Collaborative Exploration of Unknown Environments with Teams of Agents

Gustavo Pereira Gama , Sohayl Dhibi
Coimbra University - DEI
Artificial Intelligence
gustavo.p.gama@gmail.com, s.dhibi@studenti.unibs.it

Abstract

In this paper we face up the problem of exploring an unknown environment using a team of agents. We handle the problem under different perspectives, by using an already trained neural network supposing a previous knowledge of the environment's objects or by let the agents try to predict the objects using their previous knowledge of the environment and an interest based approach. With this project we want to improve the original paper "Collaborative exploration of Unknown Environments with Teams of Agents" by Tavares and Gaspar[1] and Macedo et al.[2] by proposing new approaches and by solving some of their solution's limitations.

1 Introduction

The problem of exploring unknown environments is a research field that has a lot of applications, major examples include the exploration of naturally hostile environments not freely accessible by humans (for instance volcanoes, underwater exploration, planets). We can also find more daily applications, for example having a group of robots exploring a collapsed building and helping with the rescue of victims by identifying them or by isolating the higher risk spots can be of paramount importance. A more common task can be the research of lost objects or the mining of new materials.

It is clear that to gather knowledge about the environment under those circumstances it is very helpful to have an efficient approach, and although it can be done by single agents, its effectiveness can be enhanced using multiple agents, significantly decreasing the time needed to map the area. However, we cannot use the same approach in both cases; coordination and sharing is the key to have a powerful mapping system.

Improvements

In the original project[1-2], the focus was mainly directed into the algorithm used by the agents to decide which locations/viewpoints it should explore next, using an "interest" based technique, without however taking much into account the coordination between the agents, for that matter our first goal was on trying to reduce the redundancy in mapping and making sure each agent is directed into the most favorable location by evaluating the frontiers of the shared mapped environment. We also took into account the position of the other explorers, focusing on trying to do not overlap their routes. To eliminate this type of behaviour, we clustered the map and divided the agents between their own belonging regions.

Another problem that we addressed was the fact that the performance of the classification was strongly influenced by its previous knowledge and once it was done, there was no possibility for the agents to reclassify previous objects with the new knowledge acquired. For this purpose, we implemented a reclassification technique that allows the agents to visit spots from which it has not acquired information before and from this knowledge reclassify the object, reducing each time the error.

Another approach that we wanted to implement was to use agents that already have a general knowledge of the objects that they are going to classify, in this case we tried an approach in which the agents use a pre-trained Neural Network to classify the objects. We also modified the maps and created new objects, changing also their features.

Organization

The report will be organized as follows:

- In the second chapter we will discuss the Methods used by over viewing the Multi-Agent System (MAS) and the experimental setup.
- In the third chapter, our focus is on the results of our approaches to the old system, and the discussion around them.
- For the last chapter we conclude our work and bring up some ideas for possible future works.

2 Methods (including exp setup and procedures)

In this project, our aim was to improve the original Multi-Agent System from the works of Tavares and Gaspar[1].

2.1 Overview of the Multi-Agent System

Environment

Like mentioned earlier, the work produced in this project follows the continuity of a previous work[1] with the MASON Toolkit, we kept a similar environment for our experiments in order to make the comparisons easier. The world itself is discrete and is represented by a 2D grid, each cell of the grid can either be 'empty' or contain an object and/or an explorer agent.

The objects of the environment possess different attributes in order to distinguish them by class. An object has a few main attributes: the color and the size (and shape?). The objects present in this environment can have similar attributes, for example a tree object and a bush object have both a greenish color, this will make the classifying task harder for the agents.

Architecture

Each type of architecture has its pros and cons, in our case we decided to keep the original architecture simply based on our first experiments, we considered the Master-slave and centralized approach more appropriate for the task of collaborative exploration of unknown environments. In this architecture, each type of agent has a different purpose, we're going to describe some of the main features of those agents (some of those will be discussed in detail later on). The figure 1 schematizes this architecture. On one hand we have the following two masters:

1. *The Mapper*: his role is to collect the partial maps from all the explorer agents(the ones on the "ground") and merge them together. Here are some of his purposes:
    - Store the objects that the agents have found: their index and their type (an auxiliary array for interest management, so the explorers can focus on new things).
    - Store prototypes (for correlation approach: abstract description values to compare) of the objects, this is used to assign a probability to an unknown object in order to identify it.
    - Keep track of the frontier cells of the global map that need to be explored.
2. *The Broker*: his role is to assign moves to each one of the explorers. Here are some of his purposes:
    - Store the interesting locations provided by the explorers.
    - Send/receive information from explorers and remove objects from memory (once a location is discovered, it becomes less interesting).
    - Split the map into different zones and assign them to the explorer agents. In order to have a good coordination, different techniques are used to manage the explorers agents (we will describe them later).
    - In case there's no interesting points to explore, this agent will guide the explorers into frontier cells in each zone in order to avoid that they wander randomly on the map.

On the other hand, we have *the Explorer Agents* (the slaves), those are the agents that explore the world in order to collect information about the unknown environment in order to map it. This virtual agent is equipped with some features that emulate a real life robot (like a GPS, sensors, cameras). Here are some of the main features of this type of agent:

- Capability to move in the environment (guided by the Broker Agent), and has access to it's own location.
- Observe/detect the surrounding environment with the help of sensors defined by a view range. The explorer agent can also classify the detected objects by using the following techniques:
    - By assigning a probability distribution to the object, based on that probability, an interest score is computed, and depending on a threshold, the agent can either classify it with the highest probability, or mark that object as a point of interest to be examined later.
    - By making use of a Neural Network, the agent will check the attributes of the object and attempt to classify it. This technique requires some assumptions about the environment and the agents that will be discussed in detail later.
- Send personal map to the mapper.

2.2 Experimental Setup

Now that the core of the system is introduced, we can describe how the experiments were conducted during our simulation. The main objective of this type of work in exploration of environments is to have something that scales well in real life applications. So there's always some constraints on how the experiments should be performed, for example we have to fix a certain realistic view range, also there has to be a limited number of agents, in order to reduce costs. In our experiments we tried different values for the different parameters in order to compare them.

Concerning the map format, we reproduced the scenarios introduced in the previous work. We used a 400x300 map with different setups, those setups vary by the object placement, like in the figure 1 and figure 2, we have respectively a random environment and a structured environment (more realistic). Those configurations can be changed in complexity by adding more types of objects. Here's a list of the 8 objects present in the environment: trees, bushes, walls, houses, water, animals, holes and vehicles.

We modified the features of the previous work to let them emulate better a real environment, each object has its own mean and standard deviation for both size and color, and each time an object is instantiated, its parameters will be taken from a normal distribution. In this way we are able to have a more various distribution compared with the previous one; it will be harder for the agents to identify the objects but at least it reflects better a real environment.

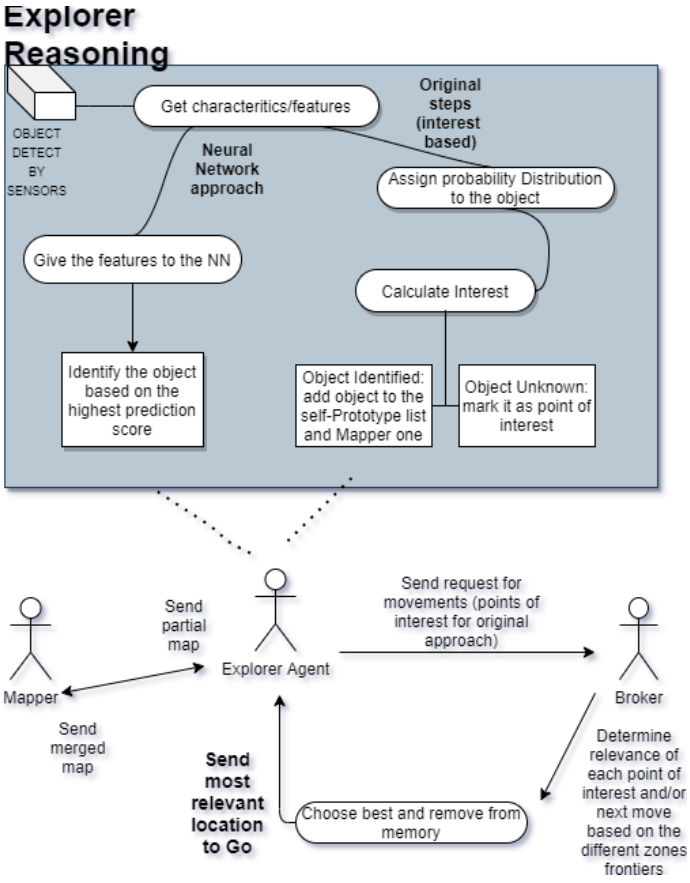

Figure 1: Simplified process of generation of data about the environment in this master/slave architecture (includes original work - surprise based steps and the NN steps)

In the next section we're going to compare the original version of this work with our approach. The criteria of performance will be mainly the number of steps required to identify the objects that are present in the map and the error of classification, we fixed the same limit of steps (5000) as the previous work. We will sometimes present some of our most relevant results with confidence intervals (95) for samples of size 5 (the figure will be representative of one of the samples), due to the fact that there's some randomness in the system (spawn location of the agents is random and some algorithms use random features).

3 Results of the improvements and Discussion

3.1 Improvements on the coordination

Random environment map

As mentioned before, one important aspect of collaborative exploration is to obtain a good coordination between the agents in order to avoid that they explore the same areas simultaneously. This was one of the main issues of the previous work, sometimes the agents ended up roaming randomly on the map when there were no points of interest to be explored, so they ended up overlapping their routes. Also, because of the randomness used during the exploration, often there was

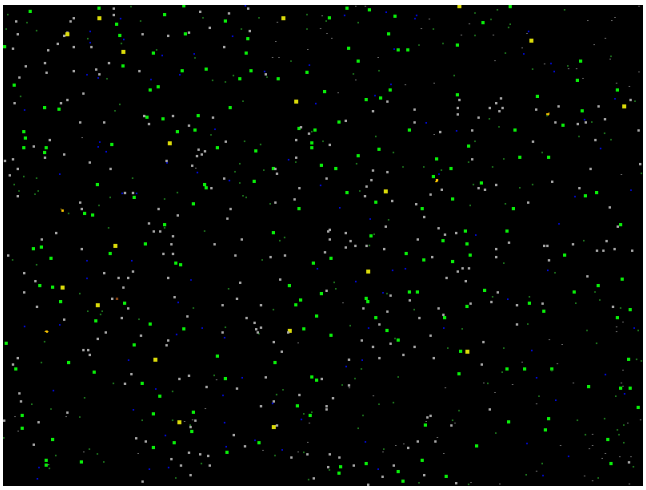

Figure 2: Random map with a few extra objects (there's also a version with more objects).

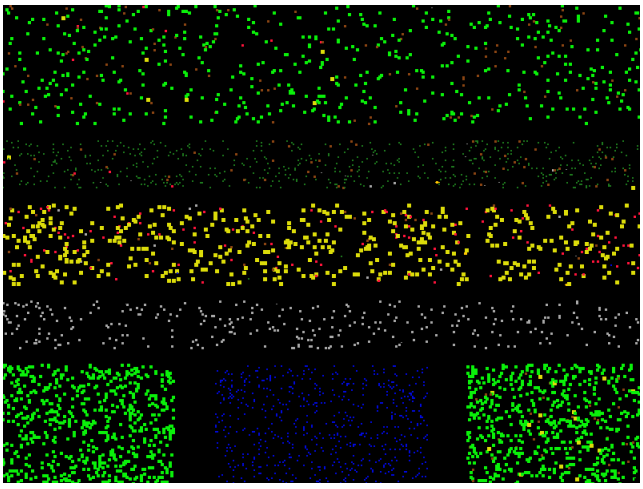

Figure 3: Structured map with a few extra objects.

a portion of the map that wasn't fully explored, specially the corners of the map.

To address those issues, we improved the Broker Agent by giving him more tasks. We added a feature that partitions the map into different smaller zones, the number of zone splits n can be defined at the initialization, usually this number depends of the map size and on the number of agents. Once the map is divided into multiple rectangles, the Broker Agent has the job of keeping track of points of interest like the previous implementation, but also to keep track of the zones and assign them to the explorer agents.

When an explorer agent is assigned to an unexplored zone, he has to fully explore this zone before switching to another one, i.e the explorer agent has to explore all the points of interest of the assigned portion of the map but also, when there are no more points of interest left, the agent has to cover the integrity of the zone by exploring the frontier cells of the zone, the frontier cell chosen is the one that minimizes the distance between the explorer agent and the frontier cell.

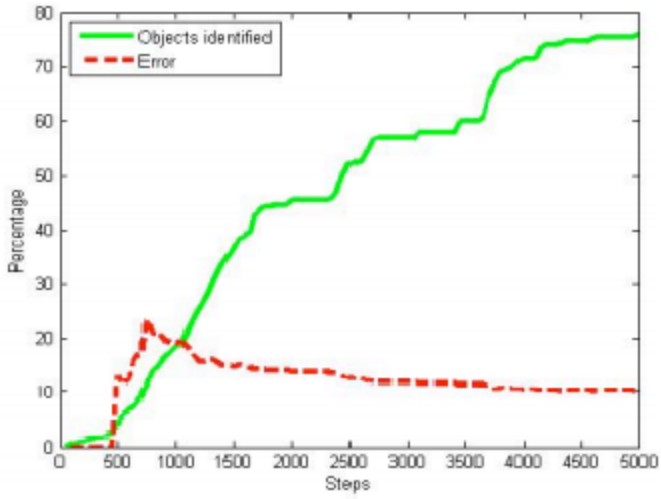

Figure 4: Random map with 1 agent (threshold of 65). Figure extracted from original work[1].

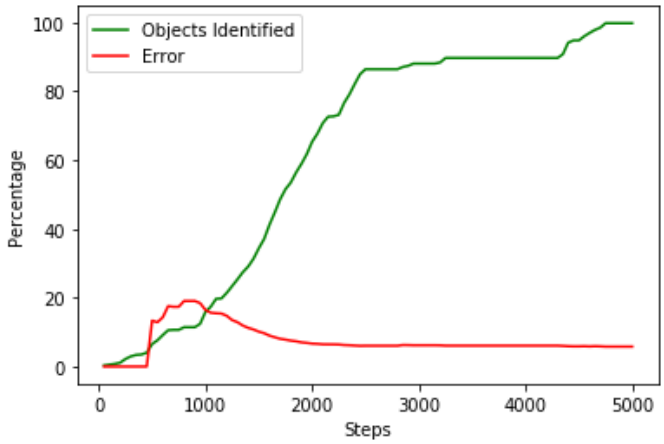

Figure 5: Random map with 1 agent (threshold of 65). Performance of the new Broker System (5 clustering zones with a random switch) with the old classification system.

When those conditions are fulfilled, there are different possibilities to perform the switch (chosen at initialization):

- Randomly switching to another zone of the map.
- Switching based on the remaining partitions to be explored and distance minimization based (Euclidean distance).

We used in a first approach this random switch mode to compare the impact of dividing the explorer agents in the different clusters, as we can see in fig. 4 and fig. 5, there's already a clear improvement concerning the coverage of the map, in our version, the map is covered in its integrity (or almost) before arriving at 5000 steps. With the usage of frontier exploration within the zones we don't have the problem of missing the corners. By using multiple agents, as expected, the coverage of the map is faster (figure 6-7).

The second mode for the switch checks in priority which zones are unexplored and assigns the nearest one based on the current location of the explorer agent, it computes the distance between the location of the agent and the center of the cluster, and minimizes it. Once the maps is fully covered, this mode prioritizes the zones where there's less agents already exploring (this is useful for later reclassification). In the figure 8 we can see the performance using 5 clustering zones with this second mode, the performance actually was pretty similar with the Random switch, but as we increase the number of clusters, this switch mode seems to perform better. With a lower number of cluster zones, the explorer agents still overlap their exploration, but with more clusters it becomes more organized and performs better(figure 9).

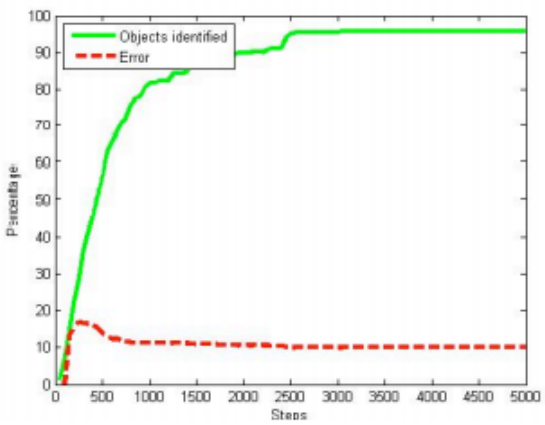

Figure 6: Random map with 5 agents (threshold of 65). Figure extracted from original work[1].

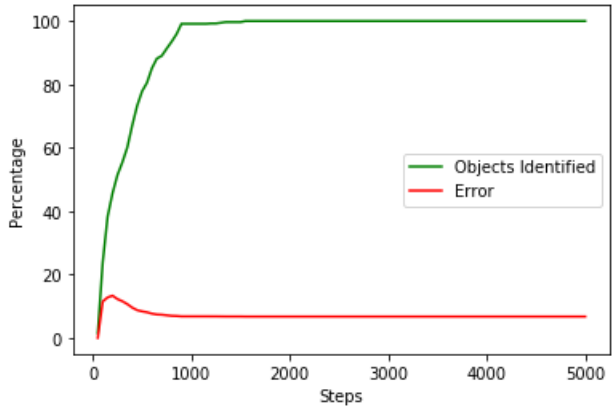

Figure 7: Random map with 5 agents (threshold of 65) - Performance of the new Broker System (5 clustering zones with a random switch) with the old classification system.

New structured environment map

As mentioned before, this structured environment is more realistic than the random one, we also modified it by adding more objects and changing their locations, it now has 8 possible objects with their parameters following a normal distribu-

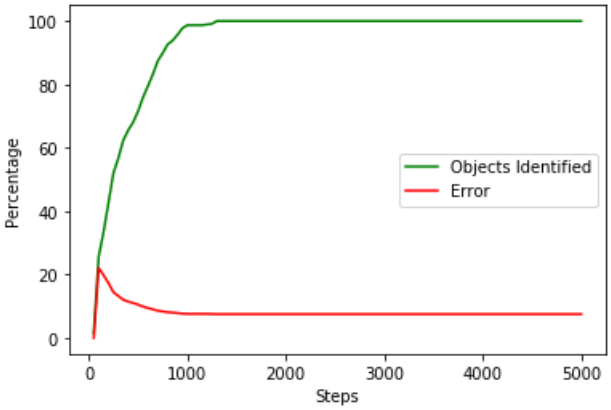

Figure 8: Random map with 5 agents (threshold of 65) - Performance of the new Broker System (5 clustering zones with a remaining zone switching) with the old classification system. Mean of steps to fully cover the map (5 samples): 1230 +/-280.

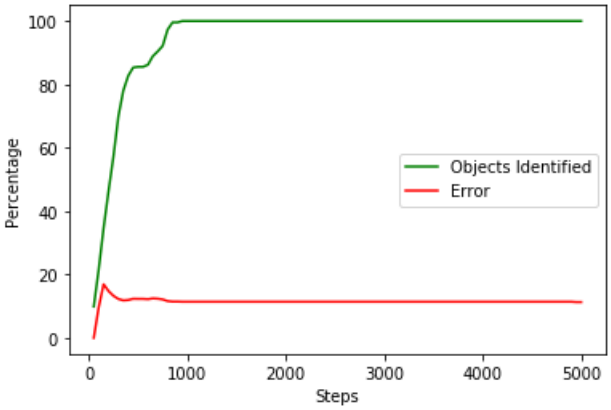

Figure 9: Random map with 5 agents (threshold of 65) - Performance of the new Broker System (8 clustering zones with a remaining zone switching) with the old classification system. Mean of steps to fully cover the map (5 samples): 840 +/-141.

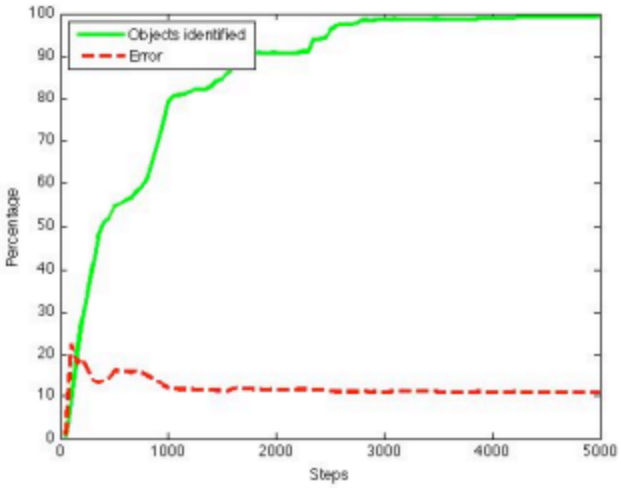

Figure 10: Structured map with 5 agents (threshold of 65). Figure extracted from original work[1].

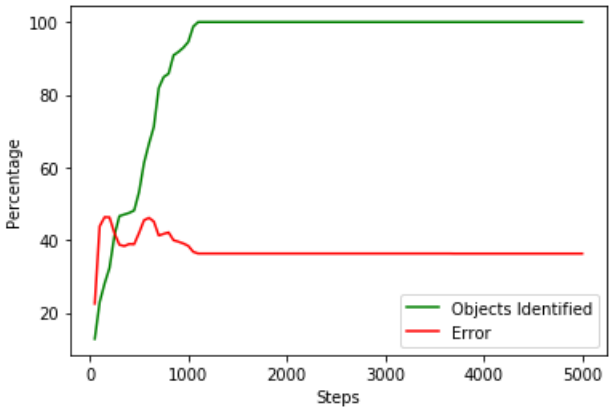

Figure 11: Structured map with 5 agents (threshold of 65) - Performance of the new Broker System (5 clustering zones with a remaining zone switching) with the old classification system. Mean of steps to fully cover the map (5 samples): 1580 +/-1000. Mean error: 30 +/- 20

tion. This made the classification with the original approach much harder. As we can see in figures 10 (original work for structured map) and 11 (new one), the explorer agents have a much harder time to classify the objects in this new environment with a bigger variety of objects and possibilities. The percentage of error in the original work is around 10 percent, but in our new environment it can vary a lot as we can see in the confidence interval: 30 +/- 20. This also seems to impact severely the coverage of the map.

To sum up this subsection, we can say that our system outperforms the original one in terms of map coverage steps for all maps. But as we started increasing the complexity of the environment with more objects, the original classification system started under performing. In the next sections we will describe some of the approaches we took in order to increase the classification efficiency.

3.2 Reclassification with the original system

Now that we have a system that explores the map with more efficiently and with more coordination, we often end up with a lot of steps left to make actions (based on our fixed 5000 steps limit). We decided to use the steps left by revisiting the already explored zones and reclassify the objects by lowering the threshold. By lowering the threshold value, it will reduce the map coverage speed (but since we already finished covering the map it doesn't really matter), but increase the focus that each explorer agent gives to each object. In theory this should reduce the classification error. We experimented this approach with different sizes of teams of agents, in the figures 12 and 13 we can find some of the results with 5 and 10 agents respectively. We can see that there's some improvement in classification by reclassifying but it comes with the cost of increasing the number of agents considerably.

Also this approach isn't always effective, it depends a lot on the location where the agents stopped covering the map, they will select a location on the zone that they ended up, and if it's already well classified, they will just waste time. We tried an approach of keeping track of the locations where they didn't find any point of interest and they just simply classified the zone with the data of the prototypes that they gathered during exploration, but the results didn't really changed as we wanted.

3.3 Classification with a Neural Network (NN)

To cover the problem of classification performance, and to avoid having to increase the number of explorer agents, we decided to try to add a different classification system, in the works of Rehfeldt and Hilliaho[3], they used a unsupervised approach, a KNN algorithm. In our case, we decided to implement a Neural Network in this Multi-Agent System. Our first approach was to use a specific Deep Learning library, but to avoid any dependencies on libraries we decided to use a much simpler network, we used a Feed-Forward Network (of course in a real life application, the NN is going to be more complex, most likely making usage of CNN for to analyze the images that the robot obtains from the environment).

It's important to note that the usage of this NN slightly changes the problem of exploration, in this case we're not starting the exploration from "scratch". We're make the assumption that the explorer agents have beforehand a database containing examples of objects from the environment, and they predict the class of the objects based on this database. This kind of scenario is reasonable, in the case of exploration of unknown environments when we're detecting/looking for specific objects, for example looking for survivors in a hazardous place (the agent needs to know what a body looks like).

As for the creation of the Neural Network, we have been helped from the work of Suyash Sonawane [4], using its classes to handle the matrices and the Neural Network training algorithm itself. for our purposes we used a Feed Forward Neural Network with three layers:

- For the first layers (the input layer) we use 4 neurons, one for each feature of the object, three for the Red Green and Blue values of the colors and one for the size.

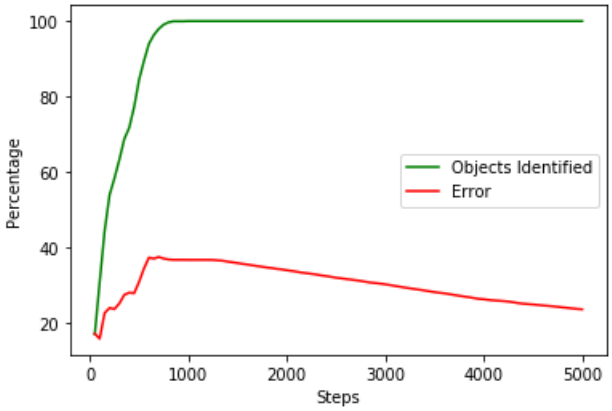

Figure 12: Structured map with 5 agents (threshold of 65), 8 clusters. Test on reclassification.

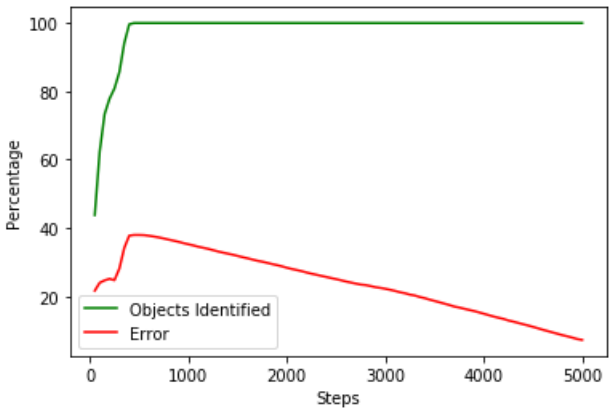

Figure 13: Structured map with 10 agents (threshold of 65), 8 clusters. Test on reclassification.

To make a comparison we also added a version in which we used 5 neurons, adding a neuron for another feature that represents the shape of the objects.

- The second layer is the hidden layer and it is the one that will have more impact on the classification of the objects; we used a layer that has 100 neurons.
- The last layer (the output layer) has a number of neurons equal to the number of objects that we are using (the first for the first object, the second for the second object and so on). To decide how the agent is going to classify the object, we are going to pick the output that has the greater value.

As for the learning algorithm we used the backpropagation algorithm, while the activation functions of the neurons will be sigmoids. As for the training we feed the network with the features of 100 objects of each type, and train the network (applying the backpropagation algorithm) for a variable number of time (epochs).

As the results show this type of classification can be very effective with respect to the other ones; since it has already in its memory the types of objects that is going to find, it is able to classify them quite easily, without the need of see them directly. If fed for the right numbers of epochs, it is able to classify almost every object correctly (sometimes with a percentage of error below 0.5%).

However, with this method we noticed that there are some limitations; although the average run is well performing, there is a high probability of outliers with a high error, if the network it is fed for a low number of epochs. This happens because if the weight of the neurons are misplaced and for some reason they classify an object with another (this happens more frequently with similar objects, for example bush and trees), it will misclassify almost every instance without the possibility of understanding its mistakes, without learning from the environment. Another problem rely on the possibility of over fitting with a high number of epochs. In those cases, it performs really well with a specific types of objects, but it will fall into confusion if we add another type in the environment. So for this types of network it is a major to choose the right number of epochs to train the network. In our environment with 8 objects, a training with 10000 epochs seem the to perform quite well without falling into overfitting. But this increase in performance requires extra computational power, because of the earlier training phase for the agents.

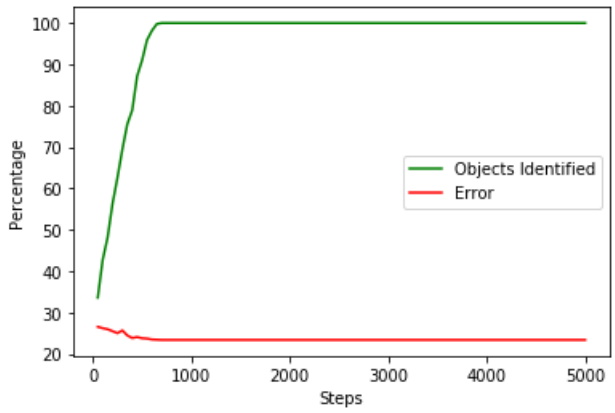

Figure 14: Structured map with 5 agents (threshold of 65), 8 clusters. Neural Network 2000 epochs. Mean steps to fully cover the map: 740.0 +/-142. Mean error: 17.2 +/- 10.3

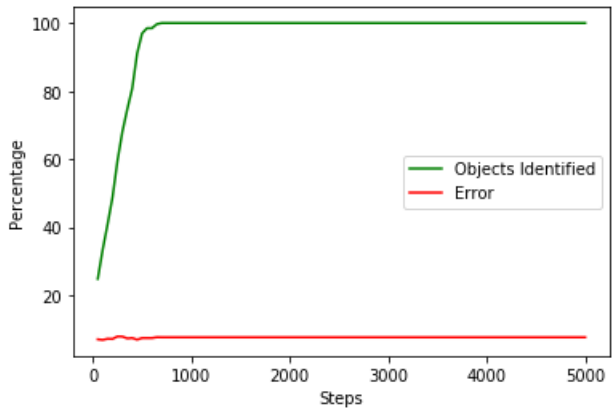

Figure 15: Structured map with 5 agents (threshold of 65), 8 clusters. Neural Network 10000 epochs. Mean steps to fully cover the map: 680.0 +/-185. Mean error: 6.3 +/- 5.7

4 Conclusions and Future work

Conclusions

In conclusion, we tested in this work some different approaches in order to improve the original Master-Slave system. Our first objective was the improve the coordination between the agents by partitioning the map into smaller zones and assigning each of the explorer agents to one of the zones to explore, this resulted in an improved coverage of the map in comparison to the original approach.

Our second main objective was to test new approaches to the classification problem (and makeup for the increasing complexity of the environment), we tried to optimize the time

given to the agents to explore the environment by creating a phase of reclassification where the agents give more interest to each object. This resulted in a reduction of the classification error for our new environment, but it required more resources (more agents).

We also applied a Neural Network approach to this exploration problem, by sensing the objects and their attributes of the surrounding environment, the agent now has the option to predict the objects class via a NN, this resulted in a very good performance both in classification and coverage of the map. Of course this approach requires some assumptions about the environment and require a database, which aren't always available.

Future works

As for a future work on this field, there are a few ideas that we can suggest:

- Concerning the partition of the agents around the different zones, a better approach could be to reduce the utility of the adjacent zones of an agent, so the others don't come in that direction, to avoid multiple agents explore zones that are adjacent and leave it to a single agent.
- The original classification of the agents rely strongly on the first prototypes visited. If a new type of prototype is visited later into the run it will have a lot less weight and impact on the classification of the objects.
- As for the reclassification algorithm, we used a very brute approach, trying to reclassify, basing on the new knowledge acquired, all the objects, right or wrong. An improvement that can be made to improve the reclassification is to try to reclassify the objects relying on degree of certainty that it is the right object.
- As for the neural network algorithm, we used a very simple one, focusing more on a qualitative comparison with the other approaches rather than focusing on the type of neural network. For this purpose, a lot of Neural Networks can be tried and compared between each other. Another limitation that should be overcame is the fact that the NN does not improve itself during the run.
- More objects and more differentiate maps can be tried. Also the objects should be identified with more features, we tried to add the shape as a parameter, but it was a primitive approach and the shape was only identified with an integer.
- Make the pathfinder A* (we attempted to make this on the system but it worked half way through only, at some point it provoked some issues).

5 References

1. *"The Exploration of Unknown Environments by Affective Agents"* by Luís Miguel Machado Lopes Macedo (2006).
2. *"Uncertainty and Novelty-Based Selective Attention in the Collaborative Exploration of Unknown Environments"* by Luis Macedo, Miguel Tavares, Pedro Gaspar, and Amilcar Cardoso.
3. *"Collaborative Exploration of Unknown Environments with Teams of Heterogeneous Agents"* by Jani Hilliaho and Sebastian Rehfeldt.
4. *https://towardsdatascience.com/understanding-and-implementing-neural-networks-in-java-from-scratch-61421bb6352c* by Suyash Sonawane.

Methods	**Percentage Objects identified (and the number of steps if reached max.)**	**Conf. Interval Objects identified**	**Percentage Error**	**Conf. Interval Error**
Original version[1-2]	RM: ∼98% (5000 steps) SM: ∼98 %(5000 steps)	-	RM:∼10 SM:∼10	-
Original Version with boosted coordination (cluster zones)	RM: ∼100% (at 840 steps) SM: ∼100% (at 870 steps)	+/- 141.0	RM:∼14	+/- 13.0
Original Version with boosted coordination + reclassification	SM(5agts): 100% (at 800 steps) SM(10agts): 100%(at 600 steps)	-	SM(5agts): ∼25 SM(10agts):∼5	-
New Version with NN (2000epochs)	RM: 100% (at 740 steps)	+/- 142	RM:∼17.0	+/- 10.1
New Version with NN(10000 epochs)	RM: 100% (at 680 steps)	+/- 185	RM:∼6.3	+/- 5.73
New Version with NN(10000 epochs) with shape attribute for objects	SM: 100% (at 610 steps)	+/- 155	SM:∼4.2	+/- 3.7

Recapitulation table of some of the approaches. RM stands for Random Map / SM stands for Structured Map (the one with more objects).