Extracting Data from Second Life

Surangika Ranathunga, Stephen Cranefield and Martin Purvis
Department of Information Science,University of Otago,
PO Box 56, Dunedin 9054, New Zealand
(surangika, scranefield, mpurvis)@infoscience.otago.ac.nz

Second Life is a multi-purpose online virtual world that is increasingly being used for applications and simulations in diversified areas such as education, training, entertainment, and even for applications related to Artificial Intelligence. For the successful implementation and analysis of most of these applications, it is important to have a robust mechanism to extract low-level data from Second Life in high frequency and high accuracy. However, currently Second Life does not have a reliable or scalable inbuilt data extraction mechanism, nor the related research provides a better alternative. This paper presents a robust and reliable data extraction mechanism from Second Life. We also investigate the currently existing data extraction mechanisms in detail, identifying their limitations in extracting data with high accuracy and high frequency.

1 Introduction

Second Life is a multi-purpose online virtual world that has already proven its use beyond being just an entertainment medium. Apart from the millions of individuals who have registered with Second Life, many private and public institutions have started to use it, to develop applications in diversified areas such as education, health, business, and entertainment. Some universities are using Second Life as an alternative medium to conduct online lectures (Gregory and Tynan, 2009; Nash, 2009; Ryan, 2008; Shi et al., 2010; Storey and Wolf, 2010), while many others, such as medical institutions are using Second Life simulations for training purposes (Diener et al., 2009; Rogers, 2009; Taylor et al., 2009). Some researchers are also using Second Life in Artificial Intelligence related research, because with Second Life, it is possible to create more sophisticated, yet much cheaper and convenient Artificial Intelligence (AI) related simulations (Veksler, 2009; Weitnauer et al., 2008).

Extracting data from Second Life environments with high accuracy and high frequency is paramount for the successful operation of most these applications. For example, for training simulations, it is possible to analyse performance of the participants using these extracted data. As another example, when implementing intelligent virtual agents inside Second Life, these agents should be able to extract their sensory information from second Life environments, for their deliberation process.

Accurately extracting low-level data from Second Life is difficult because of several reasons. First and foremost, it is not possible to re-use the already existing data extraction mechanisms suggested for other virtual worlds (e.g. (Borner et al., 2002)), in the context of Second Life, because there is no commonly agreed design or protocol among the currently available virtual worlds. Data extraction is even harder, given the proprietary nature of Second Life. Second Life has a complex design and an interface. An individual can perform diversified actions while inside Second Life, when compared to 2-D applications.

Research focused on extracting data from Second Life so far has mainly focused on extracting spatio-temporal data from arbitrary Second Life 'regions' for statistical purposes (Eno et al., 2009; Varvello et al., 2008), rather than using this data to reason on resident interactions that take place in a given region or a simulation. Data extraction in most of these research did not involve recording data in high accuracy or high frequency. Therefore the related problems have not been properly addressed.

In order to come up with a robust data extraction mechanism, we have conducted a comprehensive analysis of the available data extraction mechanisms for Second Life, and this paper carries that information as a guide to be used in future research work. A comprehensive analysis of this nature cannot be found in the existing literature related to data extraction from Second Life, mainly because there was no focus on

accurately extracting spatio-temporal data as explained earlier.

There are two main mechanisms to extract data from Second Life, and both have their strengths and limitations. Moreover, depending on the characteristics of a given simulation, various complications may arise when extracting data. Therefore we introduce two different data extraction mechanisms that work for many of the simulations in general. These new data extraction mechanisms are based on the strengths of both traditional data extraction mechanisms, and have been able to eliminate the drawbacks of the same.

We demonstrate the use of these two data extraction mechanisms in the context of two different simulations: the SecondFootball (Vstex Company, 2010) simulation in Second Life, and the Otago Virtual Hospital Simulation (OVH) simulation[1] in the New Zealand Virtual World Grid (NZVWG). The NZVWG is developed using the OpenSimulator virtual world builder toolkit (OpenSimulator, 2010), which supports the Second Life protocol. These two simulations have different requirements with respect to extracting data, and our data extraction mechanisms cover most of these requirements.

We have conducted a performance evaluation to measure the feasibility of using these data extraction mechanisms for practical applications, and have obtained positive results.

The rest of the paper is organised as follows. Section 2 contains an introduction to Second Life. Section 3 discusses the challenges and problems in accurately extracting spatio-temporal data from Second Life environments using the existing mechanisms. Section 4 presents the new mechanisms that are developed to accurately extract spatio-temporal data from Second Life and Section 5 demonstrates how these data extraction mechanisms have been used to extract data from two different Second Life simulations. Section 6 provides an evaluation of these mechanisms with respect to Second Life server performance. Section 7 contains related work, and finally Section 8 concludes the paper.

2 Second Life

Second Life is commercially owned by the company Linden Lab, and is publicly available over the Internet. Among other similar virtual worlds, Second Life clearly has a lead with a much higher number of user subscriptions than its competitors, and claims to have an average monthly repeated user logins at around 800000[2].

The Second Life virtual world is simulated in the form of land surrounded by water and new land may be designed by the Linden Lab as and when it is needed. Second Life land is maintained in regions of size 65,536 m^2, and an area of land owned by a user is termed a 'land parcel'. A Second Life resident can explore most of the regions except for those that are set as regions with restricted access allowing only the authorised residents. It is also possible to buy land for a moderate amount of fee.

A user inside Second Life is called an 'avatar', and can access Second Life using any client software that adheres to the Second Life communication protocol. Once in world, an avatar can undertake basic activities such as exploring the environment (by walking, running, flying or 'teleporting'), interacting with other residents, interacting with objects, or building content and interactive applications. As a result of Second Life being an immersive environment, a user feels a sense of presence which feels real. This experience is enhanced even further by the ability to personalise one's surroundings, as well as how one looks.

Avatar interaction is mainly based on text or voice chat, and collaborative content construction is also possible. It is also possible to join many available user groups that share different interests and or objectives. The ability to create groups is an important feature of Second Life which further enhances the social interaction opportunities.

Activities taking place inside Second Life cover a wide range and it can be seen that people are trying to use it as a supplementary medium to carry out real-life communication, business, education and social activities. If categorised further, some interesting virtual counterparts can be found for areas such as media, art and culture, sports and leisure and even politics. Linden Lab has interconnected Second Life with the

[1] http://hedc.otago.ac.nz/magnolia/ovh.html

[2] http://blogs.secondlife.com/community/features/blog/2011/01/26/the-second-life-economy-in-q4-2010

Figure 1. Southern Institute of Technology, New Zealand - The Invercargill Campus main building and its Second Life counterpart

World Wide Web, as an attempt to provide seamless integration between these two mediums and this has contributed to the increasing popularity of Second Life.

There are regions owned by various universities and organisations where they have attempted to create their virtual presence inside Second Life. Most of these places contain informative materials in the form of virtual displays, links to their official web sites, or even demonstrations. Some of these places are popular among residents and are hang-out places for those who share common interests. There are initiatives in experimenting with these virtual spaces for utilising them as a new medium of online teaching where people living in different parts of the real world can easily participate in a familiar environment (Gregory and Tynan, 2009; Nash, 2009; Ryan, 2008; Shi et al., 2010; Storey and Wolf, 2010). Some universities and organizations have gone beyond this level as well and are trying to create interactive simulations to train people. Medical institutions have paid a particular interest in this possibility (Diener et al., 2009; Blyth et al., 2010), mainly because of the risk and cost factors involved in real-life medical simulations.

Keeping its promise to provide a virtual world that is imagined and created by residents, Second Life facilitates its residents to create their own content to satisfy their requirements, and claim ownership of what they create. This autonomy contributes to the enhancement of Second Life user experience, as opposed to game worlds with a fixed story line. Many organizations have made this an opportunity to create content such as buildings and interior designs that replicate their real life counter parts to provide a much real experience to those who come in. Figure 1 shows how the Invercargill Campus of the Southern Institute of Technology, New Zealand has been replicated in Second Life. The ability to create customised content, together with the ability to animate them using embedded scripts has also made it possible for these institutions to create custom simulations, which further enhances the use of Second Life for applications in many areas. For example, Figure 2 shows part of the Emergency Department simulation created in the Ann Myers Medical Center in Second Life (Ann Myers Medical Center, 2010).

2.1 *Second Life Architecture*

Not much information is available on the design and architecture of Second Life, due to its proprietary nature. Second Life has a client/server architecture with a thin client. Clients running in user machines carry out a relatively low amount of work such as 3D rendering of the virtual world and avatar and object position extrapolation. Researchers argue that Second Life has serious scalability issues (Varvello et al., 2008) and there are research initiatives to try an alternative peer-to-peer communication architecture that would be more scalable (Varvello et al., 2009).

The Second Life server side is comprised of a large number of interconnected server processes called simulators or 'sims' (Figure 3-left). Each process is responsible for handling one or more regions in the virtual world. These interconnected simulators are collectively known as a grid and they constitute the primary part of the server infrastructure. Simulator servers are responsible for managing the states of objects, land parcels and terrain height-maps. They transmit the calculated object and land data to

Figure 2. The Ann Myers Medical Center - Emergency Department simulation (source (Ann Myers Medical Center, 2010))

clients. They also do virtual physics simulation with the use of the third party physics engine Havok. Additionally, there are other supporting servers such as a login server, a user server, and a data server that handle additional services such as user log-in, message routing and asset management (Figure 3-right). A somewhat comprehensive description of the Second Life server architecture is given by Fernandes et al. (2007), Thumann (2008) and Linden Lab (2010).

A simulator can easily get overloaded because of the limited resources and processing power it has. Overloading causes slow simulator reaction time which is commonly known as lag. Although the environment lag perceived by a user could be due to both client side settings as well as server-side issues, the latter is more problematic as it affects all the users who are connected to that server.

Simulator lag is mainly generated by physics-enabled objects because the simulator has to continuously carry out physics calculations, interactive scripts with timers and listeners, and frequently changing textures. Presence of a large number of avatars, and their high movement speeds (when running or flying) also generate simulator lag, because the server has to carry our their position and velocity calculations often and communicate that information to all others. Researchers claim that servers would get overloaded with even 20 concurrent avatars (Varvello et al., 2008), while some claim that it is as high as 40 for a typical region, and this number could be increased by more careful design (Luderschmidt, 2010). Linden Lab has introduced some proactive measures to reduce server lag, by limiting the number of avatars and objects that can co-exist inside a region. Land owners also have the option of disabling arbitrary object creation inside their land. As a reactive measure, a simulator can slow down its processing speed. Under normal conditions, a server would process 45 frames/second but amidst a heavy processing load, it can reduce the number of processed frames using a method called ‘time dilation’, where the server slows down the virtual time so that avatar and object activity is slowed down.

3 Mechanisms to Extract Data From Second Life

There are two main mechanisms to extract data from Second Life. Below we provide a comprehensive analysis of these two mechanisms along with their limitations and drawbacks in extracting spatio-temporal data.

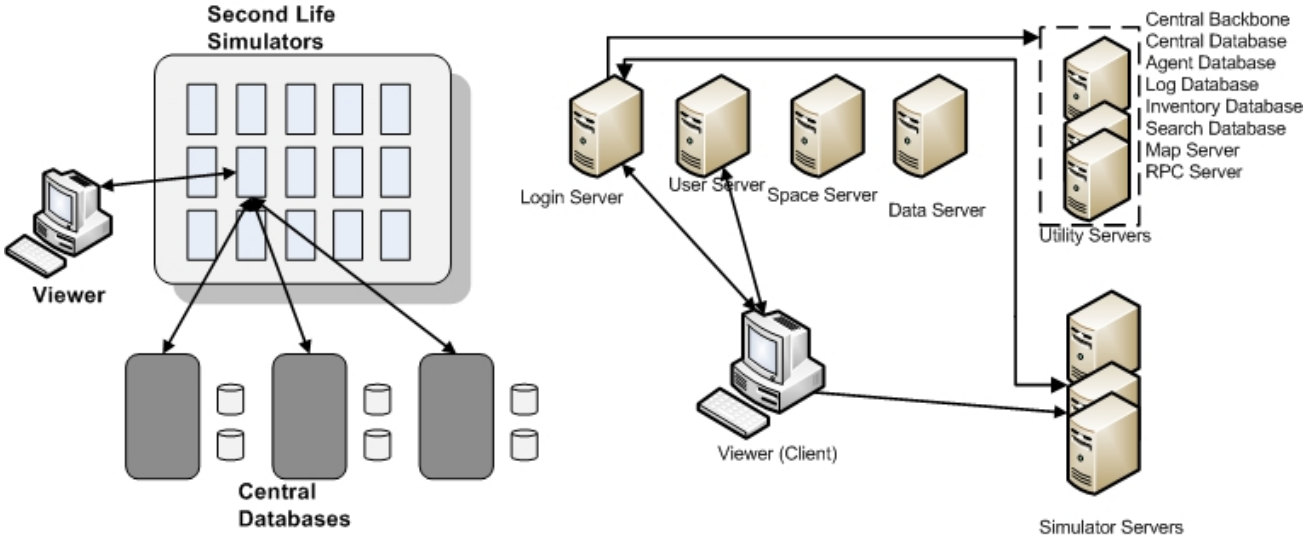

Figure 3. left- The Second Life Grid (source Linden Lab (2010)), right- A closer look at the Second Life architecture (source Fernandes et al. (2007))

3.1 *Linden Scripting Language (LSL)*

The traditional programming in Second Life is done using in-world scripts created using the proprietary language *Linden Scripting Language (LSL)*. Linden Scripting Language is the programming language capability provided to end users by the Linden Lab. These scripts are run inside objects, and can also be saved into a resident's 'inventory'.

LSL elaborates on the concept of a state engine with different states and events, so that code segments can be separated on the basis of states. Every script should have at least one state which is called the default state, and additional states can be added as needed. Events can be considered as triggers, and they exist in the context of a state. Once a particular state is active, events corresponding to it wait to be triggered. Apart from states and events, LSL has pre-defined functions, constants and data types that can be used in scripts.

Limitations of LSL begin with the size of its API. Currently, it has only around 300 library functions and about 30 event types. The programming flexibility of LSL is also hindered by the number of data types it supports. A detailed description of LSL and its limitations is presented by Cox and Crowther (2009).

LSL scripts also suffer from heavy memory limitations, and they are one of the contributors to server side lag. The lag introduced by a script depends on the amount of work it is doing, the frequency of execution of repetitive tasks, and the duration of the running script.

However, some residents claim that the new Second Life server implementation gives less priority to script execution now, as an attempt to reduce server lag introduced by scripts. Though this has not been officially clarified by Linden Lab, this suggests that while a server may not show any lag, scripts can be running much slower, amidst of heavy work load.

Spatio-temporal data extraction using LSL scripts is heavily affected by the aforementioned LSL limitations plus limitations of the individual functions used for data extraction. Data extraction is done using script-enabled objects that act as sensors inside Second Life. The sensor function can be executed periodically, and it can detect avatars and/or objects. Information recorded by the sensor may include, among other things, position data of avatars and or objects and names of the animations that the avatars are currently playing. The sensor function introduces considerable amount of lag to the Second Life servers, when executed continuously. Although some light-weight functions are available to extract position and related movement information, these require the UUIDs (Universally Unique Identifier) of the avatars and objects that should be monitored, which is not something that can be assumed to be available all the time. Listed below are the specific problems relating to in-world sensors:

- A script can only be attached to an object. However, there are some regions that do not allow the creation of scripted objects, and in this situation, the scripted object should be worn by an avatar in

that region
- A sensor can detect only 16 avatars and or objects in one round and the maximum sensor range is limited to 96 meters
- The use of multiple sensors to cover a larger area would introduces more lag to the system
- A sensor can only record animations that are being played by avatars at the time of scanning, hence animations with a short duration such as clapping or laughing may be missed out by the sensor
- The sensor repeating interval cannot be set to millisecond levels, as sensing is a costly operation. The recommended interval size is in the order of minutes, not seconds. This means the frequency of periodic data extraction is relatively low, thus missing out a considerable amount of movement and animation data
- Complex data processing cannot be carried out inside a script due to the limitations in memory, and the available LSL programming constructs. Complex processing would also introduce more lag to the system
- There is no provision to keep the recorded data in-world as an LSL script cannot write to Second Life 'Note Cards' which act as in-world text files
- Sending information out from Second Life using HTTP is a slow operation. HTTP requests are throttled to a maximum of 25 requests per 20 seconds, so approximately one request can be sent in one second

3.2 *LIBOMV Open Source Library*

"LIBOMV is a .Net based client/server library used for accessing and creating 3D virtual worlds" (OpenMetaverse Organization, 2010), and is compatible with the Second Life communication protocol. The LIBOMV library can be used to create Second Life clients, and they need a valid username and password to connect to Second Life. With appropriate programming techniques, the LIBOMV library can be used to create avatars that exactly look like and have behavioral abilities similar to those controlled by humans. Abilities of a LIBOMV 'bot' is two-fold. The first types of abilities include performing actions inside Second Life including moving abilities such as walk, run or fly, performing animations such as cry or laugh, and communication abilities using instant messaging or public chat channels.

The second type of abilities include sensing or 'seeing' the environment around. This includes the ability to receive updates of the changes that take place in the environment around it, and receiving messages posted in the chat channel or instant messages sent to it. Updates received by a LIBOMV client are generated whenever there is a change in the movement parameters such as velocity and angular velocity of an avatar or object, or whenever there is a change in an avatar animation. Thus, data received by a LIBOMV client shows similar traits to what a human controlling an avatar can see.

At its lower level, the LIBOMV library has functions to send packets to the server and receive response packets from it. Information in these packets is processed and data structures of the client environment are updated accordingly. Most of the client functionality is based on event delegation. Depending on the data contained in the received packet, the related event is triggered and, by handling the event, data sent in the update can be further processed.

Ideally, a LIBOMV bot should be able to receive updates corresponding to objects and avatars that are in its visibility range. However, some irregularities can be observed in this where the bot receives updates of objects and avatars that are clearly out of the visibility range, or the bot does not receive updates for avatars and objects that are in the visibility range. This irregularity is more visible with small moving objects [1].

There are several advantages of using LIBOMV over LSL scripts to record temporal and spatial data of Second Life avatars and objects. LIBOMV clients operate completely outside the Second Life server environment hence they introduce no lag to the Second Life servers. Since the code is running in a user machine, recorded information can be further processed or can easily be saved into a back-end storage such as a database.

[1] It should be noted that these observations are based on the current version of the LIBOMV library. We currently use version 0.8.

A major limitation of the LIBOMV-based approach for data extraction is its inability to extract accurate movement data. As with other viewer clients, the Second Life server sends information to the LIBOMV client only if there is any change in the environment perceived by the bot. This is in order to reduce the server workload. This means that the client has to 'assume' its perceived environment. For objects and avatars that are moving constantly, the client has to keep on extrapolating their position values based on the previously received position and velocity values until it receives an update from the server. Therefore the extrapolated position values may not be completely in tally with the server sent values. This might mean the client has to snap back avatars and objects into their correct positions once the server update is received. This situation is commonly known as 'rubber banding' and is visible in all Second Life viewer clients, and the LIBOMV client is no exception. The rubber banding effect is much greater for avatars and objects that move fast.

One contributor to this problem could be the change in the time dilation. The server sends the current time dilation value to the clients only when it sends an update corresponding to the position information. This time dilation value is to be used in position extrapolations, because the time dilation is the main factor used by a Second Life server to reduce its processing speed. Thus, how fast an avatar or object can move depends on the time dilation value set by the server. In the absence of any position update from the server, the client has to rely on the already received time dilation information for position extrapolation. However, it could well be the case that the time dilation value has changed in the server side, meaning that the server is using a different time dilation to calculate positions of avatars and objects.

The LIBOMV client receives updates of movement or animation changes of individual avatars as and when they occur, as opposed to the data recorded by an LSL script. Therefore these individual updates do not contain information of other avatars or objects that are present in the same environment. Moreover, a received position update for an avatar does not contain information of the animation the avatar is currently playing, neither does a received animation update contain the avatar's current position and movement information. This means that additional data processing mechanisms should be used on the received data to understand what is taking place in a Second Life environment in a given instance of time. On the other hand, it is always desirable to record all the low level data in a given instance of time, in order to derive the high level complex events that took place, if this is needed.

Another drawback of using LIBOMV bots for data extraction is the presence of another avatar in the environment that can be seen by other avatars and sensed by any script, although it does not introduce any considerable amount of lag into the Second Life servers. If the bot is acting on behalf of a virtual agent, then its presence can be justified. However, a bot just acting as a sensor may cause distraction and may be confusing to others. On the other hand, the use of bots in Second Life is nothing new. Some land owners even use bots to show non-zero avatar presence in their land. In fact, Varvello et al. claim that about 5% of avatars found in-world at any given time are bots (Varvello et al., 2008).

Another hindrance to develop LIBOMV-based data extraction mechanisms is the unavailability of proper documentation of its design details and usability options.

4 A Robust Approach to Extract Data from Second Life

As discussed in Section 3, when extracting spatio-temporal data from Second Life with high accuracy and high frequency, both available approaches exhibit their own strengths and weaknesses. Moreover, the most suitable data extraction method should be decided based on the type of simulation on focus. Second Life simulations vary in many aspects, such as in the number of avatars and objects present in the simulation, amount of mobility exhibited by these avatars and objects, their moving speed, area that is covered in the simulation and the performed actions such as playing animations or gestures and clicking on scripted objects.

It is important to identify the types of spatio-temporal data that should be extracted from Second Life. This mainly includes position and movement information of avatars and objects, animation information of avatars, and chat messages exchanged in the environment. In some simulations, it is also useful to detect whether an avatar clicked a specific object or not.

A LIBOMV client alone can be used to extract the first three types of data from Second Life. However,

in order to detect whether an avatar clicked an object or not, the script in the object being clicked should pass that information into a private or public chat channel. While a LIBOMV client can listen to the chat exchanged on the public chat channel, it is not capable of listening to chat that is being exchanged in the private chat channels.

Despite the type of simulation on focus or the types of data to be recorded from a simulation, it is always convenient to have one mechanism to accurately extract data from any Second Life simulation. However, in practice, it is difficult to rely on one such generalised mechanism. Therefore we have introduced a novel data extraction mechanism that combines the strengths of both these data extraction mechanisms, as well as an enhanced version of the LIBOMV-based data extraction. It is possible to decide on the most suitable approach from these two, based on the characteristics of the given simulation.

4.1 *A Combined Approach to Extract Data from Second Life*

In this combined approach, we attach a simple scripted object to a LIBOMV bot. The LIBOMV client detects avatars and objects in its viewing range and records their UUIDs. It then sends this UUID list to the script. The LIBOMV client continuously checks for new avatars or objects that entered the Second Life environment or simulation, based on the movement and animation updates it receives. Whenever a new avatar or an object is detected, the UUID list is updated with the newly found UUID and the list is sent to the script. This makes sure newly arrived avatars and objects are also detected. Since the script receives UUIDs of the avatars and objects, it can use the `llGetObjectDetails` LSL function to extract position and velocity information for the corresponding avatar and or object. `llGetObjectDetails` is a light-weight function and is much faster than the sensor operation because it is sensing a given set of avatars and objects. This function is repetitively called for the UUIDs in the received string. When trying to locate an avatar or an object, this function covers the entire Second Life "region". Every 500 milliseconds, `llGetObjectDetails` is executed and the collected data are sent back to the LIBOMV client. However, this function may return information for avatars that can no longer be found for a short period (about 45 seconds). If the `llGetObjectDetails` function cannot detect an avatar or an object specified by a given UUID, an empty string is returned instead of the recorded position and velocity information. In this case, the corresponding UUID is removed from the globally maintained UUID list and the list is re-sent to the script, in order to avoid searching for avatars or objects that are no longer there.

This function cannot detect the animation that an avatar is currently playing. Therefore we use the LIBOMV client to directly receive the animation changes of avatars. Communication taking place in the public chat channel is also captured directly by the LIBOMV client.

In this combined approach, the accuracy of extracted movement data is high, because the position extraction is done on the server side. The extracted values are always server generated, rather than extrapolated by the client to predict the positions of objects and avatars. This approach is well suited for a simulation containing avatars and objects that are moving constantly, and especially if it is not guaranteed that the avatars and objects will always be in the viewing range of the bot (although this approach records position information of avatars and objects that go out of the viewing range of the bot, it misses out animation changes that take place while an avatar has gone out of its visibility range).

This approach adds one more active script to simulations that may already have a large number of running scripts. Therefore this approach may not be suitable for a highly lagged simulation, as it may increase server lag. This is because the script contains a listener that is triggered every 500 milliseconds, although not much work is done in one round of execution. Moreover, the lag introduced by the script may increase when the number of avatars and or objects that have to be detected is increased, because this decides the number of iterations of the `llGetObjectDetails` function in one round.

Although as reported in Section 6 our experiment results did not show a considerable increase in server lag at the presence of the data extraction script, the client side experienced some lag when concurrent scripts are executed. Moreover, there can be some simulations that expect the participants not to wear any scripted objects.

Therefore, for simulations that contain avatars and objects that seldom move, and if it is not very critical to extract accurate position values for the avatars, we have introduced another mechanism that is based

on extrapolation of the position and motion information received by the bot.

4.2 *An Enhanced LIBOMV Approach to Extract Data from Second Life*

In this approach, we maintain data structures which correspond to avatars and objects that should be monitored. Attributes of a data structure include, among other things, position, velocity and acceleration information corresponding to a specific avatar or object, and avatar animation information. The data structure always maintains the latest information of these properties. A timer is used to extrapolate position and velocity values of the avatars and objects recorded in the latest data periodically. This ensures that the current movement information of avatars and objects closely represents the actual movement information of those inside the server. However, avatar animation information stored in the latest data structure is not updated during these periodic extrapolations. The LIBOMV client receives an update corresponding to every avatar animation change (except when the avatar is not captured inside the viewing range of the bot), therefore it is safe to assume that an avatar keeps on performing the animation already recorded.

We have currently set the timer interval to 500 milliseconds, but this interval can be increased or decreased as required. Whenever an update is received from the server, this is used as the latest property values of the corresponding avatar or object. In order to keep the recorded data as accurate as possible, we extrapolate the position and movement information of an avatar, if the received update is an animation update. This way, movement information of the avatar closely represents the movement values of the avatar at the time of the animation update. However, this still does not contain information of other avatars or objects in the environment, at that time instance. Therefore we extrapolate movement information of all the other avatars and objects each time we process a movement or animation update received for a single avatar or object. Whenever a communication update is received, we extrapolate and generate the movement and animation information of all the avatars and objects in the Second Life environment, in order to create a clear picture of the environment, at the time the communication message took place.

The main drawback of this approach is that the LIBOMV bot may not receive updates of avatars and objects that go out of its viewing range. In this situation, the extrapolated position values are wrong, as well as the recorded animation information of those avatars. Moreover, when an avatar or object is moving constantly, their extrapolated position and velocity values may not be completely accurate.

In order to receive information on the touch events performed by avatars on objects, it is important that the objects being touched pass a message to the bot, or communicate it in the public chat channel. This means that the scripts running in that simulation should be changed so that they pass this message to the bot, with a special mark to identify that the message is related to an object being clicked. However, if these objects are communicating through private chat channels (which normally happens in most of the simulations), it is possible to make the script running inside the object attached to the bot listen into those private chat channels. This way, the need to change all the scripts running in the simulation does not arise and the scripted object attached to the bot can pass all the messages it recorded from these private chat channels to the bot. This suggests that the LIBOMV bot should still wear its scripted object in this second approach as well, even though it is not used to extract position information as in the first approach.

Data extracted using these two methods have one main difference. While the second approach relies on the movement updates generated by the server to decide on avatar and object positions and velocities, the first approach ignores these movement updates. The first approach relies only on the script-sent movement information. Therefor data extracted from the same Second Life simulation using these two methods may not be identical.

4.3 *Second Life Events that Cannot be Recorded*

There are some events that cannot be detected using either of these techniques. One example for this is exchanging money between two residents or making a payment via an object. In situations like this, the residents and/or objects involved in the transactions must communicate this information through public or private chat channels, send that information to the bot as an instant message, or use some other method to

export the event information from the virtual world. A design for such instrumented "intelligent objects" has been proposed by Rodriguez et al. (2008).

The other information that cannot be recorded using either of these two approaches is the instant messages exchanged between two avatars, as this is considered private information. However, the LIBOMV bot can receive instant messages sent to it by other avatars, as well as messages sent to a group of which the bot is a member.

5 Extracting Data from Different Second Life Simulations

As mentioned in Section 4, Second Life simulations vary in many aspects. Feasibility and the accuracy of extracting data from these different simulations heavily depend on the aspects of a given simulation. In this section, we use our data extraction mechanisms to extract data from two different Second Life simulations. These two simulations have different characteristics, and impose different challenges in extracting data.

5.1 *SecondFootball Virtual Football Simulation*

SecondFootball is an interesting simulation in Second Life that facilitates playing virtual football. Integrated with a web interface, the SecondFootball system can be purchased as an out-of-the-box product to be set up on any Second Life region. This simulation contains a football field (containing two goals, and football-related field markings) along with scripted balls. Anyone who wishes to play football can create an account through the web interface and has to acquire a special purpose Head Up Display (HUD) that contains football-specific animations such as tackle, up-kick and down-kick. The SecondFootball simulation provides most of the functionality and features of a real-life football game. Figure 4 shows two players training in the SecondFootball environment.

The SecondFootball simulation is one of the complex and advanced simulations that can be found in Second Life. There are many active football fields set up inside different Second Life regions. Anyone can go into an active football field inside Second Life and play virtual football. It is also possible to join one of the football teams operating inside Second Life, and participate in football matches that are being played for money or just for entertainment.

The SecondFootball simulation is unique when compared with most of the other simulations in Second Life, because it exhibits the concurrent interactions of multiple players and a scripted ball that move around the field most of the time. Most of the other simulations do not exhibit fast movement of a large number of concurrently active avatars, or moving objects. In fact, Varvello et al. (2008) have shown that about 90% of the Second Life residents seldom move around. La and Michiardi (2008) have made the same observation and state that avatars do not travel much. Even if they move, they would be walking most of the time, and use of fast movements such as running and flying is comparatively low.

This unique nature of the SecondFootball simulation introduces several complications when extracting data from a football game. Moreover, being part of a proprietary system, the scripts that run inside the SecondFootball system cannot be accessed to alter them to assist in data extraction. We have tested both data extraction methods we implemented, and the accuracy of the extrapolation-based data extraction method was acceptable only with the avatar positions. For the movements of the ball, it was noted that in some instances the recorded position information was not accurate, especially when the ball is moving above the ground level after being kicked. However, we cannot afford to lose the position information related to the ball, as this is the most critical in a football match. On the other hand, the script-based approach was providing near accurate position information of both players and the ball. Therefore we used the script-based data extraction to record data from SecondFootball matches. As explained in Section 6, the data extraction script did not exhibit any significant impact on the performance of Second Life servers.

Animations play an important role in the SecondFootball simulation. New football-specific simulations such as up-kick, down-kick and tackle are extensively used. These are animations with very short durations. As the LIBOMV client is directly responsible for detecting animations, these short-time animations are not missed out.

Figure 4. The SecondFootball environment

The SecondFootball simulation does not exhibit the use of other scripted objects that players can interact with during the game, or any transactions involving money. Also, the communication acts of players do not play a significant role in this simulation. In fact, in the rules published for official SecondFootball matches, it is stated that players should not use voice or public chat during a football game.

5.2 *The Otago Virtual Hospital (OVH) Simulation*

The Otago Virtual Hospital Simulation (OVH) (Figure 5) resembles the Emergency Department (ED) in a New Zealand hospital. In this simulation, medical students are playing the role of junior doctors/housemen and are trying to solve open ended clinical cases (Blyth et al., 2010). The analysis of the participant behaviour tries to identify the degree to which the participating medical students recorded the salient features in a clinical case. Extracting data from the OVH simulation supports the automation of this analysis.

The OVH simulation is different from the SecondFootball simulation in many aspects. First and foremost, it contains a smaller number of participants. Normally, two doctors are involved in examining one patient. These avatars do not exhibit much mobility, and even if they move, they always walk. As opposed to the SecondFootball simulation, there are no moving objects in the OVH simulation. Therefore we used the extrapolation-based data extraction mechanism to extract data from the OVH simulation.

The OVH simulation has many scripted objects that are of importance to the simulation. Avatars should click on these scripted objects to perform clinical observations such as checking the blood pressure or taking an X-Ray. These scripts communicate with each other, and user actions on the objects are communicated to a chat logger. We made the script running in our LIBOMV bot to listen on this private logger channel, and were able to extract all the information related to avatars clicking on objects. In fact, this information can be considered as the most important, in the OVH simulation.

Unlike the SecondFootball simulation, chat exchanged in the public chat channel is important in this simulation. The participants are encouraged to use the public chat channel when talking to each other rather than using voice chat or instant messaging, because this makes it possible to directly interpret their thinking process. Chat exchanged in the public chat channel was directly captured by the LIBOMV client.

Unlike the SecondFootball simulation, the OVH simulation did not contain any special avatar animations. In fact, short-term animations do not play any important role in the current version of the simulation. The main animations the avatars are playing are sitting, standing, and walking. These are directly captured by the LIBOMV client.

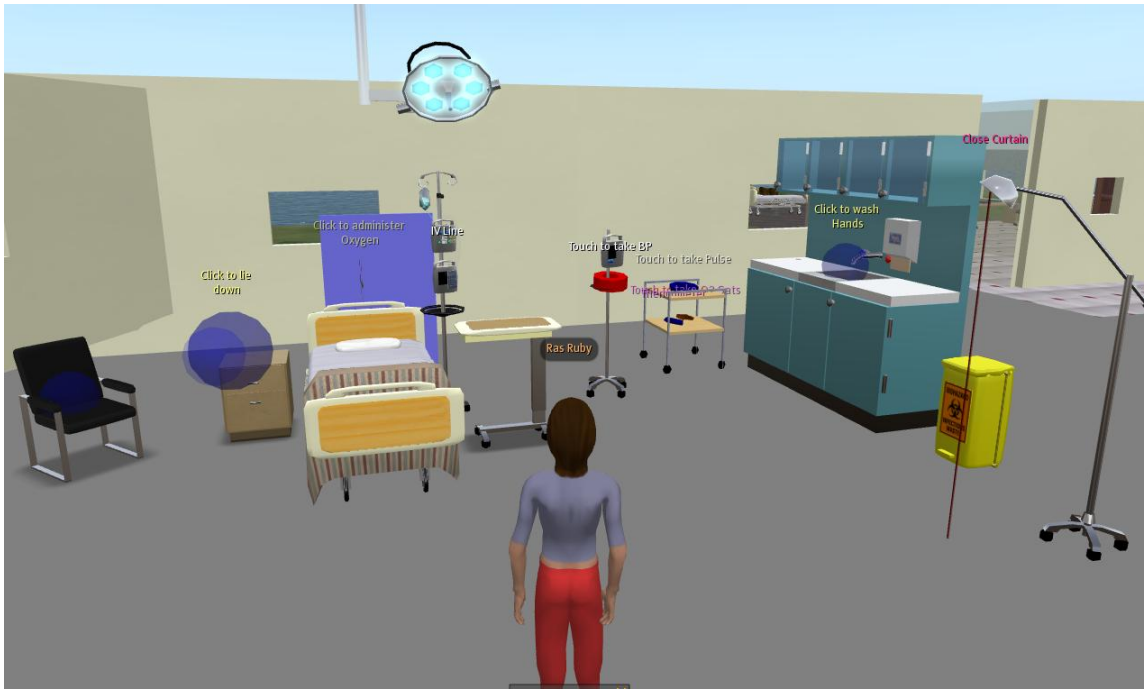

Figure 5. The Otago Virtual Hospital Simulation

6 Performance Evaluation of the Data Extraction Mechanisms

Performance evaluation on the framework is focused on measuring the amount of server lag introduced by the concurrently running scripts that record the movement information of avatars and objects. In order to measure the amount of lag experienced by the server, server time dilation value is used. Time dilation value is on a scale of 0-1: the higher the value, the lower the lag. The Second Life server includes the current time dilation value in every avatar movement update packet sent to the LIBOMV client. Therefore the server time dilation can be directly recorded at the client side.

The test was carried out with the presence of 2 - 6 avatars in the environment. One of these avatars was logged in using the Second Life viewer, and was always immobile. The other avatars were controlled by agents (Ranathunga et al., 2011), and were made to run in a loop where they would run, stop, and run again. We wanted to test the performance of Second Life with the presence of avatars that are showing mobility, as this is a major contribution to lag. On the other hand, mobility means we receive more avatar movement update packets, thus more test values for time dilation. Apart from helping to automate the testing process, the use of agent controlled bots helps to create a uniform test case, as they exhibit pre-defined movements. This is not possible to achieve with human-controlled avatars.

We used the 'one moving avatar' test case as the base case and compared the lag introduced by the following test scenarios with respect to this base case[1].

- When avatars are not wearing any scripts (ns - No Scripts)
- When only one avatar is wearing the script that periodically records movement information of objects and avatars (oss - One Script, using Script-based approach)
- When only one avatar is wearing the script, however the script is only listening on the given private chat channels, without recording movement information (osi - One Script, using extrapolation-based approach)
- When all avatars are wearing scripts that periodically record movement information of objects and avatars (mss - Multiple Scripts, using Script-based approach)

[1]It is not possible to use the test case with no avatars, because at least one moving avatar is needed for the server to send update packets to the client

- When all avatars are wearing scripts, but the script is only listening on the given private chat channels, without recording movement information (msi - Multiple Scripts, using extrapolation-based approach)

Each test was run for 10 minutes, and in the following graphs, the notation $\langle number\_of\_avatars \rangle \langle test\_case \rangle$ is used to refer to individual test scenarios. For example, $2mss$ means that the test case is the fourth one above, and it is executed for 2 concurrent avatars in the simulation. From all of these test results, we removed the data received in the first 1.5 minutes, as the server experiences high lag when trying to connect multiple avatars at the same time.

All these tests are run in the same Second Life environment. More specifically, we chose a SecondFootball field to conduct the evaluation tests. We used the Second Life statistics bar option[1] available in the Second Life viewer to monitor the server statistics. We recorded the server statistics values just before running a test, at the middle of a test, and just after finishing the test. This ensures that the number of avatars, objects and scripts present in the Second Life environment are not changed during a test. It also ensures that all the test scenarios are run under same server conditions. This required having at least one avatar logged into the given Second Life environment using the Second Life viewer as mentioned above. We made sure that this avatar is immobile during the course of a test, in order not to introduce additional lag.

The results of these test runs are shown in Figure 6. The box plot contains the mean, median, and the quartile values of the time dilation values recorded in each of the test cases, with one avatar with no scripts being the base case for comparison. It also shows the outliers presented in each test case. From this result, it can be said that there is no direct relationship between the lag experienced by the server and the number of scripts, or the amount of work carried out by the scripts. The final box represents the test case that ran for one hour, on the fifth test case given above, for five avatars. This is in order to check whether there is a significant decrease in the server performance when scripts are executed for prolonged periods. However, as can be seen in the graph, there is no significant reduction in the server performance, although this box contains a much higher number of outliers, because it was run for a much longer period of time.

The graph in Figure 7 shows the moving average of time dilation values (the subset size is set to 1 second) for the one hour test case. This also confirms this observation, as there is no significant decrease in the time dilation value over time.

One possible explanation for this behaviour could be the new changes that are said to be introduced to Second Life servers as mentioned in Section 3.1. Therefore in order to check whether the server is actually slowing down the script execution amidst of high server lag, we recorded the time taken by the script to execute one round and send the position information to the client. Figure 8 contains the graph that is generated by taking the difference between the times at which two consecutive script messages were received by the client. According to this graph, it can be seen that the time gap between two consecutive messages from the script varies, but it is always concentrated around 500 ms, which is the timer interval we used in the script.

Therefore, even if a script may not slow down an entire Second Life server, it is not possible to depend on the timer interval set in a timer. On the other hand, even though Figure 6 does not show a direct connection between server lag and the amount of avatars and the number of scripts, our Second Life viewer experienced client-side lag when the number of concurrently operating avatars is increased. Moreover, the amount of experience client-side lag was the highest, when all the 5 moving avatars are wearing scripts. Therefore we can say that the initial suggestion we made about selecting the type of data extraction mechanism is still valid, despite the fact that the server lag did not show direct evidence for this.

We also used a third party lag measuring tool[2], in order to verify our test results. When run in a Second Life environment, this tool reports the percentage of server efficiency with respect to the script running times. We executed this lag measuring tool from time to time while conducting our tests, and it did not report any reduction in server performance.

[1] http://wiki.secondlife.com/wiki/Statistics_Bar_Guide

[2] https://marketplace.secondlife.com/p/True-Script-Lag-Checker-Tester-FREE/620049?id=620049&slug=True-Script-Lag-Checker-Tester-FREE

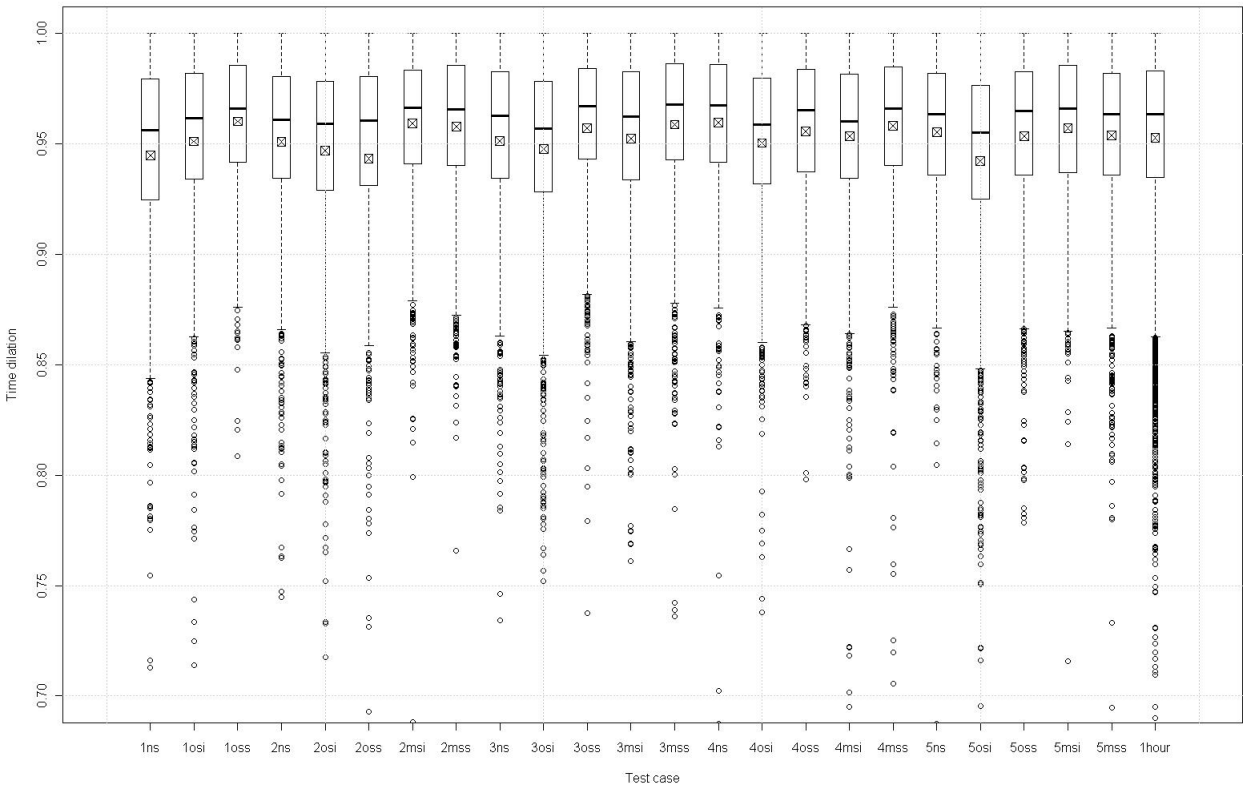

Figure 6. Box plot of time dilation values recorded for the test cases

7 Related Work

Most of the research focused on extracting spatio-temporal data related to dynamic object and avatar behaviour in Second Life has had limited scope, and most importantly, the data extraction frequencies used were comparatively low. Data extracted from Second Life has been used for statistical analysis of Second Life avatars and objects, mainly because such information on Second Life is not publicly available. This also did not require millisecond level accuracies. Moreover some researchers have tried to use Second Life data to understand the special characteristics of the virtual presence of humans and related social norms, because in Second Life they can monitor structured as well as unstructured human behaviour in a multitude of areas. However, most of these analyses have been limited to simple aspects such as interpersonal distance, and avatar contact times.

It can be seen that both data extraction methods, namely LSL scripts and bots created using the LIBOMV API, have been used to extract data from Second Life. However one important aspect to note here is that in most of the research, only one of these techniques has been used for data extraction, and not much emphasis has been given to the animations that are currently being played by the avatars.

Research that attempted to extract spatio-temporal data from Second Life using LSL scripts to identify the social behaviours inside Second Life has not been able to collect considerable amounts of data, mainly due to the limitations of LSL scripts (Friedman et al., 2008; Yee et al., 2007). On the other hand, the use of LIBOMV bots for this aspect has been more successful, and more data points have been captured using this approach (La and Michiardi, 2008). However in this research, the extracted data mainly consists of avatar position information. If animations were considered, it was limited to animations such as walking, or typing, which normally last for longer durations. Consequently, the explored social behaviours were limited to simple aspects such as interpersonal distance, duration of avatars being close to each other, eye gaze, and spatial responses.

LIBOMV has also been successfully used to design a crawler application that collected information on

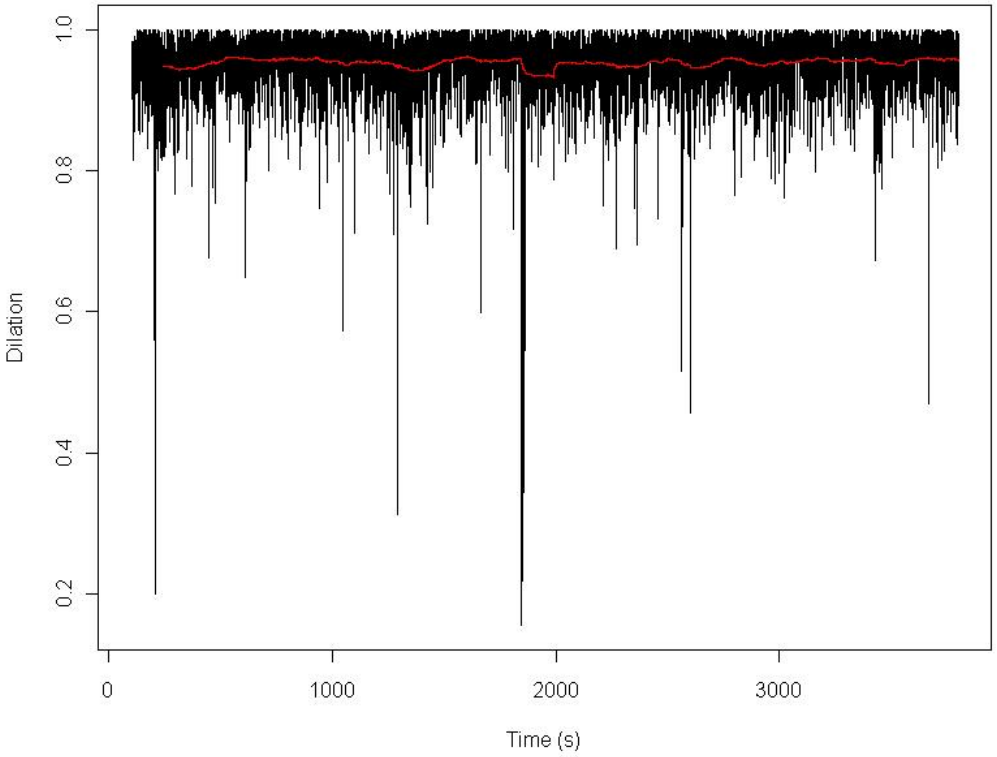

Figure 7. Moving Average of time dilation value for a 1 hour period with 5 mobile avatars

static and interactive user-created content (Eno et al., 2009). Authors have exploited different functionalities of LIBOMV in order to design several crawling subsystems, to gather information of different types of content from a variety of sources and the crawlers have been able to cover a large number of regions (more than 27,000).

A much better comprehensive study of Second Life has been carried out using several LIBOMV-based crawlers that collected data at different granularities (Varvello et al., 2008). This research has collected data about objects, avatars and servers over a period of one month and has been able to collect a large amount of data to produce a reasonable analysis. The researchers have also tried to infer social behaviour of human beings in their virtual appearance and point out that these behaviours show a lot of similarities to social theories applied to real life, in areas such as user mobility and making friends or forming social groups.

Other research has tried to exploit the power of both these data extraction mechanisms in designing a multi-level spatio-temporal data gathering tool (Kappe et al., 2009). More than 200 million records of avatar position data have been collected over a period of time from different regions in order to identify avatar usage patterns and regions of high interest, based on aspects such as the calculated interpersonal distance. The authors of this research also point out the drawbacks of using LSL scripts to extract large amounts of data from Second Life. A comprehensive description of the limitations of LSL in collecting data was also presented by La and Michiardi (2008).

While the aforementioned research has mainly focused on collecting position information of avatars and or objects, Al-Kouz et al. have attempted to collect data of dynamic nature that correspond to chat conversations and analysing them (Al-Kouz et al., 2010).

As mentioned above, the spatio-temporal data collected in most of this research has focused on statistical analysis or deriving simple behaviour patterns, and in most cases, the temporal aspect was insignificant. As a result, the time granularity used was always above 10 seconds, with 10 seconds used by La and Michiardi (2008), 30 seconds by Varvello et al. (2008) and 60 seconds by Kappe et al. (2009). Therefore most of

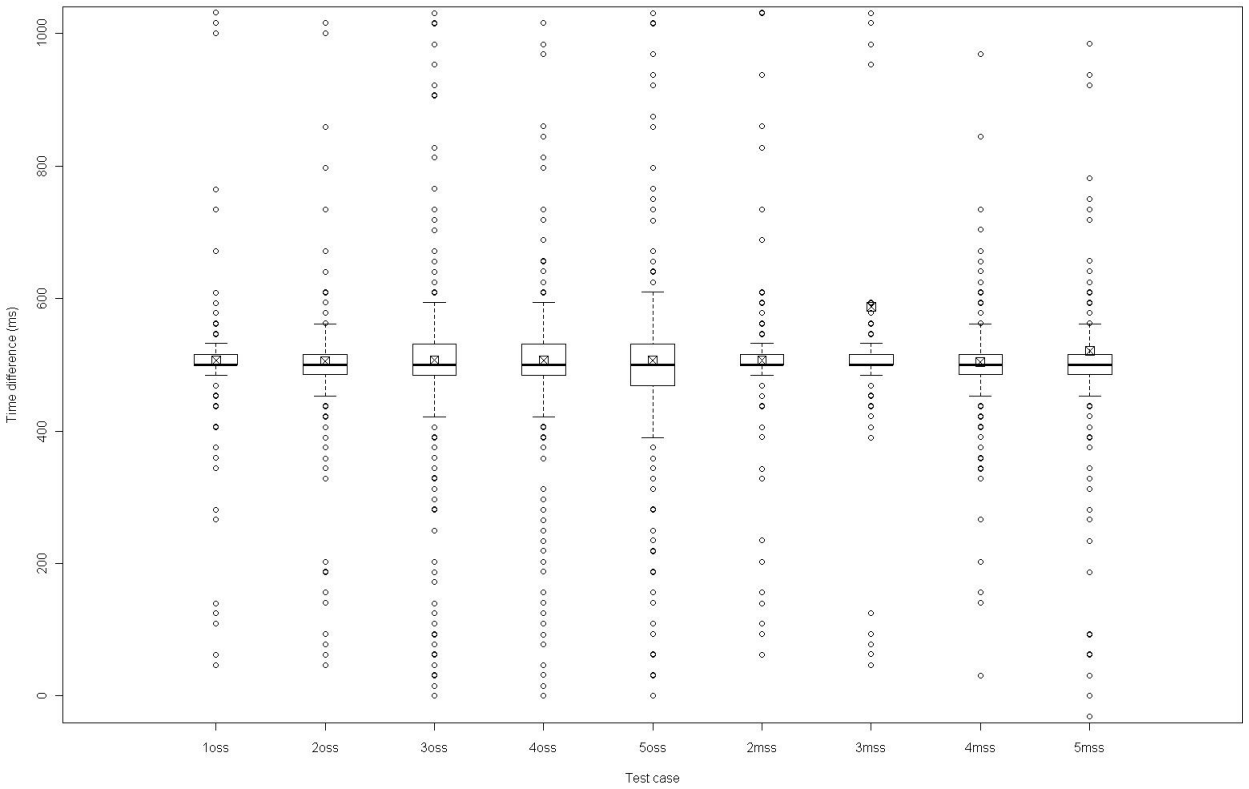

Figure 8. Box plot for the time difference between two consecutive script messages received by the client

these crawlers would not have faced the limitations of LSL and LIBOMV with regard to millisecond-level position accuracy.

A P2P network has been attempted as an alternative solution for Second Life's client-server architecture (Varvello et al., 2009). There the authors rely on the information communicated between the LIBOMV clients rather than waiting for the server to send information. In this scenario, the authors used position extrapolation for avatars in order to produce fluid avatar movements, but they did not consider the existence of objects or object movements that take place in the environment.

A rather different use of Second Life data was reported by Cranefield and Li, where they focused on extracting spatio-temporal data to identify whether members in a Second Life virtual community have fulfilled or violated the rules defined for that community (Cranefield and Li, 2009). However, this research was carried out in a narrow scope using LSL scripts that only dealt with avatar animations.

In the context of using Second Life in conjunction with the area of Artificial Intelligence, researchers have already identified the complexity of accurately extracting large amounts of spatio-temporal data an agent is faced with (Burden, 2009). The available research on modeling AI related concepts inside Second Life seems to be still at its inception stage, and work based on the use of LSL scripts has been greatly restricted by the limitations of that approach (Veksler, 2009). In most work, the focus has been to change the behaviour of the software controlled avatars (Jan et al., 2009; Ullrich et al., 2008), rather than on creating reactive agents that respond to what they perceive in the environment around them.

One exception is the work of Bogdanovych and colleagues. They have extensively used LIBOMV client functionality to generate custom virtual worlds where software-controlled agents and human-controlled agents interact in three dimensional representations of "electronic institutions" (Bogdanovych et al., 2009). The infrastructure for their work described in a recent paper (Bogdanovych et al., 2010) includes a number of useful libraries for connecting agents to Second Life. However, our work has a different focus from the work presented by these papers. We have mainly focused on identifying the limitations of the existing data extraction mechanisms from Second Life, and to provide better data extraction mechanisms to extract

data in high frequency and high accuracy. Although a similar technique may have used by that work, it is not presented in those papers.

8 Conclusion

This paper addressed the problem of extracting spatio-temporal data from Second Life with high accuracy and high frequency. This is useful in many applications implemented inside Second Life, such as modeling Artificial Intelligence theories.

We have provided a detailed analysis of the existing mechanisms to extract spatio-temporal data from Second Life along with their drawbacks. We also provided two new mechanisms that can be used to accurately extract spatio-temporal data from most, if not all of the Second Life simulations.

We have conducted a performance evaluation to measure the lag introduced by these data extraction mechanisms, and it could be seen that the lag present in the Second Life server does not have a direct relationship with the data extracting scripts. Therefore depending on the needs of a given Second Life simulation or environment, it is possible to use one of these data extraction mechanisms and extract data with high accuracy and high frequency. This fact is also supported by the two use cases we presented. Our data extraction mechanisms were successful in extracting all the important data corresponding to interactions taking pace in these simulations.

References

Al-Kouz, A., E. W. D. Luca, J. Clausen, and S. Albayrak (2010). The Smart-TSH-Finder: Crawling and analyzing tempo-patial hotspots in Second Life. In M. Atzmller, D. Benz, A. Hotho, and G. Stumme (Eds.), *Proceedings of LWA2010 - Workshop-Woche: Lernen, Wissen & Adaptivitaet*.

Ann Myers Medical Center (2010). The Ann Myers Medical Center Simulation in Second Life. `http://maps.secondlife.com/secondlife/Hospital/169/194/24`.

Blyth, P., S. K. Loke, and J. Swan (2010). Otago virtual hospital: medical students learning to notice clinically salient features. In *Curriculum, technology & transformation for an unknown future. Proceedings ascilite Sydney 2010*, pp. 108–112.

Bogdanovych, A., J. A. Rodriguez-Aguilar, S. Simoff, and A. Cohen (2010). Authentic interactive reenactment of cultural heritage with 3D virtual worlds and artificial intelligence. *Applied Artificial Intelligence 24*(6), 617–647.

Bogdanovych, A., S. Simoff, and M. Esteva (2009). Virtual institutions prototype. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pp. 1373–1374. IFAAMAS.

Borner, K., W. Hazlewood, and S.-M. Lin (2002). Visualizing the spatial and temporal distribution of user interaction data collected in three-dimensional virtual worlds. In *Information Visualisation, 2002. Proceedings. Sixth International Conference on*, pp. 25 – 31.

Burden, D. J. H. (2009). Deploying Embodied AI into Virtual Worlds. *KnowledgeBased Systems 22*, 540–544.

Cox, R. and P. Crowther (2009). A Review of Linden Scripting Language and its role in Second Life. In M. Purvis and B. Savarimuthu (Eds.), *Computer-Mediated Social Networking*, Volume 5322 of *Lecture Notes in Computer Science*, pp. 35–47. Springer Berlin / Heidelberg.

Cranefield, S. and G. Li (2009). Monitoring Social Expectations in Second Life. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pp. 1303–1304. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC.

Diener, S., J. Windsor, and D. Bodily (2009). Design and development of clinical simulations in Second Life. In *Proceedings of the EDUCAUSE Australasia Conference*.

Eno, J., S. Gauch, and C. Thompson (2009). Intelligent Crawling in Virtual Worlds. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, pp. 555–558. IEEE Computer Society Washington, DC, USA.

Fernandes, S., R. Antonello, J. Moreira, D. Sadok, and C. Kamienski (2007). Traffic Analysis Beyond This World: the Case of Second Life. In *17th International Workshop on Network and Operating Systems Support for Digital Audio & Video*, pp. 555–558.

Friedman, D., A. Steed, and M. Slater (2008). Spatial Social Behavior in Second Life. In *Proceedings of the 7th international conference on Intelligent Virtual Agents*, pp. 252–263. Springer-Verlag Berlin, Heidelberg.

Gregory, S. and B. Tynan (2009). Introducing Jass Easterman: My Second Life learning space. In *Proceedings of ascilite Auckland 2009*.

Jan, D., A. Roque, A. Leuski, J. Morie, and D. Traum (2009). A Virtual Tour Guide for Virtual Worlds. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, pp. 372–378. Springer-Verlag Berlin, Heidelberg.

Kappe, F., B. Zaka, and M. Steurer (2009). Automatically Detecting Points of Interest and Social Networks from Tracking Positions of Avatars in a Virtual World. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, pp. 89–94. IEEE Computer Society Washington, DC, USA.

La, C.-A. and P. Michiardi (2008). Characterizing User Mobility in Second Life. In *Proceedings of the first workshop on Online social networks*, pp. 79–84. ACM New York, USA.

Linden Lab (2010). Server Architecture. Article on Second Life wiki. `http://wiki.secondlife.com/wiki/Server_architecture`.

Luderschmidt, J. (2010). Technological Foundations Of Second Life. `http://static.twoday.net/lagerkoller/files/TechnolgySL.pdf`.

Nash, S. S. (2009). Libraries in Second Life: New Approaches to Education, Information Sharing, Learning Object Implementation, User Interactions and Collaborations. *Journal of Systemics, Cybernetics and Informatics 7*, 25–28.

OpenMetaverse Organization (2010). libopenmetaverse developer wiki. `http://lib.openmetaverse.org/wiki/Main_Page`.

OpenSimulator (2010). OpenSimulator Home Page. `http://opensimulator.org/wiki/Main_Page`.

Ranathunga, S., S. Cranefield, and M. Purvis (2011). Interfacing a Cognitive Agent Platform with a Virtual World: a Case Study using Second Life. In *International Workshop on the uses of Agents for Education, Games and Simulations (AEGS 2011)*.

Rodriguez, I., A. Puig, M. Esteva, C. Sierra, A. Bogdanovych, and S. Simoff (2008). Intelligent objects to facilitate human participation in virtual institutions. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 196–199. IEEE.

Rogers, L. (2009). Simulating clinical experience: Exploring Second Life as a learning tool for nurse education. In *Proceedings of ascilite Auckland 2009*.

Ryan, M. (2008). 16 ways to use Second Life in your classroom: pedagogical approaches and virtual assignments. In *Proceeding of the Second Life Education conference*.

Shi, W., G. Lee, J. Hinchley, J.-P. Corriveau, B. Kapralos, and A. Hogue (2010). Using a virtual learning environment with highly interactive elements in Second Life to engage millennial students. In *Proceedings of the 2010 International Conference on e-Education, e-Business, e-Management and e-Learning*, pp. 255–259. IEEE Computer Society Washington, DC, USA.

Storey, V. A. and A. A. Wolf (2010). Utilizing the Platform of Second Life to Teach Future Educators. *International Journal of Technology in Teaching and Learning 6(1)*, 58–70.

Taylor, D. I., R. Winter, M. Chan, R. Davies, J. Kinross, and A. Darzi (2009). Virtual Patient and Medical Device Simulation In Second Life: the Use Of Immersive Virtual Worlds For Learning and Patient Safety. In *Proceedings of the MedBiquitous Annual Conference*.

Thumann, M. (2008). Hacking Second Life. `http://www.blackhat.com/presentations/bh-europe-08/Thumann/Whitepaper/b%h-eu-08-thumann-WP.pdf`.

Ullrich, S., K. Bruegmann, H. Prendinger, and M. Ishizuka (2008). Extending MPML3D to Second Life. In *Proceedings of the 8th international conference on Intelligent Virtual Agents*, pp. 281–288. Springer-Verlag Berlin, Heidelberg.

Varvello, M., S. Ferrari, E. Biersack, and C. Diot (2009). Distributed Avatar Management for Second Life.

In *Proceedings of the 8th Annual Workshop on Network and Systems Support for Games*.

Varvello, M., F. Picconi, C. Diot, and E. Biersack (2008). Is There Life in Second Life? In *Proceedings of the 2008 ACM CoNEXT Conference*, pp. 1:1–1:12. ACM New York, USA.

Veksler, V. D. (2009). Second Life as a Simulation Environment: Rich, high-fidelity world, minus the hassles. In *Proceedings of the 9th International Conference of Cognitive Modeling*.

Vstex Company (2010). Secondfootball Home Page. `http://www.secondfootball.com`.

Weitnauer, E., N. M. Thomas, F. Rabe, and S. Kopp (2008). Intelligent Agents Living in Social Virtual Environments — Bringing Max into Second Life. In *Proceedings of the 8th international conference on Intelligent Virtual Agents*, pp. 552–553. Springer-Verlag Berlin, Heidelberg.

Yee, N., J. N. Bailenson, M. Urbanek, F. Chang, and D. Merget (2007). The Unbearable Likeness of Being Digital: The Persistence of Nonverbal Social Norms in Online Virtual Environments. *CyberPsychology and Behavior 10*, 115–121.