Extracting Movement Patterns from Video Data to Drive Multi-Agent Based Simulations

Muhammad Tufail[1], Frans Coenen[1], and Tintin Mu[2]

[1]Department of Computer Science, The University of Liverpool, Liverpool, L69 3BX, UK; [2]Department of Electrical and Electronics Engineering, The University of Liverpool, Liverpool, L69 3BX, UK

Abstract. Computer simulations are used to create and evaluate real world scenarios in a manner that is controlled, nonintrusive, cost effective and safe. One technology for realising computer simulation is Multi Agent Based Simulation (MABS), the advantage being that the entities that feature in a simulation can be expressed as agents that have all the features associated with agents (autonomy, goal driven, etc.). A particular challenge of MABS is the acquisition of the data required to define agent behaviour. One approach is to "hand craft" agent behaviour, however this is error prone and time consuming. In this paper we proposed a method whereby we can extract, what we have termed "Movement Patterns" (MPs) which in turn can be used to drive agents in a MABS environment.

Keywords: Multi Agent Based Simulation (MABS); Movement Pattern Mining; Movement Patterns;

1 Introduction

Multi-Agent Based Simulation (MABS) harnesses Multi Agent System technology to realise computer simulations of real world scenarios. MABS offers the fundamental advantage that the entities that make up a simulation can be modelled as agents which display all the characteristics of agents. Using MABS each "player" in the simulation is represented as an agent. These agents posses both desires and capabilities, which in turn allows autonomous operation and decision making. MABS is particularly appropriate for modelling scenarios that involve people or animals [1, 5, 6]; each person (animal) can be modelled as an agent.

The MABS application domain of interest with respect to this paper is behaviour analysis, more specifically animal movement behaviour analysis. Examples where MABS have been used for human behaviour analysis can be found in [3, 7, 8]. The challenge of behaviour MABS is the modelling of individual agent behaviours. One approach [1] is to "hand craft" the models on which the agent behaviour will be based by observing the real world entities to be considered. However, hand crafting is error prone and resource intensive. An alternative, and that advocated in this paper, is to use data mining techniques to learn the desired agent behaviours in an automated manner. More specifically to apply such techniques to video data, featuring the entities (agents) of interest in the intended environment, and learning the desired behaviour in the form of patterns as first proposed in [12]. This paper extends the work presented in [12] by proposing a mechanism for mining video data to extract "Movement Patterns" (MPs), which in turn can be used in the context of animal movement behaviour MABS. MPs in this context are spatially referenced patterns that specify potential follow on locations for a given agent (animal), however MPs can be applied in the context of other movement behaviour studies such as fire exit simulation and movement behaviour at rail terminal/station forecourts or airport concourses. MPs can be defined in absolute terms according to some reference origin; or in relative terms according to the nature of an agent's current location. MPs are selected in a random probabilistic manner so that on each run the behaviour of agents will not be identical. The main contributions of the work presented in this paper are thus: (i) a mechanism for capturing MPs, and (ii) a mechanism whereby MPs can be effectively utilised in a MABS setting.

The rest of this paper is organised as follow. An overview of some previous work is presented in Section 2. In Section 3 the process of data acquisition and the nature of the video data used for illustrative purposes with respect to this paper is discussed. The environment representation, an important precursor to any discussion on MPs, is presented in Section 4, followed by the pattern mining framework itself in Section 5. Section 6 discusses a proposed MABS framework which utilises the concept of MPs. Section 7 then presents an evaluation of the operation of the proposed MABS in terms of MPs. A summary and some conclusions are presented at the end of this paper, together with some suggestions for future work, in Section 8.

2 Previous Work

Computer simulation offers a number of advantages over real life experimentation. The main challenge of MABS, as in the case of computer simulation in general, is the acquisition of the knowledge required to build the simulations. In [**?**], this was done by hand, an approach found to be very resource intensive; rodent behaviour was modelled using the concept of a behavioural graph. Hand crafting is thus both time consuming and error prone. An alternative approach was considered in [12] where the idea was to learn the behaviours of agents from video data and then to utilise this in a MABS setting. In [12] only single mouse scenarios were considered which served to significantly simplify the nature of the models to be considered. In this paper a more sophisticated mechanism in the context of multiple agents is presented.

The proposed use of MPs, as will be seen later in this paper, includes the concept of *states* to represent the relationship between entities in a MABS. The idea of states has for some time been used in the context of simulation, and by extension, MABS; although it should be noted that in previous work the concept has not been used in the same way as presented in this paper. In previous work a state typically describes a set of attribute-values that an agent possesses at a given time t_1; a follow on state is then a potential future state that an agent may adopt at time t_2. With respect to behaviour simulation, in [5, 10, 11] the concept of states has been used to represent the behaviour of entities. For example in [5] states were used for the representation of animal behaviour, more specifically foraging of sheep. In [10] states were used to represent the way that a group of ants selected a "best" nest site, although the group was considered in terms of a single entity. In our case the state concept is used to capture the relative position of two mice (as discussed further in Sub-section 4.2 below).

The fundamental idea presented in this paper is that the necessary knowledge can be derived from video data through a video data analysis process. More specifically the idea is to extract "movement patterns" from video data and utilise these patterns to drive a MABS.

3 Data Acquisition And Video Data Collection

When building a MABS the most challenging task (as noted earlier) is the acquisition of the knowledge required to populate the simulation. This section discusses the acquisition of data; more specifically extracting knowledge from video data. From the literature different methods have been proposed concerning data acquisition. The usually process is one of observation or interviews with domain experts. For example in [10] a set of behavioural rules were derived through a process of observation, and in [**?**] surveys of individuals were conducted. In [**?**] the following observations were made concerning the manual processes of MABS data acquisition:

1. It is difficult to translate information gleaned from domain experts (or from observations) into usable form since such translation typically requires extensive domain knowledge.
2. Any hypotheses concerning agent behaviour, based on information obtained from domain experts (or derived from laboratory experiments) needs to be precise (information gleaned from domain experts is typically "fuzzy").
3. The manual collection of data from real world scenarios, regardless of how this is done, is a time consuming and resource intensive process.

The most significant of the above is that manual data acquisition is a resource intensive process and thus some form of automation is desirable. In [**?**] the authors briefly discuss the potential for automatically extracting the required data from existing records (documents); in this paper it is suggested that the automated extraction of knowledge from video data is the solution.

Recall that the focus for the work presented in this paper is scenarios involving two or more mice in a "box environment" (as in [12]). To this end video data was obtained in a laboratory setting by suspending a video camera over a box, whose ground area measured 1.2m^2, into which two mice and some objects (nests, obstructions and so on) were introduced. Two stills from the video data collected are given in Figures 1 and 2; in the figures the two mice can be clearly identified. Note that the stills feature slightly different scenarios; the obstructions are not in the same place in both cases. The circular objects are nests

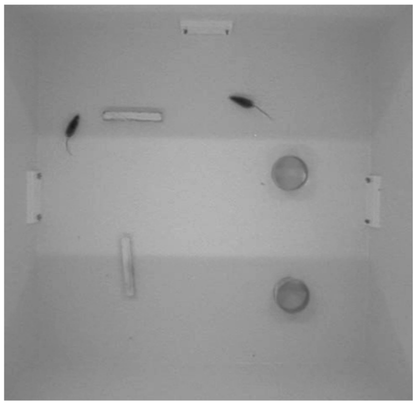
Fig. 1. Still from rodent video data (example 2)

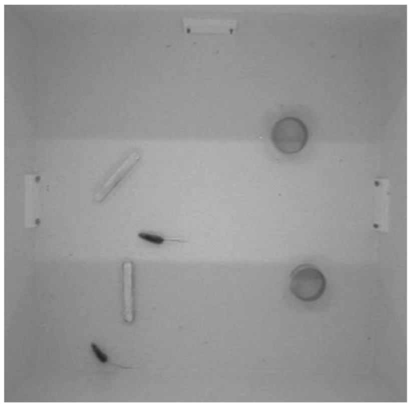
Fig. 2. Still from rodent video data (example 2)

wgich mice can enter and exit. The boxes also feature side panels but these are not of significance with respect to the desired video analysis.

For the work presented in this paper a bespoke software system was developed, for tracking "mice" in video data, founded on the "blob tracking" technique described in [2]. The software operates by processing the video data in a "frame by frame" manner. The software automatically detects the location of each mouse by detecting movement and then assigning a tracking ID to the identified blob. Blob locations are recorded at a fixed *sample interval time* s measured in term of a number of video frames. With respect to the work presented in this paper $s = 5$ frames was used (5 frames equates to 200 milliseconds). On occasion the blob tracking fails due to noise in the data or because the blob "disappears" into a nest. Where this happens the initialisation process is repeated and, once the blob has been re-identified, the tracking ID is reassigned.

The situation where both mice were lost at the same time did not occur, but erroneous reassignment of IDs would not adversely effect the data collection and the consequent mining of MPs. At the end of the mouse tracking process two sets of locations were obtained, one for each ID (more if scenarios featuring a greater number of mice are considered). The learning process continues until we have at least one MP for every potential location. In this paper we discuss two mechanism for representing locations, absolute and relative, both are considered in further detail in Section 5.

4 Environment Modelling

An important aspect of MABS is the nature of the environment (playing area) to be modelled [9]. The significance in the context of the proposed approach is that the mechanism whereby the environment is modelled influences the nature of the output from the machine learning. The proposed representation is essentially that of a "tile world" (such as that used in [4, **?**]). Using this mechanism the environment is modelled by dividing it into a collection of grid cells (squares). Each cell is given a sequential number (*address*); the set of grid numbers is then given by $L = \{l_1, l_2, l_3 \ldots\}$. Grid mechanisms of this form offer the general advantage that prescribed translations from one cell to another can be achieved by applying a constant k to the current address (a concept also used in [12]). For example, in Figure 3, to move one cell to the north $k = -14$; and to move one cell to the south east $k = 15$. Note that the value of k captures both distance and direction, hence we refer to such values as *movement vectors*; $k = 0$ indicates no movement. For the video data the 1.2m^2 environment was divided into 14×14 grid squares (196 in total); thus each square measured approximately 8.5×8.5cm, about the size of a mouse.

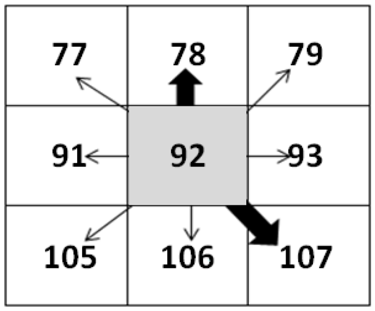

Fig. 3. Movement vectors

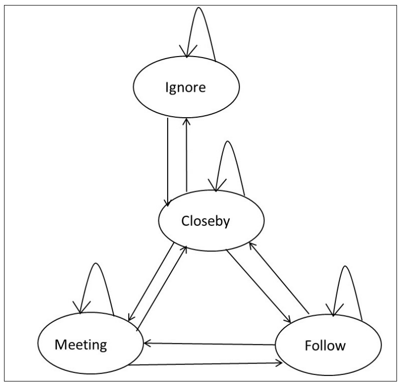

Fig. 4. State Graph

As will become clear later in this paper two kinds of MP are considered, relative and absolute. For relative MPs each grid cell describing an environment has a ground type associated with it. Five ground type labels were used $\{w, n, b, o, -\}$; where: w indicates a location next to a wall, n indicates a nest site, b indicates an obstruction of some kind (a "block"), o indicates "open ground" (and is also the default label) and $-$ indicates an area outside of a playing area. Similar ground type labels were considered in [12].

4.1 Location Descriptors

A location descriptor is a *composite ground type* label comprised of the ground types for the 3×3 sub grid surrounding current location of interest linearised from top-left to bottom right. Recall from Section 4 that we have a ground type $-$ indicating locations outside of a playing area, this is used in the case of corner and edge locations.

4.2 States

An important aspect of the proposed framework is the concept of states. As noted above, in the context of this paper, a state defines the relationship between two entities (mice in our case) in such a way that this can be incorporated into MPs. We have four different states {"ignore", "closeby","meeting", "follow"} arranged in a graph as shown in the Figure 4. The states are defined on the basis of: the distance between the two entities in term of "zones" surrounding each entity, the direction of one to the other and their mutual directions of travel.

5 Movement Pattern Mining Framework

The objective of the proposed MP mining framework is to extract (mine) MPs from the location data obtained as a result of the video analysis described in Section 3. For each entity (mouse) and each sample time stamp t_i we extract a MP describing the entity's movement from t_i to t_{i+1}.

As discussed previously, two mechanisms were considered for representing MPs: (i) absolute and (ii) relative. The distinction between the two, as the terminology suggests, is that in the first case locations are recorded relative to the origin of the environment while in the second locations are recorded relative to the local surroundings. Absolute locations are therefore expressed in terms of a specific address (a unique number), while when using relative patterns the locations are represented using descriptors. The significance is that absolute patterns can only be used with respect to simulations that feature the same environment as that from which the patterns were mined, while relative patterns are more versatile and can be used for a variety of simulations. However, relative descriptors are more complex. The advantages of using relative patterns over absolute patterns are as follows: (i) fewer location pattern are required than the number of cells in the grid (given a reasonably sized paying area), (ii) consequently storage advantages are accrued, (iii) they are more generic than the absolute mechanism (MBPs represented using the absolute mechanism can only be used with respect to a playing area identical to that from which they were extracted) and (iv) they are rotation invariant.

In this section both mechanisms are considered in further detail in the context of MPs. It should also be noted here that the extracted MPs provided a "knowledge base" with which to drive the desired MABS. The operation of this MABS is presented in the following section, Section 6.

The fundamental structure of an MP is that of a tuple of the form:

$$MP = \langle F, S, v, Path \rangle$$

Where: (i) F is the "From" location (where the movement represented by the MP starts); (ii) S is a collection of one or more states describing the spatial relationship between agents featured in a scenario; (iii) v is a *movement vector* (as described in Section 4); and (iv) $Path$ is the *path*, encapsulated by the MP, which an agent needs to follow to get to the "To" location. The nature of these elements is discussed in further detail in the remainder of this section.

The From location (F) is the start location in the grid environment from where the movement described by a MP commences, defined in terms of a location identifier loc_ID. The format of loc_ID depends on whether we are considering absolute MPs or relative MPs. In the first case it will simply be a grid cell number, in the second case it will be a location descriptor of the form described above in Sub-Section 4.1.

Each state s in S defines the relative relationship between two agents using a set of labels, {"ignore", "closeby","meeting", "follow"} defined using a set of concentric zones as explained in Sub-Section 4.2. An MP can feature one or more states depending on how many agents feature in a scenario. If we have n agents then $S = \{s_1, s_2, \ldots, s_{n-1}\}$, we are not interested on how an agent relates to itself hence $n-1$. If there is only one agent, then $S = \emptyset$. Note that a MP defines movement in terms of a single entity.

The element v of the MP tuple is a movement vector of the form described in Section 4. The value for v can be expressed as a single number or as a coordinate pair $\langle x, y\rangle$ depending on whether we are using absolute or relative MPs. The value of v when applied to an agent's current location indicates the "To" location associated with the MP.

The fourth component, $Path$, as already noted, indicates the "route" that the MP prescribes whereby an agent adopting the MP can get from its From location to the indicated To location. The number of elements in $Path$ ($|path|$) depends on how far we wish to "look ahead". With respect to the evaluation and case studies presented later in this paper $|path| = 5$ was used. Using $|path| > 1$ means that our rodent agents have a "memory", they have a planned route they wish to follow. The elements of $Path$ are all movement vectors. Thus using absolute movement patterns, where $|path| > 1$, we have a sequence of movement vectors of the form $\{v_1, v_2, \ldots, v_{|path|}\}$ were $v_{|path|}$ indicates the end location (the To location) and the remaining vectors indicate locations at intermediate locations referred to as *waypoints*. In the case of relative movement patterns, where $|path| > 1$, we have a sequence of movement vectors of the form $\{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots, \langle x_{|path|}, y_{|path|}\rangle\}$.

6 Operation of MPs in the Context of MABS

This section describes how MPs are integrated into our proposed MABS framework. Our MABS, like other MABS, operates in an iterative manner. On each iteration each agent featured in the current scenario of interest updates its location, direction and current path. This is done according to whether we have waypoints pending or not. In the first case $|Path| > 1$ and the next location $w_i \in Path$ can be selected (and removed from $Path$). This process continues until $|Path| = 0$ is reached. When $|Path| \equiv 0$ a new MP must be selected.

In the evaluation considered later in this paper each MABS agent uses the same knowledge base, but this does not have to be the case. Agents search the knowledge base with their current location (described in absolute or relative terms) and State and identify all relevant MPs and consequent To locations. Typically there will be a number of these. Given that we do not wish our simulation to operate in the same manner on each run a specific MP is selected in a probabilistically weighted random manner. In other words the most likely MP is the most likely to be selected but not necessarily so. In the case of relative patterns it is also necessary to carry out additional checks to make sure that associated To locations are *legal* locations. A legal location is one within the environment (thus not a location with ground type $-$) that is not obstructed in some way (thus not a location with ground type b). Note that on start up our mouse agents are placed at some legal location and their states set according to their relative positions in the simulation environment. Note also that in the case where no legal follow on location can be found a default next location will be selected (in practice we found that this situation did not occur).

Simulation time was calculated according to Equation 1. Simulations should operate so that the agents move in a smooth manner from one location to another location. In Equation 1, q is some constant; with respect to the evaluation presented later in this paper $q = 5$ was used as this was found to produce a smooth simulation. Recall that the sample time was 200 millisecond, thus with respect to this paper simulation time was $200/5 = 40$ milliseconds.

$$simulation\ time = \frac{sample\ time}{q} \tag{1}$$

7 Evaluation

There is no "gold standard" with which to evaluate the operation of simulations (MABS or otherwise). However, the simulation should be as realistic as possible. The mechanism adopted with respect to the work presented in this paper was that the operation of the simulations should be compared with the original video data. Consequently we could compare the distribution of MPs in the simulated data with those produced using the real video data. In the case of the relative mechanism the comparison was conducted by comparing the number of occasions that each MP was recorded with respect to the simulation and video data. However, in the case of the absolute mechanism there were too many locations to provide a meaningful comparison. Recall that the grid size used for the video data was 14×14, thus we had 196 unique locations. Thus, for the absolute analysis we divided the environment into 7×7 blocks, each block representing $2 \times 2 = 4$ "standard" grid cells. As a result 49 blocks were used for the comparison. For each block of cells, the occurrences count of the number of times that an agent visited the block was determined both with respect to the video and simulation data. For the evaluation two scenarios similar to those shown in Figures 1 and 2, featuring two agents, were used. Both the relative and absolute representations were considered, with and without the concept of states.

The results for the evaluation using absolute MPs are presented in Figures 5 to 8. Figures 5 and 6 give the results using coarse block grids for videos 1 and 2 using absolute movement patterns without states, while Figures 7 and 8 present the results for the same videos using absolute movement patterns with states. In each case, the blue bar indicates the occurrence counts extracted from the video data while the red bar the occurrence counts for the simulation. From the figures it can clearly be observed that the behaviour of the mice agents in the simulation is similar to that featured within the video data. We can place a value on the "similarity" (sim) of a simulated scenario compared with a real life scenario, given a set of simulated occurrence counts $F_s = \{s_1, s_2, \ldots\}$ and a set of real occurrence counts $F_r = \{r_1, r_2, \ldots\}$, using:

$$sim = \frac{\sum_{i=1}^{i=|F_s|} s_i \sim r_i}{|F_s|} \tag{2}$$

The similarity value obtained using absolute addressing and no states for Video 1 was 0.08 and for Video 2 was 0.11, and when using states the accuracies obtained were 0.12 and 0.13 respectively (note that ab accuracy of 0.0).

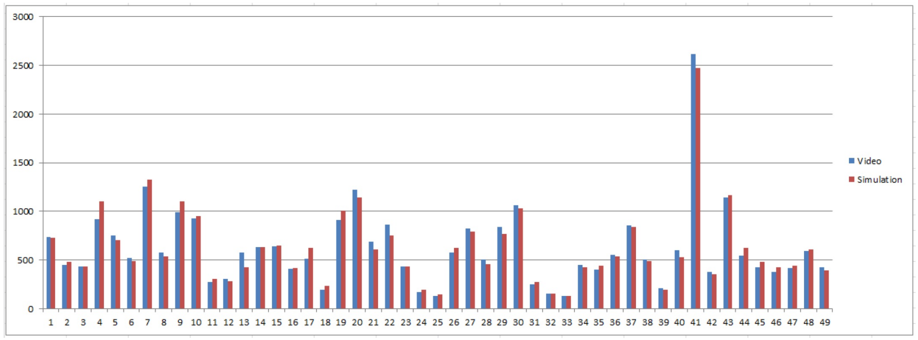

Fig. 5. Comparison of simulation data with video data in terms of blocks visited using absolute MPs without states and video 1

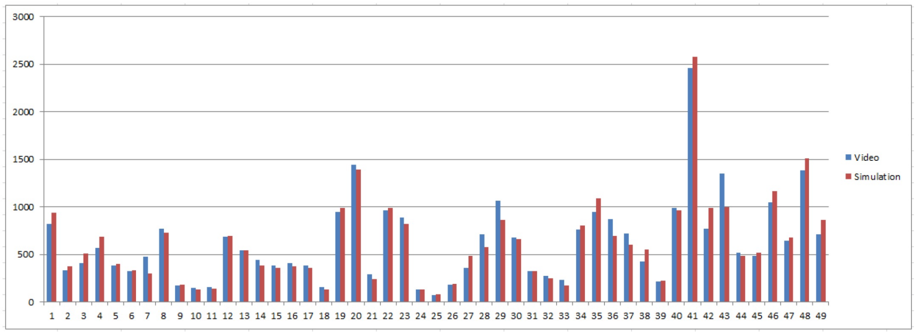

Fig. 6. Comparison of simulation data with video data in terms of blocks visited using absolute MPs without states video 2

For the evaluation using relative movement patterns, with and without states, the environment featured 109 relative addresses (location descriptors). If we include states we have 109×4 state-address combinations. Too many to present in bar graph form (and in this case it did not make sense to consider a coarse grid as in the case of the absolute addressing evaluation presented above). A fragment of the results obtained are presented in Tables 1 and 2. The tables list location descriptors on the left, followed by the difference in occurrence counts between the simulation and the real life experiments as recorded in videos 1 and 2. The last column gives the average difference. Totals are given in the last row. From the tables the overall computed similarity wiyhout using states, for Video 1 and 2, was 0.47 and 0.57 respectively; whilst when using states it was 0.58 and 0.62. For completeness fragments of the frequency count results obtained using absolute MPs are presented in Tables 3 to 2.

The results obtained are summarised in Table 5. From the summary table, and the results given above, it can firstly be seen that in general the simulations, using either absolute or relative MPs, were realistic. Secondly it can be be seen that the similarity value with respect to the absolute MPs is less than that when using relative MPs; suggesting that absolute MPs produces a better simulation. Thirdly it can be seen that not using the state concept produces a better simulation than when using states, thus calling into question usage of the concept

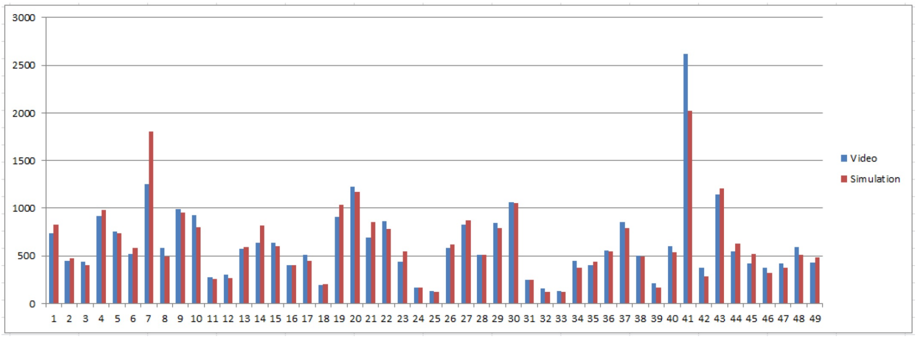

Fig. 7. Comparison of simulation data with video data in terms of blocks visited using absolute MPs with states and video 1

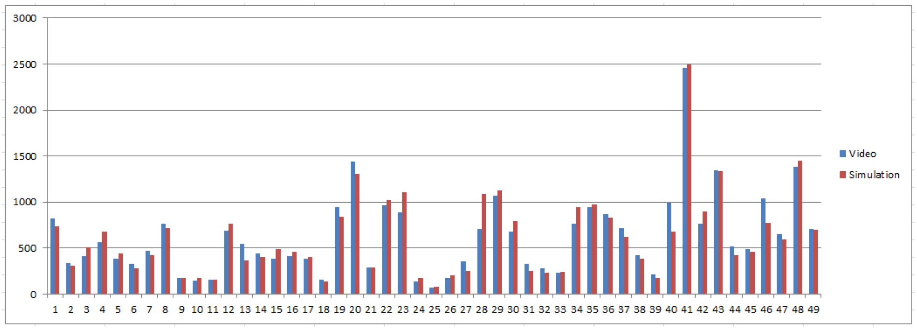

Fig. 8. Comparison of simulation data with video data in terms of blocks visited using absolute MPs with states and video 2

of states. Overall it was thus concluded that the most appropriate mechanism for our behaviour MABS was the absolute mechanism without states although the relative mechanism provides more versatility. Whatever the case, from the foregoing, it can be concluded that the operation of the simulations, using either absolute or relative MPs, was effective.

8 Conclusion

In this paper a process has been introduced for mining Movement Patterns (MPs) from video data in such a way that these can be used in the context of MABS. The idea is to learn the desired MPs from video data, a more efficient and less error prone mechanism than handcrafting. Two mechanism for representing patterns are considered, absolute and relative. Evaluation was conducted by "closing the loop" and comparing the operation of the absolute and relative mechanisms, with and without states, with the original video data. Thus four different representations were considered: (i) absolute with state, (ii) absolute without state, (iii) relative with state and (iv) relative without state. The evaluation indicated that effective simulations were attained, with the absolute representation without states producing the most realistic MABS simulations although with the caveat that relative MPs are more versatile (absolute MPS can only be sed with respect to environments identical to those from which they were generated). For

Table 1. Fragment of differences in recorded frequency counts using relative MPs without states (videos 1 and 2)

Location descriptor	**Video 1 diff.**	**Video 2 diff.**	**Avg. diff.**
wwwoowoow	0.50	1.0	0.75
- - - ssssss	7.00	6.5	6.75
-ss-ss-ss	2.00	3.5	2.75
...	...	...	...
Total Aves.	0.27	0.21	0.24

Table 2. Fragment of differences in recorded frequency counts using relative MPs with states (videos 1 and 2)

Location descriptor	**State**	**Video 1 diff.**	**Video 2 diff.**	**Avg. diff.**
- - - ssssss	CloseBy	5	3	4.0
boobooooo	CloseBy	2	2	2.0
woossosso	Ignore	2	3	2.5
...	...	...	...	...
Total Aves.		0.36	0.30	0.33

Table 3. Fragment of differences in recorded frequency counts using absolute MPs without states (videos 1 and 2)

Block No	**Video 1 diff.**	**Video 2 diff.**	**Avg. diff.**
0	7.50	117.50	110.00
1	32.00	56.00	24.00
2	131.50	106.00	25.50
3	10.50	99.00	88.50
...	...	...	...
Total Aves.	0.06	0.10	0.08

Table 4. Fragment of differences in recorded frequency counts using absolute MPs with states (videos 1 and 2)

Block No	**Video 1 diff.**	**Video 2 diff.**	**Avg. diff.**
0	92.50	89.00	3.50
1	31.50	3.50	28.00
2	28.00	144.50	116.50
3	6.00	125.00	3.00
...	...	...	...
Total Aves.	0.12	0.11	0.11

future work, the intention is to consider the propsed mechanism in the context of "many mice" scenarios (more than two) and with respect to other domains.

References

1. E. Agiriga, F. Coenen, J. Hurst, and D. Kowalski. A multiagent based framework for the simulation of mammalian behaviour. In *Research and Development in Intelligent Systems XXX*, pages 435–441. Springer, 2013.
2. Anonymous. Opencv-2.3 blob-tracking module, http://www.enl.usc.edu/enl/trunk/aqua/opencv-2.3.../blob_tracking_modules.doc, Accessed : 2013-12-01.
3. E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, 2002.
4. K. Choy, A. A. Hopgood, L. Nolle, and B. O'Neill. Implementation of a tileworld testbed on a distributed blackboard system. In *The 18th European Simulation Multiconference*, 2004.
5. B. Dumont and D. R. Hill. Multi-agent simulation of group foraging in sheep: effects of spatial memory, conspecific attraction and plot size. *Ecological Modelling*, 141(1):201–215, 2001.
6. F. Klügl and G. Rindsfüser. Large-scale agent-based pedestrian simulation. In *Multiagent System Technologies*, pages 145–156. Springer, 2007.
7. N. Malleson, L. See, A. Evans, and A. Heppenstall. Implementing comprehensive offender behaviour in a realistic agent-based model of burglary. *Simulation*, page 0037549710384124, 2010.
8. X. Pan, C. S. Han, K. Dauber, and K. H. Law. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *Ai & Society*, 22(2):113–132, 2007.

Table 5. Summary of Results

Technique	**Ave. Diff. Video 1**	**Ave. Diff. Video 2**	**Net Ave. Diff. Video. 1 and 2**	**Sim. Video 1**	**Sim. video 2**	**Avr. Sim.**
Absolute	0.06	0.10	0.08	0.08	0.11	0.09
Absolute + state	0.11	0.12	0.11	0.12	0.13	0.12
Relative	0.27	0.21	0.24	0.47	0.57	0.52
Relative + state	0.36	0.30	0.33	0.58	0.62	0.60

9. J. Pavón and J. Gómez-Sanz. Agent oriented software engineering with ingenias. In *Multi-Agent Systems and Applications III*, pages 394–403. Springer, 2003.
10. S. C. Pratt, D. J. Sumpter, E. B. Mallon, and N. R. Franks. An agent-based model of collective nest choice by the ant temnothorax albipennis. *Animal Behaviour*, 70(5):1023–1036, 2005.
11. C. J. Topping, T. S. Hansen, T. S. Jensen, J. U. Jepsen, F. Nikolajsen, and P. Odderskær. Almass, an agent-based model for animals in temperate european landscapes. *Ecological Modelling*, 167(1):65–82, 2003.
12. M. Tufail, F. Coenen, and T. Mu. Mining movement patterns from video data to inform multi-agent based simulation. In *Agents and Data Mining Interaction*, volume 9145 of *Lecture Notes in Computer Science*, pages 38–51. Springer International Publishing, 2015.