The AAAI-17 Workshop on
Human-Aware Artificial Intelligence
WS-17-10

Complete Coverage Planning: Achieving Human Interaction and Maximum Coverage During an Autonomous Robotic Vehicle Navigation of an Unknown Terrain

Balasubramaniyan Chandrasekaran, James M. Conrad
University of North Carolina at Charlotte
Department of Electrical and Computer Engineering
{bchandr3, jmconrad}@uncc.edu

Abstract

In this paper, the concept of complete coverage planning is described through a robotic vehicle navigation on an unknown terrain. The navigation application is highly interactive with the user to achieve maximum coverage of the field and gather terrain information for the user and is called as complete coverage planning. Sensor Maps are dynamic structures similar to the sensory brain maps which process the sensor information from the vehicle and re-arrange themselves based on the usage of the sensors during the navigation application's lifetime. The main idea behind such an approach is to formulate a human-inspired model for cognition. Human intervention is facilitated when the vehicle is uncertain about a decision during its traversal in the field.

Introduction

Autonomous vehicle navigation is increasingly used in several applications, such as elderly care, schools, space exploration and several other faculties of everyday lives. In (Chandrasekaran and Conrad 2015), the authors summarize the integration of human interaction and robot operations through a survey. Several applications ranging from elderly care to space exploration are highlighted and the level of human machine collaboration is demonstrated through examples of robots used in each of the areas. The authors in (Ghangrekar and Conrad 2009) introduce the concept of complete coverage planning of vehicle navigation on an unknown terrain.

Theory and Background

In (Choset 2001 ; Galceran and Carreras 2013 ; Ghangrekar and Conrad 2009), the authors summarize a few planning algorithms for the vehicle. Also, the vehicle maneuvers over the obstacle(s) by drawing a collision avoidance boundary and navigates along local paths computed while encountering obstacles and resumes navigation along the regular path (or normal path) to its destination once it is dealt with the obstacle (Ghangrekar and Conrad 2009). To achieve human interaction it is necessary to have some form of dialogue between the operator and the vehicle. Command numbers are used to achieve bidirectional communication during the assist phase when the vehicle seeks help. This is similar to the collaborative control mentioned in (Fong and Thorpe 2013).

In (Ramachandran 2011; Doidge 2015), the authors describe the cognitive model of the brain through sensory brain maps. These maps (Penfield maps) hold a slot for every sensory input in the human body and are arranged as shown in Figure 1 (Ramachandran 2011).

When a sensory input is not recognized in the brain map for a certain time, the map dynamically alters itself to accommodate more space and resources for those sensor slots that are adjacent to the one that is marked as dormant (Ramachandran 2011). This results in a few sensory areas on the map becoming larger, growing into the space that was utilized by the dormant sensor while the other sensor areas show no change in their space (those not adjacent to the dormant sensor). However, when the dormant sensor becomes active again, through the respective task (learning and training), the map again starts to alter, re-assigning its original space in the brain map, showing an increase in space at that slot. This is known as differentiation of the brain map (Doidge 2015).

These principles on which the brain operates and micromanages the sensory map is applied in this work, and is demonstrated through the use of three functions namely: Sensor Space Reduce, Sensor Space Reallocate Function and Sensor Space Deallocate Function. A selective sensor framework is introduced in (Chandrasekaran and Conrad

2016) that provides a generalized view of integrating robotic applications with vehicle cognition and embodies human interaction is the process.

Figure 1. A portion of the Penfield map of the skin surface. (Ramachandran 2011)

Experiment Description

We formalize the concept of Sensory Maps in robotic vehicle navigation through an application similar to the one highlighted in (Ghangrekar and Conrad 2009), however, with the inclusion of human interaction during uncertainty and increased coverage by visiting obstacle free cells that were not considered previously in the paper. The experiment is divided into two phases, namely, setting up the navigation application of the vehicle and using this application to demonstrate the dynamicity of the sensor map. The phase I, namely, robotic vehicle navigation is described in the following section. The phase II, sensor map dynamicity is described in the section thereafter.

Phase I – Complete Coverage Planning

We consider a grid of size 5 by 9. The robotic vehicle is non-holonomic (i.e. can make only 90 degree turns and cannot move in all directions). The vehicle traverses the field column-by-column covering as many cell blocks as possible. The borders of the field are considered reachable so that the vehicle can navigate the perimeter of the field. The vehicle makes 90 degree turns at the top and bottom cells of the field either turning right at the top of the column or turning left at the bottom of the field, to move to the neighboring column.

Two types of obstacles are considered, namely, trees and ravines. Complete coverage planning refers to the name of the application that runs on the vehicle and is not associated with any path planning algorithm. The name is given as such to indicate maximum area will be covered by the vehicle. This application does not use any path planning technique.

Implementation and Vehicle navigation

The experiment was implemented using Python language (Version 2.7) and the Turtle Graphics tool (in-built utility available in Python 2.7)

The typical layout for the simulation is shown as in Figure 2. The vehicle traverses the field, cell-by-cell, from top to bottom or bottom to top along each column. It makes left turns when it reaches the bottom of column (i.e., highest index value for row) and makes a right turn when it reaches the top of the column (i.e., row index 0). Trees are drawn as green circles and ravines are represented as thick black lines. The robotic vehicle is represented as a turtle. The trace of the vehicle (or the path taken by the vehicle) is shown in red.

The Figures 3 – 7 show the vehicle traversing the field and initiating the request for help during the help phase. Human Machine Interaction (HMI), is then achieved through the communication of help numbers between the robot and the user.

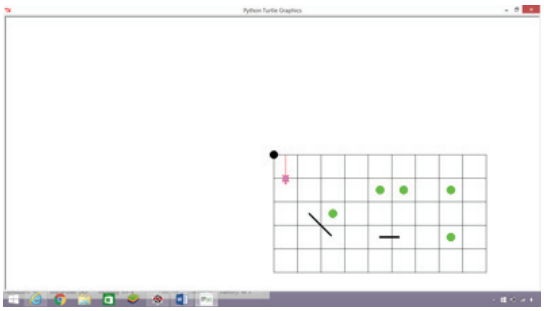

Figure 2. Simulation field for the vehicle.

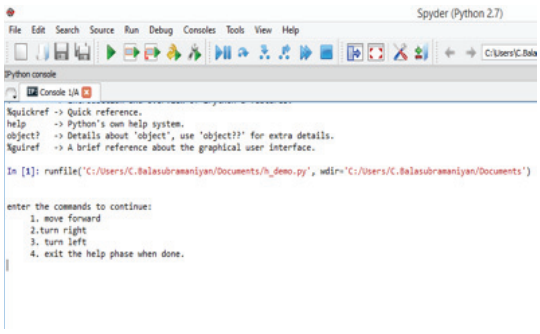

Figure 3. Help phase initiated and user chooses commands.

The vehicle seeks help from the user only when it is unable to steer around the obstacle. The user provides one of four options: enter 1 to move forward, enter 2 to turn right, enter 3 to turn left or enter 4 to exit the help phase. Only a minimal set of commands are exchanged between both entities before the robot switches back to the autonomous mode.

In the Figures 8-10, the vehicle navigates smoothly in the field by handling the obstacles by itself, even when the ravine and tree are close (Figure 9) or when the obstacles are spaced apart. The sensors used could be ultrasonic sensors or LIDAR.

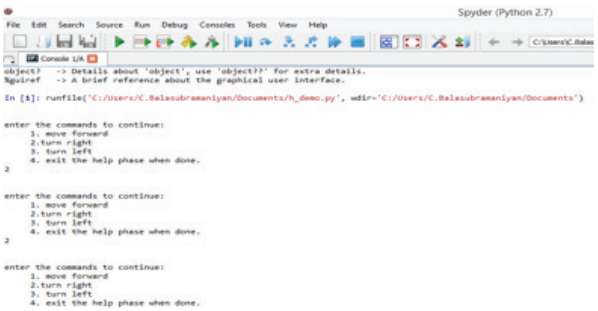

Figure 4. User enters turn right twice to avoid the obstacle.

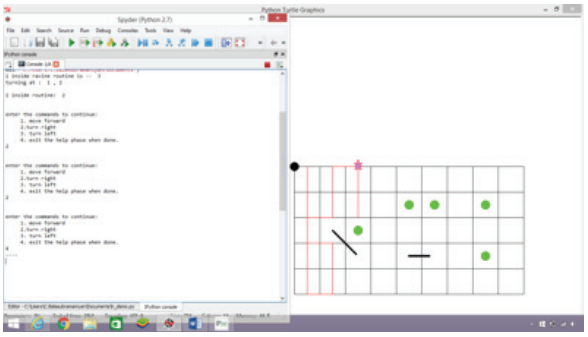

Figure 5. Vehicle swings back to autonomous mode.

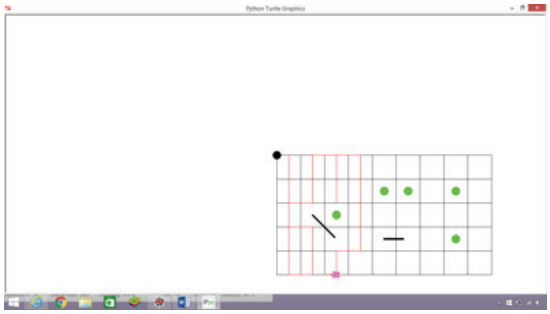

Figure 6. Vehicle covers the cells not previously visited.

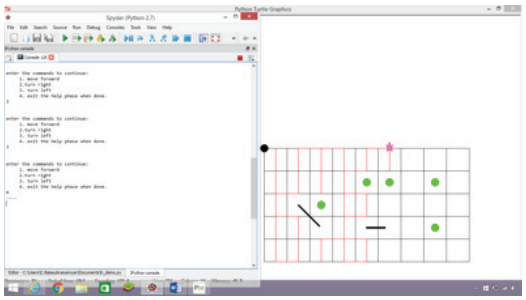

Figure 7. Autonomous navigation after seeking user's help.

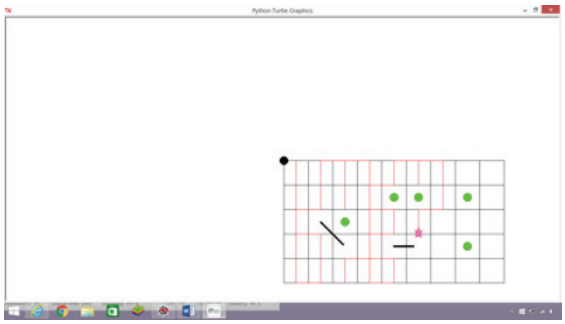

Figure 8. Vehicle handles ravine and tree autonomously.

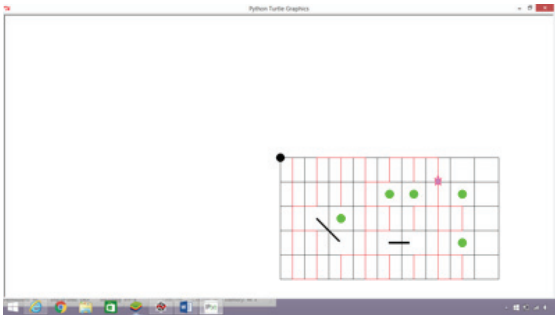

Figure 9. Vehicle avoiding the trees.

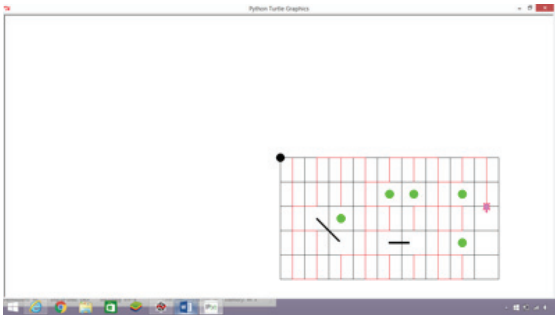

Figure 10. Vehicle follows regular path until it reaches its destination at the bottom.

Phase II – Sensor Map Dynamicity

In Phase I, we were concerned only about the application layer of the robotic system. The application is programmed into the robot and the focus was to achieve human interaction during the execution of the main task, thereby keeping the internal cognitive processing hidden from the user experience. In this section, we demonstrate the use of a robot sensor map similar to the sensory brain map (Ramachandran 2011) to perform information processing and resource management for sensors on the robot. The main idea behind such an approach is to formulate a human-inspired model for cognition. The sensor map implemented works in a very similar manner as the sensory brain map, performing operations such as sensor slot allocation, allocation of un-used sensor space to the other sensors and re-allocation of sensor space back to the sensors which had the original ownership. Such a dynamic structure allocates more space for storage and processing data from the sensors that are heavily used, resulting in efficient usage of the memory area. This is similar to how more area in the brain map is occupied by the most frequently used sensory inputs.

The sensors are assumed to be laid in a sequential manner in the sensor space. Depending on the initial conditions, the sensors could be originally assigned equally spaced memory slots. The dynamicity of the map is tested with the help of the application described in the previous section. The sensor manager holds all the relevant information related to each sensor and has exclusive ownership of the sensor map space. It keeps track of which sensors are used for the application and periodically (or based on the user given requirements) can run different functions that manage and modify the space for the sensors. Similar to the use-it-or-lose-it principle as outlined and described in (Ramachandran 2011; Doidge 2015), the sensory brain map is a dynamic structure that keeps changing depending on the usage of the sensory organs.

Machine Learning and Sensor Map Dynamicity

In order to illustrate the dynamic nature of the sensor maps, it is necessary to use a learning technique. Since we as the user can decide the criteria for memory usage and thresholds for the sensors, supervised classifiers are the best fit for this purpose. When we use such classifiers we can extend the current system to handle newer features and train the system accordingly. Each of the sensor space management functions described below actively use a supervised classifier called the Random Forest (Breiman 2001). The user can choose the label for the data sets and define them during the training phase. However, for the present application the data sets are small, hence the classifier is very reliable and highly accurate in its prediction on the test set. The description of each of the sensor management functions is described in the following paragraphs.

In Sensor Space Reduce Function, we identify the space that should be retained for each sensory input. This function keeps track of the usage of the sensors on the vehicle and determines how much space should be held in the map for each sensor. When a sensory input keeps reducing as the application progresses, or between different applications, the corresponding space in the map starts to dynamically reduce as a function of its usage. The classifier uses the frequency of the sensor's usage to determine the percentage by which the space should be reduced. This phenomenon is akin to how the brain reduces the space on its sensory map when a particular sensory input is less and less received on the map during one's lifetime (Ramachandran 2011), also known as shrinkage of sensory space on the map.

The Sensor Space Reallocate Function distributes the sensor space of a dormant sensor to its adjacent sensory members on the map. This function checks for sensors that are marked for deletion and identifies its adjacent members on the map. The classifier distributes the space to the dormant sensor's neighbors and once distributed notifies the sensor manager of these changes. As a result, the redistribution causes some areas on the map to grow larger, however, when the sensors are at the borders, the entire distribution is for the one sensor that is adjacent to it.

The Sensor Space Deallocate Function de-allocates the space from the sensors that had previously taken space from the dormant sensor and re-allocates it back to the dormant sensor when it becomes active again. The function uses a classifier that identifies the sensors that become active again and automatically trains those sensors (this is done internally inside the function by running the tasks mapped to it) to increase its usage. Again, the usage metric causes the classifier to increase the space on the map for the trained sensor while reducing the extra space from its neighbors..

Sensor Maps in Robotic Vehicle Navigation

The application is implemented in Python 2.7 and uses libraries from sklearn documentation (Pedregosa et al. 2011) to make use of some machine learning techniques (Patnaik 2007 ; Mitchell 1997) for supervised classification (Mitchell 2007). The robot's sensory map is pre-filled with the memory allocation and sensor usage. Table 1 shows the mapping between the sensors and tasks and the initial state of the map.

When a sensor has 100% space, it refers to the accessible area for that sensor (on the sensor map) but as the sensor management functions are executed this accessible region for the sensor will start to diminish and could eventually result in 0% space which will be described in the

following sections. The navigation application is executed and the vehicle updates the usage of the tasks and the sensors during this process. Once the field is navigated, the sensor manager runs the Sensor Space Reduce function. As a result, the space for each sensor starts to reduce depending on how much it was used during its previous application. The classifier present inside this function does not reduce the sensor space if its usage is above a certain threshold (user-defined), otherwise it reduces the sensor space according to its usage. Figure 11 shows the space available for each sensor after the application was executed. From the figure, we can see that as the usage is below a certain threshold the classifier marks the amount of space to be reduced for each sensor. The blue and orange bars indicate the available space before and after the navigation application. The user can specify the criteria for reduction depending on how often the task is being used. When a sensor space is reduced, it refers to the active working area only. For instance in Figure 11, we can see that space for sensor 2 was reduced by 20%, however this 20% is still under the ownership of sensor 2 (until it reaches 0).

Table 1. Initial state of the sensor map

Sensor ID and task mapping	Space available (working area in %)	Usage count
0 (move forward)	100	50
1 (move backward)	100	70
2 (turn right)	100	50
3 (turn left)	100	50
4 (detect person)	100	55

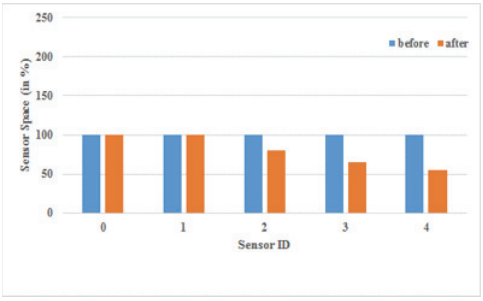

Figure 11. Sensor space distribution before and after the navigation application.

Case 1: Dynamic allocation from a border sensor slot
The navigation application is continuously executed by disabling the check for the person task, thereby reducing the space for sensor 4 (a border sensor slot) to 0. After the space reduction phase, the Sensor Manager calls the Sensor Re-allocate routine to identify the sensors with space reduced to 0. This routine then de-allocates the space from this sensor (sensor with space equal to 0) to its adjacent sensors. This causes some of the sensor spaces to be larger than others. In this experiment, we execute this function only when the space for a sensor has dropped to 0, indicating with certainty, that the sensor wasn't used in the previous application (its usage count is reduced to 0).

Figure 12 shows the map distribution after this space reallocation. It also shows the space distribution from the passive sensor to its adjacent member sensors. Since there is only sensor 3 as a neighbor to sensor 4, all of sensor 4's space is allocated to sensor 3, causing it to grow larger, which is evident from the figure.

The navigation application is executed again and the task mapped to sensor 4 is re-enabled (check for person). When this task is run, it wants to use sensor 4 for its task (detect person). Since the map had de-allocated sensor 4's space (since it was unused in the previous applications) to sensor 3, it needs to reallocate that space back to sensor 4.

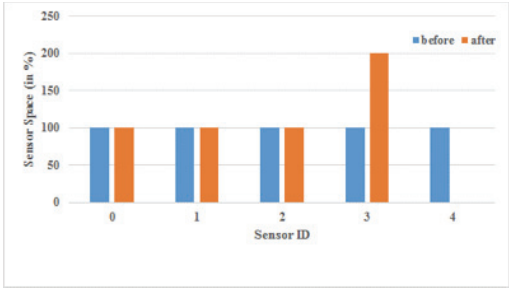

Figure 12. Space distribution after sensor re-allocation showing sensor 3 which gets extra space from sensor 4.

When the application identifies that sensor 4 has no space it calls another internal function, Handle Feasibility. This function identifies the sensor with the ID specified in the argument list and starts training the vehicle internally to increase its usage, thereby allocating space back into its area and simultaneously de-allocating the space from its adjacencies. The feasibility function uses the Sensor Deallocate function to achieve this task. As a result, the sensor space is reduced for just one sensor as shown in Figure 13.

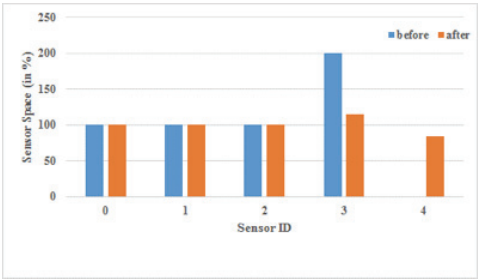

Figure 13. Space distribution after navigation application. Sensor 4 becomes active and regains the space it had sacrificed to sensor 3 previously due its dormancy.

Case 2: Dynamic allocation from a middle sensor slot
The mapping is now rearranged by changing the slots for sensor 4 and sensor 3 (hence the respective tasks). For the purpose of demonstration in the application task associated with sensor 3 is disabled and that results in sensor 3 becoming unused and the usage dropping to 0 and hence its space reduced to 0. On executing the Sensor Space Reallocate function, we get the space distribution as highlighted in Table 2. Since sensors 2 and 4 are adjacent to sensor 3 its space is distributed to 2 and 4 causing their areas to grow larger which is evident from the Figure 14.

Table 2: Sensor map after Sensor re-allocation

Sensor ID	Space available (working area in %)	Usage count
0	100	126
1	100	146
2	135	82
3	0	0
4	155	59

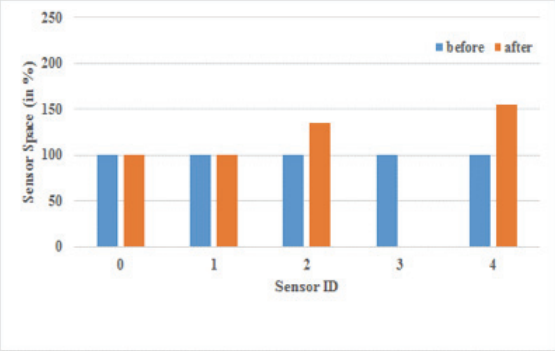

Figure 14. Space distribution after sensor re-allocation. Sensors 2 and 4 acquire the space from sensor 3 since it is considered dormant.

After re-enabling the task mapped to sensor 3 the application is executed again. Since the map had de-allocated sensor 3 space to sensors 2 and 4, it needs to re-allocate that space back to sensor 3.

When the application identifies that sensor 3 has no space it calls another function, Handle Feasibility similar to Case 1. The feasibility function uses the Sensor De-allocate function. The sensor space is then reduced for sensors 2 and 4 as shown in Figure 15. Sensors 2 and 4, which showed an increase in its space have now shrunk and sensor 3 is re-allocated its space to be used in the application. This process is also called Differentiation of the Brain Map (Doidge 2015). The application can be run continuously and hence the space for sensor 3 slot will continue to increase (reducing the extra space from sensors 2 and 4 as well).

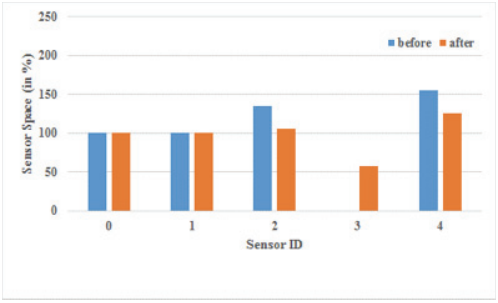

Figure 15. Space distribution after the application completed. Sensor 3 becomes active and regains the space it had sacrificed to sensors 2 and 4.

Conclusion

In this paper, we have demonstrated the navigation of the vehicle in the field in the presence of trees and ravine obstacles through complete coverage planning. The important features of this application were maximum coverage of the field, discovering ravines and invoking user help at the right places during the vehicle's journey through the field. The Help phase was kept simple and included only the most basic commands. The vehicle visits all the cells except those that were blocked by trees and ravines, thereby gathering as much information as possible about the field. A final list containing cells that are blocked and cells covered is stored by the vehicle.

Also, a dynamic structure called the sensor map using supervised learning was implemented on the robotic vehicle. This map is similar to the sensory brain map that is dynamically altered by the brain based on the usefulness of the sensory inputs. By using the robot navigation application it was possible to show the sensor space variation during the application and between applications through learning thus achieving a cognition model similar to human brain model for sensor data processing and management. This concept of dynamic map structures is very useful in resource management for multi-sensor frameworks and helps utilize the space available effectively.

Future Work

The next step in this research is to combine both trees and ravines obstacles. D-S Theory of evidence (Dempster 1968; Shafer 1976) will be used to infer a decision and the vehicle will follow accordingly. Interaction with the human user will be prompted when the fusion algorithm signals an uncertain state achieving safe maneuvering of the vehicle in the field. We will also formulate an evaluation proce-

dure for our HRI system by considering, the level of shared interaction between the user and the vehicle during the help phase, the type of information used to communicate between them, efficient interaction and scalability of the system (Scholtz 2002; Yanco and Drury 2004).

References

Breiman, L. 2001. Random forests. *Machine learning*, *45*(1), 5-32.

Chandrasekaran, B., & Conrad, J. M. 2015. Human-robot collaboration: A survey. *In SoutheastCon 2015* (pp. 1-8). IEEE

Chandrasekaran, B., & Conrad, J. M. 2016. Sensor fusion using a selective sensor framework to achieve decision and task execution. In *SoutheastCon 2016* (pp. 1-7). IEEE.

Choset, H. 2001. Coverage for robotics–A survey of recent results. *Annals of mathematics and artificial intelligence*, *31*(1-4), 113-126.

Dempster, A. P. 1968. A generalization of Bayesian inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 205-247.

Doidge, N. 2015. The Brain's Way of Healing: Remarkable Discoveries and Recoveries from the Frontiers of Neuroplasticity, New York, NY: Penguin Publishing Group

Fong, T., & Thorpe, C. 2013. Vehicle teleoperation with Collaborative control. *In Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2002 NRL Workshop on Multi-Robot Systems* (p. 195). Springer Science & Business Media

Galceran, E., & Carreras, M. 2013. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, *61*(12), 1258-1276.

Ghangrekar, S., & Conrad, J. M. 2009. Modeling and simulating a path planning and obstacle avoidance algorithm for an autonomous robotic vehicle. In *2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems* (pp. 1-3). IEEE.

Mitchell, T. M. 1997. Machine learning. Burr Ridge, IL: McGraw Hill, 45, 995.

Mitchell, H. B. 2007. Multi-sensor data fusion: an introduction. Springer Science & Business Media.

Patnaik, S. 2007. Robot Cognition and Navigation: An Experiment with Mobile Robots. Springer Science & Business Media.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Vanderplas, J. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825-2830.

Ramachandran,V.S. 2011. The Tell-Tale Brain: A Neuroscientist's Quest for What Makes Us Human, New York, NY:W. W. Norton & Company

Scholtz, J. 2002. Evaluation methods for human-system performance of intelligent systems. National Inst of Standards and Technology Gaithersburg MD Manufacturing Engineering Lab.Shafer, G. 1976. A mathematical theory of evidence (Vol. 1, pp. xiii+-297). Princeton: Princeton university press.

Yanco, H. A., & Drury, J. L. 2004. Classifying human-robot interaction: an updated taxonomy. In *SMC (3)* (pp. 2841-2846).