Andrea Tarallo

Human-Mechanical systems Interaction in Virtual Reality

Research Doctorate Thesis

Advisors:

Prof. Francesco Caputo
Prof. Giuseppe Di Gironimo

Scuola di dottorato in Ingegneria dei sistemi meccanici
Dipartimento di Ingegneria Meccanica per l'Energetica
Università degli Studi di Napoli Federico II

Coordinator: Prof. Raffaele Tuccillo

November 2009

to my family.

Acknowledgments

I must confess that I am quite embarrassed while writing these few lines, because I am aware that the "Acknowledgments" usually are the most noticed pages of an entire work. However, I can reassure the reader that I will not try at giving life lessons here, but I will simply thank all the people that have made this work possible.

First of all, I would like to thank my advisor and friend Professor Giuseppe Di Gironimo for the trust he has shown in supporting me during the last three years. I have actually had the opportunity of better knowing his moral integrity, loyalty and helpfulness.

Many thanks also to my advisor Professor Francesco Caputo for his aptitude at pleasantly combining his scientific rigor with his broad humanistic culture. Special thanks go to Professors Antonio Lanzotti, Stanislao Patalano and Salvatore Gerbino and also to my current and former colleagues: Fabrizio Renno, Mariano Guida, Adelaide Marzano, Mariangela Trotta, Pasquale Franciosa.

Thanks to the colleagues of Fondazione GraphiTech: Raffaele De Amicis, Giuseppe Conti, Gabrio Girardi for introducing me in the field of advanced computer graphics. I have to thank also vrcom GmbH support team, in particular Erik Coates and Raphael Zeiger for their helpfulness in answering my questions.

I also would like to thank all the students that have supported my work during the last three years, in particular (roughly in chronological order): Roberto del Vecchio, Amedeo Sgura, Gianfranco Fiore, Giuliano Minieri, Margherita Brasiello, Omar Chioccariello.

Last but not least, my wholehearted thanks to Agostino De Santis and Luigi Pelliccia, for their essential research contributions and my colleagues and friends Stefano Papa, Giovanna Matrone and Bruno Liguori for their friendship and for their endless support. Thank you all.

Andrea Tarallo, Naples, November 2009

Contents

List of Figures

Overview

The present work aims to show the great potential of Virtual Reality (VR) technologies in the field of Human-Robot Interaction (HRI). Indeed, it is foreseeable that in not too distant future cooperating robots will be increasingly present in human environments. Many authors actually believe that after the current information revolution, we will witness the so-called *robotics revolution*, with the spread of increasingly intelligent and autonomous robots capable of moving into our own environments.

Since these machines must be able to interact with human beings in a safe way, new design tools for the study of Human-Robot Interaction (HRI) are needed. The author believes that VR is an ideal design tool for the study of the interaction between humans and automatic machines, since it allows the designers to interact in real-time with virtual robotic systems and to evaluate different control algorithms, without the need of physical prototypes. This also shields the user from any risk related to the physical experimentation.

However, VR technologies have also a more immediate application in the field of HRI, such as the study of usability of interfaces for real-time controlled robots. In fact, these robots, such as robots for microsurgery or even *teleoperated* robots working in a hostile environments, are already quite common. VR allows the designers to evaluate the usability of such interfaces by relating their physical input with a virtual output.

In particular, the author has developed a new software application aimed at simulating automatic robots and, more generally, mechanical systems in a virtual environment. The user can interact with one or more virtual manipulators and also control them in real-time by means of several input devices. Finally, an innovative approach to the modeling and control of a humanoid robot with high degree of redundancy is discussed. VR implementation of a virtual humanoid is useful for the study of both humanoid robots and human beings.

The present work is divided into five chapters, that will be briefly described below.

Chapter 1 is an introduction to the Human-Robot Interaction (HRI) concepts and the possible uses of VR technologies in this field of study.

Chapter 2 concerns the description of the hardware technologies and the software applications that underly VR experience. In particular, the hw/sw tools used for this work are briefly described. Moreover, some issues related to the actual application of VR technologies to the product development are discussed.

Chapter 3 describes *RoboTiX*, a VR robotic simulation tool developed by the author.

Chapter 4 focuses on the use of VR for usability evaluation of Human-Robot interfaces.

Chapter 5 describes an innovative approach to the modeling and control of humanoid robots, as well as virtual humans.

Chapter 1

Introduction

In recent years, research in robotics is looking for different applications where a human being is to be conceived not exclusively as an operator programming off-line the robot, but rather as a system interacting with the machine by means of different modes, [CC00].

According to many researchers, in not too distant future, robotic devices assisting and serving humans in domestic as well as professional environment will become as common as Personal Computers nowadays are.

Cooperating robots could be very interesting not only in household environments, but especially in the industrial field. In fact, at the current stage, the flexibility of production processes is still strongly dependent upon manual production steps. This is mainly caused by the fact that manual production processes involve limited investments, are characterized by faster decision processes and require simpler and faster training if compared to those required by personnel involved in highly automatic processes.

Thus, one of the most compelling challenges of the next few years will not be a further spread of automatic processes into the industry, but instead the improving of the existing manual workstations with intelligent tools, designed to assist and support the human operators during their working, such as *power extenders* [Kaz98] (Figure 1.1).

Industrial robots today only operate within secure areas, where there is no possibility for human operators to interfere with them. In fact, sharing the workplace with an automatic robot is still very dangerous for a human being, because of the lack of efficient collision avoidance mechanisms and proper security measures.

Indeed, according to the *European Machine Directive 98/37/EC*, such flexible manufacturing systems, and in particular man-machine interaction

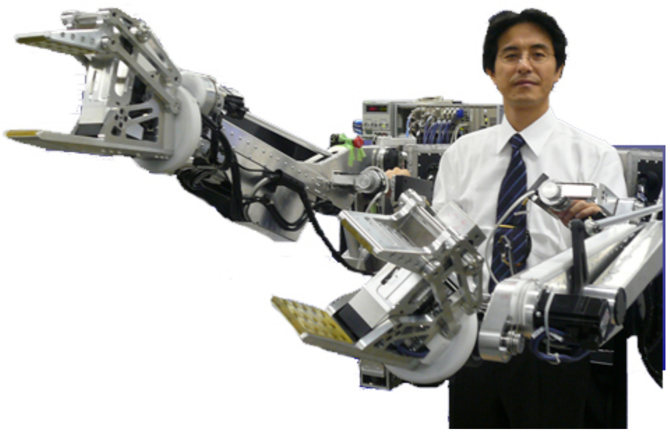


Figure 1.1: Dual Arm Power Amplification Robot (courtesy of Activelink Co.,ltd.)

systems, would require the development of algorithms specifically targeted to safety and control, capable of making the robot respond with intelligent reactions to unforeseen events. Moreover, it is fundamental to develop algorithms capable of minimizing the need to interrupt the production, ensuring, at the same time, the safety of the personnel interacting with the robot.

Another interesting research line that involves the study of HRI is the so-called *assistive robotics*. Assistive robotics refers to the study of the technologies and methodologies to develop robotic machines that operate services in environments cohabited with the human, such as wheelchair-mounted manipulators for physically disabled people [DDM*08]. In this field, it is also important to design proper control interfaces that allow the humans to easily control the robots in unstructured environments. Moreover, these robotic systems that operate in anthropic environments must be characterized by a significant degree of autonomy, reliability and security in order to make them capable to react in the most appropriate way when a failure occurs, as well as when a collision or, more generally, an unforeseen event takes place [AAB*06].

Finally, the study of Human-Robot interaction can be applied more generally to the usability evaluation of control interface for real-time controlled robots, such as robots for microsurgery [DZJ*99] and other *teleoperated robots* (Figure 1.2).

Teleoperation is also essential to perform tasks in adverse environments, such as space robotics, nuclear plant servicing, undersea operations and more

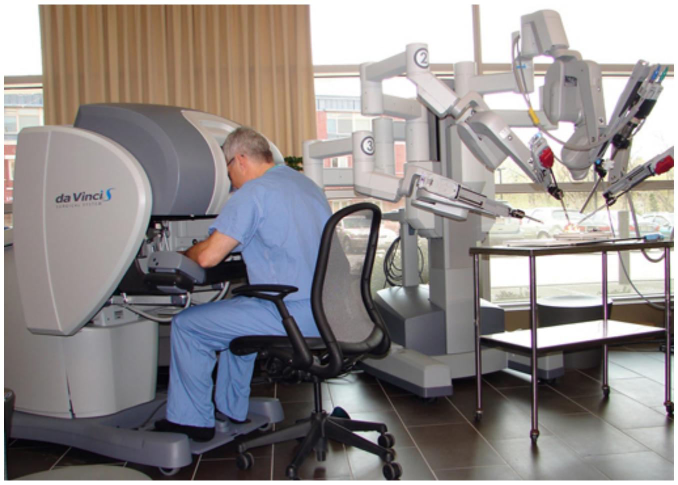


Figure 1.2: da Vinci surgical system (Boston Globe/Milton Hospital)

generally when the human operator cannot be in the operational scenario for any reason. In such cases, the operator controls the robot manipulator in real-time by means of a physical interface.

It is understood that the study of all aforementioned issues, which involves both the robot control algorithms and their control interfaces, requires new design tools that can take into account not only the functional requirements of the product itself, but also the problems arising from its interaction with the human beings.

1.1 Motivations and objectives

Virtual Reality (VR) technologies have helped the spread of the so-called *Partecipatory Design* in several industries [MBMP06], [Dav04], [MMGS09] and have already shown their great potential for designing not only industrial products, but even manufacturing systems, [CDM06a], [CDM06b], [DDM06] and work-cells [Cra97].

This work aims to extend the application of these technologies also in the field of physical Human-Robot Interaction (pHRI) and the so-called *Anthropic Robotics*. In fact, virtual and augmented reality technologies appear to be an effective way to evaluate different behavior strategies for robot and other automatic machines [Bur99]. Inside a Virtual Environment (VE), the designer can interact directly with a virtual robotic machine, and verify, for instance, the effectiveness of the implemented control algorithms or simulate human-robot shared tasks, while operators position is tracked in real-time.

VR technologies can be also used to evaluate the usability of robot control interfaces. In fact, by means of VR technologies, it is possible to relate the physical inputs coming from different control interfaces to a virtual output inside a digital immersive environment. Specifically, we can control in real-time a virtual robotic arm by means of a real control interface, such as an ordinary joystick. Thanks to stereoscopic visualization, we can also take into account both cognitive and physical aspects related to the man-machine interaction [KHK*07], [KBS*01]. This is particularly important for assistive robotics, because it is not so obvious that a human being positively accepts the physical presence of a robotic system in his own environment. The "immersion feeling" provided by stereoscopic technology improves the reliability of the opinions expressed by VR-testers by means of ordinary questionnaires.

Moreover, several unforeseen events, such as the occurring of a mechanical/electronics failure or the unexpected movement of a human operator within the robot's working space, can be simulated as well in real-time.

It is worth emphasizing that all these evaluations can be performed in a virtual environment, without the need of physical prototypes.

In short, the objective of this research line is to use VR technologies in the development and validation of safety measures necessary to provide a safe interaction between human being and mechanical systems, considering at same time the usability of their eventual control interfaces.

In particular, I have developed many modules mainly oriented to *Virtual Design Review* applications and Robotics Simulation. They will be extensively described in the following chapters.

The use of Virtual Reality as a tool aimed to the measure of experimental data has provided significant benefits in terms of performance and repeatability of the tests, ensuring controlled experimental conditions. The interaction with a virtual product also shields the user from any risk eventually related to the interaction with real prototypes. Finally, the use of VR technologies for the collection of the experimental data is fundamental in terms of safety, costs and repeatability of the tests.

Chapter 2

Virtual Reality framework

2.1 Introduction

The goal of every immersive Virtual Reality (VR) simulation is to give the user the feeling of operating in a three-dimensional environment in which he can directly interact with virtual objects, for instance by grasping and using them to perform a simulated task.

For this purpose, a stereoscopic visualization system is essential, but not enough. Indeed, a complete VR system is a large software system [Zac00], consisting of many modules (Figure 2.1).

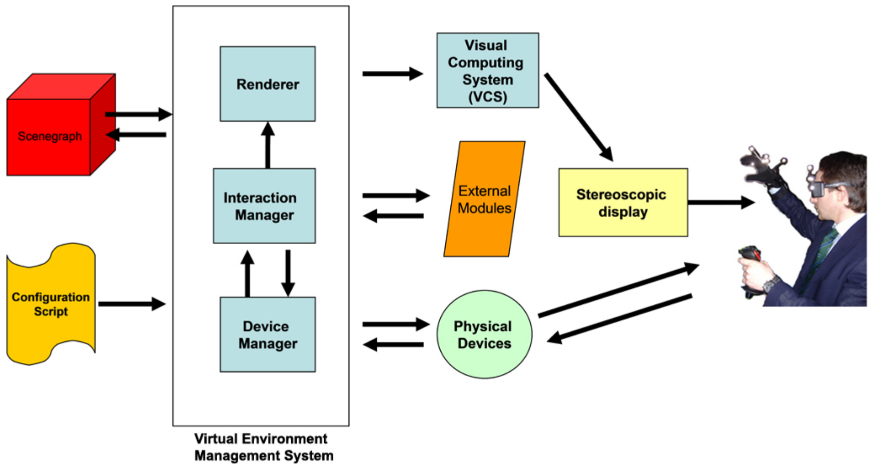


Figure 2.1: A typical VR framework.

Every VR system contains an object manager, renderer, device drivers, communication module, navigation and interaction module, etc. The visual part of a virtual environment is represented by a hierarchical *scene graph*. Everything is a node in this graph: geometries, light sources, viewpoint(s), etc. Moreover, the Virtual Environment Management System (VEMS) handles the interactions with the user. Generally, VEMSs are driven by *configuration scripts* that essentially tell which action to perform when a certain event is detected (*event-based approach*).

Another important part of our scheme is the device manager. VR application use a number of Multi-Dimensional Interactive (MDI) devices, thus a device handler with high degree of abstraction is fundamental. It must provide abstract interface to all the different input devices. This layer is very important for VEMS.

In many implementations the interaction manager integrates a *collision detection module*, responsible for detecting collisions among objects in the scene graph. The detecting of a virtual collision is essential in many situations, for instance when the user has to grab objects.

VEMSs often provide the possibility to extend their functionalities with plug-ins modules, which can be loaded at run-time by the VR system. In this way, the programmer can provide some application-specific functionalities.

Almost all modules should be able to run concurrently to each other. This is particularly true of real-time critical modules such as the renderer and collision detection module.

Finally, the Visual Computing System (VCS) and a proper stereoscopic display complete VR framework. In the following section the equipment of *VRTest* laboratory will be described.

2.2 VRTest Laboratory

The activities described in this thesis have been mainly carried out at *VRTest lab*, that is a VR laboratory set-up under the sponsorship of Campania regional authority [CD07b]. The laboratory is equipped with specific hw/sw components, which will be quickly mentioned in the following (Figure 2.2).

Computing system The computational unit is a high-end workstation based on a dual AMD Opteron™ CPU with 16GB of RAM. Both the concurrency characteristics of the processor and the amount of available system memory are indeed fundamental for real-time rendering.

PowerWall® display The PowerWall® display set-up at VRTest provides

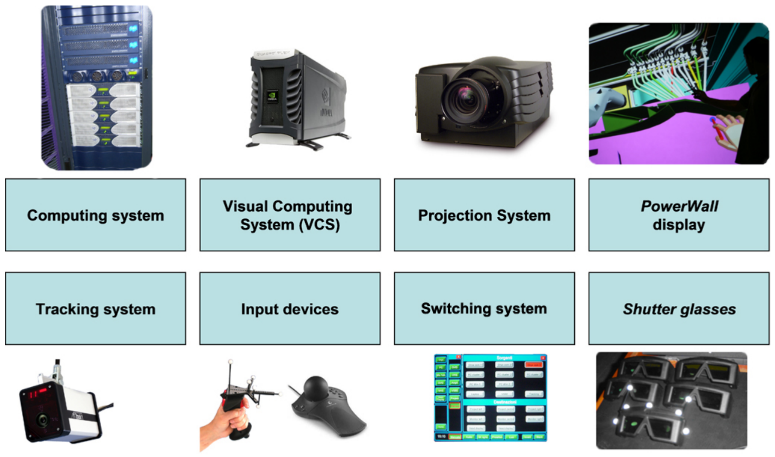


Figure 2.2: VRTest lab equipment.

a large display area (7.5 by 2.4 meters) in order to facilitate collaborations of groups of researchers and engineers. Thanks to rear-projection technology, all the collaborators can see the display clearly and without obstruction (Figure 2.3).

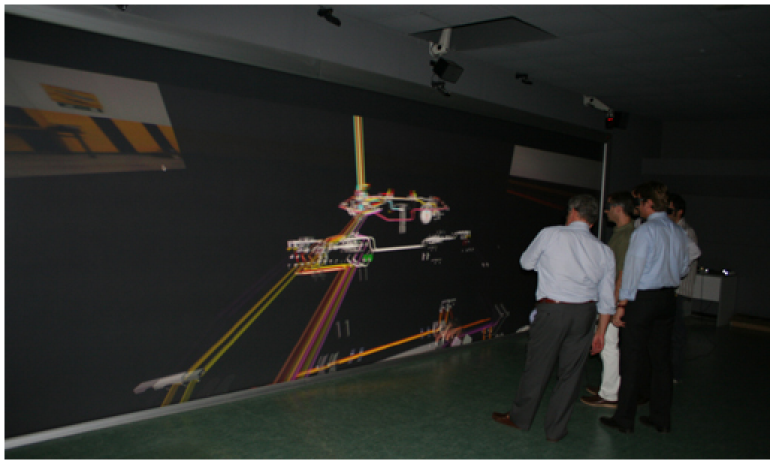


Figure 2.3: VRTest PowerWall collaborative display.

Projection system The stereoscopic display system is composed by three high-end DLP projectors, namely Barco^{TM} *Galaxy 6000+*. Their brightness (6000 ANSI Lumen) and work frequency (100Hz) always grant a clear and flickering-free visualization. Moreover, the projectors are synchronized with several Stereo3D$^{\text{TM}}$CrystalEyes® shutter glasses for active stereoscopic view.

Visual Computing System (VCS) The master workstation is connected to *nVidia*^{TM} *QuadroPlex 1000* VCS, a scalable solution for high-end computer graphics.

Tracking system The main purpose of a tracking system is to trace the position of the user and other physical objects in the virtual environment. In particular, VRTest lab is equipped with an optical tracking system, by *Advanced Realtime Tracking GmbH*. Three infrared cameras detect the position and orientation of a number of markers properly attached to the object(s) to be tracked (Figure 2.4).

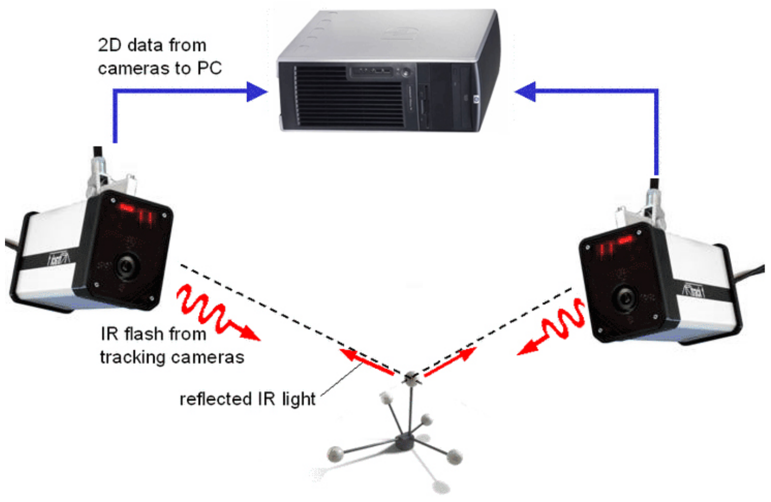


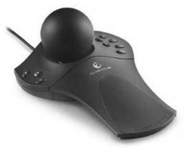
Figure 2.4: Tracking system architecture.

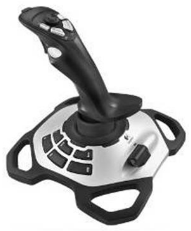
Input devices *VRTest* lab is equipped with a number of input devices that allow the user to easily navigate and interact within the virtual scene. In particular, we have two *Spacemouse*, two *Joystick*, a *Flystick* and a *Cyberglove*® (Figure 2.5).

3D audio output system The laboratory is endowed with a 3D audio output system to increase the realism of the Virtual Environment (VE).

Switching System and room control In order to simplify the control and the management of the VR laboratory there are interconnection

(a) Cyberglove


(b) Spacemouse


(c) Joystick

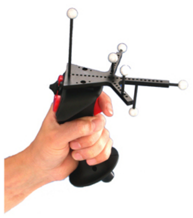

(d) Flystick

Figure 2.5: Input devices at VRTest.

devices such as video switching matrices and serial control connection equipment, provided with touch screen control panels. Any source, coming from the workstations, the tracking system, video I/O, etc. can be transmitted through a switching matrix on the projectors and on the control monitors positioned on the command console (Figure 2.6).

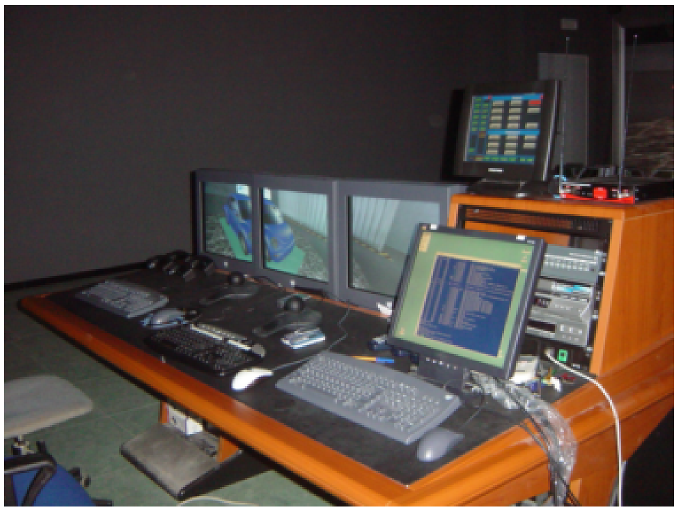


Figure 2.6: Command console at VRTest.

2.3 Architecture of Simulation Manager

As aforementioned, a VEMS properly handles all the VR-equipments in the lab. Specifically, the majority of this work has been developed starting from a commercial VEMS, namely *Virtual Design 2* (VD2) by *vrcom GmbH* [DMP06]. VD2 is an extensive tool containing many functions for product development, from the creation of Virtual Environment to assembly simulation, [AB98], or ergonomic analysis. Moreover, the Software Development Kit (SDK) allows the programmer to enhance the basic functionalities of the system by developing external modules that interface with VD2 kernel.

Since the programming of the virtual environment must go together with the knowledge of the simulation software internals, the concepts underlying VD2 internal data representation will be mentioned in the following sections.

2.3.1 Kernel modules

The kernel of Virtual Design 2 consists of three main components, [VA06]: the interaction manager, the device manager and the rendering kernel, as shown in Figure 2.7.

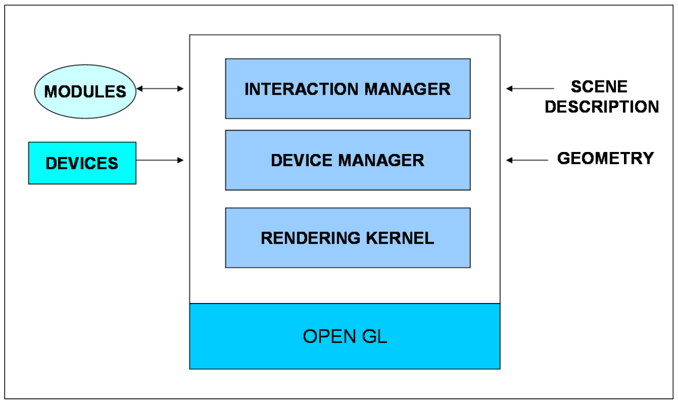


Figure 2.7: VD2 kernel architecture.

The **interaction manager** controls all actions in the Virtual Environment as well as the user's interactions within the virtual scene. It can be configured with a single *scene description file*, this is a script file that describes the static and dynamic configurations of the virtual objects. The description is based on *events*, *actions* and *objects*. The basic idea is that certain events will trigger certain actions, properties, or behaviors. For example, the virtual environment can be programmed in such a way that when the user touches a virtual button, a light will be switched on.

The **device manager** initializes and controls the hardware devices used in the Virtual Environment. It provides the mapping of physical devices to logical devices with high degree of abstraction (Figure 2.3.1) . Moreover, this mapping can be configured in a specific server-file. This concept simplifies scene building and enhances portability, limiting the concerns about which tracker-system is used or which machine it is attached to. The device manager supports the most common VR devices, like *Spacemouse*, tracking systems (*Polhemus*, *Ascension*, *ART*, *Vicon*, *Intersense*), digital data gloves (*Cyberglove*, *5DT*), and haptic systems (*Phantom*).

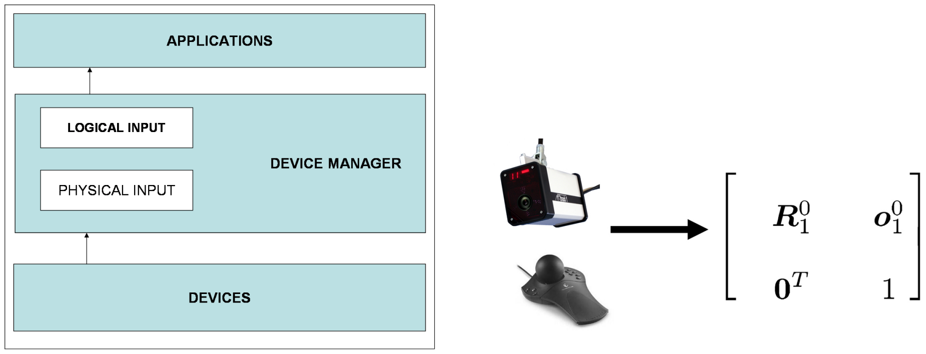

APPLICATIONS
LOGICAL INPUT
PHYSICAL INPUT
DEVICE MANAGER
DEVICES
$\begin{bmatrix} \boldsymbol{R}_1^0 & \boldsymbol{o}_1^0 \\ \boldsymbol{0}^T & 1 \end{bmatrix}$

Figure 2.8: VD2 device manager architecture.

The **rendering kernel** (called 'Y') is based on OpenGL. It loads the geometry, maintains a hierarchical *scene graph* and renders it.

VD2 stores the vertexes of a face as pointers to a single point array. Each face can have its own material and appearance (e.g. wire-frame, solid). An object is built from a list of faces and a list of points. This design has several advantages. It prevents errors, like arrays which are too small, not allocated or incorrectly indexed. All attributes of a point are easily accessible. Furthermore, faces can be accessed directly and its attributes changed independently. This is very important for collision detection to show colliding faces. The main disadvantage to this approach is that memory is used very extensively, particularly if only point positions are needed. If this is a major problem for an application, it should be easy to add another node type to the system that is based on separate arrays or on smaller point-structures.

Finally, the renderer supports multiple graphics pipes and more than one rendering window per graphic pipe. The rendering kernel also offers several built-in functions for stereo viewing. Stereo viewing can be achieved with dual pipe rendering, shutter (active stereo), MCO-style, interlaced, or anaglyph (green-red stereo).

2.3.2 Virtual scene and *scene graph*

The so-called *scene graph* essentially is the digital representation of a virtual environment. In other words, it is a structure aimed at containing a set of pointers to the *objects* that constitute the scene. In this context, the term *object* refers not only to the visual elements of the virtual environment, such as points, lines, surfaces and geometries, but also to many other items necessary to *render* the scene, such as the light sources and the current point of view.

Moreover, in order to make the interaction with the virtual scene the more realistic as possible, the *scene graph* must provide a way to consider also the relationships among the different objects. For instance, it is necessary that the movement of the object *"container"* will result in the corresponding movement of all the *"content"* objects. For this purpose, the *scene graph* not only stores the informations related to the different objects in the virtual scene, but also the *hierarchical relationships* existing among them.

This is achieved by means of a tree structure. As a tree, the *scene graph* basically has two kinds of elements (Figure 2.9):

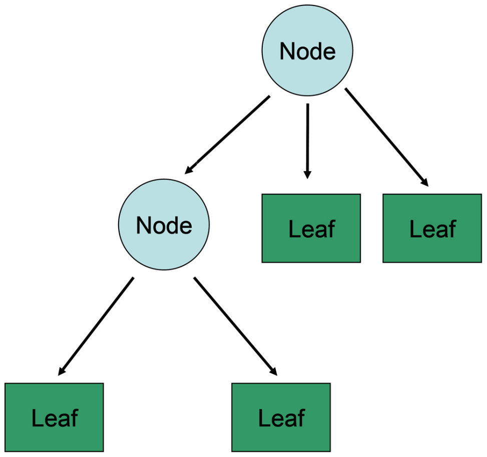


Figure 2.9: The tree structure of the scene graph

Nodes are objects which reference any number of other objects, that can be, in turn, other nodes or terminal elements, called *leaves*. In the following, objects referenced by a certain node will be referred to as the *child* objects of a defined *parent* node. There are various types of nodes, which will be described in more detail in the sections which follow.

Leaves of a scene graph are node objects without child objects or objects which cannot make further reference to additional objects. These special objects are, for example, geometric objects. Geometric objects represent the visual objects in the scene (scene objects).

It is worth noticing that a parent object can have many children, but each child object can have only one parent. In other words, the scene graph is an *acyclic* structure.

2.3.3 Transformations

One of the most interesting characteristics of VR is the possibility of interacting with the objects in the virtual environment by moving them in real-time. The transformation of an object (translation, rotation, scaling) can be defined as a set of homogeneous matrices that will be pre-multiplied to the object. In other words, moving an object in the virtual environment simply means applying a transformation matrix to it. Thus, every object has its set of matrices, that can be modified in real-time.

During the *traversing* of the scene graph, the renderer multiplies transformation matrices automatically, taking into account also the matrices related to the parent node. In this way, a transformation applied to certain node affects also its children. This characteristic paves the way to the so-called *hierarchical modeling*, that will be discussed in the following.

2.3.4 Authoring the Virtual Environment

A virtual reality software cannot be limited to the rendering. Indeed, the proper handling of the interaction among the elements in the virtual environment is a key issue. *Handling the interaction* means to provide the virtual environment with a certain behavior, that essentially is the ability of the environment to react dynamically to particular events. As aforementioned, the software module that handles the interaction is called **interaction manager** and, generally, it can be based on two different approaches [Zac00]:

Event-based approach According to this model, the programmer (*author*) first defines a sort of *story-board*, that describes the behavior of the virtual environment. Usually, this is represented by a script file in which a set of event-action relationships are defined. An event may be the push of a button, the occurrence of a collision, the expiration of a timer, etc. The actions may be instead very different: the turning on of a light, the playing of a sound, and, more generally, the modification of any geometric/visual characteristic of one or more elements in the virtual scene. In short, the interaction manager waits for the events defined in the story-board and when one of them occurs, it takes the predetermined action. Obviously, according to this approach, the virtual environment will react only to predetermined events, while all others will be ignored.

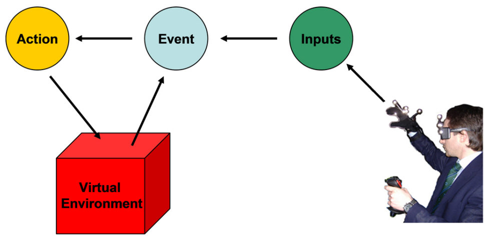


Figure 2.10: Event-based approach

Behavioral-based approach This is a more complex model: each object in the virtual environment is an autonomous agent with a number of receptors for certain virtual inputs. In this way, each element can react to a condition according to its own dynamics and thus have its own behavior. According to this approach, the result of the interaction between two objects will be automatically determined, without the need to anticipate every possible interaction.

Since *Virtual Design 2* is an industrial-oriented software, its interaction manager follows an event-based approach. Thus, programming VD2 means defining a set of event-action relationships by means of a configuration file. The grammar of the configuration file is quite simple and allows the programmer to call external library functions or even to generate user-defined events.

Authoring a virtual environment is a fundamental activity: the more accurate it is, the more realistic the behavior of the environment will be and accordingly the reliability of simulation results.

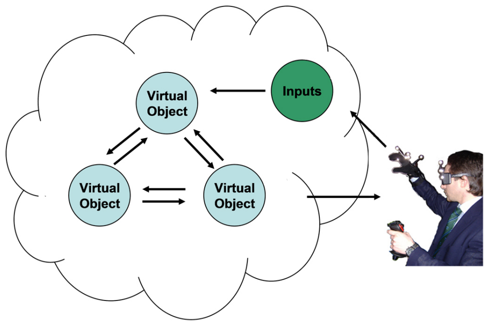


Figure 2.11: Behavioral approach

2.3.5 The basic *actions* set

VD2 provides a complete set of commands for planning the behavior of the VE in relation to the user interaction. Many commands describe *actions* that operate on the objects in the VE. Every *action* is triggered by a certain event, as defined in the so-called *scene description file*. The most important *actions* used for robotic simulations are:

- **grabbing**: what makes the virtual experience really “interactive” is the possibility to grab virtual objects and to move them through the scene. The “grab” action first makes an object “grabbable”. Then, when the hand touches it, it will be attached to the hand.

- **changing object attributes**: these actions allow the user to change some objects attributes, such as materials, visibility, position, etc.

- **sweeping**: this action traces the movement of an object in the VE, by replicating its shape.

- **animations**: some actions allow the user to record and playback the movement of one or more objects of the virtual scene.

- **gravity**: this feature increases the realism of the virtual world, making objects fall in a certain direction and bounce off some “floor objects”, that can be specified separately for each object.

- **constraints**: VD2 kernel allows the user to constrain the movement of an object in the virtual environment. These constraints provide an easy way to define simple interactive kinematics, such as virtual doors and car hoods. By default, when the constrain action is active, the object is linked to the virtual hand, so that it tries to follow the hand's motion but only within the constraint.

2.3.6 Dynamic Shared Objects

As aforementioned, the features provided by VD2 kernel can be enhanced by functions defined in external modules, called *Dynamic Shared Objects* (DSO).

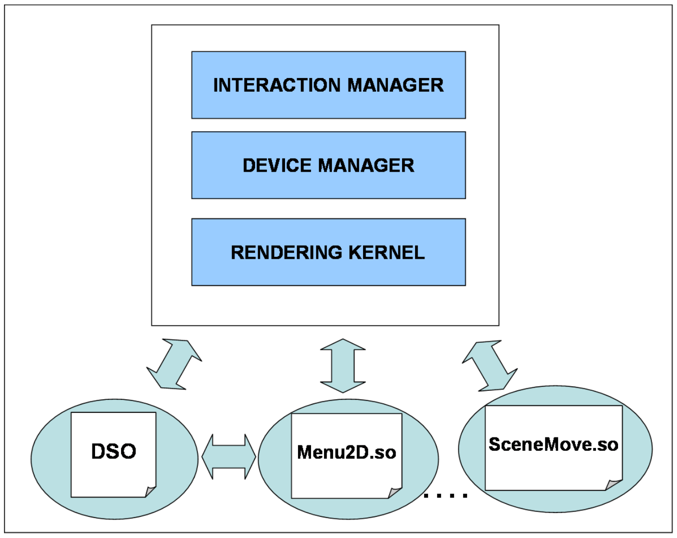


Figure 2.12: DSO modular approach.

Generally, each DSO module contains a set of functions developed for a specific application target, as a *plug-in*. The basic installation of Virtual Design 2 already provides many *plug-ins*, for instance to manage interactive menus or to make snapshots of the virtual scene.

A DSO module is a *dynamically linkable* object file, which allows the linking of the module to VD2 kernel to be made at run-time, [PW72]. Moreover, the module is *shared*, meaning that many different processes can share the library functions at the same time (Figure 2.12). This modular approach offers three main benefits:

1. The object code is loaded in the physical memory only once and then it can be used by multiple processes via virtual memory management,[Dre06];

2. It is easy to add new features to Virtual Design 2 and maintain them;

3. The object code is linked to VD2 kernel only when the features implemented in the module are really needed.

The functions provided by DSO modules can be used as well as the basic commands, by specifying them in the *scene description file*. In particular, a DSO module exports at least four kinds of function:

init function This function is called by the simulation manager just one time, during the initialization process. A string is passed to the function. Generally the `init` function is used to start up the DSO module.

loop function It is called by the renderer every frame. It is mainly used to update the data structures of DSO module during the simulation. The loop function does not accept any argument.

exit function This function is called by the simulation manager just one time, when the simulation ends.

callback function This function is called when the event defined in the *scene description file* occurs, similarly to the basic bult-in actions. Usually a DSO module exports a number of *callback functions*. The simulation manager passes three arguments, in particular the binary *state* of the calling event (0 or 1) and a string pointer.

2.4 Virtual Design Review

Virtual Design Review (VDR) is a methodology that uses VR technologies to improve the development and the critical review of projects. It can be considered a way of implementing DMU [dZ99].

The data-flow related to VDR is summarized in figure 2.13.

Digital models from CAD tools are stored and maintained in a *Product Data Management* (PDM) system, together with all the other simulation data from different CAx[1] tools. The first step in order to carry out a Virtual Reality simulation is to properly prepare these data. Unfortunately, until now, this operation cannot be done automatically, at least for complex models, that actually need a VDR session.

Indeed, a VR expert has to import the geometric models, which often come from different CAD tools, and must operate some optimizations on their structure in order to make them suitable to be rendered in real-time.

[1] *Computer Aided everything.*

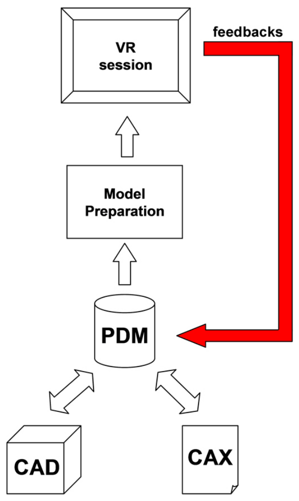


Figure 2.13: VR-based Design Review data flow

Moreover, the same expert has to prepare the virtual environment, considering the kind of analysis to perform, such as styling, assembly simulation, or even ergonomics evaluations.

Only after that, VR session can start. It is worth highlighting that VR simulation does not operate on the original CAD model, but instead on new *tessellated* geometry. This part of the methodology is quite well-established, especially in automotive field. Commercial VEMS provide the users with a number of virtual tools aimed at signaling criticality, selecting and hiding objects, verifying the accessibility of tools and parts, taking measures, etc. Furthermore, a stereoscopic visualization system helps the designers to analyze the model in all its details.

This approach has well-known benefits, if compared with traditional desktop-based design review. In particular:

- more design solutions can be evaluated in real-time;
- easier detection and resolution of technical problems, thanks to stereoscopic visualization;
- optimization of the *information flow* about the project, thanks to a more collaborative environment.

However, apart from these clear benefits, VR-based Design Review has still a weakness, that is highlighted by the red arrow in figure 2.13. In fact, to the best of author's knowledge, the efficient transferring of *feedback data* from VR to PDM is still a challenge. In this context, *feedback* is meant as meta-data related to the results of VDR session, such as the purpose of changing a product part or even a process.

For instance, we cannot import any geometry change directly to the original CAD model, since the model preparation actually is a one-way process. In other words, geometry tessellation is a *lossy data conversion*. Not to mention the number of CAD software that VR tools should eventually interface with. Moreover, often feedback data are not related to a single CAD model, but involve entire processes.

A possible way would be interfacing the virtual tools with PDM system, but unfortunately even this solution is not generally applicable. According to the author's experience, this is mainly due to the lack of a commonly accepted PDM system standard. Instead, each company tends to have its own PDM system, targeted on its particular needs. As a result, we cannot store feedback by means of a common "interchange format".

The author has faced this issue by developing several VR tools that do not directly interface with PDM system, but instead keep their feedback as general as possible. These modules will be described in the following subsections.

2.4.1 Markers and Comments

The interactive placement of markers into the scene is one of the most basic tool in any Design Review activity. Markers are useful for signaling any kind of design faults emerged.

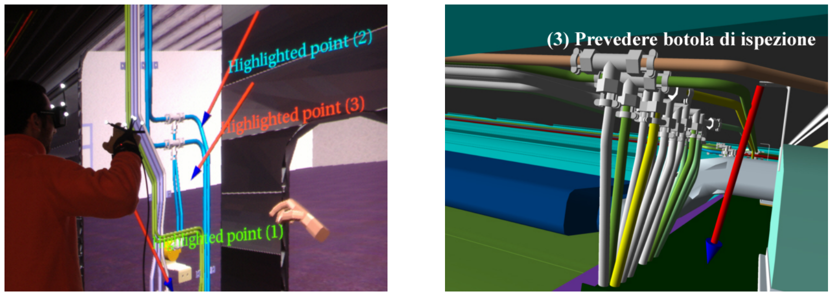
Starting from the framework described in the previous section, the author has developed a new software module which provides the following features:

- A snapshot of the virtual scene is taken every time a marker is placed;
- A console-operator can edit a comment related to the design faults highlighted;
- A HTML document containing both snapshots and comments can be automatically sent to a specified mailing-list.

An example is reported in figure 2.14

Moreover, the developed tool does not play at interfacing directly with a PDM system, anyway it provides an interesting feature that uses STEP[2]

[2]**ST**andard for the **E**xchange of **P**roduct model data.



(a) (b)

Figure 2.14: Markers and comments: a) markers placements; b) automatic snapshot.

[ISO94] in order to store the comments related to a user-selected geometry.

It is understood that this feature requires the application of a specific work-flow (Figure 2.15).

In particular, the CAD models should be firstly converted in STEP format and then prepared for VR session. In this way, we will have two models: a tessellated one for VDR and a parametric one that will be used to store the feedback from VR.

When a certain part is selected to be commented, its definition is searched inside the STEP file. Then, the comments related to the selected part are stored into the file as an instance attribute. At this point, the STEP model can be imported into original CAD software.

However, although the effectiveness of this methodology has been verified with its application to several commercial CAD tools, it must be noticed that the described work-flow becomes inapplicable for complex VR scenarios that involve heterogeneous data.

2.4.2 NURBS sketching in VR

Many researchers are involved in developing three-dimensional VR sketching systems [Dee95], [SRS91], [FdAMS02], [BMPR02], [IC09]. The basic idea is to provide the user with a set of VR tools for designing curves and surfaces directly in a three-dimensional environment, by means of ordinary VR input devices. For instance, these tools could be useful for quickly sketching new form concepts on the basis of an already existent CAD model displayed in a virtual environment.

Within the hardware and software framework presented in the previous

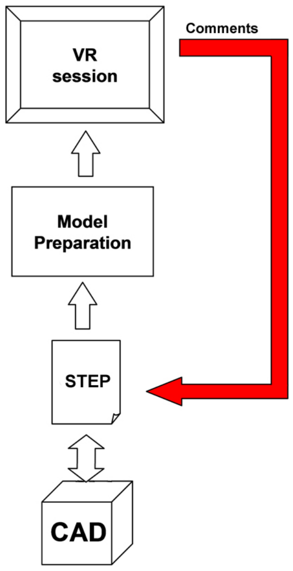


Figure 2.15: *VR-to-STEP* work-flow.

paragraphs, the author has developed a VR tool that allows the user to sketch cubic NURBS[3] curves directly in VR.

NURBS [PT66] refers to the mathematical representations of a 3D geometry, such as lines, circles, ellipses, spheres or even free-form geometries, such as cars bodies and organic shapes. However, probably the most important property of a NURBS is its *Projective Invariance*. In other words, if a projective transformation is applied to a NURBS, the same result can be achieved if the same transformation is applied to its control points. This means that projective transformations can be applied to NURBS, without loss of information.

The developed module is based on Cox-DeBoor [COX72] algorithm and allows the user to draw a NURBS curve that can be changed interactively in VR. Specifically, the user initially selects the first and the last control points of the curve. He can perform this task simply by drawing a line in the virtual environment with its virtual finger.

After that, the position of seven control points is highlighted by some spheres (Figure 2.16).

At this point, the designer can select and move any control point of the curve, simply by touching these spheres with its virtual finger while pressing

[3]Non-Uniform Rational B-Splines.

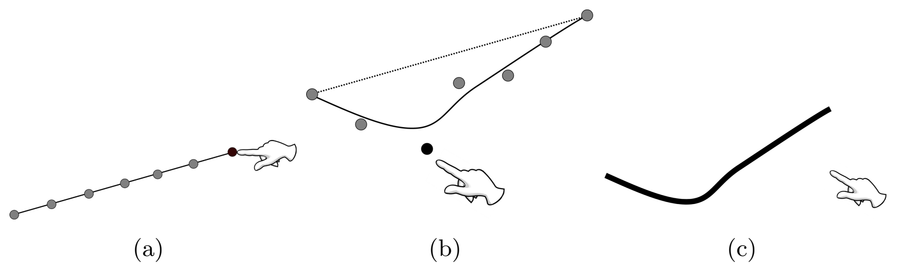


Figure 2.16: NURBS curve sketching in VR.

a button on the flystick. As a result, the user can sketch a curve in the virtual environment and save its geometry in different graphics formats[4].

Such a tool can be very useful for instance to sketch the position of pipes and wirings during a Design Review session. A future research direction can be the exporting of the control points in STEP format in order to integrate VR-sketched NURBS directly into CAD models.

2.5 Conclusions and future work

VR framework described in this chapter has been intensively used for a number of Design Review sessions [DPT09], [DPT07], [DPT07], [DT08]. The results of this research activity have been verified through collaboration with various manufacturing companies operating in transportation and nautical field and the developed tools have shown their great potential in improving the critical review of the projects.

However, the future research directions will mainly focus on the development of different tools, each targeted to a particular VR-based analysis. Finally, further studies should be carried out in order to improve the exchanging of feedback data between VR and CAD.

[4]currently only *OpenInventor* and VRML 1.0

Chapter 3

RoboTiX: a VR engine for robot control

3.1 Introduction

VD2 computational engine does not provide native support for defining and handling kinematic chains. For this reason, the author has developed a DSO module, called *RoboTiX*, aimed at managing open kinematic chain manipulators in Virtual Reality. The basic idea was to provide the users with an interactive interface for designing, studying and controlling robot manipulators without the need for complex programming [FI98]. This has been achieved with a VR-based interface which allows the user to move in real-time the robot manipulator arms, in both joint and operational space. The application kernel is a general kinematic generator which can calculate in real time the inverse kinematics of any open-kinematic mechanical structure (Figure 3.1). The description of the robot kinematic characteristics are specified in a text-based *configuration file*, which uses a simple syntax that will be described in the following. Inside this VR environment, the user can interact with the robot in an effective and intuitive way. For instance, the operator can pick any part of the robot and directly move it by means of a Multi-Dimensional Interactive (MDI) device, such as the virtual glove, or even control the end-effector positioning with a *spacemouse* or a *joystick*. Thus, trajectories can be defined, optimized and stored easily. The interface also provides the user with information about current parameters of the robot to help the user in trajectory decision making.

Moreover, during the trajectory planning, the user can display the *manipulability ellipsoid* related to the current posture. The ellipsoid is properly oriented in the three-dimensional environment and it is centered on the current end-effector. This feature allows the user to immediately identify the most dexterous posture for performing the considered task. Obviously, only linear components of the velocity are considered.

Finally, a simple collision avoidance algorithm has been implemented. The user can actually specify a virtual object from which the robot must be away during its moving.

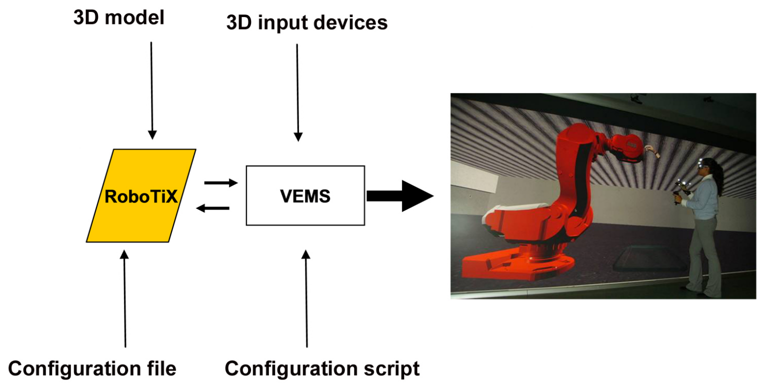


Figure 3.1: VR framework

In short, the main characteristics of RoboTiX module are:

- **Flexibility** The software and its functions are suitable for any type of open kinematic chain manipulator;
- **Programmability** The robot can be programmed directly in the virtual environment. Moreover, it is possible to store and reproduce an user-defined path;
- **Usability** The user can control the robot in real-time in both joint and operational space;
- **Integration** The module is completely integrated with underlying software framework. For instance, any eventual error condition, such as end-of-stroke, is signaled to VD2 kernel in order to be properly handled;
- **Redundancy handling** The software can handle eventual redundancy issues.

3.2 Robot hierarchical model

The first step for using *RoboTiX* module is a proper modeling of the robot mechanical structure[1].

In general, a kinematic chain is a set of rigid elements, called *links*, connected by *joints*. A joint essentially is a constraint on the geometric relationship between two adjacent links. The basic idea has been to use the *scene-graph* tree structure in order to keep the logical sequence of the different links.

Thus, the geometric model has to be arranged in such a way that each joint of the chain is represented by an *assembly node* containing three elements (Figure 3.2):

- the geometry of the link (visual element);
- an assembly node representing the subsequent joint (logical element);
- the axis related to the subsequent joint.

This structure is repeated until the final node (*hierarchical modeling*).

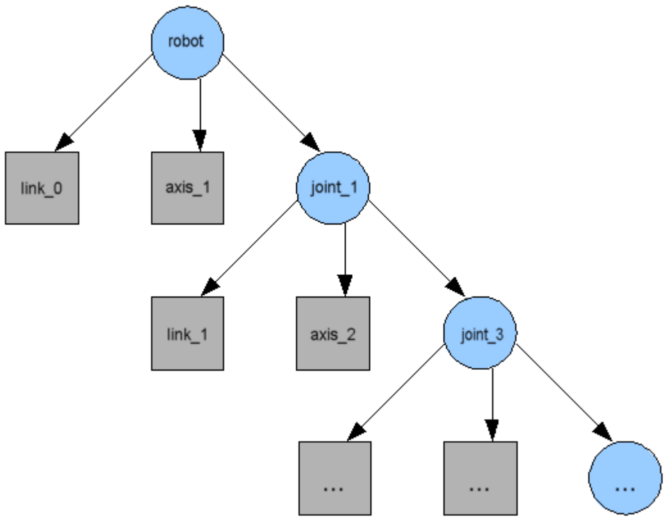


Figure 3.2: Robot hierarchical model.

[1]In general, CAD models of industrial robots are available for free, because the robot companies tends to help their potential customers in designing new production plants.

Thanks to the hierarchical structure of the *scene graph*, the programmer does not have to be concerned about the analytic solution of the direct kinematic problem. In fact, as aforementioned, during the traversing of the scene graph the renderer will apply to each element also the transformation matrix of its parent node and so on. This means that the movement of a generic joint, such as a rotation about its axis, will cause the rotation of all the subsequent links of the kinematic structure around the same axis.

In other words, for each i-node, the renderer will calculate its accumulated homogeneous matrix $\mathbf{T_i^0}$, where:

$$\mathbf{T_i^0} = \mathbf{T_1^0}\,\mathbf{T_2^1}\,..\mathbf{T_i^{i-1}}$$

In this way, the direct kinematic problem, which essentially is the computation of the $\mathbf{T_i^0}$ matrix, is numerically solved, without the need of explicitly implementing Denavit-Hartenberg (D-H) algorithm [SS00]. Moreover, the accumulated homogeneous matrix $\mathbf{T_e^0}$, which defines the position and orientation of the *end-effector* with respect to the world origin, is always available as well.

However, it is worth pointing out that the hierarchical model above described is only suitable for open kinematic chains, because, as aforementioned, the *scene-graph* has an *acyclic* structure.

3.3 Robot Configuration file

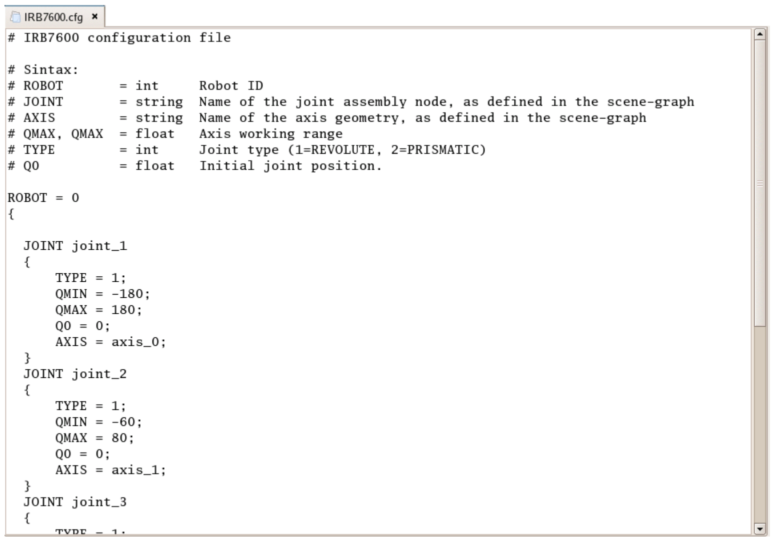
One of the most important goals is the flexibility of the module: in other words the DSO module should be able to handle different types of kinematic chain, independently from number and type of the axes the robot is equipped with. In order to achieve this, the kinematic chain has to be described in a configuration file, which specifies not only names of joints and axes, as defined in the robot *scene-graph*, but also type (revolute or prismatic) and working range of each axis, see Figure 3.3.

The *configuration file* simply is an ASCII file with several lines, such as:

```
# This is a comment

ROBOT = <robot_number>
{
   JOINT <joint_name>
   {
     AXIS = <axis_name>
```

```
    TYPE = <joint_type (1=REVOLUTE 2=PRISMATIC)>
    QMIN = <upper_end_of_stroke_angle>
    QMAX = <lower_end_of_stroke_angle>
  }

  JOINT <joint_name>
  {
    AXIS = <axis_name>
    TYPE = <joint_type (1=REVOLUTE 2=PRISMATIC)>
    QMIN = <upper_end_of_stroke_angle>
    QMAX = <lower_end_of_stroke_angle>
  }

  ...
}
```

The order of the joints definitions should respect the order of the links. Finally, notice that the keyword `ROBOT` allows the user to define the characteristics of multiple robots in the same file.


IRB7600.cfg

```
# IRB7600 configuration file

# Sintax:
# ROBOT       = int     Robot ID
# JOINT       = string  Name of the joint assembly node, as defined in the scene-graph
# AXIS        = string  Name of the axis geometry, as defined in the scene-graph
# QMAX, QMAX  = float   Axis working range
# TYPE        = int     Joint type (1=REVOLUTE, 2=PRISMATIC)
# Q0          = float   Initial joint position.

ROBOT = 0
{

  JOINT joint_1
  {
      TYPE = 1;
      QMIN = -180;
      QMAX = 180;
      Q0 = 0;
      AXIS = axis_0;
  }
  JOINT joint_2
  {
      TYPE = 1;
      QMIN = -60;
      QMAX = 80;
      Q0 = 0;
      AXIS = axis_1;
  }
  JOINT joint_3
  {
```

Figure 3.3: Example configuration file for an Industrial robot.

3.4 Robot task planning

Early industrial robots were programmed by moving the robot to a desired goal point and recording its positions in a memory, which the sequencer would read during playback. During teaching phase, the user can guide the robot directly by hand, or through the interaction with a *teach pendant*, that is a hand-held control terminal which allows to move each joint of the manipulator, [Cra03].

The DSO module provides functions to simulate in Virtual Reality both aforementioned teaching systems. Indeed, it is possible to control the kinematic chain through the *flystick*, that is a wireless interaction device designed especially for VR applications, or to make the robot follow a tracked object, such as the virtual hand.

Moreover, the set of actions described in the section 2.3.5, can be specified in the *scene description file*, in order to carry out the robotic simulation as more realistically as possible.

3.4.1 Path planning

The DSO module allows to define different postures for the robot, by specifying each of them in the *scene description file*. Moreover, it is possible to handle the kinematic chain in real-time, by relating an input from a VR device, such as a button of the *flystick*, to the handling of a specific joint, see Figure 3.4.

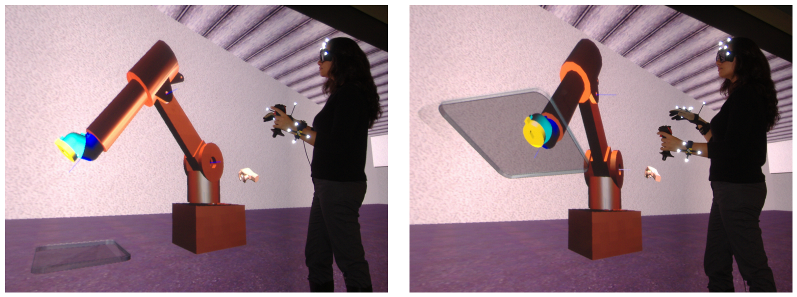


Figure 3.4: Real-time path planning.

In this way, the user causes the robot to assume the desired posture, by using the *flystick* as a *teach-pendant*. Each posture can be stored in a file, so that the user can define a point-to-point path. The reproduction of the defined path then can be triggered by an event, as it happens for any other action in the VE. Thus, the features described above provide an easy way to plan a collision-free path for the robot task. Furthermore, the integration with the underlying software architecture also allows the user to plan quite complex behaviors, so that the robot can manipulate objects or manage any eventual collision, see Figure 3.5.

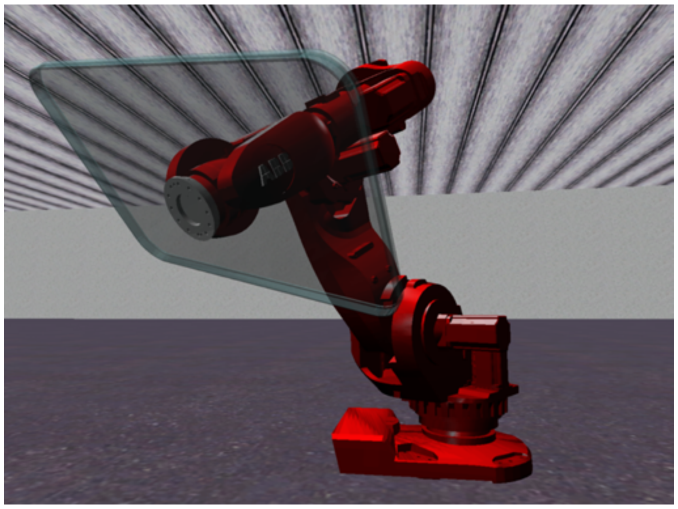


Figure 3.5: The manipulator reproducing an assembly task.

3.5 Inverse kinematics

As aforementioned, a *kinematic chain* consists of a set of *links* connected by a certain number of *joints* (Figure 3.6).

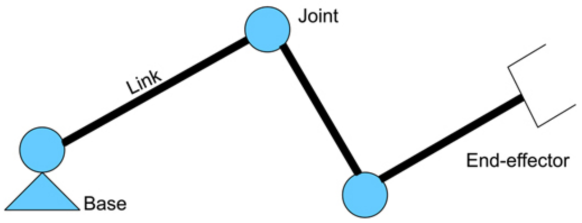


Figure 3.6: Links and joints of a kinematic chain.

The configuration of the chain at a certain time (*state*) can be defined as the set of its joint variables. The dimension of this set is the *Degree Of Mobility* (DOM) of the kinematic chain. Generally, in this work, the *Degree*

Of Freedom (DOF) is intended as the set of independent parameters required to completely specify the position and orientation of the end-effector only[2].

When DOM is greater than DOF, the kinematic chain is *redundant*, because there are DOM-DOF+1 set of joint variables that can determine the same configuration for the end-effector.

It is understood that finding the position of end-effector, given the *state* of the chain, is a trivial issue, but the contrary, namely finding the joints values for a specified configuration of the end-effector, generally remains a challenge (*inverse kinematics problem*). Indeed, this problem often does not have analytic expression, or has more than one solution, even for apparently simple chains.

The developed DSO module implements a general purpose inverse kinematics algorithm. The goal has been to develop a generic solver, suitable for different kinematic chains with an arbitrary number of joints. The result has been achieved following two different methods described in the next sections.

3.5.1 A numerical approach

With reference to the control of the end-effector of a n-joint robot, the well-known relation

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3.1}$$

maps the joint space velocities $\dot{\mathbf{q}}$ into the operational space velocity $\dot{\mathbf{p}}$ of the end-effector, where $\mathbf{J}(\mathbf{q})$ is the $(m \times n)$ Jacobian matrix.

This mapping may be inverted using the *pseudo-inverse* of the Jacobian matrix[3], i.e.,

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}(\mathbf{q})\dot{\mathbf{p}} \tag{3.2}$$

where $\mathbf{J}^{\dagger}$ is a $(n \times m)$ matrix, which corresponds to the minimization of the joint velocities in a least-squares sense [SS00]:

$$\mathbf{J}^{\dagger} = \mathbf{J}^{\mathbf{T}}(\mathbf{J}\mathbf{J}^{\mathbf{T}})^{-\mathbf{1}} \tag{3.3}$$

Unfortunately, the analytic expression of the Jacobian Matrix $\mathbf{J}$ strictly depends from the considered kinematic chain. In other words, we cannot obtain a general expression for $\mathbf{J}$ that can be suitable for every kind of kinematic chain.

[2]Thus, the concept of DOF is task-dependent. In any case, it is $DOF <= 6$

[3]also called *Moore-Penrose* inverse.

However, we can differentiate the 3.1 as:

$$\mathbf{dp} = \mathbf{J}(\mathbf{q})\mathbf{dq} \tag{3.4}$$

Thus we have the following analytic expression for $\mathbf{J}(\mathbf{q})$:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial p_1}{\partial q_1} & \frac{\partial p_1}{\partial q_2} & .. & \frac{\partial p_1}{\partial q_n} \\ \frac{\partial p_2}{\partial q_1} & \frac{\partial p_2}{\partial q_2} & .. & \frac{\partial p_2}{\partial q_n} \\ .. & .. & .. & .. \\ \frac{\partial p_m}{\partial q_1} & \frac{\partial p_m}{\partial q_2} & .. & \frac{\partial p_m}{\partial q_n} \end{bmatrix} \tag{3.5}$$

Thanks to the hierarchical modeling (see Section 3.2), $\mathbf{p}(\mathbf{q})$ is always numerically known and equation (3.5) suggests the following numerical solution:

$$J_{ij} \cong \frac{p_i(q_j + \Delta q) - p_i(q_j)}{\Delta q} \tag{3.6}$$

where Δq is an arbitrary small variation from the current joint value q_j.

At this point, the solution of inverse kinematics problem passes through the numerical inversion of the Jacobian matrix.

Unfortunately, equation 3.3 is not always applicable. In the so-called *near-singular* postures, in fact, the Jacobian matrix becomes very sensitive to small changes in joint values and the determinant of $\mathbf{JJ^T}$ abruptly takes very small values. It is understood that these posture cause many numeric problems in inversion, with sudden and high variations of $\dot{\mathbf{q}}$ that can determine shakes and jitters of the kinematic chain. For this, a *Damped Least Square* (DLS) inversion method[4] has been adopted [Wam86]:

$$\mathbf{J}^* = \mathbf{J^T}(\mathbf{JJ^T} + \lambda^{\mathbf{2}}\mathbf{I})^{-\mathbf{1}} \tag{3.7}$$

This method is theoretically justified in [WIL88], but we can roughly say that the *damping factor* λ "dirties" the rigorous solution (3.3) in order to keep it numerically stable. Its value must be a trade-off between numerical stability of $\mathbf{J}^*$ in near-singular postures and accuracy of solution in all other cases. There have been a number of methods proposed for select the value of λ considering the current posture, for instance [OAJ09], [MWM92], [DW92]. However, the author has chosen:

$$\lambda = k\left(\frac{1}{1 + \|\mathbf{JJ^T}\|}\right) \tag{3.8}$$

[4]It is also known as Levenberg-Marquardt method.

In this way, when the robot is in near-singular postures $\|\mathbf{JJ^T}\| \cong \mathbf{0}$ and thus $\lambda = k$. In all other cases, with a proper choice for the constant k, λ becomes negligible. Moreover, the possibility of changing in real-time the value of constant k has been also provided to the user.

3.5.2 A simpler approach

Generally, finding the joint angles for a given position of the *end-effector* in the operational space requires an inverse kinematic approach, as described in [ZB94]. Since this analysis is limited to open kinematic chain manipulators, a simpler but effective methodology can be adopted.

As aforementioned, VD2 provides a specific *action* that cause a virtual object to follow the user hand, within a specified constraint. For instance, an object can only rotate about a defined axis, according to the movement of the virtual hand. Unfortunately, each constraint is related to a single object in the *scene-graph* and it is treated separately from the other constraints. In other words, the user cannot define directly kinematic relationships among two or more virtual objects.

This approach is suitable for modeling simple kinematics, such as a virtual door, but it can lead to an unexpected behavior when it is applied to a kinematic chain, because generally each link of the chain will move independently from the other elements, as illustrated in Figure 3.7.

In order to avoid the breaking of the kinematic chain, each constraint operates on a different joint-node of the hierarchical model described in section 3.2, rather than directly on the geometries of each link.

In fact, thanks to the hierarchical modeling, each constraint action affects a different joint-node, that contains many links of the chain. For instance, according to the kinematic chain shown in Figure 3.8, the first joint-node contains the whole kinematic chain, the second includes only the last two links and finally the third node is just the last link of the chain.

Since all the geometries belonging to a specific joint-node act as a single "rigid object" during the movement, the geometric relationships among the different links will be kept in any case.

Many constraints can be triggered by a single event, such as the collision between the virtual hand and a specific link of the robot. In this way, the user can drag the whole kinematic chain by "grasping" the *end-effector* until the robot reaches the desired position, see Figure 3.9.

At the same time, it is also possible to move by hand only one or more links of the chain. However, this approach has some limitations:

- Eventual kinematic redundancy issues are not properly handled;

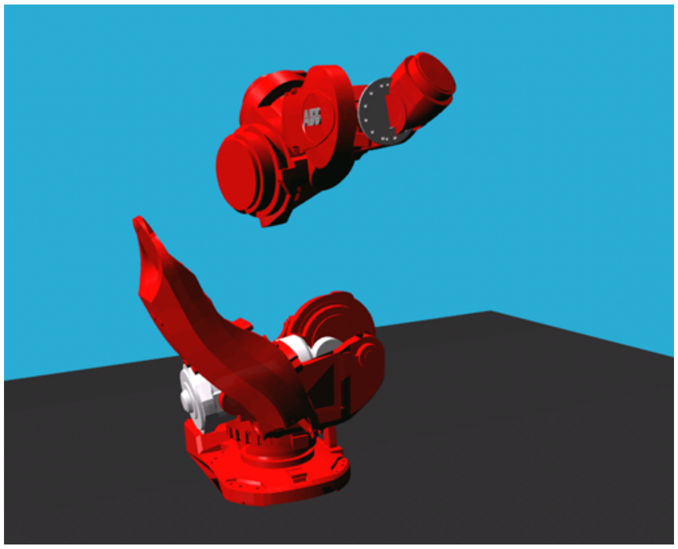


Figure 3.7: Non-hierarchical modeling: The kinematic chain breaks while moving.

- It is only suitable for open kinematic chain manipulators;

3.6 Manipulability ellipsoid

It is understood that Jacobian relates joint space velocities to the operational space velocity. Thus, a unit sphere in the joint space will be mapped as a generalized ellipsoid in the task space, where principal axes of this ellipsoid represent the direction that the arm is easy to move. For this reason, this ellipsoid is called *manipulability ellipsoid*.

Displaying the manipulability ellipsoid related to the positional Jacobian can be very useful to visually evaluate the attitude of the current posture to achieve a certain task (Figure 3.10). In order to define the axes of such an ellipsoid, it is necessary to diagonalize the Jacobian.

Since generally Jacobian is a not-square matrix, we cannot define its eigenvectors, but we can use instead the *Singular Values Decomposition* (SVD) method [Str09]. SVD allows us to factorize Jacobian matrix as follows:

$$\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}} \tag{3.9}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\boldsymbol{\Sigma}$ is diagonal. If $\mathbf{J}$ is $m \times n$ matrix, then $\mathbf{U}$ is $m \times m$ and $\mathbf{V}$ is $n \times n$. It should also be noted that singular

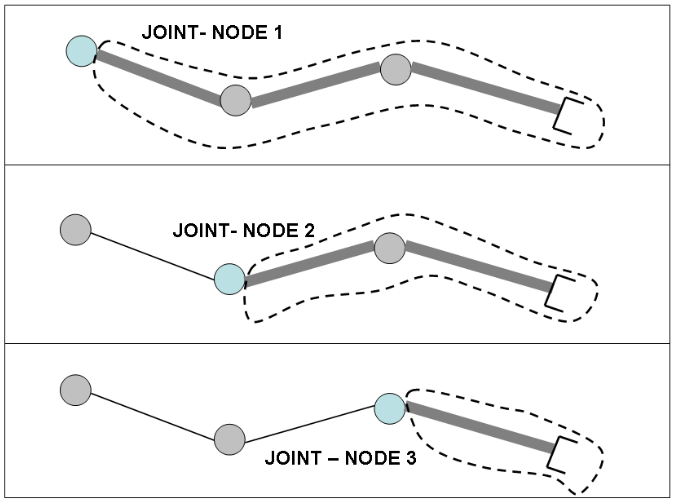


Figure 3.8: Hierarchical modeling: the joint-nodes of a kinematic chain.

value decomposition is not unique, but **U** and **V** are always orthogonal matrices. However, we can assume:

$$\mathbf{U} = [\mathbf{u_1}\ \mathbf{u_2}\ ..\ \mathbf{u_m}] \tag{3.10}$$
$$\mathbf{V} = [\mathbf{v_1}\ \mathbf{v_2}\ ..\ \mathbf{v_n}] \tag{3.11}$$

where $\mathbf{u_i}$ and $\mathbf{v_j}$ are respectively the eigenvalues of $\mathbf{JJ^T}$ and $\mathbf{J^TJ}$. They are respectively called left and right *singular vectors* of **J** matrix.

For **Σ** instead we have:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad \mathbf{D} = \mathbf{diag}\{\sigma_\mathbf{1}\ \sigma_\mathbf{2}\ ..\ \sigma_\mathbf{r}\} \tag{3.12}$$

where σ_r are the so-called *singular values* of the Jacobian.

The principal axes of the ellipsoid are given by the left singular vectors of the Jacobian and the lengths of the principal axes are given by the singular values of the Jacobian.

In particular, RoboTiX module performs the SVD decomposition with *Jacobi eigenvalue algorithm* [Ves79], then it draws the manipulability ellipsoid centered in the end-effector. The user can see the ellipsoid changing during the movement of the chain and choice the posture that better suits to certain task. Obviously, only the three linear components of the velocity are considered.

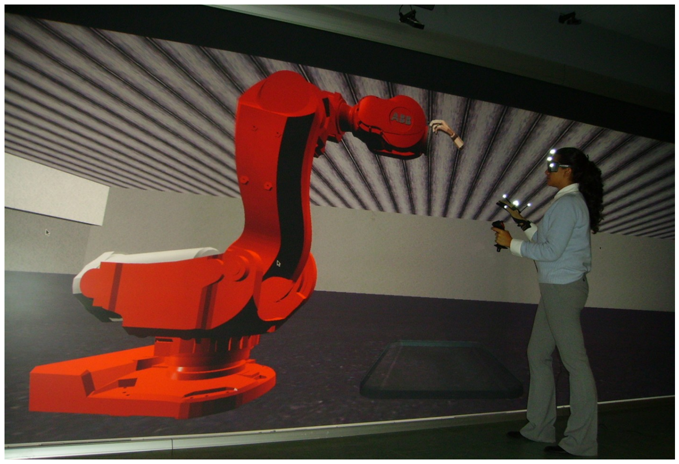


Figure 3.9: The kinematic chain being dragged by the virtual hand.

3.7 Redundancy handling and collision avoidance

Generally, referring to redundant manipulators, Jacobian is a “fat” rectangular matrix. This means that equation (3.1) has more than one solution. In other words, there are different postures that causes the same position for the end-effector, depending on the degree of redundancy of the kinematic chain (Figure 3.11).

The idea is to take advantage of the redundancy of the manipulator in order to generate movements that satisfy other low priority tasks, different from the main task assigned to the end-effector. For instance, the end-effector moves along the planned trajectory, while the kinematics structure moves in order to avoid a possible obstacle.

A possible way to achieve this is to reformulate equation (3.1) in terms of a constraint that must be satisfied, while a cost function $g(\dot{\mathbf{q}})$ is minimized. Given the cost function, this kind of problem can be solved by means of *Lagrange multipliers method* [SS00].

We can define a low-priority task in terms of a cost function as follows:

$$g(\dot{\mathbf{q}}) = \frac{1}{2}(\dot{\mathbf{q}} - \dot{\mathbf{q}}_{\mathbf{t}})^T(\dot{\mathbf{q}} - \dot{\mathbf{q}}_{\mathbf{t}}) \tag{3.13}$$

where $\dot{\mathbf{q}}_{\mathbf{t}}$ is a *target posture* expressed in joint velocities space. Indeed, finding a solution for (3.1), while minimizing $g(\dot{\mathbf{q}})$ means to achieve the main task, while the posture of robot keeps as near as possible to $\dot{\mathbf{q}}_{\mathbf{t}}$.

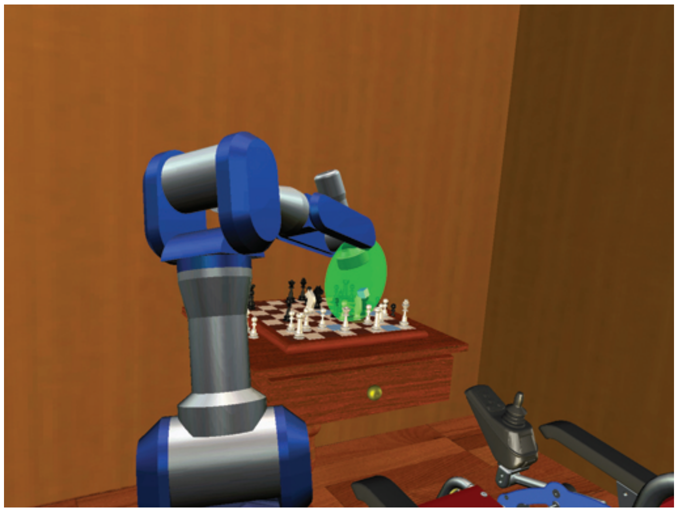


Figure 3.10: The manipulability ellipsoid during the task execution.

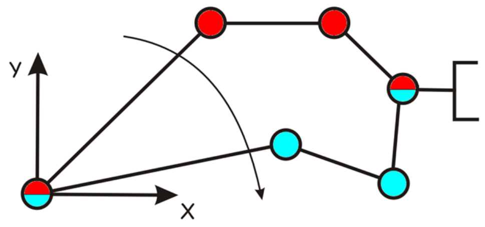


Figure 3.11: Different postures achieve the same position.

A general solution for this problem is:

$$\mathbf{\dot{q}} = \mathbf{J}^{\dagger}\mathbf{v} + \mathbf{N_J}\mathbf{\dot{q}_t} \tag{3.14}$$

where

$$\mathbf{N_J} = (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})$$

is the *null-space projector* for Jacobian matrix $\mathbf{J}$. In other words, the end-effector velocity $\mathbf{v}$ will not be affected by $\mathbf{\dot{q}_t}$, that will consequently be considered a low priority task.

Now, we have to define the expression of the vectorial function $\mathbf{\dot{q}_t}(\mathbf{q})$. For this, if we consider a further control point $\mathbf{v_1}$ (different from the end-effector) we can write:

$$\mathbf{v_1} = \mathbf{J_1}\mathbf{\dot{q}_t}$$

and consequently:

$$\dot{\mathbf{q}}_{\mathbf{t}} = \mathbf{J}_{\mathbf{1}}^{\dagger}\mathbf{v}_{\mathbf{1}} + \mathbf{N}_{\mathbf{J}_{\mathbf{1}}}\dot{\mathbf{q}}_{t_2} \tag{3.15}$$

where $\dot{\mathbf{q}}_{t_2}$ is a new secondary task in joint velocity space.

Now we can put equation (3.15) in (3.14) and then we have:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{v} + \mathbf{N}_{\mathbf{J}}(\mathbf{J}_{\mathbf{1}}^{\dagger}\mathbf{v}_{\mathbf{1}} + \mathbf{N}_{\mathbf{J}_{\mathbf{1}}}\dot{\mathbf{q}}_{t_2}) \tag{3.16}$$

This method can be reiterated till $\mathbf{N}_{\mathbf{J}_{\mathbf{i}}} = \mathbf{0}$. In this way we can have $r+1$ control points, where r is the *degree of redundancy* of the kinematic chain. They trajectories assigned to this further control points will be treated as nested low-priority tasks (Figure 3.12).

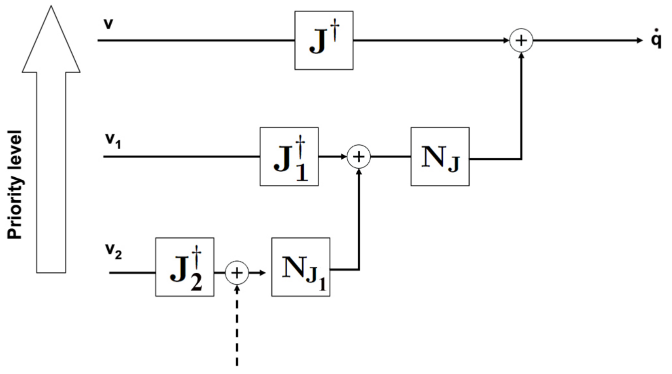


Figure 3.12: Different priority tasks.

Multiple control points with different priority level can be used for implementing collision avoidance algorithms. For instance, when the end-effector gets close to an obstacle, we can switch its priority level, in favor of a collision free trajectory, and so on.

In particular, *RoboTiX* module implements a collision avoidance algorithm that uses two priority level.

3.8 Exported functions

The developed module exports to VD2 kernel the following functions:

init `"<file_name>"`
: This is the initialization function. The argument is the *configuration file* of the kinematic chain(s) (see section 3.3).

loop It is the main loop of the program.

step `"dq1,dq2, ... dqN"`
: allows the user to move the robot in the joints space. The generic argument dq_i is the increment value of i-joint. This function can be useful to control the robot in real-time, by means of a button device, such as the *flystick* or simply the keyboard.

move `"[@ steps ];[R robot_number ] ; <q1 | *>, q2, .., <qN>"`
: moves the robot to the joint-space posture (q_1, q_2, ... q_n).

  The keyword "`@`" specifies that the posture will be achieved in `steps` rendering cycles, otherwise the software uses the default step increment. (see `setStep`).

  The keyword "`R`" specifies which robot to move, otherwise the software moves the currently active robot (see `setRobot`).

  Finally, the keyword "`*`" (asterisk) keeps unaltered the position of the respective joint.

stop (no args)
: stops the movement of the robot.

setEventEOS `"<event_name>"`
: Set the event to generate when an end-of-stroke condition is achieved during the movement.

setEventMove `"<event_name>"`
: Set the event to generate when the robot moves.

printvar (no args)
: Prints the current posture to standard output

setStep `"[+|-]<steps>"`
: Set the default duration of the movement to `<steps>`. If specified, the keywords `[+|-]` respectively increment or decrement the current steps number.

setEndEffector `"<joint_number>"`
: Set the last joint of the chain on which the inverse kinematics algorithm is active. It will be considered the end-effector.

setFirstJoint `"<joint_number>"`
: Set the first joint of the chain on which the inverse kinematics algorithm is active.

setRobot `"<robot_number>"`
: Set the robot currently active.

move_xyz `"(x,y,z)"`
: Moves the current end-effector of the currently active robot to the specified position.

displayEllipsoid (no args) Display the manipulability ellipsoid related to the current posture.

initEllipsoid `"[material_name]"`
: sets the ellipsoid material with `material_name`. If the material is not specified, a default material is applied to it.

play `"<file_name>"`
: Reproduces the path stored in `file_name`.

save `"<file_name>"`
: Save the current state of the kinematic chain (posture) in `<file_name>`. If the file already exists, then the data are appended to it.

renew (no args)
: Reloads configuration file, reset all module data structures. Debug purpose only.

move_p2p `"(x,y,z)"`
: Save the current posture and moves the current end-effector of the currently active robot to the new specified position. If the trigger event status is zero, the robot returns to the previously stored posture.

setObstacle `"<object_name>; <float>"`
: the robot will move always keeping itself at a distance of `float` from `object_name` (collision avoidance).

trackInit `"<device>"`
: allows the user to control the kinematic chain in real-time by means of the specified MDI device.

trackInitScale `"<float>"`
: sets the default scale for the device data.

trackInitDampingFactor `"<float>"`
: sets the damping factor k to `<float>`. See section 3.5.

trackToggle (no args)
starts/stop the real-time control.

trackSetScale `"<float>"`
sets the data scale to `<float>`.

trackExit exit function.

An example script is reported in the following:

```
appl init robotix.so:init "robot.cfg"
appl loop robotix.so:loop
appl callback robotix.so:step "(1, 0, 0, 0)" \
              key Left type reflect event on

appl callback robotix.so:move "@500;(20,-30,40,*)" key l
appl callback robotix.so:stop \
              collision end_effector obstacle
appl init robotix.so:setEventEOS "end_of_stroke"
wireframe robot appinput "end_of_stroke" event change
```

3.9 Conclusions and future work

As aforementioned, the library functions allow the user to easily plan an intended task for any type of open kinematic chain manipulator: it is only necessary to prepare the robot *scene-graph* and then edit the configuration file according to the type of chain. Since the DSO module is completely integrated with the underlying software framework, the user can take benefit from all the other functionalities provided by the Simulation Manager. For instance, it is possible to display the working area of the robot, highlight eventual collisions between the robot and any object in the VE or trace the path of the end-effector (*sweeping*) during the task execution, see Figure 3.13.

The basic command set can also be used to simulate specific robot behaviors triggered by certain events, eventually generated by external modules. Thus, the modular approach adopted by VD2 kernel could be used to interface a real sensor network with a virtual robot cell. In this way, it would be possible to test the safety strategies adopted to control robots that operate in anthropic domains.

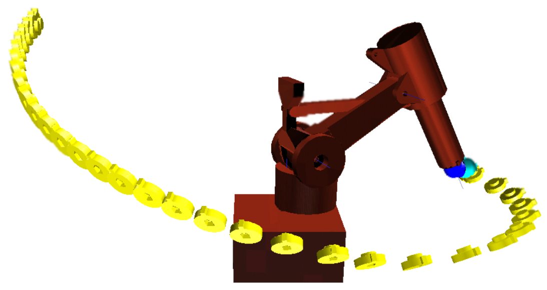


Figure 3.13: The *sweeping* action applied to the end-effector.

Furthermore, in order to use the function set provided by the DSO module, the user has to prepare the robot *scene-graph* and the related configuration file manually. Thus, a future goal will be to develop a graphic *wizard* to lead the user through the configuration process.

Chapter 4

Usability evaluation in VR

4.1 Introduction

The concept of usability comes from the studies of cognitive ergonomics about the development of graphical interfaces. A comprehensive review about the evolution of Human Computer Interaction (HCI) is reported in [SR08].

For instance, according to IEEE 610-12:1990 standard glossary [IEE90] the usability is the ease which a user can learn how to operate with, prepare inputs for, and interpret the outputs of the interaction between user and system.

ISO reference 9126-1:2001 [ISO01], which concerns the development of software products, defines the usability as the capacity of the software product to be included/understood, learned, used and attractive to the user when it is used under specified conditions.

Finally, ISO reference standard 9241-11:1998 [ISO98], which concerns the ergonomic requirements for the office work with Video Display Terminals (VDTs), defines the usability as the degree of use of a product by the user to achieve specific objectives with effectiveness, efficiency and satisfaction in a specified context of use.

Following the standard reference, Nielsen [Nie93] stresses the importance of the user satisfaction as a measure of the degree of pleasure related to the use of the system. Therefore, the usability evaluation cannot be considered apart from the assessment of the subjective aspects of the product-system interaction, that are extremely difficult to be evaluated systematically [MLG08].

Many studies about the methods to evaluate the usability have shown that such analysis can be performed not only with respect to the graphical interfaces of the computers, but also about the physical ones of the industrial products [Nor88], especially of "mass consumption", such as washing machines, mobile phones, etc. In this way, the design and usability metrics have also been extended to the field of the industrial design.

In particular, this chapter focuses on the study of usability of physical interfaces designed to drive *real-time controlled robots*, such as wheelchair-mounted manipulators. However, differently from previous studies, in which the attention is mainly focused on aspects mainly related to the *objective* features of the product, in this chapter the author highlights the importance of the *subjective* aspects arising from the user-product interaction. Furthermore, we will show that, thanks to virtual prototypes, it is possible to select the best architecture in terms of both usability and safety.

4.1.1 Usability in assitive robotics

Research in the field of assistive applications is playing a key role in the international robotics community. Several research groups are developing systems aimed at assisting disabled people in the actions and assignments typical of everyday life in both structured and unstructured domestic environments [GBSG03]. Indeed, the objective validation of the safety measures necessary to provide a dependable human-robot cooperation is becoming central for service robotics [AAB*06], [ASH07].

The main goal of robotics for assistance is to increase the quality of life of disabled people; in particular, robot manipulators are required, able to replicate human abilities in terms of strength, speed and accuracy in the manipulation of objects and tools. While such systems can offer autonomy to impaired persons, great challenges are presented by the study of the interface, the suitability of available robotic systems for special users, the usability, especially related to the kind of disability.

The assessment of the usability is a crucial issue for the design of such products, since they communicate with their users not only through their shape, but especially through their control interfaces.

In the following, we will show that these evaluations can be efficiently carried out via realistic simulation experiments.

In a first phase, the study has focused on defining a synthetic usability index on the basis of the methodologies currently in use. In a second phase, some experiments in Virtual Reality (VR) have been carried out. Indeed, the use of VR technologies for the collection of the experimental data has been fundamental in terms of safety, costs and repeatability of the tests.

Another important result has been the reduction of the sources of noise, thanks to preliminary simulations in VR and non-invasive questionnaires and interviews for capturing the subjective perceptions of users.

Finally, it is worth noticing that the developed model may show its validity also in evaluating the usability of other products. In fact, it provides a basis for a more extensive use of VR experiments for evaluating different design solutions in terms of global usability requirements, giving the user an active role in designing the product.

4.2 Usability evaluations in VR

Generally, the design of human-robot collaboration tools has to pay particular attention to the following issues:

- user's safety,
- system ergonomics and usability,
- cost-effectiveness.

The study of the aforementioned issues requires design tools able to simulate not only the robotic system and its control interface, but also unexpected behaviours in anthropic environments, depending both on the user and the system, such as the occurrence of mechanical/electronics failures or unexpected user movements within the robot workspace. Moreover, a realistic interface can be helpful for appreciating the cognitive Human–Robot Interaction (cHRI). The main advantage of the immersive Virtual Reality (VR) technology in the field of *assistive robotics* is the ability both to evaluate the control interface usability and to simulate the aforementioned dynamic events [BC03], [BC99].

The present study aims to provide a tool to easily recognize the criticality of a wheelchair-mounted manipulator, through the evaluation of its usability, taking into account not only the functional requirements, but also the subjective needs of the target user, which are not necessarily obvious. This objective is pursued through the identification of a metric for a quantitative assessment of the usability in order to compare different design alternatives.

Hence, the objective of this work is to demonstrate the effectiveness of a VR-based simulator (Fig. 4.2) for testing the usability and possible applications of wheelchair–mounted robot manipulators, with an effective solution for mounting available robot manipulators on commercial wheelchairs via a sliding rail (Fig. 4.2).

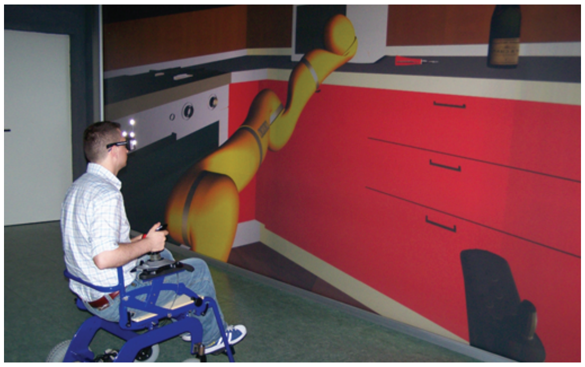


Figure 4.1: The wheelchair-mounted manipulator in the virtual environment

4.3 Wheelchair-mounted manipulators

Assistive robots can be divided in fixed structures and moving platforms. While the first solution often requires modifications of the infrastructures in order to provide a known environment, manipulators mounted on mobile vehicles or on wheelchairs offer higher flexibility. A good discussion of these issues has been addressed in [GBSG03]. It is worth noticing that often disabled people with upper-limb limitations also present mobility impairments which force them to use wheelchairs. A wheelchair-mounted manipulator [EB99], [HG94], [AMED05] can be an effective extender but, on the other hand, realistic simulations of the environment and extensive experimental activities have to be conducted for testing the effectiveness of their applications in unstructured domains. Moreover, safety issues have to be addressed in depth [AAB*06].

Finally, manipulators which almost replicate the kinematic structure of the human arm can be chosen for the legibility of their motion, which could improve the confidence of the users during physical Human-Robot Interaction (pHRI).

The use of Virtual Reality could speed up the design of such robotics solutions, because it allows the designer to set systems parameters based on feedback from experimenters, involving also cognitive aspects of the interaction with the robots. Such instrument can be used for a fast comparison of interface, appearance, kinematic parameters.

The research work described in this paper has consisted of two stages; namely, a concept stage in which the functional parameters of the wheelchair and the robotic arm have been defined, and an evaluation stage in which a

VR architecture has been developed in order to evaluate the usability of the system.

4.4 Concept stage

4.4.1 Requirements and robot choice

The integrated system has to guarantee the maximum effectiveness and usability. At the same time, the introduction of the robotic arm does not have to require any significant change on its surrounding environment. Moreover, the wheelchair-mounted manipulator has to satisfy the following requirements:

- reduced weight,
- intrinsic safety toward accidental collisions with the user.

The powered wheelchair Indoor 2003 by Neatech [Nea] has been chosen, based on consideration on its features, which are reported in Fig. 4.3.

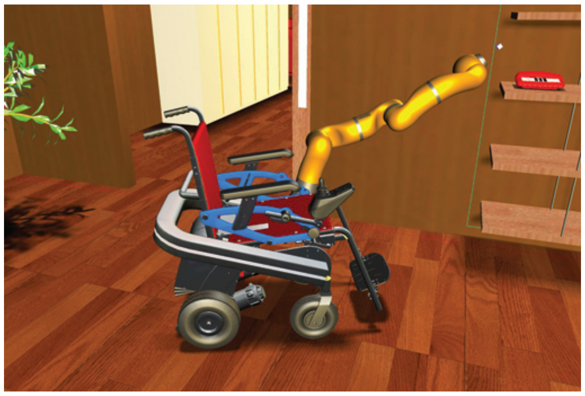
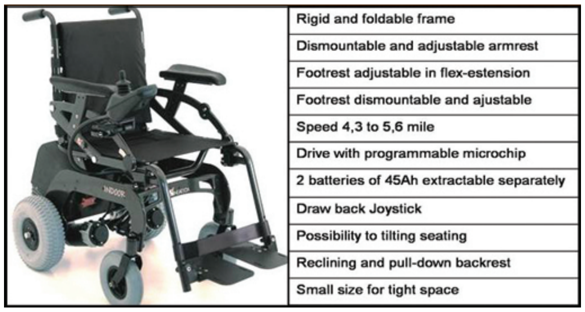


Figure 4.2: The virtual environment with a wheelchair-mounted manipulator on a sliding rail

In order to obtain a wider workspace for a robotic extender mounted on the wheelchair, a sliding rail has been considered around the powered wheelchair, with proper modeling of such a joint for exploiting it as an additional degree of freedom (DOF) available for robot control.

The manipulator can move around the wheelchair by sliding along the rail; the rail is able to rotate around an horizontal axis, providing a way to change its inclination, for adapting the workspace to the user's needs (e.g., better dexterity on the ground). Such characteristic widely increases the robot workspace.



Rigid and foldable frame
Dismountable and adjustable armrest
Footrest adjustable in flex-estension
Footrest dismountable and ajustable
Speed 4,3 to 5,6 mile
Drive with programmable microchip
2 batteries of 45Ah extractable separately
Draw back Joystick
Possibility to tilting seating
Reclining and pull-down backrest
Small size for tight space

Figure 4.3: The real Neatech Indoor wheelchair is lightweight and powered

Three different lightweight robot arms (see Fig. 4.4), have then been considered for integration with the wheelchair:

a. KUKA Light Weight Robot (LWR),

b. Amtec Ultra Light Weight Robot (ULWR),

c. Mitsubishi PA-10.

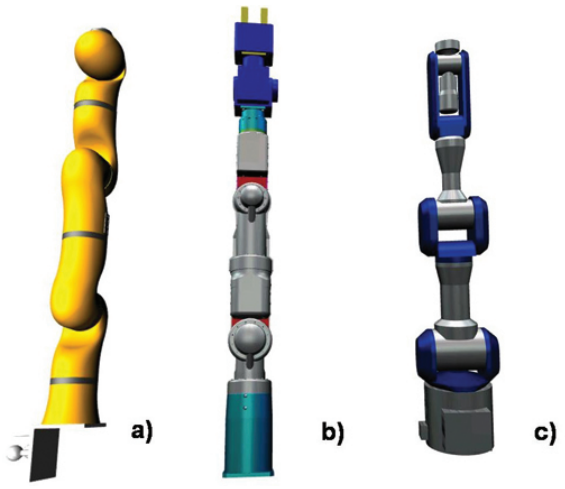


Figure 4.4: (a) KUKA LWR, (b) Amtec ULWR, (c) Mitsubishi PA-10

These manipulators have a kinematic structure similar to the human arm: moreover, the reduced weight allows using them for service robotics, while quantitative evaluation of intrinsic safety in case of rigid impacts are available only for the KUKA arm [ASH07].

Since the robotic arm and the rail mounted on the wheelchair introduce static balancing issues, it has been necessary to verify also the stability of the integrated system, which has been verified for all the three robots considering the wheelchair both with and without a person sitting on it.

4.4.2 Kinematic modeling

A key point for wheelchair-mounted manipulators is the possibility for the robot of moving around the sit, and extending this way its workspace around the user, without passing trough the front part of the wheelchair [BGH*06].

For the proposed application, the kinematic model of the considered manipulators has been extended to include the motion on the base joint.

With reference to figure 4.5, the base joint is modeled as a prismatic joint on the left, right and rear side of the wheelchair, while the sections between these segments are modeled as rotary joints. Smooth transitions between different segments have been considered.

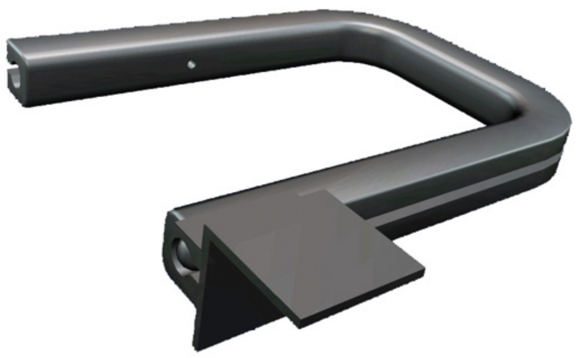


Figure 4.5: 3D model of the proposed rail.

In order to control the motion of different control point of the robotic systems, the approach described in [DASO*07] has been adopted. The user (or an automatic module for safety procedures) can control the position not only of the end-effector, but also of an arbitrary point on the articulated structure of the manipulator, moving on the robot.

For instance, a control point can be the point of the robot which is closest to a collision (monitored with exteroception), or a point (e.g., the "elbow") that it is wished to move away from its current position.

The control interface gives reference directions, which are interpreted as desired velocities, and a closed-loop inverse kinematics (CLIK) scheme is adopted for computing the reference joint values.

Some properties for the evaluation of the considered robots can be described via objective indicators such as the well-known manipulability measure [SS00], which gives an indication of the ability of the robot to change posture and, therefore, of its ability in manipulation from the current position (and orientation). It is the volume of the so-called *manipulability ellipsoid*, which gives a graphical interpretation of robot dexterity.

With a dynamic model of the arms, it is also possible to compute a dynamic version of such indicator and an additional safety measure, namely, the impact ellipsoid [Wal94].

4.5 Experimental setup

Virtual Reality technologies have been used in order to give the user the impression of moving a robotic arm attached to an ordinary powered wheelchair for physical disabled people. In particular, the case study refers to a powered wheelchair (*Indoor 2003* by Neatech srl) equipped with a *kuka* light-weight robot [DDM*08]. The main goal has been the development of a three-dimensional virtual environment in which the user was able to control a robot manipulator attached to a wheelchair, in 1:1 scale and from his own point of view.

The experimental activity has been mainly carried out at "VRoom", that is a low-cost VR laboratory equipped with two LCD projectors and polarized glasses for *passive* stereoscopic view [CDP06]. Further tests have been also carried out at *VRTest* lab, that is a high-end laboratory with three DLP projectors and shutter-glasses for *active* stereoscopic view [CD07b]. In order to enhance the impression of moving a real appendix of a wheelchair, a physical wheelchair has been placed in the laboratory in such a way that the user viewpoint coincided with the virtual wheelchair starting position. Moreover, the glasses are endowed with optical targets, and the user can also adjust the point of view on the virtual scene by moving his head. In this way, a semi-immersive VE has been set up, where the user can move and control both the wheelchair and the virtual robotic arm by means of different devices (Figure 4.6).

The first step in order to carry out the virtual simulations has been the design of the VE. The author has designed a "virtual flat" with all the common furnishing. In particular, it is completely unstructured with respect

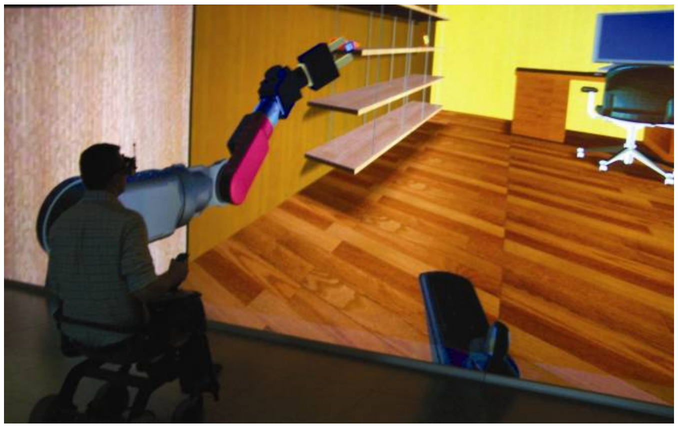


Figure 4.6: The semi-immersive set-up at VRTest

to the robotic manipulator (Figure 4.7).

The realism of the VE has been particularly considered, because a semi-immersive experimental set-up may create some problems in terms of sense of presence, especially with respect to experiments that involve both real input and virtual outputs. In fact, a low sense of presence of the user may undermine the validity of test results.

The second phase has concerned the programming of the virtual environment, that means, essentially, defining its behavior in response to the user's interaction. The software platform that have been used as Simulation Manager for this work is Virtual Design 2 (VD2), by vrcom GmbH.

In particular, the VE can be programmed with a complete set of commands that essentially describe actions that operate on the objects in the VE.

The software application [DMT07] that has been described in chapter 3 allows the user to move a kinematic chain in the virtual environment by means of a multidimensional input device, such as a *joystick* or a *space-mouse* (Figure 4.8).

The *space-mouse* is an input device with 6 Degrees of Freedom (DOF). It has a round "puck" or a "ball" that can be manipulated out of its quiescent position in order to apply rotations as well as translations.

The *joystick* is a very common input device, generally consisting of a stick that pivots on a base and reports its vectorial direction. Moreover a lever controls the "vertical elevation". Thus, the joystick is a 4-DOF input device.

Although the space-mouse and the joystick have different degrees of free-

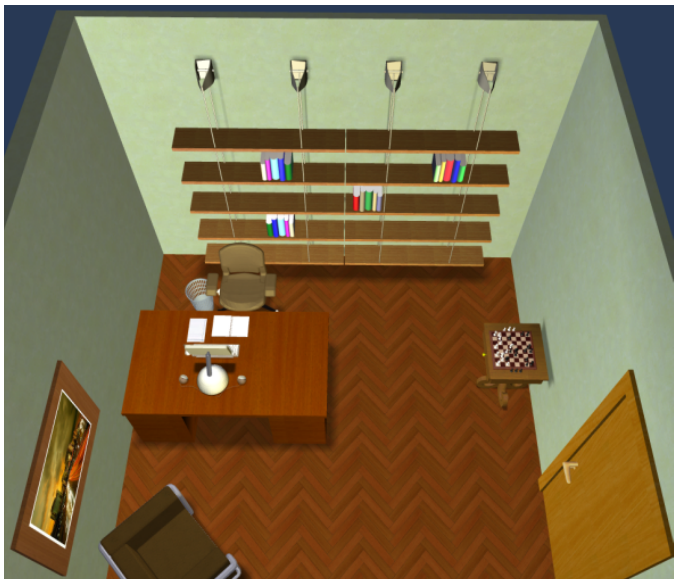


Figure 4.7: The "virtual flat" with all the common furnishing

dom, in this work only three DOF have been used, in order to control only the position of the end-effector, but not its orientation.

Consequently any orientation adjustments have to be done by operating in the joint space. For this reason, the user can individually control each joint angle by means of the joystick buttons.

This choice has been mainly due to the difficulty for training a disabled user in controlling both the position and the orientation of the end-effector at the same time with a 6-DOF control interface.

However, both the space-mouse and the joystick are equipped with several buttons that can be used to trigger user-defined actions. For instance, the user can control both the wheelchair and the robot with the same interface (e.g. the space-mouse). This is achieved by simply pressing a button, that switches the active control between the wheelchair and the robotic arm and vice versa. Moreover, some buttons of the joystick are related to predefined postures. This feature simplifies the handling of the kinematic chain, since generally it reduces the time needed to reach a desired posture. Moreover, the possibility to display the manipulability ellipsoid and the Jacobian matrix related to a certain posture is given (Fig. 4.10)

Finally, it is worth noticing that the user can even move the powered wheelchair in the virtual space with the joystick while he is controlling the

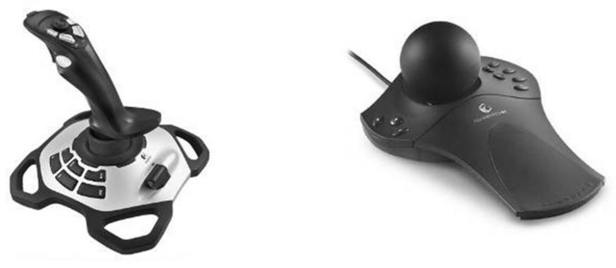


Figure 4.8: Input devices

robot with the space-mouse and that other kinds of input devices can be tested with minor modifications to the interface.

4.6 VR experiments

The second step of the experimental phase is the training of the users. The experimenter has to be briefly trained to the use of the interaction devices and also informed about the importance of their experience [DG07]. The training stage has been carried out in two phases: in a first phase, the operator has simply to move the end-effector along the three axes of the virtual space; then, in a second phase, the operator has to move the end-effector following a defined path. Only once the user is able to use properly the virtual devices, the experimental session can begin.

In order to compare the proposed robots for the considered application, an appropriate methodology of usability analysis has been developed.

The usability denotes the ease the user can handle the kinematic chain. It strictly depends on the robot type, the inverse kinematics algorithm, the user-robot interface, and the level of training of the user. In order to quantify the usability for a specific robot, we have defined a set of time-based experiments. For the considered tasks, the usability measure is based on the elapsed time for completing the required motion.

In particular, four tasks have been conceived:

1. book positioning on a shelf (reaching);
2. objects relocation on a shelf (dexterity);
3. moving a chess piece on a board (fine motion between obstacles);

4. moving objects between two different planes (free motion);

Book positioning on shelves

Twelve books of four different colors are positioned (s. Fig. 4.9) on four different shelves. The task consists of moving every book on the shelves of the same color. This test allows determining the feasibility of operations performed by disabled people and also to measure the manipulability and the usability of the robot with respect to different levels of height for the end-effector.

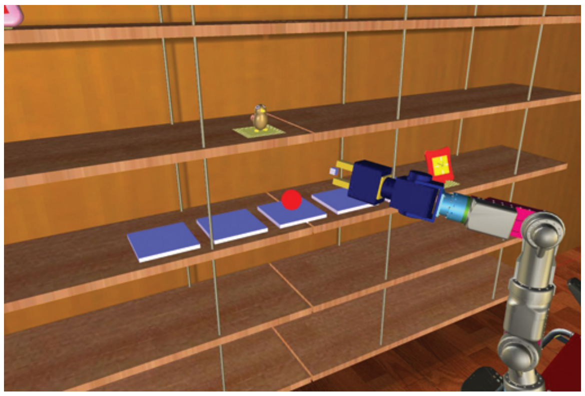


Figure 4.9: Execution of the object relocation task on a bookshelf

Objects relocation on a shelf

The user is 0.4 m far from a shelf of height 1 m, where five equidistant markers are set. The distance among the most external markers is set to 1 m. It deals with the normal dimension of the human arm workspace. A spherical object is set on the central marker. The user first has to grab the object from the central marker and then release it on the others. The test evaluates the horizontal usability of the robot.

Moving a chess piece on a board

The user is 0.380 m far from the chessboard (measured from the chest to the center of the board). The violet horse (Fig. 4.10) is the piece to be moved, the yellow box represents the starting position of the horse and blue boxes highlight the final destination of the piece. The user has to move the piece

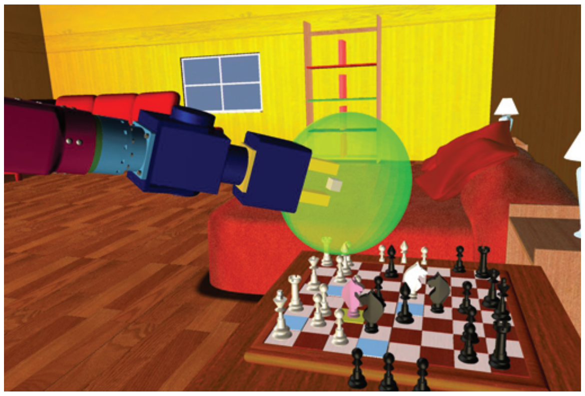


Figure 4.10: Execution of the task at a chessboard: the manipulability ellipsoid is displayed on the screen

from the yellow box to the blue one. This test evaluates the ability to handle small objects among some obstacles.

Moving objects between two different planes

In this test, the user has to move some glasses from a table to another and back again. The two tables are 0.9 m far. The starting position and the final destination of the glasses are signaled by different colored markers.

4.7 Experimental results

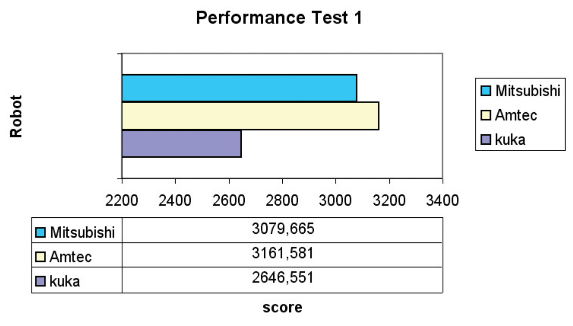
Experimental sessions have been performed by able-bodied users. We have considered a random sample of 10 users and we have asked them to carry out the four tasks using the three manipulators in a random order. The task execution time t has been considered as a performance index. In order to take in account both the mean μ and the standard deviation σ of the measured times, the *score* E for each test has been calculated in the following manner:

$$E = (\mu^2 + \sigma^2) \tag{4.1}$$

With this choice, lower scores mean better performances.

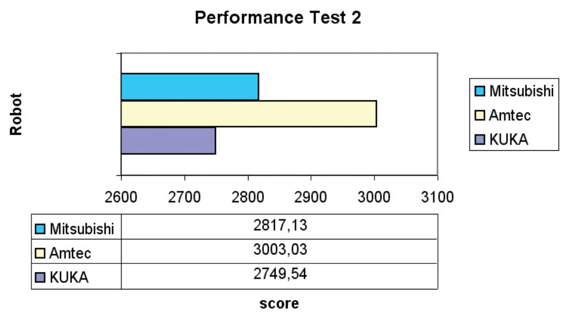
The performance of the manipulators is summarized in figures 4.11-4.14.

In conclusion, the KUKA LWR has obtained the best score in terms of usability in each test.



▨ Mitsubishi	3079,665
▨ Amtec	3161,581
▨ kuka	2646,551

Figure 4.11: Test 1 - Book positioning on shelves

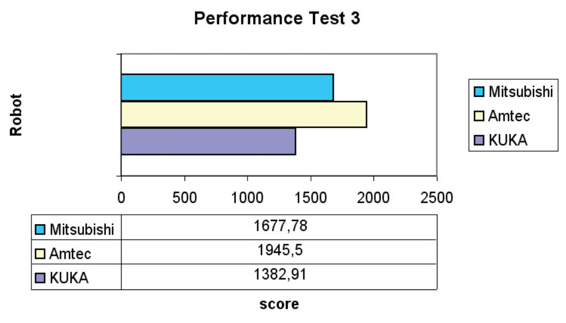


▨ Mitsubishi	2817,13
▨ Amtec	3003,03
▨ KUKA	2749,54

Figure 4.12: Test 2 - Objects relocation on shelf

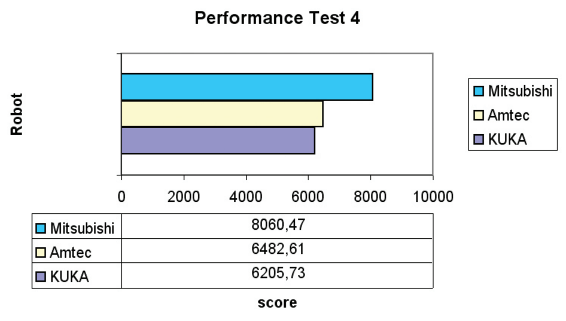
These preliminary results suggest the potential impact of the proposed tool for the evaluation of robotic extenders.

Different indicators can be considered as well, depending on the application: the possibility of carrying heavy objects, for instance, is considered important by the participants, for autonomy in their houses, while the central requirement of safety is considered somehow less central, due to the will of taking the control of the robot without autonomous robot behaviors. It is worth noticing that some weights chosen by the users can be quite surprising for an engineer. This shows that the appearance of the robot has a strong impact on the user, and even the intrinsic safety and versatility can be appreciated not enough, if not accompanied by a proper design.



Mitsubishi	1677,78
Amtec	1945,5
KUKA	1382,91

Figure 4.13: Test 3 - Moving a chess piece on a board



Mitsubishi	8060,47
Amtec	6482,61
KUKA	6205,73

Figure 4.14: Test 4 - Moving objects between two different planes

4.8 Conclusions and future work

The proposed simulation environments can allow comparison between trajectory planning schemes, kinematic optimization of robots for wheelchairs, appearance and reliability of wheelchair-mounted manipulators.

However, the described methodology can be improved by increasing the number of evaluation criteria and the sample of users considered.

Furthermore, with the addition of simplified dynamic models, joint torques due to the interaction with the environment can be generated as well, for a better modeling of the environment.

The use of robots in unstructured and time-varying environments implies the need for implementing real-time reactive strategies to cope with possible collisions [DASO*07], which are going to be implemented for completing the simulator. The proposed tool can be used also for evaluating in a very realistic way the reactions during the approach and the motion of the robot on desired or unexpected trajectories. Force feedback can be added via a proper *haptic interface* [Bur96].

Finally, tests with physically disabled people could provide additional insights in the cognitive and ethical aspects related to the introduction of robotic extenders in everyday life.

The study on the interface should take into account the possible difficulty for a disabled user in controlling both the position and the orientation of the end-effector at the same time.

Virtual reality provides a time- and cost-effective tool for the proposed comparisons.

Chapter 5

Modeling and Control of humanoid robots

5.1 Introduction

This chapter describes an approach to the modeling and control of a humanoid robot with high degree of redundancy. The increasing attention of robotics community about the so-called *humanoid robotics*, [SKSH03], [KNK*02] [OEF*06], is not simply related to the ancestral ambition of *building something that looks like a man*, but has also an immediate and objective reason. In particular, it arises from the apparently obvious fact that all actions that human beings perform daily, the objects that they manipulate and the environments in which they live have all been made "on a human scale". For instance, all the objects we manipulate have been made in order to best fit the shape of our hands and are light enough to be easily handled by one person. This means that if we really want to build machines able to cooperate with human beings, we need to design robots that not only can move through environments designed for humans, but can also handle objects particularly suited to our physical structure and our behavior. For instance, bipedal robots could potentially move in the same space where people work, such as an industrial plant with stairs and handrails. In this way, these robots could cooperate with us and even collaborate with each other using our own tools or machinery. Further considerations can be made even under the aspect of human-robot and robot-robot communication. In fact, humanoid robots could potentially use already existing human communication channels [SK] and even be used in the therapy of some forms of mental disorders [RDBB05]. Therefore, it makes sense to develop efficient models that allow us to accurately control the motion of humanoid robots. On the other hand, it is also necessary to develop simulation tools to study robots behavior in *unstructured* environments, considering the safety issues arising from the interaction with humans.

Moreover, the model of humanoid robot that will be described in the following sections, is also suitable for simulating the behavior of human beings in a virtual environment. This can be very useful for ergonomics analyses or even for manual processes simulations.

Manufacturing companies, indeed, have now taken the concept of "man adaptability" as a basic parameter of quality for their products, therefore they are giving an increasingly attention to ergonomic analyses, even from the early stages of design, [CD07a]. The so-called *virtual manikins*, provided by many process simulation software, essentially are virtual kinematic chains consisting of several segments and joints. The length of their segments are derived from anthropometric databases, which can be queried with respect to different percentiles in the population. However, these software tools generally are too much complicated to be handled and the so-called "process simulation" often becomes a very time-consuming task, mainly because of the difficulty in controlling the kinematic chain of the virtual humanoid.

Therefore, developing efficient algorithms aimed at controlling a high-articulated chain, such as the human mechanical structure, is very interesting even for fields apparently unrelated to robotics.

5.2 Kinematic modeling

The basic idea was to control the highly redundant kinematic structure of a humanoid robot, by means of only few control points. This idea is supported by the observation that generally, during the carrying out of a certain task, the postures taken by our kinematic structure largely depend on balancing and mechanics issues, that are only partially related to the considered task. In other words, a human being can play the piano without concerns about the position of his center of mass.

For this purpose, the goal has been to develop an algorithm that allowed us to concentrate only on the task-related control points, without the need of specifying the other redundant Degrees of Mobility (DoMs) of the chain.

In particular, the position of the Center of Mass (CoM) of the humanoid has been taken into account, in order that it was always consistent with the balancing issues of its mechanical structure.

5.2.1 Hierarchical model of humanoid robot

Firstly, in order to take advantage of the systematic approaches typical of serial robots, the humanoid has been modeled as the combination of four kinematic chains, which share the same starting point, called *root*. The resulting model is the hierarchical structure shown in figure 5.1.

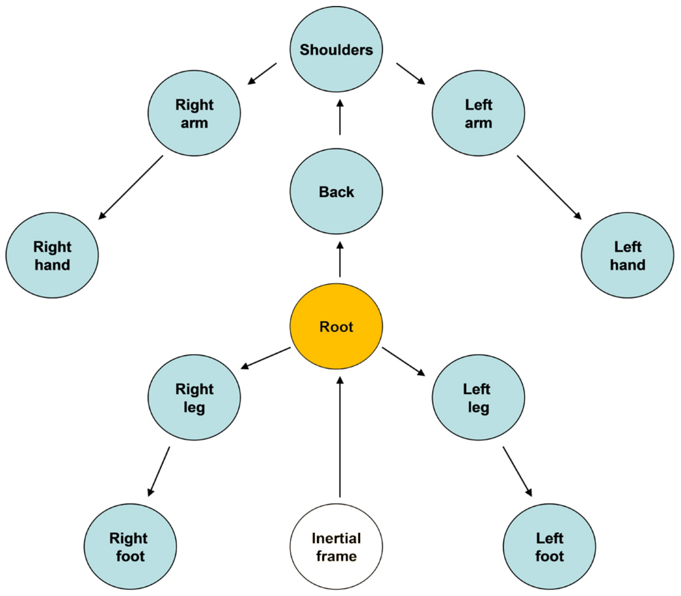


Figure 5.1: Hierarchical model of humanoid robot

Starting from this graph, it is possible to build up the Denavit-Hartenberg (D-H) model of the whole kinematic chain (Figure 5.2).

In particular, we can define a number of direct kinematics equations, with respect to the *root* reference frame. As we can see, the position and orientation of the *root* node with respect to reference frame is specified by introducing 6 *virtual joints* (see section 5.2.2). Thus, the considered kinematic structure has 39 Degrees of Freedom in all.

This kind of modeling has the advantage of simplicity, but generally it may cause a physical consistency problem, since some links (as well as all the virtual links) are shared among different kinematic chains. For instance the “back” of the virtual humanoid is shared between its right and left arms. This issue and its solution is discussed in section 5.2.5.

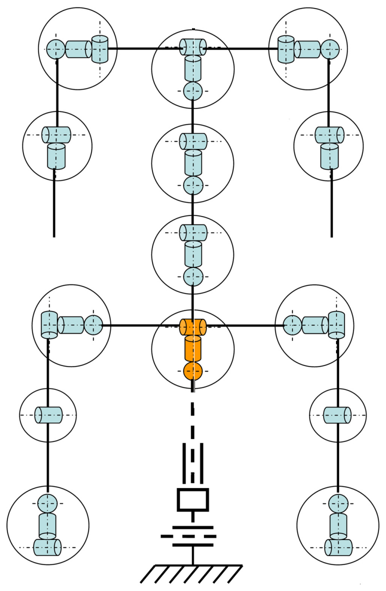


Figure 5.2: D-H model of humanoid robot

5.2.2 Virtual joints

Now, we must describe the position and orientation of the multi-legged kinematic chain with respect to an inertial frame. For industrial robots identifying such a frame is intuitive, because they have a fixed base. A humanoid robot, instead is bound to the ground by a one-way constraint, that is the current support plane, for instance one foot.

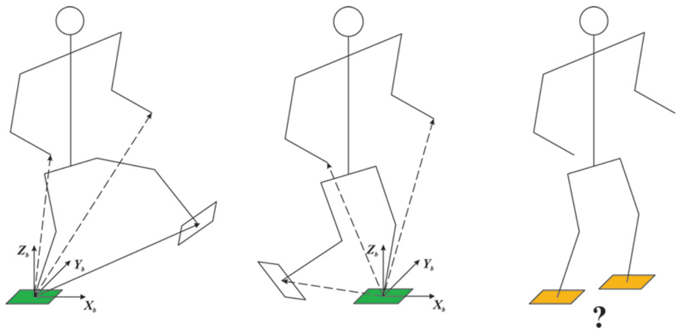


Figure 5.3: Which reference frame?

But this reference periodically changes during the walk, thus we apparently cannot identify a fixed base starting from which the Denavit-Hartenberg method can be applied (Figure 5.3).

Moreover, the presence of multiple end-effectors (two hands and two feet) implies the need to describe the position and orientation of many frames, differently from industrial robots, in which the kinematic chain has only one end-effector. This problem was overcome using the *virtual joints* approach [Yam04]. Namely, when describing the position and orientation of a robot arm, it was conceived connected to the ground plane through a virtual manipulator consisting of three prismatic and three revolute joints, which characterize its position and orientation. The attaching point has been called *root* (Figure 5.4).

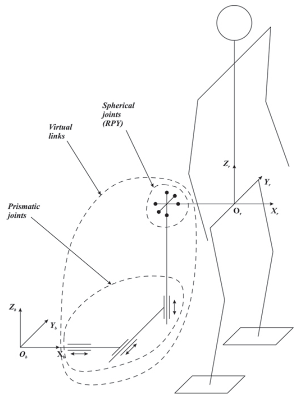


Figure 5.4: Virtual joints approach

With this approach, the hands and the feet (and even any other control point) simply are end-effectors that can be controlled with velocity references. In other words, the posture of the virtual humanoid is completely specified by the following parameters vector:

$$\mathbf{q} = \left[\mathbf{q_r}^T \;\; q_1 \; q_2 \; .. \; q_n\right]^T$$

where $\mathbf{q_r} = \left[\mathbf{p_r^{oT}} \; \omega_\mathbf{r}^\mathbf{oT}\right]^\mathbf{T}$ identifies the *root* frame.

Moreover, virtual joints technique makes unnecessary the management of closed kinematic chains during the phase of double support. Indeed, this condition becomes equivalent, from a kinematic point of view, merely to impose a null velocity reference to the feet.

5.2.3 Augmented Jacobian

Each chain has its own direct kinematic function, therefore a Jacobian matrix can be computed for a generic control point of the structure. Generally, considering n control points we can define the following set of equations:

$$\begin{aligned} \mathbf{v_1} &= \mathbf{J_1}\dot{\mathbf{q}} \\ \mathbf{v_2} &= \mathbf{J_2}\dot{\mathbf{q}} \\ &\vdots \\ \mathbf{v_n} &= \mathbf{J_n}\dot{\mathbf{q}} \end{aligned} \tag{5.1}$$

where the generic element $\mathbf{J_i}$ is the Jacobian matrix related to a specific control point.

It is understood that, if the generic joint variable q_j does not affect $\mathbf{v_n}$, it is $(J_n)_{ij} = 0$. This set of equations can be summarized as:

$$\mathbf{v} = \mathbf{J_{AU}}\,\dot{\mathbf{q}} \tag{5.2}$$

where $\mathbf{J_{AU}}$ is the so-called *Augmented Jacobian*. On one hand, this approach allows us to solve the inverse kinematic problem with only one CLIK algorithm [SK]. On the other hand, the trajectories defined for the control points will be all treated as *primary tasks*, unlike other solution methods do, such as null space based approaches [SS91], [Nak91].

In particular, in order to define the structure of $\mathbf{J_{AU}}$, the vector $\dot{\mathbf{q}}$ must be properly sorted. Since humanoid structure is composed by four kinematic chains, we can write four different vectors of unknowns:

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \begin{bmatrix}\dot{\mathbf{q}}_r^T & \dot{\mathbf{q}}_{rl}^T\end{bmatrix}^T && \textit{right leg} \\ \dot{\mathbf{q}}_2 &= \begin{bmatrix}\dot{\mathbf{q}}_r^T & \dot{\mathbf{q}}_{ll}^T\end{bmatrix}^T && \textit{left leg} \\ \dot{\mathbf{q}}_3 &= \begin{bmatrix}\dot{\mathbf{q}}_r^T & \dot{\mathbf{q}}_b^T & \dot{\mathbf{q}}_{ra}^T\end{bmatrix}^T && \textit{right arm} \\ \dot{\mathbf{q}}_4 &= \begin{bmatrix}\dot{\mathbf{q}}_r^T & \dot{\mathbf{q}}_b^T & \dot{\mathbf{q}}_{la}^T\end{bmatrix}^T && \textit{left arm} \end{aligned}$$

where $\dot{\mathbf{q}}_r$ are the velocities of the virtual joints that are shared among four kinematics chains. These vectors can be summarized in only one vector of unknowns:

$$\begin{aligned} \dot{\mathbf{q}} &= \begin{bmatrix}\dot{\mathbf{q}}_r^T & \dot{\mathbf{q}}_{rl}^T & \dot{\mathbf{q}}_{ll}^T & \dot{\mathbf{q}}_b^T & \dot{\mathbf{q}}_{ra}^T & \dot{\mathbf{q}}_{la}^T\end{bmatrix}^T \\ &= [\dot{q}_1 \;\; \dot{q}_2 \;\; .. \;\; \dot{q}_{39}]^T \end{aligned} \tag{5.3}$$

With this choice, the Augmented Jacobian takes the following form:

$$\mathbf{J_{AU}} = \begin{bmatrix} \mathbf{J_{r_i}} & \mathbf{J_{rl_i}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J_{r_i}} & \mathbf{0} & \mathbf{J_{ll_i}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{J_{r_i}} & \mathbf{0} & \mathbf{0} & \mathbf{J_{b_i}} & \mathbf{0} & \mathbf{0} \\ \mathbf{J_{r_i}} & \mathbf{0} & \mathbf{0} & \mathbf{J_{b_i}} & \mathbf{J_{ra_i}} & \mathbf{0} \\ \mathbf{J_{r_i}} & \mathbf{0} & \mathbf{0} & \mathbf{J_{b_i}} & \mathbf{0} & \mathbf{J_{la_i}} \end{bmatrix} \tag{5.4}$$

The matrix $\mathbf{J_{AU}}$ has 39 columns, while the number of its rows depends on the number of control points considered.

5.2.4 Center-of-Mass Jacobian

Unlike industrial manipulators and, more generally, not-ambulatory robots, bipedal robots must concern about their balance while performing any task. If this does not happen, obviously, the robot would lean over and fall. Moreover, humanoid robots generally have a number of joints, and a consequent degree of redundancy, much higher than those of a traditional industrial robots. Consequently, there are many postures that achieve the same position for its body terminal. Also, taking into account the balancing issues allows the humanoid to take more natural posture, similar to those of human beings.

For this, the Virtual End-Effectors (VEEs) technique [DPS06] has been implemented also with respect to the center of mass (CoM) of the digital humanoid, which becomes a further control point for the kinematic chain. In particular, the trajectory of the CoM can be defined in such a way that its vertical projection on the current support plane (namely, the Center of Pressure, CoP) belongs to the stability polygon formed by the feet (Figure 5.5). It is worth noticing that the constraint about the CoP will be treated as a primary task, as well as the other tasks.

The basic idea is to obtain a differential relationship like this:

$$\mathbf{v_G} = \mathbf{J_G}\dot{\mathbf{q}} \tag{5.5}$$

where $\mathbf{J_G}$ is a $3 \times n$ matrix, called Center-of-Mass Jacobian. Then, the equation (5.5) will be inserted in the equation (5.1) as a further control point.

For this purpose, we can define the CoM of a kinematic chain composed of n links, as:

$$\mathbf{p}_G = \frac{\sum_{i=1}^{n} m_i \mathbf{p}_{G_i}}{\sum_{i=1}^{n} m_i} = \frac{1}{m} \sum_{i=1}^{n} m_i \mathbf{p}_{G_i} \tag{5.6}$$

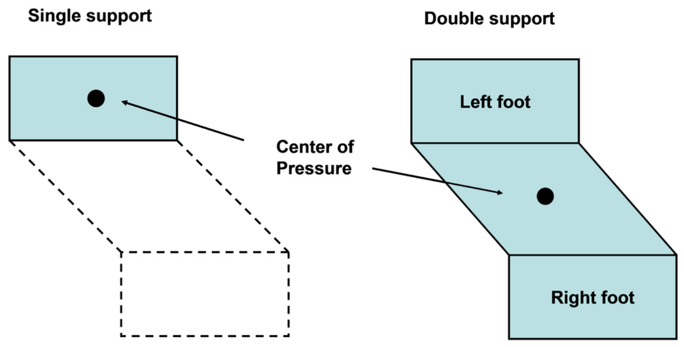


Figure 5.5: Center of Pressure and support plane

The Equation (5.6) can be derived with respect to time:

$$\mathbf{v_G} = \frac{1}{m}\sum_{i=1}^{n} m_i \mathbf{v}_{G_i} \tag{5.7}$$

Since the center of mass of each link can be considered as a *Virtual End-Effector* (VEE), it is always possible to write a differential relationship like this

$$\mathbf{v_{G_i}} = \mathbf{J_{G_i}}\dot{\mathbf{q}}$$

where:

$$\mathbf{J_{G_i}} = \begin{bmatrix} \gamma_{x,1} & \cdots & \gamma_{x,i} & 0 & \cdots & 0 \\ \gamma_{y,1} & \cdots & \gamma_{y,i} & 0 & \cdots & 0 \\ \gamma_{z,1} & \cdots & \gamma_{z,i} & 0 & \cdots & 0 \end{bmatrix} \tag{5.8}$$

Indeed, if the vector $\dot{\mathbf{q}}$ has been properly sorted, $\mathbf{v_{G_i}}$ can be affected at most by the first i links of the chain. Now, the equation (5.7) can be written as:

$$\mathbf{v_G} = \left[\frac{1}{m}\sum_{i=1}^{n} m_i \mathbf{J_{G_i}}\right] \dot{\mathbf{q}} \tag{5.9}$$

By comparing the equations (5.5) and (5.9), we can finally assume:

$$\mathbf{J_G} = \frac{1}{m}\sum_{i=1}^{n} m_i \mathbf{J_{G_i}} \tag{5.10}$$

Given $\mathbf{J_G}$, the velocity of CoM $\mathbf{v_G}$ becomes a further control point for the kinematic chain. Thus, we can insert the kinematic relation (5.5) in the equations set (5.1). As a result, we will have an Augmented Jacobian matrix with two more rows, that are related to the components of $\mathbf{v_G}$ projected on the current support plane. As aforementioned, the implemented inversion algorithm assures that a constraint on CoM velocity becomes a primary task to be achieved.

Finally, it is worth emphasizing that the expression of $\mathbf{J_G}$ suggests also the possibility to use the *static-kinematic duality* [SS00] to compute the balancing torques that support the weight of the kinematic structure of the humanoid robot, avoiding the computation of its inertia matrix.

5.2.5 Conflicting tasks

As aforementioned, some tracts of the humanoid structure are shared among apparently different kinematics chains. For instance, the right and left arms of the humanoid share a common tract, namely the back. But, if actually left and right arms was modeled as *independent* chains, they could perform different or even conflicting tasks.

For this, inversion algorithms for multi-legged robots generally provide two different solutions for the left and right arm. In particular, for the back it will be:

$$\mathbf{\dot{q}_{bl}} \neq \mathbf{\dot{q}_{br}} \tag{5.11}$$

where $\mathbf{\dot{q_{br}}}$ and $\mathbf{\dot{q_{bl}}}$ are different solutions obtained considering the back belonging respectively to right and to left arm. However, generally this issue is commonly solved with the following choice for the joints of the back:

$$\mathbf{\dot{q}_{b}} = \mathbf{\frac{1}{2}} \left(\mathbf{\dot{q}_{br}} + \mathbf{\dot{q}_{br}}\right) \tag{5.12}$$

This guarantees a physical consistent solution, but in general none of the conflicting tasks will be actually achieved.

The inversion algorithm based on the Augmented Jacobian cleverly resolves also this issue. Indeed, the vector of solution $\mathbf{\dot{q}}$ has been sorted in such a way that its elements appear just one time, thus the inversion algorithm provides only one solution that is consistent with all the physical constraints.

On the other side, the main problem related to the application of Augmented Jacobian method is the matrix inversion, due to its dimensions ($\mathbf{J_{AU}}$ has 39 columns) and consequently to the detecting of its singularities.

5.3 Simulations in VR

In order to test the proposed inversion model, several VR simulations have been carried out. First of all, a geometric model for the virtual humanoid has been built up with the hierarchical approach already discussed in section 3.2. The result has been the VRML model shown in figure 5.6.

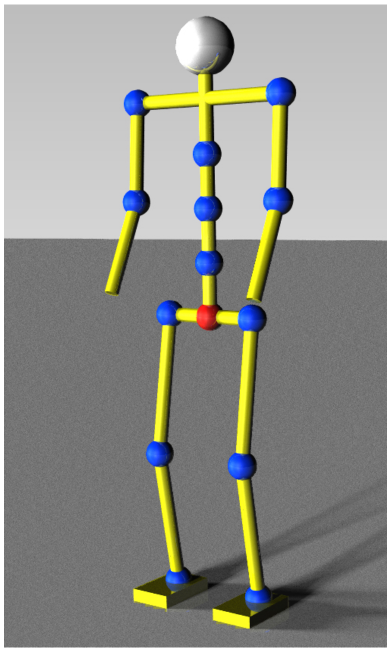


Figure 5.6: VRML model of Virtual humanoid

After that, the kinematic model of the virtual humanoid and its inversion algorithm has been implemented in *RoboTiX* module (see chapter 3). Since a weighted pseudo-inverse has been adopted to compute the inverse kinematic, a proper choice of weights and of some optimization criteria have granted quite natural and fluid movements for the virtual humanoid. As a result, despite of the ease of planning the movements of the digital humanoid, we can simulate quite complex tasks, by planning the trajectory for only a limited number of control points. For instance, the virtual humanoid can walk or even climb a ladder, as will be shown in the following sections.

5.3.1 Standing-up from a sitting position

In figure 5.7 different frames of a standing-up simulation are shown. This task has been achieved just by imposing a null velocity to the feet of the virtual humanoid and by giving a vertical velocity reference to its pelvis. As a further constraint, the CoP must always belong to the support plane (*balance control*). As shown, the virtual humanoid performs the assigned movement always keeping itself in balance (Figure 5.7).

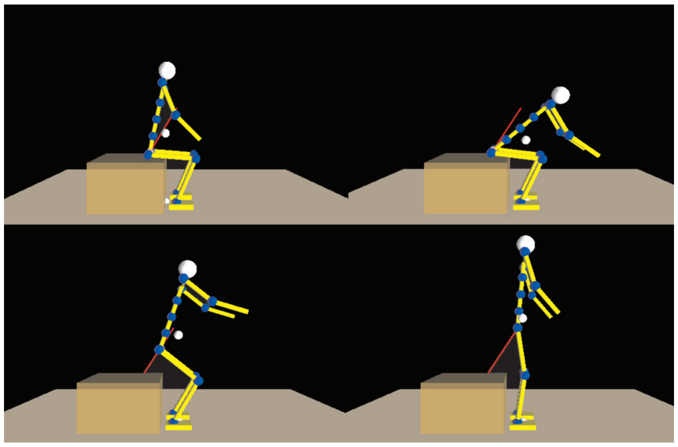


Figure 5.7: Standing up from a sitting position

In a similar way, it is possible to simulate the virtual humanoid sitting down from a standing position.

5.3.2 Collision avoidance

The VEE approach can be used to take into account also eventual obstacles in the humanoid workspace. In figure 5.8 is shown again the simulation of a standing up, but this time there is a table. This task has been achieved by assigning to the control points velocity references coming from repulsive potential fields.

5.3.3 More complex tasks

In this section the results of the simulation of quite complex tasks are reported. The inversion algorithm has always taken into account the constraints about the CoP, as aforementioned.

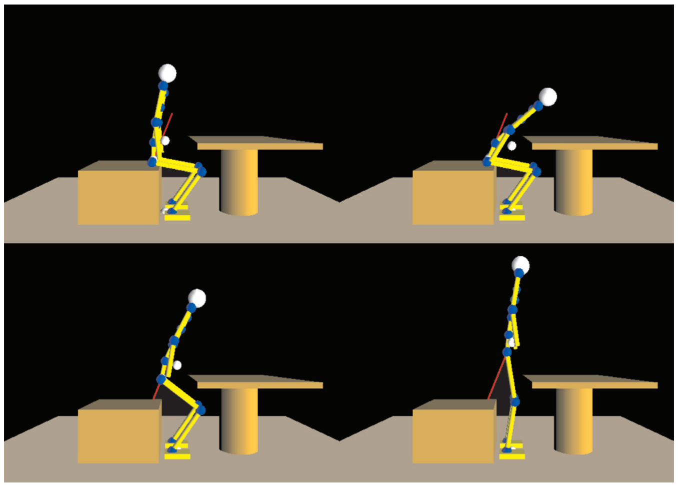


Figure 5.8: Standing up from sitting position near a table

The results shown in the following figures can be quite interesting in the field of both computer graphics and robotics.

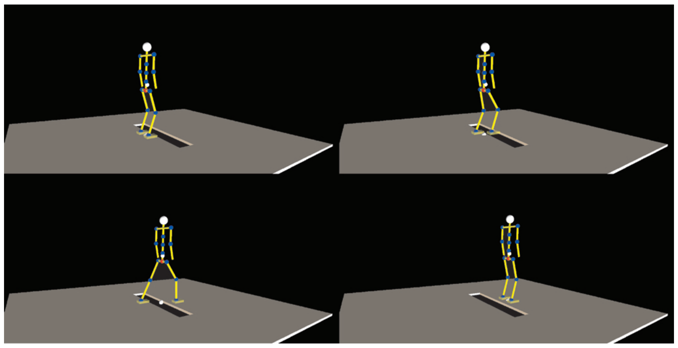


Figure 5.9: The virtual humanoid avoids a hole while walking

5.4 Conclusions and future work

The main novelty of the proposed approach has been the computation of the Jacobian matrix with respect to the center of mass of the kinematic structure. In fact, the definition of the movements of the center of pressure as a primary task has granted natural movements to the virtual humanoid, in spite of the limited number of control points considered.

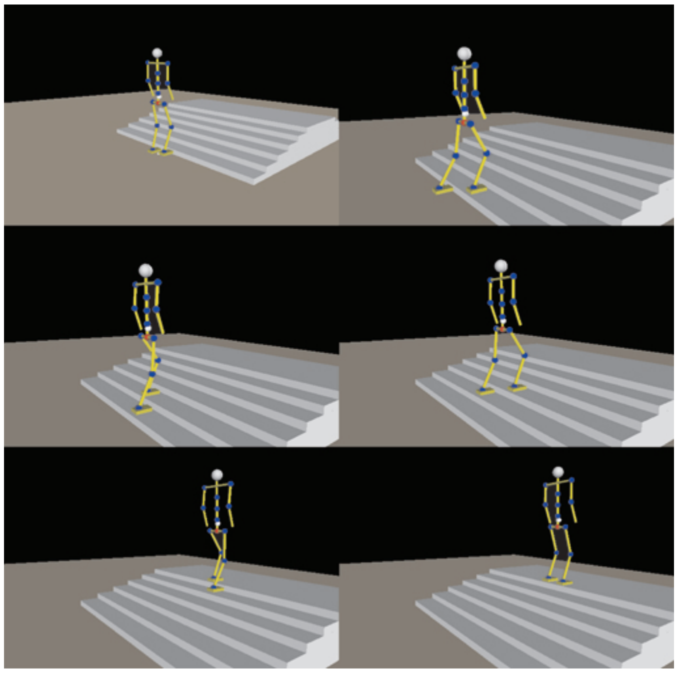


Figure 5.10: The virtual humanoid ascends the stairs

Moreover, the developed model lends itself to a very different set of applications, even not strictly robotic. Firstly, it can be used for digital animation of virtual humanoids in the field of ergonomics and process analyses. In fact, despite of the complexity and the cost of already existing software tools dedicated to this type of analysis, generally their simulation algorithms are still tied to "key-frame" animation techniques. The developed model instead makes it possible to define the kinematics simply by planning the trajectory of a limited number of control points.

Another field of application could be marker-based motion capture [MTT98], where the algorithm can be used in order to limit the number of markers needed to capture the human movements.

Finally, although the described model is advanced in terms of quality of analysis, it is also computationally efficient. Specifically, a symbolic representation for the kinematics of the digital humanoid has been derived. In this way, we can change in real-time several characteristic parameters of the chain, such as the applied loads, without further computational overload.

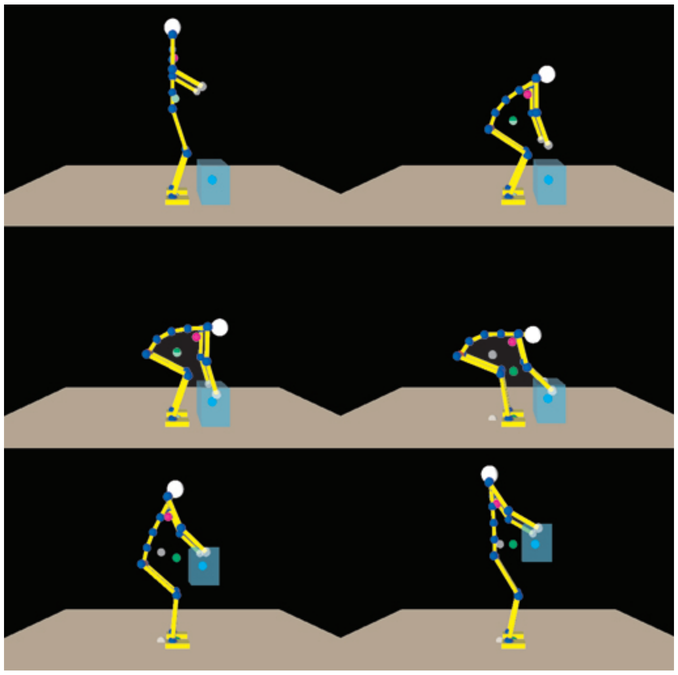


Figure 5.11: The humanoid lifts a weight.

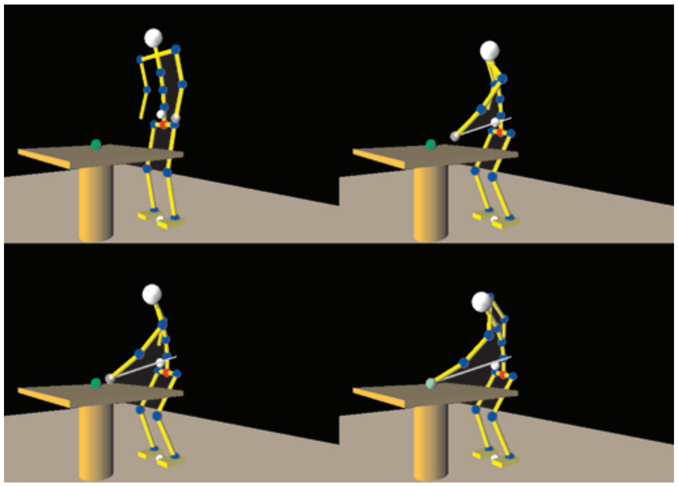


Figure 5.12: The virtual humanoid grabbing an object on the table.

Bibliography

[AAB*06] Alami R., Albu-Schaeffer A., Bicchi A., Bischoff R., Chatila R., De Luca A., De Santis A., Giralt G., Guiochet J., Hirzinger G., Ingrand F., Lippiello V., Mattone R., Powell D., Sen S., Siciliano B., Tonietti G., Villani L.: Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *IROS 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. Workshop on Physical Human-Robot Interaction in Anthropic Domains, Beijing (Chine)* (Oct. 2006). http://www.laas.fr/ felix/publis/pdf/iros06ws.pdf.

[AB98] Abshire K., Barron M.: Virtual mainteinance: Real world application within virtual environments. In *Proc. of Realiability and Maintainability Symposium, Ohio* (1998).

[AMED05] Alqasemi R., McCaffrey E., Edwards K., Dubey R.: Wheelchair-mounted robotic arms: Analysis, evaluation and development. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, CA, USA* (2005).

[ASH07] Albu-Schäffer A., Hirzinger G.: Dummy crashtests for evaluation fo rigid human-robot impacts. In *IARP International Workshop on Technical Challenges for Dependable Robots in Human environments, Rome* (2007).

[BC99] Burdea G., Coiffet P.: *Virtual Reality and Robotics.* Wiley–Interscience, 1999.

[BC03] Burdea G., Coiffet P.: *Virtual Reality Technology.* Wiley–Interscience, 2003.

[BGH*06] Balaguer C., Gimenez A., Huete A. J., Sabatini A., Topping M., Bolmsjö G.: The mats robot. service climbing robot for personal assistance. In *IEEE Robotics and Automation Magazine* (March 2006).

[BMPR02] Bruno F., Muzzupappa M., Postorino P., Rizzuti S.: Veis: Virtual environment for interactive sketching. In *First Annual Conference of Eurographics Italian Chapter* (Milano, Italy, July 2002).

[Bur96] Burdea G.: *Force and Touch Feedback for Virtual Reality.* Wiley–Interscience, 1996.

[Bur99] Burdea G. C.: The synergy between virtual reality and robotics. *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION 15*, 3 (1999), 400–410.

[CC00] Chen D., Cheng F.: Integration of product and process development using rapid prototyping and work cell simulation technologies. *Journal of Industrial Technology 16*, 1 (2000), 2–5.

[CD07a] Caputo F., Di Gironimo G.: *La realta virtuale nella progettazione industriale.* Aracne, Rome, 2007.

[CD07b] Caputo F., Di Gironimo G.: *VRTest: a Virtual Reality sysTEm for tranSporTation design.* Aracne, Roma, Italy, 2007.

[CDM06a] Caputo F., Di Gironimo G., Marzano A.: Approach to simulate manufacturing systems in virtual environment. In *Proc. of the XVIII Congreso International de Ingenieriï¿½a Grï¿½fica* (May 2006).

[CDM06b] Caputo F., Di Gironimo G., Marzano A.: Ergonomic optimization of a manufacturing system work cell in a virtual environment. In *Proc. of 5th International Conference on Advanced Engineering Design* (June 2006). Selected paper for the Acta Polytechnica Journal, forthcoming.

[CDP06] Caputo F., Di Gironimo G., Papa S.: A virtual reality system for ergonomics and usability validation of equipment controls. *ANALES DE INGENIERÌA GRÀFICA 18* (Sept. 2006), 47–64.

[COX72] COX M. G.: The Numerical Evaluation of B-Splines. *IMA J Appl Math 10*, 2 (1972), 134–149.

[Cra97] Craig J.: Simulation-based robot cell design in adeptrapid. In *Proceeding of the 1997 IEEE International Conference on Robotics and Automation, ICRA, Albuquerue* (Apr. 1997), vol. 4, pp. 3214–3219.

[Cra03] Craig J. J.: *Introduction to Robotics: Mechanics and Control*. Prentice Hall, 2003.

[DASO*07] De Santis A., Albu-Schäffer A., Ott C., Siciliano B., Hirzinger G.: The skeleton algorithm for self-collision avoidance of a humanoid manipulators. In *IEEE-ASME International Conference on Advanced Intelligent Mechatronics, Zürich* (2007).

[Dav04] Davies R. C.: Adapting virtual reality for the participatory design of workenvironments. *Comput. Supported Coop. Work 13*, 1 (2004), 1–33.

[DDM06] De Amicis R., Di Gironimo G., Marzano A.: Design of a virtual reality architecture for robotic work cells simulation. In *Proceeding of Virtual Concept 2006, Playa del Carmen, Mexico* (Nov. 2006).

[DDM*08] De Santis A., Di Gironimo G., Marzano A., Siciliano B., Tarallo A.: A virtual-reality-based evaluation environment for wheelchair-mounted manipulators. In *Proceedings of the 6th EUROGRAPHICS Italian Chapter 2008 conference* (Salerno, Italy, 2008), pp. 1–8.

[Dee95] Deering M. F.: Holosketch: a virtual reality sketching/animation tool. *ACM Trans. Comput.-Hum. Interact. 2*, 3 (1995), 220–238.

[DG07] Di Gironimo G., Guida M.: Developing a virtual training system in aeronautical industry. In *5th EUROGRAPHICS Italian Chapter Conference, Trento* (2007).

[DMP06] Di Gironimo G., Marzano A., Papa S.: Design of a virtual reality environment for maintainability tests and manufacturing systems simulations. In *Proceeding of International Conference CIRP-ICME 2006, Ischia, Italy* (July 2006).

[DMT07] Di Gironimo G., Marzano A., Tarallo A.: Human robot interaction in virtual reality. In *5th EUROGRAPHICS Italian Chapter Conference* (Trento, Italy, 2007), pp. 1–8.

[DPS06] De Santis A., Pierro P., Siciliano B.: The virtual end-effectors approach for human-robot interaction. *Advances in Robot Kinematics* (2006), 133–144.

[DPT07] Di Gironimo G., Papa S., Tarallo A.: Design review of train concepts in virtual reality. In *Proceedings of International Conference on Intelligent Processing and Manufacturing Materials* (Salerno, Italy, June 2007).

[DPT09] Di Gironimo G., Patalano S., Tarallo A.: Innovative assembly process for modular train and feasibility analysis in virtual environment. *International Journal on Interactive Design and Manufacturing IJIDeM 3*, 2 (May 2009).

[Dre06] Drepper U.: *How To Write Shared Libraries*. Red Hat, Inc., Aug. 2006. http://people.redhat.com/drepper/dsohowto.pdf.

[DT08] Di Gironimo G., Tarallo A.: Virtual design review of the alenia C27J cockpit. In *Farnborough International AIRSHOW (FIA) 2008* (Farnborough, Hampshire, July 2008).

[DW92] Deo A., Walker I.: Robot subtask performance with singularity robustness using optimal damped least squares. In *IEEE International Conference on Robotics and Automation* (1992), pp. 434–441.

[dZ99] de Sá A., Zachmann G.: Virtual reality as a tool for verification of assembly and maintenance processes. *Computers & Graphics 23* (1999), 389–403.

[DZJ*99] Das H., Zak H., Johnson J., Crouch J., Frambach D.: Evaluation of a telerobotic system to assist surgeons in microsurgery. *Computer Aided Surgery* (1999), 15–25.

[EB99] Eftring H., Boschian K.: Technical results from manus user trials. In *International Conference on Rehabilitation Robotics, Stanford, CA, USA* (1999).

[FdAMS02] Fiorentino M., de Amicis R., Monno G., Stork A.: Spacedesign: A mixed reality workspace for aesthetic industrial design. *Mixed and Augmented Reality, IEEE / ACM International Symposium on 0* (2002), 86.

[FI98] Flückiger L., ISR 98: A Robot Interface Using Virtual Reality and Automatic Kinematics Generator. In *The 29th International Symposium on Robotics* (1998).

[GBSG03] Gimenez A., Balaguer C., Sabatini A., Genovese V.: The mats system to assist disabled people in their home environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA* (2003).

[HG94] Hillman M., Gammie A.: The bath institute of medical engineering assistive robot. In *International Conference on Rehabilitation Robotics, Wilmington, DE, USA* (1994).

[IC09] Ingrassia T., Cappello F.: VirDe: a new virtual reality design approach. *International Journal on Interactive Design and Manufacturing IJIDeM 3*, 1 (Feb 2009), 1–11.

[IEE90] IEEE standard: Glossary of software engineering terminology. *IEEE Std 610.12-1990* (Dec 1990).

[ISO94] ISO 10303-1:1994: *Industrial automation systems and integration Product data representation and exchange - Overview and Fundamental Principles, International Standard.* ISO TC184/SC4, 1994.

[ISO98] ISO/IEC standard: Ergonomic requirements for office work with visual display terminals - guidance on usability. *ISO 9241-11:1998* (1998).

[ISO01] ISO/IEC standard: Software engineering product quality part 1: Quality model. *ISO/IEC 9126-1:2001* (2001).

[Kaz98] Kazerooni H.: Human power extender: An example of human-machine interaction via the transfer of power and information signals. In *5th International Workshop on Advanced Motion Control* (Coimbra, Portugal, 1998), pp. 565–572.

[KBS*01] Kuutti K., Battarbee K., Säde S., Mattelmäki T., Keinonen T., Teirikko T., Tornberg A.: Virtual prototypes in usability testing. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 5* (Washington, DC, USA, 2001), IEEE Computer Society, p. 5029.

[KHK*07] Kanai S., Horiuchi S., Kikuta Y., Yokoyama A., Shiroma Y.: *An Integrated Environment for Testing and Assessing the Usability of Information Appliances Using Digital and Physical Mock-Ups.* Springer Berlin / Heidelberg, Berlin / Heidelberg, 2007.

[KNK*02] Kuffner J., Nishiwaki K., Kagami S., Kuniyoshi Y., Inaba M., Inoue H.: Self-collision detection and prevention for humanoid robots. In *2002 IEEE International Conference on Robotics and Automation* (Washington, DC, 2002).

[MBMP06] Muzzupappa M., Bruno F., Mattanò R. M., Pina M.: A new approach to participatory design: usability tests in virtual environment. In *Research in Interactive Design, Vol. 1, Cap. X* (New York, NY, USA, 2006), SpringerVerlag., pp. 80–90.

[MLG08] Montero F., Lozano M., Gonzalez P.: Usability-oriented quality model based on ergonomic criteria. *Handbook of Research on Web Information Systems Quality* (2008).

[MMGS09] Mahdjoub M., Monticolo D., Gomes S., Sagot J.: A collaborative design for usability approach supported by virtual reality a multi-agent system embedded in a plm environment. *Computer-Aided Design* (2009).

[MTT98] Magnenat-Thalmann N., Thalmann D.: *Modelling and motion capture techniques for virtual environments.* Springer Verlag, Geneva, Switzerland, 1998.

[MWM92] Mayorga R., Wong A., Milano N.: A fast procedure for manipulator inverse kinematics evaluation and pseudoinverse robustness. *IEEE Transactions on Systems, Man, and Cybernetics 22* (1992), 790–798.

[Nak91] Nakamura Y.: *Advanced Robotics: Redundancy and Optimization.* Addison-Wesley, Reading, MA, 1991.

[Nea] Neatech srl:. http://www.neatech.it.

[Nie93] Nielsen J.: *Usability Engineering.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Nor88] Norman D.: *The Design of Everyday Things.* Perseus Books Group, New York, USA, 1988.

[OAJ09] Oetomo D., Ang Jr M. H.: Singularity robust algorithm in serial manipulators. *Robot. Comput.-Integr. Manuf. 25*, 1 (2009), 122–134.

[OEF*06] Ott C., Eiberger O., Friedl W., Bäuml B., Hillenbrand U., Borst C., Albu-Schäffer A., Brunner B., Hirschmüller H., Kiehlöfer S., Konietschke R., Suppa M., Wimböck T., Zacharias F., Hirzinger G.: A humanoid two-arm system for dexterous manipulation. In *2006 IEEE International Conference on Humanoid Robots* (Genova, Italy, 2006).

[PT66] Piegl L., Tiller W.: *The NURBS book.* Springer-Verlag, New York, 1966.

[PW72] Presser L., White J.: Linkers and loaders. *ACM Computers Surveys 4*, 3 (Sept. 1972), 150–151.

[RDBB05] Robins B., Dautenhahn K., Boekhorst T., Billard A.: Robotic assistants in therapy and education of children with autism: can a small humanoid robot help encourage social interaction skills? *Univers. Access Inf. Soc. 4*, 2 (2005), 105–120.

[SK] Siciliano B., Khatib O. E.: *Springer Handbook of Robotics.* Springer.

[SKSH03] Seto F., Kosuge K., Suda R., Hirata H.: Real-time control of self-collision avoidance for robot using robe. In *2003 IEEE International Conference on Humanoid Robots* (Karslruhe, München, D, 2003).

[SR08] Shackel B., Richardson S. J.: *Human Factors for Informatics Usability.* Cambridge University Press, New York, NY, USA, 2008.

[SRS91] Sachs E., Roberts A., Stoops D.: 3-draw: A tool for designing 3d shapes. *IEEE Computer Graphics and Applications 11*, 6 (1991), 18–26.

[SS91] Siciliano B., Slotine J.: A general framework for managing multiple tasks in highly redundant robotic systems. In *5th International Conference on Advanced Robotics* (Pisa, Italy, 1991).

[SS00] Sciavicco L., Siciliano B.: *Modelling and Control of Robot Manipulators (2nd Ed.).* Springer-Verlag, 2000.

[Str09] Strang G.: *Introduction to Linear Algebra. 4th ed.* Wellesley-Cambridge Press, Wellesley, MA, 2009.

[VA06] VA: *Virtual Design 2 - Programmers Guide 4.5.2.* vrcom GmbH, 2006.

[Ves79] Veseliĉ K.: On a class of jacobi-like procedures for diagonalizing arbitrary real matrices. *Journal Numerische Mathematik*, 2 (1979), 157–172.

[Wal94] Walker I.: Impact configurations and measures for kinematically redundant and multiple armed robot systems. *IEEE Transactions on Robotics and Automation* (1994), 670–683.

[Wam86] Wampler II C. W.: Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Trans. Syst. Man Cybern. 16*, 1 (1986), 93–101.

[WIL88] Wampler C. W., II, Leifer L. J.: Applications of damped least-squares methods to resolved-rate and resolved-acceleration control of manipulators. *Journal of Dynamic Systems, Measurement, and Control 110*, 1 (1988), 31–38.

[Yam04] Yamane K.: *Simulating and Generating Motions of Human Figures (Springer Tracts in Advanced Robotics, V. 9).* SpringerVerlag, 2004.

[Zac00] Zachmann G.: *Virtual Reality in Assembly Simulation – Collision Detection, Simulation Algorithms, and Interaction Techniques.* 2000.

[ZB94] Zhao J., Badler N. I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics 13*, 4 (1994), 313–336.