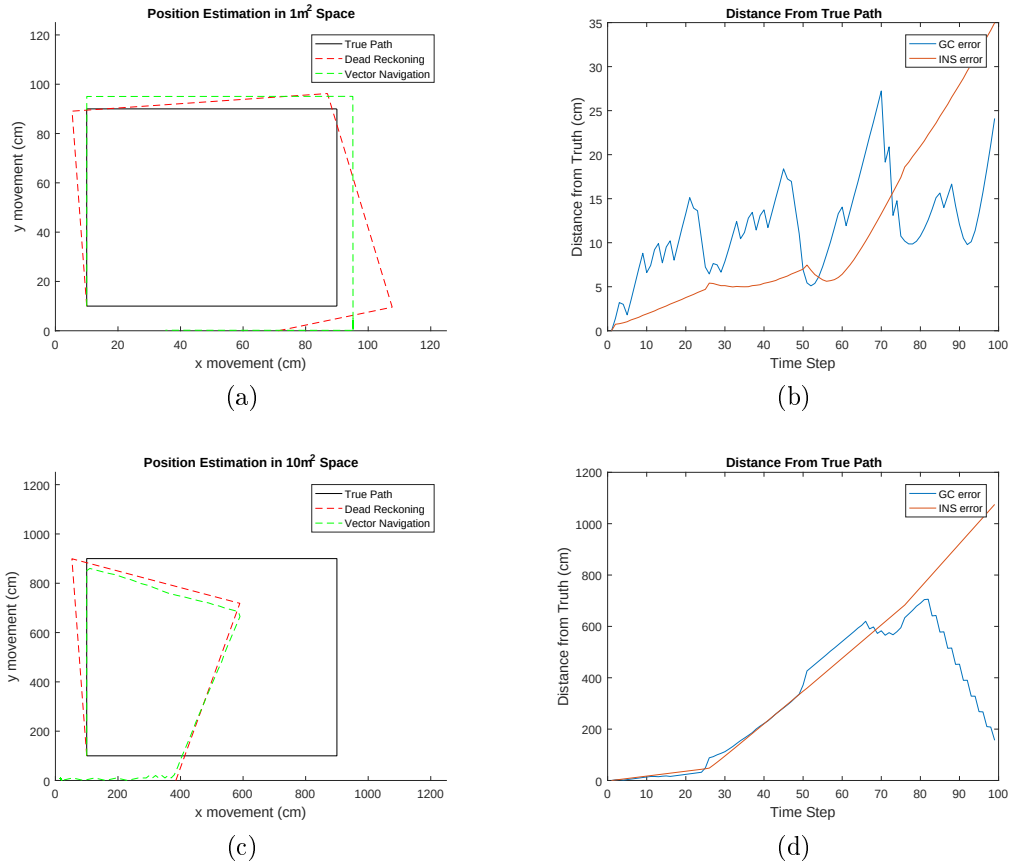
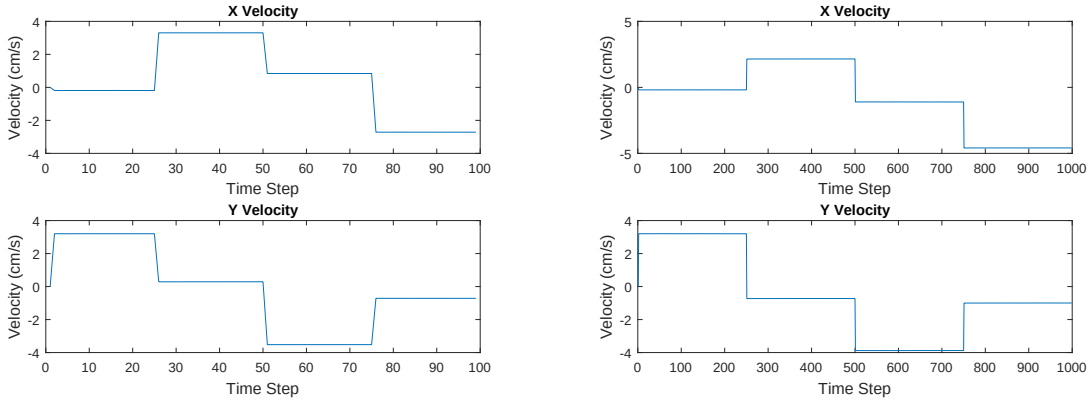



Figure 4.12 a) and b) show data for a 1 $m^2$ space, and c) and d) show data for 10 $m^2$ space. The inertial navigation with error added shows position estimations off of the true path, and the grid cell aided navigation shows an improvement compared to inertial navigation alone.

*Statistical Verification*

In order to assess if the improvement shown in Figure 4.12 is statistically significant, each one of these paths was traversed 1000 times with the inertial navigation system and the grid cell aided system as shown in Figure 4.1. The average path taken by the robot and the average error over the paths taken are shown in Figure 4.14. The average path taken by the robot is a set (4.2) filled with values calculated in




(a) (b)

Figure 4.13 This figure shows the velocity with error as the paths in 4.12 are being followed. a) shows the velocity for the 1 m path and b) shows the velocity for the 10m path

(4.3) where N is the total number of runs, n is the step in each run, and $d_{estimated(n)_i}$ is the position estimate at a step for a run.

$$p_{avg} = \{d_{avg_1}, d_{avg_2}, ..., d_{avg_n}\} \tag{4.2}$$

$$d_{avg_n} = \frac{\sum_{i=1}^{N} d_{estimated(n)_i}}{N} \tag{4.3}$$

The average error over the paths is calculated in a similar fashion. The error $e_{step(n)_i}$ is calculated in (4.6) by subtracting the true position from the estimated position for time step n. Then (4.5) is used to average the error at the same time step n for all runs 0-$N$. Finally, the average error over the path (4.4) is a set built from those values.

$$e_{avg} = \{e_{stepavg_1}, e_{stepavg_2}, ..., e_{stepavg_n}\} \tag{4.4}$$

$$e_{stepavg_n} = \frac{\sum_{i=1}^{N} e_{step(n)_i}}{N} \tag{4.5}$$

$$e_{step(n)_i} = d_{estimated} - truth \tag{4.6}$$

The data in Figure 4.12 shows that there is not improvement, on average, using the Grid Cell algorithm to aid the INS. The improvement on average is -.4593 cm and deviates by .8097 cm for the 1 $m^2$ space and -11.2014 cm and 30.3218 cm for

the 10 $m^2$ space. These results represent only one standard deviation of noise for the inertial navigation system.

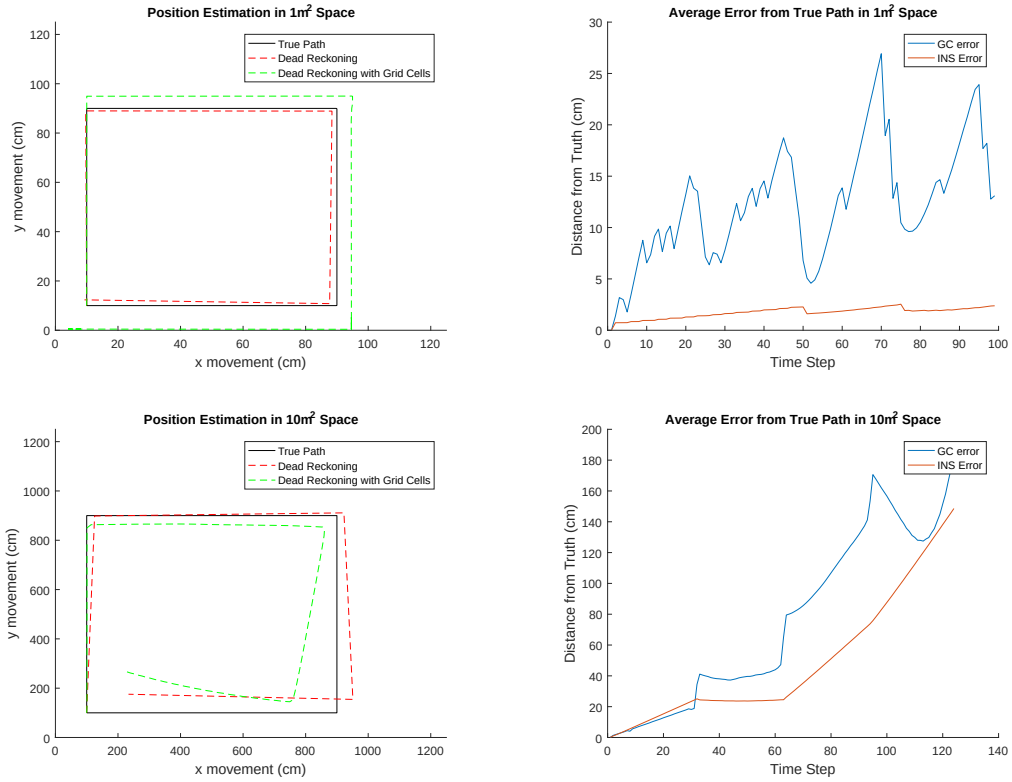



Figure 4.14 a) and b) show data for a 1 $m^2$ space, and c) and d) show data for 10 $m^2$ space. These values are the average value for 1000 runs with 1 standard deviation of noise using 4.1. The average does not show improvement unlike the results in 4.11 that uses system described in Figure 4.9.

One standard deviation of noise only shows noise that will happen at most 68% of the time. That means, that in order to get more complete results, the results for 2 and 3 standard deviations should be shown in order to include noise that will happen up to 99.7% of the time. Figure 4.15 show the average path and error of 1000 runs for the paths with 3 standard deviations of noise.

Table 4.3 summarizes the results that characterize the improvement seen when using the grid cell aided navigation systems with 1, 2, and 3 standard deviations of noise applied. The data in Table 4.3 indicates that the Grid Cell aided systems show better performance with longer paths and higher noise values.

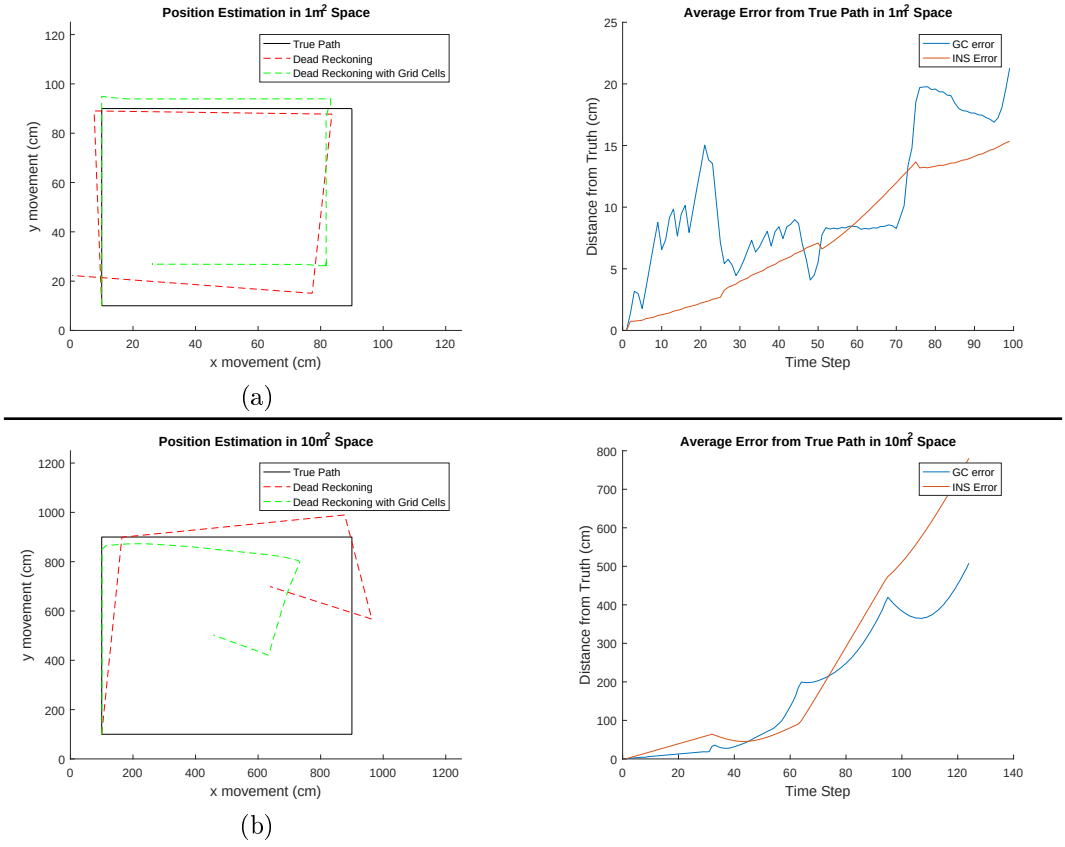



Figure 4.15 a) shows data for a 1 $m^2$ space, and b) show data for 10 $m^2$ space. These values are the average value for 1000 runs with 3 standard deviation of noise. c) and d) show that longer paths and higher error show better performance with grid cells aided navigation.

### *Grid Cell Variations*

Grid Cells come in many different sizes. Their capabilities cannot be fully tested by only using one set of grid cells. In order to evaluate the algorithm's sensitivity to grid size, sets that span the range of observed grid cell spacings were run with all three sigmas of errors within the 10 $m^2$ space. Five different spacing sets were chosen to cover the range. Each set is composed of six grid cells. The sets are: {5, 7, 11, 13, 17, 23}, {29, 31, 37, 41, 43, 47}, {53, 59, 61, 67, 71, 73}, {101, 103, 107, 109, 113, 127}, and {179, 181, 191, 193, 197, 199} with all values in cm. The 10 $m^2$ space contains positions that go beyond the largest observed grid cell spacing, making it a viable space to test out the range of the grid cells.

Table 4.3 The mean, standard deviation, min, and max of the improvement of the combined systems from inertial navigation alone. The combined systems show better performance at longer paths and higher noise values.

| **Noise** | **Mean** | **Std. Dev** | **Min** | **Max** |
|---|---|---|---|---|
| | | 1 $m^2$ Space | | |
| $1\sigma$ | -10.581 | 5.182 | -24.642 | 0 |
| $2\sigma$ | -1.861 | 4.705 | -10.967 | 2.894 |
| $3\sigma$ | -.2920 | 3.529 | -12.728 | 3.699 |
| | | 10 $m^2$ Space | | |
| $1\sigma$ | -26.763 | 26.481 | -94.771 | 5.434 |
| $2\sigma$ | -45.291 | 36.6818 | -115.706 | 0 |
| $3\sigma$ | 57.240 | 97.914 | -99.998 | 276.072 |

Table 4.4 reveals how differently sized grid cell spacings affect the performance. The largest improvement was shown to be 57.246 cm on average. With the lowest standard deviation reads at 20.243 cm. The best improvements are shown to come from grid cell spacings between the 40-80 cm spacings.

Table 4.4 The mean, standard deviation, min, and max of the different grid cell spacings are shown on this table. The data for the second and third spacing show to be the better grid cell sizes for improving position estimations.

| **GC Spacings (cm)** | **Mean** | **Std. Dev** | **Min** | **Max** |
|---|---|---|---|---|
| | | $1\sigma$ noise | | |
| 5, 7, 11, 13, 17, 23 | -12.685 | 23.352 | -101.255 | 4.670 |
| 29, 31, 37, 41, 43, 47 | -22.601 | 20.243 | -92.750 | 8.964 |
| 53, 59, 61, 67, 71, 73 | -27.458 | 24.538 | -109.763 | 9.235 |
| 101, 103, 107, 109, 113, 127 | -24.608 | 26.329 | -104.158 | 22.193 |
| 179, 181, 191, 193, 197, 199 | -63.443 | 29.949 | -162.314 | 7.857 |
| | | $2\sigma$ noise | | |
| 5, 7, 11, 13, 17, 23 | -19.933 | 60.640 | -111.731 | 143.121 |
| 29, 31, 37, 41, 43, 47 | -21.800 | 63.683 | -153.682 | 64.017 |
| 53, 59, 61, 67, 71, 73 | 41.933 | 31.848 | -135.707 | 3.546 |
| 101, 103, 107, 109, 113, 127 | 13.699 | 52.421 | -113.320 | 82.809 |
| 179, 181, 191, 193, 197, 199 | -5.218 | 47.706 | -143.581 | 77.684 |
| | | $3\sigma$ noise | | |
| 5, 7, 11, 13, 17, 23 | 42.152 | 103.623 | -46.325 | 305.117 |
| 29, 31, 37, 41, 43, 47 | 8.801 | 116.764 | -152.167 | 294.357 |
| 53, 59, 61, 67, 71, 73 | 51.951 | 105.073 | -97.777 | 289.330 |
| 101, 103, 107, 109, 113, 127 | 57.246 | 114.960 | -75.063 | 301.573 |
| 179, 181, 191, 193, 197, 199 | 1.566 | 109.684 | -171.735 | 203.854 |

# CHAPTER FIVE

## Discussion

A series of assumptions were used throughout the thesis. The first assumption is that changes in acceleration and rotation are instantaneous. This means the simulated travel on the path is formed with instantaneous acceleration changes with no loss of momentum. The focus of this thesis is in the inaccuracies of the sensors to see the value in implementing grid cells in navigation on an physical system.

Another assumption is that grid cells hold a uniform lattice and larger lattices are scaled versions of the smaller lattices. This is the main assumption for the equations derived in [21] that were used in this thesis.

One more assumption made in [21] was that grid cells with a regular periodic response hold the same period throughout their use. Research provides evidence that the assumption is correct, but tests still need to be done on the large scale travels with rats [21].

### *Future Work*

A next step for this project is to implement grid cell aided navigation on a robot that will travel the square path as in this thesis. The full dynamics of motion can be observed as well. A physical proof of improvement on the inertial navigation system with grid cells will allow for further projects with grid cell navigation to move forward.

Though the scope of this thesis is in dead reckoning/idiothetic navigation, grid cell navigation has shown improved navigation results in other forms of navigation. Research in algorithmic place cell development [36] [40], visual odometry with grid cells [50], and other forms of improvement in path integration and simultaneous local-

ization and mapping [12] show that grid cell navigation is a viable way for autonomous navigation to move forward.

One positive aspect of mimicking grid cell firing patterns on electronic devices is that lattices are not limited on electronic devices. For dead reckoning using grid cells, sizes of lattice periods outside the natural range are not a problem and could be applied without the limitations of physical neurons (keeping the assumption of periodicity not changing). If the implementation of the neuromorphic circuit were done, the grid cells could be made as small as possible to increase the resolution of the grid cell readings.

However, the physical implementations of neuromorphic circuits are the key to implementing the grid cell based navigation used in this thesis. Since we do not have that, the grid cells will rely on current navigation inputs from accelerometers and gyroscopes for spiking patterns, and it is not as accurate as modeled grid cells. The simulation of current physical capabilities shows that vector navigation can improve dead reckoning in select situations.

There have been experiments with algorithms that use visual sensors [50] to work with grid cells for navigation. This, however, adds visual cues, formations of place cells, and reliance on other sensors. This thesis looks upon the improvement of the dead reckoning method that is available without these external cues.

One proposed way of achieving grid cell spiking, is to use different sized "wheels" to represent the different lattices formed by grid cells. As the wheels rotate, the grid cell firings are stimulated to emulate the consistent firing periods described in this thesis. With the formation of consistent firing periods with wheels, grid cell stimulation occurs without the need for an INS to drive the grid cells like in this thesis. This can be seen in Figure 5.1.

Slippage is a main concern in implementing grid cells with wheels. If the wheels desynchronize, it would increase translational drift. A wheel with a 25 cm circumfer-

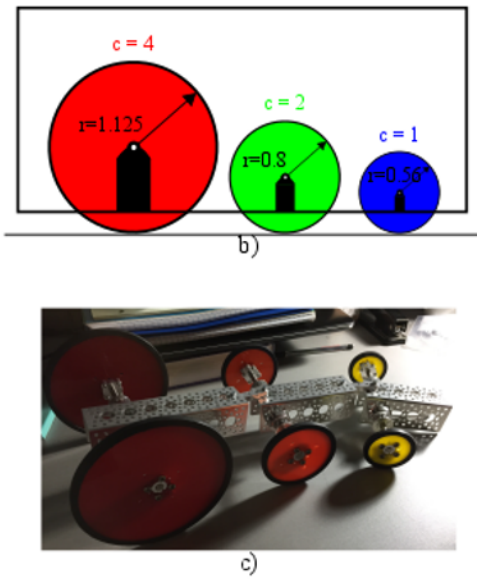



Figure 5.1 This figure (taken from [1]) shows a way in which grid cell firing patterns can be established without having to rely on an INS for input.

ence, if desynchronized by 10 mm on each wheel rotation, would be off by 1 meter after having traveled 250 m. If the desynchronization were increasing steadily by 10 mm, then at those 250 m traveled, the location reading would be off by 50.5 m.

## *Conclusion*

In this thesis, dead reckoning with only an inertial navigation system and dead reckoning with a grid cell aided inertial navigation system are shown to be comparable. This thesis proposes that grid cell dead reckoning is a viable form of autonomous and idiothetic navigation, that in select situations removes error and improves performance on average. An inertial navigation system with and without grid cell aid were simulated traveling a square of varying lengths and covering different amounts of space. The inertial navigation system was simulated by modeling accelerometer and gyroscope readings, and then estimating a position using the kinematic equations. The grid cell aid was simulated by mapping the navigation space through grid cell models and using INS provided velocities for localization. The paths were built to test

within and past the distance limitations observed through a rat's grid cell firing patterns as they navigated deprived of their senses. The simulated results validate that the grid cell aided navigation is comparable and, in some cases, better in accuracy than dead reckoning alone through different paths and spaces.

In order to progress the experiment in this thesis, physical implementation needs to be pursued to show physical results in dead reckoning and its combination with grid cell mapping and localization. It is also in the best interest to pursue a physical implementation of vector navigation in order to show the accuracy of vector navigation on its own.

# APPENDICES

# APPENDIX A

## Top Level Code for Simulation

```
clear;
close all;


%% Constants

%Making sure the seed stays the same. Any int works here.
%You can comment the function out to have a random seed every
   time.
rng(1);

global NO_NOISE
NO_NOISE = 0;

n = .1;
mapbounds = 100;
vector = 1:n:mapbounds;
precision = 1;
sam = mapbounds;
neuralmapsize = 10^5;
scaling = mapbounds/100;

params.accelnoise = .000382;
params.accelbias = 0.0003924;
```

```
params.accelsf = [1+.000382 randn*0.05 ; randn*0.05 1+.000382];
params.gyronoise = 0.038;
params.gyrobias = .006891;
params.gyrosf = 1+.0069565;

intervals = [5 7 11 13 17 23];

%% Setting Up Vectors

[sqrx, sqry] = sqr(vector, precision, 10*scaling, 80*scaling,
   mapbounds);
disp('Path Set');

%% Forming Map

[xvec,yvec,xgccombos,ygccombos] = gridcellmap(intervals,
   mapbounds, scaling);
disp('GC map formed.');

[gcsqrxpos, gcsqrypos, inssqrxpos, inssqrypos ] = ...
    localize( sqrx, sqry, xvec, yvec, xgccombos, ygccombos,
       intervals, params, neuralmapsize);

disp('INS Done');

%% Error Calculation

gcsqrid = errdist(gcsqrxpos, gcsqrypos, sqrx, sqry);
```

```
inssqrid = errdist(inssqrxpos, inssqrypos, sqrx, sqry);

improvement = inssqrid - gcsqrid;

disp('Error Calc Done')
```

code/OneRunResults.m

# APPENDIX B

## Code for Equations

```
function [accelnav, velnav] = IMU(sqrx, sqry, params)
%This builds the output for the INS from a simulated
   accelerometer and
%gyroscope. The output is the navigation frame acceleration and
    velocity.

%% Constants

%Whether or not we choose to include noise
global NO_NOISE;

% This is the time step in seconds.
dt = .1;

%% Accelerations

%Instantiating the true acceleration
trueaccel = zeros(2,(size(sqrx,2)));

disp('Forming true acceleration vector')

%Setting up the accelerations to move around the true square
   path
trueaccel(2,1) = 50*.64;                              %Y accel
```

```
trueaccel(2,round((size(sqrx,2)*.25))) = -50*.64;    %Y de-accel

trueaccel(1,round((size(sqrx,2)*.25))+1) = 50*.64;   %X accel
trueaccel(1,round((size(sqrx,2)*.5))) = -50*.64;     %X de-accel

trueaccel(2,round((size(sqrx,2)*.5))+1) = -50*.64;   %Y accel
trueaccel(2,round((size(sqrx,2)*.75))) = 50*.64;     %Y de-accel

trueaccel(1,round((size(sqrx,2)*.75))+1) = -50*.64; %X accel

disp('Finished forming true acceleration vector');


%Instantiating angular velocity and angle
truetheta(1) = 90*pi/180;  %initial angle (Truth)
truethetadot(1) = 0;       %initial angular velocity (Truth)

%Instantiating velocity for calculations
vel = zeros(2,(size(sqrx,2)));

%True Velocity (Navigation Frame)
for i = 2:size(vel,2)

    vel(:,i) = vel(:, i-1) + trueaccel(:,i-1)*dt;

end

%% Angles
```

```
disp('Forming true angle vector.');

for i = 2:size(sqrx,2)

    % sqrx and sqry are true position
    % calculate heading from the true positions

    truetheta(i) = atan2(sqry(i)-sqry(i-1),sqrx(i)-sqrx(i-1));

    if and(truetheta(i) > pi/2, truetheta(i-1) < -pi/2)

        truetheta(i) = -truetheta(i);

    end

    % calculate angular rate
    if (truetheta(i) - truetheta(i-1)) < pi/2
        truethetadot(i) = (truetheta(i) - truetheta(i-1))/dt;
    else
        truethetadot(i) = (2*pi-(truetheta(i) - truetheta(i-1))
           )/dt;
    end

end

disp('Finished forming true angle vector')
```

```
%% Simulate Accelerometer and Gyroscope from True Data

disp('Simulating Accelerometer and Gyroscope Sensors');

%Rotating from true data into body data in order to simulate
   sensors.
for i = 1:size(trueaccel,2)

    accelbody(:,i) = [cos(truetheta(i)) sin(truetheta(i)); -sin
       (truetheta(i)) cos(truetheta(i))]*trueaccel(:,i);

end

%Initialize Simulated Sensor Data
sensortheta(1) = truetheta(1);
INSpos(:,1) = [sqrx(1); sqry(1)];
velnav(:,i) = [0; 0];

accelnoise = params.accelnoise;
accelbias = params.accelbias;
accelsf = params.accelsf;
gyronoise = params.gyronoise;
gyrobias = params.gyrobias;
gyrosf = params.gyrosf;

for i = 2:length(accelbody)

    %Establishing rotation matrix for body to navigation frame
```

```
Rb2N = [cos(sensortheta(i-1)) -sin(sensortheta(i-1)); sin(
   sensortheta(i-1)) cos(sensortheta(i-1))];  %Using
   previous heading estimate


if NO_NOISE == 1
   accelnav(:,i) = Rb2N*accelbody(:,i-1);  %rotate the True
       measured Body acclerations into Navigation frame
      using the heading which is estimated based on no
      noise
else
    noisyaccel(:,i) = accelsf * accelbody(:,i-1) +
       accelnoise*randn + accelbias*randn; %add noise to
       the Body Frame accelerometer
    accelnav(:,i) = Rb2N*noisyaccel(:,i);  %rotate the
       noisy measured Body acclerations into Navigation
       frame using the estimate of the heading based on
       they gyro output
end

%Integrate acceleration to find current estimate of
   Velocity (Nav frame)
velnav(:,i) = velnav(:,i-1) + accelnav(:,i)*dt;

%Integrate velocity to find current estimate of position (
   Nav frame)
INSpos(:,i) = INSpos(:,i-1) + velnav(:,i)*dt;
```

```
    %Estimate the NEW heading (Navigation frame) based on Gyro
       data
    if NO_NOISE == 1
        sensortheta(i) = sensortheta(i-1) + truethetadot(i)*dt;
             %Gyroscope Input (truth)
    else
        noisygyro(i) = truethetadot(i)*gyrosf + gyrobias*randn
           + gyronoise*randn; %add noise to the true angular
           velocity
        sensortheta(i) = sensortheta(i-1) + noisygyro(i)*dt;  %
           Gyroscope Input
    end

end

disp('Simulation of accelerometer and gyroscope complete.');

end
```

code/IMU.m

```
function [ gcposestx, gcposesty, drposestx, drposesty ] =
   INSwGCaid(sqrx, sqry, velnav, ...
    accelnav, xvec, yvec, xgcombos, ygccmobos, intervals,
       neuralmapsize, timescale)
%% Kinematic

% Time Step
ts = .1;
```

```
% Initializing State Space Matrices

xstatespace = [sqrx; velnav(1,:); accelnav(1,:)];
ystatespace = [sqry; velnav(2,:); accelnav(2,:)];

statex(:,1) = xstatespace(:,1);
statey(:,1) = ystatespace(:,1);

%% Trapeziodal Estimations

%Integration through Trapezpoidal Approximation

disp('Integrating for INS estimations!');

for i = 2:size(xstatespace,2)

statex(3,i) = xstatespace(3,i);
statey(3,i) = ystatespace(3,i);

statex(2,i) = statex(2,i-1)+((statex(3,i-1)+((statex(3,i)-
   statex(3,i-1))/2))*ts);
statey(2,i) = statey(2,i-1)+((statey(3,i-1)+((statey(3,i)-
   statey(3,i-1))/2))*ts);

statex(1,i) = statex(1,i-1)+((statex(2,i-1)+((statex(2,i)-
   statex(2,i-1))/2))*ts);
statey(1,i) = statey(1,i-1)+((statey(2,i-1)+((statey(2,i)-
   statey(2,i-1))/2))*ts);
```

```
end

%Localization through grid cell spatial map.

disp('Using Grid Cell Aid!');

[gcposestx,gcposesty] = posfromGCmap(intervals, xgcombos,
   ygccmobos, ...
 xvec, yvec, sqrx, sqry, statex(2,:),statey(2,:), neuralmapsize
    , timescale);

%Saving INS estimations without grid cell aid.

[drposestx] = statex(1,:);
[drposesty] = statey(1,:);

end
```

code/INSwGCAid.m

```
function [xvec, yvec, xgccombos, ygccombos] = gridcellmap(
   intervals, mapbounds, scaling)

%gridcellmap will form the spatial map allowed by the
   combinations of grid
%cell firing intervals.

%Number of modules
m = size(intervals,2);
```

```
%Ideal Mapping Capacity
%phi = lcm(lcm(lcm(lcm(lcm(intervals(1),intervals(2)),intervals
   (3)), ...
%    intervals(4)), intervals(5)), intervals(6)) - 1;

%Number of distinguishable phases
invphi = sum(intervals.*intervals/mapbounds*scaling);

%Mapping capacity
capacity = (invphi)^(m-1);

%Killing the program if the capacity is not enough.
if (capacity < mapbounds)
   under = mapbounds - capacity;
   fprintf('Your capacity covers up to %f and needs to increase
       capacity by %f.'...
       , capacity, under);
   return
end

%Setting neuron spikes through mapping space for x and y
xmaxfire1 = 0:intervals(1):mapbounds;
xmaxfire2 = 0:intervals(2):mapbounds;
xmaxfire3 = 0:intervals(3):mapbounds;
xmaxfire4 = 0:intervals(4):mapbounds;
xmaxfire5 = 0:intervals(5):mapbounds;
xmaxfire6 = 0:intervals(6):mapbounds;
```

```
ymaxfire1 = 0:intervals(1):mapbounds;
ymaxfire2 = 0:intervals(2):mapbounds;
ymaxfire3 = 0:intervals(3):mapbounds;
ymaxfire4 = 0:intervals(4):mapbounds;
ymaxfire5 = 0:intervals(5):mapbounds;
ymaxfire6 = 0:intervals(6):mapbounds;

%Forming the combinations of spikes across the map
xgccombos = union(union(union(union(union(xmaxfire1, xmaxfire2)
   , ...
    xmaxfire3), xmaxfire4), xmaxfire5), xmaxfire6);

disp(xgccombos);

ygccombos = union(union(union(union(union(ymaxfire1, ymaxfire2)
   , ...
    ymaxfire3), ymaxfire4), ymaxfire5), ymaxfire6);

%This will not map phases at every integer value, so if these
   phases are
%mapped, they will look odd. It however is not.

%Instantiating the X axis and Y axis representation of phases

xvec = zeros(1,size(xgccombos,2));
yvec = zeros(1,size(ygccombos,2));
```

```
%Setting phase representations for every spike combination.

disp('Building Spatial Map');

for i = 1:(size(xgccombos,2)-1)

    %This sets every X axis location representation

    for k = 1:m

        %Setting the phase representation
        xvec(k,i) = (mod(xgccombos(i), intervals(k))/intervals(
           k))*2*pi;

    end

    %This sets every Y axis location tied to each X axis
       location rep

    for j = 1:(size(ygccombos,2)-1)

        for k = 1:m

            %Setting the phase representation
            yvec(k,j) = (mod(ygccombos(j),intervals(k))/
               intervals(k))*2*pi;

        end
```

```
    end

end

disp('Finished Building Spatial Map');

end
```

code/gridcellmap.m

```
function [ yourxpos, yourypos ] = posfromGCmap(intervals,
   xgccombos, ygccombos ...
, xvec, yvec, sqrx, sqry, velx, vely, neuralmapsize, scale)
% posfromGCmap will decode the position of the object to
   whatever motion you give
% it with the path vectors. xvec and yvec are sets of phases
   that encode locations
% on each axis.

%% Constants

m = size(intervals, 2); %modules
ts = .1; %time step in seconds
cv = 1/8; % coefficient of variance (ratio)
scalefix = 1;


phthresh1 = min(intervals)*pi/intervals(1); %threshold for
   firing interval 1
```

```
phthresh2 = min(intervals)*pi/intervals(2); %threshold for
   firing interval 2
phthresh3 = min(intervals)*pi/intervals(3); %threshold for
   firing interval 3
phthresh4 = min(intervals)*pi/intervals(4); %threshold for
   firing interval 4
phthresh5 = min(intervals)*pi/intervals(5); %threshold for
   firing interval 5
phthresh6 = min(intervals)*pi/intervals(6); %threshold for
   firing interval 6

%% Find Position from GC Map

disp('Getting positions from GC spatial map.');



for i = 1:size(velx(1,1:scale:(end-scalefix)),2)

   %positions from INS through trapezoidal estimation

   if i > 1
   sqrx(i) = sqrx(i-1)+((velx(round(i*scale-(scale-1)))+((velx(
      round(i*scale))-velx(round(i*scale-(scale-1))))/2))*ts*
      scale);
   sqry(i) = sqry(i-1)+((vely(round(i*scale-(scale-1)))+((vely(
      round(i*scale))-vely(round(i*scale-(scale-1))))/2))*ts*
      scale);
```

```
end

for k = 1:m

     % Making each position estimation into a phase value
     xoi(i,k) = mod(abs(round(sqrx(i))),intervals(k))/
        intervals(k)*2*pi;
     yoi(i,k) = mod(abs(round(sqry(i))),intervals(k))/
        intervals(k)*2*pi;

end

 % separating out the phases
 xph1 = xoi(i,1);
 xph2 = xoi(i,2);
 xph3 = xoi(i,3);
 xph4 = xoi(i,4);
 xph5 = xoi(i,5);
 xph6 = xoi(i,6);

 yph1 = yoi(i,1);
 yph2 = yoi(i,2);
 yph3 = yoi(i,3);
 yph4 = yoi(i,4);
 yph5 = yoi(i,5);
 yph6 = yoi(i,6);

 %Finding the spatial map index for the x position
```

```
a = find(xvec(1,:) + phthresh1 > xph1  & xvec(1,:)-
   phthresh1 < xph1 ...
    & xvec(2,:) + phthresh2 > xph2  & xvec(2,:)- phthresh2
       < xph2  ...
    & xvec(3,:) + phthresh3 > xph3  & xvec(3,:)- phthresh3
       < xph3  ...
    & xvec(4,:) + phthresh4 > xph4  & xvec(4,:)- phthresh4
       < xph4  ...
    & xvec(5,:) + phthresh5 > xph5  & xvec(5,:)- phthresh5
       < xph5  ...
    & xvec(6,:) + phthresh6 > xph6  & xvec(6,:)- phthresh6
       < xph6 );

if isempty(a)
    xvec_save = xvec;  %save the original xvec before
       changing zero values

for indexx = 1:length(intervals)
    zero_indices = find(xvec(indexx,:) == 0);
    xvec(indexx,zero_indices) = 2*pi;
end

a = find(xvec(1,:) + phthresh1 > xph1  & xvec(1,:)-
   phthresh1 < xph1 ...
    & xvec(2,:) + phthresh2> xph2  & xvec(2,:)- phthresh2 <
        xph2  ...
```

```
    & xvec(3,:) + phthresh3> xph3  & xvec(3,:)- phthresh3 <
        xph3  ...
    & xvec(4,:) + phthresh4> xph4  & xvec(4,:)- phthresh4 <
        xph4  ...
    & xvec(5,:) + phthresh5> xph5  & xvec(5,:)- phthresh5 <
        xph5  ...
    & xvec(6,:) + phthresh6> xph6  & xvec(6,:)- phthresh6 <
        xph6 );

    xvec = xvec_save; %return xvec back to what it was
       before to capture the other side of the symmetry

end

b = find(yvec(1,:) + phthresh1 > yph1  & yvec(1,:) -
   phthresh1 < yph1 ...
    & yvec(2,:) + phthresh2 > yph2  & yvec(2,:) - phthresh2
        < yph2 ...
    & yvec(3,:) + phthresh3 > yph3  & yvec(3,:) - phthresh3
        < yph3 ...
    & yvec(4,:) + phthresh4 > yph4  & yvec(4,:) - phthresh4
        < yph4 ...
    & yvec(5,:) + phthresh5 > yph5  & yvec(5,:) - phthresh5
        < yph5 ...
    & yvec(6,:) + phthresh6 > yph6  &  yvec(6,:) -
       phthresh6 < yph6 );

if isempty(b)
```

```
    yvec_save = yvec;  %save the original yvec before
       changing zero values

for indexy = 1:length(intervals)
    zero_indices = find(yvec(indexy,:) == 0);
    yvec(indexy,zero_indices) = 2*pi;
end

b = find(yvec(1,:) + phthresh1 > yph1  & yvec(1,:) -
   phthresh1 < yph1 ...
    & yvec(2,:) + phthresh2 > yph2  & yvec(2,:) - phthresh2
        < yph2 ...
    & yvec(3,:) + phthresh3 > yph3  & yvec(3,:) - phthresh3
        < yph3 ...
    & yvec(4,:) + phthresh4 > yph4  & yvec(4,:) - phthresh4
        < yph4 ...
    & yvec(5,:) + phthresh5 > yph5  & yvec(5,:) - phthresh5
        < yph5 ...
    & yvec(6,:) + phthresh6 > yph6  &  yvec(6,:) -
       phthresh6 < yph6 );

yvec = yvec_save; %return yvec back to what it was before
   to capture the other side of the symmetry

end

%debugging purposes
if i ==1
```

```
xovecpos(i) = round(sqrx(1)/intervals(1))+round(sqrx(1)/
   intervals(2))+...
round(sqrx(1)/intervals(3))+round(sqrx(1)/intervals(4))+...
round(sqrx(1)/intervals(5))+round(sqrx(1)/intervals(6));
elseif isempty(a)
xovecpos(i) = xovecpos(i-1);
elseif le(abs(sqrx(i)),min(intervals)/2)
xovecpos(i) = 1;
elseif size(a,2) > 1
bigx = bigx + 1;
xovecpos(i) = a(size(a,2));
else
xovecpos(i) = a;
end

if i ==1
yovecpos(i) = round(sqry(1)/intervals(1))+round(sqry(1)/
   intervals(2))+...
round(sqry(1)/intervals(3))+round(sqry(1)/intervals(4))+...
round(sqry(1)/intervals(5))+round(sqry(1)/intervals(6));
elseif isempty(b)
yovecpos(i) = yovecpos(i-1);
elseif le(abs(sqry(i)),min(intervals)/2)
yovecpos(i) = 1;
elseif size(b,2) > 1
yovecpos(i) = b(1);%b(size(b,2));
else
yovecpos(i) = b;
```

```
    end

    for g = 1:size(xgccombos,2)

    % Deterioration of the map (translation of tha map over
       time)
    xgccombos(g) = xgccombos(g) + round(sqrt((i-1)/
       neuralmapsize*cv^2),3);
    ygccombos(g) = ygccombos(g) + round(sqrt((i-1)/
       neuralmapsize*cv^2),3);

    end

    if i == 1
        yourxpos(i) = sqrx(1);
    else
        yourxpos(i) = xgccombos(xovecpos(i));
    end

    if i == 1
        yourypos(i) = sqry(1);
    else
        yourypos(i) = ygccombos(yovecpos(i));
    end

    sqrx(i) = yourxpos(i);
    sqry(i) = yourypos(i);
end
```

```
end
```

code/posfromGCmap.m

```
function [ sqrx, sqry ] = sqr( Vector, Precision, Start, Length
   , mapbounds )
%sqr is a function that forms a square in the middle of the
   boundaries of
%the space allocated.

disp('Forming true path.');

for i = 1:size(Vector,2)

    if Start > mapbounds | Start+Length > mapbounds | Start < 0
        | Start+Length < 0

        disp('The Start or Length is outside of the boundaries'
           );
        return;

    else

        if i > (3*size(Vector,2)/4)

            sqrx(i) = round(Start+Length-((i-(3*size(Vector,2)
               /4))*Length...
                /(size(Vector,2)/4)),Precision);
            sqry(i) = Start;
```

```
elseif i > (size(Vector,2)/2)

    sqrx(i) = Start+Length;
    sqry(i) = round(Start+Length-((i-(size(Vector,2)/2)
       )*Length...
        /(size(Vector,2)/4)),Precision);

elseif i > (size(Vector,2)/4)

    sqrx(i) = round(Start+((i-(size(Vector,2)/4))*
       Length...
        /(size(Vector,2)/4)),Precision);
    sqry(i) = Start+Length;

elseif i > 1

    sqrx(i) = Start;
    sqry(i) = round(Start+(i*Length/(size(Vector,2)/4))
       ,Precision);

elseif i > 0

    sqrx(i) = Start;
    sqry(i) = Start;

end
```

```
        if sqrx(i) > mapbounds
            sqrx(i) = round(sqrx(i)/mapbounds,Precision);
        end

        if sqry(i) > mapbounds
            sqry(i) = round(sqry(i)/mapbounds,Precision);
        end

    end

end

disp('Finished building true path.');

end
```

code/sqr.m

```
function [ ierrt ] = errdist(yourxpos, yourypos, ox, oy)
%errorcalc ends up calculating the error between the true path
   and
%the predicted path

% Instantaneous error

for i = 1: size(ox,2)

if round(ox(i),4) == 0 & round(oy(i),4) == 0

ierrx(i) = 0;
```

```
ierry(i) = 0;

elseif round(ox(i),4) == 0

ierrx(i) = 0;
ierry(i) = abs((yourypos(i)-oy(i)));

elseif round(oy(i),4) == 0

ierrx(i) = abs((yourxpos(i)-ox(i)));
ierry(i) = 0;

else

ierrx(i) = abs((yourxpos(i)-ox(i)));
ierry(i) = abs((yourypos(i)-oy(i)));

end

end
        ierrt = sqrt(ierrx.^2 + ierry.^2);
```

code/errdist.m

# BIBLIOGRAPHY


[1] S. Koziol, "Combining neuroscience and electrical engineering for brain inspired robot navigation," *Baylor University Technical Report*, no. 347-15, 2016.

[2] D. Bush, C. Barry, D. Manson, and N. Burgess, "Using grid cells for navigation," *Neuron*, vol. 87, no. 3, pp. 507–520, 2015.

[3] S. M. Koziol, "Reconfigurable analog circuits for path planning and image processing," Ph.D. dissertation, Georgia Institute of Technology, 2013.

[4] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2004.

[5] O. J. Woodman, "An introduction to inertial navigation," *696, University of Cambridge Computer Laboratory*, 2007.

[6] L. F. J. Borenstein, "Measurement and correction of systematic odometry errors in mobile robots," *Robotics and Automation*, vol. 12, no. 6, 1996.

[7] A. Werries and J. M. Dolan, "Adaptive kalman filtering methods for low-cost gps/ins localization for autonomous vehicles," *Research Showcase at Carnegie Melon University*, 2016.

[8] B. hui Tang and Z. xing Zhou, "The application of equipment positioning system in the based on the distributed of integrated navigation and multi-sensor data fusion," in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, June 2017, pp. 951–955.

[9] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001.

[10] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[11] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 1403–1410 vol.2.

[12] M. J. Milford, G. F. Wyeth, and D. Prasser, "Ratslam: a hippocampal model for simultaneous localization and mapping," in *Robotics and Automation, 2004.*

*Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, April 2004, pp. 403–408 Vol.1.

[13] V. Hart, P. Nováková, E. P. Malkemper, S. Begall, V. Hanzal, M. Ježek, T. Kušta, V. Němcová, J. Adámková, K. Benediktová *et al.*, "Dogs are sensitive to small variations of the earth's magnetic field," *Frontiers in zoology*, vol. 10, no. 1, p. 80, 2013.

[14] C. Nießner, S. Denzau, E. P. Malkemper, J. C. Gross, H. Burda, M. Winklhofer, and L. Peichl, "Cryptochrome 1 in retinal cone photoreceptors suggests a novel functional role in mammals," *Scientific reports*, vol. 6, p. 21848, 2016.

[15] K. J. Lohmann, N. F. Putman, and C. M. Lohmann, "Geomagnetic imprinting: a unifying hypothesis of long-distance natal homing in salmon and sea turtles," *Proceedings of the National Academy of Sciences*, vol. 105, no. 49, pp. 19 096–19 101, 2008.

[16] M. M. Walker, T. E. Dennis, and J. L. Kirschvink, "The magnetic sense and its use in long-distance navigation by animals," *Current opinion in neurobiology*, vol. 12, no. 6, pp. 735–744, 2002.

[17] Y. Burak and I. R. Fiete, "Accurate path integration in continuous attractor network models of grid cells," *PLoS computational biology*, vol. 5, no. 2, p. e1000291, 2009.

[18] A. S. Etienne, S. J. Lambert, B. Reverdin, and E. Teroni, "Learning to recalibrate the role of dead reckoning and visual cues in spatial navigation," *Learning & behavior*, vol. 21, no. 3, pp. 266–280, 1993.

[19] J. M. Loomis, R. L. Klatzky, R. G. Golledge, and J. W. Philbeck, "Human navigation by path integration," *Wayfinding behavior: Cognitive mapping and other spatial processes*, pp. 125–151, 1999.

[20] C. R. Gallistel, *The Organization of Learning*. MIT Press Cambridge, MA, 1990.

[21] I. R. Fiete, Y. Burak, and T. Brookings, "What grid cells convey about rat location," *Journal of Neuroscience*, vol. 28, no. 27, pp. 6858–6871, 2008.

[22] J. S. Taube, R. U. Muller, and J. B. Ranck, "Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis," *Journal of Neuroscience*, vol. 10, no. 2, pp. 420–435, 1990.

[23] J. Perez-Orive, O. Mazor, G. C. Turner, S. Cassenaer, R. I. Wilson, and G. Laurent, "Oscillations and sparsening of odor representations in the mushroom body," *Science*, vol. 297, no. 5580, pp. 359–365, 2002.

[24] B. L. Day and R. C. Fitzpatrick, "The vestibular system," *Current biology*, vol. 15, no. 15, pp. R583–R586, 2005.

[25] Z. Liu, "Introduction to inertial navigation and pointing control," *Zhang Liu Tutorials*, 2011.

[26] NovAtel, "Imu errors and their effects," *NovAtel Bulletins*, vol. APN-064 Rev A, 2014.

[27] C.-W. Tan and S. Park, "Design and error analysis of accelerometer-based inertial navigation systems," *California Partners for Advanced Transit and Highways (PATH)*, 2002.

[28] K. M. Smalling and K. W. Eure, "A short tutorial on inertial navigation system and global positioning system integration," *NASA STI Technical Memorandum*, no. L-20602, 2015.

[29] K. K. Lekkala and V. K. Mittal, "Accurate and augmented navigation for quadcopter based on multi-sensor fusion," in *2016 IEEE Annual India Conference (INDICON)*, Dec 2016, pp. 1–6.

[30] F. Santoso, M. A. Garratt, and S. G. Anavatti, "Visual inertial navigation systems for aerial robotics: Sensor fusion and technology," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 260–275, Jan 2017.

[31] K. Seifert and O. Camacho, "Implementing positioning algorithms using accelerometers," *Freescale Scemiconductor Application Note*, vol. AN3391 Rev 0, 2007.

[32] J. O'Keefe and J. Dostrovsky, "The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat," *Brain research*, vol. 34, no. 1, pp. 171–175, 1971.

[33] G. J. Quirk, R. U. Muller, J. L. Kubie, and J. Ranck, "The positional firing properties of medial entorhinal neurons: description and comparison with hippocampal place cells," *Journal of Neuroscience*, vol. 12, no. 5, pp. 1945–1963, 1992.

[34] B. L. McNaughton, F. P. Battaglia, O. Jensen, E. I. Moser, and M.-B. Moser, "Path integration and the neural basis of the'cognitive map'," *Nature reviews. Neuroscience*, vol. 7, no. 8, p. 663, 2006.

[35] Y. Dordek, D. Soudry, R. Meir, and D. Derdikman, "Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis," *Elife*, vol. 5, p. e10094, 2016.

[36] G. Tejera, A. Barrera, M. Llofriu, and A. Weitzenfeld, "Solving uncertainty during robot navigation by integrating grid cell and place cell firing based on rat spatial cognition studies," in *2013 16th International Conference on Advanced Robotics (ICAR)*, Nov 2013, pp. 1–6.

[37] G. Tejera, M. Llofriu, A. Barrera, and A. Weitzenfeld, "A spatial cognition model integrating grid cells and place cells," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[38] S. Saeidi and F. Towhidkhah, "Different spatial scales in mapping from grid cells to place cells: A neural network model," in *2009 4th International IEEE/EMBS Conference on Neural Engineering*, April 2009, pp. 706–709.

[39] ——, "From grid cells to place cells: A radial basis function network model," in *2008 Cairo International Biomedical Engineering Conference*, Dec 2008, pp. 1–4.

[40] A. Barrera and A. Weitzenfeld, "Biologically-inspired robot spatial cognition based on rat neurophysiological studies," *Autonomous Robots*, vol. 25, no. 1, pp. 147–169, 2008.

[41] L. Kang, V. H. Routh, E. V. Kuzhikandathil, L. D. Gaspers, and B. E. Levin, "Physiological and molecular characteristics of rat hypothalamic ventromedial nucleus glucosensing neurons," *Diabetes*, vol. 53, no. 3, pp. 549–559, 2004.

[42] J. DeFelipe, "Types of neurons, synaptic connections and chemical characteristics of cells immunoreactive for calbindin-d28k, parvalbumin and calretinin in the neocortex," *Journal of chemical neuroanatomy*, vol. 14, no. 1, pp. 1–19, 1997.

[43] T. Dowrick, S. Hall, and L. J. McDaid, "Silicon-based dynamic synapse with depressing response," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 10, pp. 1513–1525, Oct 2012.

[44] J. Lopez-Barneo, C. Darlot, A. Berthoz, and R. Baker, "Neuronal activity in prepositus nucleus correlated with eye movement in the alert cat," *Journal of Neurophysiology*, vol. 47, no. 2, pp. 329–352, 1982.

[45] M.-L. Mittelstaedt and H. Mittelstaedt, "Homing by path integration in a mammal," *Naturwissenschaften*, vol. 67, no. 11, pp. 566–567, 1980.

[46] A. S. Etienne and K. J. Jeffery, "Path integration in mammals," *Hippocampus*, vol. 14, no. 2, pp. 180–192, 2004.

[47] M. C. Fuhs and D. S. Touretzky, "A spin glass model of path integration in rat medial entorhinal cortex," *Journal of Neuroscience*, vol. 26, no. 16, pp. 4266–4276, 2006.

[48] I. Q. Whishaw, "Place learning in hippocampal rats and the path integration hypothesis," *Neuroscience & Biobehavioral Reviews*, vol. 22, no. 2, pp. 209–220, 1998.

[49] B. J. Clark and J. S. Taube, "Vestibular and attractor network basis of the head direction cell signal in subcortical circuits," *Frontiers in neural circuits*, vol. 6, 2012.

[50] H. Lu, J. Xiao, L. Zhang, S. Yang, and A. Zell, "Biologically inspired visual odometry based on the computational model of grid cells for mobile robots," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2016, pp. 595–601.