Methodical Approach to Active Node Layout Generation for Robot Navigation

Guillermo Enriquez\textsuperscript{1}, Sunhong Park\textsuperscript{2} and Shuji Hashimoto\textsuperscript{1}

\textsuperscript{1}Department of Advanced Science and Engineering, Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo 169-855, Japan
\textsuperscript{2}Korea Automotive Technology Institute, Chungnam, S. Korea
E-mail: enriquez@shalab.phys.waseda.ac.jp, shuji@waseda.jp

Abstract In the field of mobile robot navigation, active beacons are a well researched tool to provide an infrastructure for guidance systems. One drawback to the use of active beacons is that their layout is sometimes arbitrary. Moreover, it is often determined solely by the researcher’s preference or experience. While this is occasionally the easiest way to produce results, it is a significant roadblock to the general use of these systems in practical applications, as the layout for any installation must be evaluated by an expert user. In an attempt to formalize the placement of nodes in such a system, we have developed a straightforward approach for the placement of nodes. This approach has also been applied to a software tool which would allow even a novice user to generate layouts for their applications.

Keywords: mobile robots, navigation, wireless sensor networks

1. Introduction

Active beacons are a widely researched tool in the field of mobile robot navigation, for both indoor and outdoor applications. These tools, using technologies such as Wireless Sensor Networks (WSN), active radio frequency identification (RFID), and GPS all require some number of nodes to be distributed throughout the navigable area, which are then detected by a reader mounted on the mobile robot platform. In general, and in contrast to systems using technologies like passive RFID, the number of beacons is relatively low, as the range for active beacons is usually larger when compared to passive ones. With fewer nodes, it would be preferential to place them so as to maximize their utility, and so these layouts are often decided by an expert user. While this can provide efficacious layouts for their given installations, it is a significant obstacle to these types of systems being accepted on a larger scale. If a method was developed that could, independently of navigation algorithm, allow for valid infrastructure layouts, it could contribute to advances for these types of systems. For example, software could be developed to generate these layouts, and rather than an expert user, novices could easily determine how to deploy active beacons to suit their needs. Additionally, these methods could benefit researchers, as a common difficulty in comparing one’s work with that of another team is a disparity in infrastructure. It is possible that a common approach to layouts would simplify this problem.

We begin with a theoretical discussion of certain properties to consider when using virtual potential fields for the purpose of mobile robot navigation. Then, we propose a methodical set of rules developed from those theories and through empirical experience for the generation of active beacon node layouts, such that the approach can be applied to autonomous layout generation software. By analyzing a map representing the navigable area with information such as walls and static obstacles, our approach can determine where nodes should be placed to ensure successful navigations.

Much work has been dedicated to indoor and outdoor navigation approaches which make use of active beacons. Most approaches are based on the concept that knowing the position of a fixed landmark in the navigable area provides the robot...
with information about it’s own position. There are a variety methods by which this can be done, such as some form of triangulation or the overlapping of sensing ranges [1–4]. This concept is of course similar to approaches often used in vision-based navigation where, using SIFT [5] for example, landmarks are identified in a room and tracked. Some researchers also suggest modifications to antennae or tags in order to improve accuracy [6, 7]. While these approaches can produce successful navigations, there are often no attempts to formalize the placement of nodes, and rather the layouts are either random or at the developers discretion. This could be a significant drawback if not all the systems to be used in a given space in the future are robust to certain layouts. Some methods using active beacons make no attempt to estimate the robot’s location implicitly, such as in [8], but rather use some rough assessment of where the robot is in order to move towards it’s intended goal.

The use of a virtual potential field, as shown in [8], is a concept that has been used in various methods of mobile robot navigation. There has been much discussion of the limitations of virtual potential field navigation methods. For example in [9], a key limitation presented is that of “trap situations due to local minima”. The solution cited is one which entails a path planning approach which recalculates a new path when a trap situation is detected. Though our paper focuses on the layouts of infrastructure nodes, the navigational system on which our mathematic model is based handles local minima through the use of “safety points” as part of the map data. These points ensure the virtual potential field suggests a direction that avoids these trap situations. Moreover, though a map is needed, it is only utilized to perform calculations once, when a new goal is chosen. Until a new goal is chosen, the system does not reference the map, and so can continue on with relatively computationally lightweight calculations, as compared to other approaches which consistently monitor the path to avoid trap situations. The problem of passages between tightly spaced obstacles is also discussed in [9]. This too is not a concern for the basis of our model as these tight spaces would be avoided in the initial map analysis stage. Though our layouts currently handle only rooms with no parallel walls with normals pointing in opposite directions, such as “C” shaped areas, by employing the safety points presented in [8], complex situations can possibly be simplified so that the present approach is applicable.

It should be noted that the system in [8] is designed with only static environments in mind. Some systems using virtual potential fields, such as [10], also handle dynamic obstacles. However, a major drawback to the system presented in [10] is the cost of installing the system in a location with a large navigable area. As that system requires detailed location information for not only the robot, but also that of obstacles, the cost in terms of both computation and infrastructure would be significantly larger than for the system on which our mathematical model is based. While a large number of active radio beacons would be needed for a system such as in [8], the cost of these sensors can often be less than the cost of a video camera. Additionally, the cost of RFID tags would be relatively small, as they are placed only as needed and they are inexpensive. Moreover, there are many applications in which dynamic obstacle avoidance is not a requirement, such as in a storage warehouse where no humans are working. That being said, the need for dynamic obstacle avoidance is a desirable quality. In [11], the system on which our model is based was easily combined with a vision subsystem, allowing it to track the movement of a human user in relation to the robot itself. This shows how dynamic obstacle avoid could be added, at lower cost than having the entire area covered by cameras. That is to say, dynamic obstacle avoidance can be achieved with the augmentation of ad-hoc techniques that handle local sensing data.

The importance of node layouts has also been researched with RFID systems, such as in [12]. There are numerous applications for indoor navigation where the layout is mostly static, and rarely changes [13–15]. In light of this, one thing active beacon systems may all have in common is that a structured approach to the placement of these beacons would create not only a standard which could be anticipated when developing new methods for navigation, but also simplify the installation process of the infrastructure needed for these systems to be implemented in real world applications on a large scale. Unlike the above mentioned research, this paper focuses on a methodical approach to the layout of active beacons for the purpose of indoor mobile robot navigation.

This paper is organized as follows: in section 2, we outline the assumptions made in modeling our radio-based navigation system. In section 3, we describe the logical foundations for our node layout rules, which are explained in section 4. Also in section 4, we present some example layouts these rules would produce.

2. Active Beacon-Based Navigation Modeling

While some active beacon-based navigation approaches, such as triangulation, require the capability to accurately estimate the distance from the beacon to the sensor, this is not always a feasibility.
We assume that for a triangulation-based system, or similar approach, that as long as a minimum of three beacons are readable from anywhere in the navigable area the robot can successfully reach the goal. In other words, while the algorithm for placing beacons is not simple, it can be very straightforward, involving mainly a check that a minimum of beacons are within range from every point. Conversely, many methods that do not rely on the precision of distance estimation, such as in [8, 13], the method for determining layouts becomes less obvious, albeit more critical. In order to examine the method for generating active beacon node layouts, let us begin by considering the assumptions, variables, and functions used in our approach.

2.1 Assumptions

For the purposes of developing a methodical approach to active beacon node layout generation we used the navigational method described in [8, 13], as the concept of a virtual potential field is describable mathematically and has the added benefit of not requiring precise localization information, which would require an equally precise model of the radio wave-based nature of most active beacon hardwares. The basic concept of a virtual potential field for mobile robot navigation works by using the received signal strength intensity (RSSI) of incoming signals to a sensor node mounted on the robot from nodes in the navigable area as the magnitudes of vectors calculated by analyzing a map of the area. These vectors collectively form a virtual potential field that the robot moves “down” towards the goal. By calculating vector directions that point around major static obstacles, the robot can avoid them. There are, of course, numerous algorithms and infrastructures that could benefit from a structured approach to node layouts, however, as this is an initial attempt, we have limited our current focus to the virtual potential field paradigm.

For the navigable areas to be considered, we constrain the maps to areas with no free-standing obstacles such as columns. Moreover, we do not consider situations where there exist two or more walls with parallel normals to their surfaces between any node and the goal, such as a ‘C’ shaped area. The mathematical modeling of radio-based systems and the effect of distance on received signal strength intensity (RSSI), while not impossible, can be complex depending on the desired accuracy of the model. In practice, the nature of such systems inherently means we will always have some error. As such, for the purpose of this active beacon layout generation approach, it was deemed that there was no need to exactly model the systems explained within. Rather, by simplifying the system with certain assumptions, we can create a straight-forward, intuitive mathematical basis for the arrangement of active beacons within a navigable area for mobile robot navigation. The broad assumptions made are as follows:

1. The RSSI will be modeled using the Inverse-Square Law.
2. RSSI of a given node is only readable within a range described as the distance between said node and the goal.
3. The effects of environment on RSSI (corners, walls, metal surfaces) will be ignored.

2.2 List of variables

\( r \): References position from which a reading is being taken, i.e. the robot.
\( n_i \): References the \( i^{th} \) sensor node’s position.
\( g \): References the goal.
\( P \): Maximum expected RSSI value from any given sensor node.

\((r_x, r_y)\): The Cartesian coordinates of the where a node field reading is being taken, i.e. the position of the robot.

\((n_{ix}, n_{iy})\): The position, in Cartesian coordinates, of the \( i^{th} \) sensor node in the set of nodes.

\( \hat{s}_i \): The suggested direction from the \( i^{th} \) sensor node to the goal (unit vector).

\((g_x, g_y)\): The position, in Cartesian coordinates, of the goal.

2.3 List of equations.

2.3.1 Distance (Cartesian distance)

\[
d(a, b) = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}
\]

(1)

Here, \( a \) and \( b \) represent two positions.

2.3.2 RSSI modeling (Inverse square law)

\[
p_i = \frac{P}{4\pi d(r, n_i)^2}
\]

(2)
Here, the subscripts \( r \) and \( n_i \) denote the location from which an RSSI reading is taken and the location of the \( i^{th} \) sensor node. Essentially, if we move twice as far away from a node, the RSSI will be one fourth of its previous strength. It should be noted equations related to RSSI estimation would not be used on a physical robot, as the RSSI would be read by sensors and not simulated.

### 2.3.3 Normalization of RSSI model

\[
q_i = \begin{cases} p_i : p_i \leq P \\ P : otherwise \end{cases} \quad (3)
\]

Equation (3) “normalizes” \( p_i \) such that we can never exceed the maximum expected RSSI value \( P \).

### 2.3.4 RSSI-based weight

\[
w_i = \begin{cases} q_i : d(n_i, r) \leq d(n_i, g) \\ 0 : otherwise \end{cases} \quad (4)
\]

Equation (4) ensures that RSSI values are zero if the distance between the robot and the sensor node exceeds that of the distance between said sensor node and the goal. This is done to ensure that the field will point to the goal even when the goal is close to a sensor node. Although the robot position is not known to the real robot system we are emulating, this is possible because the sensor node and goal positions are known. From Eq. (2) we can estimate the RSSI the robot would read at the point where \( d(n_i, r) > d(n_i, g) \). Using this estimated value, we can create a cutoff value for use in the real system as needed.

### 2.3.5 Sensor node vector

\[
s_i = \left( \frac{g_x - n_{i_x}}{d(g, n_i)} \right)i + \left( \frac{g_y - n_{i_y}}{d(g, n_i)} \right)j \quad (5)
\]

The vector \( s_i \) represents the direction from the \( i^{th} \) sensor node’s location to the goal. Note that this equation is not useful if there exists a static obstacle (for this paper, we only consider walls) directly between the node and the goal.

### 2.3.6 Sensor node vector - Wall intersection distance

\[
t = \frac{wallN \cdot (wallPoint - r)}{wallN \cdot s_i} \quad (6)
\]

This equation is used in the final suggested direction calculation. It determines the distance from the robot reading location to a point on a plane along the sensor node vector. Here, \( wallN \) is the normal of a given wall as represented by a plane and \( wallPoint \) is a point on that plane (for simplicity we use the center).

### 2.3.7 Sensor vector node - Preferred wall normal orthogonal vector

\[
o = \begin{cases} wallN \times k : \cos^{-1} \left( \frac{s_i \cdot k}{||s_i|| \cdot ||k||} \right) \leq \cos^{-1} \left( \frac{s_i \times k}{||s_i|| \cdot ||k||} \right) \\ wallN \times -k : otherwise \end{cases} \quad (7)
\]

This equation is used in the final suggested direction calculation. The orthogonal vector to the wall normal that is closest to \( s_i \) is chosen. Here, orthogonality is determined by calculating the cross product between \( wallN \) and vectors in the \( k \) and \(-k \) directions. Each wall is checked for each node, and the wall with the smallest, nonnegative \( t \) is used.

### 2.3.8 Suggested direction vector

\[
S_i = \begin{cases} s_i : wallN \cdot s_i = 0 \text{ or } t < 0 \text{ or } t > d(n_i, g) \\ o : otherwise \end{cases} \quad (8)
\]

The vector \( S_i \) represents the suggested direction from the \( i^{th} \) sensor node. The first case checks if \( s_i \) is parallel to the wall, if the intersection with the wall is in the opposite direction of \( s_i \), or if the intersection is farther than the goal is from the node. In any of these cases, \( s_i \) can be used as is. If there is an intersection with a wall between a node and the goal, then the orthogonal vector to the wall normal as selected in \( o \) is used.

### 2.3.9 Field equation

\[
V = \sum_{i=1}^{n} w_i \cdot S_i \quad (9)
\]

The field is calculated as a summation of the sensor node vectors of each node weighted by their respective RSSI. Our aim is to determine the node layout that makes a vector field which guides the robot from an arbitrary position to the goal. To wit, the goal must be the only position within the navigable area where \( V = 0 \).

### 3. Conceptual Foundations

In order to begin modeling an active node layout generation system, certain underlying principles must be elucidated.

#### 3.1 Opposing node principle

**Proposition 1.** In order for a goal with nodes suggesting directions to it to be reachable by the robot, there must be at least one node opposite the read location and other nodes with respect to the goal.
Where by setting the given node as the origin of a new frame of reference and the goal as lying on the x-axis, any other node or nodes are opposite if their x coordinate is greater than that of the goal.

Proof. Validate the Opposing Node Principle. (In One Dimension)
Assumptions:

1. Signals are modeled as described in section 2.
2. The y value is fixed at 0, in order to simulate one dimensionality.
3. All nodes are equidistant from the goal.

Conclusions | Justifications
---|---
1. The gradient of field \( V \) will trend towards \((g_x, 0)\) when \( V = 0 \) at \((g_x, 0)\) | From Eq. (9).
2. (One node case.) When \( i = 1 \) and \( n_{1x} > g_x, V < 0 \) | From Eq. (5).
3. (One node case.) When \( i = 1 \) and \( n_{1x} < g_x, V > 0 \) | From Eq. (5).
4. (Two node case.) When \( i = 2, n_{1x} > g_x, \) and \( n_{2x} < g_x, V = 0 \) at \((g_x, 0)\) when \( d(n_{1x}, g) = d(n_{2x}, g) \) | From Eq. (9).
5. There must be at least one node with \( x \) greater than \( g_x \) and one node with \( x \) less than \( g_x \) | From conclusion 4.

In order to easily visualize this principle, let us initially limit the robot’s movement to one dimension, i.e. along a line. In Fig. 1, we see a line with a goal at \( g = 0 \), in the center. Also on the line exists a single node at \( n_1 = -1 \), depicted on the far left. Using Eq. (5), making it one-dimensional by setting the y value to 0, we see that \( s_1 = 1 \). If, for the time being, we use \( q \) regardless of the distance from the node to the goal, we can see that anywhere on the line, \( V \) will be positive. In other words, the robot will only reach the goal if it begins to the left of the goal, as depicted by the dotted-line bounding box (viable area). Let us next consider Fig. 2, where in addition to the goal, \( g \), and sensor node \( n_1 \) is a second node at \( n_2 = 1 \). Again ignoring the limit from Eq. (4), we find that now \( V \) will be positive to the left of the goal, and negative to the right (\( s_2 = -1 \)), meaning the robot can reach the goal, regardless of its initial position.

3.2 Effective range limit principle

**Proposition 2.** The RSSI of a sensor node, \( n_i \), must be ignored at distances further than the distance from \( n_i \) to the goal in order for it to be reachable from a direction opposite \( n_i \) with respect to the goal.

Proof. Validate the Effective Range Limit Principle
Assumptions:

1. Signals are modeled as described in section 2.
2. The y value is fixed at 0, in order to simulate one dimensionality.

Conclusions | Justifications
---|---
1. The gradient of field \( V \) will trend towards \((g_x, 0)\) when \( V = 0 \) at \((g_x, 0)\) | From Eq. (9).
2. There must be at least one node with \( x \) greater than \( g_x \) and one node with \( x \) less than \( g_x \) | Opposing Node Principle.
3. When \( i = 2, n_{1x} > g_x, \) and \( n_{2x} < g_x, V \neq 0 \) at \((g_x, 0)\) when \( d(n_{1x}, g) \neq d(n_{2x}, g) \) | From Eq. (9), when \( p \) is used instead of \( w \).
4. When \( i = 2, n_{1x} > g_x, \) and \( n_{2x} < g_x, V = 0 \) at \((g_x, 0)\) when \( d(n_{1x}, g) = d(n_{2x}, g) \) and \( d(n_i, g) \neq d(n_{2x}, g) \) | From Eq. (9).
5. The RSSI weight \( w \) of a node \( n_i \) when estimated for a location \( r \) must be ignored, \( w_i = 0 \), when \( d(r, g) > d(n_i, g) \) | From conclusion 4.

In the previous subsection, we ignored the limit from Eq. (4) on \( g \). This works in Fig. 2 because the goal is equidistant to both sensor nodes. This causes the polarity of \( V \) to change at the goal, where the RSSI from both nodes is the same, and the goal is reachable.
from either side. If however, as in Fig. 3, the goal was closer to \( n_2 \) at \( g_x = 0.5 \), this would not be the case. In this case, \( V \) is still zero at the origin, because only at 0 is the RSSI from each node equal. As such, the goal is only reachable if the robot begins to the right of the goal, as depicted by the dotted-line bounding box (viable area). In order to counteract this problem, the RSSI of a node must be limited to readable only if it is closer to the node than said node is to the goal. With this limitation, in the case of Fig. 3, the polarity of \( V \) will now reverse at the goal, regardless of the distance between the nodes and the goal.

3.3 Closed polygon principle

**Corollary 1.** Following the Opposing Node Principle, it can be expanded that if a goal must be between nodes, a node layout can be said to be fully navigable (i.e. the robot can navigate from any point to any other point) if and only if the goal exists within a closed polygonal chain in which the nodes form the vertices.

We have seen in the Opposing Node Principle that the goal must be between two nodes in one dimension. This extends to two dimensions. Again, let us examine this graphically. In Fig. 4 we see a two dimensional (Euclidean) space. The goal is at the center, and there exists a single node, somewhere in the third quadrant. Since a reading taken by the robot anywhere in the space will point up and to the right, the robot must start in the dotted-line bounding box (viable area), reminiscent of the one dimensional case. If we add another node, as in Fig. 5, we can see the viable area is expanded to cover the areas between the nodes and goal and the area between the two nodes. We can see that again, nodes not within the viable area will simply move further and further away. If we add another node, as in Fig. 6, the pattern becomes clear. The viable area always contains a closed polygon where the nodes are vertexes of said polygon. In Fig. 7, we see that four nodes allows us to navigate from anywhere, as long as we start within range of a node. The navigable areas depicted in this section are estimated based on the methodical model, that is to say, they are dependent on the navigational algorithm the system uses. A real robotic system using these layouts may have an area slightly larger than the ones depicted here. However, as we hope develop navigational algorithm independent layouts, we based these calculations and our mathematical model on a system in which not only signal coverage, but layout was a critical issue, such as in [8].

4. Node Layout Generation

Based on the principles described in the previous section, we developed three generic rules for the
placement of nodes in an environment without free-standing obstacles such as columns. These rules were then used to develop a software presented in [16] to automatically generate layouts based on CAD-like map data. Using the details of wall locations, the software proceeded to apply the following rules, and finally, output the locations (essentially Cartesian coordinates) of where each node should be placed.

4.1 Process 1: Corners

**Procedure 1.** At each corner, place a node.

As we have seen in section 1, it is necessary to have our desired navigable area enclosed within a closed polygonal chain of nodes. Placing a node at each corner as the initial step ensures we satisfy this constraint. Logically, it is placed at the exact corner where the walls meet, but in our approach, it is offset slightly to account for the node’s physical size in reality. The direction of this offset is simply the bisection of the internal angle formed by the two adjoining walls.

4.2 Process 2: Convex corners

**Procedure 2.** At each convex corner, remove the existing node and place two nodes slightly further along the two walls.

In general it is necessary to replace a single node placed at a convex corner in process 1 with two nodes, spaced slightly further away from the apex of the corner. This is due to the fact that a node placed at the apex will have line-of-sight to goals closer to the adjacent wall. This will cause a robot approaching from the opposite wall to receive a turn suggestion before it has cleared the wall. By placing two nodes, we can have each node handle its respective adjacent wall, ensuring the robot will clear the first wall before turning towards the goal. This step can be skipped if the system determines that the distance of the convex corner’s protrusion is smaller than some prescribed distance. For example, in a room that is 8mx8m where the acceptable accuracy is a 1m square, a small protrusion of 30cm would be deemed too small. This is adjustable, but in general anything smaller than the accuracy of the active beacon system can be ignored.

4.3 Process 3: Straights

**Procedure 3.** Place a node halfway between nodes along the same wall if their distance exceeds some prescribed limit or if a collision with said wall is probable between them.

This final, repeatable process is meant to correct any problems that may persist after applying the first two processes. In general, it fixes two problems. The first arises when two nodes are too far apart. In
In this case, there could be an area midway between them that would not receive sufficient RSSI to prevent the robot from colliding with the wall. The second problem arises when a node from an opposing direction has sufficient influence so as to “push” the robot into the wall. The end result is quite similar, so this single process handles both cases. This process is repeated until the emulated navigation no longer predicts collisions.

4.4 Emulated navigation

After each process is applied, and after each iteration of process 3 being applied, the layout is evaluated to determine if there remain any areas where collisions are likely to occur. For this, we developed an emulated navigation approach based on the system used in [8] and tested it from points on a grid over the navigable area of the map, i.e. the areas not occluded behind walls or outside the enclosed room. The resolution of this grid is adjustable and we set it depending on the required accuracy of the robot navigation. Using these grid points as start and goal points, we emulate navigation mathematically from each point as a start to every other point as a goal.

The emulated navigation is performed as follows. First, the local suggested direction is calculated for each grid point, using the equations presented in section 2. These suggested directions, a 3-dimensional vector, are then normalized to fit their nearest ordinal direction: up/down, left/right, and diagonals. Then, beginning with the “start” point, the suggested direction at each point is followed to the subsequent grid point. In this way, the simulated path follows the virtual potential field toward the goal. Due to the nature of the equations in section 2 and the “normalization” performed, each grid point will always point either toward another grid point or towards a wall. In the event the suggested direction leads to a wall, the collision is stored in a set. These collisions are then categorized as being either the result of process 2 not having been applied or the distance between two nodes is too great. This is done by first reordering the collision point set sequentially so that collisions points are in order when following the wall in an arbitrary direction, for example, clockwise. Next, this collision point set is split into subsets of collisions that are logically grouped. In other words, collision points that are close to each other are thought to be caused by the same problem. This distance is also adjustable but is usually twice the distance between grid points. Once these subsets are created, their endpoints are compared. If their collision points do not occur along a straight line, this indicates process 2 needs to be applied. If they are in a straight line, process 3 should be applied. The overall algorithm for generating layouts can be seen in Fig. 8.
In Fig. 9 and 13 we can see two example layouts that are sufficiently cleared with only the applications of process 1. The beacon nodes are depicted as the small circles with an arrow denoting their suggested direction and the goal is the wire-frame circle in the center. The field here is not displayed as normalized to ordinal directions simply to emphasize the nature of the field. However, in Fig. 17, (with the field normalized to their nearest ordinal) we see an ‘L’ shaped room, to which only process 1 has been applied. In the area marked by the dotted line, we can see the several vectors point directly toward the wall, which would most likely result in crash if the goal was toward the right as shown and the robot began its navigation in the bottom area. In Fig. 18 we can see the same map after processes 2 and 3 have been applied. The vectors now correctly point around the corner, due to the extra influence of the two nodes there.

Using the layouts generated, we tested their validity using the navigation algorithm and robot described in [8], the same upon which the mathematical model was based. In Fig. 10 we can see the paths taken on two exemplar runs taken by the robot using the layout in Fig. 9. The circular marks indicate the path, Fig. 11, when the robot started in the opposite corner from the goal, the gray area in the upper-left corner. The diamond marks indicate the path, Fig. 12, when the robot started in the adjacent corner, with an offset to allow for the robot’s rotation.

In Fig. 14 we see the paths of similar runs, Fig. 15 and 16, for the layout shown in Fig. 13. The start point in Fig. 16 is again offset to account for an initial rotation of the robot. Finally, in Fig. 19, we the paths of runs, Fig. 20 and 21, for the the final layout shown in Fig. 18.

Fig. 8 Active beacon layout generation algorithm

Fig. 9 A simple, four-cornered room requiring only a node in each corner (blue points)

Fig. 10 Exemplar paths of a robot run through the four-cornered room from two starting points
Fig. 11  Video captures of exemplar run through the four-cornered room from opposite corner

Fig. 12  Video captures of exemplar run through the four-cornered room from an adjacent corner

Fig. 13  A six cornered room, now requiring that we place six nodes

Fig. 14  Exemplar paths of a robot run through the six-cornered room from two starting points

Fig. 15  Video captures of exemplar run through the six-cornered room from opposite corner

Fig. 16  Video captures of exemplar run through the six-cornered room from an adjacent corner
4.5 Emulated navigation validation

The key concern of the emulated navigation is the validity of the point-grid evaluation method. In our case, the grid size is set such that the distance between points in the represented room is
approximately 0.5m, horizontally and vertically. This 1m × 1m box is the desired accuracy of a robot using a WSN-based navigation system, such as presented in [8]. Therefore, if the emulation can arrive to at most one grid-point away from the “goal”, then the navigation is deemed successful. By performing the aforementioned calculations at each grid-point, we can assume the navigation is effectively emulated from said points. If the navigation is then proved for the given grid points, we can assume that between two successful points, any value between them is also valid. The Intermediate Value Theorem states that if we have a function \( f(x, y) \) that is continuous over the intervals \( x = [a_x, b_x] \) and \( y = [a_y, b_y] \), where \( f(a_x, a_y) = M \) and \( f(b_x, b_y) = N \) \( (M < N) \), there exists some value pair \( (c_x, c_y) \), when \( a_x < c_x < b_x \) and \( a_y < c_y < b_y \), that will produce a value \( I \), \( f(c_x, c_y) = I \), such that \( M < I < N \). We can see that if a function is continuous and monotonic over a given range, we can infer that if a solution exists at the end points of a range, then there exists a solution at any point within that range. In other words, if we can produce a suggested direction at two grid points, we can produce a suggested direction at any point between them. Looking at the field equations presented in section 3, we can see that they are continuous over the range \( \{(1, -1), (1, 1)\} \), therefore, we can also assume that a field vector can be calculated over that range. Moreover, we can see that the equations in section 3 (of particular note Eq. (2)) are at most quadratic, and so can assume for sufficiently small ranges from \( a \) to \( b \), the field will display monotonicity. We must also consider that for very long ranges from \( a \) to \( b \), there is a possibility for a possible collision to be missed. We limit the distance between \( a \) and \( b \) to no more than half the desired accuracy of the system, also providing us with the practical ability to check if the robot has reached the goal. As such, we can assume that locations between two grid points will produce direction vectors between said grid points, and that any evaluation successfully performed from grid points within that range will also be possible at points between said grid points.

4.6 Discussion

As has been discussed, a set of processes or rules for the placement of infrastructure nodes in a mobile robot navigation system would facilitate not only easier installation, but also the development of a standard which all similar systems could be tested with. The three processes presented here form the basis for such a methodical approach. The processes were developed with three fundamental concepts based on the nature of a virtual potential field approach to indoor mobile robot navigation. Arguably, the processes are strongly coupled with that style of navigation. However, it was deemed that a virtual potential field algorithm required a specific placement of nodes in relation to the navigable area, in contrast to algorithms such as triangulation, in which the consistent reception of a minimum number of incoming signals is the key concern. In such a system, it could conceivably be sufficient to estimate broadcast ranges and ensure that all points in the field are covered. The current set of processes produce layouts that are consistent with layouts successfully used in previous studies [8,13], and were physically validated. It is clear that the key limitation of the current set of processes is the inability to handle free-standing obstacles. However, we believe with an extension on these three base processes or a partitioning of the map and alternate analysis/labeling method, it will be possible.

5. Conclusion and Future Works

As the need for permanent installations of robots in non-industrial applications increases, it is possible that many types of robots will need to share the same support infrastructure, such as active nodes for navigation. Given that it is also possible these various robots will utilize different navigation algorithms, it becomes efficacious to standardize the placement of such nodes, so that those algorithms can expect a certain layout. As a first step towards that goal, we have proposed a methodical approach to the placement of active nodes to ensure the successful navigation of a mobile robot. By first laying out underlying principles to the nature of virtual potential field navigation schemes, we were able to develop processes by which to analyze map data about the navigable area and then using a model of the node behavior combined with an emulated navigation ensure a successful navigation.

The main focus for future work on this research is the extension of the current approach to handle curved wall surfaces, multi-room environments, and free-standing, static obstacles. Future works also include the modeling and verifications of other types of active beacon systems, as well considering the incorporation of passive beacon systems (such as active and passive RFID, respectively). Finally, we hope to extend the system to navigable areas with both indoor and outdoor components, something we believe feasible due to an expected improvement in RSSI reliability when using active beacons outdoors.
and said points. If the navigation is then proved for the aforementioned calculations at each grid-point, we can conclude that navigation is deemed successful. By performing the navigation style of navigation. However, it was deemed that there could conceivably be sufficient to estimate broadcast placement of active nodes to ensure the successful navigation of mobile robots in an environment. Therefore, if the emulation can arrive to at a WSN-based navigation system, such as presented in [8], and its accuracy of the system, also providing us with the freedom to control the placement of active nodes to ensure the successful navigation of mobile robots in an environment.

Arguably, the processes are strongly coupled with that underlying principles to the nature of virtual potential field approach to indoor mobile robot navigation. The main focus for future work on this research is the extension of the current approach to handle free-standing, static obstacles. Future works also includes the development of active beacon systems, as well considering the placement of active nodes to ensure the successful navigation.

As the need for permanent installations of robots increases, it is possible to no more than half the desired accuracy of the system, also providing us with the freedom to control the placement of active nodes to ensure the successful navigation of mobile robots in an environment. In non-industrial applications increases, it is possible to no more than half the desired accuracy of the system, also providing us with the freedom to control the placement of active nodes to ensure the successful navigation of mobile robots in an environment.

5. Conclusion and Future Works

The main focus for future work on this research is the extension of the current approach to handle free-standing, static obstacles. Future works also includes the development of active beacon systems, as well considering the placement of active nodes to ensure the successful navigation.

As the need for permanent installations of robots increases, it is possible to no more than half the desired accuracy of the system, also providing us with the freedom to control the placement of active nodes to ensure the successful navigation of mobile robots in an environment. In non-industrial applications increases, it is possible to no more than half the desired accuracy of the system, also providing us with the freedom to control the placement of active nodes to ensure the successful navigation of mobile robots in an environment.

6. References

[1] K. Brieche and U.D. Hanebeck: Localization of a mobile robot using relative bearing measurements, IEEE Transactions on Robotics and Automation, Vol. 20, Issue 1, pp. 36 - 44, 2004.
[2] J.S. Esteves, A. Carvalho and C. Couto: Position and orientation errors in mobile robot absolute self-localization using an improved version of the generalized geometric triangulation algorithm, IEEE International Conference on Industrial Technology (ICIT) 2006, pp. 830 - 835, 2006.
[3] E. Nakamori, D. Tsukada, M. Fujimoto, Y. Oda, T. Wada, H. Okada and K. Matsuura: A new indoor position estimation method of RFID tags for continuous moving navigation systems, International Conference on Indoor Positioning and Indoor Navigation (IPIN) 2012, pp. 1 - 8, 2012.
[4] S. Anwar, Z. Qingjie, N. Qadeer and S.I. Khan: A framework for RF-Visual SLAM, 10th International Bhurban Conference on Applied Sciences and Technology (IBCAST) 2013, pp. 103-108, 2013.
[5] S. Se, D. Lowe and I. Link: Vision-based mobile robot localization and mapping using scale-invariant features, IEEE Intl. Conf. on Robotics and Automation 2001, pp. 2051-2058, 2001.
[6] M.S. Miah and W. Guoaiib: Indoor robot navigation through intelligent processing of RFID signal measurements, International Conference on Autonomous and Intelligent Systems (AIS) 2010, pp. 1 - 6, 2010.
[7] L. Catarinucci, S. Tedesco and L. Tarricone: Customized ultra high frequency radio frequency identification tags and reader antennas enabling reliable mobile robot navigation, Sensors Journal IEEE, Vol. 13, Issue 2, pp. 783 - 791, 2013.
[8] G. Enriquez, S. Park and S. Hashimoto: Wireless sensor network and RFID fusion approach for mobile robot navigation, ISRN Sensor Networks, Vol. 2013, Article ID 157409, doi:10.1155/2013/157409, 2013.
[9] Y. Koren and J. Borenstein: Potential field methods and their inherent limitations for mobile robot navigation, IEEE International Conference on Robotics and Automation 1991, pp. 1398 - 1404, 1991.
[10] S.S. Ge and Y.J. Cui: Dynamic motion planning for mobile robots using potential field method, Autonomous Robots, Vol. 13, Issue 3, pp. 207 - 222, 2002.
[11] G. Enriquez, S. Park and S. Hashimoto: Human-aware guidance robot using wireless sensor network, RFID, and Vision, IEEE Mecatronics 2010, DVD Proceedings, 2010.
[12] J. Zhou, J. Shi and X. Qu: Landmark placement for wireless localization in rectangular-shaped industrial facilities, IEEE Transactions on Vehicular Technology, Vol. 59, Issue 6, pp. 3081 - 3090, 2010.
[13] G. Enriquez, S. Park and S. Hashimoto: Wireless sensor network and RFID sensor fusion for mobile robots navigation, IEEE International Conference on Robotics and Biomimetics (ROBIO) 2010, pp. 1752-1756, 2010.
[14] Z. Chen and X. Li: A RFID-based dual-command method for unit-load warehouse systems, IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2011, pp. 1026 - 1030, 2011.
[15] C. Mansley, J. Connell, C. Isci, J. Lenchner, J.O. Kephart, S. McIntosh and M. Schappert: Robotic mapping and monitoring of data centers, IEEE International Conference on Robotics and Automation (ICRA) 2011, pp. 5905 - 5910, 2011.
[16] G. Enriquez, S. Park and S. Hashimoto: Node layout generation for mobile robot navigation, 30th annual conference of the Robotics Society of Japan, DVD-Proc. RSJ2012AC1B2-4, 2012.

Guillermo Enriquez received the B.S. degree in computer science from the University of Texas at El Paso in 2005 and the M.S. degree in applied physics from Waseda University, Japan, in 2008 under a Japanese government scholarship. He is a Research Associate at Waseda University while continuing his doctoral coursework in pure and applied physics at Waseda University, Japan. His research interests include mobile robots, navigation, wireless sensor networks, and domotics.

Sunghong Park received the B.S. in electronic physics from Daejeon University, Daejeon, Korea, in 2004, and the M.S. and Ph.D. degrees in applied physics from Waseda University, Tokyo, Japan, in 2007 and 2010, respectively. He is currently a researcher at the Smart Vehicle Technology Research Center, Korea Automotive Technology Institute (KATECH). His research interests include mobile robotics, smart vehicles, navigation, and radio frequency identification technology.

Shuji Hashimoto received the B.S., M.S. and Ph.D. degrees in applied physics from Waseda University, Tokyo, in 1970, 1972 and 1977, respectively. He is currently Vice President of Waseda University in charge of Academic Affairs, Research Promotion, Student Affairs, and Cultural Programs as well as a Professor in the Department of Applied Physics, Waseda University. His research interests include neural networks, image processing, humanoid robots, and kansei information processing. He is a member of the IEEE, the Robotics Society of Japan, and the Research Institute of Signal Processing, Japan.

(Received October 4, 2013; revised January 20, 2014)