Simulation-based Visual Analysis of Individual and Group Dynamic Behavior

Pawel Gasiorowski [1], **Vassil Vassilev**[2] **and Karim Ouazzane**[2]
[1]The Vinyl Factory, London, UK
[2]School of Computing, London Metropolitan University, London, UK

Abstract *The article presents a new framework for individual and group dynamic behavior analysis with wide applicability to video surveillance and security, accidents and safety management, customer insight and computer games. It combines graphical multi-agent simulation and motion pattern recognition for performing visual data analysis using an object-centric approach. The article describes the simulation model used for modeling the individual and group dynamics which is based on the analytical description of dynamic trajectories in closed micro-worlds and the individual and group behavior patterns exhibited by the agents in the visual scene. The simulator is implemented using 3D graphics tools and supports real-time event log analysis for pattern recognition and classification of the individual and group agent's behavior.*

Keywords: Visual analytics, Behavior pattern recognition, Individual and group dynamics, Agent-based simulation, Ghosting, Ray casting

1 Introduction

Behaviour analysis of individual and group dynamics in closed micro-worlds is an area of extensive research in both academia and industry due to its wide applicability to various areas - video surveillance and security, accidents and safety management, business customer insight and video games programming. Despite the recent advances in the use of various methods for visual behavior data analysis (i.e., Markov models, statistical pattern recognition, qualitative physics, etc. - see [2-5] for some recent research in analysis of individual dynamics and [6-8] in group dynamics) and the availability of some powerful tools for video data analysis in the market such as 3VR Video Intelligence Platform, savVI Real-Time Event Detection, PureTechSystems Video Analytics, IndigoVision Advanced Analytics, IBM Intelligent Video Analytics, etc. (see [23-27] for more details on these products) the problem remains difficult. Two main factors impact the real-time performance here: the enormous volume of data, which has to be processed in real-time, and the need to combine video data processing with complex analytical symbolic data processing. While the first problem can be addressed entirely by the technological development, the second one requires model-driven behavior pattern analysis which cannot be implemented by the visual data processing methods alone.

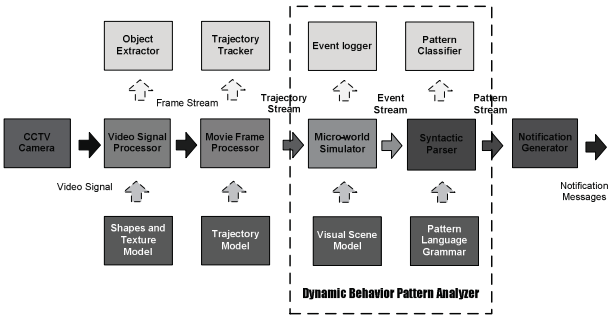

Fig. 1 General workflow of the framework

We are specifically interested in analyzing the dynamic behavior of individuals and groups of individuals moving at a walking speed within enclosed spaces (rooms, corridors, staircases, floors and open spaces) of big buildings, such as shopping malls and transport stations, as well as in large transport vehicles, such as cruiser ships. Our approach for tackling the complexity of this task is to eliminate the need for analyzing the entire video stream and replace it with the analysis of simulated data which approximates the actual video stream. The framework presented here combines two complementary methods for data analytics: visual trajectory analysis based on 3D simulation, and dynamic pattern classification based on agent's behavior logic. It forms a central part of the research program within the Cyber Security Research Group of London Metropolitan University which is dedicated to machine processing of video surveillance data in real time. It includes visual scene extraction, trajectory reconstruction, dynamic simulation and behaviour analysis for online processing of live video signals from CCTV cameras (see Fig. 1). This article reports the core of the framework, the simulation and behaviour analysis platform which has been implemented in Java using jMonkey engine [14].

2 Visual simulation as a basis for behaviour analysis

In their comprehensive book Xiang and Gong [1] classify the approaches for the development of behavior's representation model into four groups: *object-based*, *part-based*, *pixel-based*

and *event based*. In this part of the research we combine the object-based approach with the event-driven approach, which allows us to streamline the video data processing from the physical camera input to the logical notification output in real-time. Our approach to dynamic behavior analysis fits within the tradition of agent-based simulations [9-12] which is widely used in game programming. The starting point of the simulation is the reconstructed trajectories of individual objects in the visual scene [16]. As a result of the simulation the annotated live video signal is enhanced with additional information which is used for dynamic behavior analysis in accordance with the behavior pattern description. The output is an asynchronous notification corresponding to the identified pattern – i.e., calling the fire brigade, calling the ambulance, calling the police or the bomb squad.

2.1 The trajectory data

There are three separate types of input data used by the simulator – static visual scene information, dynamic trajectories of the moving objects and asynchronous event notifications. The simulator performs initial setup of the visual scene which can be updated later in the case of synchronous or asynchronous changes in the scene (e.g. appearance of a new agent as a result of moving inside the scope of the camera, appearance of a new object as a result of changing the viewing angle, disappearance of an objects from the visual scene, changing the viewing angle, receiving an additional signal, etc.). A sample XML of the visual scene is shown below:

```
<scene id="vc#00000BF>
  <camera id="cctv#0000XX">
    <frustrum>
      <viewAngle>
        <quaternionW>0.1739063</quaternionW>
        <quaternionX>-0.7228111</quaternionX>
        <quaternionY>0.19664077</quaternionY>
        <quaternionZ>0.63924426</quaternionZ>
      </viewAngle>
      <viewDirection>
        <vectorX>-0.7644838</vectorX>
        <vectorY>-0.6443617</vectorY>
        <vectorZ>-0.019037962</vectorZ>
      </viewDirection>
      <aspectRatio>
        <width>1280</width>
        <height>1024</height>
      </aspectRatio>
    </frustrum>
  </camera>
  <objects total="8">
    <object id="obj#00001 type="dynamic"
                          shape="humanoid">
      <objectsFeatures>
        <proximityTrigger>3.00f</proximityTrigger>
        <sightRangeTrigger>10.0f</sightRangeTrigger>
        <velocity>2.25f</velocity>
        ...
      </objectsFeatures>
    </object>
    ...
  </objects>
</scene>
```

Between the changes in the visual scene the simulation is driven entirely by data from the reconstructed trajectories of individual objects [16]. The trajectories are described analytically in a standard vector notation for representing location and motion. This description is based on the quaternions theory instead of purely trigonometric equations, in order to represent more complex movements involving rotation, changing the direction and twisting while moving [13]. While the trajectories provide information about the location and movements of individual agents only, the simulation generates additional information which allow the analysis of the dynamic group behavior as well. In our experiments the trajectories are simulated but the actual information will be provided by the module responsible for reconstructing the trajectories, currently under development.

2.2 The simulation entities

In the simulation we adopt an agent approach similar to the toy world which is widely used in AI for controlling intelligent robots and is also endorsed by the 3D games programming community [13]. The main entities used to build the simulation are:

Agent: an abstraction of humans or any other entities capable of some sort of movements (i.e., shopper in the shopping mall, passenger in a vehicle, traveler at the station, etc.). Their behavior is essentially either dynamic or active but always autonomous.

Pair: two agents involved in dynamic interaction (i.e., handshaking, hugging, pushing, punching or kicking each other)

Group: several agents sharing some common behaviors, allowing to treat them as one entity (i.e. flow of people moving into the direction of an open door, people climbing the same stairs, people walking in the same room, etc.). The groups exhibit both external dynamic (relative to the scene) and internal dynamic (relative to the included individual agents).

Object: an object that is part of the scene and with which the agents can interact during their physical movements (i.e., doors, stairs, floor, shelves, etc.). Typically they do not change their relative position within the scene and remain static for a period of time.

Scene: well-defined boundaries where each agent can move (i.e. a room in a building or a compartment in a transport vehicle). It provides the basis for coordinates of the restricted micro-world observed by a video camera.

The dynamics of the scene is analyzed through recognition and classification of various *events*, *activities* and *situations* which are observed within the visual scene. They correspond to the real-life dynamics observed by the CCTV cameras.

2.3 Changing the location

The key aspect of online simulation is the execution of agent's trajectories in real time. However, depending on the purpose of the simulation, the trajectory information does not have to come from a camera. With the possibility to use gravity and incoming data on agent's movements arriving at a constant rate, it is possible to calculate the movements of an agent with a relatively high precision, absolutely sufficient for visual analysis of the dynamic behavior. Calculating the positions in

the next frame, when moving horizontally along a straight line, is implemented easily using vector calculus. The position is determined on the basis of the current location, the relative velocity of movement and the forward direction vector [14]. But when the agent moves on a curved path, its position needs to be calculated through combining the motion formula with some kind of rotation. Our algorithm is inspired by the ideas of Reynolds [19], which are especially appropriate for real-time simulations due to the fact that the speed of movement is not relative to the visual frames and thus, it does not depend on the speed of the simulation. The new position is calculated using quaternions, while the actual position of the agent is used only for smoothening of the trajectory. An example of such a curved trajectory is shown in Fig. 2. The calculations in this case are relatively simple and can be done in real-time.

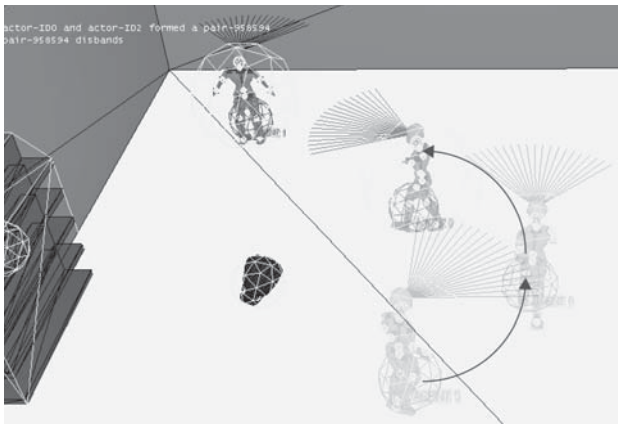

Fig. 2. An agent moving along curved path.

2.4 Gathering of agents and grouping

Unlike the statistical approach to simulation used in crowd behavior, which performs well on a macro level but do not give much on a micro level [1,6-8], we base the behavior analysis on the individual agent's behavior. It still allows group behavior analysis for small groups in enclosed spaces, or "mini-crowds", which is within our scope of interest.

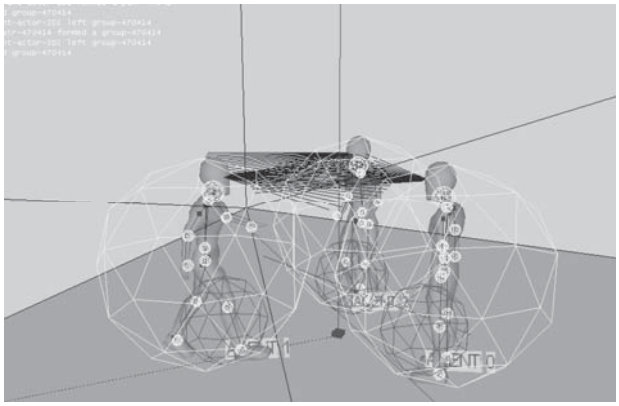

Fig. 3. Gathering of multiple agents in the visual scene

In our approach the individual and group dynamics are linked logically and not statistically. This is a critical feature of our approach to group behavior analysis since our main aim is to analyze the individual and group dynamics from the point of view of the individual interactions and interrelations between the agents in the scene. It is essential, for example, to be able to establish when a group is being formed but to continue tracking both the individual members of the group as well as the group as a whole, because the individual behavior of the agents within the group are superimposed. Fig. 3 shows a slightly more "crowded" scene with a number of agents wandering around while being in a group. It is also important to be able to analyze the group dynamics in relation to other groups which may exist in the same scene. In our case this is possible thanks to the logical approach adopted to grouping.

3 Events on the visual scene

The simulation plays a dual role in our framework. On one hand, it is used for formation of the dynamic patterns of behavior. On the other hand, it allows generating additional information relevant to the agent's behavior which is based on the laws of physics and the logics of the visual scene.

During the construction of our simulator we have incorporated a number of techniques widely used in game programming [17] and robot control systems [19]. The most important of them are the invisible bounding box volumes surrounding the agents and the ray casting [21]. The API of jMonkey we used for development utilizes these concepts in the form of listeners known as "ghosts" and "Line-of-Sights" leads (LOS). The "ghosts" in combination with LOS can be used for further enhancement of the control over agent's dynamic behavior:

- to estimate the spatial dimensions of entities within their existing space.
- to extrapolate the trajectories beyond the scene of visibility.
- to calculate the distances between objects on the path of movement or on the line of sight.
- to induce logically new relations between objects, like detecting obstacles in front/sides of the agents, preventing collisions with objects and predicting reactions.

3.1 Identification of the physical space occupied by the objects using "ghosts"

Physical entities within the focus of the camera occupy certain space and their "bounding boxes" are the starting point of the model dynamics built into our framework. The bounding boxes in the visual scene represent objects that have been successfully recognized and delivered as an input to the simulator. With the knowledge of physical boundaries of the objects, we start outlining a set of rules for the event logging strategy such as taking into account the relationships between objects based on proximity values and overlapping of their mutual spaces. These volumes or "ghosts" have been highlighted with yellow wireframes in the simulation to visually evaluate their accuracy at runtime as it is shown in Fig. 4.

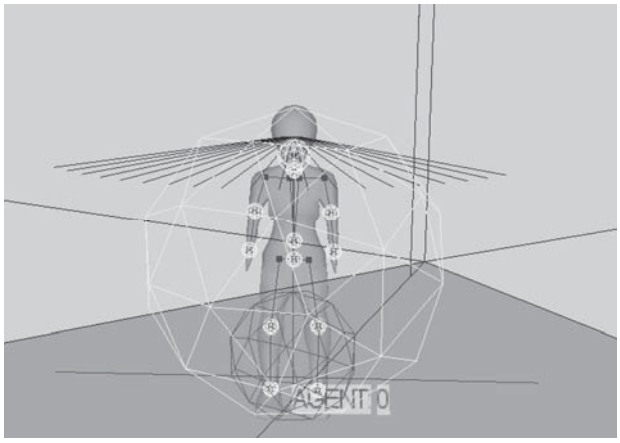

Fig. 4. Ghosts attached to an autonomous agent and its limbs.

Using bounding boxes has the advantage; the boxing does not cause any slowdowns of the simulation since the boxes are not participating in any physical collision calculations. Any recognized object that is passed to the simulator can be equipped with its own "ghost" to support better logging, but the obvious drawback is in the limited area of coverage. This problem is addressed by another technique in 3D graphics programming known as *ray casting*.

3.2 Estimating the physical dimensions using ray casting techniques

The ray casting is a technique that is based on the idea of casting a ray from one point in a specified direction and checking if any geometry comes into contact with it. This will enable us to establish the existence of geometries in a particular area of the visual space (Fig.5). This method is also often called the Line-of-Sight (LOS) and it determines whether two geometries in the environment can "see" each other with respect to another that can cause an occlusion [20]. In our simulator the ray casting technique has twofold usage - firstly it allows us to equip each agent with a "sight sense" and secondly, it enhances the event logging by scanning each agent's surroundings.

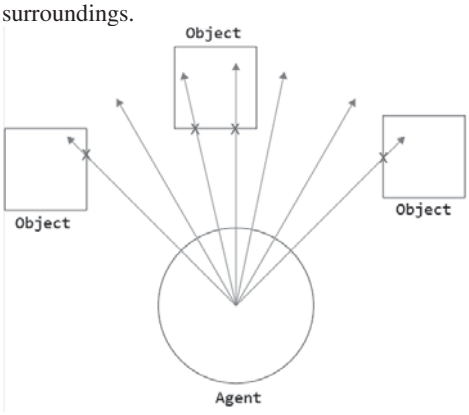

Fig. 5. Ray casting by an agent relative to movement direction

3.3 Detecting obstacles by "sighting" the agent

The full sight sense of an agent has been developed with the use of several rays casted from head position over an arc. The main difference in our approach compared to the case of a single ray casting used in robot motion control [18] lies in the positioning of rays. Each line is being re-rendered with a ¼ π * 0.1 angular offset from the previous one at each frame of the simulation. Because LOS technique has been applied to every agent, it is possible to determine any obstacle that is exactly in front of it. The way we have implemented it approximates the human perceptions, accounting the rules of peripheral vision so that it is possible to deduce agent's focus at a specific time. However, this imposes certain limitations on the way the information about the nearby environment is being gathered since all "sight" rays are being casted towards similar directions, covering a limited front area of the agent only as illustrated in Fig. 6.

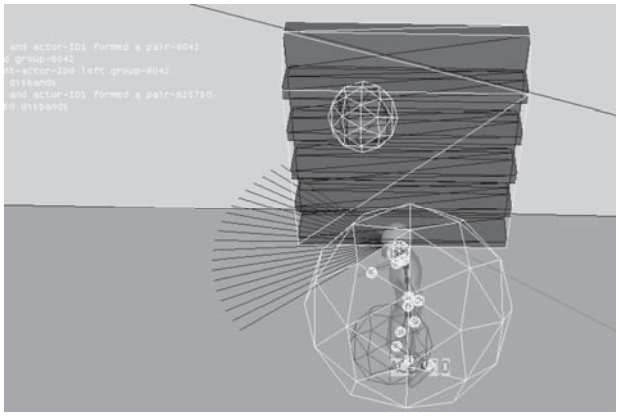

Fig. 6. The sight sense in a form of several rays casted in a viewing direction is insufficient to detect side objects

To eliminate the above limitation, we have to log additional information obtained from other source of probing. In our implementation we have attached special "ghosts" not only to the agents, but also to any object which has been recognized on the visual scene. The difference between the two types of ghosts is that unlike the agent's ghosts, the object ghosts do not cast rays relative to the direction of movement but "reflect" rays along their surface. This allows the agents to move within close proximity without "touching" the objects.

3.4 Establishing relations between agents and the surrounding using ray casting

The main reason for the introduction of ray casting in the simulation is to capture the physical placement of the objects in relation to agent's location by collecting data on entities that come into contact with rays. Through knowing the actor's forward viewing vector, it is possible to calculate its left and right directions, cast rays and gather information on the static entities that are on a side as shown in Fig. 7**.**

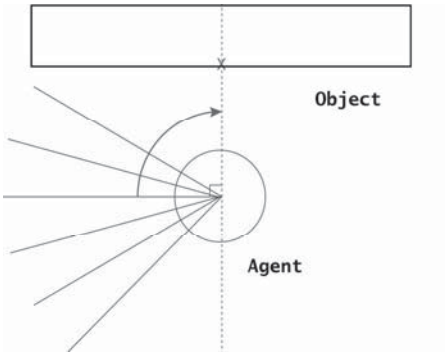

Fig. 7. Rays being casted on each of the agent's sides allow to detect any previously recognized static objects

This simple procedure executed at specific time intervals allows us to store the data in a data structure, sample it separately and potentially report it in the log. By developing this idea further, it becomes possible to recognize when an autonomous entity finds itself "on top of" or "below" a static object. In this case, instead of casting the rays along one axis only, we have to do this along two axes as depicted in Fig.8.

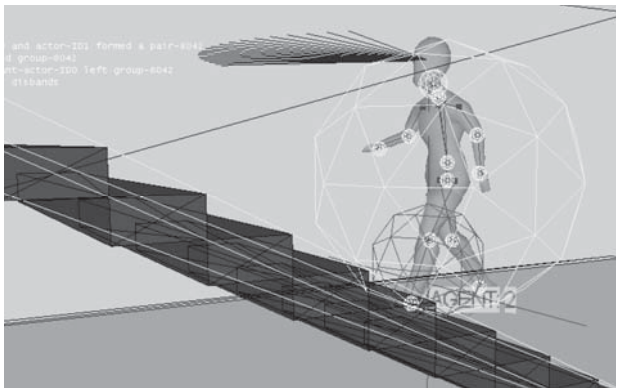

Fig. 8. Casting rays along axes relative to viewing direction to detect changing altitude

We have implemented a number of different single and multi-ray casting algorithms as part of the simulator [14]. At a later stage we plan to analyse the dependence of the rays density from the complexity of the scene in order to estimate the computational power required for the simulation.

4 Social life of the agents

An essential advantage of the agent-based simulation is that it allows analyzing the group behavior using the same mechanisms used to analyze the individual behavior. Our simulator is capable of capturing the agent's "social life", which opens an unlimited opportunity for digging further into the group behavior analysis.

4.1 "Attraction" between agents and coupling

Using the simple concepts of *coupling* and *grouping* of the agents it is possible to develop a sound foundation for analyzing group dynamics in a purely object-oriented manner using efficient algorithms. The fundamental mechanism for simulating group behavior is based on the concept of "attraction" between agents. When two agents detect each other they may become "attracted" and form a pair (Fig. 9).

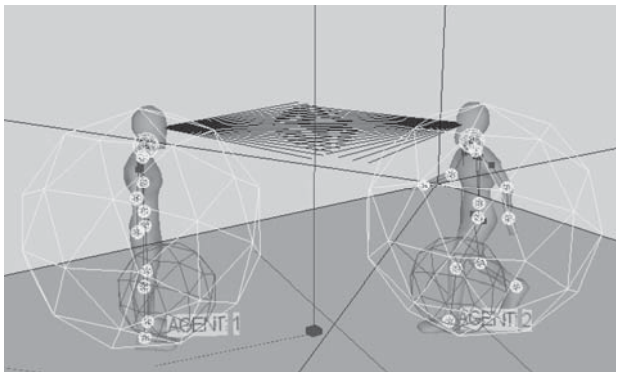

Fig. 9. The moment of "attraction" between agents

This can be used to analyze the group behavior. The main task is to formulate computationally tractable criteria of attraction. In the current version of our framework we are considering the distance between the agents only – the attraction is maximal if the agent's capsules intersect and decreases with the distance between them, which can be easily detected by the agent's ghosts (Fig. 10). In future research we intend to account more complex criteria of "attraction" which include other factors of interest such as direction of movement, as well as non-dynamic factors, such as behavioral attitude.

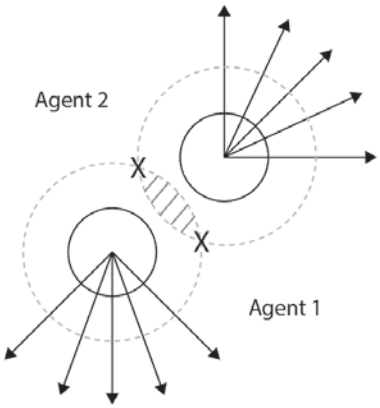

Fig. 10. Detecting coupling through ghost interaction

4.2 Group formation

Our approach to group formation is based on measuring the distances between agents to establish if they are in close proximity. For this purpose, we are calculating a median point out of the physical locations. The reference point allows treating the group as a single entity and by superposition of the individual behavior it is possible to establish a group behavior. It is important to stress that tracking group behavior does not seize tracking the individual activity, so that it is still possible to identify the individual activities in parallel.

4.3 Joining and leaving the crowd

The agents may join or leave already established groups (Fig. 11) which can be formed by gathering individual agents, by merging pairs or by joining a pair of agents by an individual agent. In the current version of the simulator the grouping override the coupling, i.e, the couples are treated as separate individuals within the group and they leave the groups individually, not pairwise. The current version of the simulator assumes that for an agent to join a group, he must first find himself within proximity to a member of an existing group. If, while coupling it is realized that the other agent already belongs to a group, the first agent joins them, but if he himself belongs to a group, the two groups merge, forming a "crowd". Analogically, to state that an agent is leaving the group, the distance from the other members of the group needs to cross certain threshold which is a parameter of the simulation.

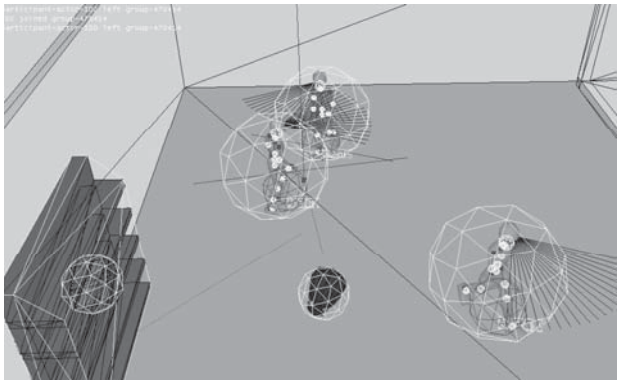

Fig. 11. Joining and leaving a group by individual agents

In our model the groups consist of 3 or more agents and can be formed by gathering of individual agents, by merging pairs or by joining a pair by an individual agent. In the current version of the simulator the grouping override the coupling, i.e, the couples are treated as separate individuals within the group and they leave the groups individually, not pairwise.

5 Implementation

The 3D simulator is written entirely in Java and uses the open source engine jMonkey [14]. This software is widely used in game programming and has been chosen for implementing the platform because it allows modeling of the physical constraints of the micro-world such as gravity and supports additional control mechanisms. We have utilized it for detecting obstacles of the agent's path, collision prevention and navigation control.

5.1 Agents

The model of a humanoid agent was developed using Blender open source modeler [15]. Each agent has an associated "ghost" with it which is equipped with ray casting algorithm for probing the space in order to detect obstacles, navigate through the space and interact with the environment and other agents. Although the agent model allows the use of the full body armature, in the current version of the simulator the gestures are not accounted. However, this will be exploited further in the next version of the simulator.

5.2 Movements

The physical movements of the agents within the visual scene are modelled using methods of standard vector algebra with the addition of quaternions theory to model rotations and movements along curved trajectories. Currently, the movements are calculated on the basis of the position, the direction of movement and the velocity. Although this method is approximate, due to the frequent recalculations, the deviation from the actual trajectories is insignificant for the purpose of the pattern matching and does not affect the quality of the analysis. This allows the analysis to be performed in real-time using entirely simulated data rather than using actual data from the video stream.

5.3 Simulator loop

The event logger works in a loop. At the beginning of each iteration, it updates the current state of the visual scene and then logs all events generated by the individual observers during the simulation. The loop is initialized when a new configuration is introduced as a result of an internal or external asynchronous event in the visual scene.

5.4 Scene changes

At each update the simulator records potential collisions caused by ghosts overlapping or rays piercing physical geometries of the objects. Some of the calculations that are needed for this update can be appropriately timed to reduce the potential frame rate drops and to ensure the data is not coming in too fast to be synchronized. During the experiments it was observed that the delay is not causing any major frame rate drops, but with the increase of the number of objects and "overcrowding" of the scene more substantial computational power may be required to keep the frame drop rate low.

5.5 Event logging

The major role of the simulation is to generate an informative log of the events occurring within the visual scene so that they can be analysed further by pattern matching techniques. In its current version, the simulator generates a log file with time-stamped entries describing each captured event:

```
...
09:39:41 :: Agent ID0 LeftLowArm touches Stairs ID1
09:39:41 :: Agent ID0 LeftHand touches Stairs ID1
09:39:41 :: Agent ID0 RightFoot touches Stairs ID1
09:39:41 :: Agent ID0 LeftFoot touches Stairs ID1
09:39:51 :: Agent ID2 moves towards Bookshelf ID2
09:39:53 :: Agent ID2 moves towards Bookshelf ID2
09:39:56 :: Agent ID2 moves along Bookshelf ID2 on left
09:39:56 :: Agent ID2 moves away from Bookshelf ID2
09:39:57 :: Agent ID2 moves towards Bookshelf ID2
09:39:59 :: Agent ID2 moves along Bookshelf ID2 on right
09:40:41 :: Agent ID2 climbs Stairs ID1 up
...
```

The event logger is implemented with architecture of an "observer", attaching a separate individual logger to each object within the visual scene. The individual observers log all events related to the observed. This allows further extension of the logging module without changing the existing code of the simulator. In the next version of our simulator we plan to incorporate fine grained event logging which account not only for the body motion of the agents but also for their gestures.

5.6 Pattern classification and beyond

The simulator log is parsed for recognizing and classification of the behaviour patterns according to the grammar of its language [22]. After the simulator the behaviour analysis can be continued solely based on the logs, while the original video data can be used to increase the precision of approximation. This approach gives the opportunity to incorporate purely symbolic techniques for behaviour analysis. In a forthcoming article we will report the visual dynamics ontology developed for this purpose. It forms another part of our research program which will be based entirely on semantic technologies.

6 Conclusion

This article introduces a new framework for real-time video data processing for the purpose of individual and group dynamic behaviour analysis based on 3D simulation and dynamic pattern classification. Our approach combines methods from games programming and robotics. The main advantage of this approach is that it allows the analysis of both individual and group dynamics in a single unified manner at different level of granularity depending on the needs. Although the framework is still under development, its core component - the simulator - is already completed and the experimental tests with simulated data in real time look very promising. The pattern classifier, which processes the event log generated during the simulation, for further analysis of the visual scene, is currently under development and will be reported separately in a forthcoming publication. It performs real time parsing according to the grammar of the event logging language and its input is the basis for the development of a suitable notification mechanisms. We plan to extend the language in order to represent more fine grained patterns of behaviour which go beyond the dynamics of pure body motion and include gestures as well.

The work reported here is conducted as part of the PhD research of the first author at London Metropolitan University and is sponsored by The Vinyl Factory of London, UK.

References

[1] S. Gong, T. Xiang, Visual analysis of behaviour from pixels to semantics. London: Springer, 2011.

[2] C. Hu, S. Wo, An efficient method of human behavior recognition in smart environments, in: Int. Conf. on Computer Application and System Modeling (ICCASM), Vol. 12, pp. 690–693, 2010.

[3] K. Yordanova, Modelling Human Behaviour Using Partial Order Planning Based on Atomic Action Templates, in: 7th Int. Conf. on Intelligent Environments (IE), pp. 338–341, 2011.

[4] C. Wang, F. Wang, A Knowledge-Based Strategy for Object Recognition and Reconstruction, in: Int. Conf. on Information Technology and Computer Science (ITCS), pp. 387–391, 2009.

[5] M. Attamimi, T. Nakamura, T. Nagai, Hierarchical multilevel object recognition using Markov model, in: 21st Int. Conf. on Pattern Recognition (ICPR), pp. 2963–2966, 2012.

[6] S. Wu, B. Moore, M. Shah, Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition CVPR2010, pp. 2054–2060, 2010.

[7] P. Saboia, S. Goldenstein, Crowd Simulation: Improving Pedestrians' Dynamics by the Application of Lattice-Gas Concepts to the Social Force Model, in: 24th SIBGRAPI Conf. on Graphics, Patterns and Images (Sibgrapi), pp. 41–47, 2011.

[8] R. Guo, H. Huang, A mobile lattice gas model for simulating pedestrian evacuation, in: Physica, Part A: Statistical Mechanics and its Applications, Vol. 387, pp. 580–586, 2007.

[9] L. Hluchy, M. Kvassay, S. Dlugolinský, B. Schneider et al., Handling internal complexity in highly realistic agent-based models of human behaviour, in: 6th IEEE Int. Symp. on Applied Computational Intelligence and Informatics (SACI), pp. 11–16, 2011.

[10] A. Varas, M. Cornejo, D. Mainemer, B., Toledo et al., Cellular automaton model for evacuation process with obstacles, in: Physica A: Statistical Mechanics and its Applications, Vol. 382, pp. 631–642, 2007.

[11] X. Ben, X. Huang, Z. Zhuang, R. Yan, S. Xu, Agent-based approach for crowded pedestrian evacuation simulation, IET Intelligent Transport Systems, Vol. 7, pp. 56–67, 2011.

[12] S. Sharma, S. Lohgaonkar, Simulation of agent behavior in a goal finding application, in: IEEE Southeast Conf. (SECON), pp. 424–427, 2010.

[13] E. Lengyel, Mathematics for 3D Game Programming and Computer Graphics, 2nd ed., Hingham, MA: Charles River Media, 2003.

[14] R. Eden, JMonkeyEngine 3.0 Cookbook, Birmingham: Packt Publ., 2014.

[15] G. Fisher, Blender 3D Basics, 2nd ed., Birmingham: Packt Publ., 2014.

[16] A. Bogdanovych, M. Bauer, S. Simoff, Recognizing Customers' Mood in 3D Shopping Malls Based on the Trajectories of Their Avatars, in: Filipe, J., Cordeiro, J. (Eds.), Enterprise Information Systems, LNBIP, Berlin: Springer, pp. 745–757, 2009.

[17] M. Wang, H. Lu, Research on Algorithms of Intelligent 3D Path Finding in Game Development, in: Int. Conf. on Industrial Control and Electronics Engineering (ICICEE), pp. 1738–1742, 2012.

[18] T. Terzimehic, S. Silajdzic, V. Vajnberger et al., Path finding simulator for mobile robot navigation, in: XXIII Int. Symp.on Information, Communication and Automation Technologies (ICAT), pp. 1–6, 2011.

[19] A. Croitoru, A., Deriving Low-Level Steering Behaviors from Trajectory Data, in: Proc. IEEE Int. Conf. on Data Mining Workshops (ICDMW), pp. 583–590, 2009.

[20] B. Salomon, N. Govindaraju, A. Sud, R. Gayle, M. Lin, D. Manocha, Accelerating Line of Sight Computations Using Graphics Processing Units, in: Proc. 24th Army Science Conference, 2004.

[21] J. Beaudoin, J. Hughes Clarke, J. Bartlett, Application of Surface Sound Speed Measurements in Post-Processing for Multi-Sector Multibeam Echosounders, International Hydrographic Review, Vol. 5, No. 3, pp. 26-31, 2004.

[22] D. Grune, C. J. H. Jacobs, Parsing Techniques: A Practical Guide, 2nd ed., NY: Springer, 2008.

[23] 3VRVideoIntelligence Platform [http://3vr.com/products/videoanalytics last visited: 31-05-2016]

[24] savVi Real-Time Event Detection [http://www.agentvi.com/61-Products -282-savVi_Real_Time_Event_Detection; last visited: 31-05-2016]

[25] PureTech Systems VideoAnalytics [http://www.puretechsystems.com/ video-analytics.html; last visited: 31-05-2016]

[26] Indigo Vision Control Center [https://www.indigovision.com/en-us/ products/management-software/control-center; last visited: 31-05-2016]

[27] IBM Intelligent Video Analytics [http://www.ibm-3.com/software/ products /en/intelligent-video-analytics; last visited: 31-05--05-2016]