Anticipatory Behaviour based on Prediction of Image Sequences

Achim Lewandowski[1], Patrick M. Poelz[1], and Georg Dorffner[2]

[1] Austrian Research Institute for Artificial Intelligence, Vienna
[2] Department of Medical Cybernetics and Artificial Intelligence, Center for Brain Research, Medical University of Vienna, Austria

Abstract. In our scenario an object moves in an environment with obstacles. We would like to provide a robot with anticipatory behaviour. The target is often not visible and the robot should learn to anticipate the location where the object will most likely reappear. The observations are coded as a sequence of views and therefore no physical motion model is needed for the moving object. We apply Prediction Fractal Machines (PFM) as well as Variable Length Markov Models (VLMM) to predict the continuation of the sequence of views. We present results of simulations and real world experiments.

1 Introduction and general approach

We are concerned with a solution for the following problem. A robot exists in a world with some obstacles (e.g. walls) and its task is to catch a moving object (e.g. a ball). In this paper, we start with a simpler scenario. The robot observes repeatedly a ball rolling behind a wall and reappearing at the other side. From time to time a second obstacle is placed behind the wall and the ball will be reflected at this second obstacle and will reappear at the same side of the wall where it had vanished.

We would like to provide the robot with the following behaviour: The first task of the robot is to recognize that the ball usually will reeappear at the other edge of the wall. If the second obstacle is present, the robot is expected to switch its expectation to the other side *before* the ball has reappeared again. Therefore, we would like to implement a change of expectation, if the originally expected situation has not occurred within a certain time.

In the case that the ball is reflected, we investigated two scenarios. The first one assumes that the reflection is accompanied by an acoustic signal, which shall be used as a sign that the ball will reappear at the same end of the wall, after which it has vanished. The second scenario supposes that the signal is not given or that the robot has no audio sensors.

We start with the assumption that the robot does not move and that his view cone remains the same during the whole experiment. The ball rolls behind the wall with different velocities. During a small percentage of the trials an obstacle is placed such that the ball will be reflected.

We show in the next section, how the robot's views are coded. The subsequent section is concerned with a suitable algorithm to predict the sequence of coded views. In the following two sections, we present experimental results, both for simulated and real situations. We close this paper with a summary and an outlook.

2 Coding

To apply the algorithms, we need to discretize the observed images in the following way. The view cone is horizontally partitioned in several smaller sectors. Assuming the absence of the ball, each sector is assigned either to "WALL" or "ELSE". A sector labeled "WALL" occludes the ball and never changes its appearance, a sector with "ELSE" is allowed to switch to the "BALL" state, if parts of the ball are visible in this sector. We therefore assign to each sector one of the three different values

2: parts of the ball can be observed in this sector ("BALL")
1: only the wall can be observed ("WALL")
0: neither ball nor wall ("ELSE")

Table 1. Assigned values to sectors

The sectors do not need to necessarily cover exactly the same angle. In our experiments, we used angles such that the rays (see figure 1) would cut pieces of equal lengths out of a line parallel to the wall.

Another sensor (only available in the first scenario) records the possible occurence of an acoustic signal in the case that the ball is reflected. Figure 1 illustrates, how the observed scenario is translated into a discrete valued vector. The acoustic signal is added as an additional element "SOUND" with possible values 0 or 1, if a sound could be heard.

If we record the robot's observation at discrete time steps, we get a sequence of vectors. Figure 2 shows the possible states of a run, for which the ball was reflected at the invisible obstacle from figure 1, indicated by the acoustic signal in the 13th column.

Even, if we vary the velocity of the ball, the number of possible views remains the same. Figure 3 shows the possible 11 states for our first scenario. The run from figure 2 could be expressed as $1-1-1-1-2-3-4-5-6-1-1-1-11-1-1-6-5-4-3-2-1-1-1-1$.

3 Learning with variable memory length

Suppose that we place the obstacle behind the wall with a probability $p \ll 0.5$ and therefore the ball more often passes the wall than it is being reflected. If

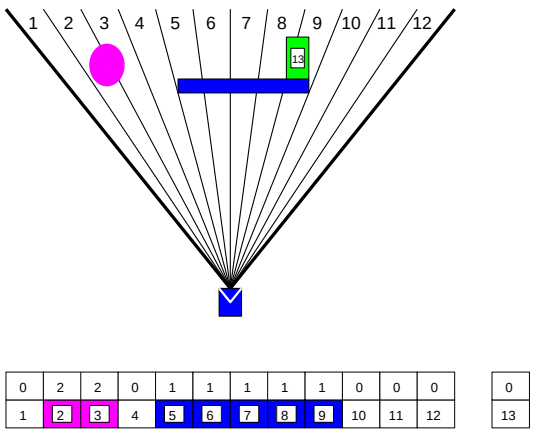

Fig. 1. The viewing cone is partitioned into e.g. 12 sectors. Each of these sectors is coded with "WALL=1", "ELSE=0", "BALL=2". Another element (13) is used to indicate whether a sound has been heard. The second obstacle appears with $p \ll 0.5$. In this example, the 12 sectors and the sound signal (missing for the given situation!) are coded as 13 dimensional vector [0 2 2 0 1 1 1 1 1 0 0 0 0].

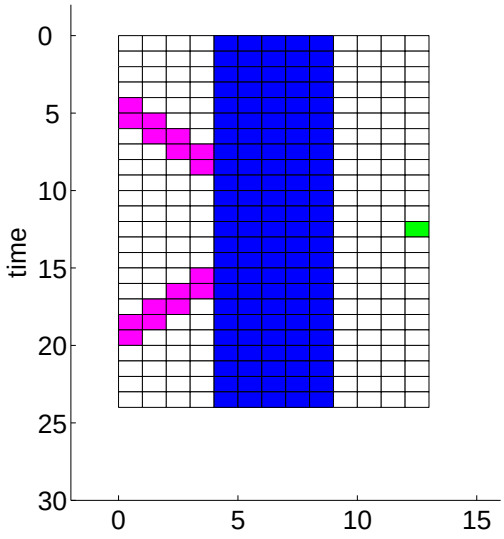

Fig. 2. A ball rolls behind the wall and is then reflected. The reflection is accompanied by an acoustic signal.

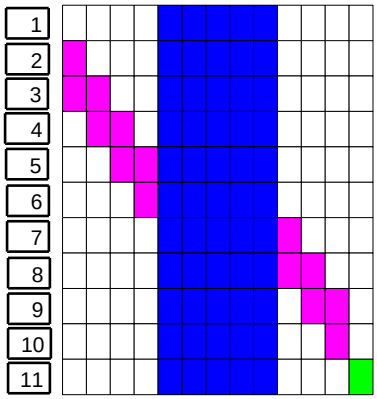

Fig. 3. The possible 11 states for the first scenario. The last column only indicates the acoustic signal.

the robot is observing a ball approaching the wall from the left, we would like to have the behaviour that in both scenarios, with or without acoustic signal, the robot should forecast that the ball will pass the wall and will reappear at the other end.

The differences between the two scenarios become clear, if we proceed in time. If an acoustic signal is given, indicating the reflection, the robot should learn to switch its forecast to "The ball will roll back to the left side". In the second, more interesting scenario the acoustic signal is not given and the robot must learn from experience, when to expect the ball at the right end and to switch the opinion, if the ball is "late". We assume that the obstacle is located nearer to the right edge, otherwise the expectation would be switched when the ball is already visible.

We would like to use forecasts that are built on the history of the sequences as introduced in the last section. Before we proceed, we explain, why different situations require different lengths of memory to make the correct decision. If we observe a ball running from the 1st sector to the 2nd sector, we could already conclude that it will eventually arrive at the 3rd sector and the needed memory has a depth of maybe 2.

In general, knowing the last two or three states is usually sufficient to roughly predict the next state. But if the ball is actually behind the wall we need a memory that must be decisively longer, at least longer than the time the ball needs to reappear from behind the wall. If we use a memory shorter than this

time, the robot will not be able to recognize, whether the ball is late or not. A related problem is to distinguish between the cases "A) The ball has not arrived yet.", "B) The ball has already left the visual cone.", "C) The ball is now behind the wall and will reach the right end." and "D) The ball is now behind the wall and will reach the left end."

It is easy to see that we need additional constraints to distinguish between the cases A and B. If we observe the scenery after the ball has passed (Case B), the needed memory to remember that we have already seen the ball increases linearly with time. The simple solution is just to use recordings one step before the ball enters from the left and stop them after the ball has left the scenery, either at the right end or at the left end.

As a simple Markov model with fixed memory length is not well suited for long memory lengths due to the exponential growth of possible branches, we used the Variable Length Markov Models (VLMM) of Ron, Singer and Tishby [DRT94,DRT96]. The main problem is to answer the question, which prediction context one should use. The prediction context is always a subsequence of the concrete sequence observed up to the current moment and the problem is to determine its *length* in the given situation. We used the criterion mentioned in [TD98] which is based on the work from [DRT96]:

Assume that the current context is given by the finite sequence w and that $\hat{P}(s|w)$ is our estimate to arrive in state s given the so far observed sequence. The question is now, whether we should extend the context and go back another step in history to use aw instead of w. The criterion says we should use aw, if at least one state s exists with

$$\hat{P}(s|aw) \geq \frac{1}{A}(1+c)c \text{ and } \frac{\hat{P}(s|aw)}{\hat{P}(s|w)} \geq (1+3c),$$

with $c \geq 0$ being a parameter to control complexity and A indicating the number of possible states. The resulting contexts can be seen as the leaves of a tree. For a given sequence, the longest context, which is identical to the last elements of the sequence, needs to be found in the tree and the prediction is calculated on the basis of all states in the training data following such a sequence.

Additionally, we applied Prediction Fractal Machines (PFM) [TD98], which try to generate contexts in a different way. Here the observed discrete sequences are mapped into a vector space, such that similar sequences will be mapped onto neighboring points. The points are then clustered, e.g. with K-means, and one-step predictions are based on the behaviour of sequences belonging to the same cluster. In this case, the number of clusters are used to fine-tune the cluster algorithm.

4 Quantitative results for simulated runs

We generated a training set with 500 sequences, the random velocities of the ball were uniformly distributed between 0.8 and 1.2 sectors per second, and the

obstacle appears with a probability of p=0.2. We present results only for the more difficult case without acoustic signal.

The mean classification error of the one-step forecast on a test set with 500 sequences is given in the table 4. We used 100 repetitions to calculate the mean error. The repetitions were necessary for VLMM, as draws can be present, and a random experiment was used to decide for the state which should be assigned to a given context. In the case of PFM, the initial clusters are randomly chosen.

Please notice that even in the case of a wrong one-step forecast the multi-step forecast for the edge at which the ball will reappear can still be correct. Multi-step forecasts are built by reusing the one-step forecasts iteratively as new sequence elements.

Algorithm	Mean Error	Remarks
VLMM	8.4%	effective depth=8, 111 leaves, c=0
PFM	13.9%	31 eff. cluster centers (nom. 50), max. depth=10
PFM	11.1%	47 eff. cluster centers (nom. 100), max. depth=10
PFM	9.7%	55 eff. cluster centers (nom. 153), max. depth=10
PFM	9.9%	59 eff. cluster centers (nom. 166), max. depth=12

Table 2. Results of first simulation

The k-means algorithm to calculate the cluster centers removed empty clusters, therefore the number of effective clusters was in average smaller than the intended nominal number. The mean error of the VLMM was slightly smaller than the result of the best PFM algorithm, but the number of leaves (=rules) was also twice as high in comparison to the PFM algorithm.

In a second experiment with a training set with 1000 runs, random velocities between 0.8 and 1.2, varying ball sizes (0.5-2.5 units) and also varying wall lengths between 5 and 7 units, the obstacle appearing with p=0.2, but placed everywhere behind the wall, we get 76 different possible states (instead of 11). The results are summarized in table 4.

Algorithm	Mean Error	Remarks
VLMM	18.8%	1317 leaves, c=0.03
VLMM	19.4%	648 leaves, c=0.10
VLMM	21.3%	351 leaves, c=0.20
PFM	42.5%	46 eff. cluster centers (nom. 50), max. depth=10
PFM	26,0%	163 eff. cluster centers (nom. 100), max. depth=10
PFM	22.8%	286 eff. cluster centers (nom. 400), max. depth=10
PFM	20.0%	477 eff. cluster centers (nom. 800), max. depth=10
PFM	19.6%	582 eff. cluster centers (nom. 1000), max. depth=10

Table 3. Results of second simulation

For comparable sizes of clusters (PFM) and leaves (VLMM), the two algorithms show similar behavior. One possible disadvantage is visible: both algorithms extract many rules from a set of 1000 runs. VLMM best error on the test set is achieved, if the large number of 1317 rules is generated. Sequences of views belonging to different wall sizes are disjunct, and therefore observing runs for a certain wall size do not help if predictions for another wall size are sought. The situation for different ball sizes is nearly the same, only the view of the pure wall will be common to all runs.

Nevertheless, the figures 4 and 5 show that the algorithm can learn to generate plausible forecasts. Here, the multi-step forecast is generated by repeatedly adding the one-step ahead forecast to the observed sequence. The figure 4 shows that usually the prediction "The ball will appear at the right edge of the wall" is given, simply because it occurs more often ($\approx$ 80% of all cases). But after the ball did not appear, when it was expected, the expectation is immediately switched and the ball is now assumed to reappear at the left edge (Figure 5).

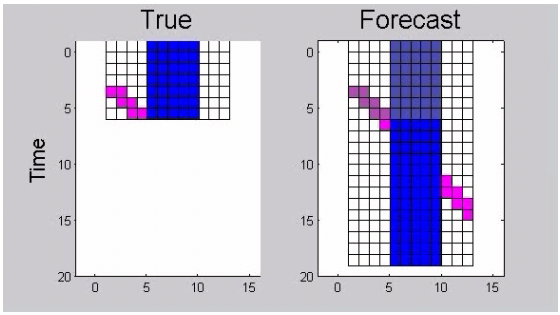

Fig. 4. After 7 time steps: The so far observed sequence (left) and the multi-step forecast (right).

The already mentioned scenario with acoustic signal is easier to solve as the acoustic signal alone is sufficient to switch expectations, and the longer history is only needed to estimate the time span after which the ball will reappear. To summarize the results, we conclude that both algorithms (VLMM and PFM) are able to learn the behavior under the assumption that a sufficient number of runs is presented. Generalization in our framework means that the algorithms are able to learn to ignore irrelevant old parts of the sequence. But for example the algorithms do not convince if they are used to generalize to unknown wall positions or unknown ball sizes. One possible solution is to replace each unknown state in an observed sequence by the nearest state, whereby the nearest state is calculated via some measure using the sector views.

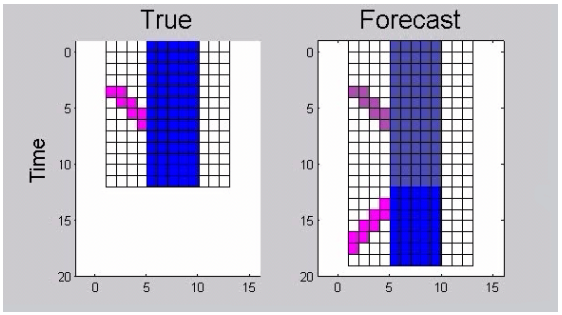

Fig. 5. After 13 time steps: The so far observed sequence (left) and the multi-step forecast (right). The algorithm has switched its expectation before the ball reappeared at the left side.

5 Real life experiments

We tested whether the approach with discrete sequences is usable if we work with real data. We performed 54 runs for the training set, for 12 of them an obstacle was placed behind the right edge of the wall, causing the ball to roll back. For the other 42 runs no obstacle was present. A test set consisted of 5 runs with obstacle and 3 runs without. As the ball was pushed by hand, a variety of different runs was generated. We used the VLMM approach for the construction of the model.

The camera image (see figure 6) was used to determine ball and wall position, whereby the colour of objects was used for the segmentation. Finally the observed ball and wall positions were used to generate a discretized version of the view (bottom right). The number of segments depends on the number of available runs and should be sufficiently high to allow an estimation of the velocity of the ball. Too many segments (in relation to the number of runs) lead to inferior results.

After 3-4 steps the fitted model predicted for each example of the test set that the ball would reappear at the right edge of the wall (see figure 7). In those 5 cases where the ball was reflected, the model switched its expectation before the ball reappeared at the left edge (see figures 8 and 9).

As we have seen now, the robot is able to predict the most probable location where the ball will reappear, and it is also capable to adjust its expectation, if the ball does not appear at the anticipated location. For developing a strategy to catch the ball, additional knowledge about velocities of robot and ball and the actual goal must be given.

If, for example, the goal is to avoid unnecessary running, then the robot might run to the *visible* ball, if it appears at the right edge, but it might already run to the left edge, if the ball is still invisible and did not "show up" in time at the right edge. This is the scenario we wanted to deal with. If the goal is to catch the ball at one of the edges as often as possible and the robot is assumed to have only one attempt, then the robot should always proceed to the right edge. Here

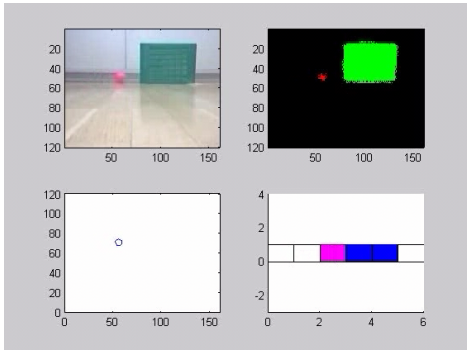

Fig. 6. Top left: Camera image. Top right: Image segmentation. Bottom left: Ball position. Bottom right: Discrete sector view.

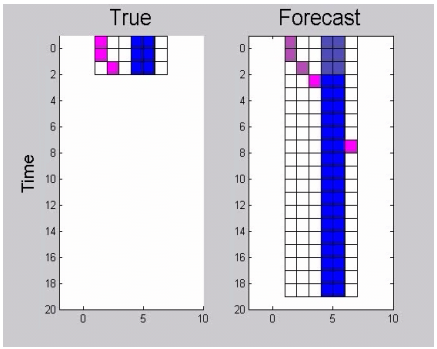

Fig. 7. After 3 time steps: The so far observed sequence (left) and the multi-step forecast (right).

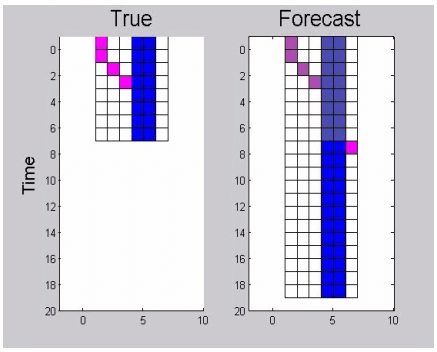

Fig. 8. After 8 time steps: The so far observed sequence (left) and the multi-step forecast (right).

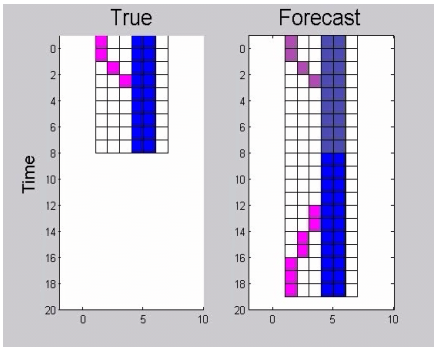

Fig. 9. After 9 time steps: The so far observed sequence (left) and the multi-step forecast (right). The expectation has switched to left, as the ball did not appear at the right side.

we assume that the robot is not fast enough to catch the ball at the left edge on its way to the right, but is sufficiently fast to catch the ball at the right edge.

6 Conclusion and Future Work

We have presented a successful approach to predict the location of reappearance of a temporarily hidden ball. The exact location depends on the random presence of a second obstacle. If this obstacle is present, the object will not appear at the anticipated location at the anticipated time and the algorithm will adjust the expected location before the object will be visible again.

The approach uses sequences of simplified views and does not rely on any physical model. Although the algorithm was able to learn the desired behaviour, some drawbacks must be mentioned. Each scenario for a fixed wall and a given ball size can be learned. However, a change of ball size or wall size or position lead often to unknown views.

So far, the robot is not allowed to move. We are currently working with scenarios, where the robot is indeed allowed to move while observing the target. Therefore, we will get sequences of views and have to incorporate the actions the robot has chosen. We are currently working on extensions to predict the future views under the assumption that the future actions of the robot are known.

Acknowledgements

This research has been done within the project "MindRACES", funded by the European Community in the "Competitive and Sustainable Growth Programme", under contract number IST-511931.

The Austrian Research Institute for Artificial Intelligence acknowledges basic financial support by the Austrian Federal Ministry for Education, Science, and Culture and of Transport, Innovation and Technology.

References

[DRT94] Y. Singer D. Ron and N. Tishby. Learning probabilistic automata with variable memory length. In *Computational Learing Theory*, pages 35–46, 1994.

[DRT96] Y. Singer D. Ron and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

[TD98] P. Tino and G. Dorffner. Constructing finite-context sources from fractal representations of symbolic sequences, 1998.