Collaborative Exploration of Unknown Environments with Teams of Agents

Miguel Gonçalves Tavares, Pedro Miguel Francisco Gaspar
{mgt,pgaspar}@student.dei.uc.pt

ABSTRACT

In this paper, we approach the problem of exploring an unknown environment using a Multi-Agent system – MAS – with the goal of providing the agents with a measure of interest for the surrounding environment, that allows them to filter which locations should be visited or not. We present a 3-parted agent architecture with a brokering system that allows the coordination of the agent team and propose an interest classification system for the explorers based on information of probability distributions calculated using the agents' previous knowledge. We test this theory in several scenarios, comparing it against usual approaches, time-wise and accuracy-wise.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Coherence and coordination. Intelligent Agents, Multiagent systems.*

General Terms

Algorithms, Measurement, Performance, Experimentation, Theory.

Keywords

Exploration, MAS, Multiagent Coordination, Interest Classification, Predictive Mapping, Mapping, Unknown Environments, Agent Behaviour, Artificial Intelligence.

1. INTRODUCTION

The research done in this field is currently being applied in various situations which require the automated exploration of unknown environments by robots in conditions in which the presence of human beings is either dangerous or impossible - two major examples are the mapping of hostile areas and threats in war zones and the exploration of other planets, like the current NASA Rover missions in Mars.

Although this exploration can be done using single agents, its effectiveness can be quite hampered in large or complex environments where one agent takes significantly more time to map the entire area. In these cases, a Multi Agent approach can be used and, by a sheer reason of numbers, reduce the mapping time. However, the algorithms and strategies used by these agents to navigate cannot be the same as those used by a single agent. Having other agents mapping the same environment, these agents must take into account the behavior of its fellow "explorers" and the locations that have already been mapped by them when deciding the next step to make, in order to reduce the redundancy in the mapping and reduce the time taken in the process. Therefore, some sort of coordination is needed in order achieve a truly collaborative behavior between the agents.

In this approach, we focused mainly on the algorithm used by the agents to decide which locations it should explore next, using an approach using affective agents. The coordination aspects of the problem are dealt with solutions found in recent research in the field. Our approach consisted of providing the exploring agents with reasoning capabilities that allowed them to rate unknown objects in the environment according to an interest level determined by the explorer. By making the agents "ignore" objects with a low interest value and make a predictive identification of such an object at a range instead of approaching the object to fully identify it, we aim to reduce mapping times of full environments, with a degree of error, of course, aiming for a "partial" fully fledged identification of the environment, and by predicting what the rest of the environment is like, using previous knowledge.

1.1 State of the Art

The robotic mapping problem is a long lasting one, as it is described in detail in [1]. For this work, however, we are more interested in the coordination and behavior aspects of a Multi Agent approach and the use of Affective agents in the process. On this aspect, the work in [2] and [3], regarding the coordination of the agents, yields significant results by reducing the redundancy in the mapping and directing each agent to the location most favorable to be explored by him. This is achieved by evaluating the "frontiers" of the currently mapped environment and having each agent bid on the next location to explore until a consensus is reached between all agents, in which no location is picked by more than one agent and each agent gets the most favorable location to explore, given the existence of the other agents.

As far as the use of Affective agents in exploration goes, the work of Macedo and Cardoso [4] proposes the use of motivational agents with various "feelings" - surprise, curiosity and hunger - in the decision-making process that defines the exploration behavior of the agents. However, in all these approaches, the goal is always to map an entire environment, with the (mostly temporal) costs associated with such an exhaustive method. Also, these works never take into account the cost of fully identifying an object – in a real-life situation, where object identification must be done, usually it is costly to have a clear classification of an object based on its characteristics – it is necessary to query large amounts of data or to prompt for input from a human being, both of which are extremely time-consuming.

In this work we propose a slightly different approach that aims to reduce those costs at the expense of a partially predictive mapping, thus reducing the exploration time of our agents by cutting the time necessary to identify and object, but, hopefully, not sacrificing too much information – which can happen in a predictive mapping, with wrong object classifications – which, if possible, is a great improvement in the field.

2. METHODOLOGIES

In this work, we propose a simple approach to the exploration problem (mainly due to the time-frame limitations we had) in which we aim to create a MAS that is able to explore a simulated environment and map it, using a surprise-based approach that tries to ensure that "common" locations (i.e. locations that are highly unlikely to possess new/useful information for the mapping, such as empty space or locations with a known geometry) won't be given as much attention by the exploring agents, resulting in a partial mapping of the environment, but one that holds the most relevant information about it (i.e. the most "interesting" locations) and that is done in a considerably shorter time-frame, making for a quicker exploration.

To that purpose, each agent gains knowledge of the environment and alters its internal state as it goes. With that knowledge, the agent, when mapping a new section of the environment, is able to be "surprised", or not, by the new discoveries, depending on whether it was expecting the new location to be as such or not. This surprise factor is then computed into the agent's decision-making algorithm, increasing the importance of the most "surprising" locations - and discarding locations with no interest whatsoever (based on a pre-determined threshold) - that possibly hold the most relevant information. By using this with coordination algorithms such as those presented earlier, we are able to produce a coordinated effect between the agents in which the exploration of "common" locations by multiple agents can be diverted to a more "interesting" location discovered by a single agent, leaving the uninteresting locations unexplored.

2.1 System Architecture

In order to efficiently map the environment, we chose a master-slave architecture. Mapping and exploration are coordinated by two separate agents with no physical presence: the Mapper and the Broker. The slave agents are called Explorers.

- The Mapper is in charge of merging everyone's maps and sending the global map back to each Explorer.
- The Broker assigns next moves to every Explorer, based on the interesting locations they spotted.
- The Explorers analyze the environment they're in, send their local map to the Mapper, pick points of interest based on their current knowledge of the area, send them to the Broker, and finally move to the location assigned by the Broker.

In our simulated environment, there are a set of properties Explorers have access to, including its position, and the list of objects it senses. While a physical agent requires a set of sensors in order to gather information about the surrounding world, our simulated Explorer has access to some properties of the objects he can see. The simulated agent can see objects in all directions within a certain distance (which usually has a very small value).

When it comes to physical agents, the Explorer would have to analyze a real environment and, for that reason, would have to have some sensors attached to it, in order to gather information about its surroundings. First of all, this agent would have to know its absolute position, so the Mapper could be able to merge all the local maps. This can be achieved with a GPS unit. Finally, the agent would need to get information about the environment it's in. A vision-enabled sensor could be used with this purpose.

2.1.1 Environment

The environment used, to simplify the problem, is a discrete, two dimensional grid, where each cell can be populated with one object, no objects or an object and an exploring agent at once. This is done because in the classification of the environment, each cell is marked as identified or not, where in the former case the class of the object identified is placed in an auxiliary grid. If multiple objects where present on the same spot, this task would be strenuous, at best, to achieve successfully.

2.1.2 Objects

Each object that populates the environment has a set of core attributes – in this case, *size* and *color* were chosen for the sake of simplicity and in order to keep the experiments simple and understandable. Apart from this, each object has a class, that identifies it as a member of a kind of object family – for example, in a nature setting, there could be classes for trees, deer, bushes, grass and rabbits – all these would have a size and color (that can distinguish them from other classes or not – bushes and grass have similar colors) and its class would be the means to identify such objects.

These two core attributes can be sensed by the agents at a distance and are what determine the probability distribution assigned by an agent to an unknown object.

2.2 Agent Architecture

In this section, we describe the various agents implemented and their in-depth functionalities.

2.2.1 Mapper Agent

The mapper agent, as stated before, has the task of collecting all the data from the environment that the explorer agents supply and aggregating that data into a map of the environment. To that purpose, the agent has in its memory a Sparse Object Grid, where it stores the objects that all the agents have seen so far and its location (by the index in the grid) and an auxiliary array that contains information whether a given object at a given location has been identified or if its type is still unknown.

This last array is meant to keep the explorers from extracting interest from objects that have already been identified, allowing them to skip through fully identified locations into more interesting areas.

The mapper, in one implementation of the MAS, also stores a list of *Prototypes* – an abstract representation of the different objects witnessed by the explorers so far, with information about the average characteristics of such an object (a kind of *idea* of an object, something that we can compare to an actual object and obtain a high correlation). These *Prototypes* are to be used by the explorer agents when they wish to assign a probability distribution to an unknown object. This is explored in depth in the section about the explorer agent.

2.2.2 Broker Agent

The broker agent acts, as the name indicates, as a broker for all the explorer's requests for new targets to explore. As such, it must maintain a list of interesting locations provided by the explorers, so that it can select from that list when an explorer agent requests a new target. This approach is highly influenced by the work of Simmons in [3], being a simplified version of its brokering system.

To implement this behavior, the broker has methods to receive information from the explorers, namely the location of an interest point and its corresponding interest, and to remove points from its memory – for example, when an explorer arrives at the point and identifies the object there, no longer that point is interesting.

The brokering itself takes place when an explorer agent requests a new target from the broker. Upon this request, the broker will determine the relevance of each interest point in its memory to the given agent. This relevance is a Benefit minus Cost function that relates to the interest of the point, the distance of the agent to the point and *Max_dist*, that is, in a way, the maximum distance that an agent is willing to travel before arriving at its target. Eq. 1 represents this relevance function.

$$Relevance = Interest - \frac{Dist \times 100}{Max_{dist}}$$

Eq.1 – Broker Relevance function

This formula assures us that an agent will pursue the most interesting points that are closer to him at the location of the request. This can, of course, be weighed, so that the influence of the interest is of utmost importance to the agent, or vice-versa.

The interest points are sorted by decreasing relevance, and the most relevant interest point is passed to the requesting explorer as its next target. This point is then removed from the list, so we can prevent situations where the broker assigns the same target to multiple agents, effectively disrupting the entire agent coordination.

2.2.3 Explorer Agent

The explorer agent is the core of this MAS. It performs the heavy-duty work of travelling around the environment, collecting information about the objects it detects and trying to extract knowledge from it. To do so, it has movement and sensing capabilities, represented by a *speed* and a *view_range,* inside which the agent is capable of detecting general characteristics of an object. In each simulation step, the agent moves towards a given target – a location on the environment – sensing its surroundings. Whenever and object in the environment comes inside its *view_range*, the agent starts a process to attain its characteristics, assign a probability distribution to that object and, based on that probability, calculate the interest of that object to the agent, which will determine what to do next: either the agent identifies it using the probability distribution, or it marks it as an interesting spot to be visited later on, and identified on location.

Note that these are different ways to identify an object: if the agent arrives at its target and there is an object there, he spends some time on it, identifying it (emulating the behavior of a real robot sending information to a central processing unit where either a large database is queried or human input is required to identify an object). In this case, the agent adds the object description to the prototype of the identified class, because he has an exact knowledge that the current object is of that class. However, the agent also possesses capabilities to identify an object at a range: if, while analyzing it, the explorer determines that an object is not interesting enough to be visited, he rather chooses to identify it at range, by picking the most likely class from the distribution. This approach does not add the object attributes to the class prototype, because the agent cannot be sure that he is making a correct classification. Thus, this prevents the agent from reinforcing a bad classification, leading to disastrous results.

To attain the object's characteristics, the agent simply queries the environment as to what are the basic attributes of the object: size and color. This emulates a real-world scenario where an agent has a limited sight and is only capable of identifying some basic attributes of objects in the environment at a distance.

2.2.3.1 Classifying objects

Afterwards, the explorer agent has the task of classifying this unknown objects, by assigning it a probability distribution based on the knowledge of the worlds it has so far – namely the *Prototypes* that he has developed. These are abstract representations of an object (more precisely, a class), which contain the average characteristics of the witnessed objects of that class – a good example is the *idea* of Tree: it's tall and it's green-ish. Not all trees are really tall or green, but, on average, this is an accurate representation of trees.

The explorer agent will browse through a list of prototypes – either its own list or a collaboratively filled list that all the agents share in the mapper agent – and correlate the attributes of the unknown object to each of the prototypes. This correlation is done by doing a weighed average of the Euclidean distance between each attribute of the prototype and the object. This gives us a good approximation of the probability for the object to be an instance of each of the prototypes.

However, this cannot be the only measure we use. If it were, by witnessing one tree, the agent would assume to possess all knowledge of trees that existed in the world, and that is not the case. This representation must take in the fact that only after several observations of the same object, some understanding of it can be extrapolated. As such, a saturation factor (Eq. 2) is applied to the correlation to take into account the number of occurrences of that object that have been witnessed.

$$saturated_{corr} = \tanh\left[\frac{(n_{Occurrs} - 5)}{2.0} + 1.0\right] / 2.0$$

Eq.2 – Saturation function

This formula was designed to saturate at near 1.0 when the number of occurrences of that given object approaches 10. So, we are assuming that the agent needs to see at least 10 instances of a given object to have any real understanding of the object as a class. This can be changed, of course, to represent different levels of learning rates, by increasing or decreasing the denominator of the fraction inside the hyperbolic tangent.

All these calculations, however, don't account for the possibility that the agent might not have witnessed any instance of the class of the object it is now trying to identify – it might be a new kind of object that needs its own prototype. This is addressed by the agent by calculating an *unknown_correlation*, a value that determines what the possibility is that the current object is "none of the above". This value is automatically calculated to 1.0 when no prototypes are present (at the beginning of the simulation) and using the following formula (Eq. 3) when there is already some knowledge of the world.

$$unknown_{corr} = \sum_{i=1}^{N} \frac{1 - corr(i)}{N}$$

Eq.2 – Unknown correlation

Where N is the number of *Prototypes* known and *corr(i)* represents the correlation of the object to the i^{th} prototype. This gives us a fair measure of, given the knowledge we have, what are the chances that this object is something new. This value is maximum when all the correlations are 0.0, and minimum when all the correlations are 1.0 – note that this is an unlikely scenario, an object that fits all of the descriptions, and will be addressed later on.

Finally, the explorer agent translates these correlations into a probability distribution, using the following formula.

$$p(j) = \frac{corr(j)}{\sum_{i=1}^{N} corr(i)}$$

Eq.3 – Probability distribution

The explorer agent now has a probability distribution, which sums to 1.0, that tells the agent, for each known class, what the probability is that the current object is an instance of that class. It also has an unknown probability, to account for the possibility that the object is something new.

After some experimenting we altered the formula above so that it uses the third power of the correlations. This has increased our success rate when distinguishing similar objects, since any slight correlation difference greatly increases the impact of that object over the others.

2.2.3.2 Determining Interest

Given a probability distribution, the explorer agent assigns an interest to such a probability distribution. This is done in two steps, that represent two different interesting situations: one where the agent sees there is a high probability of finding a new object, and one where the agent detects that the current object has similar characteristics to several known classes.

The first case – the interest for the unknown – is mapped using only the probability that the object is none of the known classes. The interest is given by the Eq.4.

$$unkn_{interest} = \tanh(2.0 \times p(unkn))$$

Eq.4 – Interest for the unknown

This formula is used instead of simply mapping probability to interest because a 50% probability of being something new has more than an interest of 50 in a scale of [0-100]. This formula increases the rate of climb of the interest, saturating when approaching 1.0, the maximum probability, representing the maximum interest, here valued at 1.0.

However, this is not the only situation where the agent discerns something interesting. It also needs to identify an interest for the chaotic – an object that matches several of the known classes well, or at least some of them. In this case, we have a simple solution – the explorer agent calculates the entropy of the probability distribution. Higher entropy values represent more chaotic situations, where all the interest of the agent should be focused. As such, we calculate the entropy (Eq. 5) of the probability distribution for that object, and map that value proportionally to an interest value.

$$chaotic_{interest} = -\sum_{i=1}^{N} p(i) \times \log(p(i))$$

Eq.5 – Interest for the chaotic

After calculating these two interest values, the explorer agent picks the one that represents the maximum interest and assigns it as the interest value for the object under scrutiny.

2.2.3.3 Agent Reasoning

After determining the interest for the current object, the explorer agents makes a decision: either he determines that the object has an high enough interest and sends its location to the broker agent so it can be identified as an interesting position; or it determines that the object is not interesting enough to be visited and classifies it at a range. This last option is the result we expect from our agent after exploring a bit of the map, seeing that it is less time consuming, does not require the agent to be in the same exact spot as the object as allows it, in a single step, to identify several objects in its *view_range.* In order to make this decision, each explorer agent has an *INTEREST_THRESHOLD,* under which he identifies the object at range and over which he sends the location to the broker, along with its interest.

The experimental tests that follow will serve as a benchmark to see whether this approach is favorable to more classical approaches, and how much error this behavior introduces in the mapping of the environment. In the following sections we will present our experimental setups

3. EXPERIMENTAL SETUP

To test our new approach, we will try out a team of explorer agents in several different scenarios (maps) with different configurations to test out several aspects of our system. Figs. 1, 2 and 3 represent the configuration of the 3 different test environments we used: an environment where the position of the objects is random, a donut-environment, whit a cluster of a class of objects surrounded by a different class of objects and finally a structured environment, that can represent a pseudo-real situation where objects are neatly grouped in different locations.

To populate these environments, we used 5 different kinds of objects: Water, Trees, Bushes, Houses and Walls. What needs to be known about these objects is that each of these classes has a central value for its attributes, and varies slightly in each instance of the object. However, some of them vary a lot in one attribute and a little in the other (Water, Houses) and some vary averagely in both attributes (Tree, Bushes). The classes Tree and Bush are very similar to one another, with the Bushes being smaller and darker than the Trees, allowing us to test the system in an extreme situation where closely related objects aren't supposed to be classified as the same.

In order to test different configurations, we varied the number of agents in the exploring team, the threshold at which an agent finds an object interesting, the starting location of the agents and finally the knowledge of the agents: global or local. The runs of the simulation have been limited to 5000, and we specified that in order to identify an object in place, an agent must spend 10 steps in that position (representing the time it takes to identify in such a way.). In addition to that, the agents have a 40-unit view radius, which will be constant throughout the experimentation.

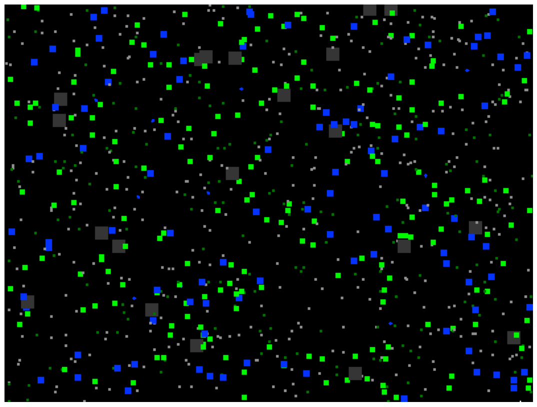
Fig. 1 - Random Environment

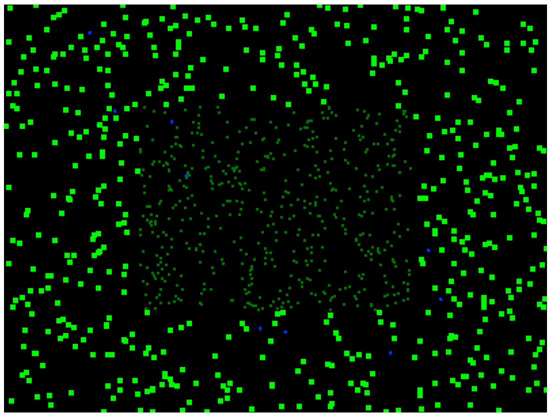
Fig. 2 - Donut Environment

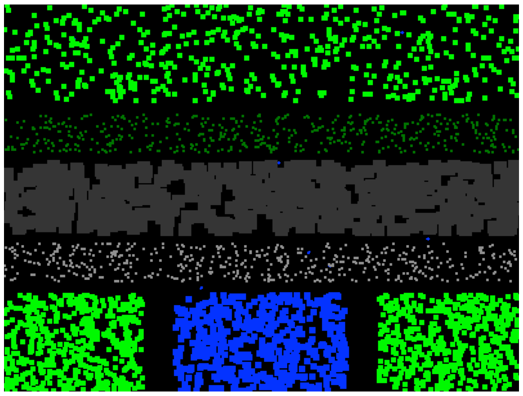
Fig. 3 - Structured Environment

4. RESULTS

We present the results in graphical form, with different variables being tested in different maps, and presenting the percentage of objects identified and the percentage of error versus the number of simulation steps.

4.1 Random Environment

In this environment, we tested the effects of varying the threshold of interest and the number of agents in the system.

Figs. 4 to 8 illustrate the results of varying the threshold in the set {0, 50, 60, 75, 90}, using teams of two agents.

We can clearly see that our systems works as intended: in a classical approach where every point must be visited (threshold = 0), our system doesn't have the time to identify half of the points, but the identification is always corrected, as seen in the zero error. However, by increasing the threshold, we achieve faster exploration (more objects are identified), but also with an also climbing error rate. We can see that with a threshold of 90 almost 20% of the points aren't correctly identified. Seeing that we have 5 classes, this can correspond to an entire class being misclassified all the time. So, the threshold must be balanced to achieve good temporal performances while still maintaining a good level of certainty on our classification. We can see that with thresholds of 60 and 75, a good compromise is reached between exploration speed and error rate.

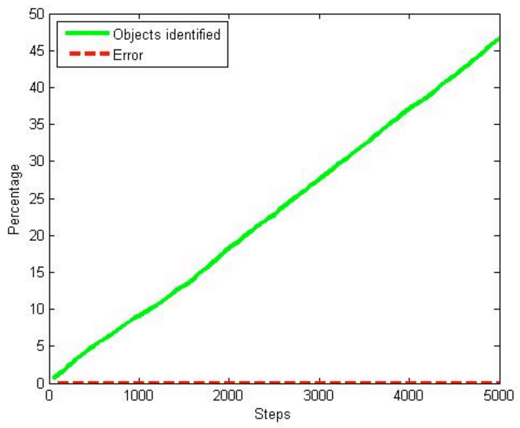

Fig. 4 - Random, 2 Agents, Threshold = 0

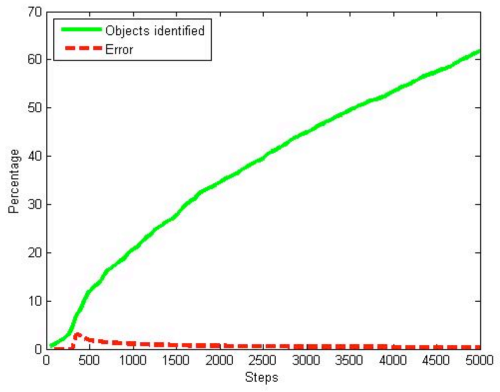

Fig. 5 - Random, 2 Agents, Threshold = 50

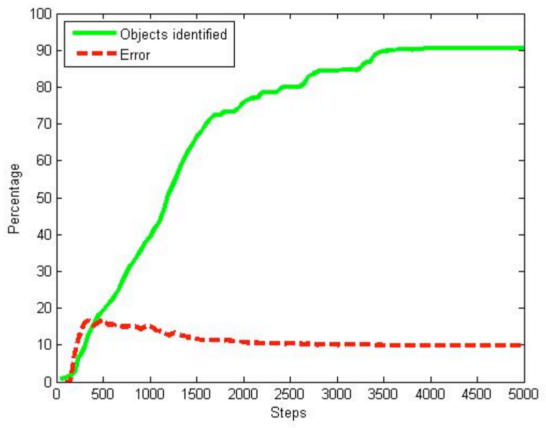

Fig. 6 - Random, 2 Agents, Threshold = 60

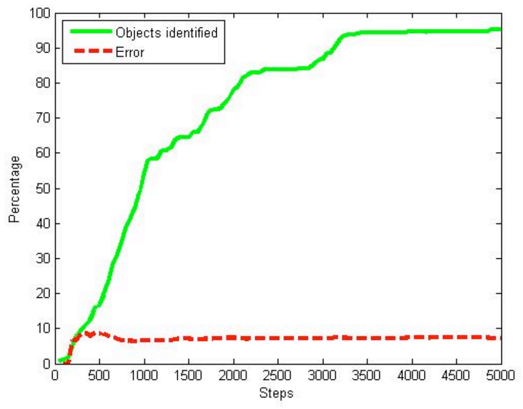

Fig. 7 - Random, 2 Agents, Threshold = 75

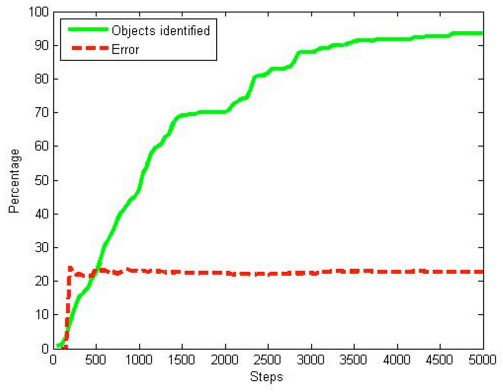

Fig. 8 - Random, 2 Agents, Threshold = 90

Still with the same environment, Figs. 9 – 11 show the results of varying the number of agents exploring in the set {1,5,10}. This is done with a threshold of 65 and with global knowledge – every agent has the same knowledge of the environment, so they share their prototypes.

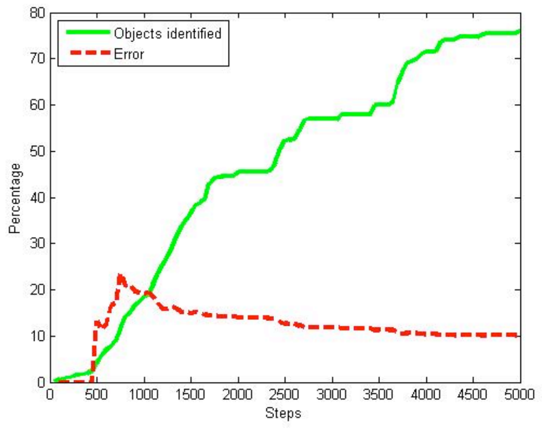

Fig. 9 - Random, 1 Agent, Threshold = 65

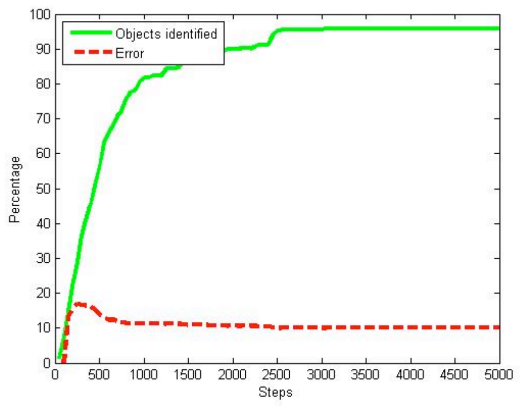

Fig. 10 - Random, 5 Agents, Threshold = 65

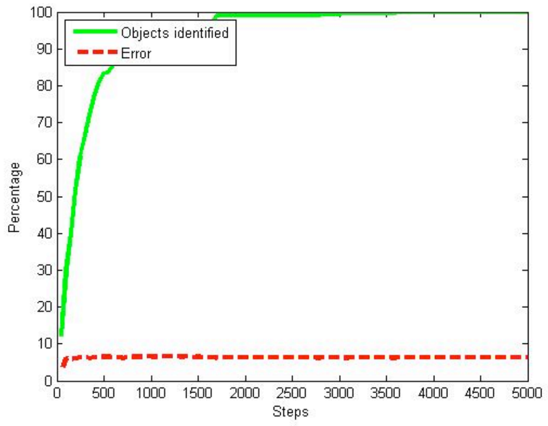

Fig. 11 - Random, 10 Agents, Threshold = 65

As we can see, the performance of the system increases with the number of agents introduced, time-wise. Also, we can see that the error of the exploration decreases with the number of increasing explorers. This is partially because of the global knowledge implementation we are using: with many agents starting in various locations, a better knowledge of the world is quickly gained by the team as a whole, as their shared experiences provide a more accurate knowledge of their surroundings than the knowledge from a single agent. This effect is more thoroughly tested in the next section.

4.2 Donut Environment

With this environment, we put our system to the test under conditions that are far from ideal. In an environment with only two classes of objects that are extremely alike, the MAS can have some trouble correctly classifying these objects. This effect can be mitigated with the use of global knowledge and by determining the starting positions of our agents by hand – a correct classification is possible if all the agents see a similar amount of each of the classes at the same time. An unfortunate positioning of the agents in a cluster of same-classed objects can be disastrous, as well as a situation where each agent has its own knowledge and does not share it.

To show these limitations, we tested our system in this environment, with Trees surrounding Bushes, 2 exploring agents, threshold of 30 and varying the knowledge mode and the starting positions of the agents. The results are show in Figs. 12 -14. We vary the positions as such: in separated clusters, each agent starts in a separate class cluster – one in the center and one in the surroundings; in the same cluster, both agents start in the center cluster.

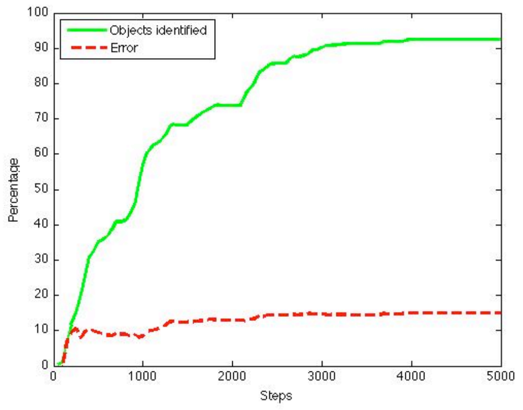

Fig. 12 - Donut, starting in separated clusters, Global Knowledge

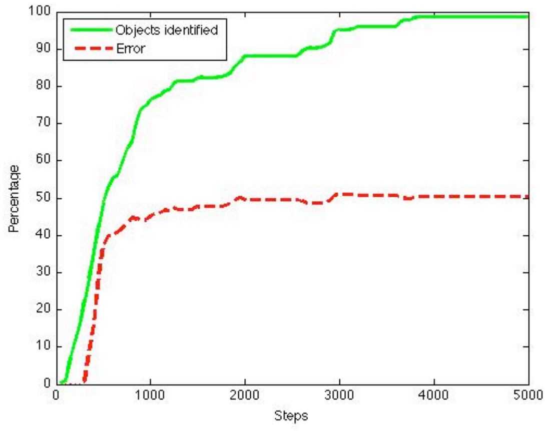

Fig. 13 - Donut, starting in separated clusters, Local Knowledge

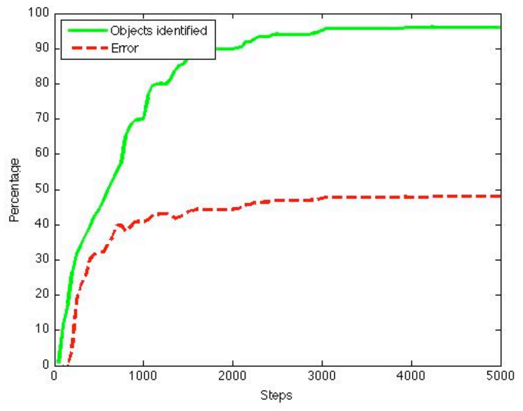

Fig. 14 - Donut, starting in same cluster, Global Knowledge

No tests were made using local knowledge with agents starting in the same cluster, seeing that it would be pointless as the results are very similar to those using global knowledge. As we can see, when both agents start in similar positions, the results are terrible, with 50% error indicating that one of the classes is misclassified every time. This is because the classes are very similar and clearly one of them saturates its prototype much faster and its correlation is rapidly high with the other class. By separating the agents at start, we can see that the results aren't much brighter using local knowledge – each agent basically has knowledge of only one of the classes and identifies each object it sees as that class. It is, however, an improvement we cannot see in these results, if we take as a fact that the error is distributed between the classes, with half of the instances of each class being misclassified, as opposed to a class as a whole not being identified correctly.

Using global knowledge this is mitigated, as we can see in the results: by sharing their knowledge and starting in separate clusters, the agents gain knowledge of both of the classes at the same time and are able to correctly identify most of the objects in a timely fashion.

We can, however, improve on these results by increasing the number of agents. In the previous environment, we saw that by increasing the number of agents with global knowledge, the results were better time and error-wise, because the necessary knowledge for a correct classification is much quickly attained by a large number of agents than by a single one. The results of the same exploration with 8 agents are depicted in Fig. 15. The starting positions of 4 of the agents are in the center cluster and the remaining 4 start in the environment corners.

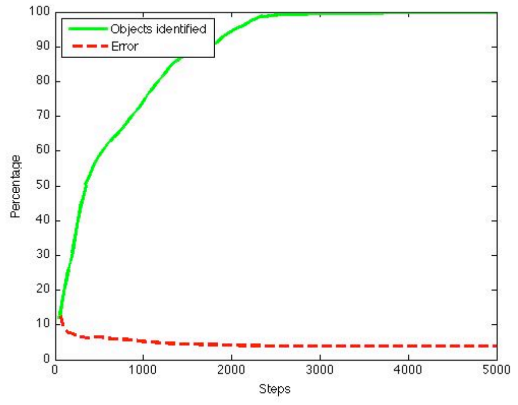

Fig. 15 - Donut, 8 agents, separate clusters, Global Knowledge

As we can see, the results are by far superior from those attained with the previous approaches, in a rather difficult scenario for our MAS. We can now see the importance of agent placement and number – as well as the benefit of using global knowledge - when using our approach.

4.3 Structured Environment

In this environment, there is a structured ordering of the objects, with specific positions for specific classes. This could represent, for instance, a neighborhood with a street full of houses, surrounded by natural scenery – rocks, ponds, a forest. In this environment we have a large number of objects – around 3500 – and is, supposedly, where our approach should have a much higher benefit than a classical approach. As a reference, Fig. 16 represents a classical approach, where every object must be visited, using a high number of agents – 10 agents.

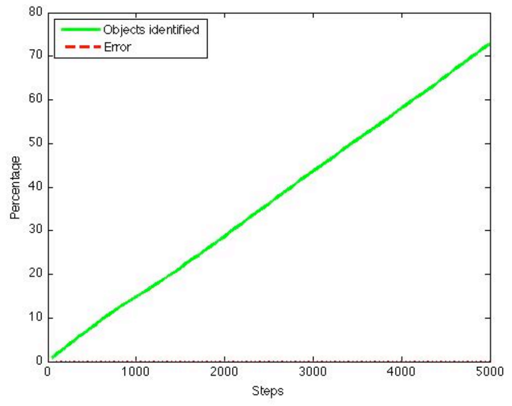

Fig. 16 - Structured, classic approach, 10 Agents

As we can see, the error is null as expected, but even with 10 agents, these can barely cover 70% of the environment in the allotted timeframe. This is due to the huge amount of objects present in the environment, which require a lot of time to identify in the usual way.

We now present results in this environment varying only aspects that we believe are beneficial – the number of agents in the simulation and the interest threshold. Global knowledge was used as it is clearly a benefit for the MAS and the location of the agents will be random, to simulate a truly unknown environment, where nothing is known. We used 5 and 10 agents, with thresholds in the set {45,55,65}. The results are illustrated in Figs. 16 – 22.

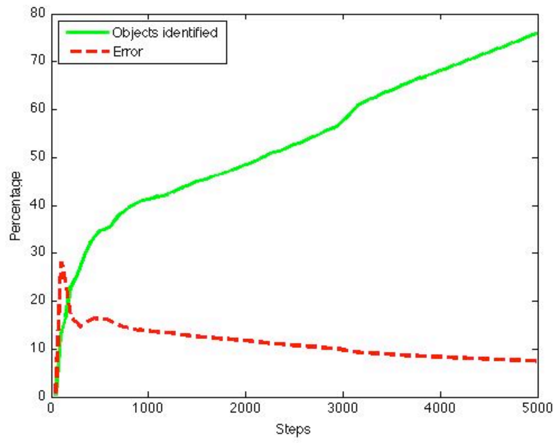

Fig. 17 - Structured, 5 Agents, Threshold = 45

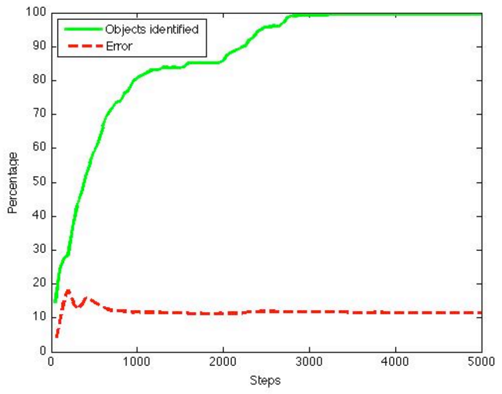

Fig. 18 - Structured, 5 Agents, Threshold = 55

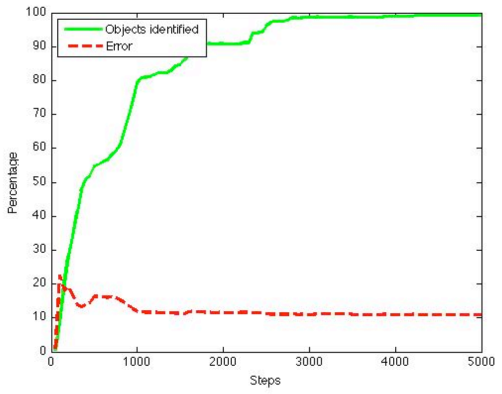

Fig. 19 - Structured, 5 Agents, Threshold = 65

As we can see so far, with half the number of agents our system is able to identify more objects, more quickly than a regular approach, with an acceptable rate of error. The results with 10 agents follow.

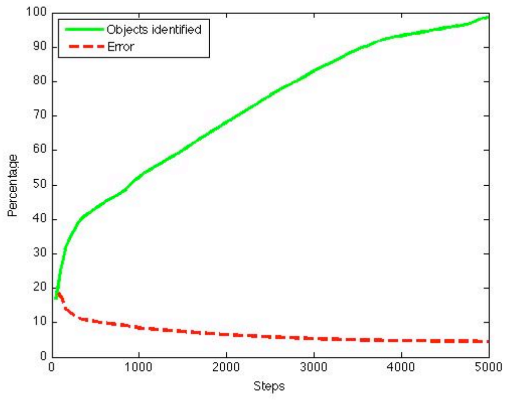

Fig. 20 - Structured, 10 Agents, Threshold = 45

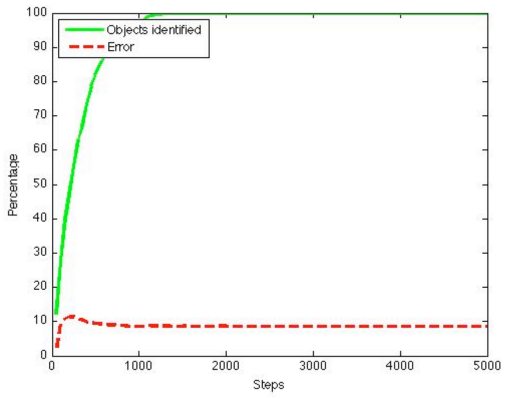

Fig. 21 - Structured, 10 Agents, Threshold = 55

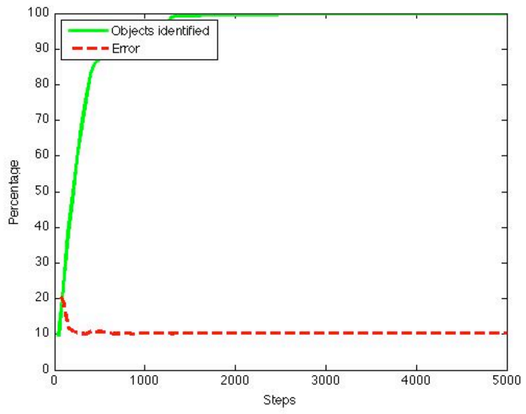

Fig. 22 - Structured, 10 Agents, Threshold = 65

As we can see, the results – time-wise – are clearly superior to those obtained using a classic approach. With ten agents we are able to get a faster coverage of the environment, identifying all of the objects inside the allotted timeframe. This comes, however with a price – some error – that we can see is not really big (with a threshold of 45, the error is about 5%, a more that acceptable margin of error.

5. CONCLUSIONS

As we could see, the approach proposed in this paper is clearly superior to a classic approach, where identifying an object is a time-consuming process in the exploration paradigm. However, this does not come without a cost – this approach, using a predictive, at-range identification, introduces some error derived from classification mistakes or over-training of the interest classification system used.

The approach is also clearly flawed in the aspect that, in the absence of interesting locations to be explored, the agents roam at random through the environment, when some approach could be taken for the agents to visit unvisited areas.

Still, this approach has its merit and, the core of it – the application of an interest threshold to know when to approach or not an object, reveals itself as a more than capable way to explore efficiently and quickly and unknown environment.

Note that this entire work is based on the assumption that no previous knowledge of the environment is given, nor of the possible objects that the agents may encounter. If some information is given, more suitable approaches that rely on more robust classification systems could be used. The approach here presented is effective when used in truly unknown environments where communicating with large knowledge centers is a costly process and where the agents are not capable of storing more than an handful of information about their surroundings.

In these circumstances, we believe this approach to be a robust one, which is able to deliver interesting results.

5.1 Future Work

Some work can be proposed towards the approach here described, in order to enhance it, especially towards reducing the error obtained and enhancing the brokering algorithm:

- The classification system is primitive, at best. Although a solution form scratch was needed due to the constraints of the problem – a classifier with a variable number of outputs was needed, as well as a measure for an unknown factor. This can be improved upon: the algorithm for determining the probability of an unknown object should take into account the already known probabilities for all the prototypes, and not just one at a time.
- Work could be developed towards a self-policing behavior of the agents – checking periodically for classification errors and adapting the system accordingly, for instance by resetting the prototypes or having its weight decrease through time, with this aging effect preventing consecutive misclassifications.
- The number of attributes of each object could be increased or tinkered with, in order to achieve more discriminant results. Some work could be done with feature selection, for instance using PCA or correlation between attributes to select distinctive attributes.

More work could be proposed, of course. If the reader has a suggestion or doubt about this paper, feel free to contact the authors at the given e-mail addresses.

6. ACKNOWLEDGMENTS

The authors wish to acknowledge the support from Luís Macedo, Phd., for his insightful contributions to the work developed and the colleagues at E.5.2 for the supplied workplace, constant understanding and good musical taste. A more personal acknowledgment goes to the close family of both of the authors, for enduring their absence during the development of this work.

7. REFERENCES

[1] [1] S. Thrun - Robotic mapping: A survey, Exploring artificial intelligence in the new millennium, 2002

[2] [2] Burgard, W. Moors, M. Fox, D. Simmons, R. Thrun, S. - Collaborative multi-robot exploration - IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2000

[3] [3] Simmons, R. Apfelbaum, D. Burgard, W. Fox, D. Moors, M. Thrun, S.; Younes, H. - Coordination for multi-robot exploration and mapping- PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2000

[4] [4] Macedo, L. Cardoso, A. - Exploration of unknown environments with motivational agents - Proceedings of the Joint Conference on Autonomous Agents, 200