Environment-Specific Novelty Detection

Stephen Marsland[*] **Ulrich Nehmzow**[**] **Jonathan Shapiro**[*]

[*]Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
UK
{smarsland, jls}@cs.man.ac.uk

[**]Department of Computer Science
The University of Essex
Wivenhoe Park
Colchester CO4 3SQ
UK
udfn@essex.ac.uk

Abstract

Novelty detection, recognising features that differ from those that are normally seen, is a potentially useful ability for a mobile robot. Once a robot can detect those features that are novel the amount of learning that has to be done can be reduced (as only new things need to be learnt), the attention of the robot can be focused onto the new features, and the robot can be used as an inspection agent.

However, features that are novel in one place could be completely normal elsewhere – for example, tables and chairs are usually seen in offices, but very rarely seen in corridors. This paper suggests a method by which a set of novelty filters can be trained for different environments and the correct filter autonomously selected for the environment that the robot is currently travelling in. The method can also extend itself, so that further environments that are seen by the robot can be added without any retraining.

1 Introduction

Recognising stimuli that differ from the usual inputs in some way is a very useful ability for both natural and artificial learning agents. This capability is known as novelty detection. For animals it can be a crucial survival instinct, enabling them to avoid potential predators, while for robots and other learning agents it can help to select particular inputs of interest, reducing the computational cost of dealing with the world.

Neural networks that can detect novelty have been used to highlight potential problems in fields such as medical diagnosis and machine fault detection. In these cases there is a lot of data where the result of the test is negative (no disease diagnosed or no machine fault), but relatively few of the important class that the network should detect. This means that normal neural network training is not suitable, as many instances of the disease may be missed. The novelty detection approach operates by having the neural network learn a model of the 'normal' data that does not show any examples of the class that should be detected, and having the novelty detector highlight inputs that do not fit into the pattern of the training set.

There are two important properties that the training set for the novelty detector should have. It should contain no examples of the inputs that should be detected, otherwise this will be learnt and so these inputs will not be found to be novel, and it should contain examples of every possible kind of 'normal' inputs, otherwise these inputs will be found to be novel.

In previous work (Marsland et al., 2000) the novelty detection approach has been used to enable a mobile robot to be used as an inspection agent. A robot equipped with a novelty filter can learn a model of that environment, perceiving its environment through whatever sensors it is equipped with, and then explore other environments, highlighting features that were not found in the original training environment.

However, there are a few problems with this basic approach. When the robot is training there is no guarantee that the training set of inputs will satisfy the two required properties of the input set. This problem is exacerbated by the fact that most novelty detection algorithms are not capable of learning on-line, so that further data cannot be added at a later date. It would also be nice if the filters could quantify the amount of novelty, so that inputs that are only fairly novel – a few similar inputs have been seen during training, but not that many – can be marked, but with less emphasis than completely novel features. Finally, in some cases features that are perfectly normal in one part of an environment should be found to be novel in other places. One example of this is that when exploring an office environment, chairs are perfectly usual within offices, but rarely found in corridors.

The novelty filter that is described in this paper is a proposed solution to these problems. The filter is based on a neural network that is capable of growing during use, so that it can be used for continuous learning. A part of the filter is a set of habituating synapses, so that the filter can quantify the amount of novelty in the current input with respect to the inputs that were seen during training. This also means that the filter has some robustness to incorrect information in the training set – if there are a small number of inputs in the training set that should not be there, they will still be found to be novel after training because they have been seen only infrequently. Finally, an extension to the algorithm is described that allows multiple novelty filters to be trained in different environments and the correct filter for the current inputs to be selected from those available. If none of the filters is suitable then a new filter can be created and trained, meaning that the system is capable of boot-strapping during the training phase.

2 Related Work

There have been a number of novelty detection techniques proposed in the literature. A more complete review is given in (Marsland, 2001). The first was Kohonen's Novelty Filter (Kohonen and Oja, 1976, Kohonen, 1993). This is an autoencoder network that is trained using back-propagation of error, so that the network extracts the principal components of the input. The network is trained on a dataset and, after training, any input presented to the network produces one of the learnt outputs, and the bitwise difference between input and output highlights novel components of the input.

(Ypma and Duin, 1997) proposed a novelty detection mechanism based on the self-organising map. They describe a number of measures by which the goodness of a SOM with respect to a particular dataset can be evaluated. In particular, they measure the average quantisation error over the dataset, and also measure how far away from each other map units that respond to similar inputs are. By training the SOM on data that are known to be normal, and then evaluating the measures on a new dataset, it can be seen whether or not the new dataset fits the same distribution as the data that generated the SOM.

Another approach using the SOM is to calculate the distance of the winning neuron from neighbourhoods that fired when training data known to be normal was introduced, and counting as novel those inputs where the distance is beyond a certain threshold. This method was used by (Taylor and MacIntyre, 1998) to detect faults when monitoring machines. The network was trained on data taken from machines operating normally, and data deviating from this pattern was taken as novel. This is a common technique when faced with a problem for which there is very little data in one class, relative to others. Examples include machine breakdowns (Nairac et al., 1999, Worden et al., 2000) and mammogram scans (Tarassenko et al., 1995). Often, supervised techniques such as Gaussian Mixture Models or Parzen Windows are used, and the problem reduces to attempting to recognise when inputs do not belong to the distribution which generates the normal data (Bishop, 1994). This is the problem of kernel density estimation. The method proposed by (Taylor and MacIntyre, 1998) relies very strongly on the choice of threshold and on the properties of the data presented to the network, which must form strictly segmented neighbourhood clusters without much spread.

Growing networks such as Adaptive Resonance Theory (ART) (Carpenter and Grossberg, 1988) can be used to define as novel those things that have never been seen before, by using a new, uncommitted node to represent them.

3 The On-line Novelty Filter

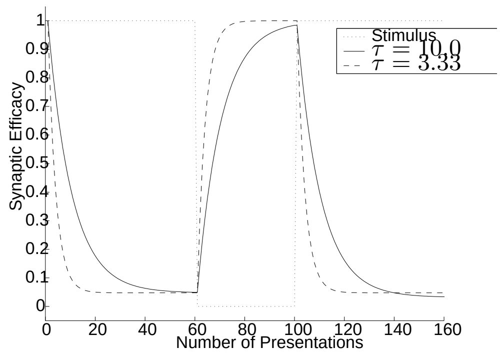
3.1 Habituation

Habituation is a reversible reduction in the behavioural response to a stimulus when it is presented repeatedly. It enables the animal to ignore stimuli that are seen often, so that it can concentrate on other, potentially more important, stimuli. Habituation is thought to be one of the fundamental mechanisms of adaptive behaviour, and can be seen in animals as diverse as the sea slug *Aplysia* (Bailey and Chen, 1983), cats (Thompson and Spencer, 1966), toads (Wang and Arbib, 1992) and humans (O'Keefe and Nadel, 1978). In contrast to other forms of behavioural decrement, such as fatigue, the response can be restored to its original level without the organism resting by introducing a change in the stimulus. An overview of the effects and causes of habituation can be found in (Thompson and Spencer, 1966).

There have been several attempts to model the effects of habituation computationally and to explain the interaction between habituation and dishabituation, the process whereby an habituated response returns to its original strength. (Groves and Thompson, 1970) suggested that dishabituation was an instance of sensitisation and therefore an independent construct that interacts with habituation to produce a net response. Their model was used by (Stanley, 1976) to simulate habituation data from experiments of the spinal cord of a cat. He described the decrease of synaptic efficacy y by the first-order differential equation

$$\tau \frac{dy(t)}{dt} = \alpha \left[y_0 - y(t) \right] - S(t), \tag{1}$$

where y_0 is the initial value of y, $S(t)$ is the external stimulation and τ is a time constant governing the rate of habituation, while α controls the recovery rate. A graph showing the effects of this equation is given on the left of figure 1. The values of the variables given in figure 1 are the ones that were used.



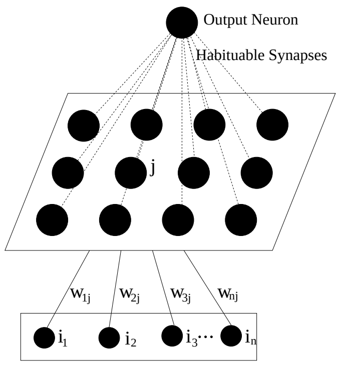


Figure 1: *Left:* The dynamics of habituation using equation 1 ($\alpha = 1.05$, $y_0 = 1$). *Right:* A diagram of the on-line novelty filter.

3.2 Using Habituation in a Novelty Filter

The effect of habituation is to filter out those perceptions that have been seen before, meaning that only novel stimuli are noticed. This is exactly the behaviour that a novelty filter should demonstrate. Habituation allows novelty to be defined more specifically as those things which have not been seen in the current context. The novelty filter described here operates by learning an on-line, adaptive representation of the current environment, testing whether the neural network has already habituated to each new perception. Habituation also allows the novelty of a stimulus to be evaluated, so that the novelty reduces with perception over time.

Before it is known whether a particular input is novel, it needs to be recognised. A schematic of the novelty filter that is described in this paper is shown on the right of figure 1. The filter uses a clustering network to identify the current perception. Each node of the clustering network is attached to an output neuron by a synapse that habituates with use (controlled by equation 1). This means that the first time a node fires the output is strong (the current input is novel), but as that node of the network fires frequently its synapse habituates and so the input is found to be normal clustering network classifies the input, selecting the node whose weights best match the current input vector. The output of this winning node is propagated along the habituable synapse, which modifies the strength of the signal. If that node has not fired before, or fired only rarely, then the signal is passed on without attenuation, and the activity at the output neuron is high. If that node has fired often, meaning that the perception is not novel, then the efficacy of the synapse is low and so the output neuron receives a very weak signal. As well as the synapse connecting the winning node to the output habituating, the synapses of nodes that are neighbours of the winning node in the map space also habituate, although to a lesser extent.

This abstract representation of the novelty filter does not require that any particular clustering network be used. The network needs to cluster similar inputs together, be able to deal with unlabelled data and should be capable of operating on-line. Previous work (Marsland et al., 2000) has shown that networks such as the Self-Organising Map are ideal except that they are not capable of on-line operation. A new neural network was therefore devised that can add new nodes according to the data that is presented, so that it can be used on-line. This network, termed the Grow When Required (GWR) network, is the subject of the next section.

3.3 The Grow When Required (GWR) Algorithm

This section first describes the GWR network and then gives details of the algorithm (section 3.3.1). The network decides when to add nodes according to the activity of the best-matching node. If a node matches an input well, then the activity of that node is close to 1. There are two reasons why the activity of the node could be low – either the node is still being positioned by the learning rule, having been added recently, or there is a mismatch, and that node is already representing another feature. If the node is a new one then it will not have fired often, and so the habituation counter that is attached to the node for the novelty detection part of the filter will be high. In that case the node should be trained to be a better match to the input.

So, if the habituation value is close to 0, then the node should be well placed in the map field and hence the activity should be high. If this is not the case then we need to introduce an extra node to match the current input better. This node is added between the winning node, which caused the problem, and the input, with the weights of the new node being initialised to be the mean average of the weights for the best matching node and the input. If the node is well placed then a neighbourhood connection is put between the winning node and the second best matching node (if it does not already exist), and the weight vectors of all nodes in the neighbourhood of the winning node (that is, nodes that have a direct neighbourhood connection to the winning node) are updated.

Thus, two thresholds are needed to decide whether or not to insert a node on the current iteration: a minimum activity, a_T, below which the current node is not considered to be a sufficiently good match, and a maximum habituation value, above which the current node is not considered to have learnt sufficiently well. In practice, the value of the habituation threshold does not seem to matter very much, and is usually set to 5 presentations. The value of a_T does make a considerable difference. If the value is set very close to 1 then more nodes will be produced to make a better match with the inputs.

3.3.1 The Algorithm

Let A be the set of map nodes, and $C \subset A \times A$ be the set of connections between nodes in the map field. Let the input distribution be $p(\boldsymbol{\xi})$, for inputs $\boldsymbol{\xi}$. Define $\boldsymbol{w_c}$ as the weight vector of node c.

3.3.2 Initialisation

The network is initialised using:

- Create two nodes for the set A,

$$A = \{c_1, c_2\} \tag{2}$$

  with their weights $\boldsymbol{w_{c_1}}$, $\boldsymbol{w_{c_2}}$ initialised randomly from $p(\boldsymbol{\xi})$

- Define C, the connection set, to be the empty set,

$$C = \emptyset \tag{3}$$

- Let $y(0) = y_0$

3.3.3 Iteration

Each iteration of the algorithm follows the following steps:

1. Generate a data sample $\boldsymbol{\xi}$ for input to the network

2. For each node in the network, calculate the distance from the input $\|\boldsymbol{\xi} - \boldsymbol{w}_i\|$

3. Select the best matching node, and the second best, that is, the nodes $s, t \in A$ such that

$$s = \arg\min_{c \in A} \|\boldsymbol{\xi} - \boldsymbol{w}_c\| \tag{4}$$

   and

$$t = \arg\min_{c \in A/\{s\}} \|\boldsymbol{\xi} - \boldsymbol{w}_c\|, \tag{5}$$

   where $\boldsymbol{w}_c$ is the weight vector of node c.

4. If there is not a connection between s and t, create it with age 0

$$C = C \cup \{(s,t)\}, \tag{6}$$

   otherwise, set the age of the connection to 0.

5. Calculate the activity a of the best matching unit (s),

$$a = \exp(-\|\boldsymbol{\xi} - \boldsymbol{w}_s\|) \tag{7}$$

6. If we should add a node, i.e., if activity $a_s <$ activity threshold a_T and habituation $y_s(t) <$ habituation threshold h_T

   - Add the new node, r

$$A = A \cup \{r\}. \tag{8}$$

   - Create the new weight vector, setting the weights to be the average of the weights for the best matching node and the input vector

$$\boldsymbol{w}_r = (\boldsymbol{w}_s + \boldsymbol{\xi})/2. \tag{9}$$

   - Insert edges between r and s and between r and t

$$C = C \cup \{(r,s), (r,t)\}, \tag{10}$$

   - Remove the link between s and t

$$C = C / \{(s,t)\} \tag{11}$$

7. Adapt the positions of the winning node and its neighbours, i, that is the nodes to which it is connected.

$$\Delta \boldsymbol{w}_s = \epsilon_b(\boldsymbol{\xi} - \boldsymbol{w}_s) \tag{12}$$

$$\Delta \boldsymbol{w}_i = \epsilon_n(\boldsymbol{\xi} - \boldsymbol{w}_i) \tag{13}$$

   where the learning rates are such that $0 < \epsilon_n < \epsilon_b < 1$

8. Age edges with an end at s.

$$age_{(s,i)} = age_{(s,i)} + 1. \tag{14}$$

9. Habituate the winning node and its neighbours using

$$\tau \frac{dy_i(t)}{dt} = \alpha[y_0 - y_i(t)] - S(t), \tag{15}$$

   where $y_i(t)$ is the strength of synapse i, y_0 is the initial strength, and $S(t)$ is the stimulus strength, usually 1. α and τ are constants controlling the behaviour of the curve. The winner habituates faster than its neighbours. Values of the parameters used in the experiments are $\alpha = 1.05$, $y_0 = 1$ and $\tau = 3.33$ for the winning node, $\tau = 10.0$ for the neighbours.

10. Check if there are any nodes or edges to delete, i.e., if there are any nodes that no longer have any neighbours, or edges whose age is greater than some constant $a_{\max}$.

3.4 Selecting Different Novelty Filters

The second part of the system that is described in this paper is the part that enables the robot to choose which, if any, of a set of previously trained filters should be used, or whether a new filter should be trained. As the robot travels through an environment it monitors how well each perception fits into the model of each of the novelty filters that has been trained. At the end of a run through that environment the robots makes a choice about which environment it is in. At this stage a set of different behaviours could be used to decide how the robot should react.

A vector of 'familiarity indices' is used to keep a record of how familiar each of the different trained novelty filters finds the current perceptions of the robot. This familiarity vector (with one element for each of the m trained novelty filters) is updated after each perception has been presented to all of the novelty filters, each of which has produced a novelty value n.

All of the elements of the familiarity vector are initialised to be $1/m$. For each input to the novelty filters the following steps are taken:

- compute the novelty value n_i (i.e., the output of the novelty filter) for the current filter, i
- update the element of the familiarity vector $\mathbf{f}$ for that network:

  $$\mathbf{f}_i = \mathbf{f}_i - c \times n_i, \tag{16}$$

  where c is a scaling constant
- update all the other elements so that the sum of the elements remains normalised:

  $$\mathbf{f}_j = \mathbf{f}_j + \frac{c \times n_i}{n_i - 1}, \ \forall j \neq i \tag{17}$$
- repeat for all the other novelty filters

In the experiments reported in the next section a value of $c = 0.1$ was used. Investigations showed that the value was not critical, although obviously it does affect how quickly the familiarity vector responds to inputs that the filters find to be novel.

Once the robot had travelled through the environment a decision was made about which environment it was. If one element of the familiarity index was significantly larger than the others (i.e., one familiarity index was above 0.7), then the corresponding environment was taken as the one being explored. However, the algorithm also stores the accumulated novelty of each novelty filter as the robot explores it. If the best-matching novelty filter has very high accumulated novelty then it could be that it is not actually a good match to the inputs, merely a better match than the others. In this case a new novelty filter should be made and trained.

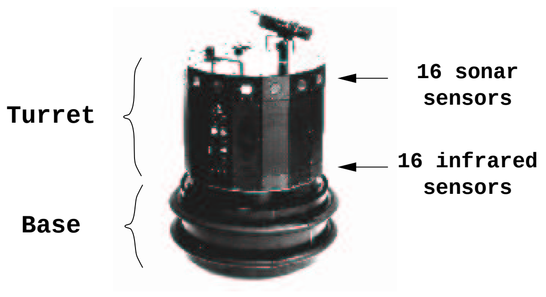


Figure 2: The Nomad 200 robot used in the experiments.

In the case where no environment is obviously more familiar than any of the others it is assumed that this is because the robot has just explored a novel environment. The accumulated novelty should also be high in this case. If the environment was novel then it would be suitable to generate a new filter and further explore the environment, training the filter. In this way the algorithm could extend itself as required.

4 Experimental Results

4.1 Training a Novelty Filter

This section describes how a novelty filter can be trained. The robot (shown in figure 2) explores two small environments, 10 m sections of corridor. In each case the robot used a pre-trained wall-following behaviour based on infra-red sensors to travel through an environment, taking sonar scans as it travelled and produced an input vector by taking the average of these readings over the last 10 cm of travel. This input vector was then presented to the novelty filter, which categorised it and produced an output of how novel that perception was according to the strength of the habituation synapse for the best-matching node.

In the first experiment the robot, initially equipped with an uninitialised novelty filter, travelled along a 10 m section of corridor. Figure 3 shows the filter learning about this environment (labelled environment A) during three learning runs. Spikes show the amount of novelty found in each input, with high spikes denoting novel features and very small spikes completely normal inputs. Initially the filter finds all perceptions novel, shown by the burst of spikes. However, it rapidly learns to recognise the wall that is seen on either side, whereupon the only novelties are around the area of the doorway on the right-hand side of the robot. By the second run these have mostly been learnt, and in the third run nothing is found to be novel.

This trained network was then used in a modified version of the environment, environment A* in figure 4. The

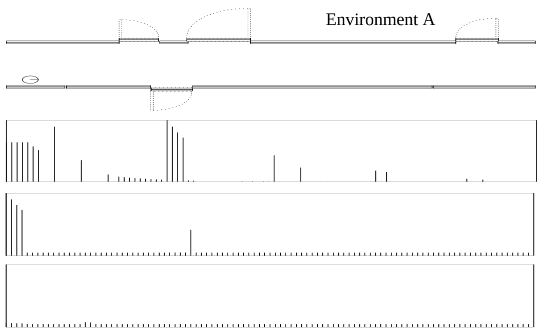


Figure 3: The novelty filter learning about environment A with no initial training. Spikes show the output of the novelty filter, with a high spike denoting a novel input and a low spike a normal input. During the first run many features are found to be novel, but these are learnt about and are not found novel by the third run.

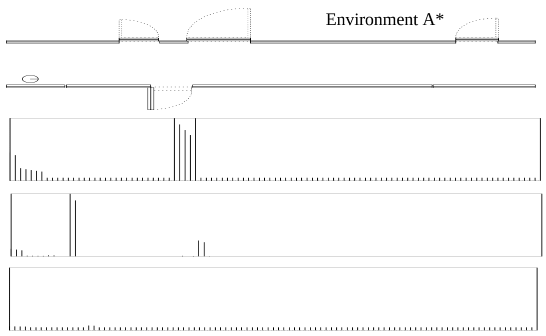


Figure 4: The novelty filter learning about environment A* after training in environment A. It can be seen that the only place where novelty is found (i.e., there are high spikes in the graph) on the first run is the area around the now-open doorway. In the second run, the spikes are caused by a crack in the wall, which is only detected occasionally, but that, when detected, dominates the sonar readings.

door on the right of the robot was opened, which meant that the sonar signals reflected off a wooden barricade amount 1.5 m further away. A cardboard box was placed in the doorway so that the infra-red sensors that controlled the wall-following motion could see it, but the sonar sensors could not. Figure 4 shows the results of this experiment. Only the area around the now-open door is found to be novel, and this is learnt after the second run through the environment.

4.2 Selecting a Suitable Filter

Using the technique described in the previous section four separate GWR-based novelty filters were trained. The first was trained in environment A, a schematic of which is shown at the top of figure 3, the second in environment A* (figure 4), and the third in a very similar area of corridor, labelled environment B. Finally, a different type of corridor in another part of the building was used as environment C. This corridor is wider and is built of different materials and is wider than the others. The appearance of the three different corridors used can be seen in figure 5.

The training in each environment was as described in section 4.1. The robot made three training runs in each of the environments, sampling the environment with its sonar sensors and presenting an average sonar reading over the last 10 cm of travel every 10 cm. Each network was initially completely untrained, and after the three training runs the filter had stopped finding any features in the environment novel. An insertion threshold of $a_T = 0.9$ was used. Since four novelty filters were trained, one in each of the environments A, A*, B and C, so the elements of the familiarity vector were initialised to $\frac{1}{4}$.

After training each of these filters the robot was exposed to an unknown environment. Five different environments were used for this purpose, each of the four used for training (A, A*, B and C) and a control environment that was completely different. This novel environment was part of the robot laboratory, which is wider than the corridors and has obstacles placed in the path of the robot, so that the perceptions were very different to those in the corridor environments.

Once the robot was placed in an environment it explored that environment using the wall-following behaviour to follow the wall to the right of the robot. As the robot travelled in this test environment the algorithm had to decide which environment the robot was exploring – one of the known environments or the control environment. As in the training, the robot moved 10 cm taking sonar scans as it moved, and after moving computed the average sonar scan over the 10 cm. It presented this average sonar scan as input to each of the novelty filters, which produced a novelty value n for the perception. The algorithm described in section 3.4 was then used to update the familiarity indices of each of the environments and the robot moved on another 10 cm.

4.2.1 Testing the Complete System

Five testing runs were performed in each of the five environments, the environments A, A*, B and C and the control environment. Table 1 shows the results of the experiments averaged over five testing runs. It can be seen that in each case the algorithm picks the correct network (shown in bold) and that the familiarity indices for the other environments are all small. Where the algorithm is tested in the control environment, which does not have a trained filter, all four of the filters have similar scores of about $\frac{1}{4}$.

Figure 6 shows a sample output when the algorithm is run in each of the environments. It can be seen that for some of the environments it takes a long time before any one of the networks is clearly the winner, while for others

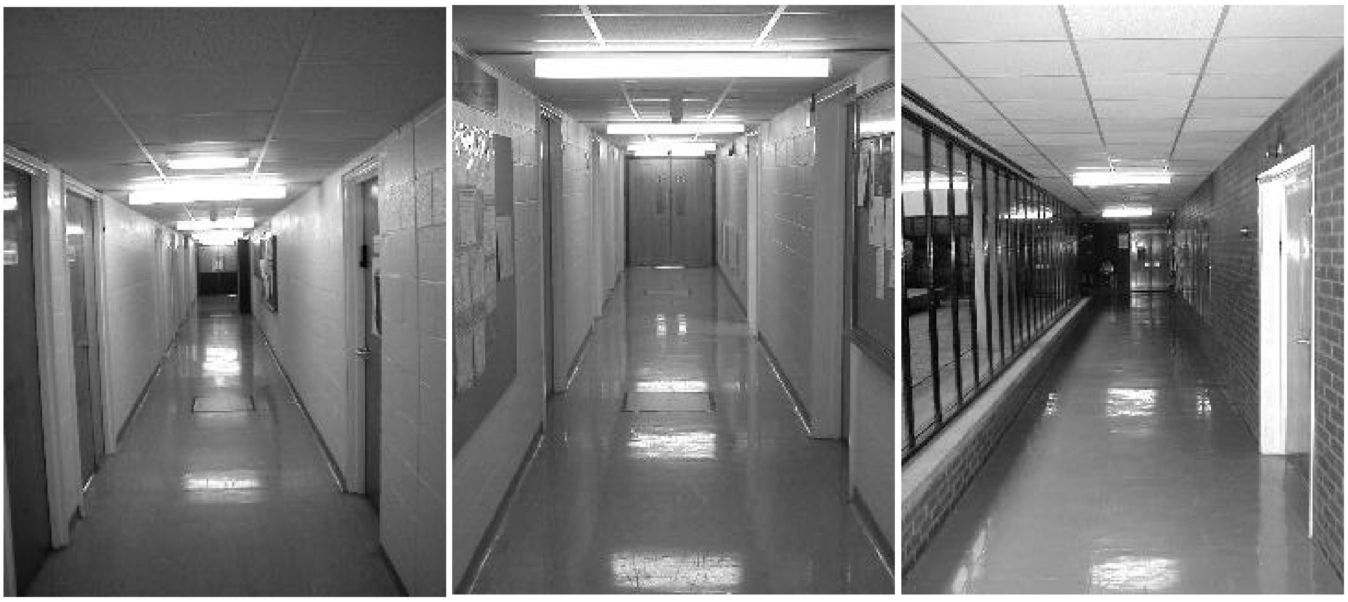


Figure 5: Photographs of environments A (*left*), B (*centre*) and C (*right*). Environments A and B are similar sections of corridor, while environment C is in a different part of the building and has brick walls instead of breezeblock.

Training	Testing Environment				
Environment	A	A*	B	C	Control
A	**0.809** ± 0.055	0.042 ± 0.031	0.115 ± 0.038	0.026 ± 0.029	0.255 ± 0.067
A*	0.157 ± 0.150	**0.723** ± 0.191	0.014 ± 0.0128	0.106 ± 0.103	0.242 ± 0.073
B	0.012 ± 0.009	0.073 ± 0.078	**0.899** ± 0.093	0.016 ± 0.032	0.245 ± 0.045
C	0.068 ± 0.060	0.102 ± 0.082	0.034 ± 0.024	**0.796** ± 0.099	0.263 ± 0.102

Table 1: The familiarity index for each of the environments for each of the trained networks, averaged over five runs. Each table entry gives mean ± standard deviation.

the winner is apparent very quickly. It is particularly interesting that when the robot explores environment C, which is very different to the others, the filter for environment C is initially very low in familiarity, and only late in the run does the correct hypothesis overtake the others. This is probably because the start of this environment contains a door very similar to those seen in the other environments and only later is the different brick, etc. apparent.

In environment B, initially all of environments A, A* and B (which do look similar) are equally likely, and it takes quite a while to settle for environment B. It does this towards the end, where the perceptions of the boxes on the wall appear. These boxes do not appear in any of the other environments. To differentiate between environments A and A*, the algorithm has to wait until the perceptions of the doorway that can be open or closed are seen. However, for the novel environment, none of the possible environments were ever seen to be similar to the perceptions. This is very encouraging.

As a further test of the system, a different type of testing was performed. Here, one of the four environments was missed out of the bank of trained filters, so that there were only three trained filters. The results of this are shown for environments B and C in figure 7. It can be seen that in the case where the robot explores environment B, environments A and C are considered equally likely, but environment A*, which contains the open door, is unlikely. In this kind of case the algorithm also looks at the total amount of novelty that has been found in the environment. This is computed by summing the novelty found at each timestep. For testing in environment B this was at least 14.4, as compared to under 5.2 for non-novel environments.

In the case where environment C is explored, environment A is initially favoured, presumably because it also has a door on the right-hand side of the robot, as does environment C, but environment B becomes more likely as the run progresses. The minimum amount of novelty found during testing in environment C in any of the five runs was 19.8. In both of these cases it is very unclear which of the environments was more likely, and the total novelty found during the testing run would be sufficiently high to suggest that this was a novel environment and

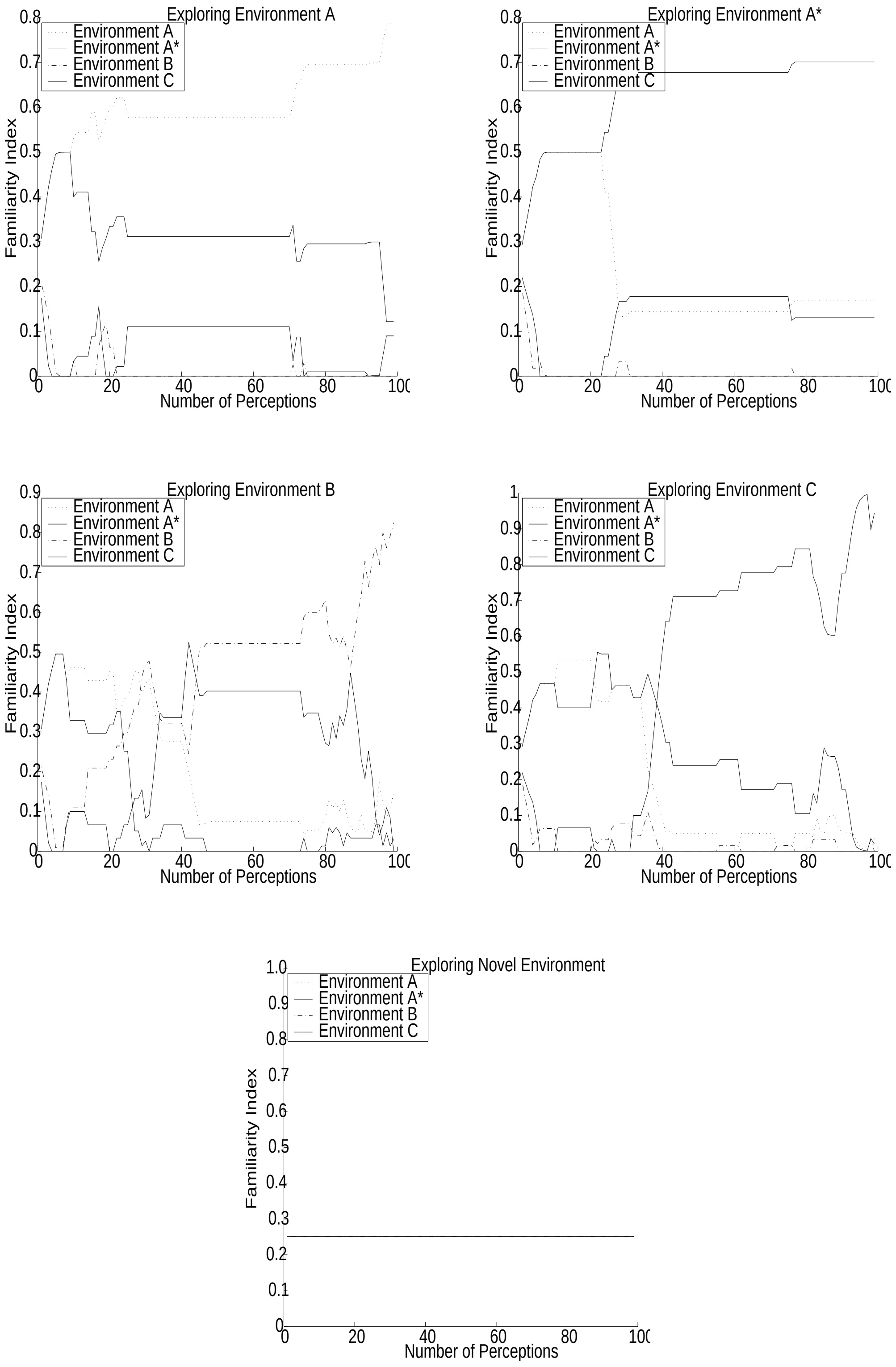


Figure 6: Plots of the familiarity indices for the trained novelty filters during sample runs in each of the five test environments. The $x-$axis shows the number of perceptions of the environment. *Top left:* Environment A. *Top right:* Environment A*. *Middle left:* Environment B. *Middle right:* Environment C. *Bottom:* Novel (control) environment. In all cases the correct environment is selected.

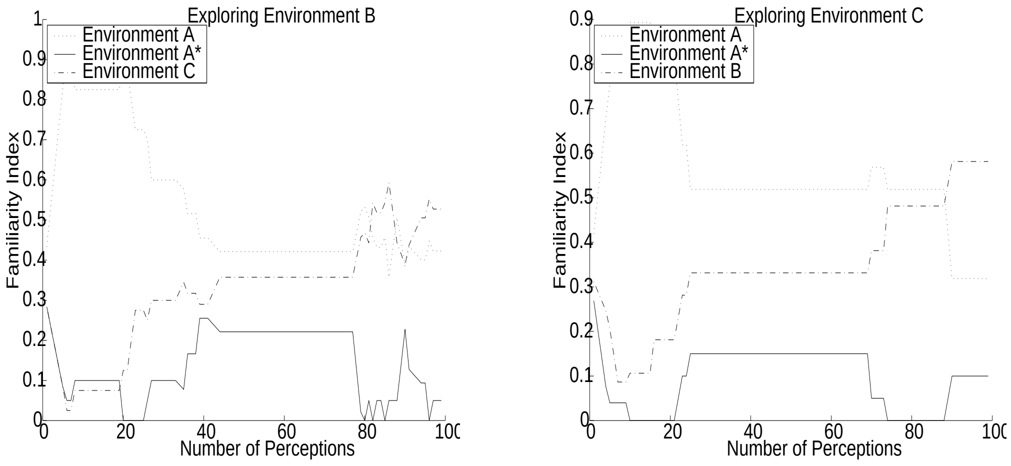


Figure 7: *Left:* Familiarity indices for the novelty filters trained in A, A* and C, for tests in environment B. *Right:* Testing on environment C after training on A, A* and B. All of the trained filters are found to be unlikely in both cases.

so a new filter should be trained.

5 Discussion and Conclusions

The novelty filter that is described in this paper is capable of learning on-line about a series of environments as the robot travels through them. This means that the robot can be used as an inspection agent, exploring a set of environments that are known to be normal (that is, to contain no unusual features or features that are known to be possible problems). Once this training set has been learned, the robot can be used to inspect further environments, highlighting any inputs that do not fit into the training set, and so are potential faults. One benefit of the novelty filter proposed here is that it learns on-line, which means that if any important feature is missing from the training set it can be added in at a later date without any requirement for retraining on the rest of the data.

The extension to the novelty filter system that has been described here means that a bank of novelty filters can be trained for the different places that the robot can visit. In this way the fact that each of these different places has different properties, and that different things may be found to be novel in each is encoded. The robot can then decide autonomously which of its bank of filters should be used to evaluate the current inputs, or whether none of them is sufficient and a new filter should be trained. It has been demonstrated that in the experiments presented here the correct filter was selected 100% of the time.

In this paper the only inputs that have been used have been the sonar sensors of the robot. It would be necessary to use more sensors, especially cameras, in order to enable the robot to act properly as an inspection agent in any sort of real world environment. While this has been considered for simple camera images (Marsland et al., 2001), it has not been covered for more complex images, nor has the question of sensor fusion yet been addressed.

Acknowledgements

This work was supported by an EPSRC studentship.

References

Bailey, C. and Chen, M. (1983). Morphological basis of long-term habituation and sensitization in *aplysia*. *Science*, 220:91–93.

Bishop, C. M. (1994). Novelty detection and neural network validation. *IEEE Proceedings on Vision, Image and Signal Processing*, 141(4):217–222.

Carpenter, G. A. and Grossberg, S. (1988). The ART of adaptive pattern recognition by a self–organising neural network. *IEEE Computer*, 21:77 – 88.

Groves, P. and Thompson, R. (1970). Habituation: A dual-process theory. *Psychological Review*, 77(5):419–450.

Kohonen, T. (1993). *Self-Organization and Associative Memory, 3rd ed.* Springer, Berlin.

Kohonen, T. and Oja, E. (1976). Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 25:85–95.

Marsland, S. (2001). *On-line Novelty Detection Through Self-Organisation, With Application to Inspection Robotics.* PhD thesis, Department of Computer Science, University of Manchester.

Marsland, S., Nehmzow, U., and Shapiro, J. (2000). Novelty detection on a mobile robot using habituation. In *From Animals to Animats: Proceedings of the 6th International Conference on Simulation of Adaptive Behaviour (SAB'00)*, pages 189 – 198. MIT Press.

Marsland, S., Nehmzow, U., and Shapiro, J. (2001). Vision-based environmental novelty detection on a mobile robot. In *Proceedings of International Conference on Neural Information Processing (ICONIP'01)*.

Nairac, A., Townsend, N., Carr, R., King, S., Cowley, P., and Tarassenko, L. (1999). A system for the analysis of jet system vibration data. *Integrated Computer-Aided Engineering*, 6(1):53 – 65.

O'Keefe, J. and Nadel, L. (1978). *The Hippocampus as a Cognitive Map.* Oxford University Press, Oxford, England.

Stanley, J. C. (1976). Computer simulation of a model of habituation. *Nature*, 261:146–148.

Tarassenko, L., Hayton, P., Cerneaz, N., and Brady, M. (1995). Novelty detection for the identification of masses in mammograms. In *Proceedings of the 4th IEE International Conference on Artificial Neural Networks (ICANN'95)*, pages 442 – 447.

Taylor, O. and MacIntyre, J. (1998). Adaptive local fusion systems for novelty detection and diagnostics in condition monitoring. In *SPIE International Symposium on Aerospace/Defense Sensing.*

Thompson, R. and Spencer, W. (1966). Habituation: A model phenomenon for the study of neuronal substrates of behaviour. *Psychological Review*, 73(1):16–43.

Wang, D. and Arbib, M. A. (1992). Modelling the dishabituation hierarchy: The role of the primordial hippocampus. *Biological Cybernetics*, 76:535–544.

Worden, K., Pierce, S., Manson, G., Philp, W., Staszewski, W., and Culshaw, B. (2000). Detection of defects in composite plates using lamp waves and novelty detection. *International Journal of Systems Science*, 31(11):1397 – 1409.

Ypma, A. and Duin, R. P. (1997). Novelty detection using self-organizing maps. In *Proceedings of International Conference on Neural Information Processing and Intelligent Information Systems (ICONIP'97)*, pages 1322 – 1325.