Handling Heterogeneity in Collaborative Networked Virtual Surgical Simulators

a dissertation presented

by

Christian Diaz

to

The Department of Computer Science

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Engineering

Universidad EAFIT

Medellin, Colombia

June 2015

Thesis advisor: Helmuth Trefftz Christian Diaz

Handling Heterogeneity in Collaborative Networked Virtual Surgical Simulators

Abstract

Stand-alone and networked surgical virtual reality based simulators have been proposed as means to train surgical skills with or without a supervisor nearby the student or trainee. However, surgical skills teaching in medicine schools and hospitals is changing, requiring the development of new tools to focus on: (i) importance of mentors role, (ii) teamwork skills and (iii) remote training support. For these reasons, a surgical simulator should not only allow the training involving a student and an instructor that are located remotely, but also the collaborative training of users adopting different medical roles during the training session. Collaborative Networked Virtual Surgical Simulators (CNVSS) allow collaborative training of surgical procedures where remotely located users with different surgical roles can take part in the training session. To provide successful training involving good collaborative performance, CNVSS should handle heterogeneity factors such as users' machine capabilities and network conditions, among others.

Several systems for collaborative training of surgical procedures have been developed as research projects. To the best of our knowledge none has focused on handling heterogeneity in CNVSS. Handling heterogeneity in this type of collaborative sessions is important because not all remotely located users have homogeneous internet connections, nor the same interaction devices and displays, nor the same computational resources, among other factors. Additionally, if heterogeneity is not handled properly, it will have an adverse impact on the performance of each user during the collaborative session. In this document, the development of a context-aware architecture for collaborative networked virtual surgical simulators, in order to handle the heterogeneity involved in the collaboration session, is proposed. To achieve this, the following main contributions are accomplished in this thesis: (i) Which and how infrastructure heterogeneity factors affect the collaboration of two users performing a virtual surgical procedure were determined and

analyzed through a set of experiments involving users collaborating, (ii) a context-aware software architecture for a CNVSS was proposed and implemented. The architecture handles heterogeneity factors affecting collaboration, applying various adaptation mechanisms and finally, (iii) A mechanism for handling heterogeneity factors involved in a CNVSS is described, implemented and validated in a set of testing scenarios.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Related Works 6

1.3 Main Contributions 23

1.4 Dissertation Overview 25

1.5 Forerunning Publications 26

2 Heterogeneity Factors Affecting Collaborative Performance in CNVSS **28**

2.1 Description of the SOFA Framework 29

2.2 CNVSS using a peer-to-peer architecture 36

2.3 CNVSS using a Hybrid Client-Server Architecture . . . 51

2.4 Conclusions 73

3 Description and Implementation of a Context-Aware Architecture for a CNVSS **76**

3.1 Components for Measuring and Managing the Context 77

3.2 Component for action and modification of the CNVSS 87

3.3 Implementation of a context-aware architecture in a CNVSS 99

3.4 Conclusions 109

4 Inference Mechanism for Handling Heterogeneity Factors in a CNVSS **110**

4.1 Inference mechanisms for handling heterogeneity in CVEs. 111

4.2 Artificial Intelligence Methods to be used as Inference Mechanisms. 116

4.3 Description of the Inference Mechanism. 120

4.4 Experimental Setup and Results 137

4.5 Conclusions 154

5 Conclusions and Future Research **156**

5.1 General Conclusions 156

5.2 Future Research 158

6 Annexes **160**

References **175**

List of Figures

1.1 Scheme describing the heterogeneity factors affecting the collaboration performance in a CNVSS. 6

1.2 Different groups of metrics used in the literature for measuring directly and indirectly the collaboration of users in CNVSSs and CVE. 24

2.1 Flow of the data and processing steps performed by a Surgical Simulator. 33

2.2 Network elements and the functional relationships that compose the peer-to-peer architecture implemented by our CNVSS. 38

2.3 Scheme describing how are distributed the *OmniDriver* and *RemoteOmni* driver component when two user are collaborating. 39

2.4 Data structures used for the simulation of the gall bladder. Left: visual data structure. Center: collision data structure. Right: deformable modeling data structure. . 42

2.5 System setup used for the experimental test. The users interact with the collaborative surgical simulation using Machine 1 and Machine 2. In Machine 3, the network impairments are simulated using NetDisturb. 45

2.6 Collaborative removal of the gall bladder. The gray instrument is the representation of the local haptic device and the yellow one is the representation of the remote haptic device. 46
2.7 Normal plot for frames per second variable. 48
2.8 Main effects plot for frames per second variable. 49
2.9 Main effects plot for task completion time variable. . . 49
2.10 Main effects plot for task completion time variable. . . 50
2.11 Network elements and the functional relationships that compose the hybrid client-server architecture implemented by our CNVSS. 55
2.12 Components developed to implement the hybrid client-server architecture. 57
2.13 Virtual reality cholecystectomy surgical scenario. . . . 58
2.14 Collaborative removal of the gall bladder. Right user handles the attaching instrument and Left one handles the carving and clip attaching instrument. 62
2.15 Daniel's Plots for determining the significance of factors 68
2.16 Main effects plots for inconsistency, number of errors and time difference response variables 70
2.17 Response Surface Plots for central composite DOE . . 72

3.1 Example of the user Interface implemented in the CNVSS for getting the role and preferences from the user. 84
3.2 Graphical representation of the three steps of a learning curve. 86
3.3 Examples of deformation (top) and collision (bottom) models of the Liver. 88
3.4 Two machines executing the bandwidth modification mechanism. 95

3.5 Two machines executing the collision mapping for allowing multiresolution models between the client and client/server. 98
3.6 Network elements, components and the functional relationships that compose the context-aware architecture implemented by our CNVSS. 100
3.7 Different surgical scenarios used for the experimental test. The bottom shows the deformation model, the center the collision model and the top the visual model. From left to right the complexity of the resolution and algorithms of the surgical scenario is decreased. 107

4.1 Black box model representation of the proposed inference machine. 121
4.2 Diagram of the proposed inference mechanism. 123
4.3 Membership functions defined for the Choosing Server Parameters Subsystem. 128
4.4 Membership functions defined for the Choosing Network Parameters. 131
4.5 Graph showing the value of J calculated for each experimental run (blue color) and the value of J predicted by the ANFIS trained (red color). 135
4.6 Screenshots of the context-awareness CNVSS when two users are collaborating. 144
4.7 Changes produced by the adaptation mechanisms in the variable FPS of the client-server machine. 145
4.8 Changes produced by the adaptation mechanisms in the Bandwidth used by the CNVSS. 148
4.9 Results obtainted for the variable task completion time. 152
4.10 Results obtainted for the variable number of errors. . . 153

I dedicate my dissertation work to my family and friends. A special feeling of gratitude to my loving wife, Luisa whose words of encouragement and push for tenacity ring in my ears, without her anything of this could be possible. She is my inspiration to address any challenge. A special dedication to my little daughter Matilde, her eyes and smile are the motivation and fuel for the most difficult problems.

This work is also dedicated to my parents, Clemente and Cristina, who have always support me and whose good examples have taught me to work hard for the things that I aspire to achieve.

I also dedicate this dissertation to my friends, Carlos and Bernardo, who have supported me through all the process.

Acknowledgments

I would like to express my gratitude to my advisor Prof. Helmuth Trefftz for support me during all of my Ph.D study and research, for his enthusiasm and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would like to thank my co-advisors Prof. Lucia Quintero, Prof. Diego Acosta and Prof. Sakti Srivastava. Their knowledge in each of their expertise topics was fundamental for the development of my Ph.D dissertation.

I must also acknowledge the people of the Clinical Anatomy Lab at Stanford University for their help during my internship; their suggestion and recommendation open my mind. Special thanks for the people of the Virtual Reality Lab at EAFIT University, they support me during the entire experimental tests required by the Ph.D dissertation.

The way to get started is to quit talking and begin doing.

Walt Disney

1 Introduction

1.1 Motivation

From its beginnings risk factors have been associated with the execution of surgical procedures, not only by the complexity involved in performing a procedure of this type on a patient, but also by the complexity involved in training an expert who will perform it. Considering the first challenge, several advances have been made for improving the conditions of the operating room and thus reducing the risk of infection or complications during the performance of a surgical procedure [10]. Moreover, considering the training of future surgeons, the model of education and conventional training in the medical field has been associated with a mentor-apprentice role and a "see one, do one and teach one" model, where the mentor, that is the expert surgeon, shows several

times the learner how she/he should perform the surgical procedure and later let her/him taking an active role in the surgical procedure under her/his supervision [86]. Often the common place for performing this training process is in the operating room making a real surgical procedure.

In the last twenty years, the concept of simulation for medical staff training has become very important in the world because of the drawbacks involved with the training model "see one, do one and teach one" [36]. Some of these drawbacks are the risks associated with the wellbeing of the patient, high surgical costs by rework, additional supplies or delays performing the procedure, longer learning and training times and no scenario available for trial and error, among others. Various simulation approximations have been proposed for reducing the impact of all the aforementioned disadvantages, most of them seeking to reproduce the conditions of the patient and operating room, in a controlled environment in order to allow the apprentice to acquire a skill set, depending on the surgical procedure, before facing a real patient [80]. So, several simulators have been developed for the training of surgical procedures such as: laparoscopic surgery [96], microsurgery [56], cardiac surgery [68], neurosurgery [18], arthroscopy [99], among others. Surgical simulators can be categorized as physical or virtual. In this document, physical simulators are understood as those using physical or real materials for simulating the sensory perception of the apprentice when she/he is interacting with the patient during the surgical procedure execution [84]. When sensory perception is mediated or simulated using a computer, a virtual environment and human interaction computer interfaces, such as haptic devices, they are called virtual simulators. This thesis focuses on the second category of simulators.

So far, three kinds of surgical simulators using virtual reality have been

developed: stand-alone, networked, and collaborative networked virtual surgical simulators. Stand-alone virtual surgical simulators allow the student to receive training on virtual surgical procedures without expert mentoring or supervision. These kinds of simulators are able to record the surgical performance of the student, and depending on his/her skills, provide him/her with feedback and curriculum guidance.

However, medical surgical training is changing worldwide, mainly due to: (i) A change to a training paradigm that highlights mentor role and also the importance of teamwork besides basic surgical skills training before entering a surgical room, and (ii) the low number of expert surgeons located in distant regions or with time avalailable to provide face to face surgical training [34, 65, 81]. For these reasons, networked virtual surgical simulators were developed due to the growing availability of computer networks and the decreased availability of expert laparoscopic surgeons in remote regions. This second kind of surgical simulator has been created to allow a student to be trained remotely by an instructor. In such a system, the instructor can perform the procedure remotely while the student not only watches, but also feels (haptic feedback provided) what the instructor is touching without actually participating in the execution of the surgical procedure. Finally, considering that a surgical procedure involves teamwork among medical practitioners, a third kind of virtual surgical simulator, Collaborative Networked Virtual Surgical Simulators (CNVSS), was proposed by [98]. CNVSS allow for the collaborative training of users located remotely with each member playing a role during the training session. Therefore, CNVSS are not only useful for training basic surgical skills, but also for training team members to work collaboratively in a surgical room. These skills are required to perform a successful surgical procedure.

Several components interact with each other in a CNVSS for creat-

ing a shared virtual surgical scenario where the mentor and apprentice can perform a training session, and for reducing the negative impact involved with the level of realism provided by the simulator and the remote geographical localization of the users. CNVSS components can be grouped as [29]: (i) simulation components, (ii) user interaction components and (ii) network and collaboration components. The first set of components contains all data structures and algorithms, named in this document as surgical scenario properties, required for simulating the events occurring during a surgical procedure, such as attaching, cutting, suturing, carving, among others. An enriched simulation environment is needed for simulating these events, involving the collision detection between the simulated structures, i.e. surgical instruments and organs, the implementation of topological changes for simulating cutting or carving operations, and the deformation of viscoelastic structures, i.e. when a surgical instrument is probing a tissue. Because the simulation of these events is complex, considering the mathematical and computational operations involved, this set of components requires the major amount of machine capabilities used by a CNVSS. The second set of components allow for the tracking of user's intention or what the user wants to do within the virtual surgical scenario, and using the simulation set of components provides feedback to the user. For example, visual and haptic rendering are processes related with user interaction components, and allow the user to see and feel the state of the surgical scenario during the execution of the virtual surgical procedure. Finally, the third set of components read the state of the simulation and transmits it among user's machines in order to guarantee that each user sees and feels the same virtual surgical scenario at any given time. However, the shared state between the user machines of the virtual surgical simulation cannot be strictly maintained because the delay of the network connection [91]. Computational complexity and cost of this data transmission will depend on the complexity of the data structures

used in the simulation components.

To guarantee a good collaborative training of users interacting through a CNVSS, it is necessary to consider that there is a set of heterogeneity factors affecting negatively the collaboration of users, and therefore the effectiveness and efficiency of training involving the mentor and apprentices. These heterogeneity factors are defined as the differences in infrastructure where the CNVSS runs, such as network conditions and machine capabilities. For example, if the available network bandwidth is less than the minimum required by the CNVSS or the user machines capabilities are not adequate for running the simulation, then the collaboration among users is affected as is described in chapter 2 of this document. For this reason, a set of mechanisms must be proposed to mitigate the negative impact of heterogeneity factors. For this purpose, other heterogeneity factors related with the differences of the virtual surgical scenario properties can be used. For example, the simulation components of the CNVSS running on each user machine can use an algorithm or data structure resolution depending on the machine capabilities, allowing a fluid execution of the simulation and a good collaborative performance. Finally, a third set of heterogeneity factors is dependent of users and is defined as differences in users and team skills involved in the collaborative session. In a collaborative task, it is common for users to have different levels of expertise and it cannot be controlled when the training session starts. The different groups of heterogeneity factors described can be observed in figure 1.1.

Therefore, the main goal of this thesis is to propose a strategy bu means of which, handling the three groups of heterogeneity factors defined above, a good collaborative training performance can be guaranteed in a CNVSS. The related works supporting the contributions

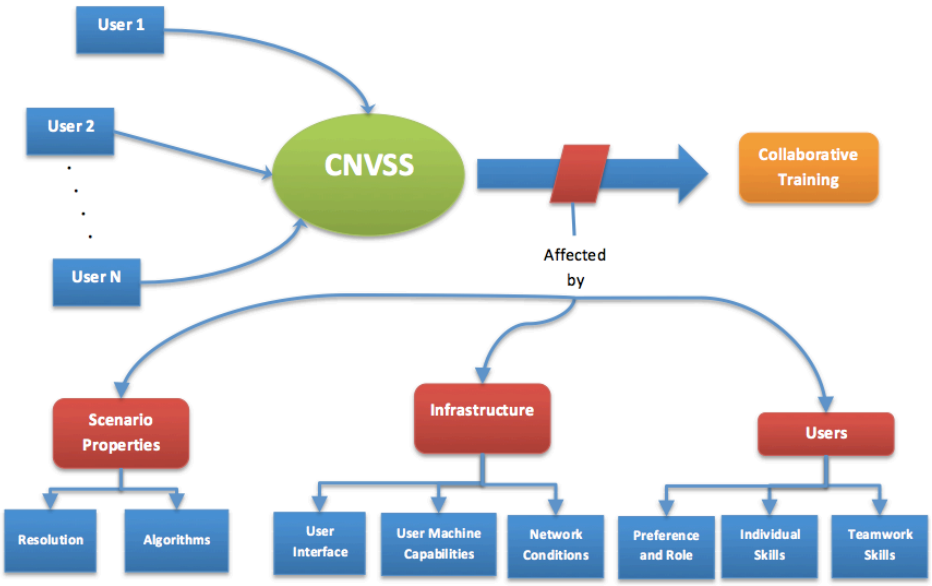


Figure 1.1: Scheme describing the heterogeneity factors affecting the collaboration performance in a CNVSS.

proposed in this thesis and the approach addressed are described in the next section.

1.2 Related Works

Several research projects have been developed in topics related to the proposed work. In order to give a general scope of the literature, these are grouped as follows: (i) collaborative virtual surgical simulators, (ii) heterogeneity factors affecting CVEs, (iii) handling heterogeneity in CVEs and (iv) Context-aware mechanisms used so far in CVEs.

In the first topic we describe CNVSSs reported in the literature and how the authors address the issues involved in this kind of CVEs. In the second topic we review the works that evaluate the effect of some heterogeneity factors related with infrastructure in CVEs and provide

a list of the factors evaluated by each author as well as the numerical values used in the experiments. In the third topic we present an overview of the methods proposed to handle heterogeneity in CVEs. Finally, considering that almost all methods proposed to handle heterogeneity in CVEs are based on adaptive systems that are self-ware about the system conditions, an overview of context-aware system in CVEs is presented.

1.2.1 Collaborative Networked Virtual Surgical Simulators

Several works have addressed the issues related to the development of CNVSSs using various strategies. Two specific middleware systems for CNVSSs are proposed in [67] and [51]. These middleware systems use specific protocols, different architectures, compensation mechanism, among other strategies to address network impairments such as jitter, delay and packet loss, in order to maintain adequate collaboration and shared state consistency of the virtual environment, even though the major part of the simulators developed have focused on reducing the effect of network latency. [38] and [37] propose a CNVSS capable of maintaining collaboration with delays of up to $170ms$. They describe a pseudo-physical simulation strategy and an event collection mechanism based on scene-graph representation of the virtual environment to avoid the instabilities in the simulation due of network latency. Additionally, the collaborative simulator allows the instructor and students to see, touch and interact with the same virtual organ in the scene and also provide haptic guidance. A collaborative simulation of the gall bladder removal procedure between an instructor located in USA and a group of students located in Australia, was used to evaluate the system.

Marks et al. [58] have evaluated the adaptability of some game engines

in order to simulate scenarios for surgical training. They evaluated three game engines (Id Tech 4, Unreal Engine and Source Engine) considering features such as scenario edition, content and gameplay. They also analyzed whether the engines supported collaborative interaction among the players, the quality of the physical simulation of rigid and deformable objects and the flexibility to create custom surgical scenarios. Although they conclude that game engines provide several interesting features for the development of CNVSSs such as, network impairments and machine resources management, the modeling of suitable physics-based deformable models and topological changes for surgical training are not properly handled. Additionally, these engines have synchronization problems in the physical simulation of rigid and deformable objects among players. Finally, the game engines only consider machine and network resources management to handle the heterogeneity of network connection and machine capabilities, but do not consider the preferences and role of the user in the collaborative task as factors generating heterogeneity.

A software framework surgical simulation based on a multi-controller and multi-viewer model-view-controller (MVC) pattern was developed and tested in [57]. This framework is designed in such a way that multiple execution threads run in parallel and concurrently at different rates and in separate cores on the CPUs allowing two-ways interaction, not only visualization but also haptics feedback. Moreover, the framework is extended to provide collaborative surgical training and telementoring capabilities with the addition of a network controller component. The Network Controller component enables remote users to collaborate and share the resources of the framework. The Controller runs asynchronously in a separate thread and allows for the communication of resources using TCP or UDP protocols. The performance of the framework under different network latency conditions is evaluated using a

master-slave setup for telesurgery applications. The difference of the instrument path in the master and slave site is reported. However, they do not report experimental tests in order to evaluate the performance of the framework in collaborative surgical tasks for training. Also, the framework proposed does not implement strategies to handle heterogeneity in CNVSSs.

An immersive CVE to train cataract eye surgery is described in [88]. This simulator not only allows for the collaborative training of the procedure, but also the haptic guidance (telementoring) of the trainee. The trainer could feel the movements of the trainee and the trainee could feel the movements of the trainer. Additionally, in this paper several mechanisms are explored to address network impairments and provide good haptic guidance between the trainer and the trainee. However, they do not handle the heterogeneity factors defined in the introduction and focused on telementoring and not collaborative tasks.

The following works propose approaches to address several infrastructure heteorogeneity factors together in a CNVSS, all of them grouped as network conditions. A middleware is proposed to handle network connection issues while maintaining the consistency of the collaborative networked virtual surgical environment in [98], [74] and [75]. The proposed middleware is composed by: network management approaches (including services management), collaboration mechanisms, adaptive protocols, various deformation models, 3D to 2D synchronization and flexible computation policies. They try to manage the heterogeneity of the network connection and machine capabilities of the user, implementing two computation policies to handle the deformation: (i) local computation policy, where deformation computation is off-loaded to individual participants; and (ii) the global computation policy, where a participant with powerful computational capability is assigned as the

server in the system. The bandwidth requirement for the former is low since the data transfer only involves the parameters of the computation. The latter is applicable when the network resources are adequate and for situations where the computing power of the other participants is limited, but the criteria (numerical formula) to switch between policies is not explained or described. In [26], the authors develop a high-performance, network-aware, collaborative learning environment. They propose a middleware system that monitors and reports network conditions to network-aware applications that can self-scale and self-optimize based on network *weather reports*. The core system and applications have been developed within the context of two medical testbeds: a clinical anatomy testbed and a clinical skills testbed.

A framework called SOFA (Simulation Open Framework Architecture) aiming at the development of surgical simulators is proposed in [5] . The SOFA architecture relies on several innovative concepts, in particular the notion of multi-model representation. In SOFA, most simulation components, deformable models, collision models, instruments, etc, can have several geometric model representations, connected together through a mechanism called *mapping*. However, this framework is not designed for the development of collaborative networked surgical virtual environments. Finally, a review about CNVSSs is presented in [76]. They identify the challenges characterizing these CVE, and provide a detailed explanation of the techniques used to address these issues. Finally, they describe collaborative surgical environments developed for different medical applications. In this review, strategies to handle heterogeneity in CNVSSs are not reported and they do not consider all the three groups of heterogeneity factors listed above.

The CVEs for surgical simulation described above are not self-ware of the heterogeneity factors affecting collaboration, and they do not

implement self-adaptation strategies in order to handle the heterogeneity factors grouped as: infrastructure, surgical scenario properties and user heterogeneity factors.

1.2.2 Heterogeneity Factors Affecting Collaboration in CNVSSs and CVEs

Early works have evaluated how different factors such as jitter, delay and packet loss degrade simulation and consistency in CNVSS [51, 67]. [38] and [37] describe how the performance of CNVSS is affected by jitter and network latency. They report that latency produces vibrations in the force effected by the haptic device and degenerate physical simulations of organs and tissues. However, they do not determine in which latency and jitter values these issues arise. To compensate for the effect of network latency in collaborative surgical simulations, a pseudo-physical approximation is proposed. This solution decreases the realism of the deformable calculation, but guarantees the stability of the simulation. In a similar manner, [26] describe an experiment to determine the effect of network latency on touch perception of virtual organs using the *SPRING* framework. From the experiments it is concluded that subjects performing a virtual surgical task with a delay longer than 50 ms are not able to perceive as different the forces of different magnitudes. Additionally, it is reported that force feedback becomes unstable when there are latencies of the order of 100 ms. [40] assessed how the shared state consistency of a surgical augmented reality environment is affected by the variation of the network delay. They found that when the delay of the network was longer than 50 ms, the consistency of the shared state was considerably affected.

A review of CNVSS by [76] shows the challenges which characterize these Collaborative Virtual Environments (CVEs) and a detailed

explanation of the techniques used to address them. Finally, several collaborative surgical environments developed for different medical applications are described. The works described above have developed different collaborative and networked surgical simulators. However, as far as we know, there are no reports available on which factors affect collaboration in these kinds of virtual environments the most, and the major part of the works have focused on evaluating the effect of latency and jitter on force feedback perception and simulation stability of CNVSS.

A middleware to provide collaboration services to stand-alone surgical simulators was developed in [74], [98], and [75]. The middleware performance was tested by evaluating the effect of the number of users and collaboration strategy (coupling and token control) on the average frame rate of each user machine and the latency measured in the network. However, in the reported experiments, neither the impact of network parameters nor the impact of machine capabilities on collaboration performance in CNVSS was evaluated. It is also reported that the middleware guarantees consistency, but no quantitative evidence is offered to prove such a claim.

Other researchers evaluated which infrastructure heterogeneity factors, specifically network conditions and machine capabilities, affected collaboration the most in other types of collaborative environments. A factorial design of experiments is proposed in [72] and [6], to evaluate the effect of delay, jitter, and complexity of the task on human performance in a CVE. It is concluded that all of these factors greatly impact collaboration. For example, the task completion time and the number of errors increase by approximately 40 % for jitter values of 263 ms and for delay values of 200 ms. However, these experiments were not conducted for virtual surgical procedures.

Some researchers have evaluated collaboration in specific tasks, such as the handshake task, under different network conditions (jitter, delay, bandwidth, and percentage of packet loss) [25, 39]. They report that delays, jitters, bandwidths and packet loss percentages larger than 20 ms, 1 ms, 128 kps and 10 %, respectively, are unacceptable for collaboration. Using the same task but only evaluating the effect of the delay on collaboration, [4] report that a delay longer than 600 ms deteriorates haptic perception. In addition, the task completion time increases more than 50 % for delays over 1800 ms. [25] and [39] report shorter delays compared to those reported by [4], because the first one considered the effect not only of the delay, but also the effect of the jitter, bandwidth and packet loss. So, the combined effects of these factors have a major impact on collaboration.

[93] evaluate how the network factors affect the haptic interaction in distributed virtual environments. Through the use of a qualitative assessment of the haptic perception, they report that delays, jitters and percentages of packet loss above 30 ms, 3 ms and 10 %, respectively, are unacceptable for effective collaborative interaction. [44] examined the impact of delayed haptic and visual feedback from the partner in a collaborative virtual environment with two operators. They found that both visual and haptic delay hinder task performance in terms of loss of contact with the target object and acquisition time. However, haptic delay had a larger impact on performance than visual latency. In the aforementioned study of [44], continuous haptic delay could be perceived to be starting from around 50 ms in a CVE.

[70] present a survey of works which reviews how network conditions affect collaboration in CVE involving haptic perception. From the review they conclude that the haptic channel is affected by small amounts

Table 1.1: Upper and lower limits of the network factors evaluated in the literature.

Author	Jitter (ms)	Delay (ms)	Packet Loss (%)
[72]	12-163	10-200	NA*
[25]	0-25	0-150	0.001-100
[4]	NA	0-2000	NA
[93]	1-15	0-50	0.1-50
[6]	NA	0-200	NA
[40]	NA	0-50	NA
[44]	NA	0-50	NA

* NA: *Not Analyzed.*

of jitter, packet loss and latency. Table 1.1 reviews the magnitude of the upper and lower network factors evaluated in the literature.

The research works mentioned describes how network conditions affect collaboration in CVE. However, as confirmed in [72], [25] and [39], network conditions affect collaboration depending on the evaluated application, the type and tightly coupled level of the collaborative task peformed [71]. Therefore, in order to determine the effect of these factors on collaboration in CNVSS, an experimental test involving surgical tasks and a surgical scenario including the simulation associated engine is required.

On the other hand, few research projects have considered the evaluation of the impact of machine factors in CVE. In [101], the impact of heterogeneity of user machines on the frames-rate of a networked virtual environment is evaluated. They conclude that the inequity of the machines in a Networked Virtual Environment (NVE) session

Table 1.2: Upper and lower factors of the machine capabilities evaluated in the literature.

Author	Processor Speed (GHz)	RAM Capacity (MB)	Graphic Card	Network Card Speed (Mbps)
[102]	0.4-1.5	256-1024	†T1-‡T2	*NA
[40]	1.5-2.8	512-1024	• T3-**T4	100
[44]	2-3.2	512-1024	NA	NA

*NA: *Not Analyzed*. †T1: Intense3D,16 MB, ‡T2: GeForce2,32 MB, • T3: GeForce 4Ti4200 and **T4: GeForce 4Ti4600

makes the machines with lesser resources vulnerable to be flooded with large amounts of information generated by high-end machines [102]. However, how the differences in user machine capabilities impact collaboration in CVEs involving rich simulation behavior [59], such as in CNVSSs, is not evaluated.

Some works evaluating the impact of network factors have used different machine capabilities in their experiment configurations [40, 44], but they do not conclude how these factors impact collaboration in the virtual environment (Table 1.2).

To summarize, to the best of our knowledge, the following issues have not been considered thus far in the literature:

1. The impact of network and machine factors on the performance in CNVSSs has not been evaluated.

2. The impact of interaction between factors on the performance in CNVSSs has not been evaluated.

3. Research experiments performed until now have not followed a

statistical design of experiments (only one variable has been evaluated at a time).

1.2.3 Handling heterogeneity in CVEs

Basically, there are two strategies to handle the heterogeneity of CVEs. The first one tries to hide the heterogeneity of the system by reducing the performance of the CVE to a *lowest common denominator*, that is, the user with the lowest capacity determines the performance of the CVE. This strategy forces that even the users with high end machines and good network connections experience a bad quality CVE. The second one tries to expose the heterogeneity of the system by distributing all the resources available among the user depending on their preferences and capabilities. However, in this strategy it is necessary to address issues of *fair play* that result when users must interact even though they have received different levels of information about the environment. The lack of fairness can reduce the quality of the experience [91]. Since our proposed method is based on the second strategy to handle the heterogeneity of CNVSSs, we describe related works using this strategy.

[19] describe the design and prototype implementation of an adaptive QoS management framework for collaborative multimedia applications in distributed, heterogeneous environments. The overall goal of the framework is to locally adapt the shared information to meet the capabilities, interests and current state of each collaborating client while preserving the semantic content of the information to maintain effective sharing.

In [101] a framework to handle the heterogeneity of CVEs is proposed. This framework was developed for applications that have the following

characteristics: (i) several types of information are exchanged between participants in a collaborative task; (ii) each type of information can be presented with different degrees of resolution; (iii) users can choose what type of information they are more interested on at a given time and (iv) participants computers are heterogeneous in terms of computing power. Then they propose a mathematical model to represent the global constraints and the users' preferences. The mathematical model is expressed as linear equations. They also propose a switchboard architecture in which a server acts as a buffer, providing slower nodes with a sub-sample of messages generated by faster nodes in a controlled manner that is determined by the client node's capacity and users' preferences.

A model providing dynamic QoS support for NVR (Networked Virtual Reality) services is proposed in [92]. This paper is focused on the overall process of end-to-end QoS (Quality of Service) negotiation and service adaptation in order to handle heterogeneity. The main issues to handle heterogeneity are: (1) user terminal and access network constraints, (2) way(s) of expressing user preferences in terms of application components (media elements), (3) dynamic resource availability and cost, and (4) mapping of user/application requirements to transport QoS parameters. In addition, they propose a generic client profile as a way of expressing client capabilities and preferences, and a generic service profile to specify dynamic service requirements. The mapping of service profile and client profiles are carried out implementing a mathematical model that takes into account the parameters mentioned before.

Araujo et al. [22] proposed an adaptive strategy to handle heterogeneous devices and different network capabilities in CVEs. This strategy considers data adaptation and control adaptation. Data adaptation implies a transformation of the data often used by the application to

versions that fit into available varying resources. Control adaptation means to modify the application flow, that is, the application behavior. The proposed system decides when to apply data adaptation and control adaptation in the following cases: (i) the user machine does not support a specific data type, (ii) minimal fps required by the user cannot be achieved during the application runtime and (iii) the client terminal does not have enough network bandwidth for receiving and transmitting required data. The mechanisms are triggered by the MPEG-J application and receives feedback from the resource monitor component when any resource reaches a threshold previously-established by the programmer.

Fijinoki et al. [33] proposed three new techniques in order to support network and machine heterogeneity in NVE based on client-server architecture. The three proposed techniques are: (i) application layer multicast transmission rate pruning, (ii) fairness control for delay-sensitive activities (the token-bucket algorithm) and (iii) bandwidth compensation by a combination of server-side and client-side dead reckoning. The first technique manages different LODs (Level of Detail) and transmission rates depending on the network conditions and machine capabilities of the client. They use multicast routers to selectively drop packets that belong to any other LODs that it is not supported by the client and to achieve the transmission rate required by the client. The second technique uses a token value attached to the messages packets in order to synchronize the shared state among the clients and the server. Whether the token of the message received by the server are considered a late update of the shared state, the message is dropped by the server. Finally, the third technique uses a combination of server-side and client-side dead reckoning to compensate issues related with reduced bandwidth and high end-to-end delay of the clients.

[45] propose a theoretical method for adaptation of the heterogeneity of distributed 3D contents. The method consists of a rendering service middleware, and device-based decision making algorithm to define a suitable LOD (Level of Detail) of the 3D contents. Additionally they propose a mathematical formulation to adapt the level of detail of the 3D content depending on the conditions of the system.

CNVSSs are CVEs exposing unique technical requirements. In this kind of CVEs the users must be able to interact, touch and cut with the same deformable object, in real-time, while the shared state consistency is guaranteed, and the heterogeneity factors discussed above are present. As far as we know, in the current literature there are not proposed strategies to handle heterogeneity in CNVSSs, considering the three groups of heterogeneity factors described in section 1.1 and applying the data adaptation and control adaptation (see [22]) of the following parameters:

- The *resolution* of the data structures and *algorithms* involved with the collision detection, physical deformation calculation and visual and haptic rendering models, that is the virtual surgical scenario properties defined in section 1.1.
- The *amount of computation* executed locally versus remotely for each user machine.
- *Mechanisms* to handle network issues such as bandwith limitations and network delays.

The strategies to handle heterogeneity described in this section are characterized by: (i) they are self-ware about the system environment, (ii) they use inference decision-making techniques to provide adaptation of the CVE to heterogeneity factors and (iii) they use methods to adapt simulation parameters and properties of the CVE. Considering

these characteristics, these systems can be considered to be conext-aware. Finally, in the next section we present some definitions and works related to this topic, because the architecture proposed for our CNVSSs is based on a context-aware architecture and mechanisms.

1.2.4 Context-aware system in CVEs

Context-aware systems have been proposed as a solution to the new network architecture challenges and distributed application issues [73]. As mentioned above, the context-aware systems adapt their behavior depending on the environment conditions. To carry out this adaptation, these systems are composed by (see [108]): (i) A context-management component, (ii) a context-based inference machine and (iii) an action component. The context management component measures the state of the context, and passes this information to the inference machine which makes decisions in order to adapt the behavior of the system calling the action component. Several research papers have been published describing various strategies for each of these components. In [108], a complete review of the works related to each one of these components is presented.

Also, context-aware has become somewhat synonymous with other terms such as: adaptive, reactive, responsive, situated, context-sensitive, environment -directed, resource-aware and user-aware [1]. Even though the term context-aware has not been used to describe CVEs with the properties mentioned in the previous paragraph, several of their synonymous have been used in some research works. These research works are described in the next paragraphs.

A graphics resource-aware middleware architecture, called MADGRAF, is described in [3]. This middleware makes it feasible to run complex

3D graphics applications on low end mobile devices over wireless networks. In MADGRAF, a server can perform mobile device-optimized pre-processing of complex graphics scenes in order to speed up run-time rendering, scale high-resolution meshes using polygon or image-based simplification, progressively transmitting compressed graphics files, concealing transmission errors by including redundant bits or perform remote execution, all tailored to the client's capabilities.

A solution for the user-aware VE (Virtual Environment) management using a multi-agent architecture that identifies, using intelligent agents, user characteristics and preferences, as well as his/her cognitive capacity during the navigation in the environment is presented in [7]. The main objective of the system proposed by the authors is to adapt the behavior of the system to the users' individual needs, capabilities and desires.

[89] propose a novel middleware called MARCO (Middleware Architecture for Remote Collaboration) to facilitate development of Distributed Real-time Collaborative Visualization applications which guarantee high data availability and scalability in terms of collaborative groups and members. MARCO is adaptable to computing devices with varying display capabilities. In order to make MARCO adaptive, two components are included. The first one is called Collaboration Engine (CE) which is responsible for real-time collaboration request handling, dynamic virtual organization management, and collaboration event notifications. The second one is called Profile Engine (PE) which is responsible for dynamically managing run-time device configurations, passing different configuration parameters to the execution logic to perform collaboration activities, and interacting with each device according to its profile configuration. The PE carries out an adaptive process.

A model of event communication to be used in an adaptive, resource aware distributed virtual environment (DVE) is proposed in [107]. They consider the problem of maintaining real-time state consistency between nodes connected over wireless networks. In order to deal with the variable resource availability, inherent to wireless networking, dynamic adaptation of DVE event streams is necessary. Such adaptation must consider the resource requirements of individual event streams as well as the available resources. Their event stream model, based on available and required resources, is then presented within the context of a client-server DVE. Analysis of the event traffic is used to describe the network resource requirements of individual event streams and demonstrate how they influence the overall stream requirements.

A new 3-D Virtual Environment (3DVE) system named *QuViE-P* has been proposed in [47], which can greatly enhance the QoS that users actually perceive as good as possible, even when the resources for computers and networks are limited. In order to accomplish this, they focus on characteristics of the user's perceptual quality evaluation of 3-D objects. Then, they propose an effective QoS control scheme that introduces a relationship between system's internal quality parameters and user's perceptual quality parameters. Two important goals of the paper are (1) Introducing and effectively applying the knowledge on characteristics of relationship between system internal quality metrics and user level quality, based on relationship between user's viewpoint/gaze-direction and positions of the objects. (2) Introducing a reasoning function to dynamically find the objects of interest during runtime.

None of the context-aware systems are designed to handle heterogeneity of CNVSSs. For that reason these systems do not include variables related with the virtual surgical scenario properties in the adaptive model

to handle heterogeneity, for instance resolution of the data structures used in collision detection, deformation computation and, visual and haptic rendering algorithms. Additionally, these systems do not introduce in the model, the role and preferences of each user in the surgical teamwork to adapt the collaborative virtual surgical environment.

1.3 Main Contributions

The main contribution of this dissertation is a mechanism for handling heterogeneity factors involved with a CNVSS in order to guarantee the collaborative training of the users. To achieve this goal the following contributions have been made to the state of the art and are described in this document:

1. Which and how infrastructure heterogeneity factors affect the collaboration of two users performing a virtual surgical procedure were determined and analyzed. The group of experiments proposed has several elements that differ from those reported in the state of the art: (i) the described experiments are performed using a collaborative task and virtual surgical scenario where each user plays a specific role and the task execution has different degrees or levels of collaboration (i.e. tightly or loosely coupled), (ii) factors related to network conditions and machine capabilities are evaluated at the same time, applying a statistical design of experiments and using two different types of network architectures, peer-to-peer and hybrid client-server, in the CNVSS, and (iii) in order to measuring the collaboration effectiveness various metrics are calculated, i.e. number of errors and task completion time, evaluating the performance of the users taking part of the virtual surgical task as a team. Most of the related works reported so far have focused on using the first two sets of metrics, right and

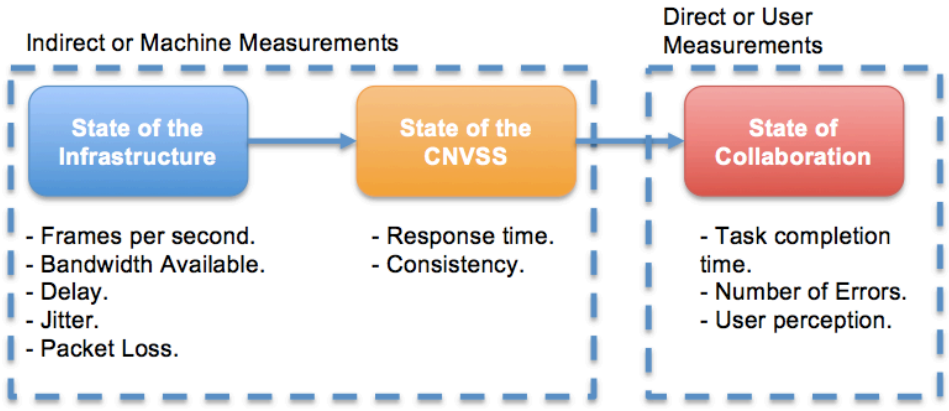


Figure 1.2: Different groups of metrics used in the literature for measuring directly and indirectly the collaboration of users in CNVSSs and CVE.

center side of figure 1.2, to evaluate the performance of a CVE and CNVSS.

2. A mechanism for handling all heterogeneity factors involved in a CNVSS is described, implemented and validated in a set of testing scenarios. This mechanism uses the infrastructure heterogeneity factors and users preferences as input variables and inferred, using artificial intelligence methods, how heterogeneity factors such as virtual surgical scenario properties and compensation mechanisms for network conditions must be modified in order to guarantee good performance of the collaboration. Metrics directly measuring the collaborative performance like number of errors and task completion time are considered.

3. A novel architecture for a CNVSS based on context-aware concept is described, developed and evaluated. This architecture allows: (i) measuring the heterogeneity factors involved in a CNVSS, (ii) applying an inference mechanism as described in the previous

contribution, (iii) sharing the state of the simulation between the machines of users taking part of the collaborative session, (iv) modifying properties of virtual surgical scenario and implementing mechanisms for the compensation of network impairments in real time, without affecting the collaboration of users performing a collaborative surgical task, even if the user is interacting with one of the objects in the surgical scenario that must be modified. The software architecture is implemented minimizing the penalties in real time performance and memory required.

1.4 Dissertation Overview

This dissertation is organized as follows:

Chapter 2 describes two types of network architectures implemented for the development of a CNVSS extending the SOFA simulation framework, as well as the development of a virtual surgical scenario to evaluate the collaboration and a design of experiments for evaluating how infrastructure heterogeneity factors affect the collaboration of users.

Chapter 3 presents the design and implementation of a novel architecture for a CNVSS based on the concept of context awareness. This architecture allows, the CNVSS, modifying in real-time its virtual surgical scenario properties and applying compensation mechanism for network impairments in order to mitigate the negative effects of infrastructure heterogeneity factors in the collaboration.

Chapter 4 describes the inference mechanism proposed for handling the heterogeneity factors involved with a CNVSS, the artificial intelligence methods used and the experiments performed for its implementation and evaluation.

Finally, in chapter 5 conclusions of this research and recommended future directions for further investigation are discussed.

1.5 Forerunning Publications

During the PhD project, parts of the research work in surgical simulation and results, reported in this thesis, have been published in conference proceedings and journals. These works published are listed as follows:

- Diaz, C., Trefftz, H., Bernal, J. and Eliuk, S. General Algorithms for Laparoscopic Surgical Simulators. Revista Ingenieria Biomedica, Volume 4, Issue 8, 2010, pp. 57-70.
- Diaz, C., Eliuk, S. and Trefftz, H. Simulating Soft Tissues using GPU approach of the mass-spring model. Proceedings of the IEEE Virtual Reality Conference, Waltham, MA, 2010, pp. 261-262.
- Diaz, C., Trefftz, H., Quintero, L., Acosta, D. and Srivastava, S. Collaborative Networked Virtual Surgical Simulators (CNVSS): Factors Affecting Collaborative Performance. In Presence: Teleoperators and Virtual Environments, Volume 22, No. 1, pp. 54 - 66, 2013.
- Diaz, C., Trefftz, H., Quintero, L., Acosta, D. and Srivastava, S. Collaborative Networked Virtual Surgical Simulators (CNVSS) Implementing Hybrid Client-Server Architecture: Factors Affecting Collaborative Performance. In Presence: Teleoperators and Virtual Environments, Volume 23, No. 4, pp. 393 - 409, 2014.

- Diaz, C., Trefftz, H., Quintero, L. Acosta, D., Srivastava, S. Adaptive Architecture to Support Context-Aware Collaborarive Networked Virtual Surgical Simulators. Lectures Notes in Computer Science, Virtual, Augmented and Mixed Reality, Applications of Virtual and Augmented Reality, Volume 8526, pp. 277-286, 2014.

Discovery consists of seeing what everybody has seen and thinking what nobody has thought.

Albert Von Szent

2

Heterogeneity Factors Affecting Collaborative Performance in CNVSS

As discussed in Chapter 1, there is a heterogeneous set of factors that affect the collaboration of users that are part of a training session on a CNVSS. These heterogeneity factors can be grouped into three categories: (i) infrastructure, (ii) user and (iii) surgical scenario properties (Figure 1.1).

Heterogeneity factors involved with user skills cannot be controlled in a training session as they depend on the individual and team skills of the

users when the training session is starting. Preferences and role of the users are included in this group as well. Meanwhile, infrastructure factors are given by the machine, network, and user interface conditions of the training session and depend largely on the available hardware for running the CNVSS. Finally, surgical scenario properties can be modified by the designer of the CNVSS or an intelligent system for better fitting with the infrastructure available and for ensuring good collaboration among users. Knowing what and how heterogeneity factors impact collaboration on a CNVSS is important because it allows to guide decisions and propose mechanisms for handling heterogeneity and guarantee user collaboration.

In this chapter the architecture of the framework SOFA [31], which was extended for the development of a CNVSS, is described. Subsequently, the implementation of two types of network architectures, peer-to-peer and hybrid client-server for the CNVSS, is reported. The architectures are used for evaluating, through a set of experiments, which heterogeneity factors impact user collaboration the most, using every type of architecture. The results reported in this chapter were published in [29] and [30].

2.1 Description of the SOFA Framework

The main goal of a framework for the development of surgical simulators, such as SOFA, is to create a virtual environment that simulates the interaction of a surgeon with the anatomy of a virtual patient associated with the surgical procedure being simulated. In most surgical procedures, two types of sensory channels are particularly important to provide appropriate realism when a virtual surgical scenario is simulated: visual and haptic perception. For these reason events such as attaching, suturing, carving, probing, attaching clip, cutting, among

others, must be simulated considering these two channels of sensory perception. To achieve this goal, from a technical point of view, a virtual surgical simulator must perform the following three tasks:

1. Sensing the interaction between the surgeon and the virtual environment in order to provide an appropriate feedback (visual and haptic rendering) to the surgeon. This is accomplished by a set of so-called interaction components.

2. Detecting the collision between the surgical instruments and the anatomical structures.

3. Calculating the deformation of the anatomical structures and applying the topological changes depending on the interaction with the surgical instruments.

The last two tasks are performed by a set of components known as simulation components. In a virtual surgical simulator each one of these tasks requires different algorithms and different data structures, such as:

- Haptic and Visual Model: Haptic and visual rendering algorithms and superficial or volumetric data structures, which are used for providing visual and haptic feedback to the user.

- Collision Model: Collision detection algorithms and superficial, volumetric or hierarchical data structures, which are used for determining if any of the virtual models, either anatomical structure or surgical instrument, are in collision.

- Behavior Model: Deformable behavior computation algorithms and volumetric data structures, which are used for computing the deformation of the virtual organs and tissues depending on the interaction with the surgical instruments

- Topological Model: Topological change algorithms and data structures.

There are different types of algorithms for each one of the tasks, which are differentiated by the amount of computation required by the algorithm. For instance, a mass-spring or finite element algorithm could be used in order to compute the deformation of the anatomical structures, the difference between these algorithms is that the mass-spring requires less amount of computation than the finite element algorithm, but the deformable behavior computed is less realistic using mass-spring than using finite element method, i.e. there is a trade-off between accuracy and amount of computation or hardware resources required. The same concept can be applied to other types of algorithms and data structures used for collision detection, topological changes, among others. Different algorithms for each one of the mentioned tasks are described in the Table 2.1.

Additionally, a method called mapping allows for the use of different data structures with different resolutions for each one of the algorithms listed in Table 2.1 [5]. So, a surgical simulator developed using SOFA can implement a coarse tetrahedral mesh for viscoelastic internal forces, a set of spheres for collision detection and modeling, and a fine triangular mesh for visual rendering.

Figure 2.1 describes a diagram of each one of the processing steps performed by SOFA to run a surgical simulation as well as the algorithms and data structures used. As you can see, each data structure has two different characteristics, the amount of elements and the types of elements, for example volumetric (tetrahedrons) or planar (triangles) elements. The value on each one of these two characteristics defines the amount of computation required by the algorithms associated with

Table 2.1: Different algorithms that can be used in a virtual surgical simulator.

Task	Type of Algorithm	High Level Computation Algorithm	Low Level Computation Algorithm
1	Haptic Rendering	Volume Haptic Rendering	Surface Haptic Rendering
1	Visual Rendering	Texture and Phong Shading	Wireframe
2	Deformation Computation	Finite Elements Method	Mass-Spring Method
3	Collision Detection	OBB (Oriented Bounding Box) Hierarchy Tree	Sphere Hierarchy Tree
3	Topological Changes	Subdivision Method	Destruction Tree

the geometric model. The amount of computation is also defined by the type of algorithm selected to carry out the task.

Besides, Figure 2.1 describes the logic and computation flow of a surgical simulation using SOFA framework. Each step of the process is described next [63]:

1. Step 1: The process begins when the user moves the physical interface of the simulator and produces a variation in the position and orientation of the virtual instruments.

2. Step 2: The simulator checks if there are collisions between the instruments and the anatomical structures. If a collision test is positive, the simulator computes the collision response force.

3. Step 3: The simulator computes the respective deformation caused by the collision between the instruments and the anatomical struc-

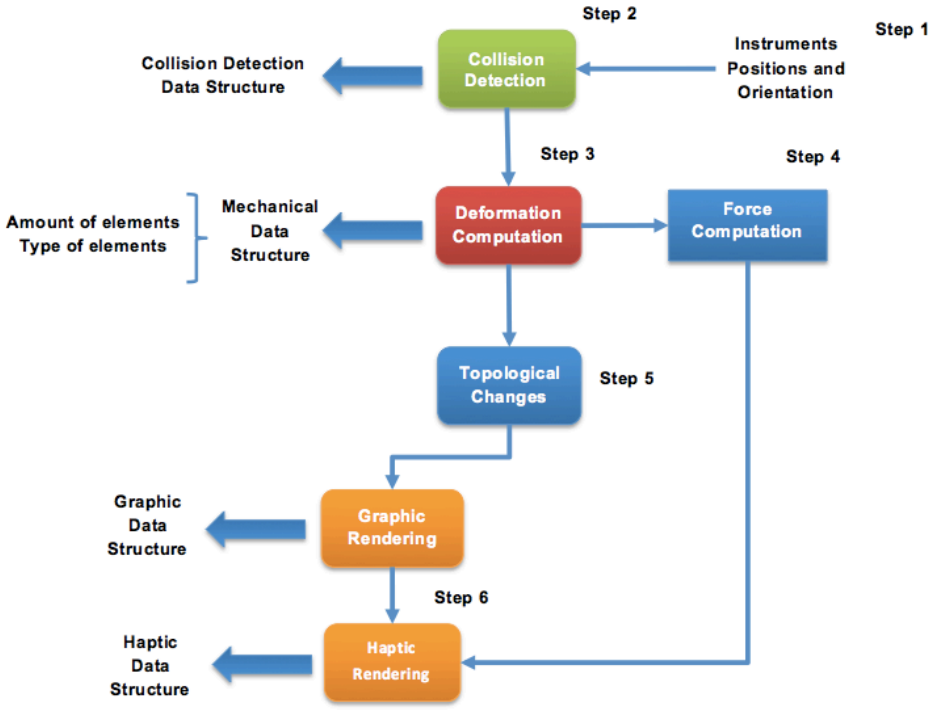


Figure 2.1: Flow of the data and processing steps performed by a Surgical Simulator.

tures.

4. Step 4: The simulator computes the force feedback for the user depending on the computed deformation.

5. Step 5: The simulator checks if the instrument manipulated by the user can produce a topological change in the mesh of the anatomical structure, for example cutting.

6. Step 6: Finally, the visual and haptic rendering is produced by the simulator depending on the force calculated and the state of the instruments and anatomical structures.

Thus, each of the processing steps described in Figure 2.1, and algorithms and data structures shown are implemented by the framework SOFA. Based on advanced software architecture, it allows developers to create complex and evolving medical simulations by using a large set of algorithms and by simply editing an XML file [5].

Additionally, SOFA introduces the concept of a multi-model representation, that allows developers, using methods called mapping, to use different resolution data structures for the deformation modeling, visual and haptic rendering, and collision detection algorithms [31]. SOFA framework groups the simulation components in several packages, here a list of the most important components are included:

- Controller: contains all of the components that permit interaction between surgical instruments and anatomical structures in the simulator using human-computer interface devices, such as haptic devices. The haptic rendering algorithms are included in this package.

- Collision: contains the pipeline and all the algorithms used to detect collisions among virtual models in the simulator. The

most used algorithms are based on hierarchies of spheres, oriented bounding boxes and aligned axis boxes. A detailed description of the component is provided in [63].

- Topology: The algorithms applied for performing topological changes in the data structures used by the simulation are contained by this package. The three most important algorithms in this topic are included such: subdivision, destruction and separation methods.

- Behavior: Algorithms and data structures required for simulating deformable bodies are included in this package. Algorithms like mass – spring, finite element method and mass-tensor are provided by the SOFA framework.

- Visual Model: This package contains all the components required to provide visual feedback of the simulation to the users. It allows to load different types of 3D formats and can be integrated with different 3D engines like OpenGL or Ogre.

- Mapping: different methods used for providing multi-model representation properties between algorithms and data structures are contained in this package. This package will be described in more detail in Chapter 3.

- Constraints: methods used for simulating different kinds of constraints for the mechanical simulation are included in this package. For example, the framework provides a constraint fixed method that allows maintaining fixed a point of the behavior model during a simulation.

Components, part of packages described, have been modified and created in order to implement the network and context-awareness architecture proposed. The components added and modified will be detailed in this chapter and chapter 3.

2.2 CNVSS using a peer-to-peer architecture

The network architecture describes the functional relationship existing between network elements that compose a CNVSS. Client-server and peer-to-peer are the most commonly implemented network architectures, each one providing different advantages and disadvantages for the development of a CNVSS. In peer-to-peer architecture, the surgical simulation environment state is stored and computed locally by each client and user input events are transmitted to other clients in order to update the status of each client's simulation. This architecture is commonly implemented when the collaborative application requires a quick response time and tightly coupled collaborative tasks are not involved. Unfortunately, most of the tasks performed during a surgery by a team demand a tightly coupled collaboration and require highly consistent shared state. By contrast, in client-server architectures, the surgical simulation environment status is stored and computed by a host computer called the server and transmitted to each of the connected clients. A copy of the simulation state is stored at each client but only for viewing purposes. This architecture is able to guarantee a better shared state consistency of the surgical simulation environment, because the simulation is centrally computed by one computer. However, the server can become a bottleneck due to the high computing load as well as the volume of data that needs to be communicated through the server [59]. Additionally, updating the simulation status, when an event occurs on a client side, requires a round-trip to the server. It is worth mentioning that client-server and peer-to-peer refer to terms defined at the network level, and are equivalent to centralized and replicated databases, respectively, defined at the application level.

SOFA does not provide components in order to implement a CNVSS because the framework lacks networking capabilities. For this reason,

the functionality added to SOFA considers the extension of collaborative functionality. This functionality is added first to the framework to determine which and how heterogeneity conditions affect collaboration in CNVSS. Figure 2.2 shows an example of the architecture where two users are collaborating. Each machine runs the simulation and maintains its version of the surgical simulation, the data shared over the network is the position and orientation of the surgical instruments. Considering the numbers appearing in this scheme and showing the execution sequence, the following steps are performed for each cycle of the simulation process, at each peer taking part of the training session:

1. Position and orientation data of the surgical instrument are sent from the module that reads the human-computer interface to the collision detection module.

2. The position and orientation of the local user is shared to remote users using the network connection.

3. In case of a collision between an anatomical structure and a surgical instrument, colliding primitives (triangles, points, or lines) are determined and sent to the surgical operation module.

4. The surgical operation module determines what type of operation is enabled for the surgical instrument controlled by the user (i.e., cutting, carving, attaching, probing, or clip attaching). Based on that, it applies the appropriate algorithms for each operation.

5. Considering the interaction determined by the surgical operation module, deformation is computed to determine the new displacements and positions of each of the points composing the updated anatomical structure.

6. Subsequently, haptic and visual rendering algorithms use the updated anatomical structure to compute the visual and force feed-

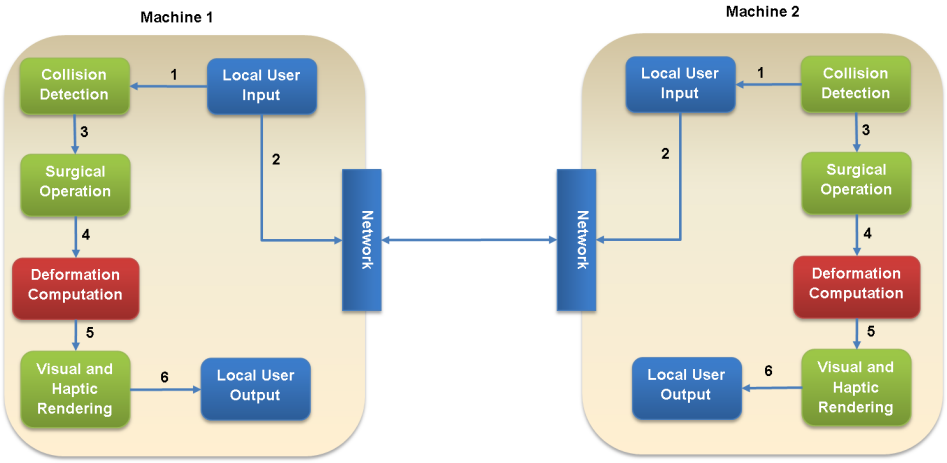


Figure 2.2: Network elements and the functional relationships that compose the peer-to-peer architecture implemented by our CNVSS.

back provided to the user through the local user output module.

The proposed architecture does not implement any strategy to guarantee consistency based on time or information management techniques as described in [23] and [24], because the goal of the experiment is to evaluate how the collaboration of users is affected using a CNVSS implementing the proposed network architecture. Additionally, events occurring in each peer are not synchronized, so that each machine has its own simulation and, possibly, a different data refresh rate.

The components developed to implement the peer-to-peer architecture in the SOFA framework are detailed next. The *OmniDriver* component was modified in the *Controller* package. This component controls the movement of a surgical instrument in the simulation using the Phantom Omni haptic device. The capability to send the state (position, orientation and buttons state) of the haptic device to a remote simulation of the collaborative virtual surgical environment was added. Also in this package, the *RemoteOmniDriver* component, which is a local

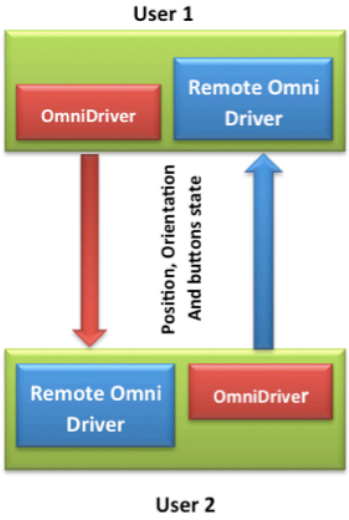


Figure 2.3: Scheme describing how are distributed the *OmniDriver* and *RemoteOmni* driver component when two user are collaborating.

representation of the state of a remote haptic device, was added. For example, as figure 2.3 shows, if two users, $U1$ and $U2$, are collaborating, the *OmniDriver* component located in the $U1$ machine sends the state of the local haptic device using UDP protocol. The *RemoteOmniDriver* component, located in the $U2$ machine, receives the state of the remote haptic device, and binds the state with a graphic representation of the surgical instrument. Neither the *OmniDriver* nor the *RemoteOmniDriver* implement mechanisms to handle network impairments, such as packet loss, jitter or delay.

The *CarvingManager* component was modified in the *Collision* package to allow for cutting and attaching operations using a local or a remote haptic device. This is achieved by binding the *CarvingManager* component to *OmniDriver* or *RemoteOmniDriver* components. The user pressing the first and second buttons of the haptic device performs the cutting and attaching operations, respectively. When the

user presses the cutting button, *CarvingManager* determines whether there is a collision between the graphic representation of the haptic device and an anatomical structure. If there is a collision, using the collision pipeline of the SOFA framework, the *CarvingManager* component determines the primitives colliding with the instrument and, using the topology changing functionality of the framework, executes the cutting operation. On the other hand, when the user presses the attaching button, *CarvingManager* again determines whether there is a collision between the graphic representation of the haptic device and an anatomical structure. If there is a collision, knowing the colliding primitives, the *CarvingManager* component creates an invisible spring between the primitives and the surgical instrument. The force exerted by this spring creates the effect of attaching the structures.

2.2.1 Surgical scenario

In the surgical education field, a procedure that is frequently used as a first step in training is the cholecystectomy [55]. The procedure provides several advantages for developing the skills of a trainee, such as:

- The surgeon needs basic anatomical and physiological knowledge of the anatomical structures that are involved in the procedure.
- The procedure allows for the manipulation of the organs and tissues using several instrument types. It demands the trainee to become familiar with the surgical instrument.
- The workspace in which the trainee moves the instruments is spacious when compared to those used in other procedures.
- The step sequence to carry out the procedure is comparatively simple.

- In the laparoscopic surgery field, this procedure is the most frequently executed and consistently improved upon.

In order to simulate a cholecystectomy, the tridimensional models of the gallbladder, liver, cystic conduct and ligaments joining the gallbladder to the liver are used. The tridimensional models of these structures were created using the method described in [27] and [28]. However, SOFA uses different data structures for visual and haptic rendering, collision detection, and deformable modeling. For visual and haptic rendering, models created by [27] are used. For collision detection, models created by [27] were decimated using the *Blender* open source application [11], to avoid affecting the real-time performance of the simulator. For the deformable modeling, SOFA needs a tetrahedral structure, and by implementing the *MeshTetraStuffing* component included in the SOFA framework, we create a tetrahedral data structure using a triangular mesh as an input. Figure 2.4 shows the various data structures used for the simulation of the gallbladder, and Table 2.2 summarizes the characteristics of the data structures used for each one of the anatomical structures involved in the surgical procedure.

2.2.2 Experimental Setup

An experimental test was developed in order to determine which infrastructrure heterogeneity factors affect collaboration in CNVSS using a peer-to-peer architecture. In the following sections details about the experimental test are provided.

Subjects

Sixteen subjects from 18 to 25 years old took part in the experiment. All subjects selected for the experiment were right-handed and had nor-

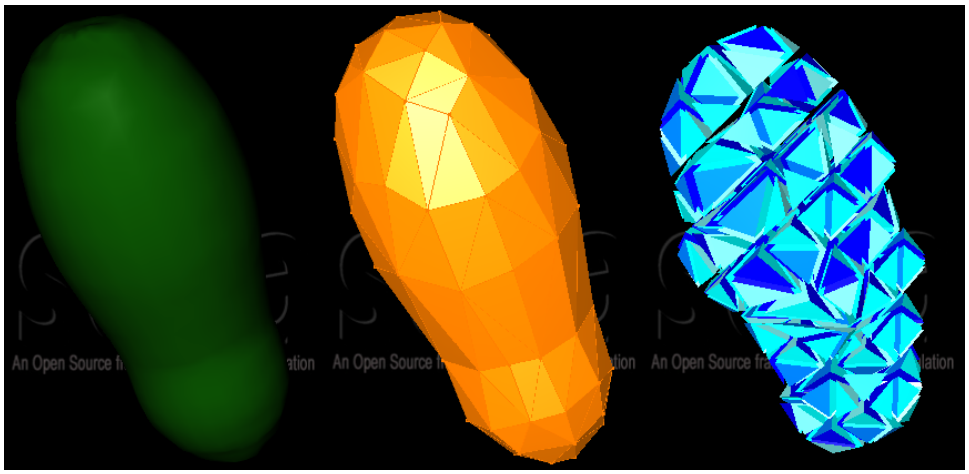


Figure 2.4: Data structures used for the simulation of the gall bladder. Left: visual data structure. Center: collision data structure. Right: deformable modeling data structure.

Table 2.2: Characteristics of the deformable, visual and collision data structures used in the simulation.

Anatomical Structure	Visual*	Collision Detection*	Deformation †
Liver	4384	2054	1382
Gall Bladder	1800	980	1155
Cystic Conduct	NA	NA	279
Ligament	NA	NA	840
Surgical Instruments	416	245	NA

NA: Not Applicable (Some anatomical structures use the same data structure for visual rendering, collision detection and deformation computation) *Number of Polygons. †Number of tetrahedron

mal visual acuity. None of the subjects selected had previous surgical experience. Two teams were randomly assigned from the participants' pool in order to form eight teams. All subjects were uninformed about the task and the purpose of the experiment.

System Configuration

Two workstations were configured to allow for collaboration between two persons in each experimental session. Each workstation consisted of a Phantom Omni haptic device, which allowed users to interact with the surgical environment, a version of the CNVSS running locally in each machine, an XML file describing the surgical scenario, and tridimensional models used for the SOFA framework to load collision, visual, haptic and deformable modeling data structures. The capabilities of each machine are described in Table 2.3. The first and second machines' speed processor, *RAM* capacity and network card speed were changed, but the same graphic card for each treatment was used because its effect was not included in the analysis. Additionally, for each treatment of the experiment, the machines taking part in the collaborative session had the same capabilities.
For this experiment, the computation required in the surgical simulator (deformation, collision detection and topological changes) was only performed in the CPU (serial processing) in order to simplify the experimental analysis. Including different kinds of processing strategies (serial processing in the CPU, parallel processing in the CPU and parallel processing in the GPU) is beyond the scope of this thesis. For example, if parallel processing strategies are included, it is required to not only include the processor speed of the CPU as a factor, but also the number of cores of the CPU.

The workstations were connected using a crossover cable. In order

Table 2.3: Capabilities of the machines used in the experimental test.

Machine	Processor Speed (GHz)	RAM Capacity (Gb)	Graphic Card	Network Card Speed (Mbs)
1	2.66-3.44	1-2	†T1	100-1000
2	2.66-3.44	1-2	T1	100-1000
3	2.8	1	‡T2	1000

†T1: Nvidia GeForce 8800 GTX and ‡T2: Nvidia Quadro FX 3000.

to add network impairments, a machine running the NetDisturb network software emulator was included in the cable path [69], as shown in figure 2.5. NetDisturb allows for control of network conditions such as: network delay, jitter, bandwidth and packet loss. The machine capabilities are described in Table 2.3. The network conditions for each treatment of the experiment are the same from machine 1 to machine 2 and from machine 2 to machine 1. Additionally, direct voice communication was established using headset microphones and speakers.

Collaborative Surgical Task

The collaborative surgical task carried out by each team was a cholecystectomy (i.e., gallbladder removal). In order to achieve this, the surgeon has to cut the two ligaments connecting the gallbladder to the liver, and cut the cystic conduct while it is being stretched with the other instrument. Referring to the surgical virtual environment shown in Figure 2.6, the collaborative surgical task is executed by the users as follows:

1. Each user configures the point of view of the simulation, depending on the task executed by each one.

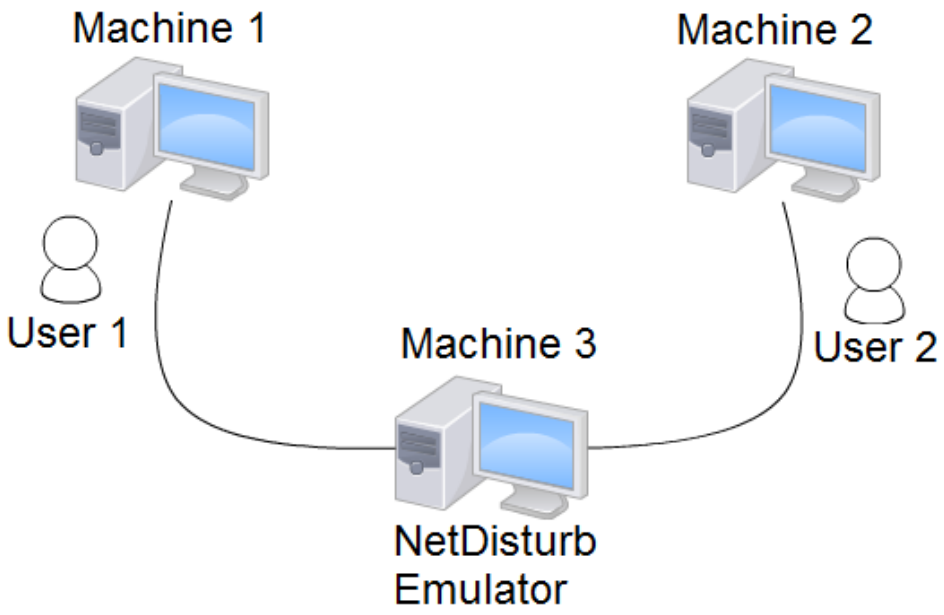


Figure 2.5: System setup used for the experimental test. The users interact with the collaborative surgical simulation using Machine 1 and Machine 2. In Machine 3, the network impairments are simulated using NetDisturb.

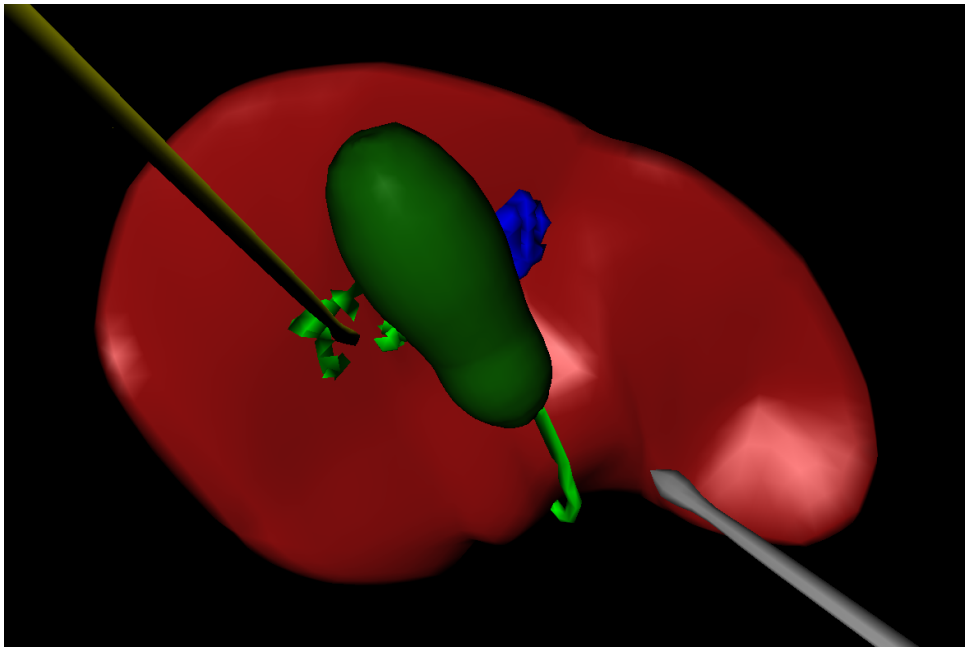


Figure 2.6: Collaborative removal of the gall bladder. The gray instrument is the representation of the local haptic device and the yellow one is the representation of the remote haptic device.

2. User 1 cuts the blue ligament and user 2 cuts the green ligament simultaneously.

3. User 1 cuts the conduct while user 2 attaches his instrument to the conduct and stretches it.

4. Finally, users 1 and 2 remove the gallbladder using the surgical instruments.

Experimental Design

A Fractional Factorial Design of Experiments (DOE) is applied when the number of factors is large and the effect of these factors, over a response variable, is to be determined. This DOE minimizes the number

Table 2.4: Factors considered in the experimental design with their respective lower and higher levels.

Factor	Lower Level	Higher Level
Delay(ms)	0	300
Packet Loss(%)	0	70
Jitter(ms)	0	50
Bandwidth(Mbps)	100	1000
Processor Speed (GHz)	2.6	3.4
RAM Memory (Gb)	1	2
Network Card Speed (Mbps)	100	1000

of experimental runs without losing important features of the problem studied. Therefore, a 2^{7-4} fractional factorial DOE was applied to determine which of the seven factors considered (*network delay*, *jitter*, *bandwidth*, *packet loss*, *processor speed*, *RAM memory capacity* and *network card speed*) affect collaboration in CNVSS [66]. Table 2.4 shows the levels considered for each factor.

For this experiment, a combination of machine and user measurements for evaluating the collaboration were used (See figure 1.2). The metrics considered as reponse variables are the frames per second (*FPS*) of each machine and the task completion time (*TCT*) of the surgical task. The R language, version 2.14.1 R-Project was used for the design of the experiments and the data analysis [78] .

Results and Discussion

Figure 2.7 shows the normal plot, indicating which factors have a major effect over the response variable *FPS*. This figure shows that the most

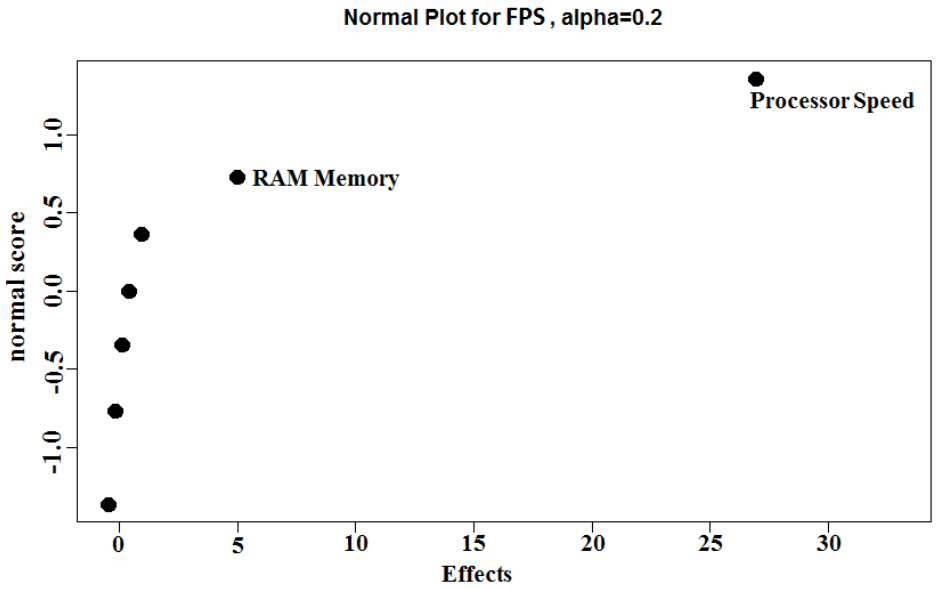


Figure 2.7: Normal plot for frames per second variable.

significant factors affecting the variable *FPS* are *processor speed* and *RAM memory capacity*.

Figure 2.8 shows, in more detail, the effect of *processor speed* and *RAM memory capacity* factors over the response variable *FPS*. The highest increment achieved by the variable *FPS* is present when there is an increment in the *processor speed*. Similarly, but to a lesser magnitude, *FPS* increases incrementally as *RAM memory capacity* increases.

Figure 2.9 shows the normal plot, indicating which factors have a major effect on the variable *TCT*. The most important factors affecting the variable *TCT* are: *delay*, *jitter*, *packet loss percentage* and *processor speed*.

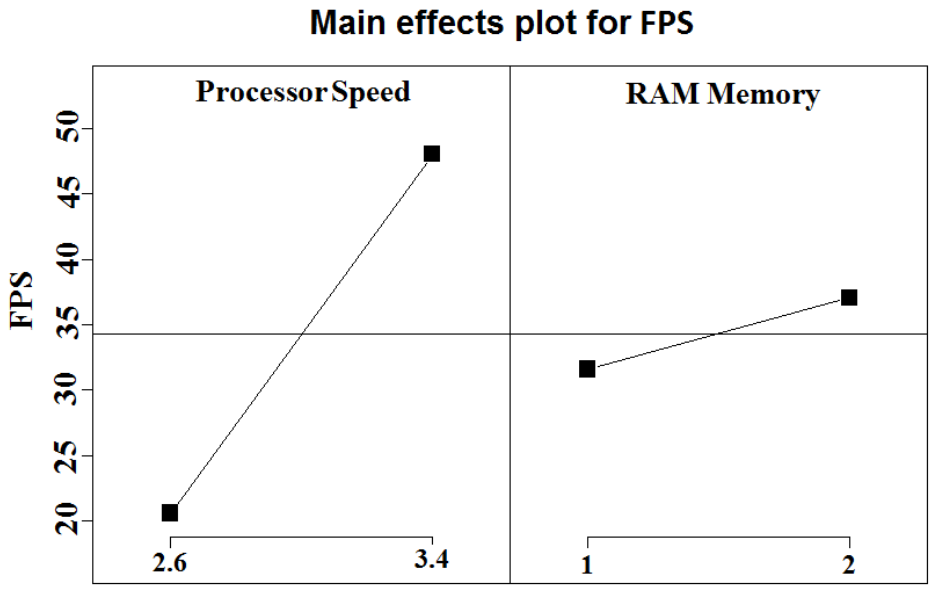


Figure 2.8: Main effects plot for frames per second variable.

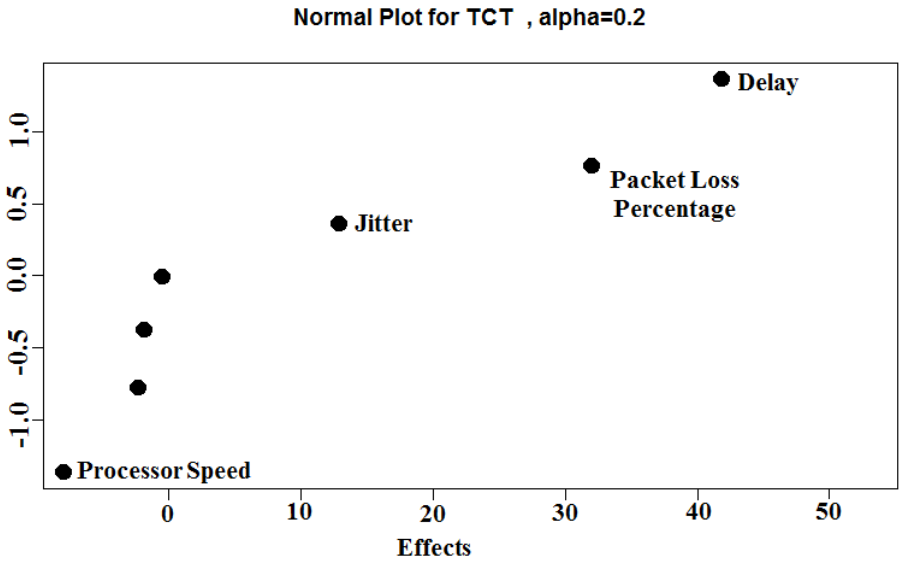


Figure 2.9: Main effects plot for task completion time variable.

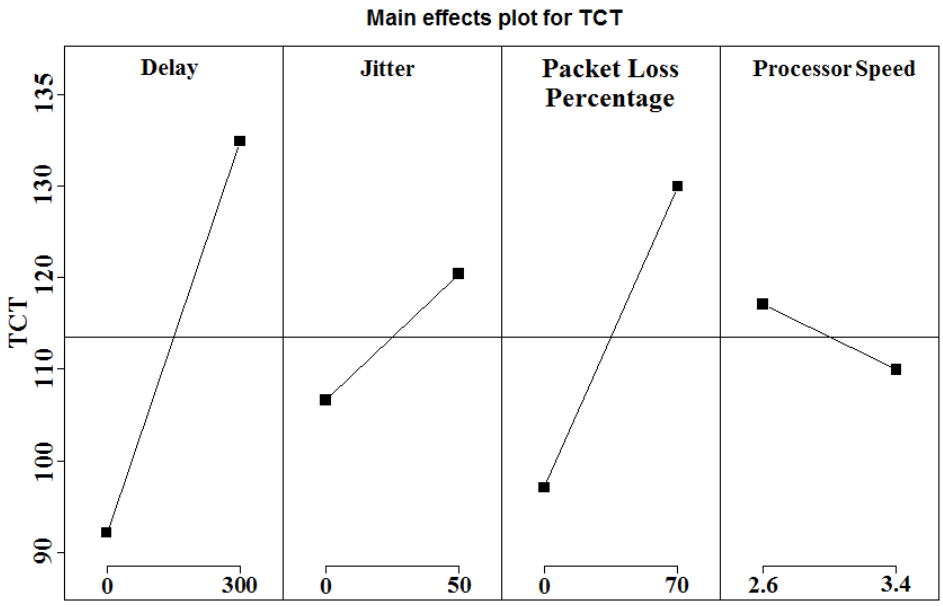


Figure 2.10: Main effects plot for task completion time variable.

Figure 2.10 shows, in more detail, the effect of *delay*, *jitter*, *packet loss percentage* and *processor speed* on the *TCT* variable. The *delay* factor most significantly impacts *TCT*. When the *delay* value increases from 0 *ms* to 300 *ms*, *TCT* increases by approximately 40 seconds. The second largest effect is produced by the *packet loss percentage* factor. When the *percentage packet loss* increases from 0 to 70 %, *TCT* increases by approximately 30 seconds. The third effect is *jitter* factor. An increase of over 10 seconds is observed for *TCT* when the value of *jitter* increases from 0 to 50 *ms*. Finally, a decrease in *TCT* is observed in the presence of an increase in *processor speed*.

From the fractional factorial DOE analysis it was found that the *processor speed* and *RAM memory capacity* have a larger effect over the response variable *FPS*, that is, the *FPS* of the simulation largely

depends on local machine capabilities. Our CNVSS runs locally on each user machine and the performance of the simulation depends on the local capabilities of the user machine. Similarly, the *jitter*, *delay*, *percentage packet loss* and *processor speed* have a considerable effect on the response variable *TCT* and the largest effect on *TCT* is produced by the *delay*, which is consistent with the results in [25], [4] and [44]. So, *TCT* is mainly affected by the network conditions and this is supported by the user comments. The users that execute the task with the higher levels of network conditions agree that step 3 of the surgical task is quite difficult. This is because the execution of step 3 of the surgical task requires tightly-coupled collaboration between the users, and this modality of collaboration only takes place with a high consistency shared state of the CVE [90]. However, the higher level of network conditions evaluated and the peer-to-peer architecture of the developed system fail to guarantee the high consistency shared state of the CVE without implementing mechanisms such as those proposed in [23] and [24].

2.3 CNVSS using a Hybrid Client-Server Architecture

The extension of the framework to provide networking capabilities implementing a peer-to-peer architecture was described in section 2.2, but, as it is concluded peer-to-peer architecture showed serious difficulties to maintain the shared state consistency, specifically, when users perform tightly coupled collaborative tasks and network conditions are not the best. To address this problem, in this section a hybrid client-server architecture, loosely based on that proposed by [52], and the components developed to apply this architecture in the SOFA framework is described. This architecture allows the system to mantain the consistency of the collaborative virtual surgical environment, centralizing the

computation of the surgical simulation on a server, and also preventing the server from becoming a bottleneck by distributing the computational load of collision, visual, and haptic rendering algorithms among each client.

In the proposed hybrid client-server architecture, one host, among those participating in the collaborative session, is selected to act as server and client at the same time, while the others act just as clients. In this context, the server role consists of computing the deformation of anatomical structures and the client role consists of running, locally, the visual rendering, collision detection, and haptic rendering algorithms. Figure 2.11 shows an example of the architecture where two users are collaborating. One of the hosts acts as client/server and the another one acts as a client. Considering the numbers appearing in this scheme and showing the execution sequence, the following steps are performed for each cycle of the simulation process, at the client-server machine:

1. Position and orientation data of the surgical instrument are sent from the module that reads the human-computer interface to the collision detection module.

2. In case of a collision between an anatomical structure and a surgical instrument, colliding primitives (triangles, points, or lines) are determined and sent to the surgical operation module.

3. After that, the surgical operation module determines what type of operation is enabled for the surgical instrument controlled by the user (i.e., cutting, carving, attaching, probing, or clip attaching). Based on that, it applies the appropriate algorithms for each operation and transmits the new state of the anatomical structure to the deformation computation module. In this step, the new state of the anatomical structure (i.e., topological changes or

changes in the forces exerted on the model) is computed. This takes place, for instance, when a probing operation is performed.

4. Considering the new state of the anatomical structure, deformation is computed to determine the new displacements and positions of each of the points composing the updated anatomical structure.

5. Subsequently, haptic and visual rendering algorithms use the updated anatomical structure to compute the visual and force feedback provided to the user through the local user output module.

6. In this step, the simulation database is updated with the following information: (i) topological changes, (ii) position and orientation of the surgical instrument controlled by the user, and (iii) X, Y, and Z coordinates of each point composing the updated anatomical structure.

7. Finally, the state of the simulation database is translated into messages (see Table 2.5) and sent to the client machine. The data sent represents the changes that take place during the simulation.

A similar process is performed on the client machine side, it differs in the following aspects: (i) the deformation computation is not performed locally for the anatomical structures and (ii) the simulation database is updated with colliding primitives, topological changes and position and orientation of the surgical instrument controlled by the user. All this data is transmitted to the client / server machine. The proposed architecture does not implement any strategy to guarantee consistency based on time or information management techniques as described in [23] and [24], because the goal of the experiment described in this document is to evaluate how the collaboration of users is affected using

Table 2.5: Description of the messages composing the application protocol of the CNVSS.

Message Type	Description
*** MSG ATTACHING, MSG CLIPATTACHING and MSG PROBING	Each one of these messages contains the primitives of the anatomical structure and the surgical tool intersecting each other. In order to simulate the phenomenon of attaching a clip or a surgical instrument, a set of virtual springs are used to connect the tool or the clip to the anatomical structure.
MSG CARVING	The algorithm implemented for carving operations use a destruction strategy. The primitives that must be destroyed compose the message.
MSG CUTTING	The algorithm implemented for cutting operations uses a separation strategy. The primitives that must be divided compose the message.
MSG INSTRUMENT	It contains the position and orientation of the instrument controlled by the user.
MSG DEFORMATION	It contains the X, Y and Z coordinates of the points composing the deformed anatomical structure.

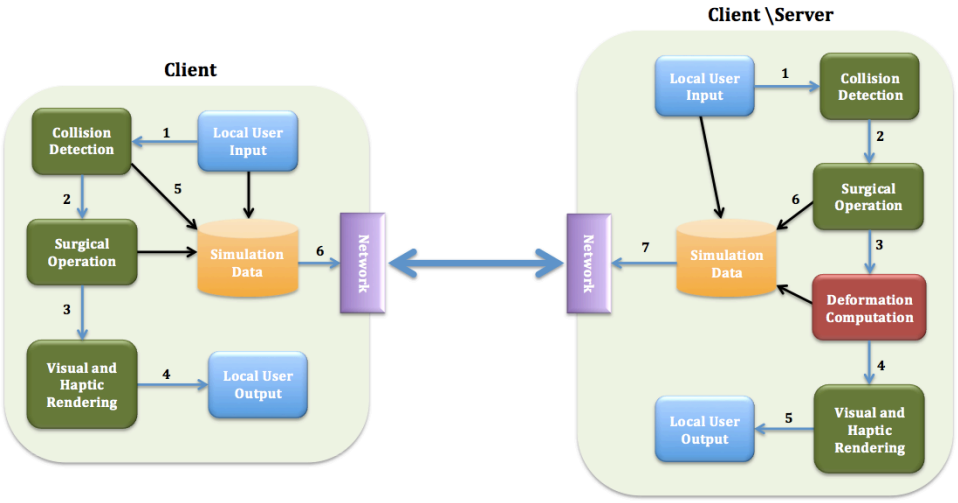


Figure 2.11: Network elements and the functional relationships that compose the hybrid client-server architecture implemented by our CNVSS.

a CNVSS implementing the proposed network architecture. The data shown in the results section will make it possible to evaluate which strategies for maintaining collaboration are appropriate under specific conditions, and propose an adaptive mechanism aimed at maximizing collaboration among users. Additionally, events occurring in the client and client/server machines are not synchronized, so that each machine has its own simulation and, possibly, a different data refresh rate. The data refresh rate of each machine defines the responsiveness provided by the system to each user considering the visual and haptic feedback.

Using the proposed architecture: (i) the computation of surgical simulation is centralized to avoid the divergence of the shared state and (ii) the load computation required by collision detection, as well as by the graphics and haptics rendering is distributed among each user's machine. The components developed to implement the hybrid client-server architecture in the SOFA framework are detailed as follows:

- *TetrahedralCFEMServer* calculates the deformation of the anatomical structures associated using a tetrahedral co-rotational finite elements method (CFEM) and makes the deformation state available to be distributed by *CNVSS-Middleware (MW)* to the clients.

- *TetrahedralCFEMClient* receives the deformation state at the client side, which is computed by the *TetrahedralCFEMServer* component at the server side. When this component receives the deformation state, it communicates it to the local components which perform visual and haptic rendering and collision detection in order to update the corresponding data structures.

- *AttachingControllerClient*, *CarvingControllerClient* and *AttachingClipControllerClient* determine whether a local surgical instrument is colliding with an anatomical structure and if it is, all the information related with the collision and the basic surgical operation performed (attaching, carving, clip attaching, among others) are stored to be sent to the server by the *CNVSS-MW* component.

- *AttachingControllerServer*, *CarvingControllerServer* and *AttachingClipControllerServer* receive all the collision and basic surgical operation data sent by each client and apply them modifying the simulation state at the server side.

- *OmniDriver* and *RemoteOmniDriver* functionality is described in section 2.2.

- *NetworkController* is a very important component in our architecture that runs at the client and server side as an independent thread. Its function is to read the state of the components described above and to determine whether there is an event to be transmitted by *CNVSS-MW* to the clients or to the server.

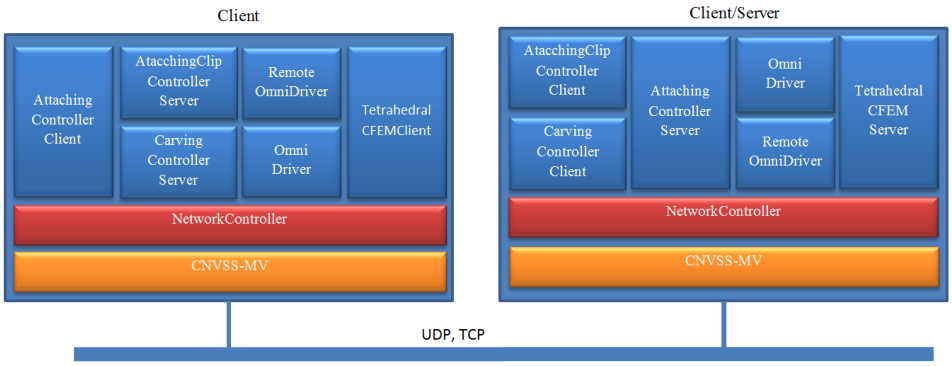


Figure 2.12: Components developed to implement the hybrid client-server architecture.

- *CNVSS-MW* is a middleware layer which provides three main networking capabilities to our CNVSS; (i) organizing the event data in messages that can be transmitted using a specific application level communication protocol developed for this purpose, (ii) defining, depending on the type of message, whether it needs to be sent using UDP (User Datagram Protocol) or TCP (Transmission Control Protocol), (iii) controlliing the connection state and managing the session data between the clients and the server.

Figure 2.12 shows the components developed in order to implement the hybrid client-server arquitecture using the SOFA framework. In this example, two users are collaboratively performing the surgical procedure, in which the client plays the role of attaching and the client/server plays the role of clip attaching and carving the anatomical structures.

2.3.1 Surgical scenario

The surgical procedure proposed for evaluating the heterogeneity factors using a peer-to-peer architecture was used for the hybrid client-server architecture, but this time the appearance and realism of the

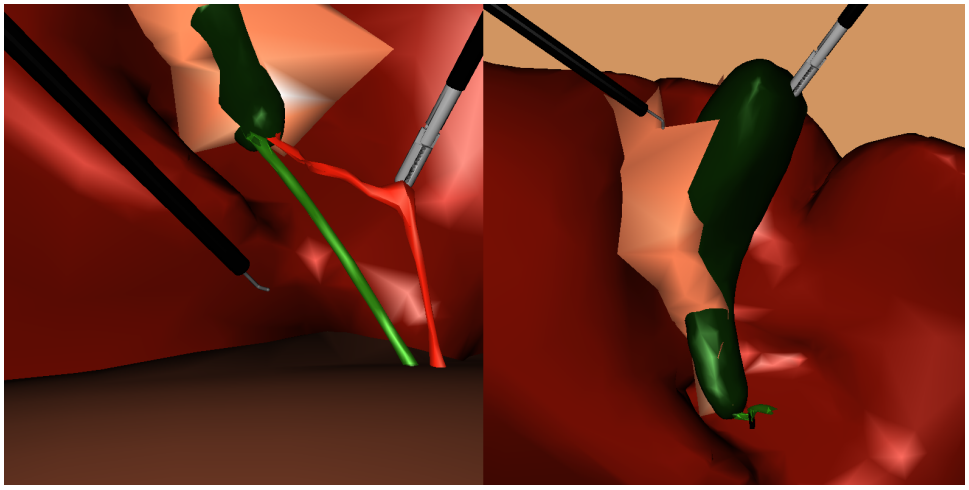


Figure 2.13: Virtual reality cholecystectomy surgical scenario.

virtual environment were improved. The tridimensional models of the gall bladder, liver, cystic duct and artery and structures joining the gallbladder to the liver were provided by Clinical Anatomy Lab affiliated to Stanford University School of Medicine. Additionally, SOFA uses different data structures for visual and haptic rendering, collision detection, and deformable modeling. The process to generate these data structures is described in detail in section 2.2.1. Figure 2.13 shows the surgical scenario created to simulate a cholecystectomy procedure and Table 2.6 summarizes the characteristics of the data structures used for each one of the anatomical structures involved in the surgical procedure.

2.3.2 Experimental Setup

Two experimental tests, a fractional factorial and a central composite DOE, were developed in order to determine which factors and how these factors affect the collaboration in CNVSS implementing the hybrid client-server architecture.

Table 2.6: Characteristics of the deformable, visual and collision data structures used in the simulation.

Anatomical Structure	Visual*	Collision Detection*	Deformation †
Liver	8006	2014	711
Gall Bladder	2040	506	1011
Cystic Duct	NA	NA	334
Cystic Artery	NA	NA	309
Tissues joining the gallbladder to liver	NA	NA	351
Surgical Instrument for attaching	3385	695	NA
Surgical Instrument for carving	976	760	NA

NA: Not Applicable (Some anatomical structures use the same data structure for visual rendering, collision detection and deformation computation) *Number of Polygons. †Number of tetrahedrons

Subjects

Forty-six subjects took part in the two experiments with ages ranging from 19 to 39. 95 % of the subjects selected for experiments were right-handed and the remaining 5 % were left-handed. All the subjects selected had normal visual acuity and none of them had pre-existing surgical experience. Two teams were randomly assigned from the participants pool in order to form eight teams for the fractional factorial DOE and fifteen teams for the central composite DOE. All subjects were uninformed to the task and the purpose of the experiment.

System Configuration

The system configuration described next was used for both experiments and is similar to the system configuration reported in section 2.2.2. Some differences can be observed considering the capabilities of user machines and the network emulator setup. Two workstations were configured to allow collaboration between two persons in each experimental session (*Machine* 1 and *Machine* 2). Each workstation consisted of a Phantom Omni haptic device which allows users to interact with the surgical environment, a version of the CNVSS running locally in each machine, an XML file describing the surgical scenario and tridimensional models used for the SOFA framework to load collision, visual, haptic and deformable modeling data structures. The capabilities of each machine are described in Table 2.7.

Netdisturb was used for controlling the network conditions such as: network delay, jitter, bandwidth and packet loss. However, *NetDisturb* only allows to manipulate three parameters at a time, so another application, developed by us, was used to control the jitter and *Netdisturb* was used to control the other parameters (delay, bandwidth and packet

Table 2.7: Capabilities of the machines used in the experimental test.

Machine	Processor Speed (GHz)	RAM Capacity (GB)	Graphic Card	Network Card Speed (Mbs)
1	2.66	1	†T1	100
2	3.20	3	‡T2	1000
3	1.86	2	†T1	• 1000

Graphic cards used by each machine †T1: Nvidia GeForce 310 and ‡T2: Nvidia GeForce 8800 GTX. • The third machine used two network cards with equal characteristics as is recommended by *Netdisturb*.

loss). Additionally, a direct voice communication was established using Skype application and a headset microphones and speakers. The network impairments were only applied to the network data transmitted by the developed CNVSS.

Collaborative Surgical Task

The collaborative surgical task carried out by each team was similar to that reported in section 2.2.2, but in this case the procedure is more detailed including additional events simulated like clip attaching. During the collaborative task, two roles were performed by the users; user 1 attaches anatomical structures while user 2 cuts and carves the tissues and applies clips to the artery and cystic duct. The role played by each user was defined randomly. Steps involved to perform the collaborative surgical task are detailed as follows:

1. Each user configures the point of view of the simulation, depending on the task executed by each one.
2. Each user moves the surgical instrument to visualize it in the surgical area.

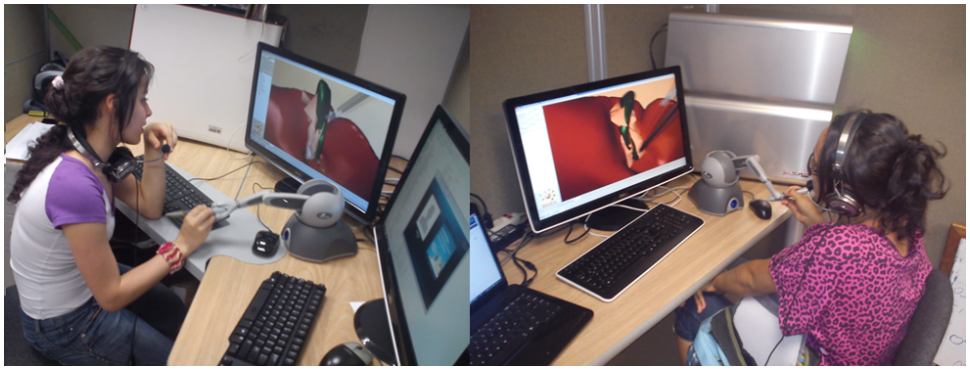


Figure 2.14: Collaborative removal of the gall bladder. Right user handles the attaching instrument and Left one handles the carving and clip attaching instrument.

3. While *user* 1 attaches the artery and separates it from cystic duct, *user* 2 applies clips to each one of the artery and cystic duct extremes.

4. *User* 2 cuts the cystic duct while *user* 1 maintains the artery separated from cystic duct.

5. *User* 1 releases the artery, after which *user* 2 cuts the artery.

6. *User* 1 attaches the gall bladder and stretches it while *user* 2 carves tissues joining the gall bladder to the liver until the gall bladder becomes released.

7. *User* 1 removes the gall bladder.

Figure 2.14 shows two users performing the collaborative surgical task during an experimental run.

Several heterogeneity factors can affect the collaborative performance of the users performing a virtual surgical task such as: infrastructure, user, and virtual surgical scenario properties as can be observed in Figure 1.1. Since the objective of the experiment was to determine

how infrastructure heterogeneity factors affect the collaboration, the users performed: (i) individual and (ii) team training sessions, in order to become proficient in the collaborative task and evaluate only the impact of system conditions. So, each team carried out two training sessions, the first one with the purpose of learning how to handle the human-computer interface (Phantom Omni) and how to perform the role played, carving or attaching and the second one with the purpose of training the collaborative work of the virtual surgical procedure as a team. Each training session stoped when each user and team performed up to seven training sessions and the learning curve becomes almost flat. The training sessions used an ideal system configuration. Before users start the training sessions, a review and discussion, using a video, of the seven steps composing the collaborative cholecystectomy surgical procedure were performed.

Experimental Design

The purpose of this experiment is to determine which and how a group of factors affect the collaboration in a CNVSS implementing the hybrid client-server architecture proposed. In order to know which factors affect the collaboration, a fractional factorial DOE with eight runs was performed to establish which of the factors (delay, jitter, packet loss, bandwidth, server and client benchmark) are the most influential on outputs (consistency, time difference and number of errors) by using a Daniel plot [21]. The parameters whose distribution cannot be considered as normal standard are statistically relevant in the fractional DOE. Therefore, they are considered to affect the collaboration in a CNVSS. In order to determine how the relevant factors affect the collaboration, a surface response central composite DOE with 15 runs was performed afterward with delay, packet loss and bandwidth as studied factors and using the same output variables. Input and output variables are de-

scribed in detail in Table 2.8 and 2.9, respectively.

In order to measure the difference in consistency between the client and server, the clock of each machine was synchronized using the Network Time Protocol (NTP). After that and while the users performed the collaborative surgical task, every 100 ms, the tridimensional positions of the points composing the deformable data structure of the gall bladder were stored in a file with a time label. Then, using the two files (stored in the client and server machines) the average euclidean distance between points was calculated and considered as the measure of inconsistency. Data from the DOE were analyzed with the software for statistical computing R with Fractional Factorial Designs with 2-level factors -FrF2-, Wrapper for Design of Experiments Functionality -DoE.wrapper- and Response Surface Method -rsm- add-on packages [50].

Results and Discussion

2^{6-3} *Fractional Factorial DOE*

Daniel's plot for inconsistency indicates that the most influential factors are *packet loss*, *bandwidth* and *latency* which deviate the most from the normal distribution curve (Figure 2.15a). Studying in more detail the effect of the factors, Figure 2.16a shows that *inconsistency* increases 0.8 cm when *delay* is increased from 0 ms to 300 ms. This result was expected, since the *delay* causes the state of the simulation in the server and the client to differ, in a given time, based on the value of the *delay*. Similarly, it is noted that, when available *bandwidth* is reduced from 100 %, minimum *bandwidth* required by our simulation, to 70 %, there is an increase in the *inconsistency* of more than 0.8 cm. In this case the

Table 2.8: Input variables considered in the experimental design.

Input Variables			
Factor	Description	Lower Level	Higher Level
Delay(ms)	The time it takes for a packet to get from one end (*machine* 1) to the other (*machine* 2).	0	300
Packet Loss Percentage (%)	Percentage of how many data packets are lost during transmission due to congestion, link failures or other problems.	0	50
Jitter(ms)	The statistical variance of the delay	0	50.
Bandwidth Available Percentage (%)	Bandwidth is defined as the amount of data (megabits) that can be transmitted in a fixed amount of time (one second) between *machine* 1 and *machine* 2. So, this input variable is the percentage of amount data that can be transmitted, considering that the bandwdith required by the application is 100 %.	70	100
Server and Client Benchmark (ms)	Time taken by the client or server machine to execute one step of the cholecystectomy simulation.	59.47	34.53

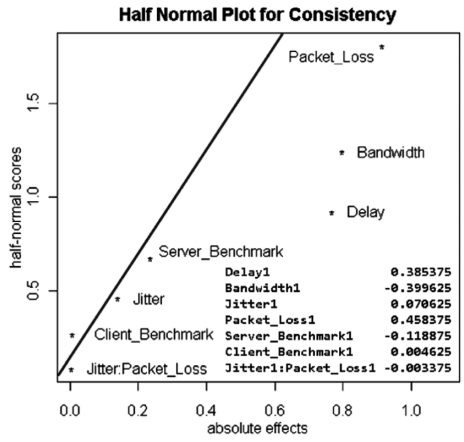
Table 2.9: Output variables considered in the experimental design.

Output variables	
Response variable	Description
Inconsistency (cm)	Average euclidean distance between points of the deformable data structures storage in the server machine and that storage in the client machines. When inconsistency is 0, it means a fully consistent CNVSS.
Difference in Task Completion Time (s)	Difference in time between the time taken by a team to perform the collaborative task in the conditions defined by the experimental run and those taken in ideal conditions.†
Number of Errors	Number of mistakes made by a user. In our simulation the user makes a mistake in two cases: (i) when a user touches a prohibited anatomical structure and (ii) when *user* 2 incorrectly attaches a clip.

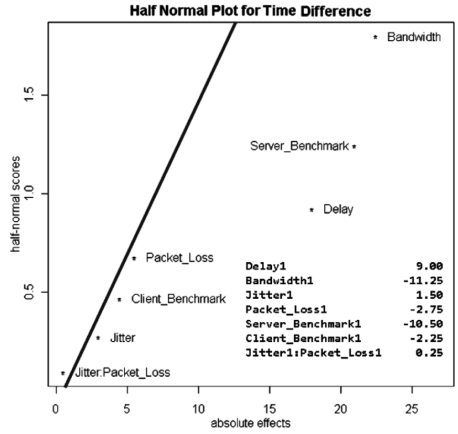
†Ideal conditions are when the benchmark of the user machines are the best and no network impairment is applied.

effect of the *bandwidth* reduction is similar to performing a queuing of the deformation data. This queuing causes the deformation data to be received at the client side after a time interval, increasing *inconsistency*. The influence of *client benchmark* is not noticeable, probably because the amount of computation performed by the client is much less than the server, since the client does not compute the deformation. This causes the client frames per second (fps) to be equal or not much less than server fps, although the *client benchmark* is about half the *server benchmark*. For this reason the client is never going to be flooded by the simulation data it receives from the server, affecting performance on the client side. Finally, when *server benchmark* is increased, there is an increase in the *inconsistency* of more than 0.2 cm. A large *server benchmark* value causes the simulation on both server and client to slow, this does not affect the *inconsistency* significantly because, even though the simulation does not run above an appropriate refresh rate, $30 - 60$ fps, at the client and server side, the state of the simulation at both sides is the same. Finally, packet loss, the most influential factor for this output variable, increases more than 0.9 cm when the percentage of packet loss is increased from 0 % to 50 %.

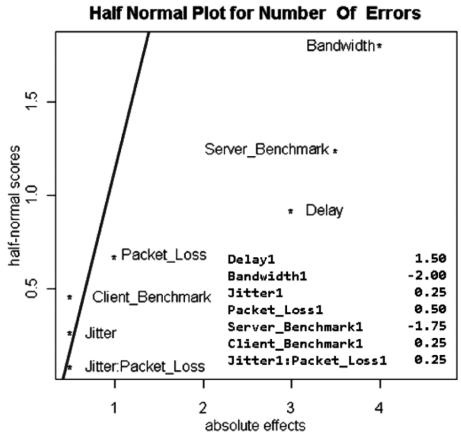
Figures 2.15b and 2.15c show the normal plots, indicating what factors have a major effect over the response variables *time difference* and *number of errors*, respectively. These figures show that the factors affecting both variables the most are *bandwidth*, *server benchmark* and *delay*. Figures 2.16b and 2.16c, show in more detail the influence of studied factors over these response variables. From the above figures it can be observed that when *delay* is increased from 0 ms to 300 ms, *time difference* and *number of errors* increased more than 15 seconds and 3 errors, respectively. This is because an increased *delay* increases the difference between the shared virtual environment simulated on the



(a)



(b)



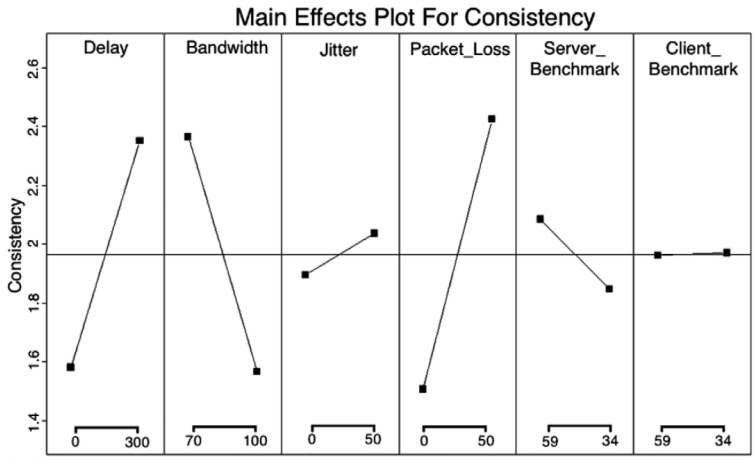
(c)

Figure 2.15: Daniel's Plots for determining the significance of factors

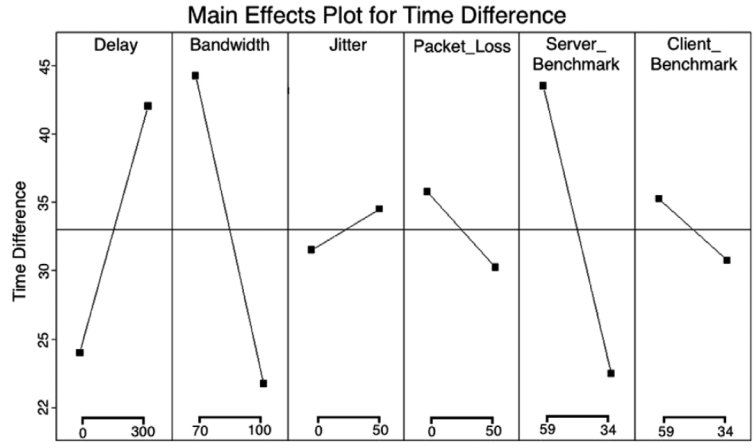
server and the one displayed on the client, causing an asynchrony between the actions performed by users taking part of the collaborative task. For this reason, steps 3, 4 and 6 become the most difficult of the simulated collaborative surgical task. During these task steps the users usually and accidentally collide their instruments with other prohibited structures, increasing the number of errors. A similar effect is observed when *bandwidth* is decreased. As mentioned, a reduction in the *bandwidth* produces a data queuing, which causes data packets from the server to arrive at the client side with a higher *delay*, increasing the difficulty of performing tightly coupled collaborative tasks. Moreover, when *server benchmark* is increased, *time difference* and *number of errors* increases 20 seconds and nearly 3 errors, respectively. This occurs because the *server benchmark* determines the smoothness of deformation computation displayed on both the client and server machines. For this reason, if the *server benchmark* is considerably high, the deformation computation is executed slowly, forcing users to perform their movements slowly, because they try to synchronize with the simulation speed. All this increases the *difference time* and the *number of errors*, even though in this case there are no differences between the simulation displayed on the client and server.

Central Composite DOE

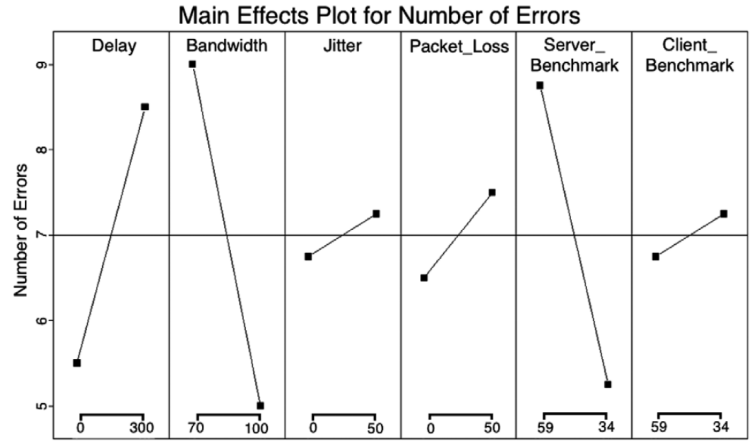
Figures 2.17a, 2.17b and 2.17c show the tridimensional surface of each response variable as a function of two of the factors, taken simultaneously, while keeping the remaining one constant. From figures 2.17b and 2.17c it can be observed that there is an interaction between bandwidth and packet loss. Time difference and the number of errors are decreased, when percentage packet loss increase and there is a reduction in the bandwidth. This may be contrary to what would be expected:



(a)



(b)



(c)

Figure 2.16: Main effects plots for inconsistency, number of errors and time difference response variables

that an increase in the percentage of packet loss, when bandwidth is reduced, deteriorates further collaboration. However, what actually happens is that the packet loss causes the bandwidth required by the application to decrease, and thus the effect of the bandwidth reduction is diminished. Additionally, as can be seen in figures 2.15b and 2.15c, the percentage of packet loss by itself does not have a significant effect on the users' performance.

Similarly, when percentage packet loss is increased and bandwidth is reduced, inconsistency is decreased but in smaller amount. Two causes explain this phenomenon: (i) the method used in *Netdisturb* to simulate packet loss only guarantees that one packet is lost at a time and (ii) the movements performed by users during the collaborative task are slow causing small deformations of the anatomical structures. For these reasons, collaboration is maintained because the geometric shape of the anatomical structures is preserved even though some deformation packets are lost and points of the mesh at the client side are outdated. The phenomenon caused by the interaction between packet loss and bandwidth can be used as a strategy to improve collaboration when the available bandwidth is not enough. For example, the CNVSS can vary the send frequency of the data or the resolution of the deformable meshes transmitted to improve collaboration.

Additionally, when the available bandwidth was less than the required bandwidth and some time of the collaborative task has elapsed, users were able to adapt their performance to the unsynchronized surgical environment using communication by voice, because audio communication data was not exposed to the network impairments during the experiments. For this reason, the user working on the server machine was able to guide, by voice instructions, the movements of the other

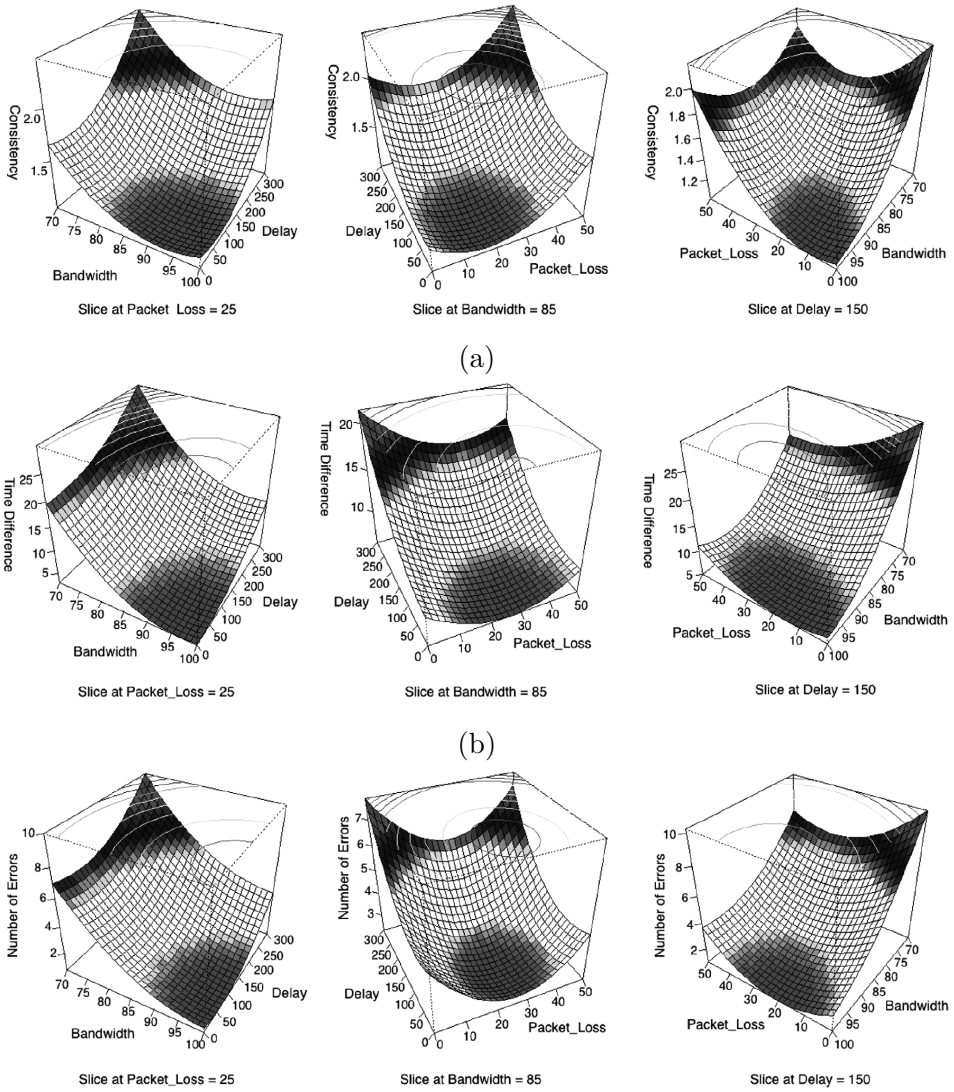


Figure 2.17: Response Surface Plots for central composite DOE

user, diminishing the potential impact on collaboration influenced by bandwidth factor, however, this kind of collaboration is not useful for surgical training.

2.4 Conclusions

The conclusions considering the experiments reported above for each of the architectures proposed are described in this section, finally a general description of the chapter is included.

2.4.1 Using a Peer-to-Peer Architecture

From the analysis of the experiment it can be concluded that *processor speed* and *RAM memory capacity* have a larger effect on the response of variable *FPS*, whereas *jitter*, *delay*, *percentage packet loss* and *processor speed* have a larger effect on the response of variable *TCT*. In brief, machine capabilities mainly affect the local performance of the simulation, and network conditions mainly affect the performance of the collaborative networked system.

Additionally, the SOFA framework was extended to support collaborative training of surgical tasks. The component based architecture allowed for the easy extension of the framework, including components to handle remote surgical instruments, and the remote cutting and attaching of anatomical structures. However, peer-to-peer architecture used for the proposed CNVSS has been found to cause serious difficulties in maintaining the shared state consistency, specifically when users perform tightly-coupled collaborative tasks and network conditions are not the best. For this reason, in the section 2.3.1 of this chapter, a hybrid architecture in which the computation load was distributed between a server and each client was implemented.

Finally, the performance of a computer running an application depends on its hardware components (network card, graphic card, processor and RAM), and its particular architecture, or the way in which these components interact with each other. So it is not suitable to define when a computer is better than another only by using a metric measuring the theoretical performance of each component and not using a metric measuring the real performance of the computer components as a whole. In order to avoid these problems which arise when categorizing the users' machines capabilities with different characteristics, the experiments performed for the hybrid client-server archicture used a benchmarking algorithm to determine which computer is better suited for optimal performance [112].

2.4.2 Using a Hybrid Client-Server Architecture

From the analysis of the experiments it can be concluded that *packet loss*, *bandwidth* and *delay* have a larger effect on the response of variable *inconsistency*, whereas *bandwidth*, *server benchmark*, *delay* and interaction between factors *bandwidth* and *packet loss* have a larger effect on the response of variables *difference time* and *number of errors*.

Finally, the *SOFA* framework was extended to support collaborative training of surgical tasks implementing a hybrid client-server architecture which centralizes the computation of surgical simulation to avoid the divergence of the shared state and distributes the load computation, required by collision detection and graphic and haptic rendering, between each user machine.

2.4.3 General Conclusions

The results obtained for peer-to-peer and hybrid client-server architectures are important because, by knowing which and how various factors

affect the collaboration in CNVSS , it is possible to develop an inference machine able to maintain the collaboration of the users under different network conditions and machine capabilities. Specifically, the results of fractional factorial and central composite DOE play two major roles in the formulation of the inference machine. The first one defines which factors must be expressed in the inference machine as input variables and the second one defines the range of values in which these factors minimize or maximize the performance of the collaboration. This range of values will be used to define the characteristics of the training set used by the inference machine, in order to choose the set of parameters which increase the collaborative performance. Because the shared state of the simulation does not diverge using a hybrid client-server architecture, it will be used for the development of the CNVSS proposed in this dissertation.

The measure of intelligence is the ability to change.

Albert Einstein

3

Description and Implementation of a Context-Aware Architecture for a CNVSS

There are heterogeneity factors that affect collaboration in CNVSS, and grouping them in three categories was proposed in chapter 1: (i) infrastructure, (ii) surgical scenario properties and (iii) user. Additionally, infrastructure factors impacting collaboration the most in CNVSS were reported in chapter 2, including the range in which the impact is negative. Negative means that there are values in which infrastructure factors make the user collaboration inappropriate, i.e. task completion time and number of errors are higher than they should be

for team and individual skills of the users involved in the collaborative task.

For these reasons, an architecture that allows a CNVSS to handle heterogeneity using the context-awareness is described in this chapter, considering all heterogeneity factors involved. A context-aware system adapts its behavior depending on the environment conditions. To carry out this adaptation, these systems are composed by (see [108]): (i) A context-management component, (ii) A context-based inference machine and (iii) An action component. The context management component measures the state of the context, and passes this information to the inference machine which makes decisions in order to adapt the behavior of the system calling the action component. Several research papers have been published describing various strategies for each of these components. In [108], a complete review of the works related to each one of these components is presented.

This chapter is organized as follows. Firstly, the context of a CNVSS is detailed and methods used to measure and manage it are described. Secondly, action or modification mechanisms implemented by the architecture for adapting a specific context are defined. Thirdly, the description and evaluation of an inference mechanism used for determining the adaptation that best fit the context is described in chapter 4. Finally, the implementation of these three components in the CNVSS are described and evaluated.

3.1 Components for Measuring and Managing the Context

In context-aware applications the context means [1]:

"any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between an user and an application, including the user and the application themselves."

If this definition is applied to our problem, three entities that are part of the CNVSS context are defined as: (i) users collaborating, (ii) CNVSS and (iii) the infrastructure required by the CNVSS to be executed. Each one of these entities is associated with a heterogeneity category defined in Chapter 1. For example, user factors characterize the state of users collaborating in the simulator, infrastructure factors determine the state of the hardware on which the CNVSS is executed and surgical scenario properties define the state of the CNVSS. Thereby, heterogeneity factors are considered to be part of the CNVSS context. Entities and its features defining the context of our application are shown in Table 3.1.

Each one of the heterogeneity factors that are part of the CNVSS context are described in detail in the following sections.

3.1.1 Infrastructure Factors

The infrastructure capabilities for running the CNVSS are determined by this kind of factors. They are categorized as network and user machine capabilities.

Machine Capabilities

This factor measures the capabilities of the machine for running appropriately the surgical simulator. These capabilities are determined by machine hardware features, such as: processor type and speed, the

Table 3.1: Entities and features defining the context of the CNVSS.

Entity	**Heterogeneity Factors**	**Units of Measure**
User	Preferences	NA
	Role	NA
	Users Skills	Time (seconds) and Number of Errors
	Team Skills	Time (seconds) and Number of Errors
Infrastructure	Machine Capabilities	Frames per second
	Bandwidth Available	Megabits per second
	Delay	Milliseconds
	Jitter	Milliseconds
	Packet Loss	Percentage
CNVSS	Collision Detection Algorithm	NA
	Deformation Computation Algorithm	NA
	Visual and Haptic Rendering Algorithm	NA
	Collision Model Resolution	NA
	Deformation Model Resolution	NA
	Haptic and Visual Model Resolution	NA

*NA: *Not Applicable*.

speed of the network card, the amount and speed of RAM and features of the graphics card. However, quantifying the performance of a machine, when running a specific application, based only on theoretical parameters of each of its components is difficult [112].In the field of computer performance the use of a benchmark is proposed for measuring machine capabilities [112]. A benchmark is the result of running a computer program, a set of programs, or other operations, in order to assess the relative performance of a machine, normally by running a number of standard tests and trials against it. The use of a benchmark allows to compare the capabilities of different machines and to know which of them are better. There are benchmarks using standard programs to assess machine capabilities, and there are benchmarks proposed to evaluate a specific kind of applications or developed for evaluating a specific application.

The advantages of using benchmarks based on the application are reported in the literature [105]. Then, a benchmark based on the CNVSS application, using standard surgical scenario properties, is proposed for evaluating the machine capabilities of the context-aware architecture. The metric used for measuring the performance is frames per second, which means the number of times per second that the CNVSS executes the process observed in Figure 1.1. The properties of the standard virtual surgical scenario are shown in Table 3.2.

Network Conditions

The factors featuring network conditions are latency, bandwidth, jitter and packet loss. Of these four factors, the most difficult to measure and implement is the calculation of bandwidth. Several approaches have been proposed to measure or quantify the bandwidth of a network con-

Table 3.2: Characteristics of the deformable, visual and collision data structures used in the standard virtual surgical scenario.

Anatomical Structure	**Visual Model Alg.**	**Res.***	**Collision Model Alg.**	**Res.***	**Deformation Model Alg.**	**Res. †**
Liver	SS	8006	AABBH	4082	TMM	2337
Gall Bladder	SS	5100	AABBH	1470	TMM	1763
Cystic Duct	SS	NA	AABBH	NA	FEM	574
Cystic Artery	SS	NA	AABBH	NA	FEM	539
Ligaments joining liver and gall bladder	SS	NA	AABBH	NA	FEM	1600
Instruments	SS	NA	AABBH	1390	NA	NA

NA: Not Applicable (Some anatomical structures use the same data structure for visual rendering, collision detection and deformation computation) *Number of Polygons. †Number of tetrahedra. AABB: Axis-Aligned Bounding Box Hierarchy. SS: Smooth Shading. TMM: Tensor Mass Method. FEM: Finite Elements Method. In the table Res. is an abbreviation of Resolution and Alg. is an abbreviation of Algorithm.

nection, but most of them overflow it for performing the measure [94]. If it happens, the collaboration of the users in a CNVSS will not be appropriately, because the bandwidth measure must be done when the users are collaborating during a training session. When a method for calculating the bandwidth available on a network connection is being chosen, three properties must be considered [35]: (i) the accuracy, (ii) the speed and (iii) intrusiveness of the calculation. In [35], a comparison of different methods was reported using these three properties, so based on their results, the Pathload [41] approach for measuring bandwidth available is chosen for the context-aware architecture proposed. Moreover, the PING command is used for measuring latency, jitter and packet loss. Analyzing the methods defined for measuring the network conditions and reported by the literature is concluded that the measure of bandwidth available is uncertainty, whereas the other three network parameters are not.

3.1.2 User Factors

These factors allow identifying or characterizing the users considering their role in the training session, preferences involving various aspects of the interaction with the CNVSS and their individual and team skills. Below each one of these heterogeneity factors are described in detail, including the method used for measuring and is mentioned if the factor is considered as part of the context in the architecture proposed.

User Preferences

This factor involves the preferences of the user considering her/his interaction with the CNVSS. Two methods have been proposed for determining the user preferences in the literature when an application is being used [15]: (i) an automatic method where user preferences are inferred from the user behavior and (ii) a manual method in which

the preferences are asked directly to the user. The second one is the method implemented for the context-aware architecture proposed.

Three kinds of preferences are taken into account in the CNVSS: interaction, deformation and visualization. The interaction or collision preference determines which the user likes when is interacting with the anatomical structures of the virtual surgical scenario, and is closely associated with the level of realism that the user wants in their perception of the collision (collision detection) and force feedback (haptic feedback). Visualization preference defines the realism and visual quality wanted by the user, and is related to algorithms and resolution of the models loaded in the surgical scenario. Finally, the deformation preference determines the realism and quality of the deformable model simulation desired by the user. These quality features are associated with the algorithm and resolution models used by simulator for calculating the deformation.

How preferences are measured in the context-aware architecture is based on the proposal made by [101]. In the user interface of the CNVSS a set of sliders are included to allow the user choosing the level of preference (Figure 3.1). When the slider value is zero, it means the user does not have any preference by that feature. When the slider value is one, it means the user have the highest level of preference for that feature.

User Role

The user role in CNVSS is defined in two ways. The first one considers the pedagogical or educational role of users taking part of the collaborative task. In this case, considering the traditional model of surgical training "see one, do one and teach one", the users can play

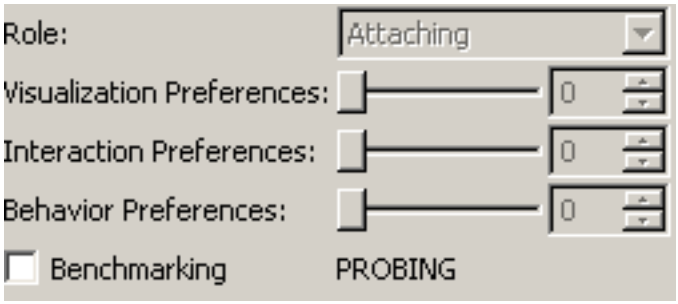


Figure 3.1: Example of the user Interface implemented in the CNVSS for getting the role and preferences from the user.

two roles, as a mentor or as an apprentice [86]. In the second one, the user role is defined by the surgical task performed by the user during the training session. The CNVSS proposed allows the user to perform surgical tasks such as attaching, cutting, carving, probing and attaching clip. The second option was chosen to define the role played by the user during a training session, and a user interface drop-down list is implemented for recording the user role (see Figure 3.1).

User Skills

The skills of the users, in a CNVSS, can be measured individually and as a team. Individual skills mean how the performance of the user is when she/he is performing an individual task like cutting a tissue, applying a surgical clip, among others. Moreover, team skills mean how the performance of the users is when they are performing a surgical procedure as a team, such as cholecystectomy, whereas each user plays a specific role, such as one grasps the anatomical structures and the other cauterizes and applies surgical clips. Team skills are not only involved with the individual skills of each user playing the role it was defined, but also depends on communication skills and synchronization achieved by the users [8] [60]. A common method has been proposed in the literature for measuring individual and team skills [12] [14], in

which a user or a group of users perform a benchmark task whereas the task completion time and numbers of errors is measured. The benchmark task is defined depending on the skills you want to measure, for example if you want to measure the ability of the user for cauterizing, a surgical scenario where the user must cauterize several tissues can be proposed. When the user is novice performing the task, as he/she increases the number of times has trained, the task completion time and the number of errors should decrease. A learning curve is graphed using the data of the task completion time of each trial performed by the user [104]. A learning curve is divided in three important steps (Figure 3.2): (i) the slow beginning is characterized by the trials taken by the user for familiarizing with the task, (ii) steep acceleration includes the process of incremental acquisition of skills by the user, and (iii) plateu means that the user has achieved his maximum performance by executing the task.

3.1.3 CNVSS Factors

These factors allow defining the properties of the virtual surgical scenario being simulated and the mechanisms that will be used to decrease the negative impact of the network conditions. Virtual surgical scenario properties are defined as the algorithms and resolution of the models used for each one of the simulation processing steps, and model resolution is considered as the number of elements having a model used by an algorithm in the simulator. Depending on the type of model, a particular type of element is used. For example, in collision detection process the element commonly used is the triangle, whereas in deformation computation a tetrahedral representation is used. Examples of liver models for collision detection and deformation computation are shown in figure 3.3.

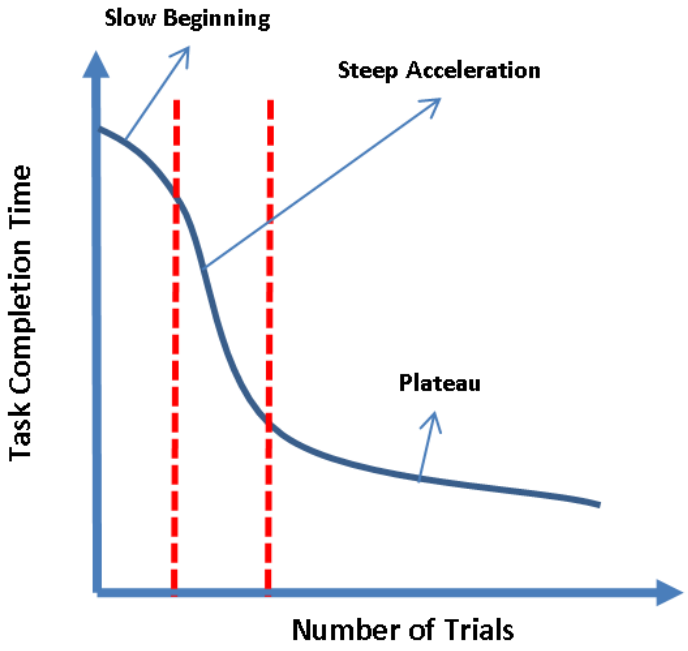


Figure 3.2: Graphical representation of the three steps of a learning curve.

The most important simulation steps in a CNVSS are: collision detection, deformation computation and visual and haptic rendering. The frames per second of a CNVSS depend largely on the algorithms and resolution model chosen for each one of the processing steps. Usually, the more computational cost has an algorithm or model has a higher resolution, the realism provided by the simulation is higher, but the performance penalty of the simulation will be larger, decreasing the frames per second achieved by the simulator [32]. Thus, a trade-off between performance simulation and realism is part of a CNVSS. Different algorithms used for each processing step of the simulation are compared in Table 3.3, considering its accuracy and computational complexity [63] [48] [95].

3.2 Component for action and modification of the CNVSS

The main goal of this component, considering the context-aware architecture, is to modify the CNVSS behavior for better fitting the current context conditions described in section 3.1. The behavior modification is determined by an inference component that will be described in detail in chapter 4. Three modification mechanisms are proposed for each one of the infrastructure factors, reported in Chapter 2, that impact user collaboration the most using a hybrid client-server architecture. These factors are machine capabilities, bandwidth and latency.

Before starting the description of each modification mechanism, a process in which they are based called mapping is defined. The SOFA

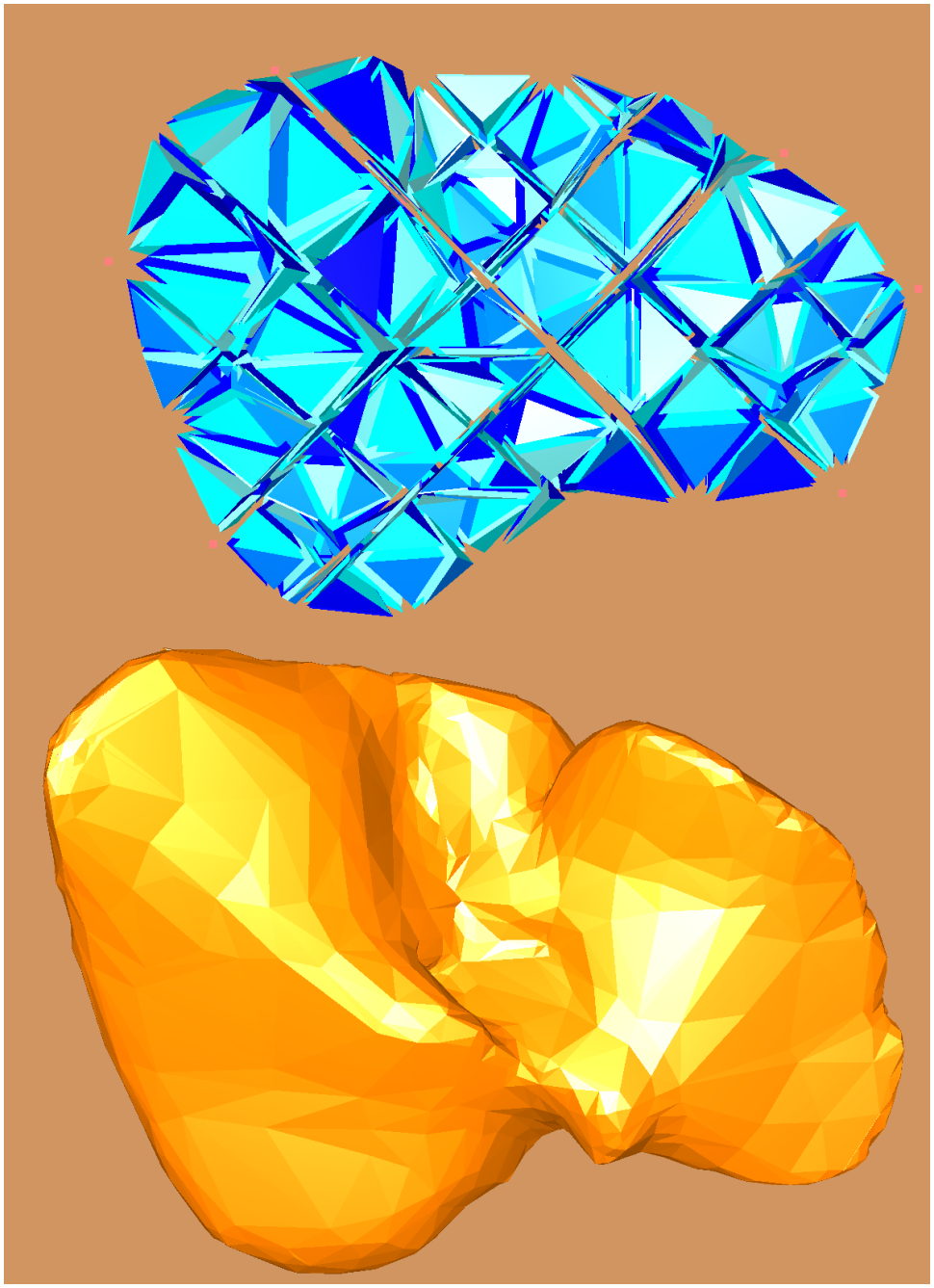


Figure 3.3: Examples of deformation (top) and collision (bottom) models of the Liver.

Table 3.3: Algorithms used in a CNVSS and categorized by Accuracy and Complexity.

	Algorithm	Accuracy	Complexity
Deformation Computation	Finite Element Method	High	High
	Mass Tensor Method	Medium	Medium
	Mass Spring Method	Low	Low
Collision Detection	OBB Hierarchy	High	High
	ABB Hierarchy	Medium	Medium
	Sphere Hierarchy	Low	Low
Visual Rendering	Smooth Shading and Texture	High	High
	Smooth Shading	Medium	Medium
	Flat Shading	Low	Low

ABB: Aligned Bounding Box. OBB: Oriented Bounding Box.

framework includes a process called mapping that allows the CNVSS to use different models with different resolutions for each processing step of the simulator such as collision detection, deformation calculation and haptic and visual rendering. The mapping process determines a relationship between a variable of an input model and a variable of an output model. Variables can be spatial positions, velocities, accelerations, forces, among others. The input model usually is known as the master model and the output model as the slave model. For example, if there are two models, one master and one slave, with variables representing a point set of each model, and points of the master model were displaced, the mapping process can be used for replicating the master's displacement to the slave model, even though if the number of points of the master and slave models differ. A detailed description of the mapping process implemented by SOFA is described in Annex 1. Different mapping processes are applied by SOFA in each simulation step for using multi-resolution models, next the most important are described [32]:

1. Mapping between collision model and deformation model: if there is a collision between the surgical instrument and anatomical structure, the contact points determined by SOFA in the collision model of the anatomical structure are mapped to the deformation model, for computing the deformation caused by the interaction. In this case the master model is the collision model and the slave model is the model of deformation.

2. Mapping between deformation model and collision model: when deformation is computed, a mapping between deformation and collision model is performed in order to update the positions of the collision model. In this case the master model is the deformation model and the slave model is the collision model.

3. Mapping between deformation model and visual model: the new

state of the deformation must be shown to the user, so a mapping between deformation and visual model is done. The deformation model is the master and the visual model is the slave.

Besides mapping between the positions of different models, mapping for other kind of variables like velocities and forces are performed [32]. For example, force mappings are used to provide haptic feedback to the user.

3.2.1 Bandwidth Modification Mechanism

Most of the bandwidth required by the CNVSS, is consumed by the network messages used for sharing the deformation state of the organs and tissues of the virtual surgical scenario (Table 2.5). In the proposed hybrid client-server architecture, the state of the deformation is computed by the client-server using a resolution that best fits the sever machine capabilities (described in detail in section 2.3), and transmitted to the client for its local interaction and visualization purposes. Thus, most of the bandwidth required by the CNVSS depends on the resolution and transmission frequency of the deformation model sent between the client-server and the client. The bandwidth modification mechanism takes advantage of this, so applying the mapping method it allows changing the resolution of the model transmitted decreasing the bandwidth required by the CNVSS. In the mapping proposed, the deformation model computed at the client-server side is the master model and the deformation model storage at the client is the slave model. The mapping algorithm proposed for modifying the bandwidth required is described in Algorithm 1.

Data: T^M : Tetrahedra set for the master model.

Result: Bc_k: Barycenter of the kth tetrahedron.

initialization;

for *each tetrahedron* $k \in T^M$ **do**

  Calculate Bc_k;

end

Algorithm 1: Calculation of the barycenter for each tetrahedron of the master model.

The algorithm described in 2 allows mapping each point of the slave model to a tetrahedron of the master models and its barycenter coordinates. The mapping criterion considers that the barycenter of the tetrahedron chosen in the master model is the closest point of the slave model.

Data: P^S: Points set of the slave model.
P_i^S: ith point fo the slave model.
d_k^i: Distance between the ith point of the slave model and the kth barycenter of the tetrahedron of the master model.
Result: Bc_i : Barycenter mapped to the ith point of the slave model and belongs to the closest tetrahedron of the master model.
t_i : Tetrahedron of the master model closest to the ith point of the slave model.
initialization;
for *each point* $i \in P^S$ **do**
    d_{min} = a large number;
    for *each tetrahedron* $k \in T^M$ **do**
        $d_k^i = P_i^S - Bc_k$;
        if $d_k^i < d_{min}$ **then**
            $d_{min} = d_k^i$;
            $Bc_i = Bc_k$;
        end
    end
end

Algorithm 2: Determines the closest tetrahedron of the master model to each point of the slave model, and assign to it the barycenter coordinates.

When the mapping has been calculated including Bc_i and t_i, every time there is a deformation at the server side and the bandwidth available is less than the required by the application, the algorithm 3 is executed in order to map the state of the deformation model from the server side to the client side using a coarser resolution of the model.

Data: $\alpha_i, \beta_i, \gamma_i$: coordinates of the barycenter Bc_i in the master model.
$P_l^{t_i}$: Point number l of a tetrahedron t_i, where $l = 1, 2, 3$ and 4.
Result: P_i^M : ith point of the slave model.
initialization;
for *each point* $i \in P^M$ **do**
  $P_i^M = P_0^{t_i}(1 - \alpha_i - \beta_i - \gamma_i) + P_1^{t_i}\alpha_i + P_2^{t_i}\beta_i + P_3^{t_i}\gamma_i$;
end
Send P^M to the Client;

Algorithm 3: Calculation the mapping between the slave and master models.

An instrument, at the client side, probing a model of the liver can be observed in figure 3.4. The simulation rendered at the client is observed at the left side whereas the simulation rendered at the client-server is observed at the right side of figure 3.4. The model shown at the client side has a coarser resolution than the model shown at the client-server side.

Additionally, the proposed bandwidth modification mechanism is able to decrease the frequency of sending data from the server to the client, so the bandwidth required by the CNVSS is modified changing the resolution and transmission frequency of the deformation model.

3.2.2 Machine Capabilities Modification Mechanism

The machine capabilities required by the CNVSS for running smoothly the virtual surgical simulation depend on the types of algorithms and resolution models chosen as the virtual surgical scenario properties. Thus, as a mechanism for modifying the machine capabilities required, the architecture proposed allows modifying the algorithms and resolution of the models in execution time. However two cases, in which the

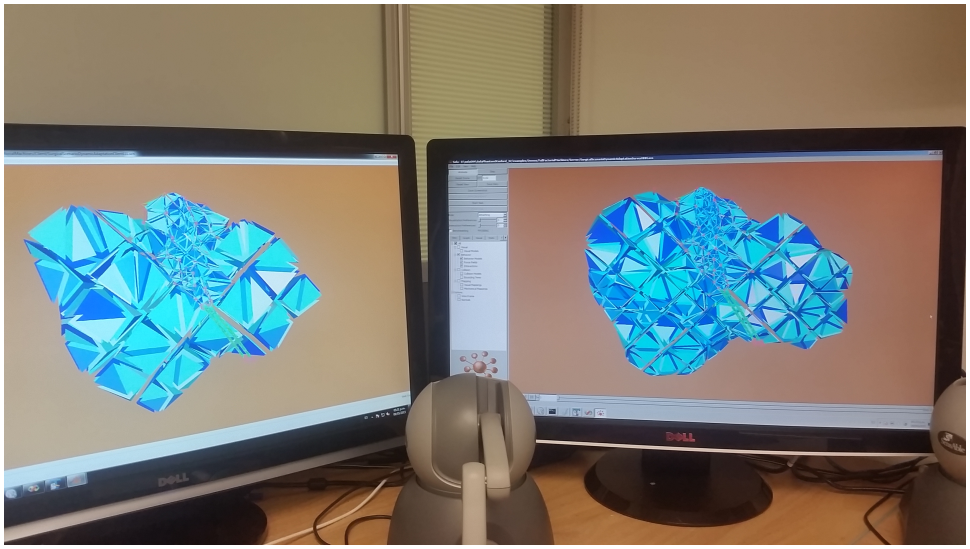


Figure 3.4: Two machines executing the bandwidth modification mechanism.

collaboration or user interaction can be interrupted when the modification mechanism is applied, must be analyzed:

1. *Case 1. If the algorithm or resolution of the deformation model should be modified when the user is attaching or probing an anatomical structure:* Changing an algorithm or resolution of the deformation model when the user is interacting with it, causes the deformation model to experiment instabilities and makes the collaboration among users to be difficult.

2. *Case 2. If the algorithm or resolution of the collision model should be modified when the user is attaching an anatomical structure:* When the user grasps an anatomical structure, the simulator creates virtual springs between the surgical instrument and the object attached to give the appearance that they are adhered each other. If a modification of resolution and algorithm of the collision model is required, the modification mechanism must deter-

mine the new points to create the virtual springs between the new two models, avoiding the new points that are distant to the older ones and changing the grasp perception of the user.

In these both cases a preprocessing step of the models before performing the modification is required for reducing its impact on user's perception and collaboration. A mapping algorithm, similar to that proposed for the bandwidth modification mechanism, is recommended to address the case 1. In this case the master model and the slave model is the deformable model before and after implementing the modification, respectively. Applying a mapping algorithm allows smoothing the transition between different models.

In order to cope with case number 2, a mapping algorithm between the collision model at the client and server side is proposed, where the former is the slave model and the second is the master model. The mapping is done similarly as the algorithm described in section 3.2.1, but in this case triangles are used for calculating the barycenter. So, if there is a collision at the client side and the resolution of the collision models at the client and server has different resolutions, the mapping algorithm is applied for determining the contact points detected at the client in the server. The mapping process described above is detailed in the algorithm number 4.

Data: C^S : Contact points set in the slave model.
Tri^M : Triangle set of the master model.
$M(C_i^S)$: Function mapping the point C_i^S to the triangle Tri^M.
Result: C^M: Contact points set mapped to the master model.
initialization;
for *each point* $i \in C^S$ **do**
    Determine $tri = M(C_i^S)$;
    $P_{closest}$ = Point of tri closest to C_i^S;
    Add $P_{closest}$ to C^M;
end
Send C^M to the server

Algorithm 4: Mapping process performed for the contact points between the instrument and collision model.

An instrument, at the client side, attaching a model of the gallbladder can be observed in figure 3.5. The simulation rendered at the client is observed at the left side whereas the simulation rendered at the client-server is observed at the right side of figure 3.5. The simulator was configured for rendering only the collision model. The model shown at the client side has a coarser resolution than the model shown at the client-server side.

Delay Modification Mechanism

When delay values of the network infrastructure are very high, in ranges that significantly affects the collaboration of users; a delay modification mechanism is proposed for promoting the client computes locally the deformation model. Then, the interaction of the user at the client side is guaranteed because the user does not need to wait for a round trip time of the messages between the client and the server.

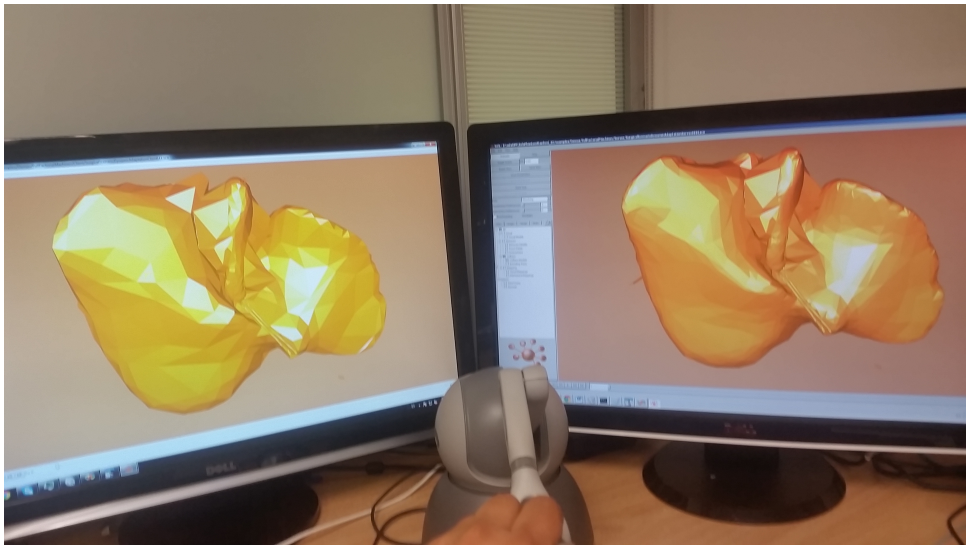


Figure 3.5: Two machines executing the collision mapping for allowing multiresolution models between the client and client/server.

The delay modification mechanism allows the client side calculation of the deformation model in order to guarantee the interaction of the client user with the surgical scenario when delay values are large. However, because the client side deformation just includes the interaction of the client user, the deformation state of the simulation in the client must be combined with the state of the deformation being computed at the server side, including the interaction of the users located remotely. A method which combines the deformation calculated on the client and server sides, using weights for giving priority to each deformation model, is proposed. Equation 3.1 is used for combining both deformation models:

$$X_i^{CC} = \frac{w^C X_i^C + w^S X_i^S}{w^C w^S} \tag{3.1}$$

Where X_i^C y X_i^S are the models representing the state of the deformation at the client and server side, respectively. In this case, $i = 1, ..., n$, where n represents the number of points of object X; X_i^{CC} is the model

representing the combined deformation state of the object X computed at the client and server side, and w^C and w^S are two constants used for weighting the importance of the model computed at the client and server in the combined model. The values used for w^C and w^S must consider the equation 3.2.

$$w^C + w^S = 1 \tag{3.2}$$

For example, if the following expression $w^C > w^S$ is true for the weights chosen, it indicates that the computation at the client side has more impact on the combined model than the deformation computed at the server side.

3.3 Implementation of a context-aware architecture in a CNVSS

The modifications performed in the hybrid client-server architecture, reported in chapter 2, in order to make the CNVSS context-aware are described in this section. Each component of the context-aware architecture is implemented in SOFA, thus avoiding affecting the performance of the simulation. The remaining of the chapter is organized as follows: (i) how the flow of the CNVSS changes for implementing the new architecture is described, (ii) the components developed and modified for extending SOFA are reported and (iii) an experiment for comparing the computational and memory requirements of each architecture, context-aware and non-context-aware, are described.

3.3.1 Processing flow of the CNVSS implementing a Context-Aware architecture

A diagram representation of the processing flow of a CNVSS using a context-aware architecture is shown in figure 3.6. Two users are col-

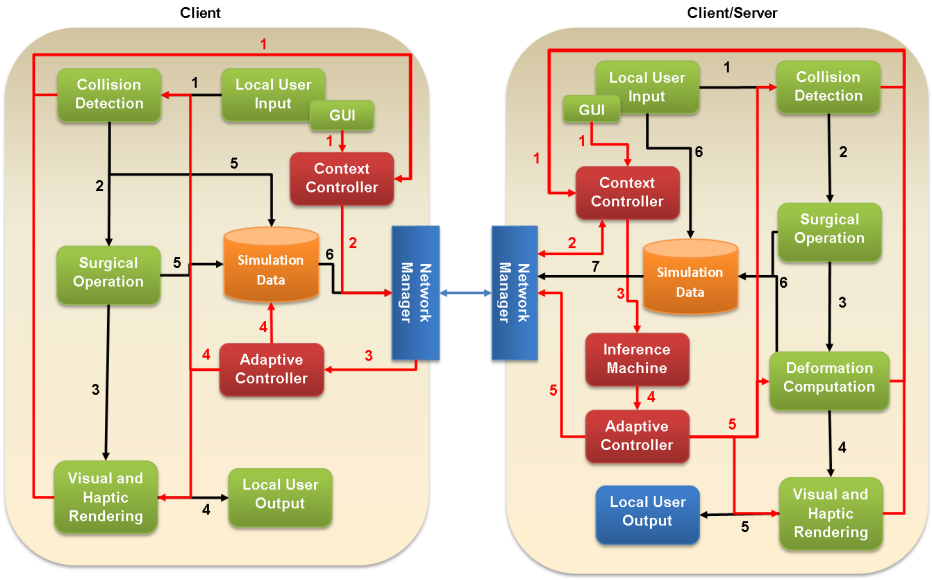


Figure 3.6: Network elements, components and the functional relationships that compose the context-aware architecture implemented by our CNVSS.

laborating in this case, one of the hosts acts as client/server and the other one acts as a client. Figure 3.6 is an extension of the diagram shown in figure 2.11, because the processing flow of the components related with the context-aware architecture are added to the simulation and distribution flow of the hybrid client-server architecture. The simulation and context-aware flows are two independent processes and are executed asynchronously. The red color numbers describe the steps of the context-aware flow and the black ones describe the steps of the simulation flow.

Considering the numbers of red color appearing in this scheme and showing the execution sequence of the context-aware process, the following steps are performed at the client-server machine:

1. The context controller component interacts with simulation and user interfaces components in order to obtain the value of the contextual factors related with the CNVSS (algorithms and resolution models), the user (Role and preferences) and machine capabilities.

2. The context controller component interacts with network manager component in order to determine the network conditions and receives the contextual data from the users located remotely (users and CNVSS factors).

3. The contextual data gathered is transmitted to the inference machine component, which implements the mechanism described in chapter 4, and determines the adaptation mechanisms required for guarantying the collaboration of the user, even though the infrastructure conditions are not the better.

4. The inference machine component transmits to the Adaptive Controller the adaptation mechanisms that must be applied for the current context of the CNVSS.

5. Finally, the adaptive controller component interacts with the components of the simulation in order to apply the mechanisms determined by the inference machine. For example, if the modification must be performed at the server side, such as the bandwidth or client-server capabilities modifications, then the adaptive controller communicates it to simulation local components. If the modifications must be applied for the client machine then the network manager is used for transmitting the modifications to the simulation flow executed remotely.

Considering the client machine, the following steps are executed:

1. Similar to step 1 at the client-server machine, the contextual data is gathered from the user interface and simulation components of the CNVSS.

2. The contextual data gathered is sent to the client-server machine using the network manager component.

3. When the inference process have already performed at the client-server side, and adaptation mechanisms must be applied at the client side, the network manager components of each machine communicate with each other for transmitting the adaptations required. After that, the network component at the client side communicates the adaptation mechanisms to the adaptive controller.

4. Finally, the adaptive controller component communicates with each simulation component in order to apply the local mechanisms, such as machine capabilities modifications or local or remote computation, required for guarantying collaboration.

3.3.2 Components developed for extending SOFA

In order to implement the context-aware architecture using the SOFA framework, it was necessary to create and extend several components, which had to be integrated into the framework. Next the modified simulation components are listed and details of its main modifications or functionalities added are included:

- *MultipleMeshLoader:* Multiresolution models stored in files are loaded by this component. An independent data structure is used in SOFA for storing each model with a different resolution.

- *MultipleMeshTopology:* Different types of topologies, using the 3D models loaded by the component MultipleMeshLoader, are

created by this component. The types of topologies considered by this component are points, lines, triangles and tetrahedral.

- *MultiplePointSetTopologyContainer, MultipleEdgeSetTopologyContainer, MultipleTriangleSetTopologyContainer, MultipleTetrahedronSetTopologyContainer:* The topologies created by the component MultipleMeshTopology are stored in these components for each one of the resolutions loaded.

- *MultipleTetrahedralHybridForceFieldServer:* The bandwidth modification mechanism is implemented in this component. To achieve this goal the component creates several data structures with different resolutions and implements three deformation computation methods mass-spring, tensor-mass and finite elements method. The switch between the resolution and algorithm can be performed in execution time.

- *MultipleTetrahedralHybridForceFieldClient:* The data structures required to store the deformable models transmitted by the client-server machine are implemented by this component. The local computation of the deformable at the client side is computed by this component as well.

- *MultipleFixedConstraintSet and MultipleAttachConstraintSet:* These components allow handling two types of constraints when the deformable model has multi-resolution capabilities. The first type are constraints that fix a point of the deformable model to a space location during all de simulation. The second type are constraints that fix a point of the deformable model to other deformable objects. In order to apply the multi-resolution capability related to constraints components in the surgical scenario properties, the constraints points for each resolution must be defined.

- *MultipleBarycentricMapping and MultipleVisualBarycentricMapping:* The mapping process using barycenter coordinates and multiresolution models are implemented by this component. The first one is used for the mapping between the deformation model and collision model, the second one is used for the mapping between the deformation model and visual model.

- *MultipleOglModel:* The algorithms and resolutions of the visual model are modified in execution time using this component. The visualization algorithms modified are rendering flat and smooth, and using or not using texture.

- *MultipleSphere, MultipleCube and MultipleTriangle:* The algorithms and resolutions applied for the collision resolution model can be modified in execution time using these components.

- *EulerImplicitSolverHybrid:* A Semi-implicit time integrator using backward Euler is implemented by this component. With the component MultipleTetrahedralHybridForceFieldClient it allows for applying the remote and local computation used for the delay modification mechanism.

- *AttachingManagerHybrid, CarvingManagerHybrid and ClipAttachingManagerHybrid:* Different interactions modes between the surgical instrument and the deformable objects are implemented by these components. They are able to perform the interaction, even though the algorithms and resolutions are changed in execution time. The interactions included are attaching, carving and clip attaching.

Several components must be created for implementing the context-aware architecture. Next they are listed and the main functionality is described:

- *NetworkController:* The functionality described for this component in chapter 2 is maintained. New functionality is added for the interaction with context-aware components like ContextController and AdaptiveController. The communication of the context application and modification mechanisms between users machines are done by this component. Finally, the collaboration strategies of the CNVSS are implemented here.

- *ContextController:* The context of the application is monitored by this component. It implements functions for evaluating changes in the context of local and remote machines, including network properties. NetworkMeasuring and RealGUI are components implemented for helping ContextController to measure machines benchmarking, user factors and network conditions.

- *InferenceMachine:* It implements the inference mechanisms detailed in chapter 4. Together with the ContextController and AdaptiveController components, this component forms the core of the proposed context-aware architecture.

- *AdaptiveController:* the modification mechanisms applied by the architecture are managed by this component. It communicates with NetworkController component if the modification must be performed in remote machines or with simulation components if the modifications must be applied locally.

- *MappingNetworked:* The mapping algorithms for the bandwidth modification and the case 2 for machine capabilities modification are implemented by this component.

- *CNVSS-MW:* is a middleware layer including five main networking capabilities, where three of them were described in chapter 2. The other two are: (i) the collaboration strategies are implemented in this component with the help of the NetworkController

and (ii) it verifies if network messages related with the context-aware process are applied correctly in the CNVSS.

3.3.3 Experiment for evaluating the memory and computational requirement of the Context-Aware architecture

The benefits of flexibility and modification provided by the context-aware architecture proposed can result in penalties for the application performance like (i) reducing the frame per second rate achieved by the simulation and (ii) increasing the memory requirements. In order to allow for the modifications to be performed at run time avoiding the impact in the frames per second of the simulation, all data structures and models required by the CNVSS are loaded off-line to have them available in memory during the execution of the simulator.

An experiment was done in order to evaluate the impact on the simulator performance of the context-aware architecture. This architecture was compared with a non-context-aware architecture like the one explained in chapter 2. For each one of the architectures compared, three different surgical scenarios using different algorithms and resolutions were used in the experiment. The properties of each one of the surgical scenarios are listed in the Annex 2. For each architecture and surgical scenario, the frames per second of the simulation and the amount of memory required were recorded.

The three surgical scenarios evaluated in the experiment, using different algorithms and resolutions, are shown in figure 3.7. The complexity of the virtual surgical scenario depends on the complexity of the surgical scenario properties used; they vary from low to high, from left to right.

Figure 3.7: Different surgical scenarios used for the experimental test. The bottom shows the deformation model, the center the collision model and the top the visual model. From left to right the complexity of the resolution and algorithms of the surgical scenario is decreased.

The results obtained for each one of the surgical scenarios are summarized in Table 3.4.

Table 3.4: Results of the experimental test for evaluating the performance of the Context-Aware Architecture.

			Simulation Complexity		
			Low	**Medium**	**High**
Context Aware	**Client**	**FPS**	60.1	59.7	60.2
		Memory (KB)	107.692	108.564	109.092
	Server	**FPS**	33.5	26.9	19.2
		Memory (KB)	129.416	132.016	134.532
Non Context Aware	**Client**	**FPS**	60.3	59.7	60.2
		Memory (KB)	81.875	98.374	102.235
	Server	**FPS**	31.4	25.1	22.2
		Memory (KB)	101.036	117.148	122.528

The results show that the penalty in performance or FPS of the simulator, when implementing a context-aware architecture, is approximately 2 FPS. In contrast, from the client side, there is not a penalty in the simulation performance. Considering the amount of memory used by each architecture, the difference is higher, because the simulator spends memory in order to allow the run time modifications of the algorithms and models. Additionally, there is not an increase in the frames per second of the simulation, when the complexity of the surgical scenario are changed from low to high, and the type of architecture is modified. In contrast, a change in the memory requirements is observed for this case.

3.4 Conclusions

A definition and implementation of a context-aware architecture in a CNVSS for handling the heterogeneity factors and guarantee users collaboration is described in this chapter. Besides, several modification mechanisms were proposed considering the heterogeneity factors that impact the most the collaboration in a CNVSS and reported in Chapter 2. These modification mechanisms take advantage of the characteristics provided by SOFA framework like the mapping process used for handling multiresolution models. The hybrid client-server architecture described in chapter 2 and implemented using the SOFA framework was extended in order to implement the context-aware architecture proposed in this chapter, several components were modified and others were created for implementing the three main components of the architecture. Finally, an experimental test was performed to evaluate the performance penalties, frames per second of the simulation and amount of memory used, when a context-aware and non-context-aware architecture is used in a CNVSS. Since the fluidity of the simulation in a CNVSS, i.e. frames per second, is important for the collaboration performance of the users; the implementation of the context-aware architecture proposed chooses to increase the memory consumption compared to a non-context-aware architecture in order to guarantee the best run time performance of the simulation. So, the decreasing of the frames per second is approximately 2 FPS independent of the scene complexity.

I don't dream at night, I dream all day; I dream for a living.

Steven Spilberg

4

Inference Mechanism for Handling Heterogeneity Factors in a CNVSS

Heterogeneity factors affecting collaboration in CNVSS were described in chapter 1 and 2. Besides, a set of experiments based on statistical design in order to determine which factors impact collaboration the most, was performed. In chapter 3, the concept of context-awareness as a strategy for handling heterogeneity factors in CNVSS was introduced, detailing two main components of the context-aware architecture: the context and the modification mechanism. Also, software components and architectural modifications, required for implementing the concept of context-aware using SOFA framework, were described.

Therefore, an inference model that has been developed for the context-aware architecture proposed in CNVSS is described in this chapter. As far as we know, no similar inference models have been proposed for this purpose. Our hypothesis is that using the context-awareness concept in a CNVSS, including the inference model proposed in this chapter, it is possible to handle heterogeneity factors described in order to guarantee collaboration in a training session held in CNVSS.

This chapter is organized as follows. First, mechanisms for handling heterogeneity in CVE proposed in the state of the art are analyzed. Limitations associated with these approaches and the differences between a CVE and CNVSS, motivating the application of the context-awareness concept and a new inference mechanism, are defined. Second, the inference mechanism proposed is introduced and detailed, and fourth, the results and experimental test for evaluating the performance of the inference mechanism is reported.

4.1 Inference mechanisms for handling heterogeneity in CVEs.

Several works have been proposed in the literature for handling heterogeneity in CVEs [101] [22] [19]. These works are different to each other mainly by: (i) the heterogeneity factors handled, (ii) the inference methods used or based for the mechanism proposed and, (iii) the optimization criterion applied.

Considering heterogeneity factors handled by the proposed mechanisms, most of the work has focused on the management of infrastructure factors. Strategies for handling limitation in machine capabilities on a CVE are proposed in [45]. Moreover, the evolution of the internet for including support in new technologies such as mobile devices have

caused an increase in its heterogeneity, and in the number of works focusing on strategies for handling infrastructure factors related with network conditions, most of them have developed strategies for handling bandwidth and latency [33]. Recently, some studies have included the handling of variables related to the user heterogeneity [7] [92], most of them have used user preferences to solve the trade-off between the quality of the shared data or algorithms used in the CVE and infrastructure and resources available.

Several inference methods for handling heterogeneity in CVEs have been explored. For example, a mathematical model based on linear equations involving performance system and user satisfaction variables was proposed in [101]. A limitation of this method is associated with the requirement of performing a benchmark, before starting the collaboration session, in order to analyze how the infrastructure variables are related to each other. The execution time of the benchmark is dependent on the number of variables and if the system conditions such as bandwidth are changed, the benchmark must be performed again. Later, an approach using heuristics and data structures for determining the system behavior during its executing was proposed in [77], in order to solve the problems caused by the benchmark. However, the approach has limitations to guarantee the collaboration while it stabilizes the system.

Other strategies proposed the implementation of rules using absolute thresholds or mathematical expressions to determine when and how to apply CVE adaptations. The mechanism using rules were implemented for variables like bandwidth, frames per second and other metrics used for measuring machine capabilities [22] [19] [33]. One drawback with methods using thresholds is that the adaptations in the CVE are applied more frequently due to the boolean nature of the rules affecting

user perception and performance in the CVE. Moreover, some studies have proposed the use of QoS policies to handle specific machine and network conditions. These QoS policies are also based on thresholds or mathematical expressions and are defined for improving a specific heterogeneity factor at a time, without considering that all them are related each other. Besides they have associated the same problems involved with thresholds [45] [92] [74]. Other inference methods, like fuzzy logic [9] and neural networks [53], have been proposed to be applied in context-aware based applications, however these applications are not CVE nor CNVSS and the approaches have not addressed the specific characteristics of this kind of applications.

Considering the optimization criterion, most inference mechanisms proposed try to improve indirect or machine measures featuring the infrastructure on which runs the CVE (See Figure 1.1), assuming that improving these measures will improve the collaboration between users [77] [22] [45]. Examples of the indirect measures considered are the frames per second of user machines, bandwidth available, and consistency, among others. Other authors have included user factors, like user satisfaction or perception, as an optimization metric or constraint in the inference mechanisms proposed, establishing a relationship between indirect measurements and user perception [101] [7] [19]. However, no one, as far as we know, has focused on improving and optimizing metrics like task completion time and number of errors, that measure directly the collaboration performance measured in CVE sessions.

Table 4.1 categorize the works proposing mechanisms for handling heterogeneity, considering the three criteria mentioned above.

Analyzing the table 4.1 we can conclude, to the best of our knowledge,

Table 4.1: Related works proposing inference mechanism for handling heterogeneity.

Author	Heterogeneity Factors	Inference Method	Optimization Variable
[101]	User Preferences, Machine Capabilities	Linear Equations	State of the Infrastructure, State of the CVE
[77]	User Preferences, Machine Capabilities	Heuristics	State of the Infrastructure, State of the User
[19]	User Preferences, Machine Capabilities, Network Bandwidth	Thresholds (QoS policies)	State of the Infrastructure
[22]	User Preferences, Machine Capabilities	Rules based on thresholds	State of the Infrastructure
[74]	Bandwidth	Policies for local and remote computing	State of the Infrastructure
[92]	User Preferences, Machine Capabilities, Network Bandwidth	Policies based on mathematical expressions	State of the Infrastructure
[33]	Network Capabilities	Rules based on threshold	State of the Infrastructure
[45]	Machine Capabilities	Policies based on mathematical expressions	State of the Infrastructure
[7]	User Preferences	Ontologies	State of the CVE
[107]	Network Capabilities	Linear Equations and sets theory	State of the CVE
[9]*	User Preferences, Machine Capabilities	Fuzzy Logic	State of the Infrastructure
[85]	Machine Capabilities	Rules	State of the Infrastructure

*Mechanisms proposed for applications different from CVE and CN-VSS.

that none one has proposed the following: (i) a strategy considering all user, infrastructure and scenarios factors described in Chapter 1, (ii) artificial intelligence methods for handing heterogeneity, and (iii) collaboration of users as optimization criterion (in most approaches the state of infrastructure and CVE, and user preferences were the criteria to guarantee or improve).

Moreover, the mechanisms proposed in the literature have addressed heterogeneity factors in CVEs, as far as we know no one has proposed a mechanism for handling heterogeneity in CNVSS, considering the factors described in chapter 1. In addition, CNVSS have characteristics that differ from conventional CVEs, used for testing the mechanisms proposed in the state of art:

- An enrichment simulation environment is required for simulating surgical events like topological changes, object deformation, collision detection, among others.
- Several steps of the surgical training collaborative task requires tightly couple interaction between users, because users must manipulate the same virtual object, each one playing different roles.
- The enrichment simulation environment of the CNVSS demands for high performance requirements in the infrastructure.
- CNVSS have additional heterogeneity factors introduced by the different properties of the surgical scenario.

For all the reasons described in this section, an inference mechanism for handling heterogeneity with the following features is proposed:

- Each one of the heterogeneity factors, i.e. infrastructure, surgical scenario properties and user preferences, must be handled, avoiding limitations such as that experimented by approaches using benchmarks.

- The mechanism must not only guarantee the proper use of the infrastructure resources, but also the good collaboration of the users.
- The uncertainty and imprecision of the measured variables, composing the context of the CNVSS, must be handled.
- The proposed mechanism must allow handling linguistic variables, such as user preferences.
- Special features of the CNVSS, described above, must be considered.
- The proposed approach must benefit from the expert knowledge available from the performed experiments.

4.2 Artificial Intelligence Methods to be used as Inference Mechanisms.

The chosen methodology for the development of the inference mechanism is based on artificial intelligence, because of the complexities related with the problem, the degree of uncertainty of the variables, the dependence of expertise knowledge and non-linearity of the system. The different artificial intelligence methods that will be used in the proposed inference mechanism are described in the next sections.

4.2.1 Fuzzy Logic

Fuzzy logic allows for a proper representation of the human knowledge, which is linguistic and qualitative, through a mathematical language using fuzzy sets theory [109] [110]. The fuzzy sets theory allows for the partial membership of an element in a set, that is, each element has a degree of membership, a value between 0 and 1, in a fuzzy set [109]

[110]. This degree of membership is defined by a membership function associated with the fuzzy set. For example, for each value that can take an input variable X the membership function $\mu_A(X)$ determines the degree of membership of the value of X to the fuzzy set A. Suppose that $X =$ "Speed" then A can assume various linguistic terms such as "Fast", "Average" and "Slow" which are characterized by membership function $\mu_{slow}(X)$, $\mu_{average}(X)$ and $\mu_{fast}(X)$, respectively. This leads to the concept of linguistic variables ("Speed" in this case) whose values (linguistic values) are words or sentences in a natural or synthetic language. Many concepts of classical set theory can be extended to fuzzy sets; others are unique and inherent to the theory of fuzzy sets [46].

Another important concept of the fuzzy logic is fuzzy inference rules. Fuzzy if-then rules or fuzzy conditional statements are expressions of the form IF A THEN B, where A and B are labels of fuzzy sets [111]. Due to their concise form, fuzzy if-then rules are often employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in an environment of uncertainty and imprecision [46]. An example that describes a simple fact is

$$If\ speed\ is\ fast,\ then\ friction\ is\ high. \tag{4.1}$$

Where speed and friction are linguistic variables, fast and high are linguistic values or labels that are characterized by membership functions. Another form of fuzzy if-then rule, proposed by Takagi and Sugeno [97], has fuzzy sets involved only in the premise part. By using Takagi and Sugeno's fuzzy if-then rule, the premise part is a linguistic label characterized by an appropriate membership function. However, the consequent part is described by a non-fuzzy equation of the input variable.

Through the use of linguistic labels and membership functions, a fuzzy if-then rule can easily capture the spirit of a "rule of thumb" used by humans [79]. Fuzzy if-then rules form a core part of the fuzzy inference system (FIS) a fuzzy-rule-based system widely used as a fuzzy controller. Using fuzzy sets, linguistic values, membership functions and fuzzy rules a fuzzy inference system is defined [17]. Fuzzy inference systems are categorized as [49]: type I [103], type II [49] and type III [97], where the consequent part of the fuzzy rule defines the differences among each type.

4.2.2 Artifical Neural Networks

Neural Networks (NN) were initially studied by Rosenbalt [82], who proposed single layer perceptrons, but the limitations that single layer systems had and pessimism felt regarding multilayer systems [64], caused the interest on neural networks to declined for several years. However, during the eighties, the development of new learning algorithms [83] [106] and parallel processing approaches [54] promoted the study of neural networks again.

Neural networks are appropriate for solving problems that have no clearly defined algorithm for mapping an input into an output [46]. To achieve this mapping, a set of representative examples of the desired transformation to train the system is usually employed, which, in turn, is adapted to produce the desired outputs when it evaluates the learned inputs.

A neural network is composed by a set of interconnected computing units called neurons or nodes, which using a non-linear function, called activation function, maps multiple inputs to one output. The activa-

tion function is usually sigmoidal or hyper-tangent; when the function is sigmoidal, the network is called perceptron [82].

Before applying the activation function, input values are multiplied by a weight and added together plus a threshold [46]. Each one of the weights composes the set of parameters that are modified in order to achieve the desired mapping between input and output. The algorithm used to modify the weights in a neural network algorithm is called learning or adaptation rule. In literature the backward propagation error is proposed as a learning algorithm [83]. This algorithm changes the value of the weights in order to minimize the error between the output of the neural network and the desired output.

4.2.3 Adaptive Neuro-fuzzy Inference Systems (ANFIS)

ANFIS systems were proposed, when the authors in [42] identified the limitations of neural networks and fuzzy inference systems, and consider that they have disadvantages that make them complementary. For example, fuzzy inference systems allow for the definition of linguistic variables, membership functions and rules based on expert knowledge. However, there is not a systematic way to transform the expert knowledge in a knowledge base for the fuzzy inference system [79].

Moreover, neural networks do not depend on expert knowledge. Instead, a learning algorithm and a training data set are applied to adapt the network parameters (weights) so that the neural network outputs correspond to the desired outputs of the dataset [46]. However, if expert knowledge about the system is available, there are no mechanisms to modify the parameters of the neural network to include it. Additionally, determining the knowledge learned by the neural network, based on the value of the weights, is not easy. For these reasons, ANFIS

combines features of a fuzzy inference system and a neural network in order to eliminate the disadvantages of each of the systems separately [42].

An ANFIS is composed by a fuzzy inference system of type Sugeno [97], where the parameters of the membership functions in the premise part and the parameters of the linear combination in the consequent part are calculated using a hybrid learning algorithm. A combination between a least-squares and back-propagation gradient descendent methods are used to determine the best values for the parameters. In the forward pass of the learning algorithm, consequent parameters are estimated applying least squares. In the backward pass, the premise parameters are updated by the gradient descent algorithm [43].

4.3 Description of the Inference Mechanism.

The inference mechanism proposed is represented in Figure 4.1 as a black-box system, where input variables are mapped to output variables.

The input variables considered for the proposed inference mechanism are: (i) infrastructure factors that impact the collaboration according to the results obtained in chapter 2, and (ii) user heterogeneity factors, without considering user and team skills. The reason for not taking into account user and team skills, is because the development of methods for measuring automatically the surgical expertise level of an individual or a team are considered outside of the scope of this thesis.

On the other hand, output variables are mainly related with the adaptation strategies, described in chapter 3, and required by the context-

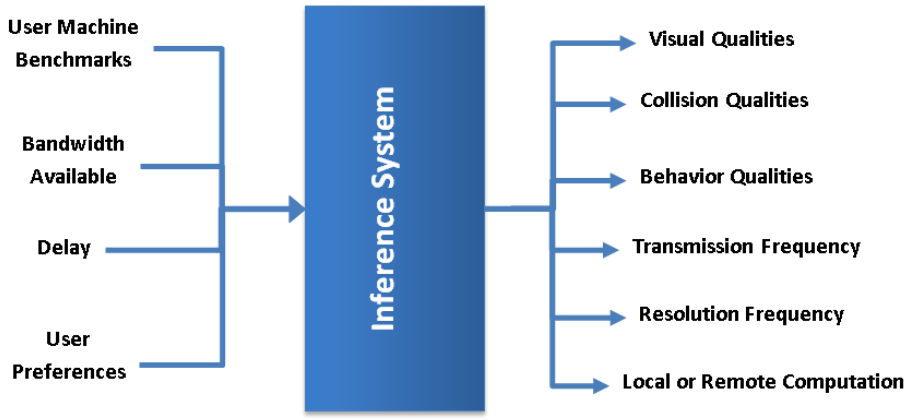


Figure 4.1: Black box model representation of the proposed inference machine.

aware architecture. In this way, the strategy proposed handles infrastructure factors through the modification of surgical scenario properties and mechanisms for managing network conditions, with the goal of ensuring the collaboration of the users training in the CNVSS.

However, a strategy for the inference mechanism that represents the black box model shown in figure 4.1 is a challenging task, because the number of input and output variables is large. Regardless, whether the approach chosen to solve the problem is based on training data (i.e. ANFIS or NN) or an expert system of rules (i.e. FIS), the number of training data or rules required for developing the inference system can reach the thousands depending on the values that input and output variables may have.

Fuzzy inference systems have a limitation called rules explosion, which occurs when the number of fuzzy rules of the system is very large because the number of input and outputs is very large as well [100]. For an expert it is difficult to define a fuzzy system with a large amount of

rules and several mistakes can be made during the development of the knowledge base of the FIS [79].

A strategy called hierarchical design has proved to be successful when the rules explosion problem occurs [16]. This strategy consists on group the input and output variables, so a simple FIS is defined for each group of variables. Then, using the set of simple FIS it is possible to define a hierarchical fuzzy system, where the output of a simple FIS is part of the input of another simple FIS. The development of several simple FIS with small number of input and output variables organized as a hierarchical system allows for a reduction in the number of rules required from the expert, however, a disadvantage of this approach is the difficulty for an expert to find a real sense for the output variables linking each simple FIS [100].

Considering artificial intelligence methods like NN or ANFIS, the biggest challenge is to have a set available data that properly represent the problem. As mentioned above, the number of inputs and outputs associated with the problem, including the values that each variable have, makes the number of needed training samples in the order of thousands. Specifically, in our case, constructing a training dataset requires configuring the CNVSS with different values for each heterogeneity factor, and performing training sessions involving users in order to measure the collaboration obtained for each experimental run. Whether the number of experimental runs is of the order of thousand, creating the dataset will be very expensive in terms of time and effort.

For the above reasons, that contrast the strengths and weaknesses of the artificial intelligence methods considering the problem, an inference mechanism is proposed with the following features:

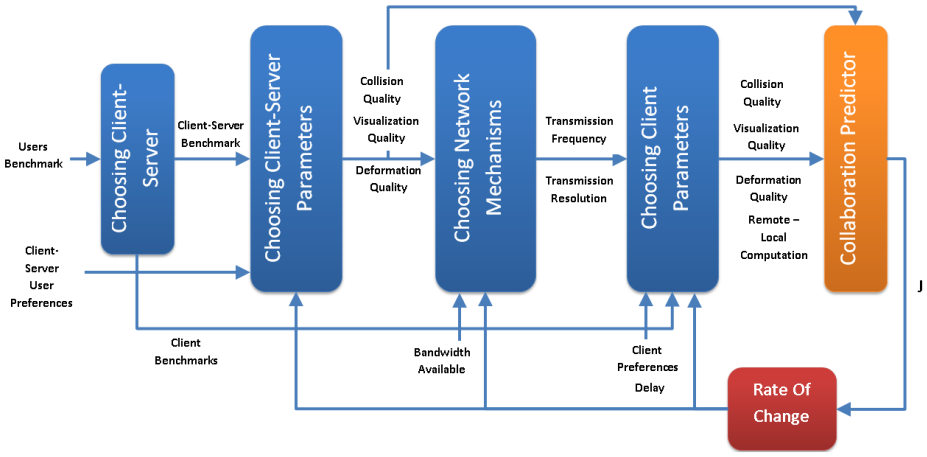


Figure 4.2: Diagram of the proposed inference mechanism.

1. It is composed of subsystems based on artificial intelligence that allows splitting the complexity of the problem.

2. It applies a hybrid approach that uses different artificial intelligence methods, based on knowledge and data-driven, each one the most appropriated for each subsystem.

The inference mechanism including each one of the subsystems is shown in figure 4.2.

The inference mechanism is composed of six subsystems, four of which are based on artificial intelligence methods, and the other two implement algorithms. Each subsystem is intended to determine a subset of output parameters that are part of the complete output required for handling the heterogeneity in a CNVSS. Thus a hybrid hierarchical strategy is implemented, where the problem is divided into different subsystems and artificial intelligence methods are applied to address each subsystem [2]. Below are listed and briefly described each of the

subsystems:

1. Choosing Server: This subsystem chooses among the user machines the most appropriated to play the client-server role in the network architecture proposed.

2. Choosing Server Parameters: Since the machine to play the client-server role was chosen, this subsystem determines the surgical scenario properties to be executed in this machine.

3. Choosing network mechanism: Taking into account the frames per second performance of running on the server and network conditions, the adaptation mechanisms of the network are determined, in this case related to the bandwidth.

4. Choosing client parameters: Once the surgical scenario properties on the server and bandwidth adaptation mechanism have already been defined, the surgical scenario properties of the machine playing the role of client and the strategy to handle delay effects are determined.

5. Collaboration predictor: Considering the value of all current heterogeneity factors that are part of the context and the adaptation mechanisms applied, this subsystem predicts the performance of the collaboration of users, in terms of a metric that depends on the task completion time and number of errors of the collaborative task. Performance prediction considers that users are experts in the collaborative task, both individually and as a team.

6. Reason of Change: this subsystem evaluates whether the predicted collaboration is appropriate, if it is not, restarts the whole process of inference from Choosing Server parameters subsystem.

If collaboration is appropriate, the inference mechanism continuous monitoring until a change in the context of the CNVSS is detected, and then the process starts again.

4.3.1 Choosing Server Subsystem

This subsystem applies an algorithm to determine from machines that will be part of the training session, which of them should be chosen to play the client-server role. The criteria used to choose the most appropriate machine is to select one that has the best computation performance, considering the benchmark value of the machine. The algorithm is based on a variation of the master-slave distributed algorithm used to select a master machine within a group of machines [87]. Details of the algorithm implemented, when the training session is starting, are described in Algorithm 5:

Data: U : Set of users machines that will take part of a training session.
U_i : Defines the ith machine of the user taking part of the training session, where $i = 1, ..., n$ and n is the number of users.
Bch_i: Represents the benchmark of the ith machine.
M_{temp}: User machine chosen by the system as a temporal master.
Result: M: User machine chosen as the master of the session.
initialization On each user machine the following is executed:
Calculate Bch_i Send Bch_i to M_{temp} On the M_{temp} chosen, the following is executed: **for** *each machine* $i \in U$ **do**
  $Bch_{best} = 0$ **if** $Bch_i > Bch_{best}$ **then**
    $Bch_{best} = Bch_i$ $M = U_i$
  end
end
Send M to each machine $i \in U$

Algorithm 5: Algorithm to choose the client-sever machine.

The algorithm described chosen a temporal master based on the

implementation developed by [87]. Since the temporal master have been chosen, each machine calculates its benchmark and sends it to the temporal master. Then, the temporal master runs the algorithm that chooses the machine with the best benchmark and notifies each machine taking part of the training session, the machine chosen as the new master of the session. The new master of the session is chosen for playing the client-sever role of the proposed architecture. The reason for choosing the machine with the best benchmark is because the client-server must execute the simulation of the deformable objects and share it among the clients, and as discussed in previous chapters the computation of the deformation requires machines with high-end performance.

4.3.2 Choosing Server Parameters Subsystem

This subsystem is responsible for choosing the surgical scenario properties running in the client-server machine, considering the benchmark and preferences of the user interacting on this machine. It worth to remember that client-server machine centralizes the state of the simulation in order to maintain the shared state of the surgical scenario, so the deformation is computed and shared by this machine. Therefore, the deformation quality perceived by all users depends on the deformation quality calculated by the client-server machine, except when a client machine locally estimates the deformation.

A fuzzy inference system was developed for implementing this subsystem. The linguistic terms, membership functions and rules of the FIS proposed were based on expert knowledge. The input variables of this subsystem and the FIS proposed are: (i) the benchmark of the client-server machine, measured in frames per second, and (ii) the user preferences described in Chapter 3, which are the interaction, visualiza-

tion and behaviour. Membership functions and linguistic terms chosen for input and output variables can be observed in Figure 4.3. The membership functions used for defining the three qualities and preferences are the same.

Four linguistic terms were proposed to represent the FPS benchmark variable. The type of membership function and the range of values are chosen considering the way frames per second impacts the performance of the users interacting with a virtual environment. For this purpose, the contributions evaluating how FPS impacts the performance of the users in virtual environments and videogames made by [20] were taken into account. Regarding preference variables, three linguistic terms was used, defined as low, medium and high preferences to each topic. In our approach the user chooses between three types of preferences related with the quality of the simulation:

- Visualization: Determines how much the user prefers a good quality on the visualization of the surgical scenario. The quality of the visual rendering algorithms and resolution of the visualization model are too related with this variable.
- Interaction: Determines how much the user prefers a good quality on the interaction of the surgical scenario. The quality of the collision algorithms and resolution of the collision model are too related with this variable.
- Behavior: Determines how much the user prefers good quality on the physical simulation of deformable objects of the surgical scenario. The quality of the deformation algorithms and resolution of the deformation model are too related with this variable.

Similarly, three types of qualities were defined for the simulation: Visualization, Interaction and Deformation Qualities, and they were

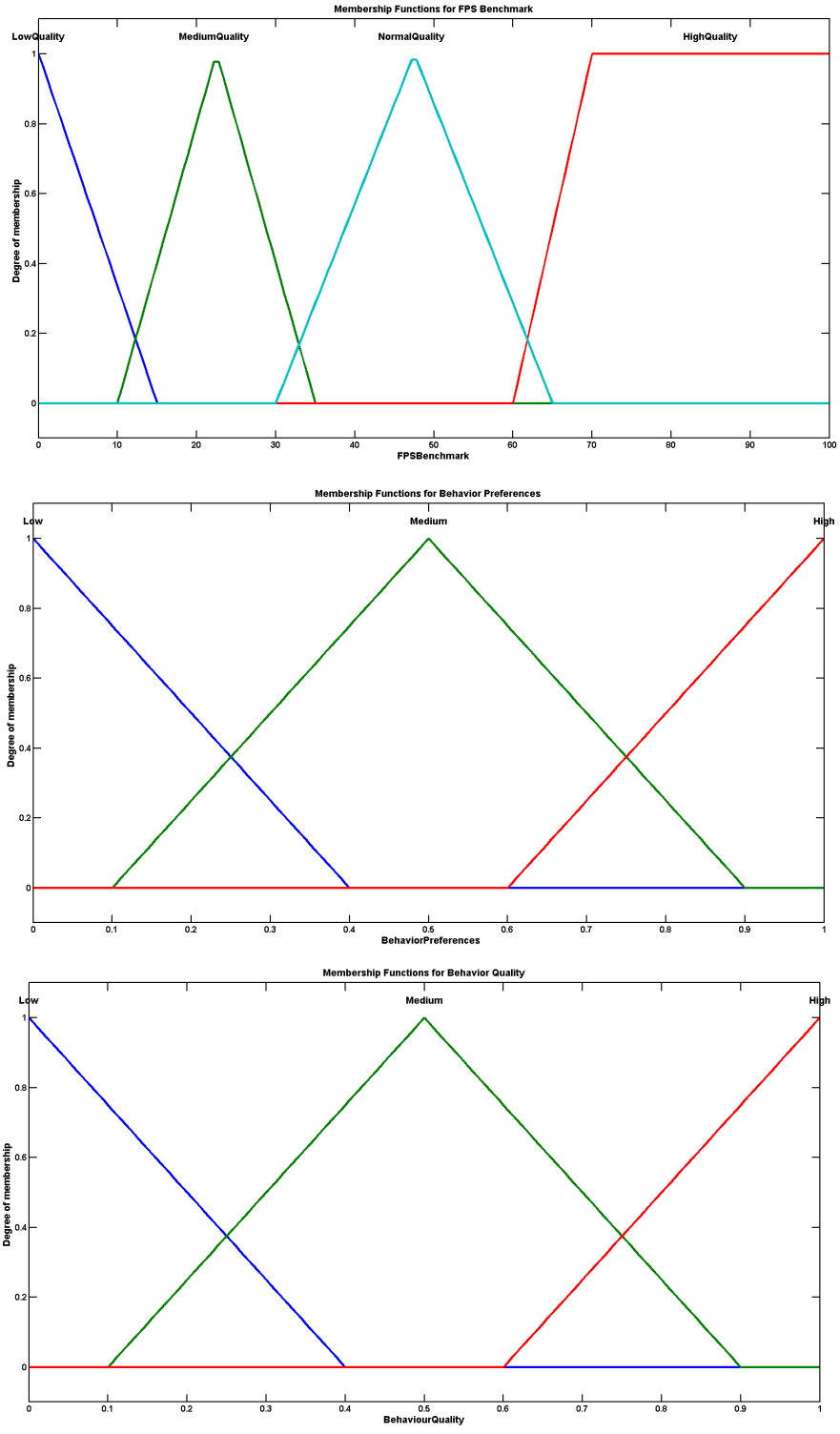


Figure 4.3: Membership functions defined for the Choosing Server Parameters Subsystem.

considered as the output variables. For each one of the qualities three linguistic terms were defined as low, medium and high, where algorithms and resolution of different qualities are used to represent each linguistic term as was shown in figure 4.3.

Using the linguistic terms defined for input and output variables, a set of rules combining them were defined considering expert knowledge. In Annex 3, the fuzzy rules defined by the expert, doing a mapping between the benchmark of the user machine and user's preferences with the qualities of the simulation, are detailed. The fuzzy rules defined by an expert for this subsystem are grouped considering the following objectives and principles:

1. If the quality of the FPS benchmark is low, the objective is to decrease the collision, visual, and behavior qualities in order to increase the FPS or the client-server machine.

2. If the quality of the FPS benchmark is medium, the algorithms of lower computational cost, visual and collision algorithms, are modified to fit the user's preferences, the behavior quality is decreased for improving the client-server FPS benchmark.

3. If the quality of the FPS benchmark is normal, the modification of the simulation qualities are defined by the user's preferences.

4.3.3 Choosing Network Parameters Subsystem

This subsystem defines the bandwidth mechanism appropriate for the current context of the CNVSS. For the development of this subsystem a fuzzy inference system was proposed whose linguistic terms, membership functions and rules were based on expert knowledge. The input variables considered for this subsystem are: (i) the benchmark of the client-server machine, which is measured in frames per second, (ii) the

quality of deformation, visualization and interaction chosen for the surgical scenario executed at the client-sever machine and (iii) the amount of bandwidth available, considering that the bandwith required by the CNVSS running medium qualities for visualization, collision and deformation are the minimum bandwidth required. Similarly, the output variables are the transmission frequency and resolution of the model deformation. This adaptation mechanism is described in chapter 3.

The membership functions defined for bandwidth variable are observed in Figure 4.4. Considering the other variables that are part of this inference subsystem, their linguistic values and membership functions are the same as defined in the previous section.

To define the ranges and the membership functions for the bandwidth variable was taken into account the results of the experimental test described in chapter 2. Considering that a small limitation of bandwidth causes a great deterioration of the collaboration, the membership functions and ranges defined are very close to 100 % of the available bandwidth, as can be seen in Figure 4.4. Bandwidth with 15 % below the amount required by the CNVSS is considered a low bandwidth.
For this reason, linguistic terms defined as low, medium and high are used, but they are distributed in a range of values similar to the used in the experiment described in Chapter 2. The fuzzy rules defined by the expert and doing a mapping between the benchmark client-server machine, the quality of the algorithms and models of deformation, collision and visualization chosen on the client-server machine and network parameters with the adaptation mechanisms for bandwidth and delay are detailed in Annex 3.

The reason why the benchmark of the machine client-server and the

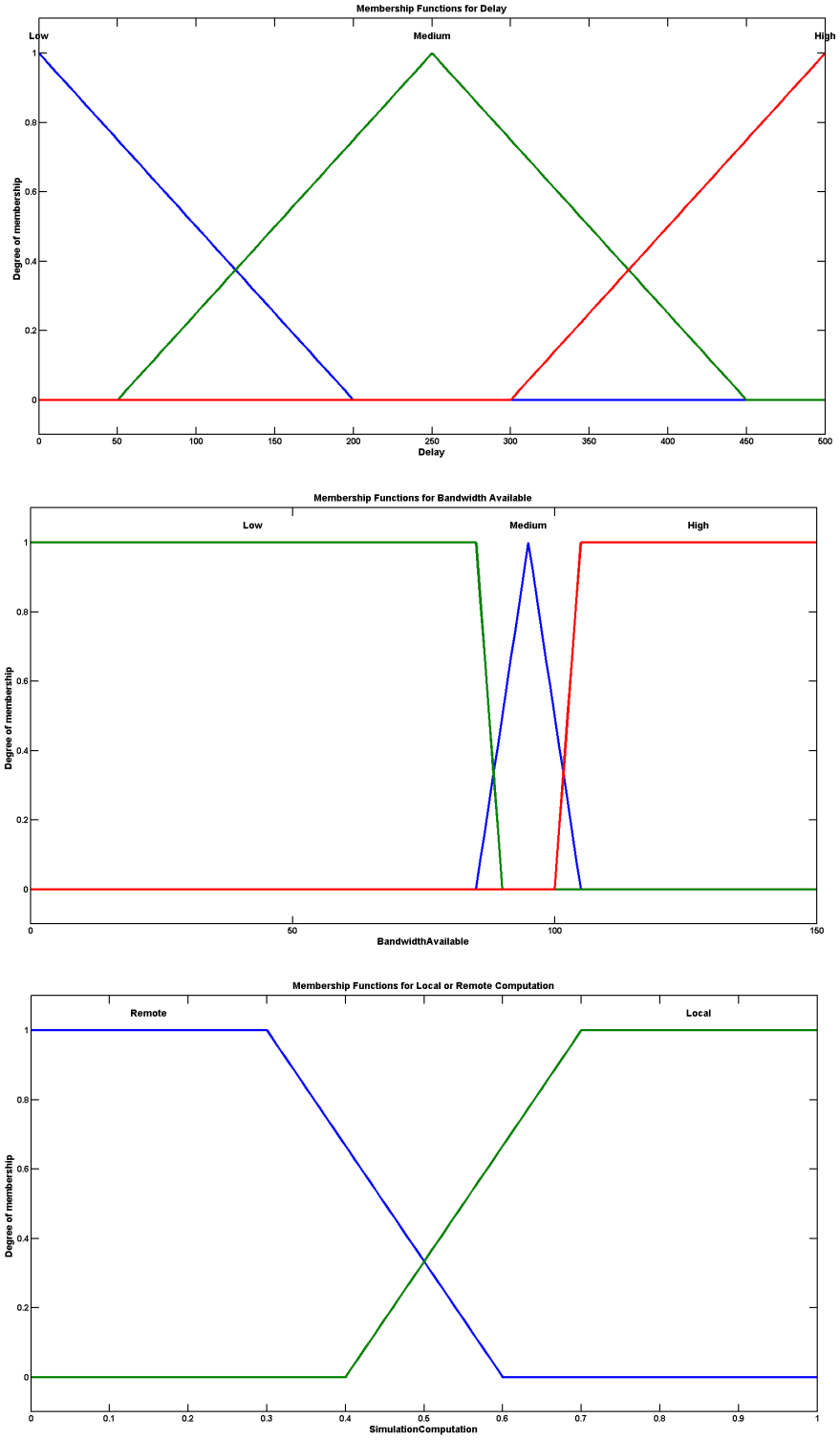


Figure 4.4: Membership functions defined for the Choosing Network Parameters.

quality of the surgical scenario properties chosen are associated with the adaptation mechanism proposed, is because these two input variables define the maximum frequency for sending data from client-server machine to the other client machines, because they define the maximum refresh rate of the simulation at the server machine. Then the fuzzy rules choose resolution and sending frequencies depending on the bandwidth available and the higher refresh rate provided by the client-server machine.

4.3.4 Choosing Client Parameters Subsystem

This subsystem is responsible for choosing the surgical scenario properties running in the client machine and if the computation of the deformation is performed locally o remotely, considering the benchmark, preferences of the user interacting on this machine and the delay of the network connection. The rules defined to choose the qualities executed on the client machine are similar to those defined for the client-server machine, but in this case depend on which machine the calculation of the deformation took place. For example, if the calculation of the deformation is performed in the client machine, the subsystem should consider this in order to choose the qualities of the simulation, because more machine capabilities will be demanded for the client, unlike if the calculation is done on the server. To define the ranges and the membership functions for the delay variable was taken into account the results of the experimental test described in chapter 2. Linguistic terms defined as low, medium and high are used for the delay variable. The membership functions and linguistic terms are the same as those used for the other subsystems where these variables are involved. The fuzzy rules defined for this subsystem are detailed in Annex 3. For the implementation of the fuzzy inference systems described above Matlab version 2014a was used [61].

4.3.5 Collaboration Predictor

This subsystem predicts the collaboration achieved by the users, considering the value of the heterogeneity factors and the adaptation mechanism chosen by each one of the subsystems described above. In this case, two metrics, the task completion time and the number of errors, are used for measuring the collaboration. These two metrics are predicted by this subsystem. It is worth mentioning that the prediction calculated is considering that users are experts performing the task as individuals and a team.

An ANFIS approach was proposed for the development of this subsystem. For developing an ANFIS, a training dataset representing the input and output space to be mapped is required, thus an experiment, where the input and output variables of this subsystem were varied, was performed. For this experiment the same collaborative task and the same experimental setup described in section 2.3.2 was used, but this time the properties of the virtual surgical scenario were changed for each experimental run, considering the table shown in Annex 2 and Figure 3.7. Additionally, the adaptation mechanisms for bandwidth and latency, network context and machine capabilities of client-server and client machines were modified as well. For each experimental setup detailed in Annex 4, the task completion time and number of errors were measured. The users who performed the experiments are experts developing the collaborative task, i.e. the performance of the users are in the plateau zone (see Figure 3.2) of the learning curve for both individual and team skills.

To determine how many experimental runs are required for creating a representative dataset training for the ANFIS, it is needed to calculate all the possible combinations of input and output variables of the

Table 4.2: Cases considered for creating the training dataset for the Collaboration Predictor Subsystem.

Parameters	Best Case	Average Case	Worst Case
Delay(ms)	0	300	600
Bandwidth Available (%)	100	90	80
Server Benchmark (ms)	33*	33	66
Client Benchmark (ms)	33	66*	66

*For a simulation cycle time of 33 ms the aproximate frames per second rate is 30, for 66 ms the frames per second rate is 15.

subsystem [13]. Because, the time and effort required to perform all the necessary experiments involving users collaborating is very costly, the input variables were grouped in three representative cases: worst cases, average cases and best cases. The variables considered to define each one of these cases are the infrastructure factors.

The worst cases were defined as the cases in which the infrastructure factors that most impact the collaboration, regarding the results of the experiment described in chapter 2, have values causing the deterioration of the collaboration. In turn, in the best cases the values of the variables were those that do not affect the collaboration. Finally, in average cases medium values of these variables between the worst cases and the best cases were considered (See table 4.2).

The other variables related to the surgical scenario properties or adaptation mechanisms were varied in order to have a representative sample of the possible combinations for the best, average and worst

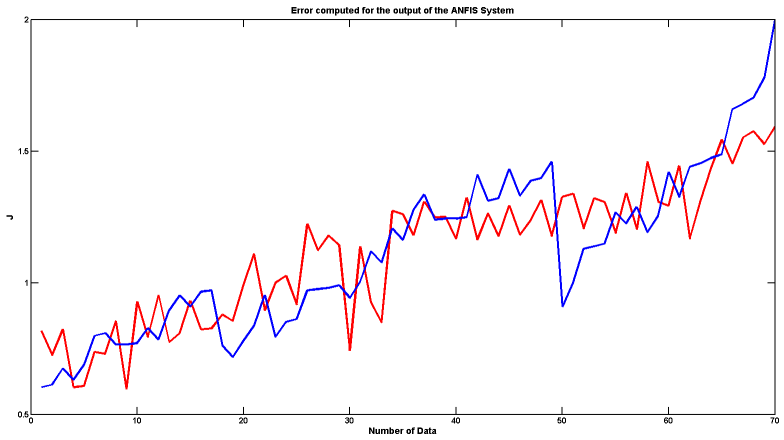


Figure 4.5: Graph showing the value of J calculated for each experimental run (blue color) and the value of J predicted by the ANFIS trained (red color).

cases. Examples of some of the combinations tested and the results obtained for the task completion time and number of errors are reported in Annex 4. For the implementation of this component Matlab version 2014a was used in order to develop the ANFIS system, using the data collected from the experimental test. Once trained and generated the ANFIS using Matlab and the dataset, an evaluation of the error of the ANFIS trained was performed. For the evaluation of the error the same training dataset was used due to the reduced amount of data available. In figure 4.5 can be observed the original output of the dataset used for training the ANFIS system and the value of the output predicted by the ANFIS trained. The root mean square error computed is approximately 0.167. It is worth mentioning that the trained ANFIS system considers as its output the metric J which will be described in the next section. This metric depends on the task completion time and the number of errors of the users during the execution of the collaborative surgical task.

4.3.6 Rate of Change Subsystem

J is an objective function that relates the task completion time and number of errors predicted for the current conditions of CNVSS (factors of infrastructure, surgical scenario properties and adaptation mechanisms) with those determined for the worst case:

$$J = \frac{TCT_p}{TCT_{max}} + \frac{NE_p}{NE_{max}} \tag{4.2}$$

Where TCT_p and TCT_{max} are the predicted and maximum task completion time, respectively, and NE_p and NE_{max} are the predicted and maximum number of errors, respectively. Specifically, TCT_{max} and NE_{max} are the task completion time and number of errors calculated for the worst case of the experimental runs. The objective function J is used to diagnose whether under a specific infrastructure conditions, the adaptation mechanisms applied are appropriate to guarantee the user collaboration.

Analyzing the equation 4.2 is observed that when $TCT_p = TCT_{max}$ and $NE_p = NE_{max}$, the value of $J = 2$, indicating that the infrastructure of the CNVSS and the adaptation mechanisms are not appropriated to guarantee the user collaboration.

On the contrary, whether TCT_p and NE_p are close to minimum task completion time and number of errors and J is close to 0, then the infrastructure and adaptation mechanisms are appropriate to guarantee a good collaboration. The minimum and maximum task completion time and number errors are dependent on the type of collaborative task. The rate of change of this subsystem is applied when $J > \epsilon_c$, where ϵ_c is a tolerance threshold meaning the minimum allowed collaboration in the

CNVSS.

To avoid continuous variation in adaptation mechanisms implemented in the CNVSS, due to any change in the context of the simulator either frames per second or some network parameter, the metric J is used to predict whether the change in the infrastructure factors is significant for impacting the collaboration of the users, and if so the inference mechanism proposed is restarted considering the current conditions. For the development of the inference mechanism proposed and for deploying it in the CNVSS developed, Matlab version 2014a and Matlab compiler version 7 was used, respectively [61] [62].

4.4 Experimental Setup and Results

In this section, an experimental test for determining whether the inference mechanism and context-aware architecture proposed, are able to guarantee the collaboration of users under inappropriate conditions of infrastructure, is described. Eight different cases considering inappropriate infrastructure conditions were involved in the experimental test. A last case was added in order to evaluate the CNVSS under ideal infrastructure conditions of collaboration. Each of these cases was evaluated in two groups, an interventional and a control group. In the control group, two expert users in the collaborative task and in the use of the CNVSS, performed each of these cases using a non context-aware CNVSS. In the interventional group the same two expert users, performed each of the experimental cases, but this time using the proposed context-aware CNVSS. Considering the interventional group, for each case three experimental runs with different values of user preferences for visualization, interaction or collision and behavior were evaluated. In the case of non-context-aware CNVSS where the user's preferences cannot be defined, the experimental run with the same infrastructure

conditions was repeated three times. The experimental runs were performed in a random order for both groups. The different cases and experimental runs considered for evaluating the system proposed can be observed in table 4.3.

The CNVSS for the two groups was started using surgical scenario properties in medium quality, including the resolution of transmission between the client and the client-server machine. From this configuration the bandwidth required by the application is determined for each experimental run. After that, using this bandwidth, the new available bandwidth is calculated by mean of the percentage described in column 3 of Table 4.3 and calculating the 90% and 80% of required bandwidth. Additionally, in column 6 of Table 4.3, the different preference conditions are defined.

The CNVSS architecture used for the experimental runs is described in Chapter 3. Different machines were used for modifying the machine capabilities required by the experimental runs. Considering the network factors, these were modified using Netdisturb and implementing a system configuration as described in Figure 2.5. Surgical scenario properties applied for each of the qualities handled by context-aware architecture are reported in Annex 2. As non context-aware CNVSS the same simulator was used but disabling the context-aware functionality. The task completion time and number of errors were measured for each experimental run of the control group. In the case of the interventional group the following output variables were measured:

1. The output of the inference machine.

2. The FPS of the users machines and bandwidth required by the CNVSS after applying the adaptation mechanisms.

Table 4.3: Cases considered for evaluating the context-awareness CN-VSS proposed.

Case	Experimental Run	Banwidth Available	Delay	Client-Server Benchmark	Client Benchmark	Preferences
	1	90	250	28	28	Low
1	2	90	250	28	28	Low-High
	3	90	250	28	28	High
	4	80	500	28	28	Low
2	5	80	500	28	28	Low-High
	6	80	500	28	28	High
	7	90	250	8	8	Low
3	8	90	250	8	8	Low-High
	9	90	250	8	8	High
	10	80	500	8	8	Low
4	11	80	500	8	8	Low-High
	12	80	500	8	8	High
	13	90	250	28	8	Low
5	14	90	250	28	8	Low-High*
	15	90	250	28	8	High
	16	80	500	28	8	Low
6	17	80	500	28	8	Low-High
	18	80	500	28	8	High
	19	100	0	38	38	Low
7	20	100	0	38	38	Low-High
	21	100	0	38	38	High

*The first linguistic term means the preferences chosen by the client-server machine user and the second one means the preferences of the client machine user. When one linguistic term is used the preferences of the both users are the same.

3. The predicted value of J for the initial conditions and resulting conditions after applying the changes defined by the inference machine.

4. The task completion time and the number of errors of the users.

The output of the inference mechanism for each one of the experimental runs can be observed in tables 4.4, 4.5 and 4.6.

Two screenshots of the client and client-server machine running the CNVSS context-aware after applying the inference mechanism proposed in this chapter and the adaptation mechanisms described in chapter 3 can be observed in figure 4.6. The visual, collision and deformation qualities were modified in each of the machines by the context-aware CNVSS. In figure 4.6, it can be observed that users are collaborating although the simulation qualities running on each machine are heterogeneous.

Additionally, while the collaboration was performed by the users some system metrics related to the state of infrastructure (see Figure 1.2) were measured: (i) frames per second of the user machines, (ii) the bandwidth required by the CNVSS when the surgical scenario properties are defined as medium quality, (iii) the bandwidth reduced when the infrastructure conditions are modified and (iv) the bandwidth decreased when the adaptation mechanisms are applied in the CNVSS by the inference mechanism. All these measurements were recorded only for the context-awareness CNVSS, since in the non context-aware CNVSS no adaptation mechanism is applied, thus the initial parameters of each experimental run was maintained throughout the collaborative

Table 4.4: Output of the inference mechanism defining the surgical scenario properties for the Client-Server Machine.

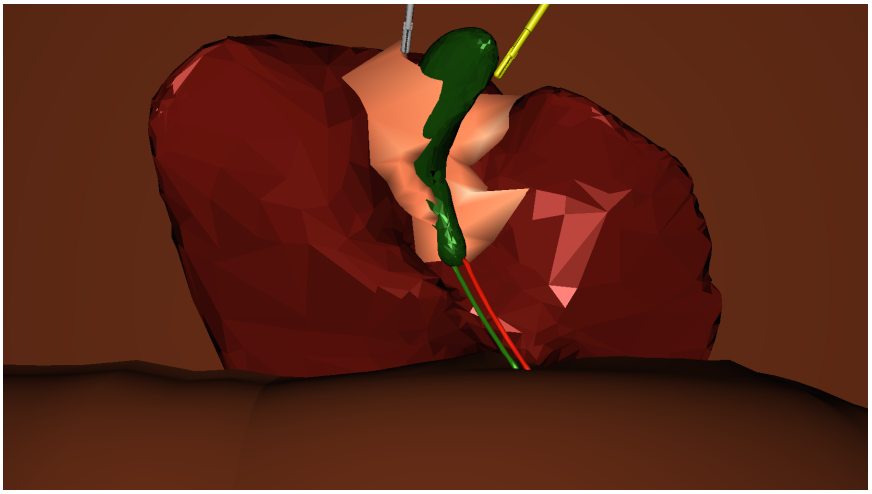
Run	Behavior Quality	Collision Quality	Visual Quality
1	Low	Low	Low
2	Low	Low	Low
3	Low	Medium	Medium
4	Low	Low	Low
5	Low	Low	Low
6	Low	Medium	Medium
7	Low	Low	Low
8	Low	Low	Low
9	Low	Low	Low
10	Low	Low	Low
11	Low	Low	Low
12	Low	Low	Low
13	Low	Low	Low
14	Low	Low	Low
15	Low	Medium	Medium
16	Low	Low	Low
17	Low	Low	Low
18	Low	Medium	Medium
19	High	High	High
20	High	High	High
21	High	High	High

Table 4.5: Output of the inference mechanism defining the parameters for the network adaptation mechanisms.

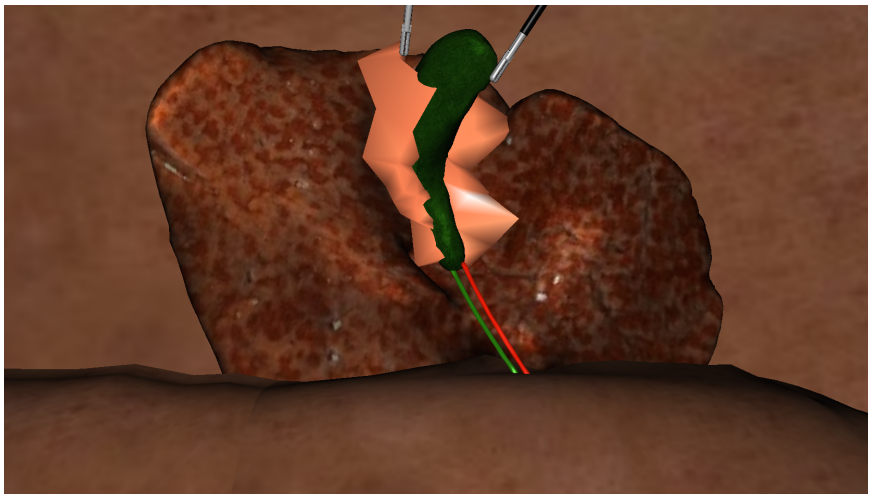
Run	Transmission Resolution	Transmission Frequency	Local or Remote Computation
1	Medium	Medium	Local
2	Medium	Medium	Local
3	Medium	Medium	Local
4	Low	Medium	Local
5	Low	Medium	Local
6	Low	Medium	Local
7	Low	Medium	Local
8	Low	Medium	Local
9	Low	Medium	Local
10	Low	Low	Local
11	Low	Low	Local
12	Low	Low	Local
13	Medium	Medium	Local
14	Medium	Medium	Local
15	Medium	Medium	Local
16	Low	Medium	Local
17	Low	Medium	Local
18	Low	Medium	Local
19	Medium	Medium	Remote
20	Medium	Medium	Remote
21	Medium	Medium	Remote

Table 4.6: Output of the inference mechanism defining the surgical scenario properties for the Client Machine.

Run	Behavior Quality	Collision Quality	Visual Quality
1	Low	Low	Low
2	Low	Medium	Medium
3	Low	Medium	Medium
4	Low	Low	Low
5	Low	Medium	Medium
6	Low	Medium	Medium
7	Low	Low	Low
8	Low	Low	Low
9	Low	Low	Low
10	Low	Low	Low
11	Low	Low	Low
12	Low	Low	Low
13	Low	Low	Low
14	Low	Low	Low
15	Low	Low	Low
16	Low	Low	Low
17	Low	Low	Low
18	Low	Low	Low
19	High	High	High
20	High	High	High
21	High	High	High



(a) Screenshot of the Client-Server Machine.



(b) Screenshot of the Client Machine.

Figure 4.6: Screenshots of the context-awareness CNVSS when two users are collaborating.

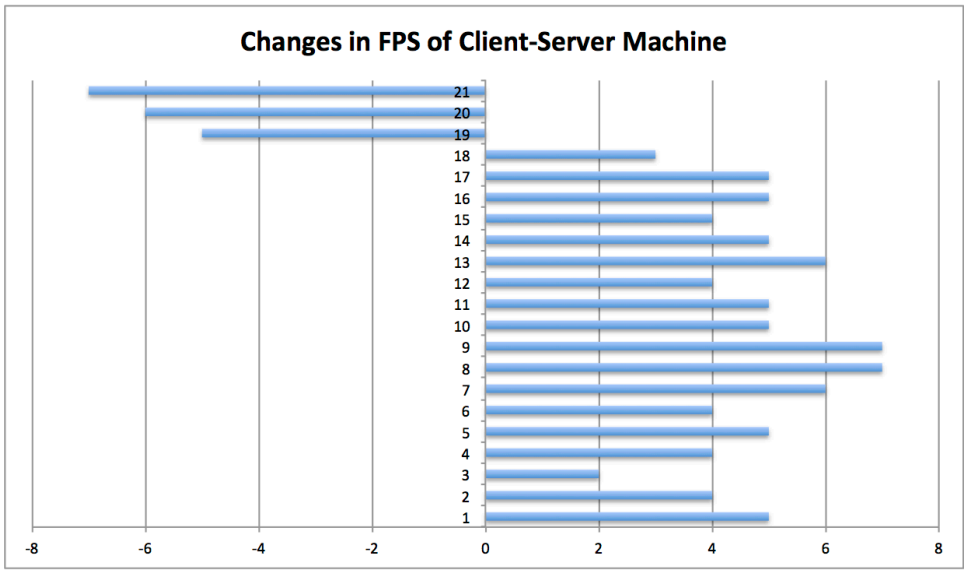


Figure 4.7: Changes produced by the adaptation mechanisms in the variable FPS of the client-server machine.

session. The measurements recorded are reported in table 4.7.

Comparing the FPS measured for the client-server machine and those measured starting the collaborative session and reported in table 4.3, it is possible to observe that the adaptation mechanisms applied by the inference machine allows for an increment in the FPS. Even if the FPS of the machine was categorized as low or medium quality machine, the changes applied allow increasing their categorization as medium or normal quality, respectively. The increase of the client-server machine FPS for each experimental run can be observed in figure 4.7. This increase in FPS was produced by the adaptations inferred and applied by the proposed context-aware architecture.

The largest increases are produced when the FPS of the client-server

Table 4.7: Medidas del estado de la infraestructura realizadas al CNVSS context-awareness.

Run	FPS Client-Server	FPS Client	Required Bandwidth	Reduced Bandwidth	Adapted Bandwidth
1	33	34	10.24	9	5.8
2	32	32	10	9	5.76
3	30	31	10	9	5.2
4	32	31	10	8	4.8
5	33	32	10	8	4.88
6	32	32	10	8	5.6
7	14	15	3.1	2.79	2.1
8	15	15	3.1	2.79	2.24
9	15	15	3.1	2.79	2.093
10	13	14	3.1	2.48	0.93
11	13	13	3.1	2.48	0.96
12	12	14	3.1	2.48	1.34
13	34	15	12.3	11.07	6.29
14	33	13	12.3	11.07	5.44
15	32	14	12.3	11.07	5.4
16	33	15	12.21	9.76	4.64
17	33	14	12.21	9.76	5.328
18	31	16	12.21	9.76	5.5
19	33	57	13.95	13.95	6
20	32	60	13.95	13.95	4.8
21	31	58	13.95	13.95	5.1

The Bandwidth is expressed in Megabits per second.

machine under initial conditions is low quality, because the inference mechanism sets all the simulation qualities to low in order to improve the client-server machine performance. Additionally, in case 8 (see table 4.3), where the system conditions are appropriate to guarantee the collaboration, a decrease in FPS is produced by the inference mechanism. It is because the machine is normal quality and the inference mechanism determines to improve the properties of the surgical scenario.

Moreover, the bandwidth adaptation mechanism applied by the context-aware CNVSS allows for a decrease in the required bandwidth close to 50%. For this reason, when the available bandwidth is reduced by 80% and 90% for each of the experimental runs, the simulation of the CNVSS is not affected, unlike what occurs in the non context-aware CNVSS. The percentage of bandwidth decreased by the inference mechanism can be observed in figure 4.8.

The experimental runs in which the inference mechanism applies a greater decrease in the used bandwidth, are those where the reduction of the bandwidth was greater, that is when the bandwidth was reduced by 80%. Otherwise, when there is a delay categorized as medium or low in the network conditions, the inference mechanism decides to perform the deformation computation locally in the client machine. Besides, for the non context-aware CNVSS the deformation computation is always computed remotely in the client-server machine, then the FPS of the client machine will be always in the order of 55 to 60 FPS. For these reasons, the FPS of the client machine, under conditions of network latency, will always be lower in the context-aware architecture than in the not context-aware one.

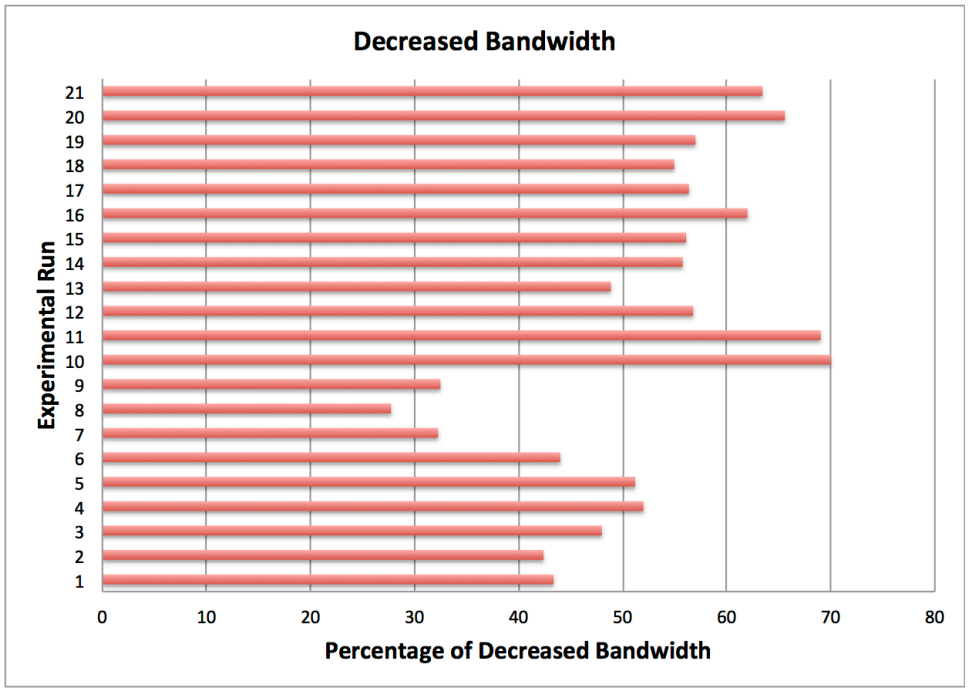


Figure 4.8: Changes produced by the adaptation mechanisms in the Bandwidth used by the CNVSS.

Table 4.8: Value of J at the begining of the each experimental run and after apply the adaptations suggested by the inference mechanism.

Run	$J_{Initial}$	J_{Final}
1	1.153	0.834
2	1.153	0.840
3	1.153	0.813
4	1.505	0.843
5	1.505	0.849
6	1.505	0.823
7	1.105	0.796
8	1.105	0.777
9	1.105	0.796
10	1.458	0.669
11	1.458	0.679
12	1.458	0.662
13	1.212	0.893
14	1.212	0.893
15	1.212	0.867
16	1.565	0.902
17	1.565	0.903
18	1.565	0.876
19	0.847	0.699
20	0.847	0.695
21	0.847	0.696

The results obtained for the predicted value of J, under initial conditions of each experimental run ($J_{initial}$), and the predicted value of J after applying the changes suggested by the inference mechanisms (J_{final}), are reported in table 4.8.

In all experimental runs from 1 to 18, in which the infrastructure conditions are not appropriate for guaranteeing the collaboration, a decrease can be observed between the values of $J_{initial}$ and J_{final}. Since

J_{final} approaches 0, it could be suggested that the predicted collaboration in the CNVSS, after the adaptation mechanism are applied, are better, compared with the initial conditions. It is worth remembering that the J metric was proposed to predict whether the CNVSS conditions are appropriate to ensure the collaboration or not. Besides, variations in the value of J, between cases where the conditions of bandwidth and latency are appropriate or not, can be observed in the calculated values. It is worth mentioning that, in the case of users who collaborate using the non context-aware CNVSS, the value of the $J_{initial}$ metric is maintained throughout the collaboration session.

Finally, in Table 4.9, the task completion time and number of errors obtained for each of the experimental runs, considering each of the experimental groups, are reported.

The errors made by users in a CNVSS for this experimental test are defined in table 2.9. The differences in task completion time and number of errors between the two groups can be observed in figures 4.9 and 4.10. For all cases, except for experimental runs 19, 20 and 21, there is a decrease in the time and number of errors when the infrastructure factors are not appropriate, and users collaborate through the context-aware CNVSS. The task completion time and number of errors are similar in both groups for experimental runs 19, 20 and 21, because in these cases the infrastructure factors are ideal, therefore collaboration in the non context-awareness CNVSS is not affected.

In order to determine whether the architecture and the proposed inference mechanism are able to guarantee the collaboration when the values of the infrastructure factors are not appropriate, a paired t-

Table 4.9: Task completion time and number of errors for each one of the groups.

	Control Group		Experimental Group	
Run	Task Completion Time*	Number of Errors	Task Completion Time	Number of Errors
1	157	9	128	4
2	157	10	130	2
3	159	9	125	2
4	156	16	130	5
5	165	9	132	8
6	163	15	130	7
7	160	14	134	6
8	163	16	135	4
9	161	16	136	6
10	179	18	139	9
11	175	15	140	7
12	166	17	135	8
13	146	15	131	4
14	154	14	127	3
15	156	16	130	3
16	152	12	126	6
17	155	17	124	6
18	168	18	133	5
19	123	4	129	5
20	126	6	124	7
21	127	5	131	4

*Task Completion Time is measured in seconds.

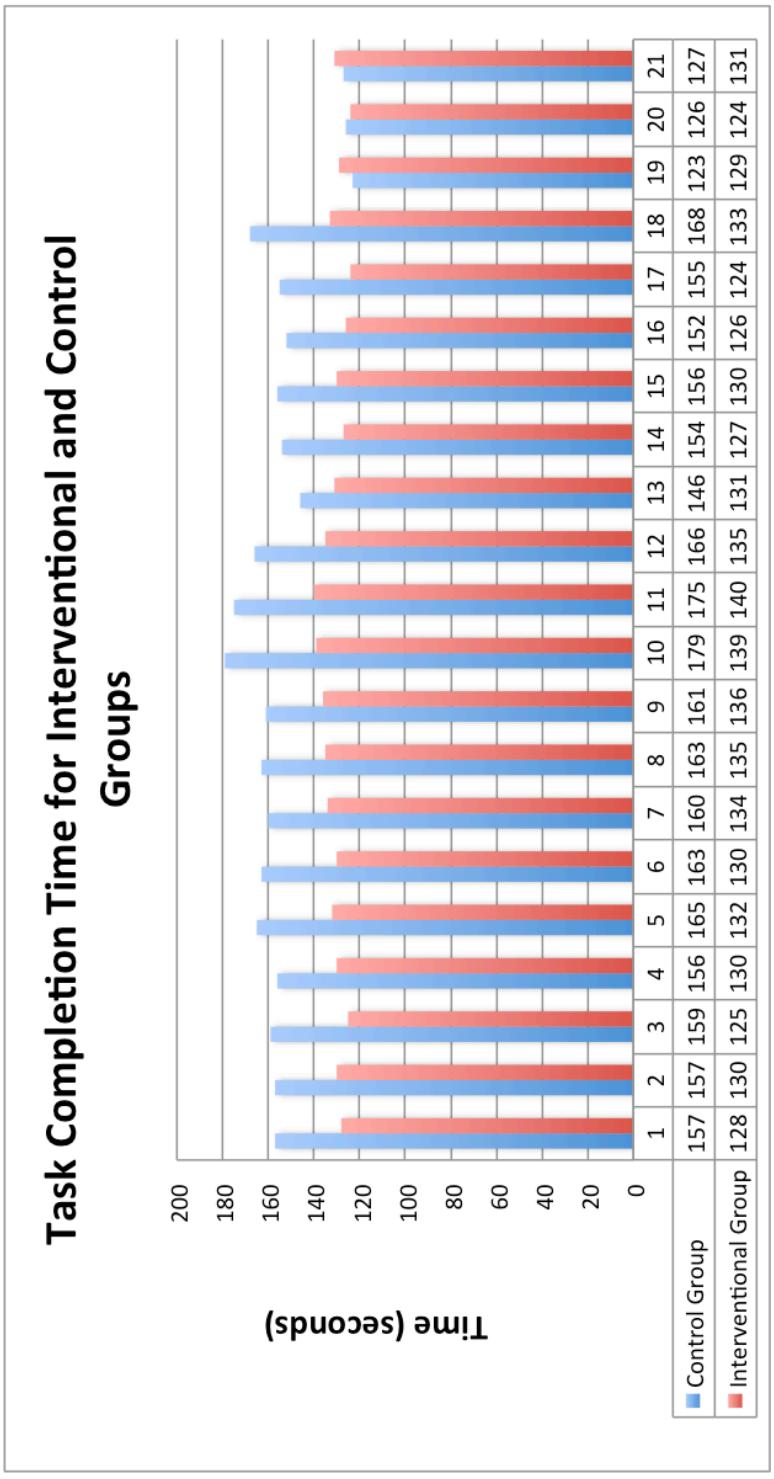


Figure 4.9: Results obtainted for the variable task completion time.

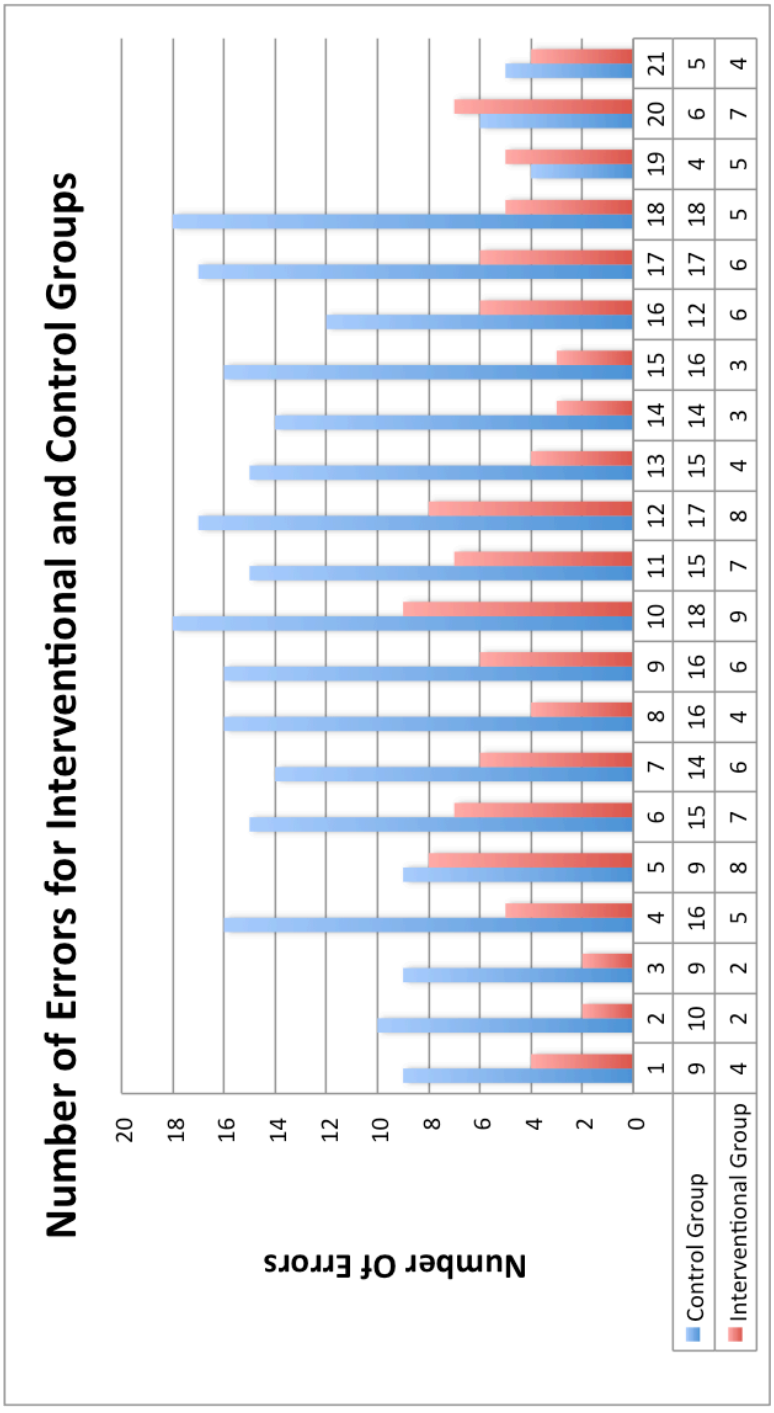


Figure 4.10: Results obtainted for the variable number of errors.

Student test was conducted verifying the hypothesis that the groups are not similar. The task completion time and the number of errors were used for two separate test t-student. The R statistics package was used for performing the t-student test.

The computed p-values for the task completion time and number of errors comparing both groups were $4.029x10^{-14}$ and $7.812x10^{-11}$, respectively.These p-values suggest, with a confidence interval larger than 95%, that the control and interventional groups are not similar. Additionally, the differences in the mean values suggest that when experienced users collaborate through the non context-aware CNVSS, there is an increase of 29.27 seconds in the task completion time and 9.27 for the number of errors, when compared with the time taken and the errors made by the experts users collaborating through the context-aware CNVSS.

Regarding that the collaborative task is the same in all experimental runs and that the users are experts as individual and as a team, it can be suggested that when the conditions of infrastructure are inappropriate, the context-awareness CNVSS proposed is able to guarantee the collaboration, supported by the values of the task completion time and number of errors. This is also evidenced noting that values of users performance, collaborating through the context-awareness CNVSS when infrastructure factors are inappropriate, are close to values obtained for experimental runs in which infrastructure factors are ideal for collaboration.

4.5 Conclusions

An inference mechanism that is able to handle heterogeneity factors for maintaining the collaboration when infrastructure factors are not

appropriate was described in this chapter. The proposed mechanism uses, as input, the context of the CNVSS in order to determine the adaptation mechanisms to be applied as output, specifically the properties of the surgical scenario at the machine users and the parameters of the network adaptation mechanism. Besides, the inference mechanism is composed of several subsystems based on artificial intelligence methods, which reduce the complexity of the problem and involve the expert knowledge obtained from the experiments reported in chapter 2.

Finally, the inference mechanism and context-aware architecture were evaluated experimentally, in cases where the infrastructure factors are not appropriate for guaranteeing a good collaboration. During the experimental test metrics that quantify the state of infrastructure and the state of collaboration in CNVSS were recorded. From the results, it can be concluded that the inference mechanism and the proposed context-aware architecture improved, the value of infrastructure factors such as the available bandwidth and frames per second of the users machines in all experimental runs where infrastructure conditions are inappropriate. Additionally, the task completion time and the number of errors, when infrastructure conditions are not appropriate, were reduced when the users collaborate through the context-aware CNVSS, even obtaining values of the task performance close to those obtained on CNVSS running on ideal infrastructure conditions. This highlights that the collaboration performance of expert users, collaborating through the proposed context-aware CNVSS, is guaranteed even in the presence of not appropriate infrastructure factors.

5

Conclusions and Future Research

5.1 General Conclusions

The paradigm change in surgical training that is taking place in recent years, moving from "see one, do one and teach one" model in which the training is performed directly with the patient, to one model based on the use of simulators for training, is promoting the development of tools to train all skills required in an operating room. Besides, the limited availability of experts and their location in large urban centers makes approaches such as CNVSS to be demanded more and more in the field of surgical training, not only for training manual surgical skills but also for training of team skills and new procedures. However, various factors of heterogeneity as described in section 1.1 can affect the collaboration of users in a CNVSS and therefore limit the effectiveness and efficiency

of the training session.

For these reasons this dissertation makes the following major contributions in order to improve the level of collaboration provided by the CNVSS under heterogeneous conditions:

1. The heterogeneity factors involved with a CNVSS were identified and described. Based on these factors, a set of experiments for determining which infrastructure factors affect collaboration the most in a CNVSS, using two types of network architectures, were performed. From the results it was concluded that there are infrastructure factors that do not affect the collaboration and others that affect considerably the performance of the collaborative task doing by the users. The factors that affect collaboration the most, in the presence of a hybrid client-server architecture, are bandwidth, latency and client-server machine capabilities.

2. A context-aware architecture that handles heterogeneity factors affecting collaboration, applying various adaptation mechanisms, was proposed and implemented. The architecture determines, in real-time, the context of the CNVSS and applies changes in the surgical scenario properties and network adaptation mechanism, in order to guarantee the collaboration of the users, without stopping or affecting it. This flexibility to change and adapt was provided to the CNVSS without significant penalty in the performance of the simulation.

3. An inference mechanism that uses the context of the CNVSS and smartly determines the most appropriate adaptations, in order to guarantee the collaboration when infrastructure conditions are not ideal, was proposed and implemented. The inference mechanism is based on artificial intelligence methods, the knowledge

base is built based on the conclusions obtained from the experiments.

4. Finally, an experimental test, involving experts users in order to evaluate whether the context-aware architecture and inference mechanism are able to handle heterogeneity factors and guarantee the collaboration during a training session, was undertaken. From the results it was concluded that the level of collaboration measured through the task completion time and the number of errors is guaranteed by using the context-aware CNVSS, even though the infrastructure conditions are not appropriate.

5.2 Future Research

In this final section the directions of future research following from the results presented in this dissertation are discussed. The future works is organized in the following directions:

1. One of the main contribution of this dissertation was to evaluate and determine which heterogeneity factors affect collaboration among users. The metrics used determine whether the collaboration was good or not, but the metrics are not capable of determining if the training session was successful or not. Besides, the reason why experts or trained users were used in the experimental test, was to avoid individual or team variabilities to affect the collaboration measurements, and to obtain the true impact of the heterogeneity. It would be interesting to understanding how poor collaboration among users, in a CNVSS, affects the acquisition of skills and dexterities.

2. The case study analyzed in this dissertation, involved a laparoscopic surgical procedure where only two users were interacting

with the virtual surgical environment. It would be valuable to extend the CNVSS in order to support other surgical roles, such as an instrumentalist or an anesthesiologist, in the simulation scenario. Similarly, the CNVSS should allow for the simulation of other surgical scenarios. It would be interesting to evaluate the applicability of the proposed context-aware architecture for such collaborative tasks.

3. A limitation of this research was the use of a limited number of simulation qualities, since the resolutions and algorithms were predetermined and loaded when the simulator was starting. A larger number of simulation qualities could allow for a better tuning of the system.

4. Different algorithms and methods could be explored in order to create, in real-time, the appropriate resolutions for a specific set of infrastructure conditions.

6

Annexes

Annex 1. Description of the Mapping Process Applied by SOFA.

The description presented here for the mapping process applied by SOFA is based on that reported in [32]. Object simulated in SOFA typically rely on several models: one for the internal model, one for collision, and one for the visual rendering. To enforce consistency, one of them, typically the internal model, acting as the master, imposes its displacements to slaves (typically the visual model and the collision model), using mappings. Mapped model can be masters of other models in turn, creating a hierarchy whith the independent DOFs at the root. The independent DOF of the objects, on top, are the masters of contact models based on triangle vertices. When the contact models collide, pairs of contact points are created, each point a slave of a contact model.

Let M be the function used to map the positions x_m of a master model to the positions x_s of a slave:

$$x_s = M(x_m) \tag{6.1}$$

The velocities are mapped in a similar way:

$$v_s = Jv_m \tag{6.2}$$

The Jacobian matrix $J = \frac{\partial x_s}{\partial x_m}$ encodes the linear relation between the master and slave velocities. Accelerations can be mapped using:

$$a_s = Ja_m + \frac{\partial J}{\partial x_m} v_m \tag{6.3}$$

In linear mappings, operators M and J are the same, otherwise M is nonlinear with respect to x_m and it can not be written as a matrix. For surfaces embedded in deformable cells, matrix J contains the barycentric coordinates. For surfaces attached to rigid bodies, each row of the matrix encodes the usual relation $v = \dot{o} + \omega(xo)$ for each vertex.

The positions and the velocities are propagated top-down in the hierarchy. Conversely, the forces are propagated bottom-up to the independent DOFs, where Newton's law $f = Ma$ is applied. Given forces fs applied to a slave model, the mapping computes and accumulates the equivalent forces fm applied to its master. Since equivalent forces must have the same power, the following relation holds:

$$v_m^T f_m = v_s^T f_s \tag{6.4}$$

The kinematic relation $v_s = Jv_m$ allows us to rewrite the previous equation as

$$v_m^T f_m = v_m^T J^T f_s \tag{6.5}$$

Since this relation holds for all possible velocities v_m, the principle of virtual work allows us to simplify the previous equation to obtain:

$$f_m = J^T f_s \tag{6.6}$$

When a model has several slaves, each slave accumulates its contribution to the forces on the master using its mapping. This hierarchical kinematic model allows us to compute displacements and to apply forces

at all levels. So far, 22 variants of mappings have been implemented in SOFA to attach models to rigid objects and deformable primitives such as tetrahedra, hexahedral grids, splines, blended frames, flexible beams and scalar fields. Mappings are also used to connect generalized coordinates, such as joint angles, to world-space geometry.

Annex 2. Properties of the Surgical Scenarios.

The characteristics used for the virtual surgical scenarios in low, medium and high quality can be observed in the Tables 6.1 and 6.2. Table 6.1 shows the characteristics related to resolution and the table 6.2 shows the characteristics related to algorithms. The other anatomical structures that are part of the surgical scenario, such as: cystic duct, cystic artery and ligaments joining liver and gall bladder only handle one quality of resolution and algorithm, and its characteristics were detailed in Table 3.2.

Table 6.1: Characteristics of the deformable, visual and collision data structures used in the low, medium and high quality virtual surgical scenarios.

	Anatomical Structure		Visual Model	Collision Model	Deformation Model
Low Quality	**Liver**	Points	2005	1011	145
		Elements	4002	2014	1293
	Gall Bladder	Points	1022	255	157
		Elements	2040	506	1292
Medium Quality	**Liver**	Points	4007	2045	246
		Elements	8006	4082	2337
	Gall Bladder	Points	2552	737	205
		Elements	5100	1470	1763
High Quality	**Liver**	Points	7558	4007	414
		Elements	15102	8006	4062
	Gall Bladder	Points	5051	1991	368
		Elements	10098	3978	3355

The type of geometry primitive used and called elements in this table depends on the type of model. In this case the elements used in the collision and visual models are triangles and in the deformation model are tetrahedra.

Table 6.2: Characteristics of the deformable, visual and collision data structures used in the low, medium and high quality virtual surgical scenarios.

	Anatomical Structure	**Collision Model**	**Behavior Model**	**Visual Model**
Low Quality	**Liver Gall Bladder**	SH	MSM	FS
Medium Quality	**Liver Gall Bladder**	AABBH	TMM	SS
High Quality	**Liver Gall Bladder**	OBBH	FEM	SST

SH: Sphere Hierarchy. AABBH: Axis-Aligned Bounding Box Hierarchy. OBB: Oriented Bounding Box Hierarchy. MSM: Mass Spring Method. TMM: Tensor Mass Method. FEM: Finite Element Method. FS: Flat shading. SS: Smooth Shading. SST: Smooth Shading and Texture.

Annex 3. Fuzzy Inference Rules Defined for the Subsystems.

In the following sections of this annex some examples of fuzzy inference rules composing the subsystems of the inference mechanism described in chapter 4 are described.

Fuzzy Inference Rules for the Choosing Server Parameters Subsystem

As mentioned in Chapter 4, this subsystem uses, as input variables, the FPS benchmark of the machine chosen as client-server and user preferences. Using these input variables the system determines the most appropriate simulation qualities that should be run in the client-server machine. Four examples of the inference rules are listed next.

1. If (FPS Benchmark is Low Quality) and (Behavior Preference is Low) and (Collision Preference is Low) and (Visual Preference is Low) Then (Behavior Quality is Low)(Collision Quality is Low)(Visualization Quality is Low).

2. If (FPS Benchmark is Medium Quality) and (Behavior Preference is High) and (Collision Preference is High) and (Visual Prefer-

ence is High) Then (Behavior Quality is Low)(Collision Quality is Medium)(Visualization Quality is Medium).

3. If (FPS Benchmark is Normal Quality) and (Behavior Preference is Low) and (Collision Preference is High) and (Visual Preference is High) Then (Behavior Quality is Low)(Collision Quality is High)(Visualization Quality is High).

4. If (FPS Benchmark is High Quality) and (Behavior Preference is Medium) and (Collision Preference is High) and (Visual Preference is High) Then (Behavior Quality is High)(Collision Quality is High)(Visualization Quality is High).

Each example listed above considers a type of machine capability defined in the inference mechanism. Regarding the rule 1, where the capabilities of the machine are low, the users preferences are not taken into account, since the main objective is to increase the FPS of the machine. In the rule 2, where the capabilities of the machine are medium, user preferences are taken into account but only for the qualities related with the collision and visualization. The behavior quality of the simulation, which is impacting more the performance of the machine is set low in order to increase the FPS. Whereas rule 3, where the capabilities of the machine are normal, the objective is to fit the qualities in order to meet all user preferences, because the machine have enough capabilities. Finally in rule 4, where the capabilities of the machine are high the goal is to fulfill beyond the expectations of the user considering his/her preferences, and defining all the qualities as high.

Fuzzy Inference Rules for the Choosing Network Parameters Subsystem

As mentioned in Chapter 4, this subsystem has as input variables FPS benchmark of the machine chosen as client-server, the qualities of the

surgical scenario defined by the subsystem described in previous section and the percentage of bandwidth available. Using these input variables the system determines the most appropriate resolution and frequency transmission of the deformation data. Two examples of the fuzzy rules defined are listed below.

1. If (FPS Benchmark is Low Quality) and (Collision Quality is Low) and (Behavior Quality is Low) and (Visual Quality is Low) and (Bandwidth Available is Low) Then (Resolution Transmission is Low)(Frequency Transmission is Low)

2. If (FPS Benchmark is Medium Quality) and (Collision Quality is Low) and (Behavior Quality is Low) and (Visual Quality is Low) and (Bandwidth Available is High) Then (Resolution Transmission is Low)(Frequency Transmission is High)

Considering rule 1, where the bandwidth is low, the objective of the rule is to decrease the quality of data sent between the client machine and the client-server machine in order to adjust it to a reduction in the bandwidth required by the CNVSS. The proposed CNVSS is able to decrease the initial required bandwidth approximately by 50%, varying the quality of data and frequencies of transmission. In the case of rule 2, one could question if the bandwidth available is 100%, why the quality of the data sent is reduced. The answer is that the context-aware CNVSS starts the collaborative session running with medium qualities in the simulation, and in these conditions the minimum bandwidth required by the application is defined. Then, although the bandwidth is not reduced, the client-server frequency of sending is increased by decreasing the qualities, for that reason it is necessary to consider a reduction in the quality of the data transmitted.

Fuzzy Inference Rules for the Choosing Client Parameters Subsystem

As mentioned in Chapter 4, this subsystem makes use of the FPS benchmark of the machine chosen as client, the user's preferences and the delay associated with the network connection as input variables. Using these input variables the system determines the most appropriate simulation qualities running at the client machine and whether the computation of the simulation for the client will be executed locally of remotely. Two examples of the fuzzy rules defined are listed below.

1. If (Delay is Low) and (FPS Benchmark Client is Low Quality) and (Collision Preferences is Low) and (Behavior Preferences is Low) and (Visual Preferences is Low) Then (Simulation Computation is Remote)(Behavior Quality is Low)(Collision Quality is Low)(Visual Quality is Low)

2. If (Delay is High) and (FPS Benchmark Client is Medium Quality) and (Collision Preferences is Low) and (Behavior Preferences is High) and (Visual Preferences is Low) Then (Simulation Computation is Local)(Behavior Quality is Low)(Collision Quality is Low)(Visual Quality is Low)

As per rules 1 and 2, the variable that determines whether the calculation is performed remotely or locally is the latency network of connection. Depending on where the simulation calculation will be performed, the inference rules define the qualities for the simulation considering the FPS benchmark of the client machine and the user preferences. If the calculation is done locally on the client machine, the rules are very similar to those defined for the subsystem choosing client-server parameters. If the calculation were done remotely, the client would have more computing capabilities available to define the quality of collision and visualization.

Annex 4. Result of the Experimental Test Performed to Create a Training Dataset for The Collaboration Predictor.

Some examples of the dataset used for training the ANFIS system is shown in the tables 6.3, 6.4 and 6.5. Each row of the tables together form an input-output pair used to train ANFIS. The input-output pairs were separated into three tables to make it more legible on the document. They are categorized by best, average and worst case. A total of 70 experiments were performed to create the full dataset.

Table 6.3: Part of the dataset used for training the ANFIS system implemented by the inference mechanism. The data involving the quality of the client-server machine is included in this table.

	FPS Benchmark		Client-Server Quality					
	ClientServer	Client	Delay	Bandwidth Available	Behavior	Collision	Visual	J
Best Case	Normal Quality	Normal Quality	Low	High	Low	Low	Low	0.688
	Normal Quality	Normal Quality	Low	High	High	Medium	Medium	0.966
Average Case	Medium Quality	Normal Quality	Medium	Medium	Medium	Low	Medium	0.794
	Medium Quality	Medium Quality	Medium	Medium	High	High	Low	1.311
Worst Case	Low Quality	Low Quality	High	Low	Low	Medium	Low	1.488
	Low Quality	Low Quality	High	Low	High	High	High	1.779

Table 6.4: Part of the dataset used for training the ANFIS system implemented by the inference mechanism. The data involving the network adaptation mechanisms is included in this table.

	FPS Benchmark		Delay	Bandwidth Available	Transmission Quality		Local Remote	J
	Server	Client			Resolution	Frequency		
Best Case	Normal	Normal	Low	High	Medium	Low	Remote	0.688
	Quality	Quality						
	Normal	Normal	Low	High	Medium	Medium	Local	0.966
	Quality	Quality						
Average Case	Medium	Normal	Medium	Medium	High	Medium	Remote	0.794
	Quality	Quality						
	Medium	Medium	Medium	Medium	Low	High	Local	1.311
	Quality	Quality						
Worst Case	Low	Low	High	Low	High	High	Local	1.488
	Quality	Quality						
	Low	Low	High	Low	High	High	Local	1.779
	Quality	Quality						

Table 6.5: Part of the dataset used for training the ANFIS system implemented by the inference mechanism. The data involving the client machine qualities of the simulation is included in this table.

	FPS Benchmark			Bandwidth	Client Quality			
	Server	**Client**	**Delay**	**Available**	**Behavior**	**Collision**	**Visual**	**J**
Best Case	Normal Quality	Normal Quality	Low	High	Low	Low	Low	0.688
	Normal Quality	Normal Quality	Low	High	Medium	Low	High	0.966
Average Case	Medium Quality	Normal Quality	Medium	Medium	High	Low	Medium	0.794
	Medium Quality	Medium Quality	Medium	Medium	Low	High	Medium	1.311
Worst Case	Low Quality	Low Quality	High	Low	Low	Low	High	1.488
	Low Quality	Low Quality	High	Low	High	Medium	High	1.779

Bibliography

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, 1999. Springer-Verlag. ISBN 3-540-66550-1. URL http://portal.acm.org/citation.cfm?id=647985.743843.

[2] Ajith Abraham. Hybrid artificial intelligence systems. In Emilio Corchado, JuanM. Corchado, and Ajith Abraham, editors, *Innovations in Hybrid Intelligent Systems*, volume 44 of *Advances in Soft Computing*, pages XVI–XVI. Springer Berlin Heidelberg, 2007.

[3] Emmanuel Agu, Kutty Banerjee, Shirish Nilekar, Oleg Rekutin, and Diane Kramer. A middleware architecture for mobile 3d graphics. In *Proceedings of the Third International Workshop on Mobile Distributed Computing - Volume 06*, pages 617–623, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2328-5-06. doi: 10.1109/ICDCSW.2005.15.

[4] M. Osama Alhalabi, Susumu Horiguchi, and Susumu Kunifuji. An experimental study on the effects of network delay in cooperative shared haptic virtual environment. *Computers & Graphics*, 27 (2):205 – 213, 2003. ISSN 0097-8493.

[5] Jeremie Allard, Stephane Cotin, Francois Faure, Pierre-Jean Bensoussan, Francois Poyer, Christian Duriez, Herve Delingette, and Laurent Grisoni. Sofa: an open source framework for medical simulation. In *Studies in Health Technology and Informatics*, volume 125, pages 13 – 18, 2007.

[6] Robert S. Allison, James E. Zacher, David Wang, and Joseph Shu. Effects of network delay on a collaborative motor task with telehaptic and televisual feedback. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and its Applications in Industry*, pages 375–381, New York, NY, USA, 2004. ACM. ISBN 1-58113-884-9.

[7] Marcus S. Aquino, Fernando F. de Sousa, and Alejandro C. Frery. Multi-agent architecture for generating and monitoring adaptive virtual environments. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pages 515–517, Washington, DC, USA, 2005. IEEE Computer Society. doi: 10.1109/ICHIS.2005.77. URL http://portal.acm.org/citation.cfm?id=1111688.1112220.

[8] Margaret Bearman, Robert O'Brien, Adrian Anthony, Ian Civil, Brendan Flanagan, Brian Jolly, David Birks, Mary Langcake, Elizabeth Molloy, and Debra Nestel. Learning surgical communication, leadership and teamwork through simulation. *Journal of Surgical Education*, 69(2):201 – 207, 2012.

[9] Mounir Beggas, Lionel Médini, Frederique Laforest, and MohamedTayeb Laskri. Fuzzy logic based utility function for context-aware adaptation planning. In Abdelmalek Amine, Ait Mohamed Otmane, and Ladjel Bellatreche, editors, *Modeling Approaches and Algorithms for Advanced Computer Applications*, volume 488 of *Studies in Computational Intelligence*, pages 227–236. 2013. doi: 10.1007/978-3-319-00560-7_27.

[10] Rasiah Bharathan, Rajesh Aggarwal, and Ara Darzi. Operating room of the future. *Best Practice & Research Clinical Obstetrics & Gynaecology*, 27(3):311 – 322, 2013. ISSN 1521-6934. Advances in Gynaecological Surgery.

[11] Blender. Open Source 3D Content Creation Suite, feb 2012. URL http://www.blender.org/.

[12] James Bliss, Alexandra Proaps, and Eric Chancey. Human performance measurement in virtual environments. In Kelly S. Hale and Kay M. Stanney, editors, *Handbook of Virtual Environments*. CRC Press, 2014.

[13] M. Buragohain. *Adaptive Network based Fuzzy Inference System (ANFIS) as a Tool for System Identification with Special Emphasis on Training Data Minimization*. PhD thesis, Indian Institute of Technology Guwahati, 2008.

[14] J. Huegeland O. Celik, A. Israr, and M. O'Malley. Expertise-based performance measures in a virtual training environment. *Presence: Teleoperators and Virtual Environments*, 18 (6):449–467, 2009.

[15] Annie Chen. Context-aware collaborative filtering system: Predicting the user's preferences in ubiquitous computing. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1110–1111, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7.

[16] Yuehui Chen and Ajith Abraham. Hierarchical fuzzy systems. In *Tree-Structure based Hybrid Computational Intelligence*, volume 2 of *Intelligent Systems Reference Library*, pages 129–147. Springer Berlin Heidelberg, 2010.

[17] Vladimir Cherkassky. Fuzzy inference systems: A critical review. In Okyay Kaynak, LotfiA. Zadeh, Burhan Türkşen, and ImreJ. Rudas, editors, *Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications*, volume 162 of *NATO ASI Series*, pages 177–197. Springer Berlin Heidelberg, 1998.

[18] Nusrat Choudhury, Nicholas Gelinas-Phaneuf, Sebastien Delorme, and Rolando Del Maestro. Fundamentals of neurosurgery: Virtual reality tasks for training and evaluation of technical skills. *World Neurosurgery*, 80(5):e9 – e19, 2013. ISSN 1878-8750.

[19] R. Chowdhury, P. Bhandarkar, and M. Parashar. Adaptive qos management for collaboration in heterogeneous environments. In *Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM*, pages 90 –100, 2002. doi: 10.1109/IPDPS.2002.1015667.

[20] KajalT. Claypool and Mark Claypool. On frame rate and player performance in first person shooter games. *Multimedia Systems*, 13(1):3–17, 2007.

[21] C. Daniel. Use of half-normal plots in interpreting factorial two level experiments. *Technometrics*, 1(4):311–341, 1959.

[22] Regina de Araujo, Alessandro e Silva, and Glauco Todesco. Adapting multiuser 3d virtual environments to heterogeneous devices. *Journal of the Brazilian Computer Society*, 12:59–69, 2006. ISSN 0104-6500. 10.1007/BF03192388.

[23] Declan Delaney, Tomás Ward, and Seamus McLoone. On consistency and network latency in distributed interactive applications: a survey–part I. *Presence: Teleoperators and Virtual Environments*, 15:218–234, April 2006. ISSN 1054-7460.

[24] Declan Delaney, Tomás Ward, and Seamus McLoone. On consistency and network latency in distributed interactive applications: a survey–part II. *Presence: Teleoperators and Virtual Environments*, 15:465–482, August 2006. ISSN 1054-7460.

[25] P. Dev, D. Harris, D. Gutierrez, A. Shah, and S. Senger. End-to-end performance measurement of internet based medical applications. In *AMIA Fall Symposium 2002*, pages 205–209, 2002.

[26] Parvati Dev and Wm. Heinrichs. Learning medicine through collaboration and action: collaborative, experiential, networked learning environments. *Virtual Reality*, 12:215–234, 2008. ISSN 1359-4338.

[27] C. Diaz, H. Trefftz, and J. Bernal. Development of a laparoscopic surgical simulator to train a cholecystectomy procedure. In *Proceedings of the IV Iberoamerican Symposium in Computer Graphics*, pages 121–126, June 2009.

[28] C. Diaz, H. Trefftz, J. Bernal, and S. Eliuk. General algorithms for laparoscopic surgical simulators. *Revista Ingeniería Biomédica*, 4(8):57–70, 2010. ISSN 1909-9762.

[29] Christian Diaz, Helmuth Trefftz, Lucia Quintero, Diego A. Acosta, and Sakti Srivastava. Collaborative networked virtual surgical simulators (cnvss): Factors affecting collaborative performance. *Presence: Teleoperators and Virtual Environments*, 22(1):54 – 66, 2013.

[30] Christian Diaz, Helmuth Trefftz, Lucia Quintero, Diego A. Acosta, and Sakti Srivastava. Collaborative networked virtual surgical simulators (cnvss) implementing hybrid client-server architecture: Factors affecting collaborative performance. *Presence: Teleoperators and Virtual Environments*, 23(4):393 – 409, 2014.

[31] Francois Faure, Christian Duriez, Herve Delingette, Jeremie Allard, Benjamin Gilles, Stephanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stephane Cotin. Sofa: A multi-model framework for interactive physical simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29013-8.

[32] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, and Stéphane Cotin. Sofa: A multi-model framework for interactive physical simulation. In Yohan Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer Berlin Heidelberg, 2012.

[33] Hiroshi Fujinoki. On the support for heterogeneity in networked virtual environment. In *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, NetGames '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-589-4. doi: http://doi.acm.org/10.1145/1230040.1230060.

[34] Atul Gawande. Personal best: Top athletes and singers have coaches. should you? *The New Yorker - Annals of Medicine*, pages 1–9, 2011.

[35] Emanuele Goldoni and Marco Schivi. End-to-end available bandwidth estimation tools, an experimental comparison. In *Proceedings of the Second International Conference on Traffic Monitoring and Analysis*, TMA'10, pages 171–182, Berlin, Heidelberg, 2010. Springer-Verlag.

[36] Derek A. Gould, Nicholas Chalmers, Sheena J. Johnson, Caroline Kilkenny, Mark D. White, Bo Bech, Lars Lonn, and Fernando Bello. Simulation: Moving from technology challenge to human factors success. *CardioVascular and Interventional Radiology*, 35 (3):445–453, 2012. ISSN 0174-1551.

[37] Chris Gunn. *Using Haptics in a Networked Immersive 3D Environment*. PhD thesis, University of Western Australia, 2007.

[38] Chris Gunn, Matthew Hutchins, and Matt Adcock. Combating latency in haptic collaborative virtual environments. *Presence: Teleoperators and Virtual Environments*, 14:313–328, June 2005.

[39] David Gutierrez, Amol Shah, and Dale A. Harris. Performance of remote anatomy and surgical training applications under varied network conditions. In *Proceedings of ED–MEDIA World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 662–667, 2002.

[40] Felix G. Hamza, Anand Santhanam, Cali Fidopiastis, and Jannick P. Rolland. Distributed training system with high-resolution deformable virtual models. In *ACM–SE 43: Proceedings of the 43rd Annual Southeast Regional Conference*, pages 268–273, New York, NY, USA, 2005. ACM.

[41] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth. In *In Proceedings of Passive and Active Measurements (PAM) Workshop*, pages 14–25, 2002.

[42] J. Roger Jang. *Neuro-Fuzzy Modeling: Architectures, Analysis and Applications*. PhD thesis, University of California, 1992.

[43] J.S. Roger Jang. Rule extraction using generalized neural networks. In *Proceedings of the 4th IFSA World Congress*, pages 82–86, 1991.

[44] Caroline Jay, Mashhuda Glencross, and Roger Hubbold. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment. *ACM Transactions on Computer–Human Interaction*, 14(2), August 2007.

[45] Hakran Kim, Yongik Yoon, and Hwajin Park. Adaptation method for level of detail (lod) of 3d contents. In *Proceedings of the 2007 IFIP International Conference on Network and Parallel Computing Workshops*, NPC '07, pages 879–884, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2943-7.

[46] Bart Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice-Hall International, 1992.

[47] Takayuki Kuroda, Takuo Suganuma, and Norio Shiratori. Quvie/p : An effective qos control scheme for 3-d virtual environments based on users perception. *Advanced Information Networking and Applications, International Conference on*, 0:297–304, 2007. ISSN 1550-445X. doi: http://doi.ieeecomputersociety.org/10.1109/AINA.2007.119.

[48] Christian Laugier. *Interactions en Temps Reel de Tissu Mou Decoupe 3D et Retour dEffort*. PhD thesis, IPNG Grenoble, 2003.

[49] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller. i. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(2): 404–418, Mar 1990.

[50] R.V. Lenth. Response-surface methods in r, using rsm. *Journal of Statistics Software*, 32(7):1–17, 2009.

[51] Vincenzo Liberatore, M. Cenk Cavusoglu, and Qingbo Cai. GiPSiNet: An open source/open architecture network middleware for surgical simulations. In *Studies in Health Technology and Informatics*, volume 119, pages 316–321, 2006.

[52] Shiyong Lin, Roger J. Narayan, and Yuan-Shin Lee. Hybrid client-server architecture and control techniques for collaborative product development using haptic interfaces. *Computers in Industry*, 61(1):83–96, 2010. ISSN 0166-3615.

[53] Tsungnan Lin, Chiapin Wang, and Po-Chiang Lin. A neural-network-based context-aware handoff algorithm for multimedia computing. *ACM Trans. Multimedia Comput. Commun. Appl.*, 4 (3):17:1–17:23, September 2008.

[54] R.P. Lippmann. An introduction to computing with neural nets. *ASSP Magazine, IEEE*, 4(2):4–22, Apr 1987.

[55] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A survey of surgical simulation: applications, technology, and education. *Presence: Teleoperators and Virtual Environments*, 12(6):599–614, 2003.

[56] Soto-Miranda MA. and Ver Halen JP. Description and implementation of an ex vivo simulator kit for developing microsurgery skills. *Annals of Plastic Surgery*, 72(6):208 – 212, 2014.

[57] Anderson Maciel, Ganesh Sankaranarayanan, Tansel Halic, Venkata Arikatla, Zhonghua Lu, and Suvranu De. Surgical model-view-controller simulation software framework for local and collaborative applications. *International Journal of Computer Assisted Radiology and Surgery*, pages 1–15, 2010. ISSN 1861-6410. 10.1007/s11548-010-0527-3.

[58] Stefan Marks, John Windsor, and Burkhard Wünsche. Evaluation of game engines for simulated surgical training. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, GRAPHITE '07, pages 273–280, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-912-8. doi: http://doi.acm.org/10.1145/1321261.1321311. URL http://doi.acm.org/10.1145/1321261.1321311.

[59] James Marsh, Mashhuda Glencross, Steve Pettifer, and Roger Hubbold. A network architecture supporting consistent rich behavior in collaborative interactive applications. *IEEE Transactions on Visualization and Computer Graphics*, 12:405–416, 2006. ISSN 1077-2626.

[60] Italo Masiello. Why simulation-based team training has not been used effectively and what can be done about it. *Advances in Health Sciences Education*, 17(2):279–288, 2012. ISSN 1382-4996.

[61] Mathworks. Matlab: The language of technical computing. http://www.mathworks.com/products/matlab/, 2013-2014.

[62] Mathworks. Matlab compiler: Build standalone application from matlab programs. http://www.mathworks.com/products/compiler/, 2013-2014.

[63] Cesar Mendoza. *Soft tissue interactive simulations for medical applications including 3D cutting and force feedback*. PhD thesis, IPNG Grenoble, 2003.

[64] M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.

[65] Mecker Moller, John Karamichalis, Nikunj Chokshi, Haytham Kaafarani, and Heena P. Santry. Mentoring the modern surgeon. *Bulletin of American College of Surgeons*, 93(7):19–25, 2008.

[66] Douglas C. Montgomery. *Design and analysis of experiments*. John Wiley & Sons, 2008.

[67] Kevin Montgomery, Cynthia Bruyns, Joel Brown, Stephen Sorkin, Frederic Mazzella, Guillaume Thonier, Arnaud Tellier, Benjamin Lerman, and Anil Menon. Spring: A general framework for collaborative, real–time surgical simulation. In *Medicine Meets Virtual Reality '02 (MMVR)*, pages 23–26. IOS Press, 2002.

[68] Jesper Mosegaard. *Cardiac Surgery Simulation*. PhD thesis, University of Aahrus, 2006.

[69] NetDisturb. Impairment Emulator Software for IP Networks, feb 2012. URL http://www.zti-telecom.com/EN/NetDisturb.html.

[70] Jonathan Norman and Felix G. Hamza-Lup. Challenges in the deployment of visuo-haptic virtual environments on the Internet. In *Proceedings of the 2010 Second International Conference on Computer and Network Technology*, ICCNT '10, pages 33–37, Washington, DC, USA, 2010. IEEE Computer Society.

[71] Oliver Otto, Dave Roberts, and Robin Wolff. A review on effective closely-coupled collaboration using immersive cve's. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, VRCIA '06, pages 145–154, New York, NY, USA, 2006. ACM. ISBN 1-59593-324-7.

[72] Kyoung Shin Park and Robert V. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. In *Proceedings of the IEEE Virtual Reality*, VR '99, pages 104–109, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0093-5.

[73] Li Qilin, Zhen Wei, and Zhou Mintian. Research on key issues for the next generation middleware technology. In *Proceedings of the 2009 International Forum on Information Technology and Applications - Volume 01*, pages 776–779, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3600-2. doi: 10.1109/IFITA.2009.23.

[74] Jing Qin, Kup-Sze Choi, Wai-Sang Poon, and Pheng-Ann Heng. A framework using cluster–based hybrid network architecture for collaborative virtual surgery. *Computer Methods and Programs in Biomedicine*, 96(3):205 – 216, 2009.

[75] Jing Qin, Kup-Sze Choi, and Pheng-Ann Heng. Collaborative simulation of soft-tissue deformation for virtual surgery applications. *Journal of Medical Systems*, 34:367–378, 2010. ISSN 0148-5598.

[76] Jing Qin, Kup-Sze Choi, Wai-Man Pang, Zhang Yi, and Pheng-Ann Heng. Collaborative virtual surgery: Techniques, applications and challenges. *The International Journal of Virtual Reality*, 9:1 – 7, 2010.

[77] A. Quiroz. Metodo para el control optimo de recursos para clientes heterogeneos en ambientes virtual colaborativos. Technical report, Universidad EAFIT, 2004.

[78] R-Project. For Statistical Computing, feb 2012. URL http://www.r-project.org/.

[79] I. Rejer. Precision of an expert fuzzy model. In ZdzisawS. Hippe, JuliuszL. Kulikowski, and Teresa Mroczek, editors, *Human – Computer Systems Interaction: Backgrounds and Applications 2*, volume 98 of *Advances in Intelligent and Soft Computing*, pages 33–47. Springer Berlin Heidelberg, 2012.

[80] Reznick RK and MacRae H. Teaching surgical skills - changes in the wind. *The New England Journal of Medicine*, 355(25):2664 – 2669, 2006.

[81] John L. Rombeau, Amy Goldberg, and Catherine Loveland-Jones. *Surgical Mentoring: Building Tomorrow's Leaders*. Springer Verlag, 2010.

[82] Frank Rosenblatt. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books, 1962.

[83] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[84] Andrae Sansregret, Gerald M. Fried, Harrith Hasson, Dennis Klassen, Maryse Lagac√©, Robert Gagnon, Stephen Pooler, and Bernard Charlin. Choosing the right physical laparoscopic simulator? comparison of lts2000-ism60 with mistels: validation, correlation, and user satisfaction. *The American Journal of Surgery*, 197(2):258 – 265, 2009. ISSN 0002-9610.

[85] K. Schwenk, G. Voss, and J. Behr. A system architecture for flexible rendering back-ends in distributed virtual reality applications. In *Cyberworlds (CW), 2012 International Conference on*, pages 7–14, Sept 2012.

[86] Christopher Sewell. *Automatic Performance Evaluation in Surgical Simulation*. PhD thesis, Stanford University, 2007.

[87] A.A. Sharifloo, M. Mirakhorli, M. Esmaeili, and A.T. Haghighat. A leader election algorithm for clustered groups. In *Industrial and Information Systems, 2007. ICIIS 2007. International Conference on*, pages 1–4, Aug 2007.

[88] Xiaojun Shen, Jilin Zhou, Abdelwahab Hamam, Saeid Nourian, Naim R. El-Far, François Malric, and Nicolas D. Georganas. Haptic-enabled telementoring surgery simulation. *IEEE MultiMedia*, 15(1):64–76, 2008. ISSN 1070-986X. doi: http://dx.doi.org/10.1109/MMUL.2008.9.

[89] Chia-Yen Shih, Jie Hu, Jinhwan Lee, Raymond Klefstad, and Doug. Tolbert. Marco: A middleware architecture for distributed multimedia collaboration. *Multimedia, International Symposium on*, 0:366–373, 2005. doi: http://doi.ieeecomputersociety.org/10.1109/ISM.2005.76.

[90] Shervin Shirmohammadi and Nicolas D. Georganas. An end–to–end communication architecture for collaborative virtual environments. *Computer Networks*, 35(2-3):351–367, 2001.

[91] Sandeep Singhal and Michael Zyda. *Networked virtual environments: design and implementation*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999. ISBN 0-201-32557-8.

[92] Lea Skorin-Kapov and Maja Matijasevic. Dynamic qos negotiation and adaptation for networked virtual reality services. In *Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, WOWMOM '05, pages 344–351, Washington, DC, USA, 2005. IEEE Computer Society. doi: http://dx.doi.org/10.1109/WOWMOM.2005.37.

[93] Rima Tfaily Souayed, Dominique Gaiti, Wai Yu, Gordon Dodds, and Alan Marshall. Experimental study of haptic interaction in distributed virtual environments. In *Proceedings of EuroHaptics '04*, pages 260–266, 2004.

[94] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 39–44, New York, NY, USA, 2003. ACM. ISBN 1-58113-773-7.

[95] Kenneth Sundaraj. *Real–Time Dynamic Simulation and 3D Interaction of Biological Tissue: Application to Medical Simulators*. PhD thesis, IPNG Grenoble, 2004.

[96] G. Szekely, Ch. Brechbhler, R. Hutter, A. Rhomberg, and P. Schmid. Modelling of soft tissue deformation for laparoscopic surgery simulation. In WilliamM. Wells, Alan Colchester, and Scott Delp, editors, *Medical Image Computing and Computer-Assisted Interventation, MICCAIí98*, volume 1496 of *Lecture*

Notes in Computer Science, pages 550–561. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-65136-9.

[97] T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operators control actions. In *Proceedings of the Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, pages 1–7, 1983.

[98] Sze-Wai Tang, King-Lung Chong, Jing Qin, Yim-Pan Chui, S.S.-M. Ho, and Pheng-Ann Heng. ECiSS: A middleware based development framework for enhancing collaboration in surgical simulation. In *IEEE International Conference on Integration Technology, 2007. ICIT '07*, pages 15–20, 2007.

[99] Charison Tay, Ankur Khajuria, and Chinmay Gupte. Simulation training: A systematic review of simulation in arthroscopy and proposal of a new competency-based training framework. *International Journal of Surgery*, 12(6):626 – 633, 2014. ISSN 1743-9191.

[100] Vicenc Torra. A review of the construction of hierarchical fuzzy systems. *International Journal of Intelligent Systems*, 17(5): 531–543, 2002.

[101] Helmuth Trefftz. *System-wide constraints and user preferences in collaborative virtual environments*. PhD thesis, Rutgers University, New Brunswick, NJ, USA, 2002.

[102] Helmuth Trefftz, Ivan Marsic, and Michael Zyda. Handling heterogeneity in networked virtual environments. *Presence: Teleoperators and Virtual Environments*, 12:37–51, February 2003. ISSN 1054-7460.

[103] Y. Tsukamoto. An approach to fuzzy reasoning method. In *Advances in Fuzzy Set Theory and Applications*, pages 137–149, 1979.

[104] E.G. Verdaasdonk, L.P. Stassen, M.P. Schijven, and J. Dankemlman. Construct validity and assessment of the learning curve for the simendo endoscopic simulator. *Surgical Endoscopy*, 21(8): 1406–1412, 2007.

[105] Jun Wang and Gregory M. Provan. Generating application-specific benchmark models for complex systems. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence,AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 566–571, 2008.

[106] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974.

[107] Stephen Workman, Gerard Parr, Philip Morrow, and Darryl Charles. Modelling event communication to enable adaptive behaviour in resource- constrained distributed virtual environments. In *Proceedings of the International Conference on Autonomic and Autonomous Systems*, ICAS '06, pages 23–, Washington, DC, USA, 2006. IEEE Computer Society. doi: http://dx.doi.org/10.1109/ICAS.2006.36. URL http://dx.doi.org/10.1109/ICAS.2006.36.

[108] Jong yi Hong, Eui ho Suh, and Sung-Jin Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509 – 8522, 2009.

[109] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[110] Lofti A. Zadeh. Fuzzy logic. *Computer*, 21(4):83–93, April 1988.

[111] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3(1):28–44, Jan 1973.

[112] Xiaolan Zhang. *Application-Specific Benchmarking*. PhD thesis, Harvard University, 2001.