RoboCup Rescue Simulation 2016 – Technical Challenge Team Description MRL (Iran)

Pooya Deldar Gohardani, Siavash Mehrabi, Peyman Ardestani, Erfan Jazeb Nikoo, Sajjad Rostami, Mahdi Taherian

1 Mechatronics Research Laboratory, Islamic Azad University, Qazvin Branch, Qazvin, Iran
{pooya.gohardani, siavash.mehrabi, peyman.ardestani, erfan.jazebnikoo}@gmail.com
http://www.mrl.ir

Abstract. MRL team is among the most award winning teams in rescue simulation league, and its subcategories for several years. With new changes happening in the league and enforcing teams to use ADF for next year's competition, we have decided to participate in this year's technical challenge in order to test the framework and get familiar with it.

Keywords: RoboCup, Rescue Simulation, ADF, Police Force

1 ADF Platform Evaluation

We are going to implement our police force agent in the new framework to apply the standards to our code in addition to test out the flexibility and functionality of the framework and get prepared in case that the usage of ADF becomes mandatory. We should have enough experience and technical knowledge to refactor and migrate our whole code to the new framework.

MRL police utilizes many algorithms and behaviors in order to function well, e.g., partitioning, choosing paths, choosing blockages to clear and etc. the framework provides a good support and structure for implementing and managing those utilities.

2 Implementation of Police Agents

As we mentioned before we are implementing police force agents for technical challenge competition, here we describe some details about our implementation.

2.1 Tactics

In ADF there is a new concept called Tactics that has specific definition for each type of agent. We have implemented the tactics for police force agent in a class called *TacticsPolice*. This class includes initialize, *precompute*, *resume*, *preparate* and think methods that each are responsible for a specific task like decision making, routing, clustering and etc.

2.2 Complex Module

There is an important directory in this section called *targetSelector* which contains implementation for different algorithms of target selection; we have used *BlockadeSelector* in order to select the best blocked road at first to evaluate its performance but then we have refactored our own algorithm to be compatible with the new structure and used our own algorithm.

2.3 Algorithm Module

2.3.1 Path planning

Using a path planner is essential for finding optimum paths in environments like rescue simulation platform. The ADF has provided one implementation for path planning as an example; we have implemented our own cached path planning algorithm according to the provided sample.

2.3.2 Clustering

In scenarios that include vast segments of urban environment running some algorithms are very resource-consuming and inefficient, so we need to use clustering algorithms in order to simplify the problem and be able to run our algorithms more efficiently. We are using a K-means algorithm that exists in *algorithm.clostering* directory and enhanced it to satisfy our needs.

2.4 Action

The police behavior changes based on having a target or not. For example if police does not find any blockade it will start searching the environment until it finds a blockade to clear out. Based on these behaviors the police have two main functionalities, clear and search.

2.4.1 Clear

In case that police agents are aware of existing target blockades, an optimum number of agents will be assigned to those targets based on *ActionExtClear* in extraction directory to clear those blockades, there is a method called *calc* in the class that processes how to remove the blockade; our codes reside in *calc* method.

2.4.2 Search

If a police agent does not find any blocked road in the map it will start the searching task. There is a class called *ActionSearchBlockade* in extraction directory that also includes *calc* method, which contains an optimum search algorithm for police search.