Algorithms about early multimodal (haptic, visual, acoustic) scenarios and rendering

Deliverable D2.1

List of Partners: Swiss Federal Institute of Tech. Zurich, CH
Hocoma AG, Volketswil, CH
University of Ljubljana, SLO
Universitat Politècnica de Catalunya, SPA
Neurological Clinic Bad Aibling, D

Document Identifier: MIMICS-D2.1-pu.pdf
Version: 4.0
Date: 2008-09-30
Organisation: UPC
Deliverable: 2.1
Milestone: 2.1
Work Package: 2
Task: 2.1, 2.2, 2.3, 2.4
Dissemination: public
Authors: Alexander König, Marc Bolliger, Martin Simnacher, Robert Riener, M. Mihelj, A. Olenšek, M. Munih, J. Podobnik, J. Ziherl, L. Jensterle, Jason Kastanis, Mel Slater
Approved by: Robert Riener

Table of Contents

1 Summary

This is the D2.1 deliverable of the MIMICS project, funded by the European Community's Seventh Framework Program under Grant Agreement n° 215756. This report describes the early developments of a multimodal immersive interactive display system. The multimodal system consists of three different rendering modalities, comprising haptic, visual and acoustic cues. Interaction and immersion are achieved by the modeling of engaging scenarios that will motivate the user of the system to perform the selected tasks. The report will present details on the individual elements, as well as a complete view of the system architecture.

2 Introduction

MIMICS will require a complex system consisting of multiple hardware and software components. It includes a variety of physiological signals, such electrocardiogram (ECG), galvanic skin response (GSR), electromyogram (EMG) and respiration (RESP), as well as tracking and haptic signals. Signals from the robotic device like positioning and forces are included within the haptic term. The visualization part will be on a standard virtual reality device, for example a head mounted display (HMD), a power wall or a cave automated virtual environment (CAVE). Further output is channeled through audio and haptic devices.

Design of system architecture
The design demonstrates how the rendering and the scenario modeling will interact with other modules of the system. Such a framework will allow flexible scenarios and hardware setups to be developed. The design of different experimental scenarios will be achieved with minimum effort, simply by replacing modules and not the complete system.

The main part of system is split in four basic parts which repeat until the task is completed. It is this part of the system that controls the patient interaction. The entry point of the system is the task manager describing the scenario and general flow of the task. This will connect with the output devices and render the current state within the virtual environment. The user of the VR system will then react in some way and input will be measured. This will affect the scenario of the VR task either directly or through the physio processing engine. For the graphical representation refer to Figure 1.

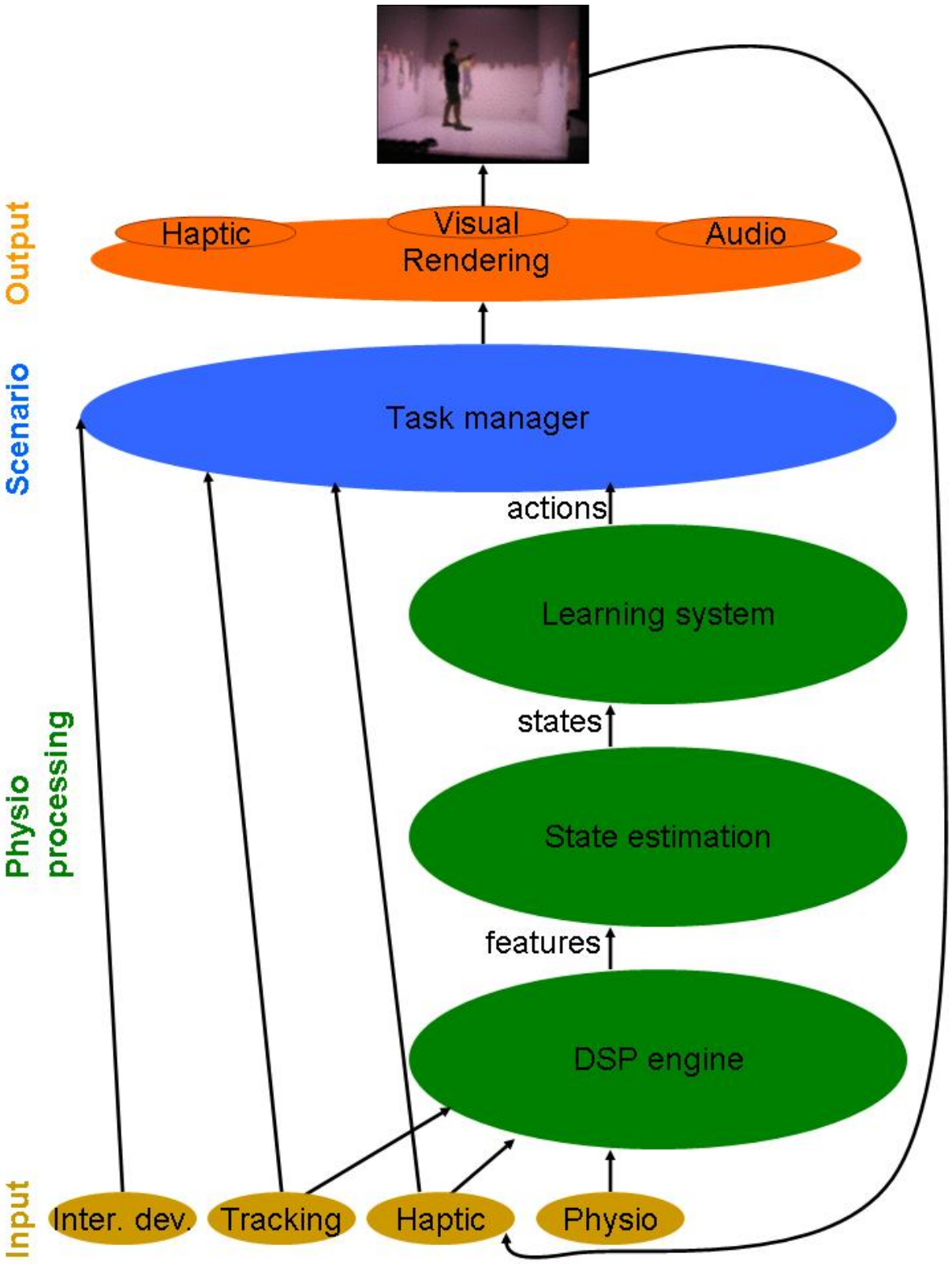


Figure 1 MIMICS system architecture

The input system consists of the physiological signals and the VR signals. In general physiological signals affect the flow of the task indirectly while the VR ones have a more direct effect. The relevant physiological signals to this project are ECG, EMG, GSR and RESP. The signals coming from the haptic and tracking devices could control the application both directly and indirectly, while the interaction device would have a more direct influence on the flow of the application.

The raw signals coming from the input devices will be processed by the digital signal processing (DSP) unit to extract features. These features will include for example heart rate variability (HRV) and GSR peak rate. The extraction of these features from the raw data should run in real-time.

The features outputted from the DSP unit will be analyzed by the state estimation unit. The main purpose of this unit is to obtain physiological and psychological states such as arousal, valence and physical effort level. Analysis of these signals will possibly require the implementation of fusion algorithms that combine multiple features in to single output states.

The learning system will incorporate algorithms that aim to adjust the parameters of the task by selecting actions that will produce the desired physiological and psychological states. The actions can include positioning of avatars within the virtual environment, task complexity, appearance of the environment, changes in the scenario flow, and alternations in the sound, difficulty and duration.

The task manager module contains the main definition for each task, ie. the scenario of how the user will interact with the environment and the environment itself, as well as the specific goals of each task. It will accept direct input from the user of the system such as positioning provided by the tracking, robotic or haptic devices. At the same time some aspects of the scenario will be controlled indirectly through the physiological processing module.

The final module of the system is the rendering part. It is the output to the user and it consists of three units, haptic, visual and audio. Tools for visual rendering are described in Section 5.2.

3 Scenario modeling

3.1 Introduction to the goals of scenario modeling

Two types of scenarios are introduced in the following sections. Modality specific scenarios developed for the HapticMaster and the Lokomat modeling rehabilitation exercises and general purpose scenarios examining aspects common to both modalities. The goals of all scenarios are to develop appropriate tasks and simulating environments for the patients. At the same rendering module of the system architecture (Figure 1) as well as other modules are being advanced.

3.2 Scenario modeling for the HapticMaster

For training patients two scenarios were developed and are designed for training arm movement and grasping simultaneously. In the following sections these two scenarios are described: the wavy tunnel task and the pick and place task.

3.2.1 Scenario 1: For internal dissemination only

Since this section contains confidential intellectual property, it has been removed from the public version of the deliverable.

3.2.2 Scenario 2: Wavy tunnel task

In scenario 2 the static tunnel is placed in an empty space. Tunnel is placed in the frontal plane. It is wavy and has a changing diameter. The working space is shown on the floor. A round ball marks the position of the arm. With the grasp force the patient is changing the size of the ball. The object of the task is to travel the whole tunnel and reach its end. The Figure 2 shows the start of the task.

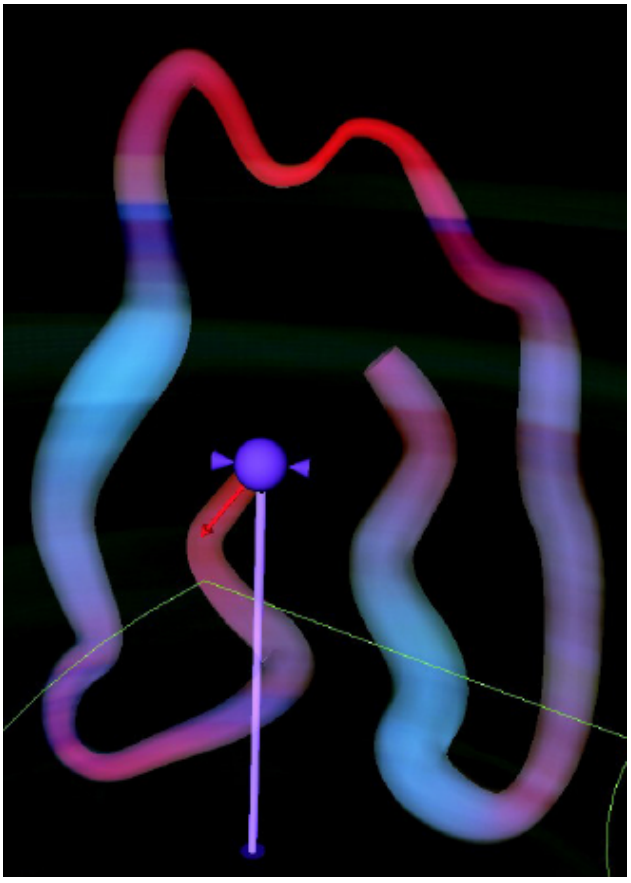


Figure 2 The tunnel in the frontal plane and the ball at the start of the task

3.2.3 Scenario 3: Pick and place task

Scenario 3 consists of a stall placed on the floor, an apple and some trees. The stall is placed on the edge of the working space. The apple is located on the floor. The

trees are placed around the working space. The patient picks the apple and settles it down at the marked point on the stall. After the successful set down, the apple falls from random place to the floor. The Figure 3 shows the scenario during transport of the apple to its end place.

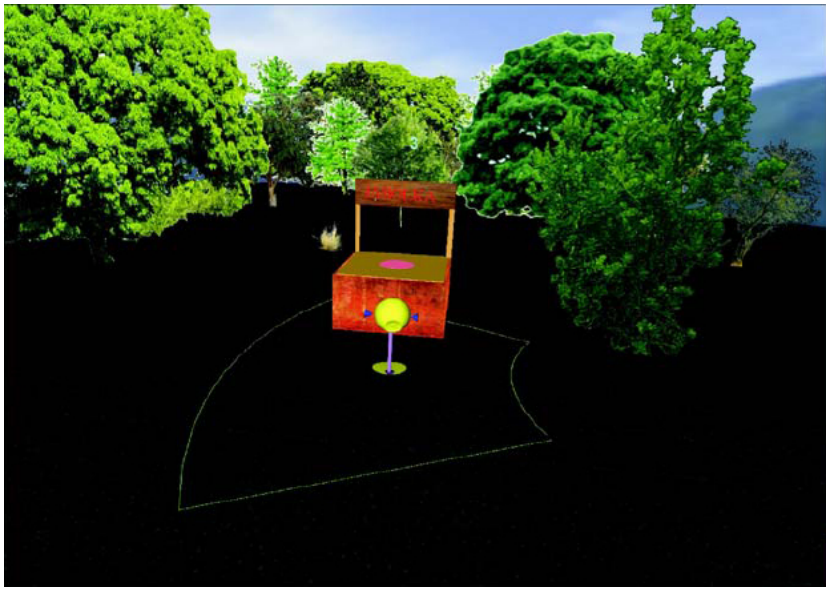


Figure 3 The scenario of the pick and place task

3.3 Scenario modeling for the Lokomat

For the driven gait orthosis three early multimodal (haptic, visual, acoustic) scenarios were developed and tested with healthy volunteers. In the following sections the three scenarios are described.

3.3.1 Scenario 1: Walking through a city

Scenario 1 consists of a street with rows of houses on each side. Trees are placed on both sidewalks every 40 meters. The avatar walks in the middle of the street. The scenario can be displayed in 6 different views and has an approximated area of 0.64 km^2. Figure 4 shows the scenario from ego-view perspective.

Figure 4 Ego-view perspective of scenario 1

3.3.2 Scenario 2: Walking over a canyon

In scenario 2 the avatar walks from a broad street towards a narrow beam. The beam leads over a narrow canyon (Figure 5), that is 200 m wide. The scenario can be displayed in 6 different views.

Figure 5 Avatar walking on a narrow beam over a canyon (side view)

3.3.3 Scenario 3: Soccer game

The soccer scenario takes place in scenario 1. The avatar is controlled by the patient, who is able to kick a soccer ball. The soccer ball is placed in the middle of the street. In order to enhance the patients' motivation, an opponent can be placed into the scenario (Figure 6).

Figure 6 The avatar controlled by the patient (white shirt) plays soccer against a virtual opponent

3.4 General purpose scenario modeling

Scenarios have been modeled to validate aspects common to both the HapticMaster and Lokomat systems. These scenarios will examine the interaction of the user with virtual characters. The aim is to build characters that will engage and encourage the user to perform the tasks of a given scenario. In particular, the presented scenarios

will examine the extent to which the user feels the bodily connection to the avatar and in the second one, the ability of the learning system to drive humans in a particular state. The first scenario's scope is to examine the effects of using a physical connection between avatars as well as multisensory correlations that can potentially increase the immersion to the environment particularly for the ecco-centric viewpoint. The second scenario is the starting point for the creation of intelligent avatars that interact with the user. It will also be looking at the interface between the learning and the rendering systems providing useful information on the communication between them.

3.4.1 Scenario 1: Connection to an avatar.

What does the user see?

The subject is placed in a room where they can see their own virtual body, another body in front of them, and a chord connecting both bodies (Figure 7). Both avatars are wearing the same clothes. The chord connects the stomach of the avatar representing the real subject and the back of the second avatar.

The system can change dynamically the chord's length and radius, so that different parameters can be tested throughout the experiment. Those values are changed by a GPU shader responsible for the chord rendering.

In addition to that, the shader creates some special effects to provide visual feedback to the subject when some virtual tapping applies to the chord's surface. The tapping feedback is given in three different ways: the virtual tapping ball itself, some concentric ripples starting at the collision point between the ball and the cord, and a shadow according to the ball position.

What can the user do?

The subject is wearing a Head Mounted Display (HMD) with a tracker attached or is looking at a large stereo screen system (in which case the egocentric virtual body is not needed). He or she can freely move the upper part of their body. As the subject moves, he can observe his own virtual body moving in the same way. The system can apply the same inverse kinematics (IK) to the avatar the subject sees in front of him. This way, the second avatar mimics the subject's movements.

What will the subject experience?

There is synchronised tapping on the subject's body and on the body of the avatar. The hypothesis is that the chord will add to the sense of connection between the person and the avatar that they can see in front. There is already some published literature that suggests that the person will feel themselves to be where the avatar is located [3], however, we have been unable to reproduce those results in our lab, and therefore we are following the idea of the chord to see if this improves the connection.

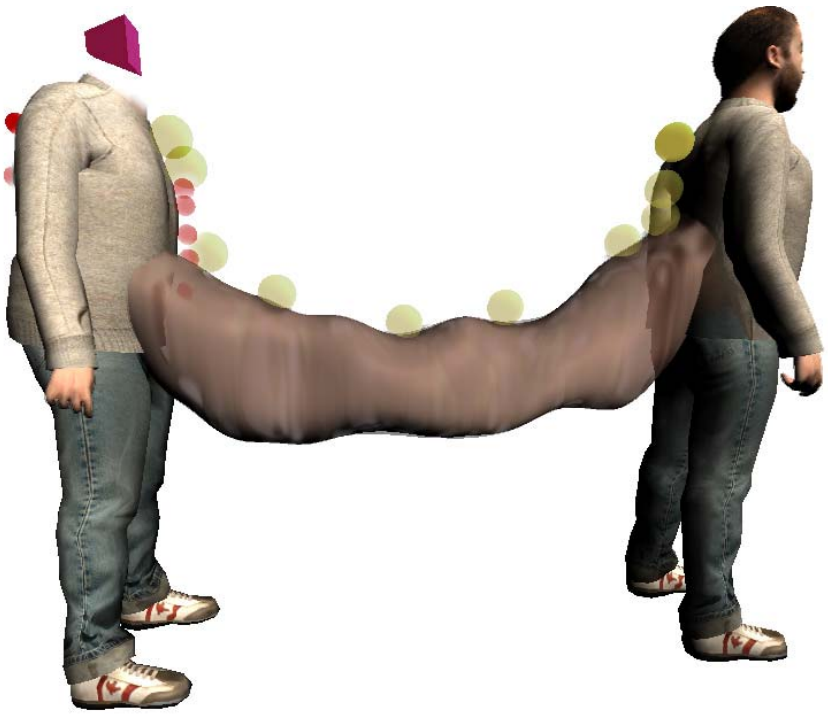


Figure 7 The subject is placed in a room where they can see their own virtual body, another body in front of them, and a chord connecting both bodies. The chord connects the stomach of the avatar representing the real subject and the back of the second avatar.

3.4.2 Scenario 2: Walking by machine learning

The subject is located in a virtual environment and he can observe the surrounding space through an ego perspective. Within the virtual environment he can see both his own body and an avatar. The avatar is controlled by the learning system and its goal is to move the subject to a specific location in a space by performing a series of actions. The choice of actions, from a pre-determined set of possibilities, is entirely dependent on the machine learning system.

The aim of this experiment is to demonstrate that the learning system can successfully change the state of the human subject, in this case his position in space (physical or virtual). The development of this scenario will lead to a more detailed understanding of the connections between the rendering, task manager and learning modules enabling the improvement and refinement of a framework for the MIMICS system and it marks the first steps to the creation of intelligent and engaging computer controlled avatars. The same technology could be used to encourage, provide motivation to, and enhance presence of participants, through the agent having goals based on other aspects of the subject's state (not just their position in space).

4 Robotic control and haptic rendering

4.1 Introduction to the goals of robotic control and haptic rendering

Haptic rendering is defined as the process of computing and generating forces in response to user interactions with virtual objects. The process consists of two main steps: collision detection and contact force computation.
These forces are fed back to the patient for various reasons. First of all, the haptic sense can increase the degree of realism of the virtual environment. An object can hardly appear real if it cannot be touched. Secondly, haptic force feedback can guide the patient on a given trajectory by bounding the workspace of the robot to desired areas and thereby give the patient a clue on desired behavior.

4.2 Robotic control and haptic rendering for the HapticMaster

4.2.1 Introduction

The HapticMaster has three active degrees of freedom and a workspace as shown in Figure 8. Its world coordinate system (WCS) is placed in the center of the workspace. Tool coordinate system (TCS) is attached at the robot end-effector (three axis force sensor) as shown in Figure 9. Force directions correspond to the TCS axes.

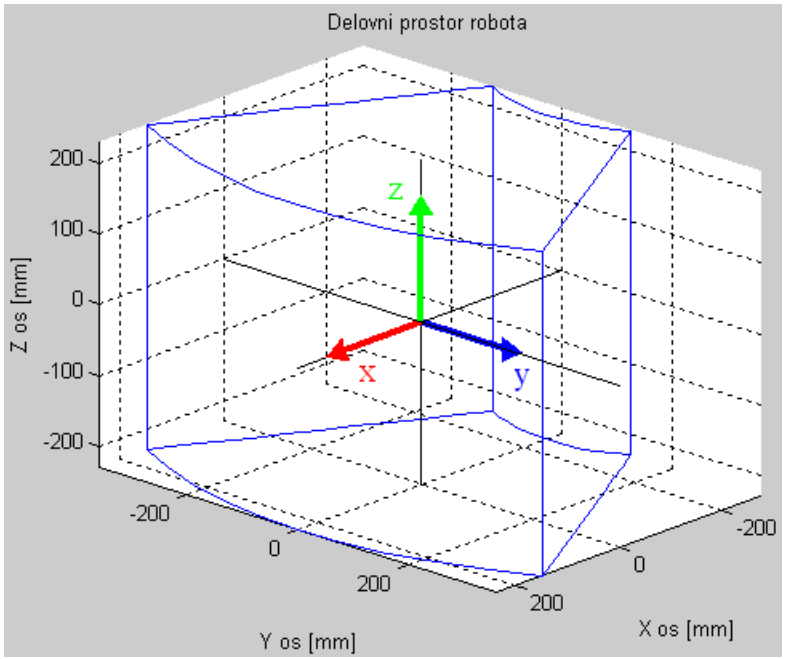


Figure 8 HapticMaster robot workspace

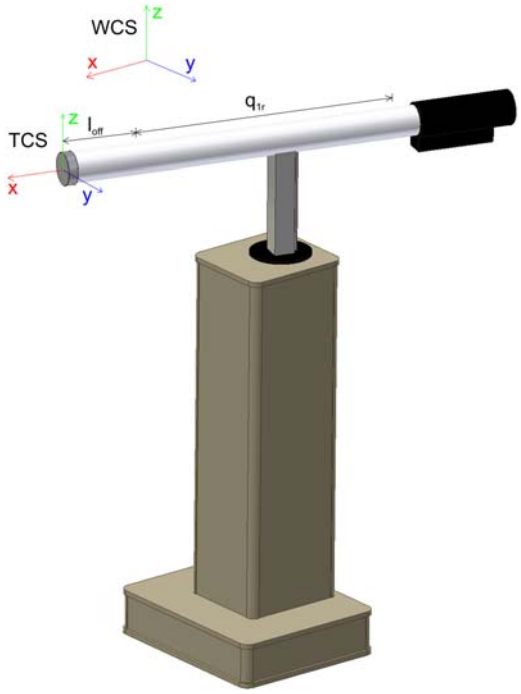


Figure 9 HapticMaster robot, world coordinate system (WCS), tool coordinate system (TCS)

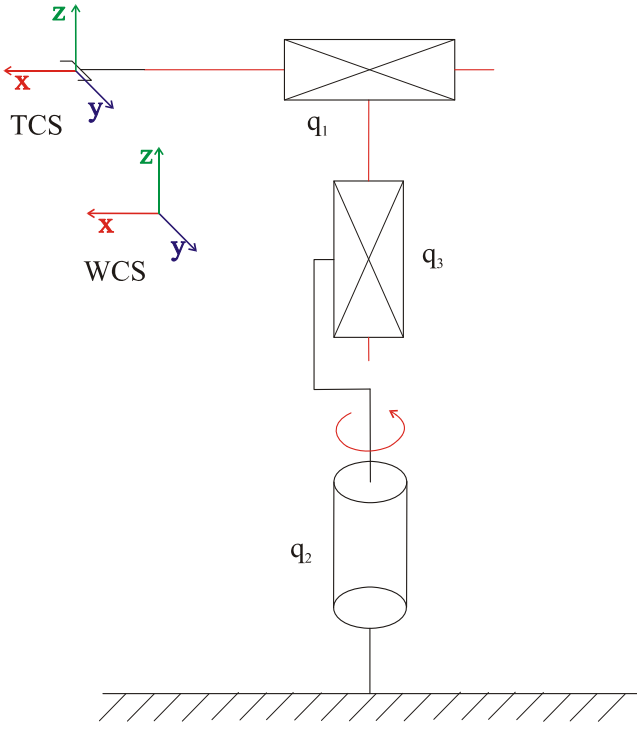


Figure 10 Robot coordinate systems and degrees of freedom

Robot control system is implemented in Matlab xPC Target environment. In the next sections HapticMaster Simulink library blocks will be introduced.

4.2.2 Robot direct kinematics

Direct kinematics block enables computation of the pose of the robot end-effector based on the joint measured angles. Input to the block are joint angles, output is the end-effector pose.

Figure 11 HapticMaster direct kinematics block

Computation of the robot direct kinematics model is based on Figure 12 and Figure 13:

$$\begin{aligned} x &= (q_1 + d_1)\cos(q_2) - d_1 \\ y &= (q_1 + d_1)\sin(q_2) \\ z &= q_3 \\ d_1 &= \frac{q_{1r}}{2} + l_{off} \end{aligned} \quad , \qquad (1)$$

where q_1 represents horizontal translation, d_1 is the offset of the WCS (see Figure 9 for details) from the rotation axis, q_2 is the angle of horizontal rotation, and q_3 represents vertical translation.

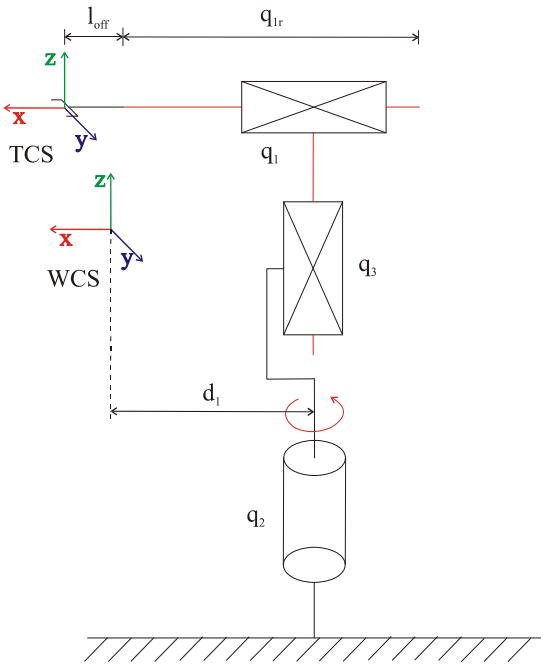


Figure 12 Direct kinematic model

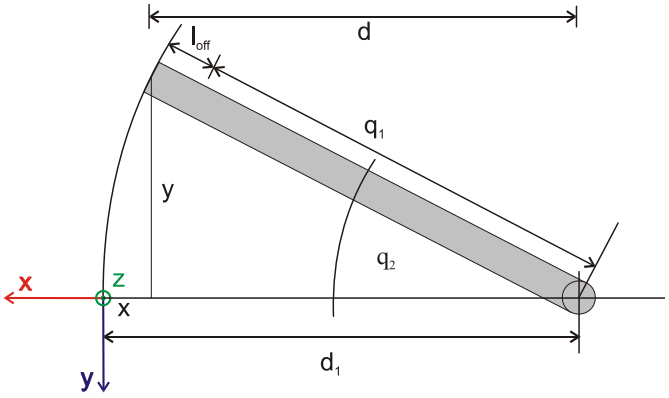


Figure 13 Top view on the robot

4.2.3 Robot inverse kinematics

Inverse kinematics block enables computation of joint angles based on the end-effector pose. Inverse kinematics is computed based on the relations in Figure 12

$$
\begin{aligned}
q_1 &= \sqrt{(x+d_1)^2 + y^2} - d_1 \\
q_2 &= arctg\left(\frac{y}{x+d_1}\right) \\
q_3 &= z \\
d_1 &= \frac{q_{1r}}{2} + l_{off}
\end{aligned}
\qquad \textbf{(2)}
$$

Robot end effector position is defined with *x, y* and *z* coordinates.

The input to the inverse kinematics block is the end-effector pose, output are the corresponding joint space positions.

Figure 14 Inverse kinematics block

4.2.4 Robot Jacobian matrix

Jacobian matrix relates joint space velocity with world space velocities or world space forces with joint space torques (transposed Jacobian). Robot Jacobian was computed from the following relation

$$\mathbf{J} = \frac{d\mathbf{p}}{d\mathbf{q}}. \qquad (3)$$

Size of the Jacobian matrix is 3×3 in the case of HapticMaster robot. End-effector orientation is not considered:

$$\mathbf{J} = \begin{bmatrix} \frac{dx}{dq_1} & \frac{dx}{dq_2} & \frac{dx}{dq_3} \\ \frac{dy}{dq_1} & \frac{dy}{dq_2} & \frac{dy}{dq_3} \\ \frac{dz}{dq_1} & \frac{dz}{dq_2} & \frac{dz}{dq_3} \end{bmatrix} \qquad (4)$$

Input to the Jacobian matrix block are robot joint positions, output of the block is the Jacobian matrix.

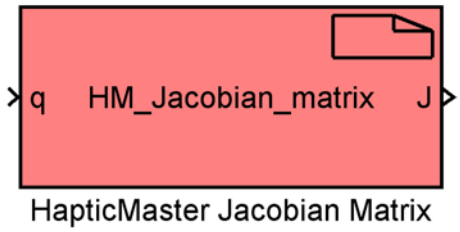


Figure 15 Jacobian matrix block

4.2.5 Joint space PD controller

Joint space PD controller is the basic robot control system that enables joint space trajectory tracking. Block scheme of the controller is shown in Figure 16.

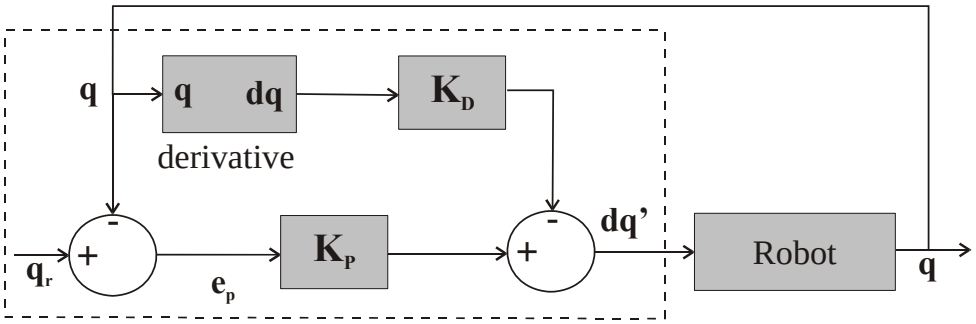


Figure 16: Block scheme of the joint space PD controller.

The output of the PD controller is defined as:

$$\mathbf{dq}'(t) = \mathbf{K_P}\ (\mathbf{q}(t) - \mathbf{q_r}(t)) - \mathbf{K_D}\ \frac{\mathbf{dq}(t)}{dt}, \qquad \textbf{(5)}$$

where $\mathbf{dq}'(t)$ is the required robot velocity, $\mathbf{q_r}$ is the reference position, $\mathbf{K_P}$ is the proportional gain, $\mathbf{K_D}$ is the velocity gain.

PD controller block has two inputs and one output:
q – joint space positions

- *qr* – reference joint space positions
- *dot_q* – required robot velocities, output to the manipulator.

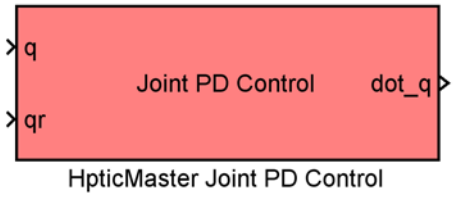


Figure 17 Joint space PD controller block

The mask of the block enables setting of two parameters (see Figure 18):

- Position gain $\mathbf{K_P}$ – vector defines position gains for all three axes
- Velocity gain $\mathbf{K_D}$ – vector defines velocity gain for all three axes.

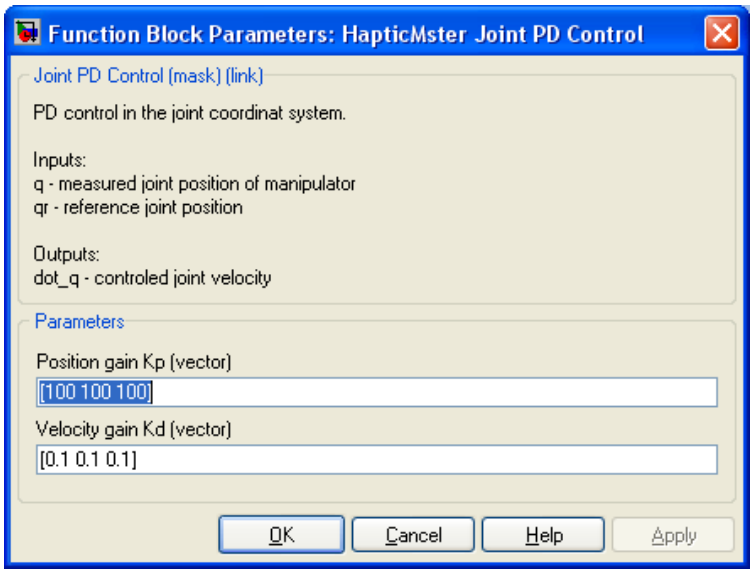


Figure 18 Joint space PD controller block mask

4.2.6 World space PD controller

World space PD controller enables tracking of reference world space positions. Controller is designed based on the joint space PD controller and the use of inverse kinematics block. The controller is shown in Figure 19.

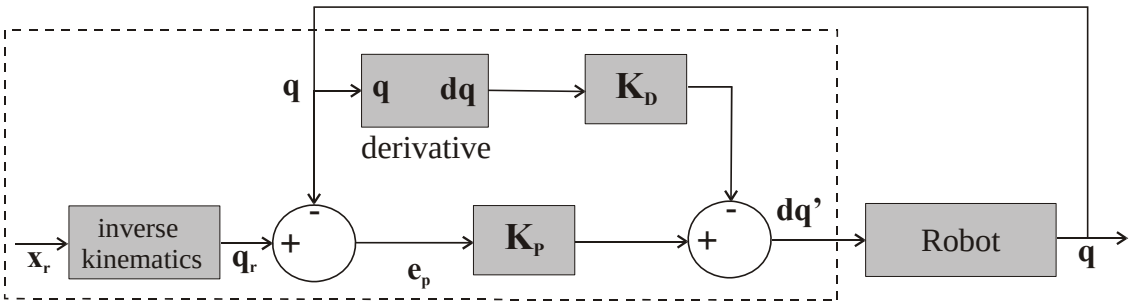


Figure 19 Block scheme of the world space PD controller

PD controller block has two inputs and one output:

- *q* – joint space positions
- x_r – reference world space position of the robot end-effector
- *dot_q* – required robot velocities, output to the manipulator.

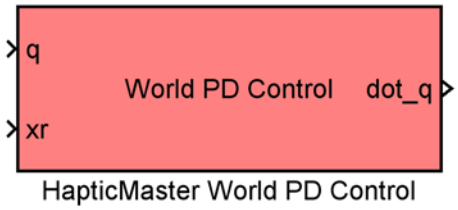


Figure 20 Joint space PD controller block

The mask of the block enables setting of two parameters (see Figure 21):

- Position gain $\mathbf{K}_P$ – vector defines position gains for all three joint axes
- Velocity gain $\mathbf{K}_D$ – vector defines velocity gain for all three joint axes.

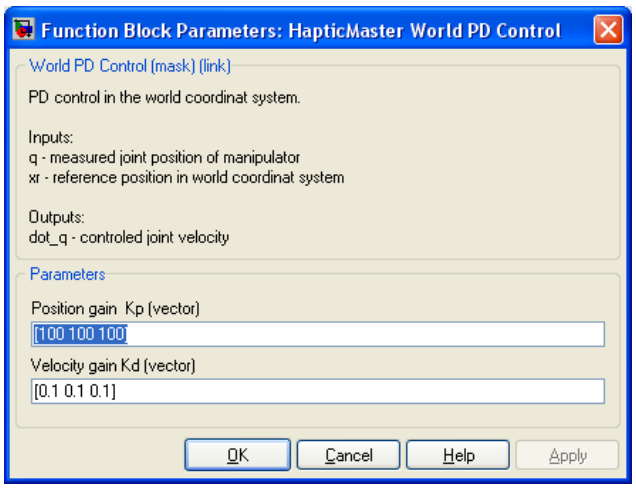


Figure 21 World space PD controller block mask

4.2.7 Admittance control

Admittance control mode is the basic mode used for haptic rendering. Robot end-effector behaves as a point mass *m.* Two forces act on this mass, user's physical interaction force $\mathbf{F}_{meas}$ and the force of the virtual environment $\mathbf{F}_{virt}$. Force $\mathbf{F}_{meas}$ is measured using a force sensor attached at the robot end-effector. Force $\mathbf{F}_{virt}$ is the

output force of the programmed virtual environment. Due to the particular kinematic structure of the HapticMaster robot it is possible to write three admittance controllers, one for each joint axis.

Admittance control for the translational joint can be written as

$$\ddot{q} = (F_{meas} + F_{virt}) / m$$
$$\dot{q} = \int \ddot{q}\ dt = \int (F_{meas} + F_{virt}) / m\ dt \text{,} \qquad \textbf{(6)}$$

While for the rotational joint (here we have to consider the radius of rotation)

$$\ddot{q} = (F_{meas} + F_{virt}) / (m\ r)$$
$$\dot{q} = \int \ddot{q}\ dt = \int (F_{meas} + F_{virt}) / (m\ r)\ dt \text{.} \qquad \textbf{(7)}$$

All parameters are scalar values. Virtual mass *m* has to be as small as possible to increase the transparency of the haptic interaction. On the other hand, decreasing the mass value increases robot instability. Stable operation was achieved using a virtual mass of 3 kg. Admittance control for a single axis is shown in Figure 22.

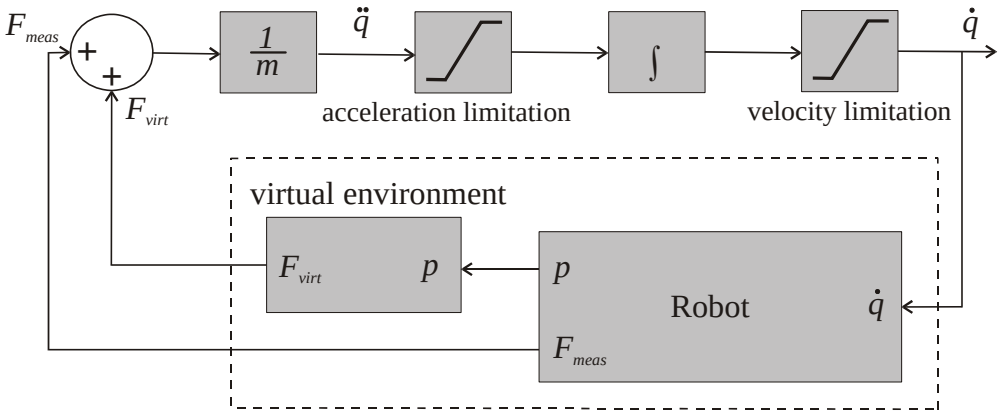


Figure 22 Block scheme of a single axis admittance model

Admittance control block has three inputs and one output:

- *F_meas* – measured force on the force sensor (interaction force between the subject and the robot).
- *F_virt* – virtual environment output force.
- *q* – robot angle joint positions.

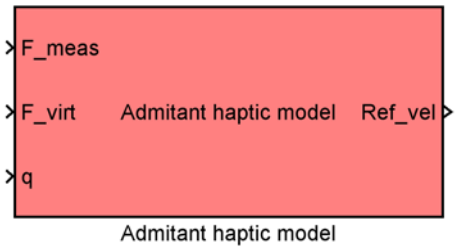


Figure 23 Admittance control block

Admittance control block mask enables setting of three parameters:

- Virtual mass of the robot end-effector (at least 3 kg is required for stable operation).
- Maximum joint accelerations.
- Maximum joint velocities.

Admittance control block mask is shown in Figure 24.

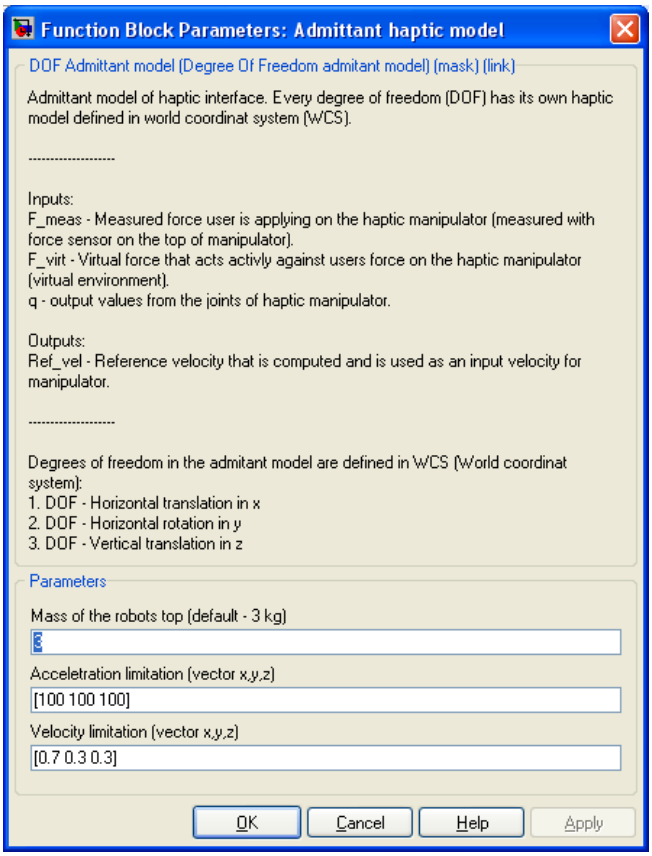


Figure 24 Admittance control block mask

4.2.8 Haptic objects

Interactive virtual environment is built based using haptic object blocks. These blocks compute haptic interaction between the user and the haptic environment. Based on the interaction and the physical model the movement of haptic objects is computed. Output of the block is the virtual force later used in the admittance control block.

The following haptic objects are implemented: cube, sphere, cylinder, and wall. For more complex objects these basic blocks can be used as bounding geometries for haptic computations.

Haptic object is defined with the following parameters:

1) dimensions – dimensions of outside walls of the particular object,
2) position and orientation – each object has its own coordinate system attached to its center of mass; pose of this local coordinate system $\mathbf{T_{obj}}$ relative to WCS defines position and orientation of the object,
3) mass – mass defines object inertia,
4) viscous friction – viscous friction felt when touching the object,
5) stiffness – stiffness of object walls,
6) initial translational and rotational velocities.

4.2.8.1 Haptic collision

Haptic collision is considered between the robot end-effector (**HIP** – haptic interaction point). The collision first needs to be detected and the interaction forces between the **HIP** and the object are computed. Forces are then applied to the object on one side and through the HapticMaster robot to the user on the other side.

Consider a haptic object cube in a pose defined by a transformation matrix $\mathbf{T}_{obj}$ relative to WCS. In order to simplify collision detection between **HIP** and the object, **HIP** is first transformed from WCS to the object local coordinate frame (see Figure 25)

$$\mathbf{HIP}' = \mathbf{T}_{obj}^{-1}\ \mathbf{HIP} \qquad \textbf{(8)}$$

In order to detect collision it is only necessary to compare absolute value of **HIP'** coordinates with cube dimensions.

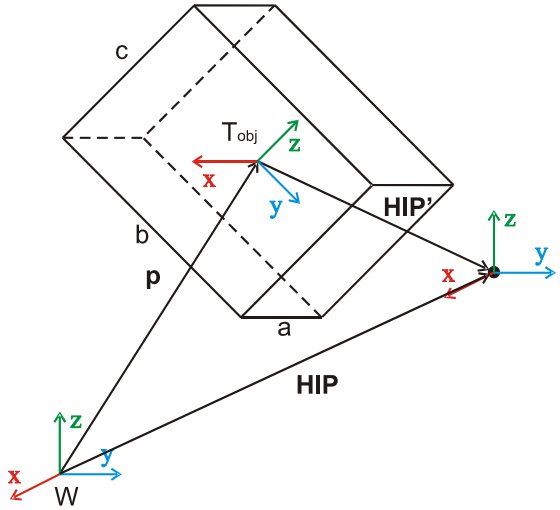


Figure 25 Object and HIP in a virtual environment

4.2.8.2 Computation of collision force

When **HIP** is outside of the object, the collision force is zero. The collision with the object is modeled as a spring-damper system with stiffness *k* and viscous friction *b*. Virtual force can then be computed as

$$\mathbf{F}'_{virt} = (k\ d + b(\mathbf{v}_O - \mathbf{v}_{HIP}) \cdot \mathbf{n})\mathbf{n}\,, \tag{9}$$

where $\mathbf{F}'_{virt}$ is the virtual collision force in the object coordinate frame, *d* is the collision depth **HIP**, $\mathbf{v}_O$ is the object velocity, $\mathbf{v}_{HIP}$ is the velocity of the robot end-effector and **n** is a normal vector to the collision surface. Collision model is shown in Figure 26.

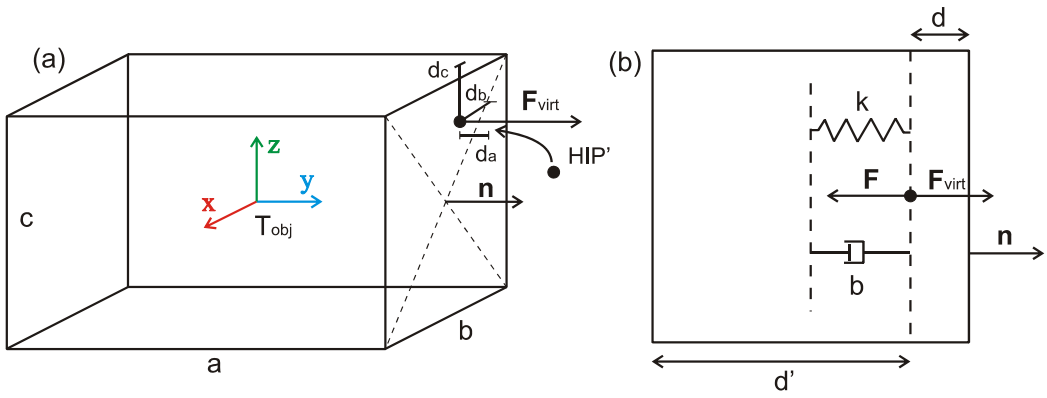


Figure 26 Collision detection and collision model

Special attention needs to be paid in the corners of the virtual object where the virtual force might quickly change direction as shown in Figure 27. Virtual force always needs to be parallel to the normal on the surface through which the object was penetrated and cannot change direction, while **HIP** remains within the object.

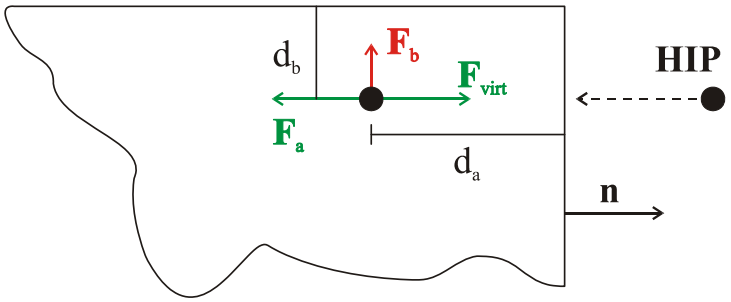


Figure 27 Collision detection ambiguities. Force must always be parallel to normal vector n

In the last step the virtual force, which was computed in the local coordinate frame of the object needs to be transformed to the WCS

$$\mathbf{F}_{virt} = \mathbf{R}_{obj}\mathbf{F}'_{virt} \tag{10}$$

$\mathbf{F}_{virt}$ is the force acting on the virtual object. The opposite force acts on the user through the HapticMaster robot. The sum of all forces acting on the object defines its acceleration.

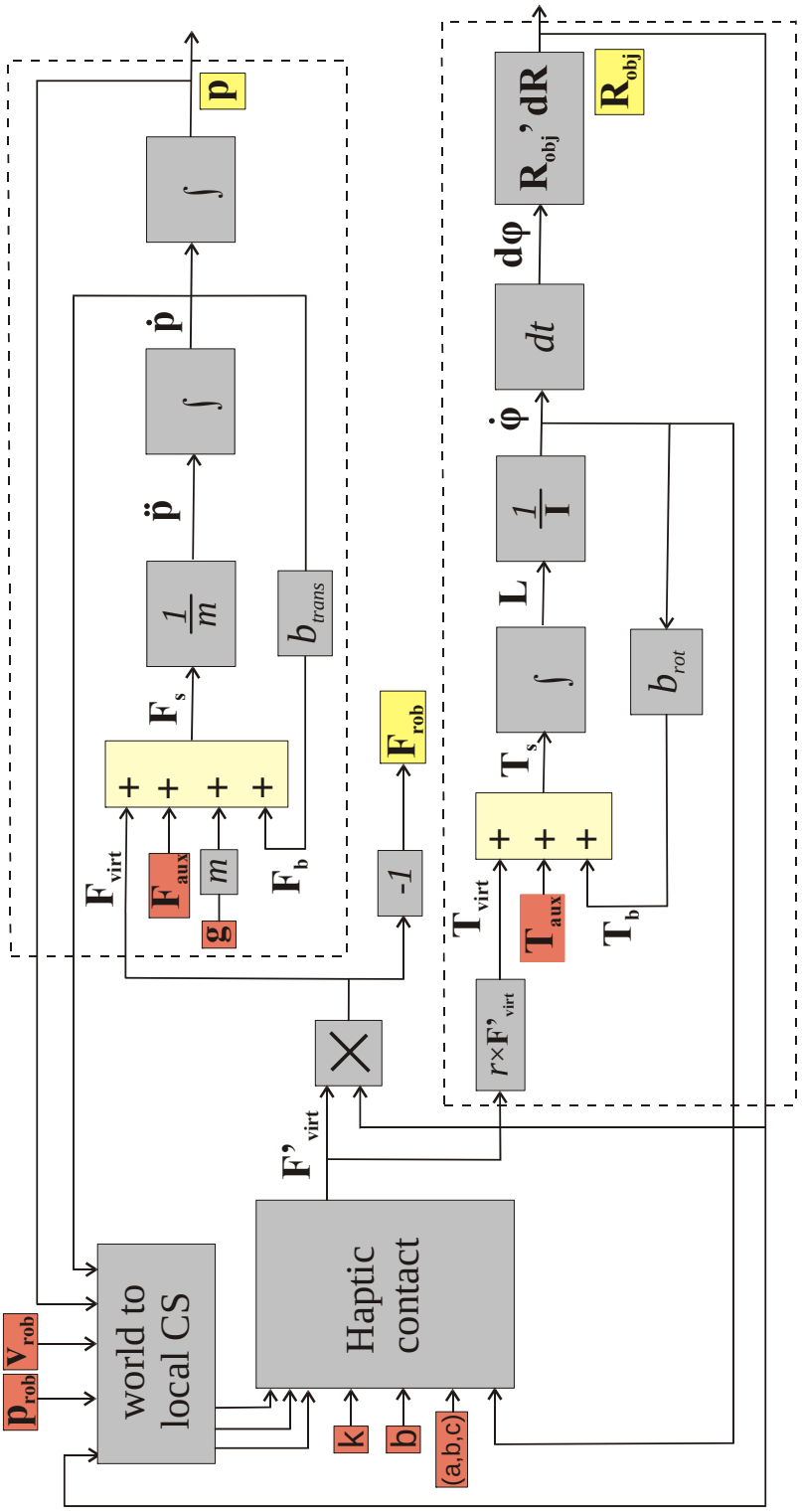


Figure 28 Block diagram for haptic object dynamic model

$$\begin{aligned} &\mathbf{F}_{aux} + \mathbf{F}_{virt} - \mathbf{F}_b + \mathbf{F}_g = m\ \ddot{\mathbf{p}} \\ &\ddot{\mathbf{p}} = m^{-1}(\mathbf{F}_{aux} + \mathbf{F}_{virt} - b_{trans}\ \dot{\mathbf{p}} + m\ \mathbf{g}) \end{aligned}, \qquad \textbf{(11)}$$

where $\mathbf{F}_{aux}$ is the force of the virtual environment on the particular object, $\mathbf{F}_b$ is the friction force of the environment on the object and b_{trans} defines translational friction.

Angular velocity of the object can be computed based on the sum of all torques acting on the object.

$$\begin{aligned} &\mathbf{L} = \int (\mathbf{T}_{aux} + \mathbf{T}_{virt} - \mathbf{T}_b)\ dt \\ &\mathbf{L} = \mathbf{I}\ \dot{\varphi} \\ &\dot{\varphi} = \mathbf{I}^{-1} \int (\mathbf{T}_{aux} + r \times \mathbf{F}_{virt} - b_{rot}\dot{\varphi})\ dt \\ &d\varphi = \dot{\varphi} dt \\ &d\mathbf{R} = rot(x, d\varphi_x) \cdot rot(y, d\varphi_y) \cdot rot(y, d\varphi_y) \end{aligned}, \qquad \textbf{(12)}$$

where **L** is the object angular momentum, $\mathbf{T}_{aux}$ is the torque of the virtual environment on the object, $\mathbf{T}_b$ is the viscous friction torque, b_{rot} is the rotational friction, $\dot{\varphi}$ is the angular velocity and r torque arm for virtual force $\mathbf{F}_{virt}$. The change in object orientation is defined by transformation matrix $d\mathbf{R}$.

HapticMaster Library includes blocks for the following haptic objects: sphere, cube, cylinder and wall. Mask for haptic object cube is shown in Figure 29. Masks for other objects are similar with the only difference in the parameters of object dimensions.

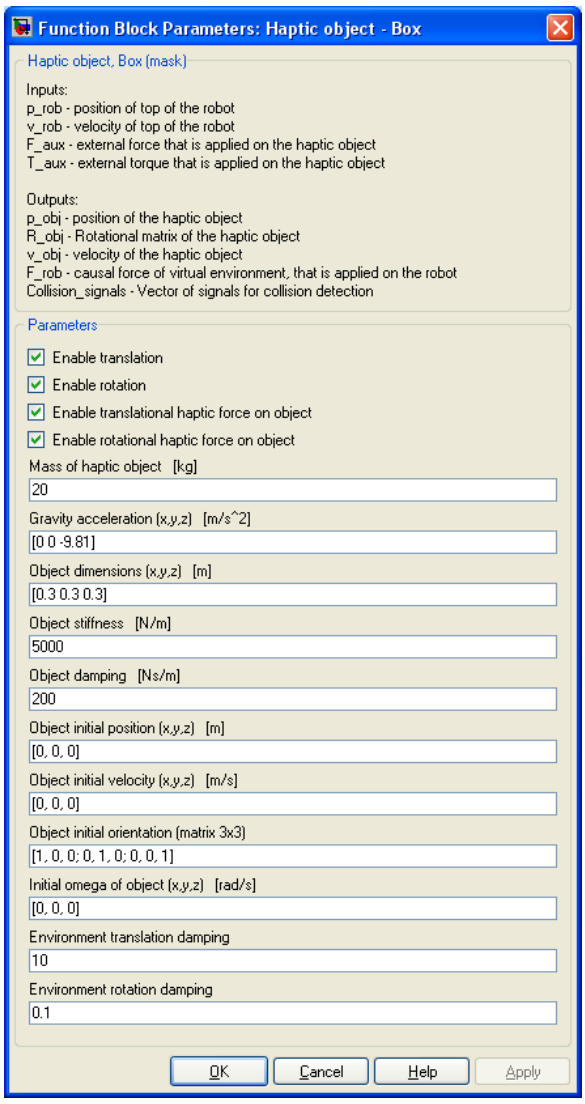


Figure 29 Haptic cube block mask

Haptic object block input signals:

- *p_rob* – end-effector position,
- *v_rob* – end-effector velocity,
- *F_aux* – virtual environment force on the object,
- *T_aux* – virtual environment torque on the object.

Haptic object block output signals:

- *p_obj* – haptic object position,
- *R_obj* – haptic object orientation in the matrix form,
- *v_obj* – haptic object translational velocity,
- *F_rob* – reaction force on the robot end-effector (on the user),
- *Collision_signals* – signals used for collision detection between the objects.

Haptic object block parameters:

- *Enable translation* – enable/disable object translational movement.
- *Enable rotation* – enable/disable object rotational movement.

- *Enable translational haptic force on object* – enable/disable **HIP** force on object.
- *Enable rotational haptic force on object* – enable/disable **HIP** torque on object.
- *Mass of haptic object* – hatpic object mass.
- *Gravity acceleration* – gravity acceleration (possible in all three directions).
- *Object dimensions* – object dimensions,
- *Object stiffness* – stiffness of the object walls,
- *Object damping* – viscous friction of the object walls,
- *Object initial position* – initial position of the object,
- *Object initial velocity* – initial velocity of the object,
- *Object initial orientation* – initial orientation of the object,
- *Initial omega of object* – initial angular velocity of the object,
- *Environment translation damping* – translational viscous friction of the virtual environment,
- *Environment rotation damping* – rotational viscous friction of the virtual environment.

4.2.8.3 Haptic object sphere

Figure 30 shows the model of a haptic sphere and the computation of the penetration depth.

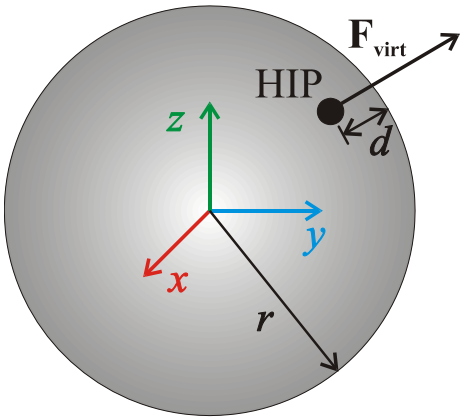


Figure 30 Haptic object sphere

Penetration depth can simply be computed as

$$d = \left|\mathbf{HIP'}\right| - r\,, \tag{13}$$

where **HIP'** is the position of the robot end-effector in the local coordinate frame of the sphere. Sphere Simulink block is shown in Figure 31.

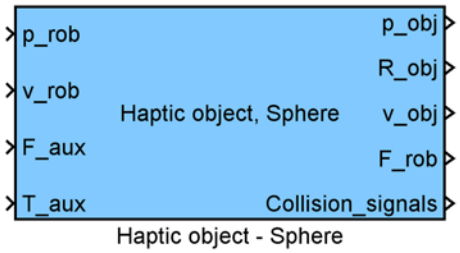


Figure 31 Haptic sphere block symbol

4.2.8.4 Haptic object cube

Cube dimensions are defined with vector (*a, b, c*). Penetration depth is computed as.

$$d = \min(\mathbf{HIP'} - \mathbf{dim}_{obj} / 2), \tag{14}$$

where **HIP'** is the position of the robot end-effector in the local coordinate frame of the cube and $\mathbf{dim}_{obj}$ is the vector of object dimensions.

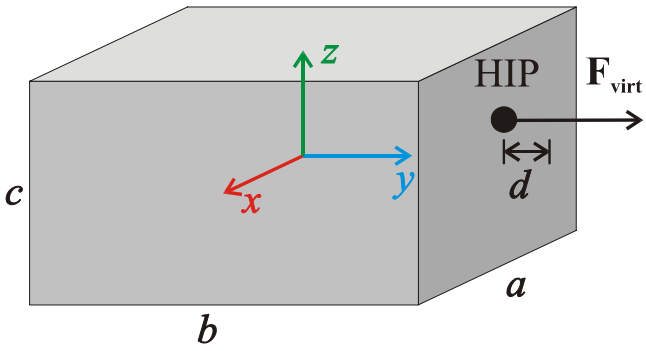


Figure 32 Haptic object cube

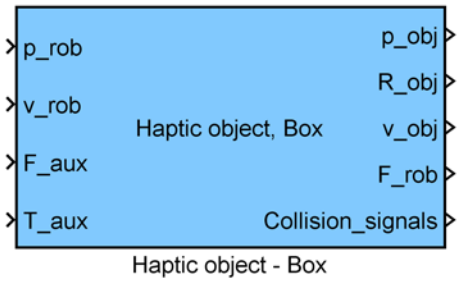


Figure 33 Haptic cube block symbol

4.2.8.5 Haptic object cylinder

Model of a haptic cylinder is shown in Figure 34.

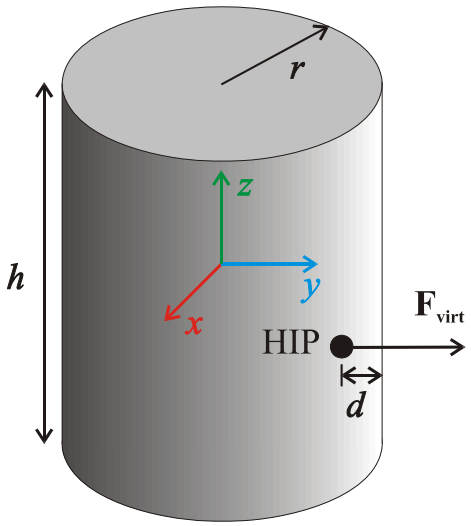


Figure 34 Haptic object cylinder

For the computation of penetration depth two options need to be considered. In the first case penetration occurs in the curved surface of the cylinder, while in the second case the penetration occurs in one of the two circles. Penetration depth can be computed as

$$d = \sqrt{x^2 + y^2} - r \quad , \qquad d = z - \frac{h}{2} \tag{15}$$

where *x, y* are the components of the **HIP'** vector.

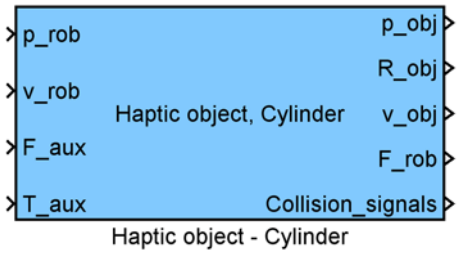


Figure 35 Haptic cylinder block symbol

4.2.8.6 Haptic object wall

Haptic wall limits the space of the virtual environment and consists of six walls, which can be positioned or disabled independently. Wall is firmly attached to the virtual environment and cannot move. Wall models are shown in Figure 36. Penetration depth can be computed as

$$d = \mathbf{HIP'}(z)\,, \tag{16}$$

where **HIP'**(*z*) is the *z* component of vector **HIP'**.

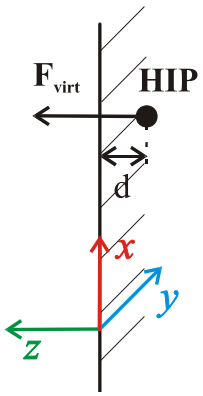


Figure 36 Haptic object wall (six such objects are contained in a haptic wall block)

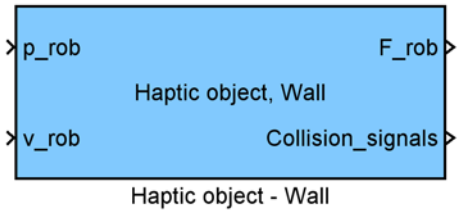


Figure 37 Haptic wall block symbol

Mask for the haptic wall block is different from other haptic object block masks and is shown in Figure 38.

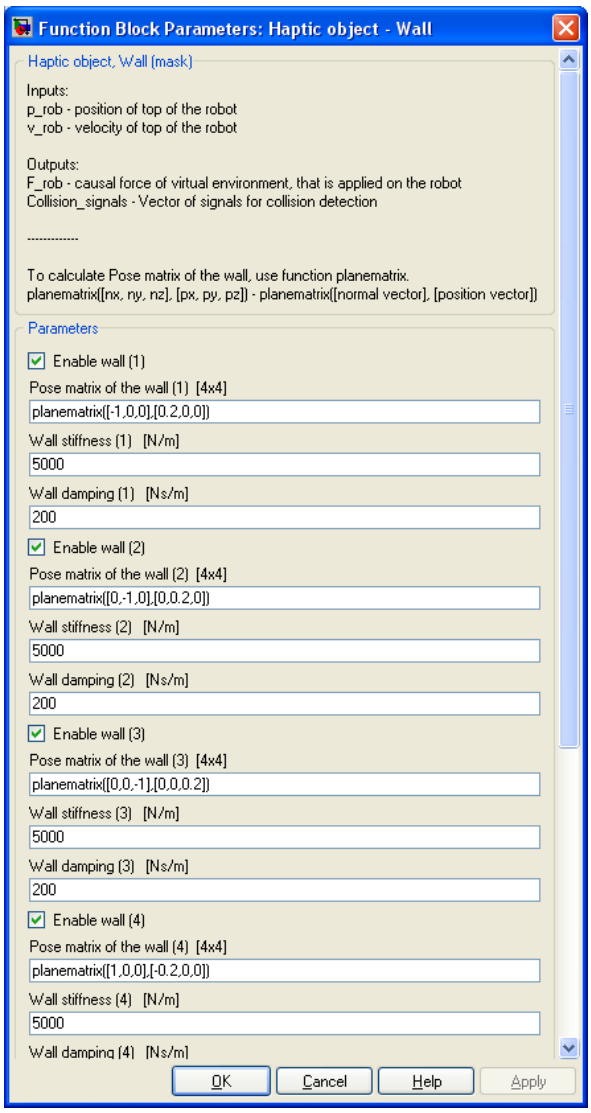


Figure 38 Haptic wall block mask

Mask parameters for the haptic wall block are the following:

- *Enable wall* – enable/disable wall
- *Pose matrix of the wall* – matrix that defines position and orientation of the wall.
- *Wall stiffness* – stiffness of the wall.
- *Wall damping* – damping of the wall.

4.2.9 Collision detection

Collision detection between **HIP** and haptic object is already included in the haptic object model itself. Here we will consider collisions between haptic objects. In a case study a collision between two spheres will be analyzed. Collisions between other haptic objects are similar, only collision detection is more complex due to more complex object shapes. Collisions are always considered between two objects. If a virtual environment consists of more than two objects, each possible collision pair needs to be considered.

4.2.9.1 Collision of two spheres and a sphere-wall collision

To simplify collision detection, one object is always chosen as the reference object and the position of the second object is transformed in the coordinate frame of the reference object. Now we can consider the relative position of the second object to the first object.

4.2.9.2 Force computation

Collision force is computed based on the penetration of one object into the other object. In Figure 39 a collision between two spheres and a collision between the sphere and a wall are shown.

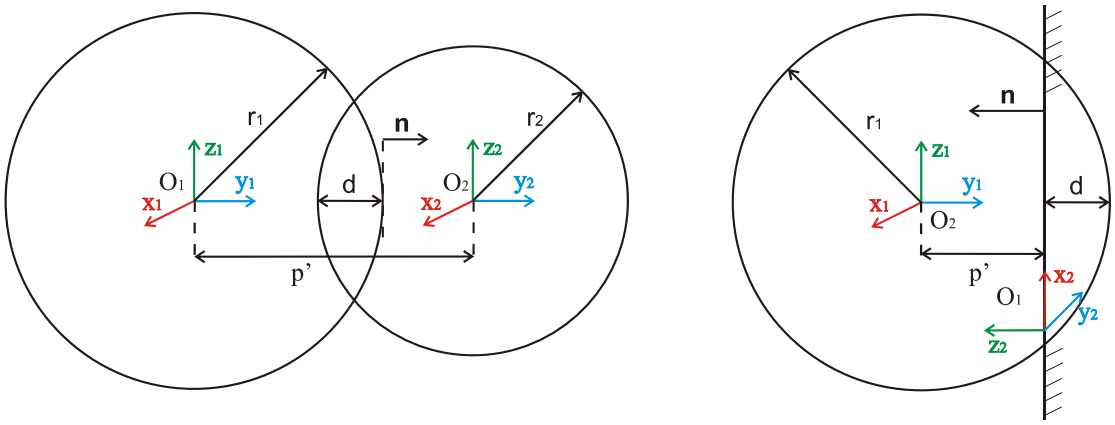


Figure 39 (a) Collision of two spheres and (b) collision between the sphere and a wall

Collision force can be computed as

$$\begin{aligned} \mathbf{F}_1' &= -(k_s\ d - b_s \mathbf{v}_d \cdot \mathbf{n})\mathbf{n} \\ d &= r_1 + r_2 - p' \\ \mathbf{v}_d &= \mathbf{v}_1 - \mathbf{v}_2 \\ k_s &= (1/k_1 + 1/k_2)^{-1} \\ b_s &= (b_1 + b_2) \end{aligned} \quad , \qquad \textbf{(17)}$$

where $\mathbf{F}_1'$ is the force acting on the first object expressed in the local coordinate frame of the object, p' is the distance between the sphere centers, d is the penetration depth, k_s is the stiffness of the interaction, b_s is the viscous friction of the interaction, $\mathbf{v}_d$ is the relative velocity between two spheres, $\mathbf{n}$ is the normal vector to the contact surface. Force on the second object is opposite to the force on the first object. Both forces need to be transformed into the WCS. In the case of two colliding spheres it is not necessary to compute collision torque. However, for other objects this needs to be considered and can simply be calculated as the cross product between the collision force and the torque arm defined as the vector between the contact point and the object local coordinate frame.

Computation of the collision between the sphere and a wall is similar to the computation above. Sphere position is transformed into the coordinate frame of the wall and then collision force is computed.

4.2.9.3 Collision detection blocks

Collision detection is always considered between two haptic objects. Therefore each collision detection block has two inputs that include all parameters necessary for collision detection. Outputs of collision detection blocks are forces and torques on each haptic object involved in the collision.

Here is a list of bocks for collision detection between various haptic objects:

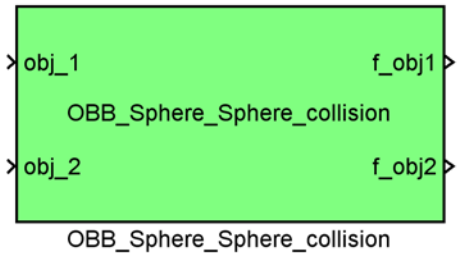


Figure 40 Sphere-sphere collision detection block symbol

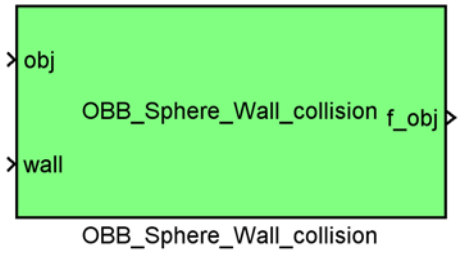


Figure 41 Sphere-wall collision detection block symbol

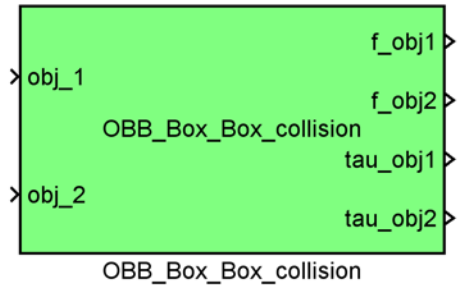


Figure 42 Box-box collision detection block symbol

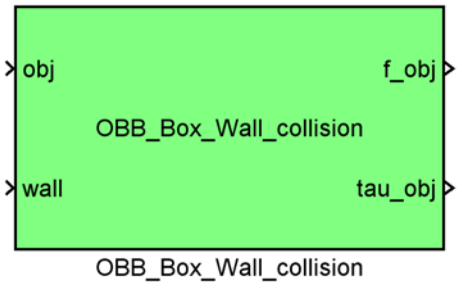


Figure 43 Box-wall collision detection block symbol

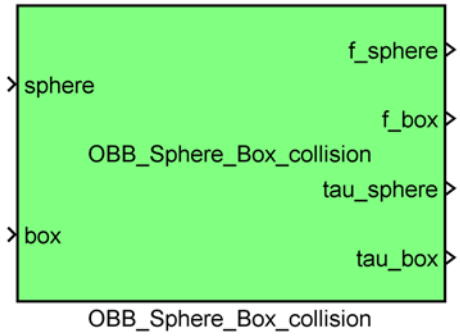


Figure 44 Sphere-box collision detection block symbol

4.2.10 Haptic object tunnel

The haptic object tunnel enables movement along the designed trajectory from start point to end point. The forces that affect the user are calculated during the movement through the tunnel. These forces tend towards movement along a bisector of the tunnel. The bisector is generated by B-splines. The parametric space trajectory is presented by B-splines, which are used for local approximation and interpolation of the trajectory with lower degree polynomials. The splines are used as basis functions, which are smooth at contact points between segments. For the purpose of describing the arm trajectory from point to point over individual coordinate system dimensions we can use different numbers of tunnel parameters, depending on the complexity of the tunnel. The parameters are the coefficients of the approximation with basis functions and the trajectory is the sum of the basis functions. An example of five basis functions is shown in Figure 45.

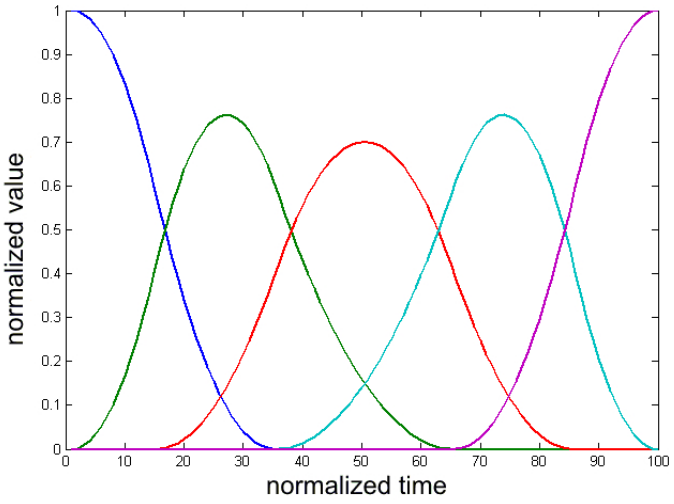


Figure 45 An example of five basis functions for use with B-splines

The radius of the tunnel varies along its trajectory. If the radius of the tunnel is equal to zero on the whole trajectory, the user can move only along the bisector. The top of the robot is presented as a ball. The position of the ball is defined by the position of the top of the robot. The size of the ball is measured by using a grasping module. The user can move the ball along the tunnel, as well as vertically when the radius of the ball is smaller than the radius of the tunnel. When the ball penetrates into the wall, the reaction force - which forces the ball out of the wall - is computed. Figure 46 shows the cross-section of the tunnel when the ball penetrates into the wall. If the radius of the ball is larger than the radius of the wall, the movement along the tunnel is difficult. Figure 47 shows the longitudinal section of the tunnel when damping is applied along the tunnel. The collision between the object and the tunnel wall is modeled as a spring-damper system with a stiffness *k* and a viscous friction *b*.

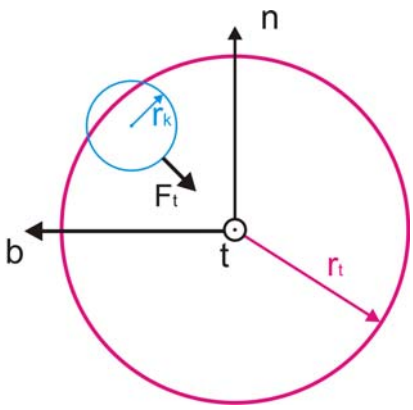


Figure 46 Cross-section of the tunnel when the ball penetrates into the tunnel wall

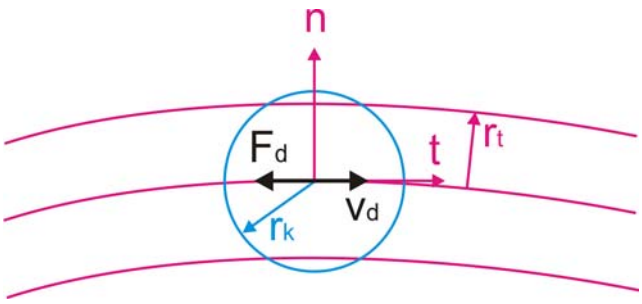


Figure 47 Longitudinal section of the tunnel when damping is applied

All forces are computed in a coordinate system of the tunnel, which is defined by a direction vector of tangent $\mathbf{t}$, a direction vector of normal $\mathbf{n}$ and a direction vector of binormal $\mathbf{b}$. Scalar r_t marks the radius of the tunnel and the radius of the ball is marked with r_k. Vector $\mathbf{v}_d$ shows the speed of the ball in direction of the tunnel. Vector $\mathbf{F}_t$ is a force of the tunnel wall and vector $\mathbf{F}_d$ is a force of the damping along tunnel.

Figure 48 shows an example of movement through the tunnel. The blue line represents the path of the top of the robot, the red line represents the path of the bisector of the tunnel and the green lines are the walls of the tunnel. The whole tunnel lies in the frontal plane xz.

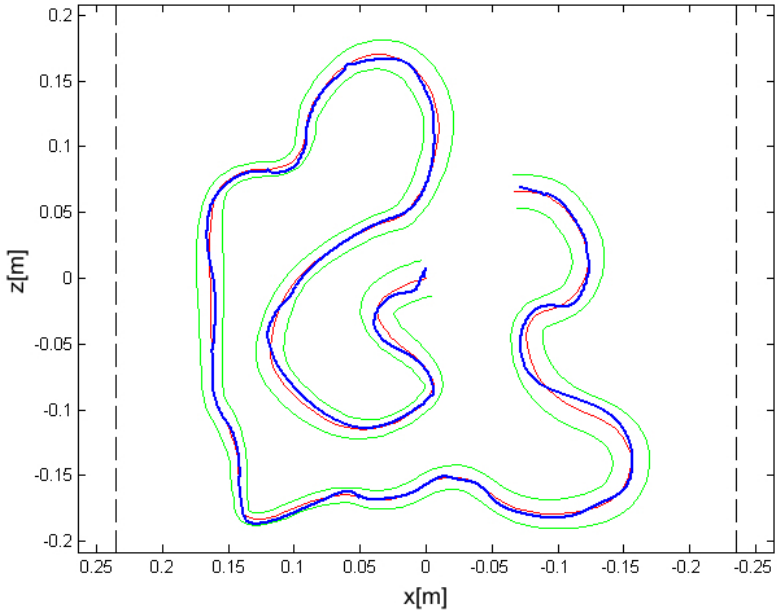


Figure 48 An example of movement through the tunnel. The tunnel lies in the frontal plane xz

4.2.11 HapticMaster library

HapticMaster library is part of the Matlab/Simulink library and includes all previously described blocks. It is divided into five subsystems that include similar blocks.

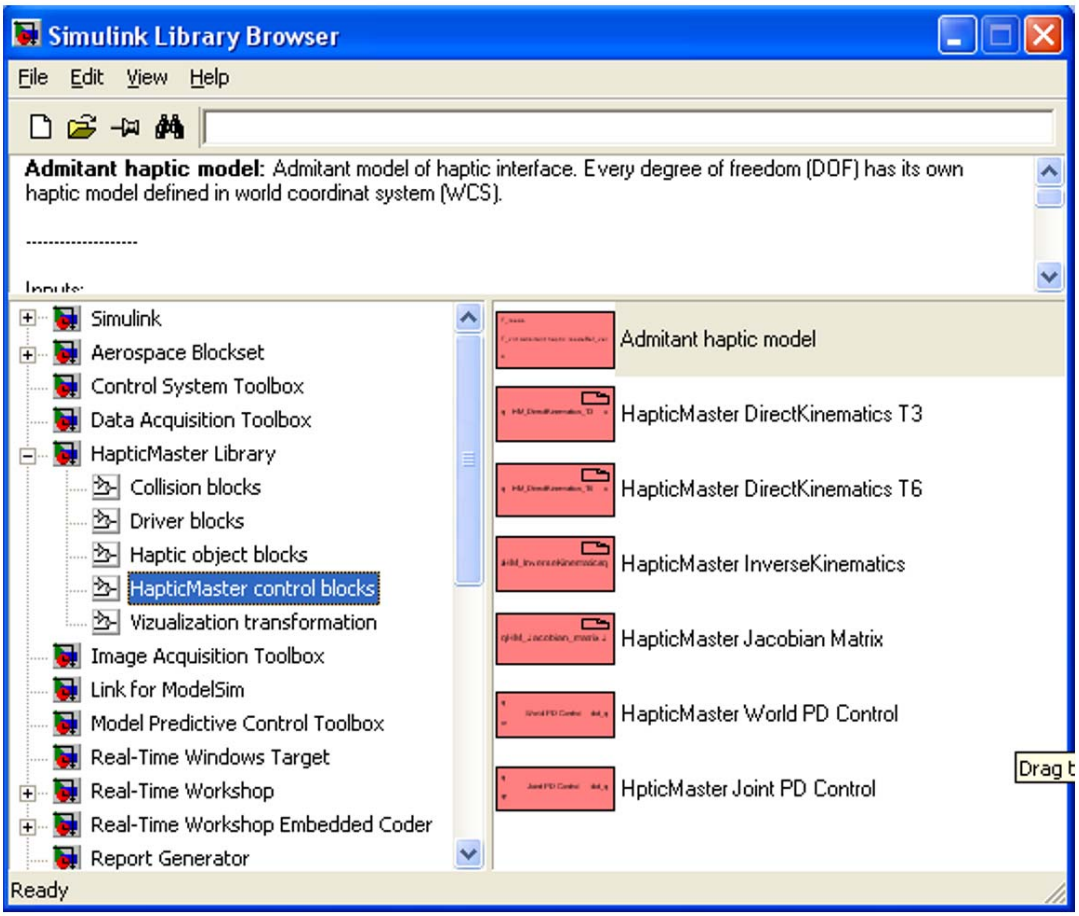


Figure 49 HapticMaster Library in Simulink browser window

4.2.12 Example task

Figure 50 shows a Matlab/Simulink model for haptic rendering that includes admittance control, three free floating haptic objects and six walls. Building scenarios only requires selection of blocks from HapticMaster library and making proper connections between the blocks. No additional programming is required.

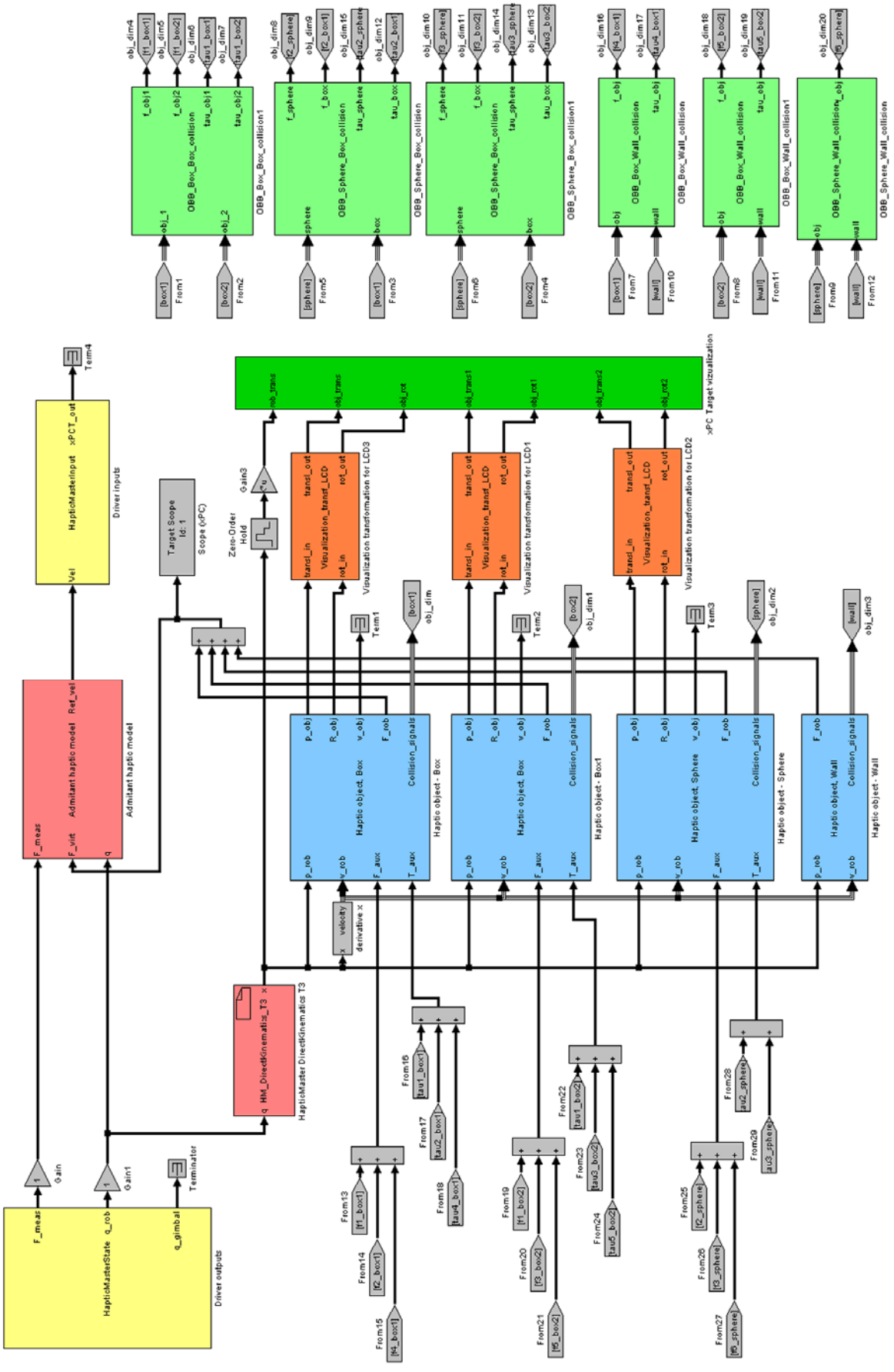


Figure 50 Example application

4.3 Robotic control and haptic rendering for the Lokomat

4.3.1 Introduction

For haptic rendering typical sampling times are in the range of 500-1000 Hz. Below 500 Hz, the rendering can become unstable, thus producing vibrations. To ensure the required update rate, the haptic rendering is therefore directly included within the Lokomat control architecture.
An example of haptic rendering is the display of a haptic wall. The user can touch the virtual wall with his foot and experience friction, stiction as well as impact forces. Together with proper acoustic and visual feedback, this can increase the realism of the simulation.
Display of haptic interaction forces can be done with two general computational frameworks. Impedance and admittance control. While impedance control allows the display of soft objects, admittance control performs better with rigid objects.

We developed pure impedance and a combined impedance/admittance control framework. The pure impedance performs better in zero impedance mode, whereas the combined framework is capable of taking the best out of both world with minor drawbacks in the performance of zero impedance control

4.3.2 Computation of haptic forces in impedance control

During impedance control, the position of the foot tip can be computed from the angles of hip and knee and an estimation of the foot length.

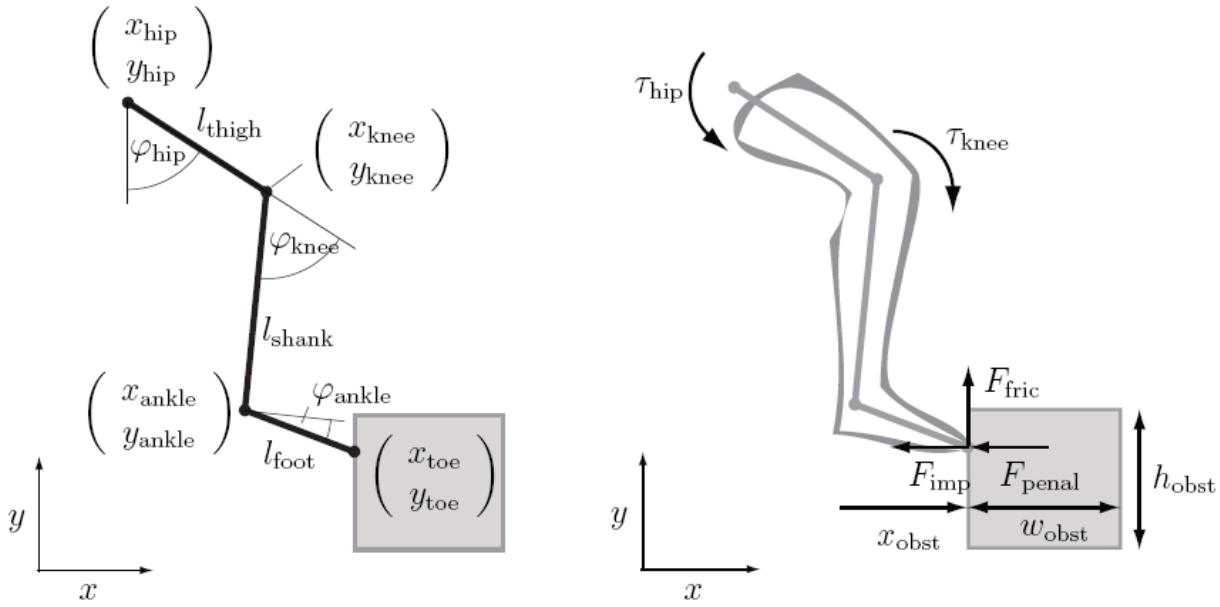


Figure 51: Left: computation of the foot position form the Lokomat joint angles. Right: inverse kinematics for computation of resulting interaction torques from interaction forces.

When a collision occurred, the resulting forces are three fold. An impulse force F_{imp} is applied on contact and pushes the leg out of the virtual object. The desired haptic interaction forces is computed as $F_x = k\Delta x$ on contact with the object. A penalty force F_{penal} is computed depending upon the penetration depth and velocity as a weighted

spring damper system $F = k\Delta x + c\Delta x * b\Delta x'$. Additional friction terms can help creating a realistic feeling. The friction force F_{fric} can be computed as $F_{Fric} = \begin{cases} \mu F_{penal} \frac{v_y}{|v_y|} & if\ |v_y| \geq v_{threshold} \\ \mu F_{penal} \frac{v_y}{v_{threshold}} & o.w. \end{cases}$.

4.3.3 Combined admittance and impedance control

For admittance control, two models are needed to display haptic objects. The first has to model the leg during a free movement in the Lokomat space. The second has to model the interaction behavior between the leg and the virtual object.

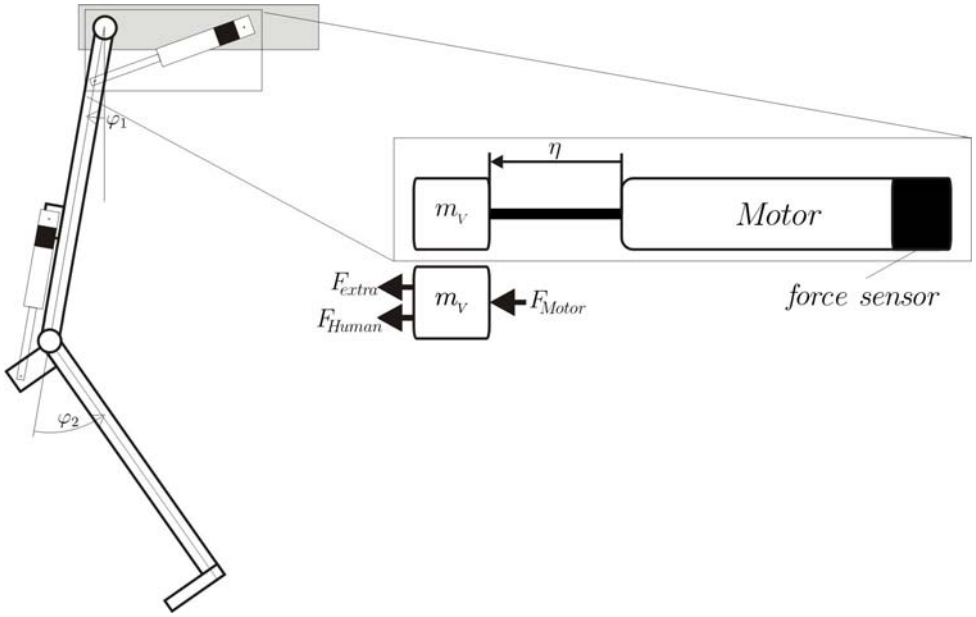


Figure 52 Schematics of the Lokomat leg. Two motors drive the hip and knee joint respectively. The force sensor that measures interaction forces is located in line with the motor. Our model assumes a foot length based on approximations done by Winter et al 1990.

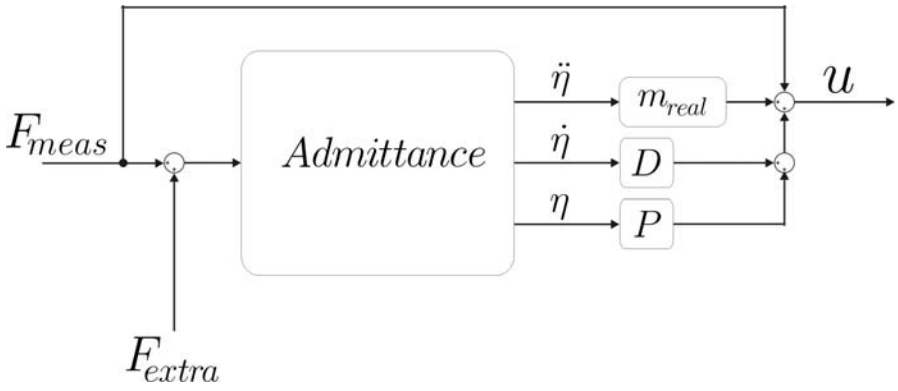


Figure 53 The control of the leg and the computation of the interaction forces take place in the Admittance block. The recorded forces are summed with the forces that occur during object collision. These forces are then translated into a desired position of the leg. (e.g., on object collision, the desired position of the leg is the surface of the object) and translated into a motor current u.

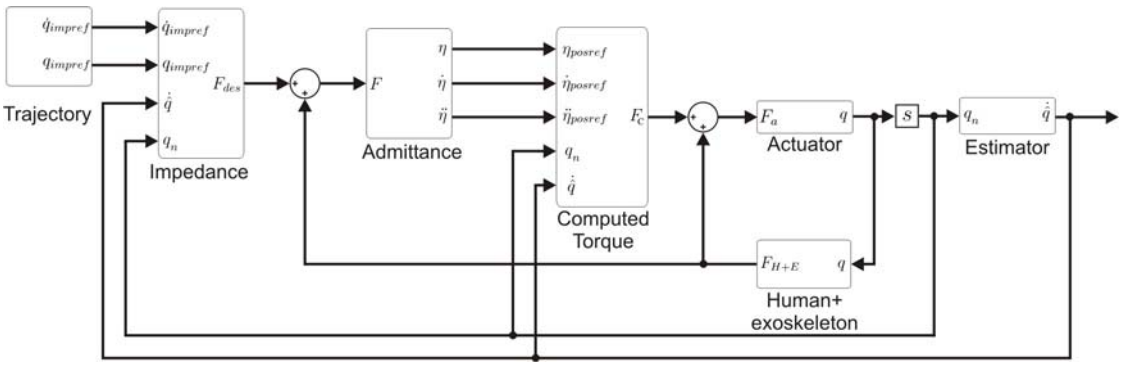


Figure 54 Complete model: the reference trajectory for free movements is generated in the trajectory block. The combined impedance/admittance blocks allow the display of rigid as well as soft objects. The computed torque controls the desired position that results from the admittance controller

The leg behavior is defined at motor level, which means that the leg is modelled as a virtual mass m_V at a distance η from the motor. The forces acting on the virtual mass are the motor force F_{Motor}, the human force F_{Human} and an arbitrary chosen virtual additional force F_{add}.
F_{add} represents the sum of different additional virtual effects, such as impedance and friction on the virtual object.

$$F_{add\ i} = F_{imp\ i} + F_{fr\ i} \quad (1)$$
$$F_{meas\ i} = F_{Human\ i} + F_{Motor\ i} \quad (2)$$

The index $i = 1$ refers to the hip and $i = 2$ the knee joint.
Using the sensors attached to the Lokomat, we are not able to measure the human interaction force F_{Human} directly, but only the combination of F_{Human} with the motor force (Figure 52).
The system equation for each joint is:

$$m_{Vi}\ddot{\eta}_i = F_{meas\ i} + F_{add\ i} \quad (3)$$

The measured forces are directly fed to the differential equation (3). The resulting displacements of the differential equation (3) are directly fed to the position controller. The position controller generates a motor torque which drives the measured η_{real}, $\dot{\eta}_{real}$ and $\ddot{\eta}_{real}$ to the desired η_{posref},$\dot{\eta}_{posref}$ and $\ddot{\eta}_{posref}$.

The impedance controller can be added at the input of the admittance controller which generates a F_{imp} force which tries to minimize the difference between the actual angle position and the reference trajectory.

4.3.4 Collision detection in the combined impedance-admittance control

During a collision with a virtual object, the knee motor shaft length η_2 is calculated as geometrical function of the hip motor shaft length η_1 and the virtual object shape.

$$\eta_2 = f(\eta_1, object) \quad (4)$$

The admittance dynamics at the knee, equation (3) index 2, is omitted and replaced by the geometrical equation .

Since the knee admittance dynamics are omitted during collision, the measured knee force has no influence on the entire system. To reintroduce the knee dynamics omitted earlier, the measured force is calculated to a torque acting on the knee joint and shifted to the hip joint. This is then calculated back to a force axial to the motor shaft and added to the hip admittance dynamics as an external force.
So the resulting dynamical equation for the hip joint is a function

$$\ddot{\eta}_1 = g(F_{meas\ 1}, F_{meas\ 2}, F_{external\ 1}) \tag{5}$$

This method allows displaying a rigid object. The only penetration into the object is caused from the deviation of the position controller. The maximal position controller is, thus, also the maximal object stiffness.

A friction effect can be introduced adding a rule to F_{fr1}. The friction force can be computed as a viscous friction

$$F_{fr1} = -\mu\dot{\eta}_1 \tag{6}$$

or a Coulomb friction

$$F_{fr1} = -\mu F_N\dot{\eta}_1 \tag{7}$$

4.3.5 Measurement results

A wall is situated at $0\ m$ and the free moving space is on the negative side of the wall. x_{real} is the foot tip position calculated from the measured knee and hip angles. x_{des} is the position which is generated from the admittance model – this is the position which the position controller tries to follow. F_x is the normal force of the virtual wall acting on the foot tip.

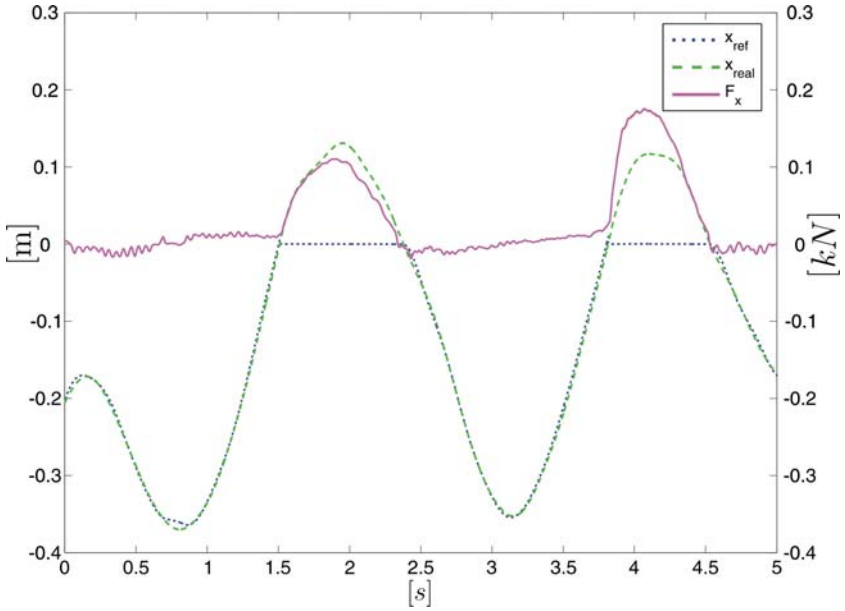


Figure 55 A measurement of a collision of the foot tip and a vertical wall at 0 m

When a collision occurs, the desired foot position x_{des} is kept constant at the border of the wall. The real foot tip position x_{real} penetrates the object and the error to the desired position increases. The position controller generates a force which pushes the foot back to the wall border.

4.3.6 Applicability of haptics in the Lokomat

While haptics surely play a major role in the haptic master, it can only be used sparsely during Lokomat training. Any force that might destabilize the walking pattern of the patient must be avoided, which means, that hard, rigid objects can only be displayed if the patient is standing, not during walking. If displayed during walking, they have the potential to make the patient stumble. While we can display soft objects like a soccer ball during walking, the patient has to stop walking and stand still when interactions with objects like an obstacle, with a curb stone or with a building occur. We have therefore implemented a Lokomat training sequence, which can occur during the training. The following describes the sequence of states that will take place if the user encounters an obstacle in the virtual world. It is of course equally adaptable for interactions with other objects.

1. User walks with guidance force in Lokomat
2. Obstacle is enabled
3. Lokomat stops the patient at obstacle position and keeps the motors of hip and knee engaged in order to ensure the patients stability.
4. Collision detection is enabled
5. Guidance force in one leg is reduced to zero – the other leg is kept fully extended to ensure patient stability
6. The patient can now explore the object with the foot that has zero guidance force.
7. If the user has overstepped the obstacle, the guidance force is increased to previous value.
8. Lokomat switches to start position
9. Normal walking is continues

5 Visual rendering

5.1 Introduction to the goals of visual rendering

Visual rendering means the computer generation of 3D images of real world and even virtual world. The explanations of developed and already available tools used for the visual rendering module are described in the following paragraphs. None of these tools are in conflict with each other and can be integrated in a common application.
Description of tools used in the scenarios.

5.2 Tools for visual rendering

5.2.1 OGRE - Object-Oriented Graphics Rendering Engine

OGRE is an object oriented graphics rendering engine. It is a flexible 3D object oriented C++ class library intended for simple hardware related 3D applications. Even though the tool is merely a rendering engine, the open source and object oriented scheme enables simple plug-in inclusion and therefore considerable modularity. Features that distinguish OGRE match many features of the commercially available 3D rendering engine:

- support for both OpenGL and Direct3D
- support for Windows, Linux, and Mac OS X platforms
- automatic handling of render state management and hierarchical culling
- powerful and sophisticated material management and scripting system, allowing maintenance of materials and fallback techniques
- support for render-to-texture techniques and projective texturing
- access to vertex and index buffers, vertex declarations, and buffer mappings
- support for shadowing techniques, including stencil, texture, additive, and modulative
- support for easy-to-use skyboxes, skyplanes, and skydomes and many others.

OGRE Design

OGRE provides an object-oriented method of access to procedural data processing: rendering simple geometric primitives to a render target (usually a screen buffer).With an object-oriented approach to rendering geometry there is no need to deal with geometry. Instead one should only deal with the scene in terms of the objects that make up the scene: manipulation with movable objects in the scene, static objects that make up the world geometry, lights, and cameras. Once the objects are in the scene, detailed aspects of scene manipulation as well as visual rendering is handled by OGRE itself and whereas the user only has to deal with objects, their properties, and intuitive methods of manipulation instead of trying to manage them in terms of vertex lists and triangle lists and rotation matrices.

When handling scene objects one will appreciate that the scene graph is completely decoupled from the scene contents as opposed to typically coupled scene content and the scene graph in an inheritance hierarchy that forces the subclassing of content classes as types of scene nodes, which presents a considerable and inflexible problem in later stages of graph algorithms alterations. OGRE operates on its scene graph at an interface level; that is it operates on the scene graph only through its signature and is completely ignorant of the underlying graph algorithm implementation. Also, OGRE's scene graph interface is concerned only with the graph structure. Only the *renderable* level contains the content access or management functionality. The rendering properties (*materials*) for these renderables are contained in Entity objects, which in turn contain one or more SubEntity objects. These subentities are the actual renderable objects (Figure 56).

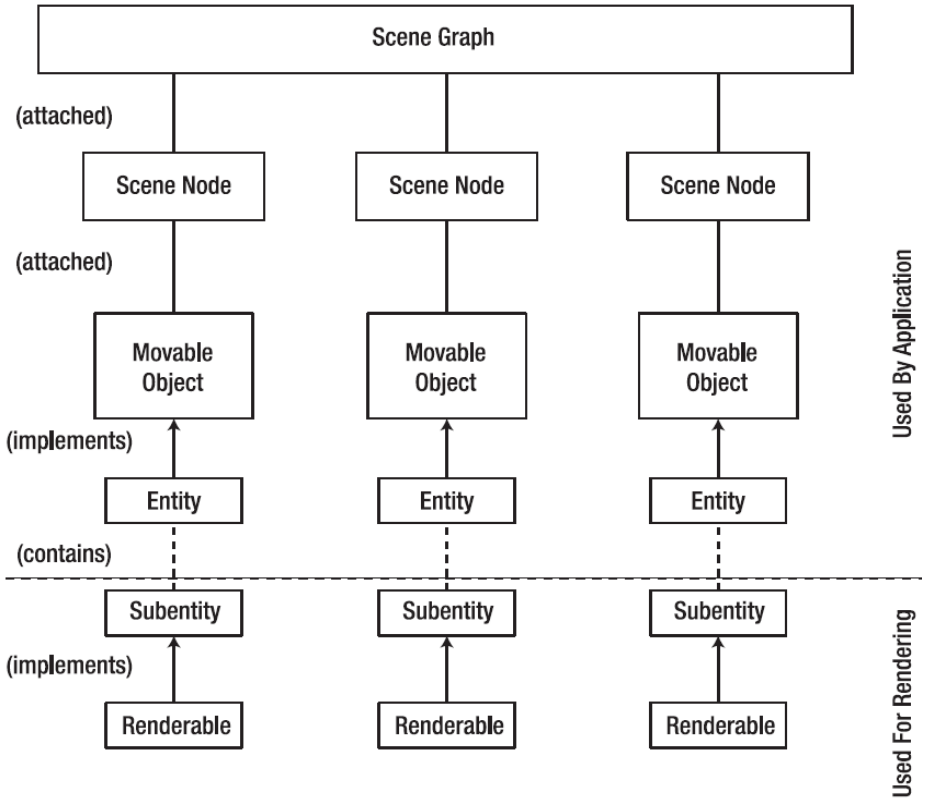


Figure 56 Relationship between the scene graph structure and content management objects in OGRE

The main point of access to an OGRE application is through the Root object and from this point on all of the object handling is performed via corresponding managers. Scene design can be a time consuming task, especially if the scene is complex with many details. For this reason entire scene was designed in a 3ds Max Studio and exported with oFusion exporter. Certain upgrades in OGRE application are necessary to load the scene in its entirety to form a virtual environment, whereas object manipulation is performed in a standard way.

5.2.2 XVR

XVR is a complete system for the development of VR-oriented applications. It consists out of a state-of-the-arte graphics engine, a scripting language, an integrated development studio and an ActiveX module which can be embedded to various container applications such as web browsers. It offers support for a wide range of VR devices (trackers, motion capture devices and stereo display systems), multimedia including 3D audio and network communications. XVR can be extended using its own scripting language as well as external dynamic link libraries (DLL). While the scripting language offers high-level functionality, OpenGL functions can be used within the script for low-level control. The user of XVR can develop applications on a desktop computer, and with minimum effort transfer them into a full VR system. For more information about XVR refer to [1].

5.2.3 Hardware Accelerated Library for Character Animation (HALCA)

A library for character animation has been developed to provide an interface to the Cal3D library. It also extends it with GLSL shader support and other additions to the original Cal3D animations system such as morph animations. The library offers support for loading complete character models including textures, skeleton systems and animations. Utilities for motion capture, keyframe animation enable the user to blend and loop different animation sequences. A simple inverse kinematics allows the virtual characters to perform shoulder and elbow rotations. This feature can be used to achieve realistic looking arm movements based on information read from a tracker device. Further to that, GLSL shaders can be used for very efficient skin deformations.

The library has been developed as dynamic link library and it can be embedded to both XVR and OGRE frameworks. Documentation, tutorials and further information are available on the internet [1] and an example application using the character library can be found in [2].

5.3 Visual rendering for the HapticMaster

UPPER EXTREMITY SCENARIO

Since this section contains confidential intellectual property, it has been removed from the public version of the deliverable.

5.4 Visual rendering for the Lokomat

All scenarios modeled in section 3.3 were rendered using the open source graphics engine OGRE, which allows the easy placement and manipulation of 3D objects, lights and cameras at runtime. Until now, no additional software had to be purchased to achieve further effects e.g. global lightning or shadows. Shadows are directly supported by OGRE and will be added in a subsequent iteration. Houses used in the scenario were exported for OGRE from Google Warehouse using SketchUp Pro, a simple CAD program that allows the fast creation of buildings and other 3D content. On a Dell D830 Latitude Laptop with an Intel graphics card, an average frame rate of 20 fps was achieved with the maximum triangle count of 363173.

5.5 General purpose visual rendering

The general purpose scenarios have been developed using XVR and the character library described in sections 5.2.2 and 5.2.3 respectively. The scenarios will be run in a Head Mounted Display system. Elements of the environment were exported from Google Warehouse and the characters used were hand-rigged characters from the

[1] http://www.lsi.upc.edu/~bspanlang/animation/avatarslib/doc/

AXYZ-Design company[2] that are represented by about 5K-10K polygons. The same scenarios can be used with other types of display, such as large screen stereo displays.

6 Auditory rendering

6.1 Introduction to the goals of auditory rendering

The auditory rendering is an important factor to optimize the patient's sense of presence during the rehabilitation training. There are empirical results suggesting that the perceived quality of the visual display can be improved when presented in conjunction with sound. Since hearing is a passive process, we can easily listen while being occupied with other tasks.

Different sound samples will be used to simulated sound as generated from walking steps, manipulation of objects or other objects.

6.2 Auditory rendering for the HapticMaster

Auditory rendering is the task of playing a sound properly timed with a graphics or haptic event. An example is the collision of the virtual object (cup) with a virtual obstacle (table). When the object impacts the obstacle, the graphical and haptic displays show the contact with a minimal time shift to acoustics.

Fmod open source library (www.fmod.org) is used for acoustic rendering. The 5.1 surround sound is featured in this framework – the only programming steps are setting the position of the sound source in x, y and z coordinates, setting the volume and playing the sound, which is normally stored as a wave file.

(Since this section contains confidential intellectual property, part of it has been removed from the public version of the deliverable.)

6.3 Auditory rendering for the Lokomat

Auditory rendering for the Lokomat is playing a sound properly timed with a graphics or haptic event. In order to create a realistic impression, the sound cannot be delayed more than 20 ms after the occurrence of the graphical and haptic event. An example is the collision of the foot with a virtual obstacle. When the foot impacts the virtual wall, the graphical and haptic displays show the foot contact with the wall with a minimal time shift to acoustics.

We used the Fmod open source library (www.fmod.org) for acoustic rendering. The 5.1 and 7.1 surround sound is featured in this framework – the only programming steps are setting the position of the sound source in x, y and z coordinates, setting the volume and playing the sound, which is normally stored as a wave file.

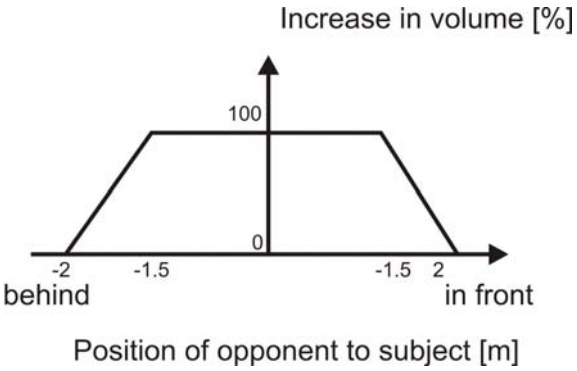
[2] http://www.axyz-design.com/

For the first test, we implemented three virtual tasks – a city walk, in which the subject explores a virtual city, a soccer scenario in which the subject plays soccer in a stadium against a virtual opponent and a canyon walk, where the subject has to cross a narrow bridge over a canyon (refer to chapter 3.3).

Table 1: Scenarios for auditory rendering

City walk	Soccer scenario	Canyon walk
Stepping sounds	Stepping sounds of avatar	Stepping sounds
	Stepping sounds of opponent	Wind sounds
	Ball kicking sounds	
	Cheering crowd	
	Goal score sound	

Additional effort was put into making the footstep sounds of a virtual opponent within the soccer scenario more realistic. Rendering the step sounds of the opponent that follows the subject in the VR, we placed the sound source behind the subject, and increased the volume of the approaching character's stepping sound with decreasing distance to the patient. If the character came close to the patient, the patient therefore obtained an acoustic clue on the distance between them.



6.4 General purpose auditory rendering

The Head Related Transfer Function (HRTF) describes how any given sound source is filtered through the anatomical characteristics of one's body, such as the geometry and characteristics of the body, torso and ears, before the sound reaches the inner ears' mechanism. Through binaural audio recording, we can create more immersive environments as the sound reproduced includes much more spatial information than simple stereo recordings, increasing the listener's information on sound localization. Currently a binaural recording setup has been developed, which comprises of a dummy head created by face casting a real head. The materials used for the casting of the dummy head are a thin layer of silicon on the outside that is used to simulate the skin, and a rigid / hardened foam interior to represent the internal bone structure. Two microphones have been fitted internally in the dummy head, ending at the ear

canal just in front of where the actual ear drum would be. The microphones are then connected to a 2-channel recorder and each channel records the corresponding ears' sound input. The combination of the two channels produces the binaural recording.

Binaural recordings can be used to add realism to the experiment and enhance the immersion of the user to the system.

7 References

[1] Carrozzino, M., Tecchia, F., Bacinelli, S., Cappelletti, C., and Bergamasco, M. 2005. Lowering the development time of multimodal interactive application: the real-life experience of the XVR project. In *Proceedings of the 2005 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology* (Valencia, Spain, June 15 - 17, 2005). ACE '05, vol. 265. ACM, New York, NY, 270-273.

[2] Mortensen, J., Yu I., Khanna P., Tecchia F., Spanlang B., Marino G., Slater M. 2008. Real-Time Global Illumination for Virtual Reality Applications, *to appear in IEEE Computer Graphics and Applications A Special Issue on VR.*

[3] Lenggenhager, B., Tadi, T., Metzinger, T. and Blanke, O., 2007. Video ergo sum: Manipulating bodily self-consciousness. Science. 317, 1096-1099.