Collaborative Exploration of Unknown Environments with Teams of Heterogeneous Agents

Jani Hilliaho
Tampere University of Technology
Korkeakoulunkatu 10
33720 Tampere, Finland
jani.hilliaho@student.tut.fi

Sebastian Rehfeldt
Hasso Plattner Institute
Prof.-Dr.-Helmert-Straße 2-3
14482 Potsdam, Germany
sebastian.rehfeldt@student.hpi.de

ABSTRACT

In this paper, we tackle the problem of exploring unknown areas with teams of heterogeneous agents by enabling the agents to predict classes of unknown objects and assigning interest values to the objects. Based on the interests and an interest threshold, unknown objects are either directly classified over distance or completely classified which takes longer. As no previous knowledge about the classes is available during the exploration, we put particular emphasis on tackling this so-called cold-start problem. With this work, we prosecuted the research of Macedo et al.[4] by introducing a new classification approach, a new agent type and a clustered broking approach. The results are tested in several scenarios and compared by their coverage of the map and classification errors.

Categories and Subject Descriptors

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

General Terms

Algorithms, Artificial Intelligence, Exploration

Keywords

Multi Agent Systems, Cooperation and Coordination, Mason, Exploration of Unknown Environments, Teams of Agents, Classification of Objects, Cold-Start Problem

1. INTRODUCTION

The exploration of unknown environments is a scientific research field of great human interest. An exploration can become necessary out of various reasons, e.g. when an accident or natural disaster occurs and rescuers need to salvage involved people or, due to the curious nature of the humans, to scout new countries or planets. In cases where it is too dangerous or time-consuming for humans to explore these areas, autonomous and intelligent robots can be used to take on the job of exploration.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

In a simple scenario a mobile and autonomous robot could explore an unknown environment and report his observations to a human. Whether this is possible in small and easy scenarios, it will not work out in larger scenarios or in scenarios where lives are in danger and the time for exploration is limited. For these complex scenarios a Multi Agent approach could be used to reduce the time of acquiring a model of an unknown area. However, the usage of multiple robots can lead to redundancy and it is clear that the algorithms used in a Single Agent approach will not work out in a Multi Agent approach. The usage of many robots will only lead to a reduction of the mapping time when the robots collaborate and share their knowledge to explore an environment. Therefore, they need to coordinate their tasks and minimize the redundancy in the exploration.

In this paper, we want to build on the work of Macedo et al.[4] and compare different strategies for robots to select the locations, which should be explored next, such that a maximum knowledge about the environment is derived while only a minimum amount of resources and time is used. We put attention on the identification and classification of unknown objects, as these objects are of particular interest in real world scenarios. Therefore, different approaches like a classification based on the correlation of attributes or k-nearest-neighbors will be presented and compared. The classification can be used to predict classes of unknown objects instead of doing a full examination which would lead to longer mapping times. With this approach we want to show that it is acceptable to introduce small errors to the mapping, due to the predictions, if the ratio of gained knowledge and used resources (robots and time) improves drastically. We also provide our robots with the ability to rate objects according to their interest/ utility. In cases where time is rare, objects of low interest could be ignored and predicted by our classifier to reduce the mapping time. When there is time left, this time can be used to reclassify objects with predicted class labels to reduce the error. Dynamic interest thresholds and different metrics for the interest are presented and discussed to gain as much knowledge about the environment as possible. Furthermore, we discuss different broking approaches and investigate whether different agent types can be beneficial to the exploration.

In the next section, we give an overview of related work and different kinds of Agent systems. We introduce the environment used for our research in section 3 before we give a detailed description of our system in section 4. Subsequently, we show the results of this system in an experi-

mental setup in section 5. In a last part, the paper will be rounded off with a discussion of our results in section 6.

2. RELATED WORK

The acquisition of a model for a physical environment by autonomous robots is called the robot mapping problem. According to Thrun[7] it is the most important problem to solve when it comes to building autonomous robots. The problem is widely researched and there are robust methods for mapping static environments. The mapping of dynamic environments is still a subject under study and not completely solved.

According to Alonso et al.[1] it is crucial to intelligent agents that they are able to learn. In the context of exploration, agents need to learn from their interaction with the environment they explore. Thrun[6] classified the learning of agents into passive and active learning. In contrast to the passive learning, where an agent is only a receiver of training data generated by the environment, an agent interacts with an environment in the active learning. This, clearly, is the case for the exploration of unknown areas where an agent needs to decide for actions which contribute to the learning task.

The exploration strategies can be divided into two categories: undirected and directed exploration.[8] Poncela et al.[5] describe undirected schemes as exploration based on randomness, so that every action of an agent is based on a probability distribution. In contrast to that, directed exploration uses exploration specific information like the current representation of the explored environment. These information are used to determine actions which are optimal in maximizing the ratio of knowledge gain and effort. In that way, agents could discover parts of the environment with the currently highest uncertainty. An additional exploration strategy is given by Macedo and Cardoso[3] who propose the use of motivational or affective agents. Under that strategy, decisions are made based on the feeling of agents, like surprise, curiosity and hunger which are integrated into the traditional models of Artificial Intelligence.

Agent systems can further be categorized into Single or Multi Agent Systems (MAS). Whether the first approach is more simple to implement, a MAS is believed to have advantages over Single Agent Systems. Burgard et al.[2] name three of these advantages. First multiple robots can fulfill tasks faster than a single robot. Furthermore, they can localize themselves more efficiently by communicating with each other. Finally, the usage of multiple robots is more fault-tolerant as redundancy is introduced. The biggest problem with MAS is to handle redundancy.

To handle this problem two architectures are possible. A Master-Slave architecture where a master agent collects the data from the exploring agents and combines them into a global map. Based on this map, the master dispenses jobs to the other agents. In that approach, the master is an agent which cannot be used for exploration and which is also a probable bottleneck. In a second approach, agents could be more free and contain individual maps and communicate with each other to reduce redundancy. In this approach the communication overhead can probably decrease the efficiency of the robots, but it is more fault-tolerant as there is no global map or master agent.

As most approaches try to explore the complete environment, it is sometimes not necessary or possible to discover the whole area and a maximum knowledge gain given a limited time is desired, e.g. rescues after a natural disaster. Macedo et al.[4] introduced an approach where objects are rated to an interest level. In their approach uninteresting objects are ignored by the agents as they do not strongly contribute to the knowledge gain. Also they try to predict object class labels instead of processing a complete classification. This further safes time to explore more objects in a shorter time by introducing a small classification error.

In this paper, we want to prosecute their work and especially focus on the classification of objects which is originally based on the correlation of attributes of an unknown object and the averages of the existing classes. Our classification will rely on statistically more reliable values and is extended with a k-nearest-neighbour approach with dynamic interest thresholds. Furthermore, we use free time to reclassify objects and introduce a new broking approach. Different environments are tested to discuss the different classification and broking approaches.

Furthermore, we will compare different interest and relevance functions and investigate whether the use of heterogeneous agents is superior to the use of homogeneous.

3. ENVIRONMENT AND EXPERIMENTAL SETUP

In this approach, we use the experimental setup introduced by Macedo et al.[4] and compare their system with our adjustments. Therefore, we adopted their general idea and environments to make our results comparable in a fair manner. As they introduced in their paper, the exploration is based on "surprise". Therefore, agents are send to areas which hopefully provide the most information for the exploration whether regions without any interest are discarded and objects under a given interest threshold are identified with the classification algorithm. By using multiple explorer agents with collaborative map building and a broking algorithm, the unknown environment should be discovered rapidly. To measure the exploration performance multiple experiments were conducted.

3.1 Preparing the experiments

To get any useful analysis of the exploration, the experimental setup needs to be close to real world scenarios. Macedo et al. therefore provided the agents with the abilities to locate themselves on the map, e.g. using a GPS module, and to see objects in a given view range, e.g. using a camera. Furthermore, the agents have a fixed velocity and need time to completely identify an objects as it would be the case in the reality. To make the classification more challenging, no previous knowledge about the classes is given. This also tests whether unknown or unexpected classes can be detected by the agents as we cannot assume that every object in a real world scenario would be expected, e.g. in the Mars exploration scenario. The lack of training data is called cold-start problem and limits the choice of classification algorithms at the beginning.

Finally, the experiments and environments are implement using MASON, a multi-agent system toolkit implemented in Java. Figure 1 shows three different maps with varying layouts, object types and object proportions. The MASON maps are 2-dimensional grids where each cell can consist of

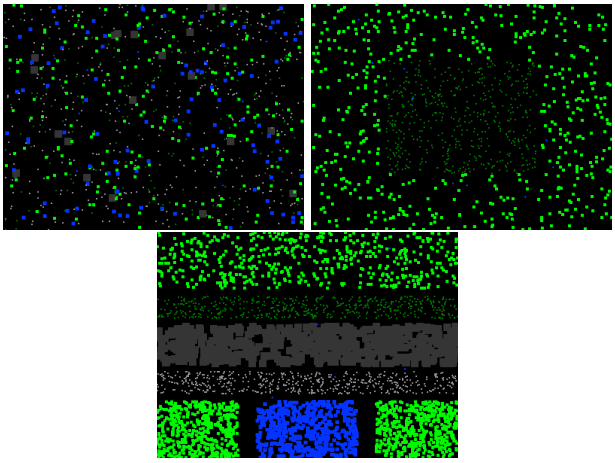

Figure 1: Experimental environments - Random, Donut and Structured

one agent, one object or both together.

The figure shows a random map which contains all types of objects in random locations. This map can be used as a baseline analysis for the exploration. The second environment only contains bushes, darkgreen and small, and trees, lightgreen and big. For simplicity color and size are the only attributes describing the different objects. The concrete values for the attributes are surrounded around a center value. Especially in the case of bushes and trees the attributes are close to each other, which makes the classification really hard and hence, this environment can be used to analyze the classification capability of the agents. In a final environment, the map contains also water (blue), houses (big, gray) and walls (small, gray). This map is structured and contains a bigger amount of objects. By having more objects, this map can be used to measure the time necessary to identify all objects. Also this map is probably the most realistic map and hence, used for final evaluations.

3.2 Comparing different strategies

Strategies in this work are evaluated by the percentage of identified objects and the error of classification. To make these measurements comparable, it is important that important parameters stay fixed. Otherwise, they would interfere the results and make the strategies incomparable. We therefore decided to not change parameters like the velocity and view range of agents. Varying view ranges, velocities or identification times could be used in the development of different agent types, e.g. agents with a shorter identification time could be dedicated for completely examine objects. But as this was not possible and studied in the original paper, we leave the idea of dedicated agents open. Instead, we focus on new classification and broking approaches.

4. SYSTEM ARCHITECTURE

This section gives an overview over the components and agents used in the exploration and is mainly a summarization of the architecture of Macedo et al[4]. This should give a better understanding of the system and allow the reader to better follow and understand the next section which introduces and discusses different strategies.

4.1 Introducing the system architecture and agent types

We decided to keep the Master-Slave architecture as we think that it is superior to a flat architecture. In a flat architecture, with distributed maps, agents need to communicate a lot with each other to avoid redundancy in the exploration. As the communication grows exponentially with the amount of agents, we assume that the communication overhead will slow down the exploration too much. Another possible architecture is to use many masters. This could be useful when the master is the bottleneck, due to limited computation and communication capabilities, or a failure of a master is too risky. Nevertheless, this is not the case for us and we stick to the initial architecture consisting of three agent types.

- A **Mapper** agent is used to collect the information about identified objects from the explorers and to merge them into a global map. This keeps the communication overhead linear and reasonable.
- A **Broker** is used to keep track of interesting locations and to coordinate the exploration.
- **Explorer** agents are used to explore the environment and to send knowledge about the environment to the Mapper.

The next parts describe the different agent types in more detail and introduces bigger changes we made.

4.1.1 Mapper Agent

The Mapper merges information about identified objects and stores them with their locations in a Sparse Object Grid. Also the Mapper provides auxiliary arrays for known locations and objects which can be queried by the explorers. This enables explorers to skip objects and locations which have already been explored and helps in the classification which relies on k-nearest-neighbours. Another task of the Mapper is to contain a list of Prototypes which can be seen as templates of identified objects. These prototypes can be queried by the explorers and be used for classification as they describe the known object classes.

One change to the original implementation is that we not only store the average attribute values in the prototypes, but also all objects. This was a necessary change to enable the knn classification.

4.1.2 Broker Agent

The Broker receives interesting points from the explorers and manages a list of locations with their interests. When an explorer requests a new target from the broker, a relevance function is applied to each point to create a ranking for the explorer. The Broker then sends the explorer to the top-ranked location.

As one change to the Broker, we added a clustering to the original broking. Therefore, the map is divided into a grid and explorers are sent to dedicated areas to keep the distances small and to ensure that all areas are explored.

4.1.3 Explorer Agents

Explorers request targets from the Broker and then go the location. On their way they see unknown objects in a given view range and create a probability distribution for the classes for each object. Based on this distribution, an

interest is assigned to each object and compared to an interest threshold. If the interest is below the threshold, the object is not interesting enough and it is directly classified with the most probable class from the distribution. Otherwise, the location is marked as interesting and sent to the Broker which safes the location for a later examination.

To make it easier to create different explorer agents, we created a parent class for the explorer which contains all of its' important functions. New explorer types can be simply added by sub-classing the parent class and overwriting methods. In that way we were able to introduce new agents types which rely on different interest functions.

4.2 Drawbacks of the initial system

We see two big areas for improvement in the initial system. First, the classification was trivial and second, explorers are running around randomly when the Broker does not know any interesting points.

We put much emphasis on the classification by introducing a second classification approach and testing different functions and thresholds. Also we are using the time when the Broker does not know any interesting points to reclassify objects with predicted class labels. This should reduce the error and lowers the cold-start problem in which early predictions can be error-prone. With reclassification we can patch the wrongly classified objects when more training data is available and hence, use the time more rational.

Also we use clustering in the broking to keep distances small and to reduce the random running of agents when no interest points are available. Another advantage of the clustering is that prototypes for all classes are found more rapidly as it would be the case when many explorers start in the same area and stay there.

The next section will introduce detailed changes in the code and discusses their impact on the exploration performance.

5. IMPROVEMENTS AND RESULTS

Various experiments were conducted to compare the different strategies. During the testing we tried to only make one change at a time to avoid interference and get clear conclusions on what works. The first five experiments deal with the classification, the sixth analyzes the clustering and the final experiment compares the old system with our final version.

5.1 Improved classification

In a first approach to improve the classification, we modified the prototypes so that they contain a list of all their occurrences. Instead of relying on the average attributes of a prototype class for the correlation calculation, we use the median for each attribute as the median is more statistically reliable. Furthermore, we check if the attributes of the unknown objects lie between the 25% and 75% quantile of the prototype class. If the attributes are outside this range for all classes, we say that we were not able to classify it and assign 1 to the probability of an unknown class. As this did not result in huge performance improvements, we added a knn classification which replaces the initial calculation of class probabilities when enough training data is available.

In the knn algorithm we compute the 10 nearest neighbors inside the identified objects to the unknown object based on Euclidean distance. The correlations are then computed as the ratio of examples for a class during the neighbors and 10. The initial knn implementation starts when 60 examples for a prototype are available. This was set up after experimenting and is a fixed value. During our experiments we recognized that the classification in the end is really reliable and led to a fast exploration as almost every point was directly classified using knn. Only in the beginning it resulted in high errors. That is why we decided to start with a low interest threshold and increased it when more training data was available. This prevented knn from directly classifying everything in the beginning. In our experiments we started with an interest threshold of 35, increased it to 40 after 60 examples and from there on increased it by 5 after every 10 prototype examples until it reached 80.

During the experiments no other functions were changed and the interest was computed from the probability distribution like in the original work. Figure 2 and 3 show the performance of 5 agents in the random environment.

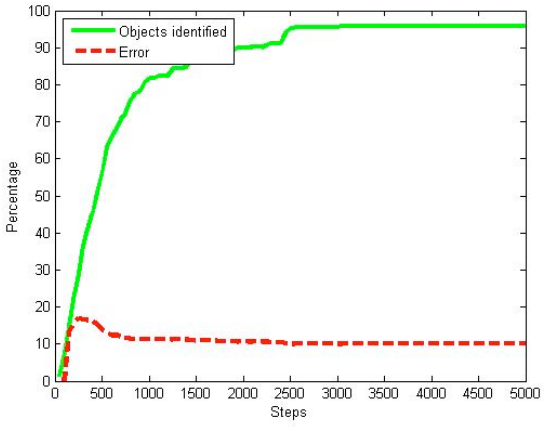

Figure 2: Random Environment with 5 agents - Old

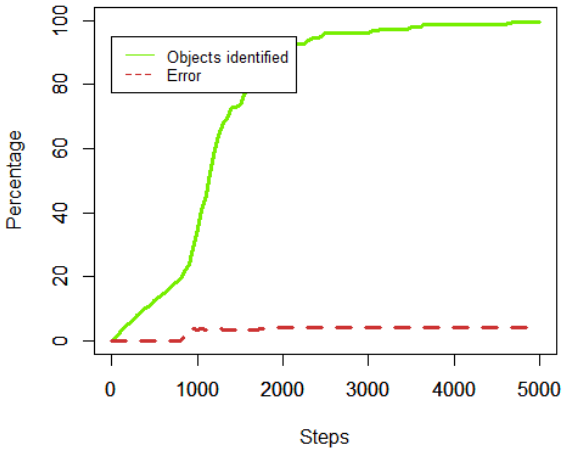

Figure 3: Random Environment with 5 agents - KNN

You can see from the second figure, that the knn classification is used after 1000 steps. Before only 20% of objects were covered in contrast to around 80% in the initial implementation. After 1000 more steps the knn implementation almost reached 100% and catched up the initial implementation. By using knn the classification error went slightly up and then stayed at a constant level. In the end, the knn implementation covered all objects whether the original implementation missed some percent, probably objects in the corner as they are more unlikely to be in the view range of the explorers. Also it can be seen that the classification error is significantly lower in the second which shows that the new approach can be beneficial in reducing the error while keeping the speed.

The improvement in the error is also studied inside the donut environment. Figure 4 and 5 show the results for 2 agents in this environment.

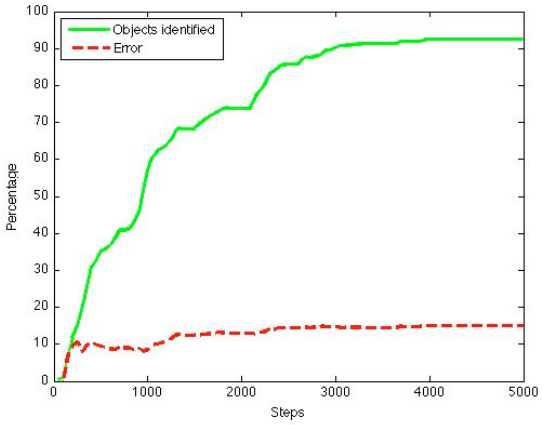

Figure 4: Donut Environment with 2 agents - Old

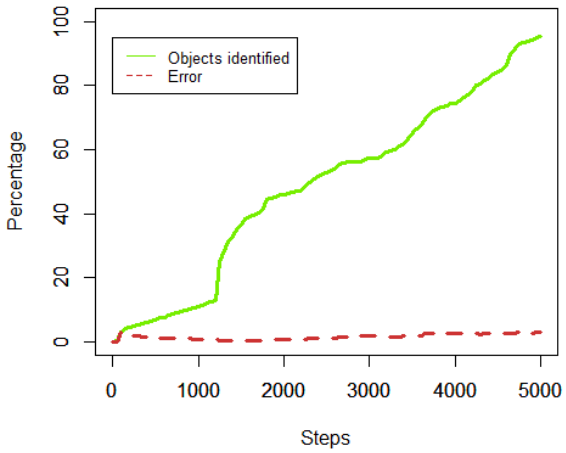

Figure 5: Donut Environment with 2 agents - KNN

The difference in the error is even stronger in this environment. Both approaches reached a coverage around 90%, but the knn classification had a significantly lower error with only 3% whether it was around 15% in the initial implementation. That underlines our assumption from the first environment that the new classification is superior.

As the threshold was set by heart, we tested in every environment and with all agent types. We came up with the problem that the knn classification was nether used when only one agent was available, because the needed amount of training data was not reached. This resulted in a very slow exploration which can be seen and compared in figures 6 and 7.

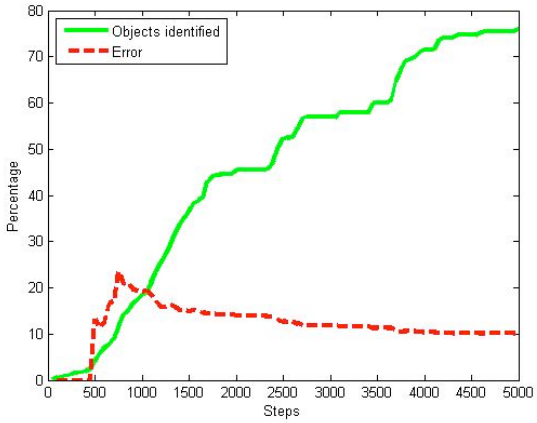

Figure 6: Random Environment with 1 agent - Old

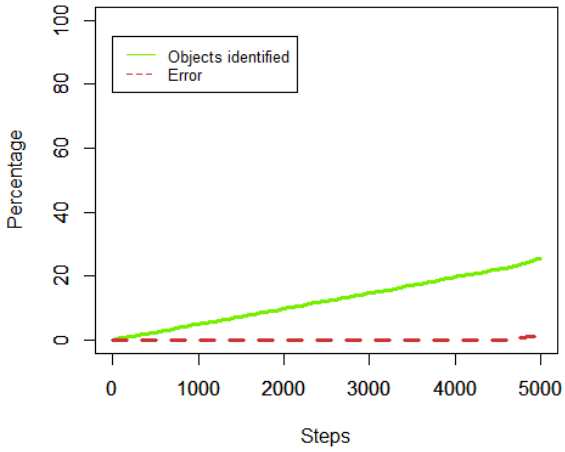

Figure 7: Random Environment with 1 agent - KNN

As mentioned above knn was never used and also the interest threshold was never increased. This led to a straight line for the coverage and showed that the fixed threshold needs to be adopted.

5.2 Dynamic Threshold and weighting

As shown in the former section, the threshold needs to be adopted when the amount of explorers changes. Small

amounts of explorers need a lower threshold in the beginning so that they use the new classification while big amounts of agents can still use a low threshold which keeps the error low. Also the start of knn needs to be earlier for small amounts of agents. As a consequence of the results from the former section, we introduced a dynamic interest threshold and knn start which depend on the amount of explorers. The formula which computes the initial interest threshold is given in equation 1.

$$threshold = max\Big(35, 55 - (\sqrt{explorers} * 5)\Big) \quad (1)$$

This formula computes a threshold of 50 when only one agent is available. The threshold than decreases with the square root until it reaches 35. We used the square root as the changes should be big during small amounts of explorers and small when many agents are available. The threshold than increases with the number of available prototypes until it reaches the maximum of 80. The threshold when using small amounts of explorers needs to be increased more rapidly than using many explorers, meaning that the amount of rising steps should be smaller. The formula to compute the amount of steps is given in the following.

$$steps = 40 + \sqrt{explorers} * 5 \quad (2)$$

The minimum amount of steps is 45 and it increases with the square root of the amount of explorers as the changes should become smaller with many agents. Based on the initial threshold and the amount of steps, we can compute the new interest threshold given a certain amount of examples for the prototypes. The formula is given below.

$$newT = min\Big(80, initT + \Big(\frac{80 - initT}{steps} * (examples - start)\Big)\Big) \quad (3)$$

When the knn start is reached this formula computes the new threshold. The fraction in the product defines how strong the threshold is increased when a new prototype example is observed and the difference in the product computes how often the threshold needs to be increased.

The last important equation calculates the starting point of knn. As discussed before the start needs to happen earlier when using only few explorers. The corresponding equation is given in the following.

$$start = 60 - \Big(3 - min\Big(3, log(explorers) * explorers\Big)\Big) * 5 \quad (4)$$

The formula is designed to start knn when 45 training examples of a prototype are available and when one explorer is available. The formula then delays the start value until 60 with a raising slope.

During our experiments we found out that it can be beneficial to use a weigthed correlation in the beginning, until half of the steps are passed. So the probability of a class is calculated using the initial correlation and the knn correlation given the following formula.

$$cor = 0.9 * knnCorr + 0.1 * oldCorr \quad (5)$$

Figure 8 shows the performance for the random environment which was the problematic case when using one agent in the initial implementation.

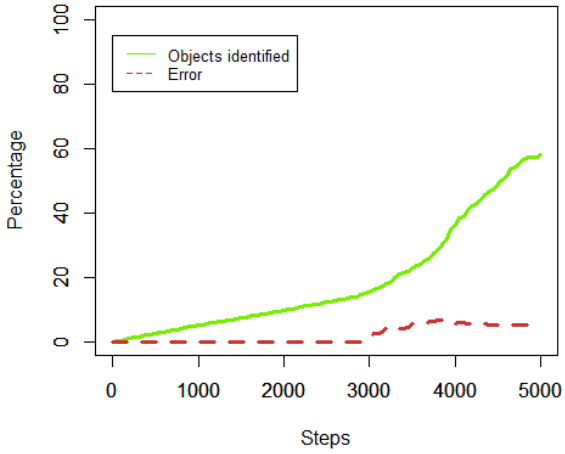

Figure 8: Random Environment with 1 agents - Dynamic KNN

The dynamic knn implementation shows big advantages over the initial knn implementation shown in figure 7.

More experiment results can be seen in tables 1, 2 and 3. They show that the coverage was drastically increased when few agents were used. For five or more agents the coverage did not vary very much because it was almost 100%. The error in both implementations was also close to each other. The variation can be explained by the random creation of the map and the random placement of agents. This explains why the new approach sometimes works better and sometimes worse. Nevertheless, the differences are not big and especially for small amounts of explorers, the dynamic implementation is superior.

Agents	Seen old	Error old	Seen new	Error new
1	25.54	1.28	58.04	5.62
2	95.0	5.26	96.6	4.72
5	99.35	4.27	98.91	5.6
8	99.35	6.46	99.57	3.06
10	99.67	1.96	99.78	4.03

Table 1: Random environment - KNN with dynamic threshold

Agents	Seen old	Error old	Seen new	Error new
1	59.0	1.69	72.6	5.51
2	95.5	2.93	95.1	4.52
5	98.7	2.23	99.5	10.75
8	99.8	4.41	99.6	4.72
10	99.7	2.81	99.6	3.61

Table 2: Donut environment - KNN with dynamic threshold

Agents	Seen old	Error old	Seen new	Error new
1	18.49	7.11	39.57	5.70
2	44.26	4.26	81.23	4.4
5	99.34	6.99	97.4	4.61
8	99.94	4.89	99.91	8.21
10	99.94	5.63	99.86	3.32

Table 3: Structured environment - KNN with dynamic threshold

As the tables only show the results after 5000 steps, we want to show the curve on the structured environment with five explorer agents. The results can be seen in figure 9.

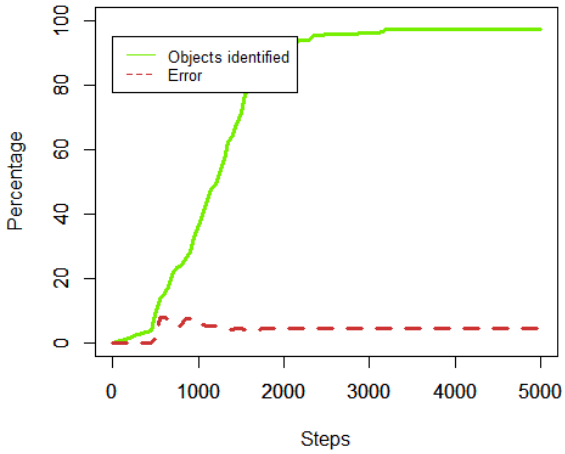

Figure 9: Structured Environment with 5 agents - Dynamic KNN

The figure shows that an almost full coverage is reached after 2000 steps. So the dynamic threshold does not decrease the exploration time.

5.3 New interest functions

This part introduces new interest functions. At first, the probability for the unknown was slightly changed to make the unknown more interesting. This has the advantage that agents are more likely to go to these objects and gain important knowledge for the classification as unknown objects can be instances of classes which have not been visited so far. From a classification point of view it is crucial to collect examples of every class as fast as possible. The following formula shows the interest function of the unknown where the factor 4 replaces the old factor 2.

$$int = tanh(4 * prob) \tag{6}$$

The formula is designed to already reach an interest of 0.5 out of 1 when the probability for the unknown is 0.1 and to reach the 1 at a probability of around 0.5. That makes it really likely to visit objects with an unknown class.

Furthermore, we changed the formula which calculates the chaotic interest. In the initial implementation the entropy for that task. We decide to decrease the entropy when the probability for a class is very high, because we are already sure about the class and do not need to go there. The new chaotic interest is computed by the following formula where max is the maximum probability in the distribution.

$$int = entropy * \left(1 - \left(max - \frac{1}{nClasses}\right)^2\right) \tag{7}$$

The original interest is decreased by a factor which depends on the maximum probability of the distribution. The higher the difference between the maximum probability and the fraction of 1 and the number of classes, the stronger the interest is decreased. That makes sense as objects with a high probability for one class can be directly classified to safe some time. Results for the donut environment and one explorer can be seen in figures 10 and 11.

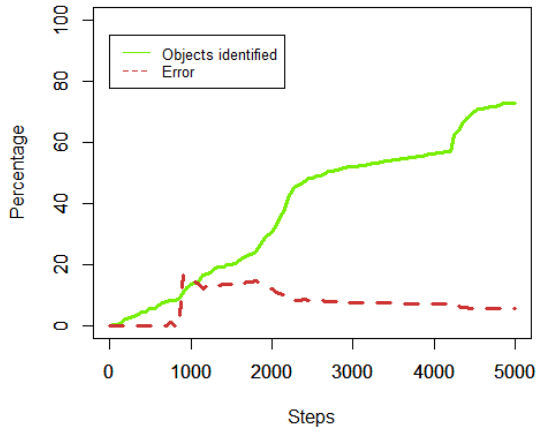

Figure 10: Donut Environment with 1 agent - Original explorer

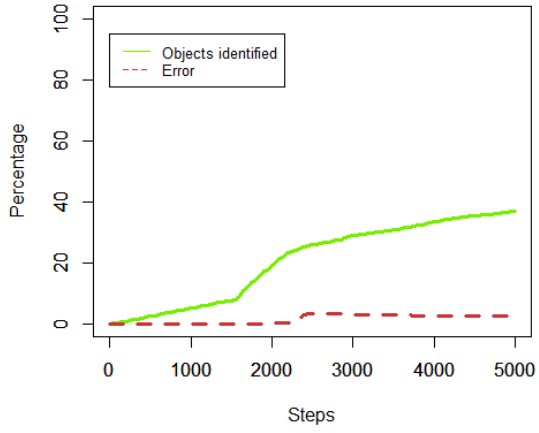

Figure 11: Donut Environment with 1 agent - New explorer

The figures show that the new explorer agent covered less objects, but with a lower error rate. The difference was even bigger when using the structured environment which contains much more objects than the donut map. Results for five agents can be seen in figure 12.

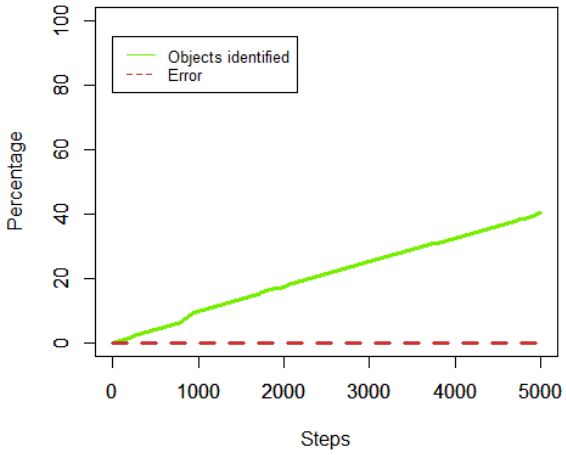

Figure 12: Structured Environment with 5 agents - New explorer

Here, even five agents were not able to cover all objects. We can conclude from the results that the new interest functions are not superior to the initial functions. Nevertheless, they keep the error lower due to the higher interest for objects with unknown classes. In that way the new functions could be used in a second explorer type which is used in addition to the one with the original interest functions.

5.4 Different explorer types

This section analyzes whether the usage of agents with different interest functions can lead to an improvement in the performance and which ratios work best. Experiments were conducted for 2,5 and 10 agents. When using two agents, one original and one new explorer were used. For five agents, we used 3 original and 2 new agents. The amount of original explorers when using ten agents can be obtained from the table. The results for the experiments are shown in tables 4 and 5.

Agents	Seen o2 n0	o1 n1	Seen o5 n0	o3 n2
Random	96.63	96.6	98.91	99.57
Donut	95.1	95.5	99.5	99.6
Structured	81.23	75.77	97.4	99.94
Agents	Error o2 n0	o1 n1	Error o5 n0	o3 n2
Random	4.72	5.31	5.60	2.73
Donut	4.52	5.76	10.75	5.22
Structured	4.4	5.84	4.61	4.97

Table 4: Different explorer ratios - 2 or 5 agents

Agents	Seen o10	Seen o3	Seen o5	Seen o7
Random	99.78	99.78	99.78	99.78
Donut	99.6	99.7	99.8	99.8
Structured	99.86	99.91	99.97	99.97
Agents	Error o10	Error o3	Error o5	Error o7
Random	4.03	2.29	4.9	4.9
Donut	3.61	5.01	4.91	8.82
Structured	3.32	3.4	3.23	6.23

Table 5: Different explorer ratios - 10 agents

The tables show that there was almost no difference in the final coverage. The only example were the approaches differed is the structured environment with 2 agents. Here it was better to take two of the explorers with the original interest functions as they are faster. Surprisingly, also the error was higher when using two different explorers. A plausible explanation for this is that the explorer with original functions is much faster than the other explorer and collected more training data for the classes at his starting point. As the other explorer is slower it did not collect much training data at a different location. Due to the imbalance of examples for classes, the fast explorer made more mistakes in the classification, but still the difference was not big.

For the experiments with 10 agents, the best performance was achieved by using 10 original explorers. This can be explained by the above discussed problem and is maybe also due to the random map creation and positioning.

Nevertheless, the data shows one big advantage of the new explorer type. This can be seen in the donut environment with 5 agents. The error using 5 original explorers is twice the error as when using 3 original and 2 new explorers. This underlines our assumption that the new explorer type lowers the risk of making classification errors in the beginning.

It is still hard to say whether the new agent type improved the results as there were random conditions and hence, always differences. Nevertheless, the amount of explorers with the original functions should be higher to keep the mapping time low.

5.5 Reclassification

During the work we decided to implement a reclassification after noticing some problems in the agent behaviour. In the original implementation agents mostly explored the whole area before the end of the simulation, which caused them to wander around the map randomly. The agents also usually made mistakes in the classification at the start of simulation, because they did not have enough data for classification.

Reclassification was implemented for the new kind of agents. When an agent gets a random target instead of a point of interest, it starts to reclassify objects near to it. When the class of an object is not same than the agent assumes it to be, it is removed from the known objects and added again as a new point of interest. The amount of removed objects is restricted to 4 to prevent agents removing too much objects at a time.

Tests for the reclassification were run with 2 original and 4 new explorers on all maps. The starting points of explorers were in the upper left corner (location 0,0) or they were randomly selected. The percentages of seen objects (F-%) and error (E-%) were collected from steps 2500 and 5000.

The results without reclassification are shown in table 6 and the results with reclassification are shown in table 7.

Map type	F-% 2500	E-%	F-% 5000	E-%
donut	98.8	10.12	99.8	10.02
random	98.48	3.64	99.67	3.6
struct.	97.66	11.06	99.94	10.81
donut corner	99.4	10.97	99.4	10.97
random corner	98.59	5.07	98.91	5.05
struct. corner	59.49	6.96	99.86	6.07
average	92.07	7.97	99.6	7.75

Table 6: Test results without reclassification or clustering

Map type	F-% 2500	E-%	F-% 5000	E-%
donut	98.5	3.96	99.4	2.92
random	99.57	2.18	99.67	1.2
struct.	91.97	3.26	99.8	2
donut corner	99.1	3.53	99.9	2.2
random corner	91.74	3.67	99.89	1.85
struct. corner	55.17	4.4	99.69	2.26
average	89.34	3.5	99.72	2.07

Table 7: Test results with reclassification

As can be seen in the tables, reclassification improved the results dramatically. The percentages of explored areas were almost the same, but the error percentages were 55% and 73% lower at steps 2500 and 5000 with reclassification than without it. The error percentages did not change much between steps 2500 and 5000 in the simulation without reclassification, but with reclassification the average of error percentage decreased by 42%.

5.6 Improved Broking

The agents sometimes left some areas unexplored during the simulation, especially corners. They also made a lot of mistakes in classification before they have seen all kind of objects. To fix that problem we decided to implement a new broking method (clustering).

This new method fits the explored area to the smallest possible rectangle and then divides it to 3,4,6,8,9 or 12 smaller rectangles assigning each rectangle to a single agent. The clustering method is used only with the new kind of agents. The reason for this decision is that the new explorers should try to find examples of all classes as fast as possible. Using also the old explorers in the clustering led to a slower exploration as some clusters only were explored by the new explorer type which is slower. By using clustering for the new agents only, the new agents can collect training data quickly and the old explorers help with classifying all objects inside the clusters.

The agents using the new clustering are believed to explore the corners and all different kinds of objects faster than in the original system leading also to a lower classification error. The tests were similar to the tests with reclassification, and the results are shown in table 8.

Based on the results, the new broking method did not improve the exploration speed, but the average of the error percentages at the end of the simulation was 26% lower with the new broking system than without it.

Map type	F-% 2500	E-%	F-% 5000	E-%
donut	99.4	10.06	99.7	9.93
random	97.17	2.91	99.78	2.83
struct.	89.74	4.08	99.89	3.98
donut corner	99.5	6.23	99.6	6.12
random corner	95.54	4.89	99.67	4.69
struct. corner	70.06	8.73	99.91	7.09
average	91.9	6.15	99.76	5.77

Table 8: Test results with clustering

5.7 Comparing the initial and final systems

Finally, we decided to compare the performance of the original system with a system which uses every of our new features. The results are shown in table 9.

Map type	F-% 2500	E-%	F-% 5000	E-%
donut	98.4	9.55	99.7	1.91
random	97.39	2.12	99.78	0.98
struct.	66.69	1.84	99.71	2.41
donut corner	96.3	2.18	99.7	2.61
random corner	94.46	3.57	99.67	2.51
struct. corner	73.14	5.39	98.71	2.29
average	87.73	4.11	99.55	2.12

Table 9: Test results with reclassification and clustering

As can be seen in table 9, the error percentages are much lower than in the system without reclassification or clustering whose results are shown in table 6. The error percentage of the final system is 48.44% lower at step 2500 and 73% lower at step 5000. At the end of the simulation almost the full environment was explored without any exceptions. The error percentage also decreased significantly between steps 2500 and 5000, going below a final error of 1% at its best.

To compare the results graphically with the results of the initial system, a simulation with 6 agents was run in the initial system. The resulting graph is shown in figure 13.

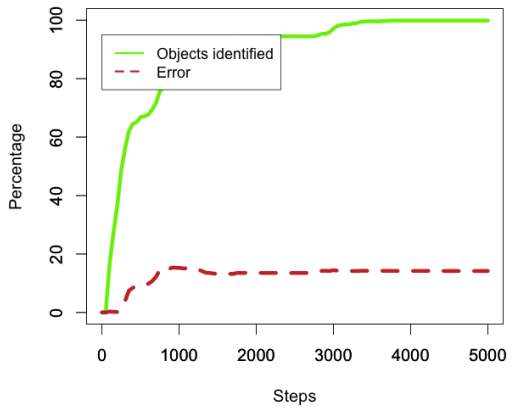

Figure 13: Structured map with 6 explorers starting from random points in the initial system

The most notable difference between these graphs is the error percentage, which is 14.2% in figure 13 and 2.29% in figure 14. That is a caused by the new features, because the error percentages are nearly same around step 1500. The knn makes agents to classify objects more precisely, and clustering and reclassification help to overcome the cold-start problem.

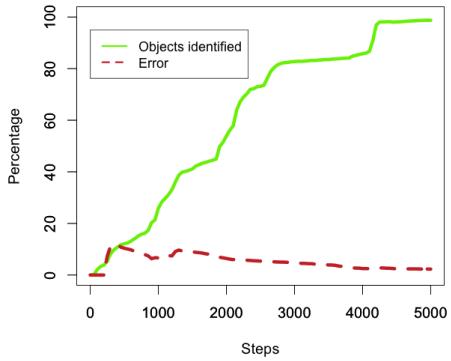

Figure 14: Structured map with 2+4 explorers starting from corner with new features

The exploration speed with our final system is not as good as in initial system, because 4 of 6 agents are quite slow in exploring, but we consider the final version as superior as the trade-off of time and classification error is much better.

6. CONCLUSION

Within this paper, we showed that a partial exploration with class predictions for unknown objects can be superior to the classical approach as it is much faster and solely introduces a small and reasonable classification error. In this work, we built on the system of Macedo et al. and improved it in two ways. First, knn classification and clustering were used to drastically reduce the error and second, the time, when no interest points were left, is used more reasonable by reclassifying objects also leading to a smaller error.

We see the contribution of the paper in the tackling of the cold-start problem. Without training data it is really hard to build a reliable classification and it is difficult to find good thresholds which work well for all settings. By making them dependent on the amount of explorers, seen prototypes and using a reclassification, we found a robust and reliable solution for this problem in our scenario. We are convinced that the presented approach can be applied to different scenarios in a similar way and helps to overcome the problems of a cold-start.

6.1 Future Work

Whether the results of this work are superior in terms of a reduced error, more future work can be done to further improve the system or to make the scenario more realistic.

- The experimental setup could be made more realistic by adding more classes and more attributes. Also more layouts and different object proportions can be tested. For example really rare events could be hardly detected by the current system. Also changes in explorer attributes like speed, view range or investigation time are imaginable and open the way for the development of heterogeneous agents which was only briefly studied in this work.
- Improvements are also possible in the classification. The current classification only uses attributes of objects to compute a correlation to known classes. It could be analyzed whether the location of an unknown objects helps to predict the class, e.g. by analyzing which objects are close to the unknown object on the map as trees are more likely to stand beside trees than in the water or in houses.
- Research can also be conducted in the brokering. It might be beneficial to not send an explorer to the most relevant point, but to the second most relevant point. This might be the case when the second most relevant point is close to other very relevant points. By sending explorers to interesting clusters instead of interesting points, distances could be reduced.

More work related to the exploration of unknown environments could be proposed and conducted to improve the exploration performance. If the reader has any suggestions or doubts about this paper, feel free to contact the authors using the given e-mail addresses or check the code at Github.[1]

REFERENCES

[1] E. Alonso, M. D'inverno, D. Kudenko, M. Luck, and J. Noble. Learning in multi-agent systems. *The Knowledge Engineering Review*, 16(03):277–284, 2001.

[2] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 476–481. IEEE, 2000.

[3] L. Macedo and A. Cardoso. Exploration of unknown environments with motivational agents. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 328–335. IEEE Computer Society, 2004.

[4] L. Macedo, M. Tavares, P. Gaspar, and A. Cardoso. Uncertainty and novelty-based selective attention in the collaborative exploration of unknown environments. In *Portuguese Conference on Artificial Intelligence*, pages 521–535. Springer, 2011.

[5] A. Poncela, E. J. Perez, A. Bandera, C. Urdiales, and F. Sandoval. Efficient integration of metric and topological maps for directed exploration of unknown environments. *Robotics and Autonomous Systems*, 41(1):21–39, 2002.

[6] S. Thrun. Exploration in active learning. *Handbook of Brain Science and Neural Networks*, pages 381–384, 1995.

[7] S. Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1–35, 2002.

[8] S. B. Thrun. Efficient exploration in reinforcement learning. 1992.

[1]https://github.com/SebastianRehfeldt/collaborative-exploration