Distributed data exchange with Leap Motion

Mirko Pani[1] and Fabio Poiesi[2]

[1] Department of Information Engineering and Computer Science - Università degli Studi di Trento, Italy - mirko.pani@studenti.unitn.it,
[2] Technologies of Vision - Fondazione Bruno Kessler - Trento, Italy - poiesi@fbk.eu

Abstract. Collaborative virtual environments can connect people in social virtual spaces even when they are geographically distant from each other. Hand interactions are fundamental to enable natural collaboration and immersive experiences as they are a visually intuitive means of communication. However, scalability is challenging as numerous participants typically produce a large volume of visualisation data that may overload a single node if the management is centralised. In this paper we propose a transmission strategy where the high-throughput visualisation data (e.g. hand joints) is exchanged amongst participants in a distributed fashion. We use a level-of-detail strategy to further reduce the network traffic accounting for spatial distances amongst participants in the virtual space. We design an experiment where we analyse the network traffic in a virtual environment with up to seven participants whose hands are tracked using Leap Motion. We show that the proposed method can effectively reduce the network traffic of visualisation data when compared to a centralised approach.

Keywords: Collaborative virtual environments, distributed communication, level of detail, hand tracking, Leap Motion, Unity3D.

1 Introduction

Virtual Reality (VR) experiences are evolving from being limited to a single participant to being a multi-user collaborative reality [16]. Applications, such as education [15], manufacturing [14], engineering and construction [18], medics [19] and video gaming [8], have shown great interest in collaborative VR. Effective and immersive collaborations are guaranteed if users can interact in VR using their hands in an intuitive manner. Commercially available products, such as HTC Vive and Oculus Rift, provide remote controllers to permit hand-based interactions. However, hand visualisation in VR does not yet appear to be natural as these remote controllers do not track hand joints [10, 21]. To achieve total immersion and to open up new opportunities in VR, technology developments are shifting towards in-air hand tracking using devices such as Leap Motion [4]. Fig. 1 shows an example of collaborative virtual environment (CVE) where participants interact using hand gestures. Unfortunately, there are still several challenges that prevent an uncompromising experience using hand-free interactions in VR,

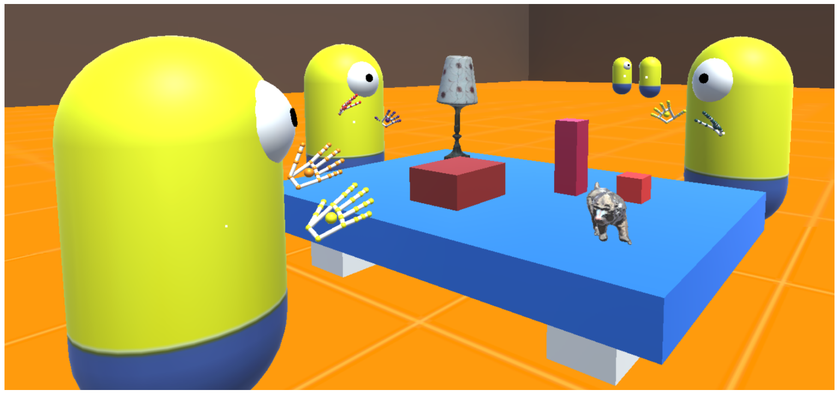


Fig. 1. Collaborative virtual environment where two groups of participants are interacting via hand gestures. Visualisation data exchanged amongst clients occurs peer-to-peer and using a strategy based on levels of detail. Clients that are close to each other exchange data at a higher level of detail. Because hand gestures of distant clients (e.g. those in the background) are barely visible, their visualisation data will not be transmitted to the clients near to the camera. This strategy allows a more effective communication in populated virtual environments as we can noticeably reduce the data transmitted over the network and managed by the host of the VR session.

such as transmission latency, high computational demand and hand tracking robustness [17, 22].

Interactions in CVEs are typically managed at network level via an *authoritative server* [22, 24]. An example is Unity3D that provides the High Level API to design CVEs based on authoritative mechanism [2]. The server has the authority to update the states of the objects in the environment, namely *mutable objects* (e.g. position of a player, colour of an object), upon the requests received from the clients. We refer to the term *client* to underline the role of a participant within a network that communicates either with the server or directly with another participant. The authoritative mechanism provides data consistency on each client and the advantage of being robust against malicious players' behaviours (e.g. game cheating). However, CVEs that are designed using an authoritative server are hardly scalable. The larger the number of users that join the same CVE, the larger the delay of state updates and the more likely visualisation lags [11]. A solution to promote scalability is via peer-to-peer communications (i.e. the communication between two clients does not involve a server), at the expenses of more sophisticated strategies to handle state updates and prevent cheats [24]. Leap Motion produces high-frame rate hand visualisation data that leads to high throughput when the visualisation data is exchanged over a network populated by several clients [6, 22]. To mitigate the problem of high throughput, one can employ different data transmission strategies based on peer-to-peer or hybrid communications. Hybrid approaches use server and peer-to-peer communications interchangeably, and are typically employed in massively multiplayer online games (MMOGs) [13, 20]. One can also use an area-of-interest (AOI) based approach where a VR space is divided into zones and data is transmitted peer-to-peer amongst users that are located within the same zone [12].

In this paper we propose a scalable mechanism to exchange visualisation data amongst clients in collaborative virtual environments[3]. Our solution uses the ordinary authoritative mechanism to update the states of mutable objects, but a distributed strategy to handle high-throughput visualisation data exchanges. Differently from AOI-based approaches where the data transmission between clients is either active or inactive without accounting for levels of detail, we use clients' relative distances in the VR space to dynamically variate the resolution of transmitted visualisation data. We will show that our distributed approach effectively reduces the network traffic amongst clients when exchanging the visualisation data as opposed to a centralised approach. We evaluated our approach by implementing it on Google Cardboard devices (or simply Cardboard) [1] and by using Leap Motion [4] for hand tracking. We used Unity3D cross-platform game engine as development environment [9].

2 Proposed approach

A set of participants that interact in the same virtual environment exchange visualisation data under reciprocal requests of levels of detail that depend on their spatial distance in the VR space. In the next sections we will describe the proposed communication mechanism.

2.1 Communication mechanisms

The communication amongst participants is decoupled to handle data differently based on its type. The authoritative server handles client enrollments, while peer-to-peer communications handle high-throughput visualisation data produced by Leap Motion.

A user that initiates a VR session can be both the host and a client. The host is in charge of (i) updating the states of the mutable objects that require synchronisation amongst clients and (ii) managing client enrollments/disenrollments. When a client enrolls to a VR session, the host broadcasts its IP and port addresses (enclosed in a broadcast message) to the other connected clients. When a client disenrolls, the host informs the connected clients that a client has left the virtual environment. These broadcast messages are used by each client to store and maintain the network addresses of the connected clients updated on their internal *Sync Table*. Each client has the same copy of the Sync Table and can use this global knowledge to establish peer-to-peer communications to exchange visualisation data with the other clients. This is an important element because when devices like Cardboard are used for CVEs, they have limited computational capability and battery duration. With peer-to-peer communications high-throughput data does not go through the host thus reducing its computational load.

[3] Project webpage: `tev.fbk.eu/distributedLeapMotion`.

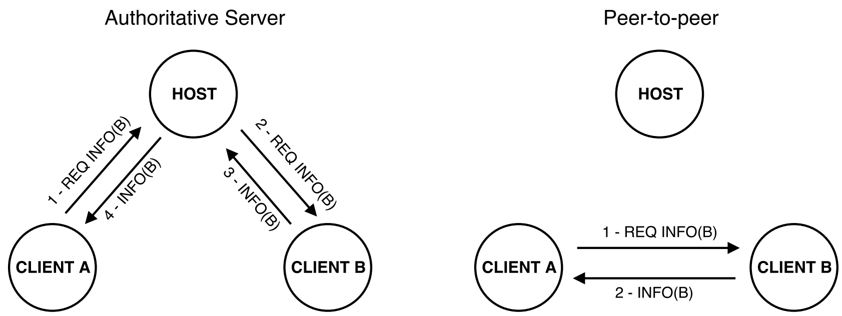


Fig. 2. Difference between authoritative and peer-to-peer mechanisms. Requests (REQ) and information (INFO) exchanged over the network are halved in the case of peer-to-peer communications.

Fig. 2 shows an example where the requests and the information exchanged over the network are halved in the case of peer-to-peer communications.

2.2 Distance-based level of detail

Data that represents hand gestures are typically sampled at a high frame rate to visualise natural movements. However, when hand gestures in VR are seen from far, details might be unnoticeable. Typically, levels of detail are used in these situations to define multiple representations of a model with decreasing resolution in order to reduce the rendering cost for distant or less important objects [23].

We design a request-based mechanism to handle levels of detail dynamically. Clients that are involved in an interaction, reciprocally request their desired level of detail to other clients based on their spatial distance in the VR space: the closer the two clients, the higher the details of the visualisation data they request. The queried clients that accept this request will transmit their visualisation data at the requested levels of detail. A system based on requests can also provide the possibility to extend this distance-based criterion to additional criteria, for example based on network or rendering capacity. In this work we analyse the case of distance only.

Fig. 3 shows five clients that are connected to the same VR space, one is the host/client and the others are clients. The top part of the figure shows an example with four levels of detail:

L0 defines the maximum level of detail where no approximations of the hand joints are applied. This could be the case where two or more clients interact "face to face" and highly-detailed joint movements are necessary to visualise gestures accurately;

L1 defines an intermediate level of detail where a subset of joints are transmitted, while ensuring that a well approximated hand motion can still be perceived. This could be the case where the client requesting the visualisation data is not interacting directly with the queried client, but their distance is such that their

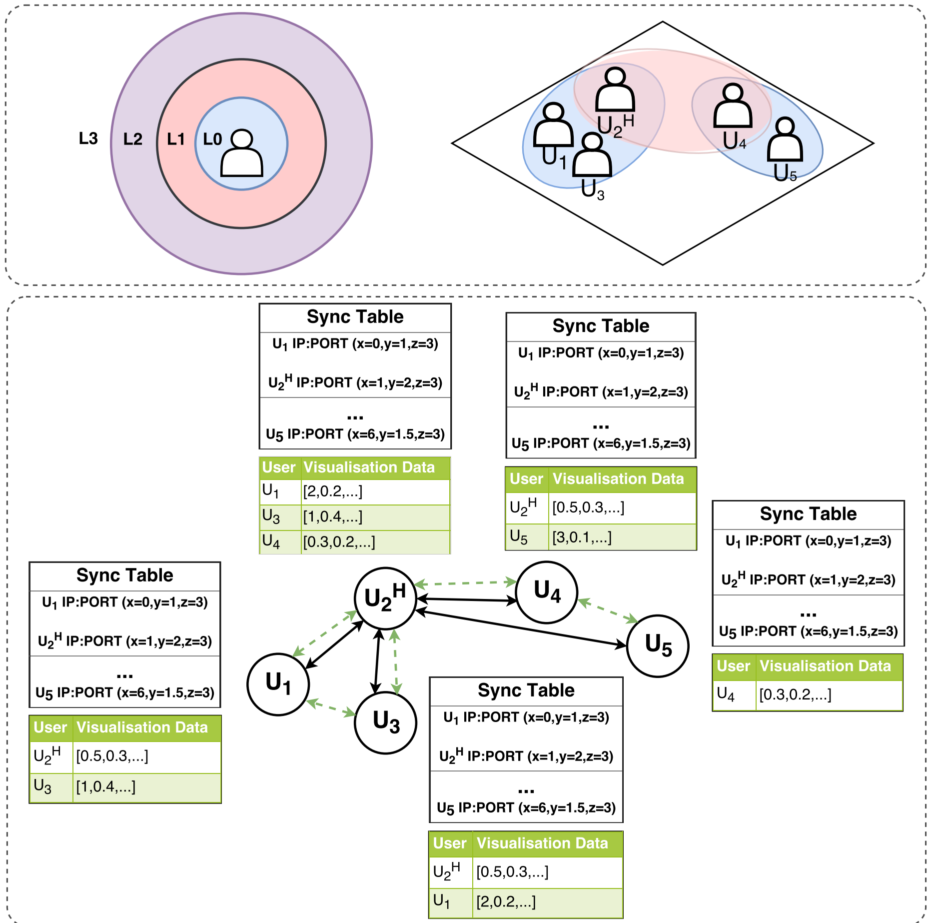


Fig. 3. Five clients are connected to the same VR space, one is the host/client and the others are the clients. On the top, four levels of detail are defined: L0 defines the maximum level of detail, L1 and L2 define intermediate levels of detail, and L3 defines no data transmission. The lower part of the figure illustrates the connections (i.e. black arrows) the server uses to broadcast the information about enrollments/disenrollments that are then used to update the Sync Table on each client. Each client uses the information included in this Sync Table to establish peer-to-peer connections with other clients. Green arrows show the peer-to-peer connections between clients to exchange high-throughput visualisation data.

hand movements are still visible;

L2 defines another intermediate level of detail where a minimal subset of joints is transmitted to show basic hand movements. This could be the case where the client requesting the visualisation data is far from the queried client and its hands are barely visible;

L3 defines no data transmission as clients are out of line of sight.

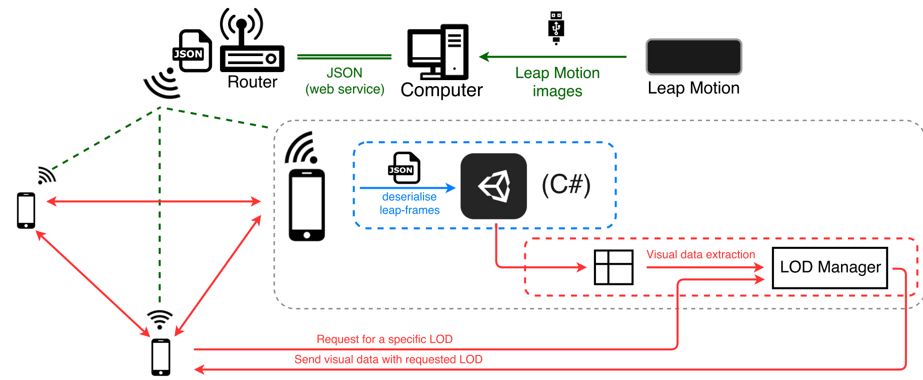


Fig. 4. Illustration of the proposed distributed mechanism. Each client is connected to the Leap Motion's web service that is hosted on a computer and leap-frames are transmitted using JSON format. The client deserialises the JSON-formatted leap-frames using C# classes of Unity3D. Requests are dealt by the level of detail (LOD) manager. The clients send visualisation data at the requested level of detail. The figure shows three conceptual blocks: green for the local Leap Motion connection, blue for the local processing of the hand visualisation and red for remote hand visualisation.

The connections (i.e. black arrows) the server uses to broadcast the information about enrollments/disenrollments are also used to update the Sync Table on each client. Each client uses the information included in this Sync Table to establish peer-to-peer connections with other clients. Green arrows show the peer-to-peer connections between clients to exchange high-throughput visualisation data.

3 Implementation

We have implemented our proposed distributed approach in a scenario where each client can experience VR using Cardboard and interact with Leap Motion. We implemented the client-host requests using the Unity3D's native authoritative mechanism provided through the High Level API (HLAPI) [2]. Peer-to-peer visualisation data exchanges are managed by the proposed distributed request-based mechanism. We use the UDP protocol for timely delivery of data. Each client periodically listens for incoming UDP packets over the network socket. Fig. 4 illustrates an overview of the implementation.

3.1 Message handling

To enable effective communications, we define a protocol that specifies the structure of the messages exchanged between the host and clients to update the Sync Table, and between clients to send requests and transmit visualisation data at different levels of detail. Table 1 lists the messages defined by our protocol.

The protocol functions in the following way. A client joins a VR session through HLAPI. The host assigns a unique identifier (*NetID*) to this client and

Table 1. Messages defined in the communication protocol.

Header	Syntax	Description
AIP	[AIP]:<ip>,<port>,<netID>	Message the host sends to inform clients to add a new entry in the Sync Table
RIP	[RIP]:<netID>	Message the host sends to inform clients to remove the entry of the disconnected client from the Sync Table
RQ<n>	[RQ<n>]:<netID>	Message a client sends to another client to request visualisation data at a specific level of detail
AK<n>	[AK<n>]:<HandsParam>,<netID>	Message a queried client sends to a requesting client that contains the visualisation data at a specific level of detail in the payload; <HandsParam> integer value that defines the type of hand contained in the payload (0: left, 1: right, 2: both)

broadcasts a message informing the connected clients that an enrollment has occurred. Each message is composed of a header and a payload. The header uses the first three bytes of the message to define the type of data carried by the payload. When a client connects, the host sends an 'AIP' message to inform clients that a new entry in the Sync Table should be added. When a client disconnects, the host sends a 'RIP' message to inform the clients that a client should be removed from the Sync Table.

The clients are responsible for requesting the visualisation data to other clients. For example, if client A wants visualisation data of client B, A will send a request message 'RQ' to B requesting the visualisation data at the desired level of details (i.e. based on their distance). When B receives the request, B encapsulates the visualisation data in a message 'AK' and sends it to A. This procedure is repeated periodically at a pre-defined frequency to show hand movements. Algorithm 1 details the overall message handing procedure.

3.2 Leap Motion (visualisation) data

Leap Motion does not yet support its direct connection to Cardboard [22]. Therefore we connect Leap Motion to a computer and use the native Leap Motion web service to transmit the hand tracking data to Cardboard.

On the computer, Leap Motion software periodically sends hand interaction elements, such as position and orientation of joints, the list of tool and pointable objects detected in the current frame, etc., through *leap-frames* that are encoded with JSON format [3]. Leap-frame transmission rate is typically greater than 60 frames per second [6]. When the leap-frames are received by Cardboard, all the hand interaction elements are used by Unity3D on each client to enable the interactions with mutable objects and to trigger the object physics. These interactions are centrally managed by the authoritative server.

Because the appearance of a hand can be represented only by the position of its joints, we use a subset of these elements to visualise hands on Cardboard. This subset of elements can then be further reduced using levels of detail and in our case we use four levels of detail (Sec. 2.2): L0, L1, L2 and L3. L0 defines the highest level of detail, L1 and L2 are intermediate levels, and L3 defines no data

```
Data:
t_1 = Time.Now; t_2 = Time.Now;
Δ_s = 0.2s; Δ_r = 0.3s;
m = message; c = client;
LOD = level of detail;
Handle(m) = message handling as defined in Tab. 1;
while true do
    if (t_1 - Time.Now) > Δ_s then
        handData = getHandsData();
        t_1 = Time.Now;
    end
    if (t_2 - Time.Now) > Δ_r then
        for c in ConnectedClients do
            LOD = calculateDistance(localClient, c);
            requestHandData(c, LOD);
        end
        t_2 = Time.Now;
    end
    if m == received then
        Handle(m);
    end
end
```

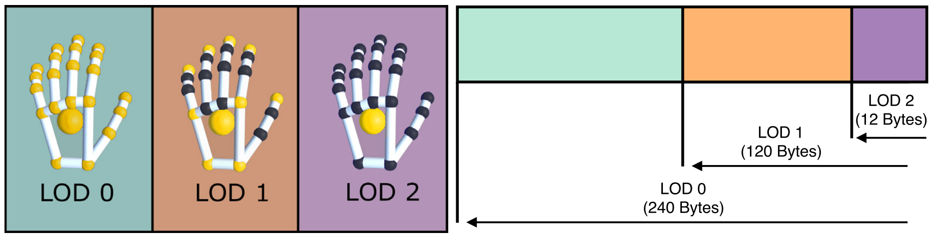
Algorithm 1: Message handling procedure.

transmission. We use the *CapsuleHand* format to represent Leap Motion hands, which models the hand structure using a set of capsules (i.e. joints) connected by cylinders [5]. Note that our communication approach is independent from this computer-Cardboard connection because hand joints' data is handled at network level as it were processed on Cardboard and the JSON-encoded leap-frames are reconstructed in the same way as they were processed natively.

Fig. 5 illustrates how these capsules are defined at different levels of detail and how visualisation data are organised to be efficiently accessed. One hand is represented at L0 with 240 bytes, at L1 with 120 bytes and at L2 with 12 bytes.

4 Experiments

We evaluated the proposed method by performing two experiments. Firstly, we measured the number of bytes transmitted during a simulated collaborative VR session. Secondly, we used multiple smartphones as Cardboard devices and measured their transmission latency within a local wireless network. In all the experiments, we compared the performance of our distributed approach against a centralised (i.e. authoritative) approach, and by enabling and disabling the level of detail strategy, namely LOD and NO LOD, respectively. All experiments involved up to seven clients connected to the same VR space. One of the clients was the host. In order to stress the system, we streamed sequences of leap-frames corresponding to two tracked hands, i.e. 480 bytes per leap-frame.



(a) (b)

Fig. 5. Example of three levels of detail with associated bytes count for the Leap Motion hand. Yellow joints are encoded and transmitted, black joints are ignored.

4.1 Network traffic

To measure the network traffic, we designed a simulation where the seven clients were positioned in the VR space randomly. The VR space was 20×15 Unity3D units. The level of detail regions had radii of 4, 8, 12 and >12 Unity3D units to define L0, L1, L2 and L3, respectively. Because the VR space was limited and because the duration of the experiment was 150 seconds, we could simulate several combinations of relative distances between clients and hence trigger transmissions with different levels of detail.

Fig. 6 shows the trends of the total number of bytes transmitted as a function of the duration of the experiment. When we account for the level of detail, we can effectively reduce the network traffic using the proposed distributed approach as opposed to the centralised approach. Interestingly, when the level of detail is not used, i.e. the visualisation data are exchanged amongst clients regardless their distance, the centralised approach generates less network traffic than the distributed one. This happens because, although the centralised approach is used, the level of detail strategy led to an effective reduction of bytes when clients are distant from each other. For example, if client A is located in L3 with respect to B, B will request visualisation data using the NO LOD distributed strategy, whereas with a LOD centralised strategy no data will be requested.

The oscillations that are visible in NO LOD strategies are due situations where clients are distant from each other in the VR space. In this case data transmissions are fewer or even absent sometimes.

4.2 Transmission latency

In this experiment we evaluated the latency of the data transmission. We quantified the latency as the delay measured by a client to send the request to another client and to receive the data. We tested scenarios with three, five and seven clients connected via wireless. We used Google Nexus 5X, Huawei Honor 8 and Nvidia Shield for the scenario with three clients, these three devices plus two Huawei P10+ for the scenario with five clients and these five devices plus two computers for the scenario with seven devices. Nvidia Shield was used as the host in all scenarios.

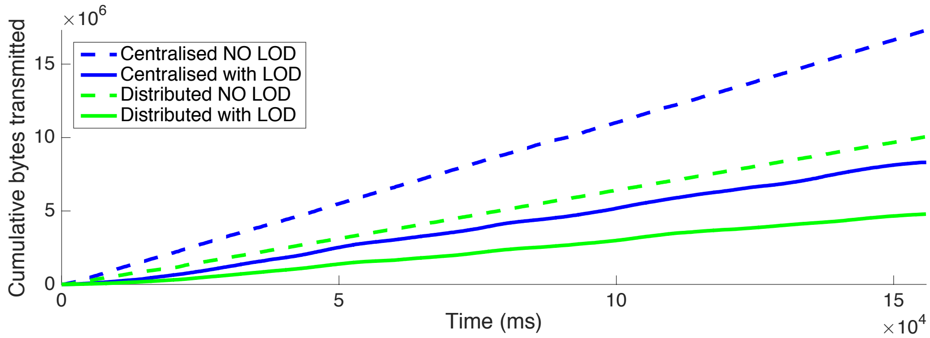


Fig. 6. Cumulative network traffic measured in bytes in the case of seven clients. Centralised and distributed approaches that consider level of detail (with LOD) and that do not consider it (NO LOD).

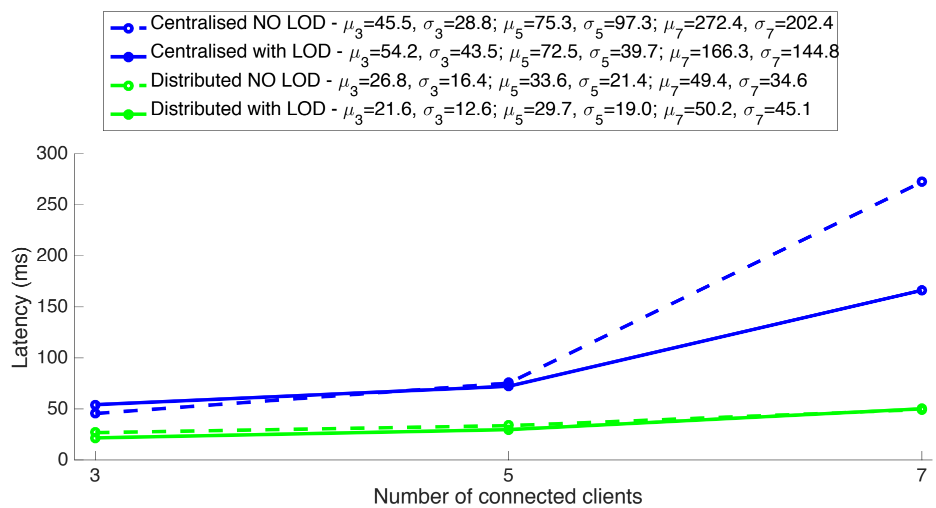


Fig. 7. Average latency as a function of the number of clients corresponding to centralised and distributed approaches that consider level of detail (with LOD) and that do not consider it (NO LOD). The network latency is the average of latency measurements collected over a period of three minutes.

Fig. 7 shows the average latency as a function of the number of clients corresponding to each scenario. The exact values of average and standard deviation are reported in the legend of the graph. The average latency measured when visualisation data are exchanged with the distributed strategy is smaller than the latency measured with the centralised strategy. The use of levels of detail reduces in general the latency. We can observe that the variance is in general fairly large and this is due to background threads of the devices that delay the processing of the received packets.

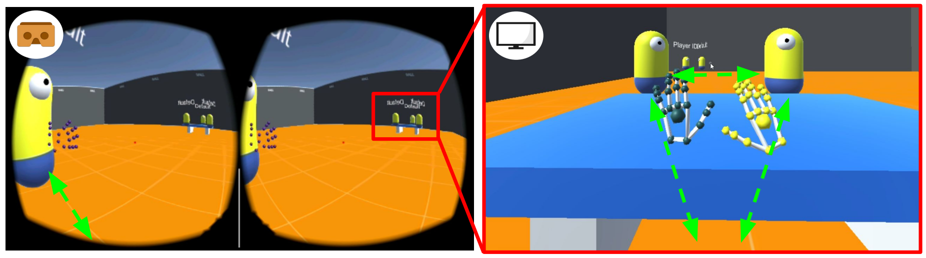


Fig. 8. Scenario where five participants are within the same VR space and visualisation data are exchanged through the proposed distributed mechanism using levels of detail. The green arrows illustrate whom the visualisation data is shared with. The video of this experiment can be found on the project webpage.

4.3 Qualitative analysis

Fig. 8 shows a scenario where five clients are within the same VR space and visualisation data are exchanged using the proposed distributed mechanism with levels of detail. The left-hand figure is a screenshot taken from a Cardboard device, while the right-hand figure is a screenshot taken from a desktop computer. The distance between clients is such that they create two groups of data exchange: one group of two clients (left-hand figure) and one group of three clients (right-hand figure). In each group there is one client waving its hands. With this configuration L0 (max resolution) is applied between clients of each group and L3 (no data) is applied between clients of different groups. The green arrows illustrate whom the visualisation data is shared with. From the Cardboard's view we can see the group of three clients in the background and we can observe that their hands' movements would not be visible even if their visualisation data were transmitted to the clients of the other group.

5 Conclusions

We have presented a distributed mechanism to exchange Leap Motion (visualisation) data in collaborative virtual environments. We developed a solution where the distance between users is used to select the level of detail at which the visualisation data are exchanged. We created a protocol to share messages that include the reciprocal knowledge about the position of clients and the requests of the level of detail. These messages are exchanged via UDP for timely delivery. We have evaluated our approach in both simulated and real-world scenarios using Google Cardboard, and showed that the proposed method outperformed the centralised approach provided in Unity3D. Although Google Cardboard was used in our tests, the proposed distributed mechanism can also be employed on other VR devices.

Future research will involve the reduction of request messages between clients by implementing requests with durations. Instead of sending one request for each leap-frame we will explore how to send a request for a group of leap-frames. Then we will explore how to utilise our strategy to transmit other visualisation data, such as Kinect point cloud [7]. In this way clients will be able to project their scan in the VR space and make use of the distributed mechanism with levels of detail to reduce the number of points transmitted over the network.

Acknowledgments

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 687757.

References

1. Google Cardboard. `vr.google.com/cardboard`, accessed: May 2018
2. High Level API. `docs.unity3d.com/Manual/UNetUsingHLAPI.html`, accessed: May 2018
3. Leap-Frame. `developer.leapmotion.com/documentation/java/devguide/Leap_Frames.html`, accessed: May 2018
4. Leap Motion. `leapmotion.com`, accessed: May 2018
5. Leap Motion hand assets. `developer.leapmotion.com/documentation/csharp/unity/Unity_Hand_Assets.html`, accessed: May 2018
6. Leap Motion latency. `blog.leapmotion.com/understanding-latency-part-1`, accessed: May 2018
7. Live Scan 3D. `github.com/MarekKowalski/LiveScan3D`, accessed: May 2018
8. Second Life. `secondlife.com`, accessed: May 2018
9. Unity 3D. `unity3d.com`, accessed: May 2018
10. VibeHub. `youtu.be/azUXUr6rWSc`, accessed: May 2018
11. Abdulazeez, S., Rhalibi, A., Al-Jumeily, D.: Evaluation of scalability and communication in MMOGs. In: Proc. of Consumer Communications and Networking Conference. Las Vegas, USA (Jan 2016)
12. Ahmed, D., Shirmohammadi, S.: A Dynamic Area of Interest Management and Collaboration Model for P2P MMOGs. In: Proc. of IEEE Symposium on Distributed Simulation and Real-Time Applications. Vancouver, CAN (Oct 2008)
13. Carlini, E., Ricci, L., Coppola, M.: Reducing server load in MMOG via P2P Gossip. In: Proc. of Workshop on Network and Systems Support for Games. Venice, IT (Nov 2012)
14. Choi, S., Jung, K., Noh, S.: Virtual reality applications in manufacturing industries: Past research, present findings, and future directions. Concurrent Engineering 23(1), 40–63 (Mar 2015)
15. Freina, L., Ott, M.: A literature review on immersive virtual reality in education: State of the art and perspectives. In: Proc. of eLearning and Software for Education. Bucharest, RO (Apr 2015)
16. Greenwald, S., Corning, W., Maes, P.: Multi-user framework for collaboration and co-creation in virtual reality. In: Proc. of Conference on Computer Supported Collaborative Learning. Philadelphia, US (Jun 2017)

17. Guna, J., Jakus, G., Pogacnik, M., Tomazic, S., Sodnik, J.: An analysis of the precision and reliability of the Leap Motion sensor and its suitability for static and dynamic tracking. Sensors 14(2), 3702–3720 (Feb 2014)
18. Hilfert, T., Konig, M.: Virtual reality applications in manufacturing industries: Past research, present findings, and future directions. Visualization in Engineering 4(3), 1–18 (Dec 2016)
19. Khademi, M., Hondori, H., McKenzie, A., Dodakian, L., Lopes, C., Cramer, S.: Free-hand interaction with leap motion controller for stroke rehabilitation. In: Proc. of Human Factors in Computing Systems. Toronto, ON (May 2014)
20. Kim, K., Yeom, I., Lee, J.: HYMS: A Hybrid MMOG Server Architecture. IEICE Trans. on Information and Systems 87-D, 2706–2713 (Jan 2004)
21. McMahan, R., Alon, A., Lazem, S., Beaton, R., Machaj, D., Schaefer, M., Silva, M., Leal, A., Hagan, R., Bowman, D.: Evaluating natural interaction techniques in video games. In: 2010 IEEE Symposium on 3D User Interfaces. Waltham, US (Mar 2010)
22. Pretto, N., Poiesi, F.: Towards gesture-based multi-user interactions in collaborative virtual environments. In: Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLII-2/W8. pp. 203–208. Hamburg, GE (Nov 2017)
23. Schmalstieg, D., Gervautz, M.: Demand-driven geometry transmission for distributed virtual environment. Computer Graphics Forum 15(3), 421–431 (Aug 1996)
24. Yahyavi, A., Kemme, B.: Peer-to-peer architectures for massively multiplayer online games: A survey. ACM Comput. Surv. 46(1), 1–51 (Jul 2013)