A CONTEXT-AWARE APPROACH TO THE NAVIGATION OF MOBILE ROBOTS

Conference Paper · October 2015

DOI: 10.13140/RG.2.1.3715.8481

2 authors:

Fernando de Lucca Siqueira
Federal University of Santa Catarina
9 PUBLICATIONS **118** CITATIONS
SEE PROFILE

Edson Roberto De Pieri
Federal University of Santa Catarina
140 PUBLICATIONS **638** CITATIONS
SEE PROFILE

XII Simpósio Brasileiro de Automação Inteligente (SBAI)
Natal – RN, 25 a 28 de outubro de 2015

A CONTEXT-AWARE APPROACH TO THE NAVIGATION OF MOBILE ROBOTS

Fernando de Lucca Siqueira*, Edson Roberto de Pieri*

*PPGEAS - CTC - UFSC 88040900 Florianopolis, SC Brasil

Emails: fernandols@inf.ufsc.br, edson.pieri@ufsc.br

Abstract— Humans gather context about the environment using their senses. These information allows humans to make an intelligent navigation through a scenario. Mobile robots can also benefit from this process. This paper proposes a framework for intelligent navigation of mobile robots using context information about the environment to enrich a semantic trajectory, using these information to define the proper behavior of the agent with a behavior tree. Experiments show that the semantic trajectory data can be used to incorporate more context information and build a more complete and complex behavior tree.

Keywords— Intelligent decision making, Autonomous mobile robots, Semantic Trajectory, Context Information

1 Introduction

Humans use their senses to obtain context from the surrounding environment and planning a path to a destination (Huang and Liu, 2004). This process of using context information for adapting the movement to a scenario is called intelligent navigation. Intelligent navigation is important for an agent to plan and execute tasks, changing its behavior when some change on the scenario impacts the original plan.

Context information describes the state of an entity. This information can be used in several ways to adapt a system to the current context. Past works uses context-aware systems in robotics (Bacciu et al., 2014) but they failed to link the context information with the robot trajectory. The current and future location of the robot could be impacted by the application context, meaning that the robot trajectory has a strong relationship with the context. This relationship can be modeled by a semantic trajectory and a behavior tree.

Semantic trajectory is a trajectory enriched with information and context data (Bogorny et al., 2014). This new concept was created to improve the raw trajectory (a trajectory without annotations) that has only information about the time and location of the object, limiting the possibilities to represent interesting knowledge. Usually in mobile robot applications, the trajectory is only used on the path planning, ignoring the executed trajectory. An executed trajectory can record the robot movement with all context related and be used to analyze the robot behavior and all context associated. This allows, for instance, to identify the current context of a failed task and correct possible implementation mistakes.

Robot behavior control can be implemented using approaches like finite state machine (Marino et al., 2009), or Markov model (Seyhan et al., 2013). Recently a new approach uses behavior tree for robot control (Colledanchise and Ogren, 2014). A behavior tree is a hierarchical finite state machine with custom nodes and leafs. These leafs can be an action or a condition. The execution of a behavior tree control the robot behavior, helping the decision making process.

An intelligent navigation of a mobile robot should have control of the current context, tracking of the robot trajectory and a robust behavior control. This paper proposes a framework for intelligent navigation in mobile robots with three concepts: context information, semantic trajectory and behavior tree. The context information describes the current state of the environment and the robot, providing data for the semantic trajectory building. The semantic trajectory contains all information necessary for the decision making process implemented through behavior tree. All data is stored to make queries of past interactions and improve the behavior tree. This framework could be used, for example, in a flight path control for a plane. With context information, like the weather, the plane can change its behavior to avoid a storm on the original path.

The paper is structured in the following way: In section 2 we present the basic concepts and definitions. In section 3 we present the framework structure and how each step works. In section 4 we present an initial experiment to test the framework. Finally, in section 5 we conclude the paper and describe the future works.

2 Intelligent Navigation Approach

Before presenting the framework, we must define the main concepts used in this work.

2.1 Navigation Concepts

Navigation is the ability of an agent to act based on its knowledge and sensor values to reach its goal positions as efficiently and as reliably as possible (Siegwart et al., 2011). There are three main approaches for the navigation of mobile robots: deliberative, reactive and hybrid.

The deliberative process requires a map and the initial position of the robot. A path plan-

ning algorithm computes a path from the robot location to the destination. This approach relies on having a map and a static scenario without changes.

In the reactive approach the robot moves reacting to the environment, therefore it does not have a planning trajectory. This approach is cheaper than the others, but does not guarantee the optimal path. It relies heavily on sensor data.

The hybrid approach merges the planning of the deliberative approach with the adaptation of reactive approach. A hybrid navigation has a path planning algorithm to create a trajectory plan for the robot and, while the agent moves, it senses the environment and adapt the trajectory to new or unpredictable obstacles.

For an intelligent navigation a robot must be capable to plan a path with previously known information and adapt to unpredictable changes on the map, so we use a hybrid approach. The hybrid approach has two trajectories: the planned trajectory and the executed one. This means that we can enrich both trajectories with context information.

2.2 Context Scenario

The main focus of this work is about how the context scenario can change the agent behavior. So, at first, we must define context. There is several definitions of context, but there is not an agreement (Bazire and Brézillon, 2005). This paper will use Dey (2001) definition:

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Any information that can characterize the situation of a entity in the application is consider as a context, more formally:

Definition 1 (Context) *A context C_i is an attribute that describes a characteristic or state of an entity, and a set $C = \langle C_1, C_2, ..., C_n \rangle$ is all the context associated with that entity. A context is an inner context when the entity is the agent and an outer context when the entity is the environment.*

Each entity of the application can be described by a set of context. For a mobile robot, the context is provided by sensors of the robot and the environment. The context of an entity can be changed as the time pass. For example, the context of energy level for a mobile robot will change its value as the robot keep moving and spending its energy. So we must have a structure where we can associate the context of the entity with a time reference. To accomplish this, we present the concept of semantic trajectory.

2.3 Semantic Trajectory

As the raw trajectory has few details of the moving object, recent studies proposed a new concept to add knowledge on trajectories called semantic trajectory. Bogorny et al. (2014) defines a semantic trajectory as a raw trajectory that has been enhanced with context information. These information can come from several sources and are connected with the trajectory location and time.

We use the semantic trajectory to represent all the information about the agent and the environment through the agent semantic trajectory. Figure 1 illustrates a semantic trajectory of a mobile robot where D_1, D_2 and D_3 are the robot destinations. Each point captured of the robot trajectory is associated with a set of context. This representation allows us to understand the context of each step of the agent, how the context evolve with time and the consequences of these changes on the agent movement. For example, the point of id 30 was captured at time 35 and has 80% of energy level. As the agent continues to complete tasks and moving through the map, the energy is consumed. At point of id 145 the agent has 15% of energy level and changed its behavior to recharge its battery. The semantic trajectory allows a structured view of the agent behavior and of the context evolution.

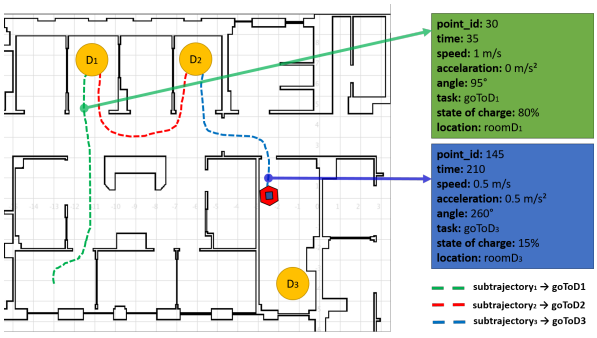

Figure 1: Semantic points and their attributes

Bogorny et al. (2014) proposed CONSTAnT (CONceptual model of Semantic TrAjecTories), the first model representation for semantic trajectory. CONSTAnT includes several information and the relationship between each information and the trajectory data. As CONSTAnT was developed specifically for humans trajectories, in this work we use an adaptation of the CONSTAnT model.

2.4 Behavior Tree

Another important concept in our work is the behavior tree. Marzinotto et al. (2014) defines behavior tree as a directed acyclic graph, with nodes

and leaves where any node without a child is a leaf node and the node without a parent is the root. A parent node can be a selector (represented by a ? symbol) or a sequence (represented by a → symbol). A leaf node can be a condition or an action.

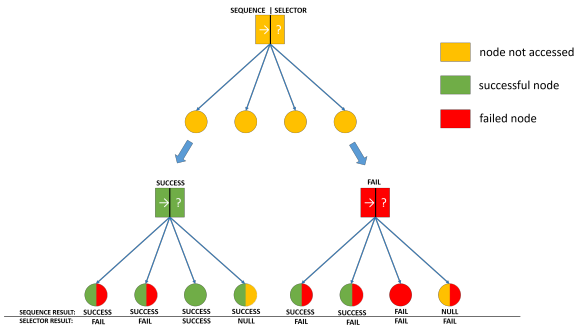

Figure 2: A Sequence behavior tree (left) with the success and fail condition and a selector behavior tree (right) with the success and fail condition

The execution of a behavior tree is based on the feedback of the leaves nodes. A condition can return success if is true or failure if is false. The action can return success if succeed, failure if fails, or running if is not complete yet. The result of the leaves will determine if the parent will fail or succeed. For a selector parent return success, at least one child must succeed. For a sequence parent return success, every children must succeed. Figure 2 illustrates the success and fail conditions of the selector and sequence. The execution of a behavior tree is always from the left to the right, meaning that the more to the left, the higher the priority. For example, in figure 2, the leaf node a_2 has higher priority then leaf node a_3, so it will be executed first.

Behavior tree can be used in robotics for decision making and behavior control (Colledanchise and Ogren, 2014). Marzinotto et al. (2014) integrate robot control with behavior tree proposing a framework implemented on ROS[1].

3 Contribution

In this section we present the new work definitions and contributions. With the concepts presented in the previous section, we built a semantic intelligent navigation framework for mobile robots. First, we must adapt the CONSTAnT model for mobile robots and formalize some definitions used by the model.

[1]ROS - Robot Operating System. Available at http://www.ros.org/

3.1 Conceptual Model of Semantic Trajectories for Mobile Robots

The semantic trajectory is the core element to link the context information with the agent behavior. The semantic trajectory model proposed in this paper is an adaptation of the CONSTAnT (CONceptual model of Semantic TrAjecTories). As CONSTAnT was developed for human trajectories, some changes was needed like, for instance, change objective to task and remove the transportation mean. For more information please see Bogorny et al. (2014).

To adapt the model, some definitions must be reviewed. First of all, the concept of semantic task is defined on definition 2.

Definition 2 (Semantic Task) *A task is an n-tuple* $\langle x, y, d, c, q, A\rangle$ *where* x *and* y *are the coordinates of the task location,* d *is the deadline,* c *is the total cost,* q *is the priority and* A *is the set of actions required to complete the task.*

A task is the goal of the movement. To complete a task, the agent must execute a set of actions. The concepts of semantic trajectory, semantic subtrajectory and semantic point from CONSTAnT were adapted to the framework. A semantic trajectory is a record of the movement of the agent with context information. An agent can have several semantic trajectories.

Definition 3 (Semantic Trajectory) *A* semantic trajectory Ts *is an triple* $\langle oid,tid,Ss\rangle$ *where oid is the identification of the agent, tid is the identification of the trajectory and* Ss *is a set of semantic subtrajectories.*

The main context information associated with the semantic trajectory is the agent. The agent can have an identification, name, dimensions, hardware, etc. The semantic trajectory is composed by a set of semantic subtrajectories. A semantic subtrajectory, defined on definition 4, is a fragment of the semantic trajectory associated with a specific task.

Definition 4 (Semantic Subtrajectory) *A* semantic subtrajectory Ss *of* Ts *is a n-tuple* $\langle tid, sid, P, task, c, startTime, endTime\rangle$ *where tid is the semantic trajectory identification, sid is the semantic subtrajectory identification,* P *is a time oriented set of semantic points, task is the semantic task of the subtrajectory,* c *is the total cost of the movement, startTime is the time of start and endTime is the time of end.*

The semantic point is the atomic element of a trajectory. The semantic point is also responsible for keeping track of the context information, as each point is related to a set of context that can change its value through time.

Definition 5 (Semantic point) *A* semantic point *ps is a n-tuple* $\langle sid, pid, x, y, \theta, t, C \rangle$*, where sid is the identification of the semantic subtrajectory, pid is the identification of the semantic point, x and y are the spatial coordinates of the point in a two dimension plan,* θ *is the angular orientation of the agent with respect to the x axis, t is the timestamp and C is a set of context information.*

The time parameter keep each context information related to the corresponded semantic point. This way we can relate each agent behavior to its context.

3.2 Framework Proposal

For a mobile robot to complete a task successfully it must analyze its state, the environment and its context to choose the best possible actions. Our framework is based on three main steps to accomplish this: context gathering, semantic trajectory building and decision making.

The first step is context gathering. To understand the environment and the state of the agent, we gather context information available on the application. Each scenario or application will have its own context. Context information can come of four sources: agent, environment, user and previously known information. The agent context, or inner context, describes internal values and states like speed, acceleration, energy level, etc. These information are obtain through sensors of the robot. The environment context, or outer context, is important to understand the state of the scenario. The user can also input its own context via console like, for example, an user preference, a command, etc. And finally, the context can be previously known information like, for example, a map, a task, or any other information stored on a database.

An agent has two trajectories: a planned trajectory and an executed trajectory. The planned trajectory is the trajectory generated by a path planning algorithm and describes the path the agent should follow. The executed trajectory is the trajectory that the agent execute, that could be different from the planned trajectory by unpredictable factors on the map or error in following the path. The planned trajectory is enriched with previous knowledge or by predict the context values. The executed semantic trajectory is enriched with the context captured by the previous step. As the robot moves, each timestamped location is captured and transform into a semantic point. To associate the context value to a trajectory point we use the timestamp of the elements. A context of timestamp t_i will be associated to a semantic point of timestamp t_i. The task information is decided by the decision making algorithm, and it is associated to the semantic subtrajectory, where a set of points are related to the same task. At the end of a task, we will have a semantic subtrajectory, where the set of semantic points will represent the evolution of all the context associated through the time.

At the same time as the trajectory building, the framework uses the context information for the decision making. The decision making process uses a behavior tree to choose the most appropriated behavior. The set of possible behaviors are application dependent. For example, in a patrol application where the agent must patrol the areas A and B; the main behavior should have two actions: goToA and goToB. As the agent must execute all the actions in this order, they are connected together by a sequence node named *patrol*. Besides patrol the area, the agent must not run out of energy. So we must have a behavior to keep the agent alive. For this behavior, first we need to verify the state of charge (SoC) of the battery. This step is solved by a condition node that will check if the agent need to recharge. If the agent need to recharge, the next step is to move to an energy source. This is a move action. The last action of the behavior tree is to recharge the battery when the agent get to the energy source. The *keep alive* behavior is a sequence node connecting the condition and both actions nodes.

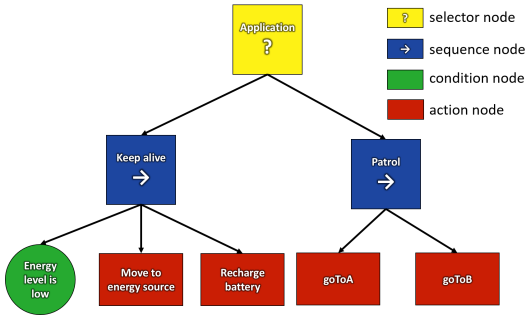

Figure 3: A behavior tree example for the patrol application

With the *patrol* and *keep alive* behavior tree we have the two behaviors to this application. Now we must connect them. As the agent can not patrol without energy, the *keep alive* behavior must have the highest priority. The figure 3 illustrates the complete behavior tree. First, the root node will try to execute the *keep alive* behavior. If the condition (energy level) returns success, it will keep running the *keep alive* behavior. If the condition fails, the behavior tree will execute the *patrol* behavior.

With more context information we can define more complex behavior of the agent to adapt the navigation to the current state of the application. The semantic trajectory can be used to analyze the context changing and choose the proper behavior.

4 Experimental Verification

To test our framework, we implemented a patrol application for a mobile robot on the Gazebo[2] simulator. The code was implemented on ROS framework in C++ with the behavior tree library by Marzinotto et al. (2014). Our simulation uses a x80sv driver implementation by Saxion Lifescience Engineering and Design[3]. We implemented a simple experiment to easily demonstrate the application of the proposal, but the framework can be used on much more complex scenarios.

4.1 Simulation Scenario

Our patrol application follows the same example presented on section 3.2. The robot must continuous patrol two areas on the map (A and B). The robot should also check the SoC of the battery and, if is low, go to the energy source to recharge at an energy source. The behavior tree is the same as figure 3.

For this application, we implemented an ignore-failure sequence node. The sequence node, as presented on section 2.4, executes each child node as they return success and stop the execution if some return failure. But in a patrol application, the robot should continue to patrol the next point even if it fails to patrol the previous. For example, if the access door for point A is closed, the robot will failure to patrol this area. With a sequence node, it will not try to patrol the next area, as the patrol A action will return failure and the whole patrol behavior returns failure. So in the patrol behavior, each patrol action will not return failure, but the patrol behavior can (more on that later).

A problem with this behavior tree is the energy level. The keep alive behavior is responsible to check the SoC and recharge the battery. But as the robot is patrolling the area, the energy can reach the critical SoC. As the patrol behavior is a sequence node, it will try to execute all children nodes before resetting the behavior tree, executing the keep alive behavior. To avoid this situation, as the robot moves, the patrol behavior keeps checking the SoC. If the SoC get to the critical level, the patrol action is aborted and the patrol behavior returns failure. So in the next tick, the keep alive behavior will be activated, as the battery is on critical SoC. We simulate the battery of the robot as an ideal battery (never gets addicted, the state of health is always 100% and get full charge at recharges). A full battery has 100 power units. The simulation took sixty minutes in an empty world and the critical SoC was set to 20%.

[2]Gazebo Robot Simulator. Available at: http://gazebosim.org/

[3]Saxionled/x80sv github. Available at: https://github.com/SaxionLED/x80sv

4.2 Intelligent Navigation Results

The robot patrol go from region A to region B and recharge the battery at an energy source. At the end, of 120 tasks, the robot failed 12 (10% of the tasks) and completed 108 tasks. To understand the failures, we must analyze the semantic trajectory of the agent. The semantic trajectory showed that the robot start a task without knowing if it would have enough power to finish it. For example, at a certain time, the robot was almost finishing a goToB task but reach the critical SoC, aborting the task and triggering the keep alive behavior. The robot then changes the behavior to keep alive, going to an energy source to recharge the battery.

To try decreasing the number of failed tasks we must change the behavior tree. The semantic trajectory of the robot provide information about the execution and allows us to create a new proper behavior tree. The semantic points stores the energy value of each time instant of the trajectory and the sub-trajectory has the total cost of current task. The average cost of each task (go to region A and go to region B) was calculated around 10 power units. In the previous example, when leaving region A, the SoC was 28 power units. As the task cost 10 power units, the robot reached the critical level before finish the task. So we update the data to store that the average cost of each tasks is 10 power units, creating a new context information on the application. Now we have to change our behavior tree to incorporate this new context.

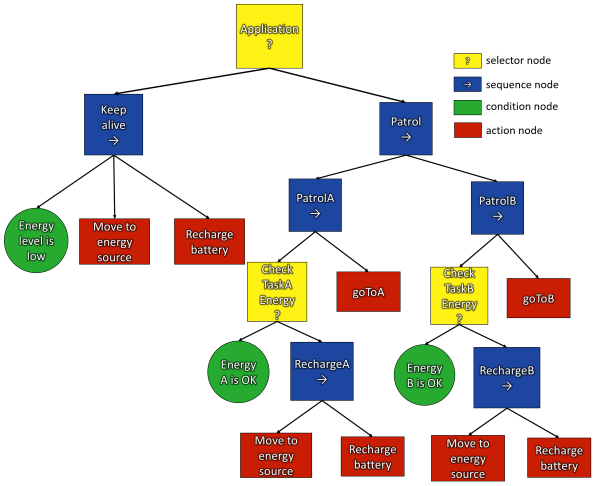

Figure 4: The new behavior tree generated after analyzing the context information

The figure 4 illustrates the new behavior tree, now with more context information. The patrol behavior now has two sequence nodes: PatrolA and PatrolB. Each of these nodes has two children nodes: a selector node and the goTo action. We will describe the execution of PatrolA behavior

tree. The selector node CheckTaskAEnergy first check if the current SoC is bigger than the critical SoC plus the estimate cost of the task. If this condition returns true, the selector node returns true and the PatrolA node executes the goToA action. But if the EnergyAOk condition returns false, the robot SoC is not sufficient to execute the goToA action, so the selector node executes the next child node: RechargeA. The RechargeA node is a sequence node that has two actions: moveToEnergySource and rechargeBattery. So, the execution of RechargeA will recharge the robot battery and returns true. The selector node will then return true and the robot will execute the goToA action. This same process is executed on PatrolB. This ensures that a patrol task will only be executed if the robot has enough SoC. But, if for some reason, the cost of the task is increased (a new obstacle in the way, for example) and reaches the critical SoC, the robot will abort it and execute the keep alive behavior as before.

We made a new simulation with the new behavior tree. The table 1 summarizes the results. This new behavior tree proved to be more efficient than before, without any failure and costing less energy. As more context information are added on the system, more intelligent the navigation becomes. New context information can be added to prepare the robot to new situations. The behavior tree must be revised to incorporate this new context, resulting in new behaviors.

Table 1: Comparison of tests for each behavior tree. Number of Tasks(NoT); Tasks succeded(TS); Success rate(SR); Recharges(R); Total cost(TC)

Test	**NoT**	**TS**	**SR**	**R**	**TC**
1	120	108	90%	13	118290
2	106	106	100%	11	105141

5 Conclusion and Future Works

A human navigates by knowledge of the map and absorbing context information of the environment. This context help the decision making process, changing the behavior based on the current information. An intelligent navigation for mobile robots can reproduce the same process. We propose a new framework for context aware navigation of mobile robots. The framework is based on three main concepts: context (information about the agent and the environment), semantic trajectory (a concept model to organize and linking the context information with the agent movement) and behavior tree (a decision making model that defines the proper behavior of the agent based on the context).

Our experiments shows that these three concepts together manage to make an intelligent navigation for a patrol robot based on its energy level. The data stored were then used to compute new context information, resulting in a more complex behavior tree that improves the efficiency of the application.

Future works include multiple robots cooperation thought the framework, behavior prediction based on historical data, ontology-based context and more complex applications for experiments.

References

Bacciu, D., Gallicchio, C., Micheli, A., Di Rocco, M. and Saffiotti, A. (2014). Learning context-aware mobile robot navigation in home environments, *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference on*, IEEE, pp. 57–62.

Bazire, M. and Brézillon, P. (2005). Understanding context before using it, *Modeling and using context*, Springer, pp. 29–40.

Bogorny, V., Renso, C., Aquino, A. R., Lucca Siqueira, F. and Alvares, L. O. (2014). Constant–a conceptual data model for semantic trajectories of moving objects, *Transactions in GIS* **18**(1): 66–88.

Colledanchise, M. and Ogren, P. (2014). How behavior trees modularize robustness and safety in hybrid systems, *Intelligent Robots and Systems (IROS)*.

Dey, A. K. (2001). Understanding and using context, *Personal and ubiquitous computing* **5**(1): 4–7.

Huang, B. and Liu, N. (2004). Mobile navigation guide for the visually disabled, *Transportation Research Record: Journal of the Transportation Research Board* **1885**(1): 28–34.

Marino, A., Parker, L., Antonelli, G. and Caccavale, F. (2009). Behavioral control for multi-robot perimeter patrol: A finite state automata approach, *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, pp. 831–836.

Marzinotto, A., Colledanchise, M., Smith, C. and Ogren, P. (2014). Towards a unified behavior trees framework for robot control, *Robotics and Automation (ICRA), 2014 IEEE International Conference*.

Seyhan, S. S., Alpaslan, F. N. and Yavaş, M. (2013). Simple and complex behavior learning using behavior hidden markov model and cobart, *Neurocomputing* **103**: 121–131.

Siegwart, R., Nourbakhsh, I. R. and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*, MIT press.