Implementation of a Prototype Context-Based Reasoning Model onto a Physical Platform

Viet C. Trinh, Brian S. Stensrud, Avelino J. Gonzalez

Intelligent Systems Laboratory
University of Central Florida
Orlando, FL
vtrinh@isl.ucf.edu, brian@isl.ucf.edu, gonzalez@ucf.edu

Abstract
This is SAWMAS Context-Based Reasoning (CxBR) is an intuitive paradigm for human behavioural representation, presenting a new and unique method for modelling autonomous intelligent agents. This modelling paradigm is based on the simple fact that knowledge is modularized into contexts, which contain all of the intelligence needed to successfully address particular situations. CxBR models have shown promise in recent exercises within simulated environments. To date, these simulations have involved tactical missions undertaken by submarines, tanks, and other military vehicles as well as driving missions undertaken by various automobiles. This paper discusses the first attempt at integrating this modelling paradigm in a physical platform - specifically the implementation process and the many issues addressed when translating this model onto a live agent. In a simulation, CxBR models execute in a controlled environment, with controlled inputs (i.e. sonar, positioning) to the agent. The major problems encountered with porting this agent onto a physical platform are caused by environment unpredictability, input noise, and agent capability degradation. An iRobot ATRV-Mini agent entity was used in this integration. Due to the basic functionality of the ATRV-Mini, a very primitive CxBR model was developed. The model, in no sense, presents the full potential of the modelling paradigm nor the iRobot itself, but instead illustrates the ability of integrating a CxBR model onto a physical platform. This paper will include the details of the scenario used to prove the feasibility of controlling physical entities through CxBR, it will also describe the evaluation made of the exercises. Although this task seems fairly simple in nature, it becomes a challenging task to integrate the CxBR model to the physical robot. This integration marks the first attempt to incorporate a CxBR model into a real-world scenario, and its success paves the road for integration of much larger and sophisticated models on live physical platforms. Furthermore, this proof-of-concept demonstration is a springboard for applying and evaluating new CxBR research - such as collaborative CxBR models, the competing context concept, and the automation of the CxBR model development process using machine learning or knowledge acquisition tools - on not only simulated but also physical autonomous platforms.

Introduction
This is In the past decade, Context-Based Reasoning (CxBR) has been applied in various research projects. It has established itself as a successful paradigm for human behavioural representation of intelligent autonomous agents, as shown in its many publications. The initial concepts of CxBR started as scripts implemented by Gonzalez and Ahlers [Gonzalez & Ahlers 1993], and enhanced by Norlander’s CxBR Framework [Norlander 1999], the first attempt at constructing an engine for CxBR. Current research in CxBR includes learning, competing contexts, planning, and collaborative behaviors (teambowr). All previous research has been in tactical applications in the military-related domain, with the exception of automobile driving. It involves the control of one or more autonomous agents in a simulated environment. Simulations can be used for training of military personnel and testing of scenarios with simulated vehicles. Although simulations are useful in training and analysis, models can also be put to practical use in the real-world. There is more control of the environment in
simulation than in real world situations. Real world situations are unpredictable, input data can be noisy and the actions taken by the agent are not perfect. Although noise can be added to simulations modelled after the real world, it still remains a well controlled environment. Therefore the idea of implementing and testing a CxBR model on a physical platform is necessary to fully evaluate the potential of CxBR. Military vehicles as well as automobiles are designed to function in the real world, giving greater importance in integrating artificial intelligence models into these physical platforms. This paper provides an introduction to CxBR, describes the physical platform (an iRobot), the scenario used for this endeavour, the implementation process and methods, and the results obtained from the study.

**CxBR Overview**

Context-based Reasoning (CxBR) is “a novel behaviour representation paradigm that can effectively and efficiently be used to model the behaviour of intelligent entities in a simulation.” [Gonzalez & Ahlers, 1998]. This paradigm is designed to be used for human behaviour representation, utilizing the use of contexts as a reasoning method, similar to how humans make tactical decisions [Gonzalez & Ahlers, 1998]. CxBR models have shown success in recent exercises within simulated environments. To date, these simulations have involved tactical missions undertaken by submarines, tanks, and other military vehicles, as well as automobiles on the road. CxBR is based on the idea that [Gonzalez & Ahlers, 1998]:

- “Any recognized situation calls for a set of actions and procedures that properly address the current situation.”
- “As a mission evolves, a transition to another set of actions and procedures may be required to address a new situation.”
- “What is likely to happen under the current situation is limited by the current situation itself.”

CxBR is focused on the concept that humans think in terms of contexts. [Norlander, 1999]. Contexts encapsulate knowledge/intelligence about appropriate actions needed to address specific situations [Gonzalez & Ahlers, 1998]. The context controls the behaviour of the autonomous agent, assigning it appropriate actions based on the active major context in which the agent is situated. The CxBR paradigm is composed of a tactical agent, a mission context, major contexts, sub-contexts, and sentinel rules.

**Tactical Agent**

This is the agent that is completely controlled by the CxBR model. It is given a mission to execute, and all of the allowable contexts required to accomplish the particular mission assigned to it. Actions the agent performs are controlled by the CxBR model, specifically the contexts within the model.

**Mission Context**

The mission context has no direct control of the agent. It is a high-level specification mechanism for the entire scenario. The mission context defines all of the parameters used for the scenario and the agent itself (i.e., terrain data, scenario boundaries). The objective of the mission is contained in the mission context, along with the list of applicable major contexts [Gonzalez & Ahlers].

**Major Context**

A major context encapsulates all of the knowledge required to execute a certain task, based on the current situation. The context contains all of the functions, rules, and compatible context transitions for that context, as well as a list of all applicable sub-contexts [Fernlund & Gonzalez, 2002]. Only one major context is in control of the agent at any one given time-this is called the active context. There is always exactly one major context active. Defined in the mission context are the default and initial major contexts. The default context is the major context activated when no applicable contexts available. The initial context is the active context with which the mission begins.

**Sub-Context**

A sub-context is a representation of low level procedures that are auxiliary to major contexts [Gonzalez & Ahlers]. A major context can contain many sub-contexts to assist the context in its execution. Sub-contexts can also belong to many major contexts, this allows reusability in CxBR [Gonzalez & Ahlers].

**Sentinel Rules**

Sentinel rules are those monitoring rules that dictate that a transition to another context is necessary. Sentinel rules constantly monitor the condition of the agent and its environment. Whenever a context is no longer valid and a transition is necessary, a sentinel rule is fired and a context switch is initiated.

The CxBR Framework developed by Norlander [Norlander, 1999] allows the integration and development of various CxBR models. The framework performs all of the lower level actions required of CxBR, (i.e., maintaining states, transitions, initializing contexts, etc.) allowing the developer to concentrate on generating the mission, the contexts, the agent interface, and the agent itself. The implementation in this paper utilizes this framework to execute the CxBR models.
iRobot Physical Platform

The physical platform used as a host for integrating CxBR is iRobot’s ATRV-Mini. The ATRV-Mini is an all terrain robot designed specially for research purposes, ideal for this project. It is the smallest of the ATRV series, therefore the simplest to implement. This paper will makes references to the iRobot ATRV-Mini as simply “iRobot”. The iRobot is controlled by the Mobility™ software, residing in its onboard computer [iRobot Corporation 2003]. The iRobot’s computer runs on a Redhat Linux 6.2 operating system. Some of the key features of the iRobot are [iRobot Corporation 2003]:

- Battery Operated: 2 - 4 hours
- Centralized, general purpose onboard computer
- Breezenet wireless network
- 16 Sonars (6 front, 8 side, 2 rear)
- An internal odometer
- Joystick control

The wireless network and the battery operation allows the iRobot to freely roam around a terrain (within the wireless network’s range) and establish a constant connection to a client computer. The iRobot’s centralized onboard computer allows remote access through SSH as well as programming and compiling directly on the iRobot. For this case, the CxBR program controlling the iRobot can reside onboard, eliminating the problem of integrating a networked application. The robot is capable of moving forward and backwards, with 360° of rotation. While moving or rotating, the iRobot’s internal odometer maintains its relative position and heading. The sonar enables the iRobot to detect objects in its immediate surroundings. The mobility software allows the iRobot to detect objects up to four meters away. However, through field testing, the iRobot is only capable of detecting objects approximately two meters away. The iRobot has built-in C++ header files that allow a CORBA connection to the Mobility™ software that controls the iRobot. A full list of features can be obtained from the official iRobot Web site (http://www.irobot.com) [iRobot Corporation 2003]. Details on the integration are presented in the Integration section.

CxBR Mission Scenarios

The scenario put forth for this paper is a simple tactical mission. The mission is designed as a proof of concept, and has no practical utility. The mission is a basic scouting mission in which the iRobot moves around a terrain in search of an enemy entity. Sub-contexts were not used because of the simplicity of the mission and contexts. The proof of concept is to integrate a CxBR model onto an iRobot and analyze its performance.

Mission Objectives: The objective is to maneuver an iRobot around an open area looking for a single enemy entity. Upon detection, determine the hostility level of the enemy. If it approaches, consider it hostile and retreat. If the enemy retreats, follow it at a close distance. If the enemy is not responsive (i.e., stationary), execute an end of mission signal and retreat to the original starting position.

Mission Constraints:

- Flat 8m x 8m surface (paved parking lot).
- One enemy present, no other objects.
- A predefined wait time
- Objective is to detect a stationary enemy.

Context Set:

- locateEnemyContext - Maneuver the iRobot around the field searching for the enemy. Moving from waypoint to waypoint. (*initial and default context)
- determineEnemyHostilityContext - After the enemy is detected, the iRobot determines the enemy’s hostility by waiting for it to move in either a retreating or advancing direction. If the enemy is not moving, it is considered stationary. After an approach or retreat maneuver, the enemy’s action is re-scanned therefore transitioning back to this context.
- retreatContext - If the enemy is approaching, retreat moving backwards.
- approachContext - If the enemy is retreating, approach the enemy slowly. (Half the distance between the iRobot and the enemy).
- stationaryEnemySignalContext - Signal (using a signature maneuver) that the enemy has neither retreated or advanced for certain period of time, then conclude mission by leaving the area and going back to its original position.

Context Methodology

The five contexts or the context-base that is implemented in simulation assumed ideal iRobot actions and a perfect environment to interact with. In the case of simulation, these aspects can be made ideal and perfect, however, in the real-world these factors can not be controlled as well or at all. As simple as the mission and implementation may be, the real world model needed many modifications to adjust the contexts and AIP to these inconsistencies and inaccuracies experienced with the iRobot and its environment. Below is a description of how each of the contexts is implemented. Figure 1 illustrates all of the legal context transitions.
locateEnemyContext

Originally, the iRobot was supposed to move and rotate simultaneously to achieve a more circular and continuous movement scheme. Because of the terrain, it was difficult to perform both actions at the same time while remaining accurate. Therefore rotating and moving were separated to reduce possible inaccuracies. The iRobot now moves from waypoint to waypoint, moving in a straight direction and turning only at waypoints. Because the iRobot moves and rotates to a tolerance, the position and heading might be somewhat inaccurate. Performing one straight movement to the waypoint would cause the iRobot to possibly miss it because it would never be within the certain range tolerance. Therefore its movement is broken down into small straight movement sections with minor rotations to compensate for the iRobot being off course.

determineEnemyHostilityContext

This context is used to determine the enemy’s actions and how the iRobot should react to the current situation. This context is transitioned to whenever the enemy’s action needs re-evaluation. In order to accurately determine the enemy’s hostility level, the iRobot remains stationary. Whenever the enemy moves, the iRobot’s uses its internal odometer and sonar to analyze the enemy’s movement and position. If the enemy is moving closer, it is considered approaching and the retreatContext is called upon. If retreating, then the approachContext is called upon. In order to detect a stationary enemy, the enemy has to remain within a certain tolerance area when detected. This tolerance area allows to compensation of sonar fluxations.

retreatContext

When an enemy is approaching, it natural for a human to turn around and retreat. However, in vehicles, it is more advantageous to advance backwards because turning is quite slow. If turning is performed to quickly, the vehicle might flip over. The iRobot has sonar in the rear allowing it to have limited visibility of the enemy while moving backwards. Originally, the iRobot would only move a small distance away from the enemy and then perform another detection cycle on it. This process was too slow and the iRobot was susceptible to collision with the oncoming enemy. Therefore the iRobot quickly moves three meters backwards completely avoiding the enemy.

approachContext

Whenever the enemy is retreating, the iRobot will approach slowly with reduced speed. Ideally, the iRobot would trail the enemy robot until it becomes offensive and advances on the iRobot. However, dealing with speeds, distances, heading, and timing simultaneously on the iRobot was very difficult. Therefore, the iRobot would approach the enemy by moving half way between its position and the enemy’s position. Sometimes the enemy can retreat very fast and drop out of the iRobot’s sonar. This is will shift the context back into locateEnemyContext.

stationaryEnemySignalContext

This context performs a signature move for the iRobot by rotating in one direction, then rotating in the other direction. This was the simplest context to implement because it did not involve the enemy. The iRobot was to move back into its original starting position to measure the level of inaccuracy caused by executing the mission.

Implementation Process

The implementation process was broken down into three steps: Developing the CxBR model on the current CxBR framework, development of a CxBR to iRobot interface, integration of the CxBR model with the iRobot. After implementation testing, refining, and validation of the model occur.

CxBR Model Development

The initial CxBR model was developed on a separate machine running an identical operating system to that of the iRobot (Redhat Linux 6.2). This eliminated discrepancies between the development platforms and
Figure 2: The Scouting Mission Scenario layout with the iRobot in the locateEnemy context.

compiler versions. The CxBR framework is build to run on various platforms [Norlander 1999]. Within the CxBR model, the iRobot entity is represented by an agent class called AIP (Autonomous Intelligent Platform). This agent class performs all of the high level functions of the iRobot utilized by the contexts. The contexts classes were then implemented, calling upon the iRobot agent to perform actions. Within the contexts classes are the sentinel rules for the contexts and the valid context transitions. The mission class is then implemented with the agent and all of it respective contexts. The models developed were run using a dummy robot agent interface to validate of the functionality of the model.

iRobot Interface

The iRobot is implemented as an inherited class (iRobot AIP) from the base AIP class. Within the workings of the iRobot agent class (iRobot AIP), the iRobot interface is called upon by this class to allow the framework, more particularly, the contexts used by the framework, to control of the iRobot through its functions. The functions to control the robot require a CORBA connection to the mobility software running on the iRobot. The iRobot interface performs the low-level functions of the iRobot (i.e. communicating with the mobility software), while the iRobot tactical agent performs the high-level functions. The commands to move and rotate the iRobot were the next important when programming the interface. The command to move the robot only accepts the speed in meter/sec (positive-forward direction, negative-reverse direction) and not a position to move to. However, in the scenarios implemented, a distance/position is required. To fix this problem, the odometer was used. The robot was given a certain fixed speed to travel and the iRobot positioning on the odometer was constantly monitored until the desired position was reached within a certain tolerance. The same problem occurred with rotation, instead of translational velocity, angular velocity and the odometer heading was used. The iRobot was given a certain fixed angular velocity to rotate, and the heading on the odometer was constantly monitored until the desired heading was reached, again within a certain tolerance. The sonar is used to detect objects in close proximity of the iRobot. There were sixteen sonar and the data collected by the each sonar was the x (front) and y (side) distance from an object. If the absolute distance is four meters (the range of the sonar) then it is assumed there is nothing in front of that particular sonar. However, the relative detectable distance used is two meters. The stop command was issuing a zero speed command. These are the low level command implemented by the interface along with monitoring and reporting the iRobot’s internal odometer and sonar readings.

CxBR Model Integration

The integration of the model with the iRobot required porting the CxBR model, framework, and interface directly onto the iRobot itself. After the establishment of the iRobot interfacing, the iRobot AIP, along with the CxBR software, the model was ready for integration. Figure 3 is a diagram of the components used for the integration, and the flow of control from the CxBR model to the physical iRobot. The iRobot is ultimately controlled by the contexts implemented in the CxBR model. However, the contexts use high-level robot control functions implemented in the iRobot agent class. The iRobot agent in turn calls lower level functions implemented in the interfacing code. The mobility software is what controls the iRobot’s motor and sonar functions. The integration process began by porting the CxBR model from simulated iRobot AIP to the physical iRobot AIP implementing the iRobot interface.

Testing

The perfect world/agent assumption did not prove true in the real world as it could in simulation. It was quickly noticed during testing that the iRobot had inaccurate movement, rotations, and sonar readings. To the iRobot itself and the mobility software, the movement and sonar readings are perfectly executed. The iRobot did not know of its own inaccuracies. However, as the developer, the inaccuracies were apparent by observing the iRobot in action. Therefore the iRobot interface and the iRobot agent code were refined to address these particular issues. The refinement of the model was fine tuned to allow this margin of error in movement and sonar detection to occur while maintaining adequate performance levels. The agent
actions were modified to partially compensate for these inaccuracies. Compensations were introduced in the low-level iRobot interface. The purpose of testing was to determine whether the CxBR model was proficient enough to adequately compensate for the inaccuracies of the iRobot and its function as well as its performance in the real world. Five runs for scouting scenario were used to test the capabilities of the CxBR model on the robot.

The test metrics used for testing were:

1. A sequence of events is plan for the enemy. Does the iRobot react appropriately to the enemy actions?
2. Does the model correctly transition contexts? Are these transitions distinctly visible during the execution of the mission?
3. Does the model complete its mission successfully?
4. How did the iRobot telemetry and sonar inaccuracies affect the performance?

These test metrics are implemented to analyze the proficiency of the CxBR model to execute the specified missions. Testing showed how CxBR was able to cope with a real world environment.

Results

After extensive execution and refinement of the CxBR model, the results proved very interesting. The iRobot is able to execute its missions and the contexts are visible during the execution. However, the context transitions are not perfect. Some transitions are completely missed at times when a transition is required, as well as incorrect transitions made due to the degraded capabilities of the iRobot and the physical environment. Below is an explanation of the results. There were timed pauses made between transitions in the CxBR model. This provided the framework a small time period where it was able to analyze the situation and transition properly. If the time gap is removed, the sonar would read very similar values before and after enemy evaluation, possibly causing incorrect transitions to occur. This time delay mechanism was integrated in the refinement stage to allow time to differential between movement and error. These pauses also constitute a time-step in which the iRobot performance can be evaluated. Performance evaluation based on the test metrics:

1. The iRobot is able to detect the enemy whenever it comes within the sonar radius of the iRobot. However, as the battery life degrades the enemy has to come closer to the iRobot for detection to occur. When the enemy moves very close to the robot representing a hostile action, the iRobot is not able to detect such close movement. When the enemy is approaching (hostile) or retreating (not-hostile) the iRobot is able to detect the majority of these occurrences and reacts to the enemy by retreating and approaching respectively.

2. For the locateEnemyContext, the transitions are executed properly. During normal operations of locateEnemyContext, the iRobot is constantly moving from one waypoint to another. When an enemy is detected, the iRobot immediately halts and transitions to determineEnemyHostilityContext. The determineEnemyHostilityContext is the most problematic of all. This context connects all other contexts together and therefore has the most work to perform. Sometimes the iRobot would not detect the robot moving closer or further from it and remains in determineEnemyHostilityContext, ignoring the need for a context switch. Sometimes the iRobot thinks the enemy is lost and begins locate-EnemyContext in search for the enemy when the enemy is still nearby. To prevent problems with context transitions in the retreat and approach contexts, an automatic context transition is made at the end of each execution of the actions (retreating or approaching) back to determine-EnemyHostilityContext. However, there were no incorrect transitions to invalid contexts, except for
transitioning to locateEnemyContext when the enemy is still close. Is occurs when the enemy drop off the sonar readings. This is a problem of the physical iRobot, and not of the CxBR model. To the CxBR model, all transitions are valid.

3. The iRobot as able to detect a stationary enemy and complete its mission by executing the stationaryEnemySignalContext.

4. The iRobot’s sonar inaccuracies greatly affect the performance. Sometimes, there are misreading in the sonar and the enemy disappears off of the sonar map, thus causing an incorrect transition to locateEnemyContext. The telemetry inaccuracies have a minor affect on the performance. The CxBR model is still able to correctly transition and execute contexts with telemetry error. Only the actions are only minimally affected.

Overall, the iRobot is able to perform the mission. It encounters difficulties in the determineEnemyHostility-Context and the inaccuracies in the telemetry and sonar reading. Determining the difference from a not-in-range, stationary, hostile, and non-hostile enemy action can be confusing to the iRobot at times. Because of this time gap introduced between contexts, the iRobot may not be able to react to all of the enemy’s actions. However, with the compensations added, these errors are rare, and the iRobot is able to perform proficiently in its environment.

**Summary**

The iRobot is a well-developed platform for testing the validity of the implementation of a CxBR model onto a physical platform. Overall, the model was successful in controlling the iRobot through the missions. However, there were some problems faced. The main problem with the iRobot was the inaccuracy of its internal odometer and the sonar. Within the scenario, one of the end-of-mission actions is for the iRobot to return to its original (0,0) position and face forward at 0°. When the iRobot returned to its original position, the iRobot was positioned incorrectly by a half to one full meter in the x and y directions. The direction the iRobot faces is between 5° to 10° off of the original position. This inaccuracy was also detected by observing the iRobot constantly move out of its programmed boundaries. This error is not related to the software, but instead to the physical machinery of iRobot itself. The problem mainly arises from the tires of the iRobot and the terrain the iRobot is in. The testing was performed in a large parking lot with an asphalt pavement. When the wheels move on the pavement, the iRobot loses some traction and the actual positioning is different from the desired and internally measured position. Internally, the iRobot thinks it travels a certain distance, but in reality it has moved less than the desired distance, creating an inaccuracy in the odometer. Therefore, every movement the iRobot performs can further skew the odometer reading. The sonar inaccuracy occurs slowly with time. Since the iRobot is battery powered, the sonar’s detection radius decays with the time the iRobot is operating in the field. This affects how the iRobot detects the enemy. With degraded sonar capabilities, the enemy has to move closer to the iRobot to be detected, and a collision may occur, limiting the time for testing.

**Future Work**

The scenario used in this integration was simple. It does not portray the full robustness of the context-based reasoning paradigm. Integrating a more complex CxBR model would be the next step in this research. Because of the limitations of the current iRobot platform, better robotic platform will be used to create this model. Learning from observation is another topic which uses this robotic implementation. With the ability of porting CxBR from simulation to real world, learning can be performed by observing experts maneuvering an agent in the real world, building a CxBR model through the knowledge acquired, test it in simulation, and port the model back onto the original agent. One major extension to this work should include the development and integration of a full CxBR model onto a complex physical platform.

**References**


