RoboCup 2018 – TDP Rescue Agent Simulation Aura (Iran)

Alireza Kandeh and Arman Absalan

University of Tabriz, Iran
[alirezaknd@yahoo.com, armanaxh@gmail.com]

Abstract. This is Aura Team Description Paper (TDP). This paper is a summery of team efforts on rescue agent simulation studies and is seeking on main module and strategies. One of the most important challenges is Path Planning that we presented a high practical and precise solution. Also, we introduced a new module that helps agents to free from sticking around blockades. Moreover, Fire Simulator and agents main strategies are covered in this paper.

Keywords: RoboCup, Rescue Simulation, Path Planning

1 Introduction

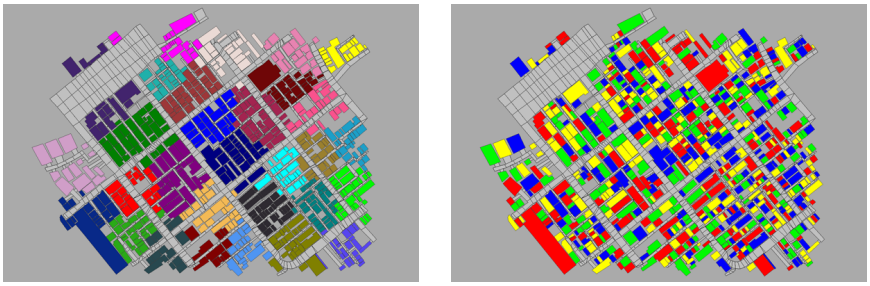
Since Aura is a new team in RoboCup Rescue Simulation (RCRS) league, so we started our research studying on primary problems and gained some reasonable solutions which is described in the following. Clustering, Path Planning, Fire Simulator, Ambulance and Firebrigade primary strategies are the main parts of this paper.

2 Modules

In this section, we describe the features and improvement implemented in the modules used by all agents like clustering, path planning, etc.

2.1 Clustering

We use K-Means algorithm [3] for clustering and hangarian algorithm [4] for assigning agents into clusters optimally. Also, we color each agnet and each building by 4 colors randomly (Figure 1b). When two or more agents are working in the same cluster, the building with the same color as an agent, has higher priority for that agent.



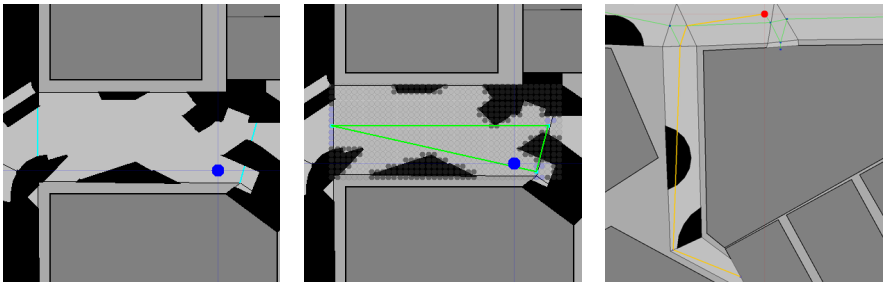
(a) Clustering using K-Means algorithm (b) Coloring buildings and agents with 4 colors randomly

Fig. 1: Clustering

2.2 Path Planning

In order to path planning and finding the shortest path from the agent position to each area, we need a world graph. To construct this graph, first of all, we divide each passable edge into passable segments (Figure 2a). Then, inside each area, considering each passable segment as a node and by using grid based path finding, we calculate the area graph (Figure 2b). After constructing the area graph for each area, we construct the world graph by combining these area graphs.

To find the shortest path from the agent position to each area, first, we find the reachable passable segments inside the agent position area from the agent position. Then considering the agent as a node and adding new edges from the agent node to the reachable passable segment nodes, we use dijkstra's algorithm [1] to find the shortest path (Figure 2c).



(a) Two entrances divided into 3 passable segments (b) Constructing area graph using grid based path finding inside area (c) Shortest path from the agent position to a buidling

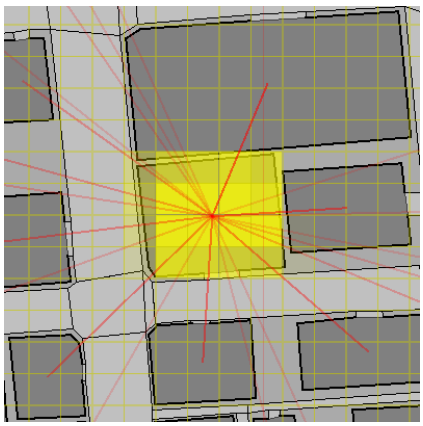
Fig. 2: Path Planning

2.3 Walk Watcher

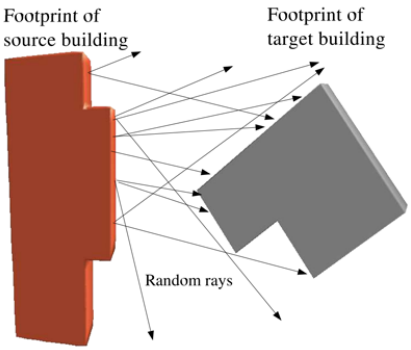
Occasionally, the agent gets stuck among the blockades when moving. We implemented a new module that all move actions are passed throght this module. This module stores the move action history and each cycle when recieves a new move action checks the history. If an stuck is detected, this module replaces the received move action with a new suitable move action to free the agent. To achieve this, when an stuck is detected, this module generates some random points around the agent and then selects the best point to move there with respect to the received move action and movement history. After selecting the best point, that point will be the destination of the new move action.

2.4 Fire Simulator

Currently we are working on a fire simulator for the agent (Figure 3a). Using this fire simulator, which considers fuel consume, energy radiation, heat transfer, wind shift, water cooling, etc. we will be able to predict how and where a fire will grow. Also, we could estimate the minimum amount of water required to extinguish a fire. All algorithms and calculations about the fire simulator are explained in [5].



(a) Amount of transfered energy by radiation from the burning building (red rays), Air cells (yellow grids) and air cells that building transfers heat with them (yellow filled squares)



(b) Randomly emitted rays: The percentage of rays hitting the target building determines the amount of transfered energy by radiation [5]. We do these calculations in precompute phase.

Fig. 3: Fire Simulator

3 Strategies

In this section, we describe the features of the main modules implemented specifically to each type of agent and their main strategies.

3.1 Fire Brigade

Extinguishing To prevent the fire expansion, the most practical method is try to extinguish the fire zones from their borders. To find the border of a fire zone one can use Convex or Concave Hull algorithms.

Each cycle, among the perceptible buildings, the agent calculates the percentage of burning buildings. If this calculated percentage exceeded a threshold, which means probably we are inside a fire zone, the agent tries to reach the border of the fire zone instead of extinguishing.

Using our fire simulator we can estimate the current amount of energy of each building, which means we can estimate the minimum requried water quantity to extinguish the fire. Currently, we have not implemented this idea completely and we are working on it, so the formulas will be presented in the camera-ready version.

Search In firebrigades searching phase, we use the information that is gained from the fire simulator. Buildings that are suspicious to being on fire, have higher priority in search. Each firebrigade tries to travel to its own cluster and check the buildings according to their priority.

In order to check buildings, for each building, we need to calculate the area which has line of sight to that building. These areas are called "perceptible area"s of the buildings (Figure 4). The idea of how these perceptible areas are calculated (Algorithm 1) is as same as the "sensible area" calculation of S.O.S team (2014) [1]. Obviously this method has low precision.

Algorithm 1 Calculating The Perceptible Area of A Building

1: By considering the building cenetr as origin, generate some rays with infinite length (Blue lines in Figure 4).
2: For each ray generated in step 1, find the furthest intersection with self building edges (Red points in Figure 4).
3: For each point calculated in step 2 as origin, generate another ray with length of maxViewDistance and direction of the related ray in step 1 (Yellow lines in Figure 4).
4: For each generated ray in step 3 find the closest intersection with around buildings walls (Magenta points in Figure 4).
5: Sort all points calculated in step 4 around the building center in clockwise order.
6: Construct the perceptible area polygon using sorted points in step 5 as vertices. (Green polygon in Figure 4)

[1] sos.base.util.FireSearchBuilding.setSensibleArea()

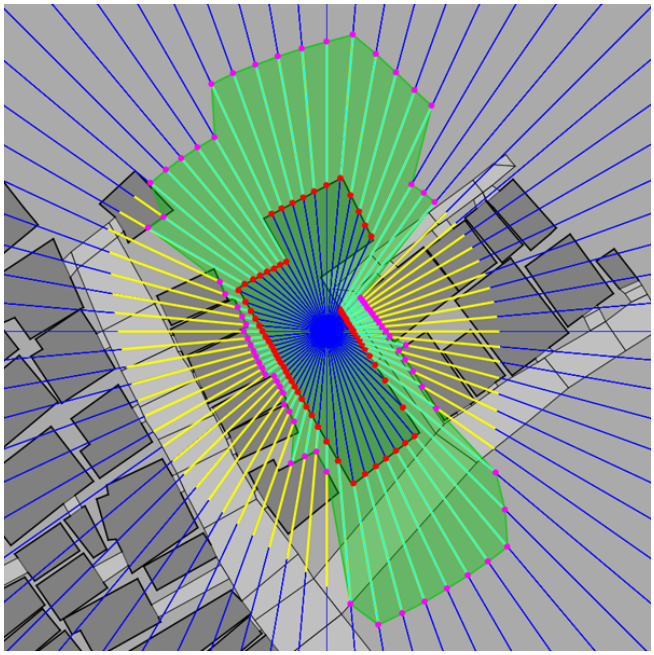


Fig. 4: Perceptible Area of A Building (green polygon)

3.2 Ambulance Team

The task of ambulance agent is to rescue the largest number of civilians in limited time. To make this happen first we must search the civilian and find those which are under the rubbles. After finding the civilian we have to rescue them in the best way possible. In order to implement such ambulance the information below is needed:

- Travel time to the refuge
- Civilian death time
- The fire simulator

To find the Travel Time (the number of cycles that takes to move through a path) between two areas, first, we get the distance of the shortest path between these areas using Path Planning module. Then, knowing the velocity of the agent, we can estimate the time it takes to move through this path. To estimate the death time of the agents and the civilians, we are studying and working on different methods, such as Artificial Neural Network and Particle Filtering, to find a reasonable and efficient solution.

Using the data above we simulate the process of rescuing the civilian and calculate the possibility of rescue and if saving them was possible we do the calculations for rescuing the largest number of civilians during the remaining time.

Ambulance Team Search As mentioned above the first task of the ambulance agent is searching for civilians. In order to search the buildings faster and be safe from the fire we should not enter the buildings; Instead we calculate "Sight Area"s for buildings (Figure 5). From every view point in this area the center of the building is visible to a certain radius. Then we move to a point of the sight area to check inside of the building to find civilians.

In the agent search phase if safe Buildings are observed, they won't be checked since there is no possibility of injured civilians in them. According to the data gained from the fire simulator, if the agent predicts the fire will increase to the areas which civilians are in and therefore rescuing them is not possible in those areas before doing the concerning activities, these areas will be the last priority of the agent.

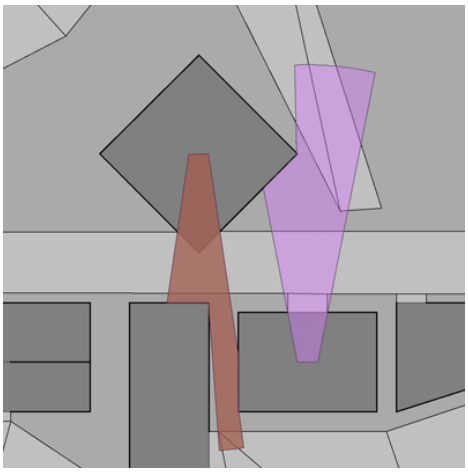


Fig. 5: Sight Area of 2 Buildings

Choosing The Civilian Presume some civilians are in a building; rescuing all of them in all circumstances and in limited time is not possible so we have to rescue the maximum number we can. After calculating the death time and travel time to the refuge we have to find the latest time in that cycle to rescue each civilian.

By knowing the latest time to rescue the civilian, using scheduling algorithms we can assign the civilian to ambulances in a optimal way, to rescue the largest number of civilians in the limit time.

If several Ambulances are working on the same building, they should work in groups in order to rescue more civilians. In this case if Ambulances arrive at the specified building in T=18 and if they do not work in groups for example: A1 rescues C1 and A2 rescues C2, we will only be able to rescue C2 because in T=18 one Ambulance cannot rescue C1. But if both A1 and A2 rescue C1 first and then save C2 after both civilians will be rescued with the team work of the Ambulances.

4 Viewer

In order to accelerate the process of the study and testing the ideas we implemented a viewer (Figure 6).

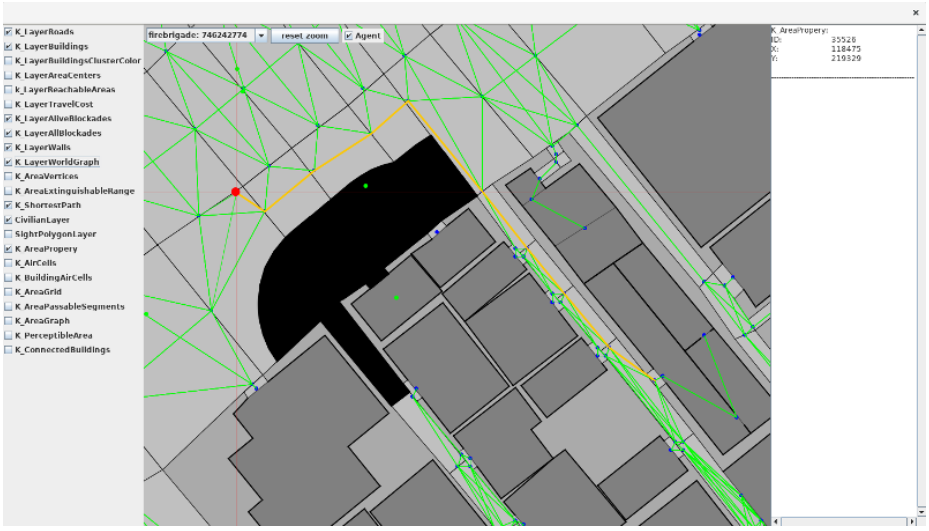


Fig. 6: Viewer

5 Results

The results will be presented in the camera-ready version.

6 Acknowledgement

We thank University of Tabriz for its support of the team, Dr. Farnaz Mahan for their useful comments and Dr. Farshid Faraji for their much needed guide and help for advancing the team goals from behalf of the Aura team members.

References

1. Dijkstra's algorithm. In: Introduction to Algorithms. MIT Press and McGrawHill (2001)
2. Rescue simulation league team description s.o.s (iran). In: Proceedings of RoboCup (2014)
3. G. Sathiya, P.K.: Efficient enhanced k-means approach with improved initial cluster centers. Anna University (2014)
4. Kuhn, H.W.: The hungarian method for the assignment problem. In: Naval Research Logistics Quarterly. Kuhn's original publication (1955)
5. Timo A. Nussle, A.K., Brenner, M.: Approaching urban disaster reality: The resq firesimulator