Uncertainty and Novelty-based Selective Attention in the Collaborative Exploration of Unknown Environments

Luis Macedo, Miguel Tavares, Pedro Gaspar, and Amilcar Cardoso

University of Coimbra,
Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, Polo II, 3030 Coimbra, Portugal
`macedo@dei.uc.pt;{mgt,pgaspar}@student.dei.uc.pt;amilcar@dei.uc.pt`

Abstract. We propose a multi-agent approach to the problem of exploring unknown environments that relies on providing the agents with a measure of interest for the viewpoints of the surrounding environment. Such measure of interest takes into account the expected decrease in uncertainty provided by acquiring the information of objects seen from a viewpoint and the novelty of the potential class label of those objects. This allows the agents to visit selectively the objects that populate the environment. This single agent exploration strategy is combined with a multi-agent exploration strategy relying on a brokering system that allows the coordination of the agent team according to the agents's personal interest and their distance to the viewpoints. The advantages of these forms of selective attention, together with those of the collaborative multi-agent exploration strategy, are tested in several scenarios, comparing our approach against classical ones.

Keywords: Exploration of Unknown Environments, Interest, Curiosity, Active Learning, Selective Attention, Classification, Coordination, Collaboration

1 Introduction

The exploration of unknown environments is a specific kind of active learning [23, 21]. It can be defined as the process of selecting and executing actions in such a way that a maximum of knowledge of a given domain is acquired (e.g., [20]). In the case of physical exploration, the result is the acquisition of a model of the physical environment. Because exploring unknown environments requires resources such as time and energy, there is always a trade-off between the amount of knowledge that can be acquired and the costs of acquiring it. Therefore, exploration strategies that minimize costs and maximize knowledge acquisition have been proposed for artificial agents. These strategies have been grouped into two main categories: undirected and directed exploration [20]. Strategies belonging to the former group (e.g.: random walk exploration, Boltzman distributed exploration) use no exploration-specific knowledge and ensure exploration by merging randomness into action selection. On contrary, strategies belonging to the latter group rely heavily on exploration specific-knowledge for guiding the learning process. Several techniques have been proposed and tested either in simulated and real, indoor and outdoor environments, using single or multiple agents (e.g., [1, 2, 5, 9, 10, 18, 26, 23, 22, 21, 20, 27, 28, 30, 31]). The exploration domains include planetary exploration (e.g., Mars, Titan or lunar exploration) (e.g., [19, 3, 29]), the search for meteorites in Antarctica (e.g., [15]), underwater mapping, volcano exploration, map-building of interiors (e.g., [24, 26, 28]), etc. The main advantage of using artificial agents in those domains instead of humans is that most of them are extreme environments making their exploration a dangerous task for human agents. However, there is still much to be done especially in dynamic environments such as those mentioned above.

Real exploration environments contain objects. For example, office environments possess chairs, doors, garbage cans, etc., cities are comprised of many different kinds of buildings (houses, offices, hospitals, churches, etc.), as well as other objects such as cars. Many of these objects are non-stationary, that is, their locations may change over time. This observation motivates research on a new generation of mapping algorithms, which represent environments as collections of objects [7, 8]. At a minimum, such object models would enable a robot to track changes in the environment. For example, a cleaning robot entering an office at night might realize that a garbage can has moved from one location to another. It might do so without the need to learn a model of this garbage can from scratch, as would be necessary with existing robot mapping techniques. Object representations offer a second, important advantage, which is due to the fact that many environments possess large collections of objects of the same type. For example, most office chairs are examples of the same generic chair and therefore look alike, as do most doors, garbage cans, and so on. As these examples suggest, attributes of objects are shared by entire classes of objects, and understanding the nature of object classes is of significant interest to mobile robotics. In particular, algorithms that learn properties of object classes would be able to transfer learned parameters (e.g., appearance, motion parameters) from one object to another in the same class. This would have a profound impact on the accuracy of object models, and the speed at which such models can be acquired. If, for example, a cleaning robot enters a room it has never visited before, it might realize that a specific object in the room possesses the same visual appearance of other objects seen in other rooms (e.g., chairs). The robot would then be able to acquire a map of this object much faster. It would also enable the robot to predict properties of this newly seen object, such as the fact that a chair is non-stationary, without ever seeing this specific object move.

To our knowledge, the classification methods used to achieve such object models are mostly non-memory based [11, 26, 25, 28]. However, given that most of the environments in which exploration occurs lack a domain theory and are characterized by unpredictability or uncertainty, memory-based classification methods are suitable to classify objects of those environments [6]. Previously, Macedo and Cardoso [12, 14] addressed this issue, but they used a single agent approach.

On contrary, [4, 18, 12, 13] provided evidence that multi-agent approaches are better in comparison with single agent ones, reducing the time required to explore the environment.

However, the exploration strategies used by these multi-agent teams 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. On this aspect, regarding the coordination of the agents, [4, 18, 12, 13] achieved significant results by reducing the redundancy in the mapping and directing each agent to the location most favorable to be explored by it. 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 addition, 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. The work of [12, 13] 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. In all these approaches, the goal is always to map an entire environment, with the (mostly time) costs associated with such an exhaustive method. Also, these simulation works never take into account the cost of fully identifying an object. In a real situation, in which object identification must be done, usually it is costly to have a clear classification of an object based on its characteristics as 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 [16].

In this paper we focus on the selective attention and coordination aspects of an exploration strategy used by the agents to decide the next viewpoints. The coordination aspects of the problem are dealt with a brokering system. Regarding the selective attention, our approach consists of providing the exploring agents with reasoning capabilities that allow 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 making a predictive identification of such an objects at distance instead of approaching them to fully identify them, we aim to reduce mapping times of full environments, while keeping a non significant misclassification level.

The next section presents a multi-agent approach for the exploration of unknown environments. Section 3 describes the functionalities of each type of agent of the multi-agent system. Section 4 presents the experimental tests. Finally, section 5 discusses the results and presents conclusions.

2 Overview of the Multi-Agent System for the Collaborative Exploration of Unknown Environments

Our approach to the exploration problem relies on ensuring that locations that are highly unlikely to possess new and useful information for the mapping, such as empty space or locations with a known geometry, won't be given much attention by the exploring agents, resulting in a partial mapping of the environment.

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 inhabit, 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. Much like a physical agent requires a set of sensors in order to gather information about the surrounding world, our simulated explorer has access through simulated sensors to some properties of the objects it can see. The simulated agent can see objects in all directions within a certain distance.

The environment used is considered as a discrete, two dimensional grid, in which 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, while in the former case the class of the object identified is placed in an auxiliary grid.

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

3 Agents' Description

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

3.1 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 also stores a list of *prototypes*, i.e., an abstract representation of the different objects witnessed by the explorers so far, with information about the average characteristics of 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.

3.2 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 et al. [18], being an adaptation of their 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 presented as follows that takes into account the interest of the point ($Interest$), the distance of the agent to the point ($Dist$), and the maximum distance that an agent is willing to travel before arriving at its target (Max_{dist}) (Equation 1). This formula assures that an agent will pursue the most informative viewpoints that are closer to it.

$$Utility = Interest - \frac{Dist \times 100}{Max_{dist}} \tag{1}$$

The interesting points are sorted by decreasing relevance, and the most relevant one 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.

3.3 Explorer Agent

The explorer agent is the core of this multi-agent system. 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 probabilistically, or it marks it as an interesting spot to be visited later.

Note that these are different ways to identify an object: if the agent arrives at its target and there is an object there, it 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 it has an exact knowledge that the current object is of that class. However, the agent also possesses capabilities to identify an object at distance: if, while analyzing it, the explorer determines that an object is not interesting enough to be visited, it rather chooses to identify it at distance, by picking the most likely class from the probability distribution of classes computed for the object. This approach does not add the object attributes to the class prototype, because the agent cannot be sure that it 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 (e.g., 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 distance.

Classifying objects Afterwards, the explorer agent has the task of classifying the unknown objects, by assigning it a probability distribution based on the knowledge of the worlds it has so far, namely the prototypes that it 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 greenish. Not all trees are really tall or green, but, on average, this is an accurate representation of trees.

The explorer agent browses through a list of prototypes, either its own list or a collaboratively filled list that all the agents share in the mapper agent, and correlates 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 this 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 (Equation 2) is applied to the correlation to take into account the number of occurrences (n_{occurs}) of that object that have been witnessed.

$$Saturation_{corr} = \tanh(\frac{\frac{(n_{occurs}-5)}{2}+1}{2}) \tag{2}$$

This formula was designed to saturate at near 1 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 as it might be a new kind of object that needs its own prototype. This is addressed by the agent by calculating an unknown correlation value that determines the possibility that the current object is not yet known. This value is automatically calculated to 1 when no prototypes are present (at the beginning of the simulation) and using the following formula (Equation 3) when there is already n prototypes, i.e., there is already some knowledge of the world:

$$sim(O_{new}, O_{unknown}) = 1 - \max_i sim(O_{new}, O_i) \tag{3}$$

where $sim(O_{new}, O_i)$ represents the correlation or similarity of the object O_{new} with feature vector x_{new} to the i^{th} prototype O_i with feature vector x_i and class label y_i and is computed as the Euclidean distance between O_{new} and O_i, and more precisely between x_{new} and x_i.

This gives us a fair measure of the chances that this new object O_{new} is something new, given the knowledge the agent has. This value is maximum when all the correlations are 0, and minimum when at least one of the correlations is 1.

Finally, the explorer agent translates these correlations into a probability distribution, using Equation 4.

$$p(y_{new} = y_j | x_{new}) = \frac{sim(O_{new}, O_j)}{\sum_{i=1}^{n+1} sim(O_{new}, O_i)} \tag{4}$$

The explorer agent now has a probability distribution, which sums to 1, that tells the agent, for each known class and for the unknown class, the probability that the current object O_{new} is an instance of that class y_j.

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 Equation 5.

$$I_{unknown}(O_{new}) = \tanh(2 \times P(y_{new} = y_{unknown})) \qquad (5)$$

where O_{new} is an unlabeled new object with feature vector x_{new}, and $P(y_{new} = y_{unknown}|x_{new})$ is the probability that the new object is from a new, unknown class.

This formula is used instead of simply mapping probability to informativeness because we assume that a $x\%$ probability of being something new has more than an informative value of x in a scale of [0-100]. This formula increases the rate of climb of the informative value, saturating when approaching 1, the maximum probability, representing the maximum informativeness, here valued at 1.

However, this is not the only situation where the agent discerns something interesting. It also needs to identify an interest for the chaotic, i.e., 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 using Shannon's approach [17]. Higher entropy values represent more chaotic situations, where all the interest of the agent should be focused. As such, we calculate the entropy (Equation 6) of the probability distribution for that object, and map that value proportionally to an interest value.

$$I_{uncertainty}(O_{new}) = -\sum_{j=1}^{n+1} P(y_{new} = y_j) \times \log(P(y_{new} = y_j) \qquad (6)$$

After calculating these two interest values, $I_{uncertainty}$ and $I_{unknown}$, the explorer agent picks the one that represents the maximum interest and assigns it as the interest value for the object under scrutiny. Note however that other approaches may be considered such as the mean of those values or even their sum (the comparison between these approaches is a future work).

Agent Reasoning After determining the interest for the current object, the explorer agents makes a decision: either 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 distance. This last option is the result we expect from our agent after exploring a bit of the map, becoming aware that this option is less time consuming and it does not require the agent to be in the same exact location of the object. In order to make this decision, each explorer agent has an interest threshold under which he identifies the object at distance and over which it sends the location to the broker, along with its interest.

The experimental tests that follow allows to see whether this approach is favorable to more classical approaches, and how much error this behavior introduces in the mapping of the environment.

4 Experiments

4.1 Experimental Setup

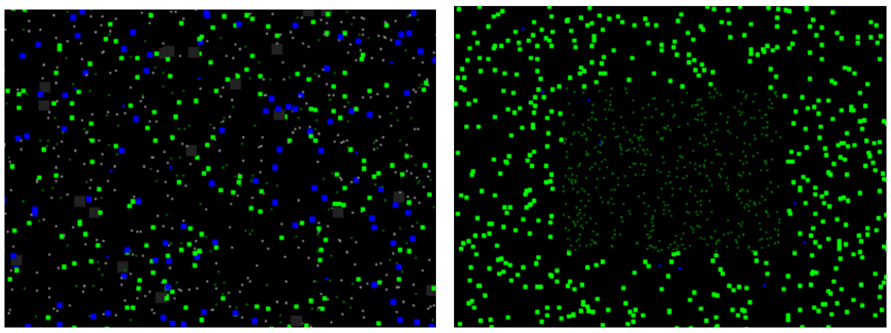
To test our approach, we run a team of explorer agents in several different scenarios (maps with 400 units of width by 300 units of height) with different configurations to test out several aspects of our system. The testing environments were populated with 5 different kinds of objects: Water, Trees, Bushes, Houses, and Walls. These objects were randomly distributed by the environment. Figure 1 presents an example of these testing environment. Objects of different classes are represented with different colors. 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 others (Water, Houses) and some vary averagely in all 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 informative, the mode of sharing knowledge (either sharing their knowledge – global knowledge – or not sharing it – local knowledge), and the starting positions of the agents.

The runs of the simulation have been limited to 5000, and we specified that in order to classify an object in place, an agent must spend 10 steps in that position (representing the time it takes to identify it). In addition to that, the agents have a 40-unit view range radius, which will be constant throughout the experimentation.

4.2 Experimental Results

We present the results of the exploratory study about the influence of the informativeness threshold and number of agents on the percentage of objects classified or misclassified.

Figure 2 illustrates the results of varying the threshold in the set {0, 50, 75, 90}, using teams of two agents. The agents share their knowledge, so every agent has the same knowledge of the environment. We can clearly see that our system works as hypothesized: in a classical approach where every object must be visited (threshold = 0), the system doesn't have the time to classify half of the objects, but the identification is always correct. However, by increasing the



(a) Random environment. (b) Donut environment.

Fig. 1. Two kinds of environment.

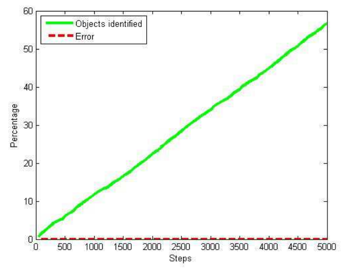
threshold, we achieve faster exploration (more objects are identified), but also with an also climbing error rate. We can see that with a threshold of 90 over 20% of the objects are misclassified. So, the threshold must be balanced to achieve good temporal performances while still maintaining a good level of certainty on the classification. It can be seen that with a threshold of 50 a good compromise is reached between exploration time and error rate.

Figure 3 shows the results of varying the starting positions of the agents and the strategy of sharing or not sharing their knowledge. This is done with 2 explorer agents and a threshold of 30. It also shows a comparison between using 2 and 8 agents, while keeping constant the starting positions of the agents and the sharing knowledge mode.

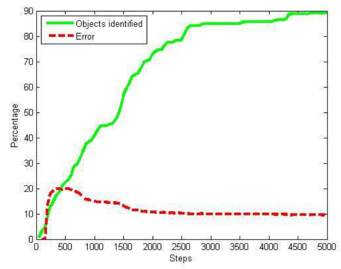
As it can be seen, when both agents start in similar positions, the results are worse, with 50% error indicating that one of the classes is misclassified every time. This is because the classes are very similar and clearly one of them saturates its prototype much faster and its correlation is rapidly 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 it can seen in the graphs: 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 efficiently.

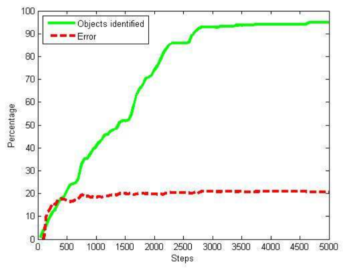
As it can be seen, increasing the number of agents with global knowledge, the results are better time and error-wise, because the necessary knowledge for a



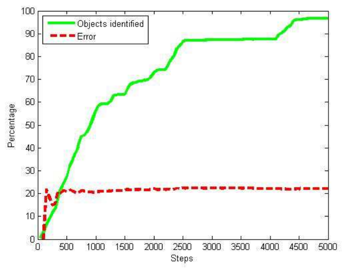
(a) Agents=2; Informativeness Threshold = 0.



(b) Agents=2; Informativeness Threshold = 50.



(c) Agents=2; Informativeness Threshold = 75.

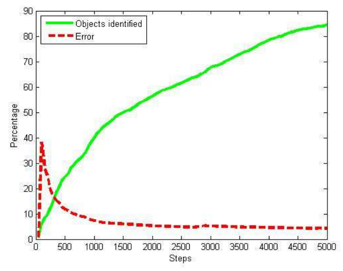


(d) Agents=2; Informativeness Threshold = 90.

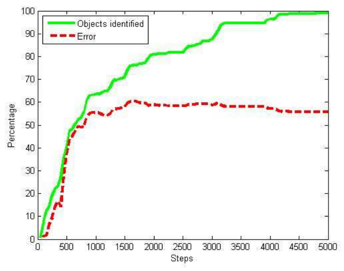
Fig. 2. Results for 2 agents and different informativeness threshold.

correct classification is much quickly attained by a large number of agents than by a single one. The starting positions of the 8 agents are: 4 of the agents are in the center cluster and the remaining 4 start in the environment corners.

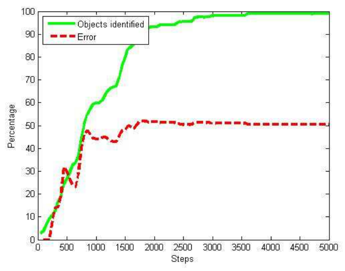
The results are much better than those attained with the previous approaches, in a rather difficult scenario for our multi-agent system. The results show evidence for the importance of agent placement and number of agents, as well as for the benefit of using global knowledge. The performance of the system increases with the number of agents introduced. Also, 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.



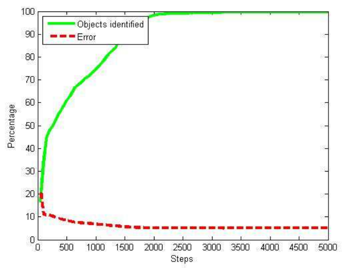
(a) Agents=2; pos=separated; knowledge=global.



(b) Agents=2; pos=separated; knowledge=local.



(c) Agents=2; pos=same; knowledge=global.



(d) Agents=8; pos=separate; knowledge=global.

Fig. 3. Results in the donut environment for different sharing modes of knowledge, different starting positions for the agents, and different number of agents.

5 Discussion and Conclusions

We proposed a multi-agent system for the classification of objects that populate unknown environments, in which each explorer agent is equipped with a classifier that selects for labeling those unlabeled objects that are more informative. The informativeness of an unlabeled object is measured in terms of the decrease in uncertainty by labeling it and also in terms of the novelty of its potential class label.

There is evidence indicating that the approach proposed in this paper is superior to a classic approach, in which 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 distance identification, introduces some error derived from classification mistakes. The approach is also clearly flawed in the aspect that, in the absence of informative objects to be explored, the agents roam 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 informativeness 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. The experimental results obtained with different sizes for the team of agents each one with that selective classification of objects agree with the previous results obtained in previous studies.

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 promises to be effective when used in real unknown environments where communicating with large knowledge centers is a costly process and where the agents are resource bounded by not being able to store a large amount of information about their surroundings. In these circumstances, we believe this approach is a robust one, which is able to deliver interesting results.

Some extra work should be done in the future in order to enhance this approach, especially towards reducing the error obtained and enhancing the brokering algorithm. For instance, some 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 knowledge base or having its weight decrease through time, with this aging effect preventing consecutive misclassifications.

Further experimental tests are required to study the influence of the kind of the environment in the performance of the agents. What are the results obtained in structured environments? What happens with environments of different complexity? Moreover, more statistical techniques such as ANOVA should be carried out to assess the significance of the results. In this case, we may find conclusions about the influence and interaction of the independent variables (the number of agents, informativeness threshold, starting positions of the agents, and knowledge sharing mode) on the dependent variable (classification correctness). Preliminary results obtained with a structured environment indicate evidence of the same behaviour achieved in unstructured environments. Furthermore, more information metrics may be used, and their influence on the performance of the classifier may be studied using factorial experiments.

References

1. Amat, J., Mantaras, R., Sierra, C.: Cooperative autonomous low-cost robots for exploring unknown environments. In: Khatib, O., Salisbury, K. (eds.) Proceedings of the 4th International Symposium on Experimental Robotics IV, Lecture Notes in Control and Information Sciences, vol. 223, pp. 40–49. Springer, Stanford, CA, USA (1997)
2. Anguelov, D., Biswas, R., Koller, D., Limketkai, B., Sanner, S., Thrun, S.: Learning hierarchical object maps of non-stationary environments with mobile robots. In:

Darwiche, A., Friedman, N. (eds.) Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence, pp. 10–17. Morgan Kaufmann Publishers, Inc., Alberta,Canada (2002)

3. Bresina, J., Dorais, G., Golden, K., Smith, D., Washington, R.: Autonomous rovers for human exploration of mars. In: Proceedings of the Mars Society Founding Convention. Boulder, Colorado (1999)
4. Burgard, W., Fox, D., Moors, M., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 476–481. IEEE Computer Society, San Francisco, CA, USA (2000)
5. Burgard, W., Moors, M., Schneider, F.: Collaborative exploration of unknown environments with teams of mobile robots. In: Beetz, M., Hertzberg, J., Ghallab, M., Pollack, M. (eds.) Advances in plan-based control of robotic agents, Lectures Notes in Computer Science, vol. 2466. Springer Verlag, Berlin (2002)
6. Gupta, K., Aha, D., Moore, P.: Case-based collective inference for maritime object classification. (pp.). :. In: Proceedings of the Eighth International Conference on Case-Based Reasoning, pp. 443–449. Springer, Seattle, WA (2009)
7. Hähnel, D., Burgard, W., Thrun, S.: Learning compact 3d models of indoor and outdoor environments with a mobile robot. Robotics and Autonomous Systems 44(1), 15–27 (2001)
8. Hähnel, D., Triebel, R., Burgard, W., Thrun, S.: Map building with mobile robots in dynamic environments. In: Proceedings of the International Conference on Robotics and Automation, pp. 1557–1563. IEEE Computer Society, Taipei, Taiwan (2003)
9. Lee, D.: The map-building and exploration strategies of a simple, sonar-equipped mobile robot; an experimental, quantitative evaluation. Phd, University College of London (1996)
10. Lee, D., Recce, M.: Quantitative evaluation of the exploration strategies of a mobile robot. International Journal of Robotics Research 16(4), 413–447 (1994)
11. Liu, Y., Emery, R., Chakrabarti, D., Burgard, W., Thrun, S.: Using em to learn 3d models of indoor environments with mobile robots. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 329–336. Morgan Kaufmann Publishers, Inc., Williams College, Williamstown, MA, USA (2001)
12. Macedo, L.: The Exploration of Unknown Environments by Affective Agents. Phd thesis, University of Coimbra (2007)
13. Macedo, L., Cardoso, A.: Exploration of unknown environments with motivational agents. In: Jennings, N., Tambe, M. (eds.) Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 328 – 335. IEEE Computer Society, New York (2004)
14. Macedo, L., Cardoso, A.: Using cbr in the exploration of unknown environments with an autonomous agent. In: Calero, P., Funk, P. (eds.) Advances in Case-Based Reasoning: Proceedings of the 7th European Conference on Case-Based Reasoning, pp. 272–286. Springer, Madrid, Spain (2004)
15. Moorehead, S., Simmons, R., Apostolopoulos, D., Whittaker, W.: Autonomous navigation field results of a planetary analog robot in antarctica. In: International Symposium on Artificial Intelligence, Robotics and Automation in Space. Noordwijk, Holland (1999)
16. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
17. Shannon, C.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423 and 623–656 (1948)

18. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence, pp. 852–858. Austin, Texas, USA (2000)
19. Simmons, R., Krotkov, E., Chrisman, L., Cozman, F., Goodwin, R., Hebert, M., Katragadda, L., Koenig, S., Krishnaswamy, G., Shinoda, Y., Whittaker, W.: Experience with rover navigation for lunar-like terrains. In: IROS (1995)
20. Thrun, S.: Efficient exploration in reinforcement learning. Tech. Rep. CMU-CS-92-102, Carnegie Mellon University, Computer Science Department (1992)
21. Thrun, S.: The role of exploration in learning control. In: White, D., Sofge, D. (eds.) Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches, pp. 527–559. Van Nostrand Reinhold, New York (1992)
22. Thrun, S.: Exploration and model building in mobile robot domains. In: Proceedings of the International Conference on Neural Networks, pp. 175–180. San Francisco, CA (1993)
23. Thrun, S.: Exploration in active learning. In: Arbib, M. (ed.) Handbook of Brain Science and Neural Networks, p. 381384. MIT Press, Cambridge, MA (1995)
24. Thrun, S.: Learning maps for indoor mobile robot navigation. Artificial Intelligence (1997)
25. Thrun, S.: Probabilistic algorithms in robotics. Tech. Rep. CMU-CS-00-126, School of Computer Science, Carnegie Mellon University (2000)
26. Thrun, S.: Robotic mapping: a survey. In: Lakemeyer, G., Nebel, B. (eds.) Exploring Artificial Intelligence in the New Millenium, pp. 1–35. Morgan Kaufmann Publishers, Inc., San Francisco, CA (2002)
27. Thrun, S., Burgard, W., Fox, D.: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation, pp. 321–328. San Francisco, CA, USA (2000)
28. Thrun, S., Thayer, S., Whittaker, W., Baker, C., Burgard, W., Ferguson, D., Hhnel, D., Montemerlo, M., Morris, A., Omohundro, Z., Reverte, C.: Autonomous exploration and mapping of abandoned mines. IEEE Robotics and Automation Magazine 11(4), 79–91 (2005)
29. Washington, R., Bresina, J., Smith, D., Anderson, C., Smith, T.: Autonomous rovers for mars exploration. In: Proceedings of the 1999 IEEE Aerospace Conference. Aspen, CO, USA (1999)
30. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proceedings of the Second International Conference on Autonomous Agents, p. 4753. Minneapolis, MN, USA (1998)
31. Yamauchi, B., Schultz, A., Adams, W.: Integrating exploration and localization for mobile robots. Adaptive Systems 7(2), 217–230 (1999)