POLITECNICO DI MILANO

Facoltà di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in

Ingegneria Biomedica

Recognition of Intentional Violations of Active Constraints in Cooperative Manipulation Tasks

Relatore: Elena DE MOMI, PhD

Correlatori: Ferdinando RODRIGUEZ Y BAENA, PhD

Stuart BOWYER, PhD

Prof. Stefano PASTORELLI

Tesi di Laurea di:

Mario ARICÒ Matr. 801270

Anno Accademico 2013 - 2014

Table of Contents

Sommario

Gli “Active Constraints” (vincoli attivi) sono algoritmi di controllo ad alto livello, utilizzati per assistere un operatore umano, durante la cooperazione uomo-robot. Gli Active Constraint sono definiti nel piano preoperatorio e il loro scopo è quello di delimitare regioni dello spazio di lavoro in cui il manipolatore può muoversi e incidere i tessuti in modo sicuro. Per migliorare le performance del chirurgo durante la cooperazione, sono stati introdotti vincoli virtuali di tipo adattativo, che adattano il livello di assistenza sulla base di informazioni sul task eseguito, sull’hardware o riguardanti l’utente che esegue il task. In letteratura, un primo esempio di impiego di vincoli adattivi è stato implementato sulla base di un modello di Markov continuo, utilizzato per il riconoscimento online dell’intenzione di un utente di allontanarsi da una traiettoria predefinita e evitare una serie di ostacoli circolari. In questo lavoro vengono presentati due metodi di classificazione per il riconoscimento di violazioni intenzionali e violazioni non intenzionali degli Active Constraint, sfruttando segnali di forza misurati in corrispondenza dell’end-effector dell’interfaccia aptica: un modello di Markov (HMM) continuo, and un modello basato su reti neurali di tipo feedforward. Entrambi gli approcci sono stati valutati su prove che prevedono il raggiungimento di un target o inseguimento di una traiettoria.

Durante la cooperazione, le violazioni di tipo intenzionale sono dovute ad un momentaneo disaccordo tra l’azione che l’utente esegue e la finalità del vincolo stesso. In genere, ciò è dovuto a limitazioni intrinseche del manipolatore, che non è in grado di adattarsi all’azione corrente dell’utente. Il vincolo è quindi percepito come un ostacolo per l’esecuzione della prova. Al contrario, le violazioni intenzionali sono causate da errori accidentali che l’utente commette durante il task, nonostante il vicolo sia correttamente applicato. In genere, gli Active Constraint sono classificabili in base alla loro finalità come: *Guidance Constraint*,

cioè vincoli impiegati per guidare il movimento del manipolatore lungo determinate traiettorie; *Regional Constraint*, cioè vincoli impiegati per delimitare i movimenti del robot all'interno di regioni sicure. In questo lavoro sono stati considerati entrambe le tipologie, che sono state implementate sulla base di una geometria planare, secondo un modello viscoelastico.

I segnali di forza misurati, sono stati impiegati per l'addestramento e la validazione di tre classificatori: un modello di Markov continuo basato sul modulo della forza di interazione; una rete neurale basata su variabili statistiche estratte dai segnali di forza; una rete neurale basata sulla distribuzione di energia del segnale di forza su differenti livelli di una decomposizione Wavelet. Per l'esecuzione delle prove, è stata impiegata l'interfaccia aptica "Phantom Omni" (Sensable Technologies, Inc.). Il sistema di controllo è stato implementato utilizzando librarie compatibili con la piattaforma di calcolo Matlab/Simulink R2014b. Il feedback visivo è stato fornito tramite uno schermo. Per valutare le performance di classificazione sono stati considerati due blocchi di prove:

- "Inseguimento di una traiettoria". All'utente viene chiesto di muovere l'interfaccia lungo una traiettoria 2D in modo più accurato possibile, aiutato da un vincolo di tipo "guidance". L'utente deve, inoltre, superare alcuni ostacoli circolari posti lungo il percorso;
- "Raggiungimento di un target". All'utente viene chiesto di posizionare il cursore in modo accurato sui punti target visualizzati che, con probabilità del 50%, giacciono all'interno di una regione proibita

È stato chiesto a 12 soggetti di effettuare 10 prove per blocco. Il segnale di forza è stato misurato per estrarre i diversi segmenti di interazione ($\boldsymbol{f} > 0$), che sono stati successivamente etichettati come "intenzionali" o "non intenzionali" a seconda di dati di posizione di ostacoli/limiti regionali. Dalle prove di tutti gli utenti, sono stati costruiti due set di dati distinti per l'addestramento supervisionato. Ciascun set è stato ulteriormente diviso in un set per l’addestramento (50%) ed un set per la validazione (50%). I tre classificatori sono stati applicati ad entrambi i set di dati. Le performance dei modelli sono state valutate in termini di sensitività (Se) e specificità (Sp), ottenendo l’andamento delle curve ROC nel tempo. Poiché abbiamo ritenuto che, in uno scenario applicativo reale, sarebbe più affidabile in termini di sicurezza un classificatore con alta specificità, è stato scelto di discretizzare l’output continuo dei modelli sulla base di una specificità impostata dall’operatore. L’indice di sensitività è stato conseguentemente ricavato.

I risultati hanno mostrato come, in generale, le prestazione dei classificatori migliorino all’aumentare dell’intervallo di tempo considerato dopo l’inizio della violazione, in quanto le curve ROC tendono asintoticamente al classificatore ideale. Ciò nonostante, il tempo minimo richiesto per raggiungere determinate prestazioni è differente. In generale, il modello StNN è risultato il più veloce rispetto al modello HMM, che a sua volta è più veloce del modello SpNN. Per prove di inseguimento di una traiettoria, tutti i modelli sono in grado di identificare violazioni intenzionali con una sensitività che supera il 90%, in un intervallo di tempo minore di un 1s. Per prove di raggiungimento di un target, i modelli StNN e HMM raggiungono prestazioni simili entro 1s dall’inizio della violazione, mentre il modello SpNN richiede un tempo maggiore di 2s. Possibili sviluppi futuri includono la valutazione di metodi per sfruttare l’output continuo del classificatore per modulare il livello di assistenza in base alla probabilità associata alle diverse intenzioni dell’utente durante cooperazione chirurgica.

Abstract

Active Constraints (ACs) are high-level control algorithms that are deployed to assist a human operator in man-machine cooperative tasks. Active constraints are preoperatively computed to define regions of the workspace within which it is safe for the robot to move and cut. To enhance the performance in cooperative surgical tasks, adaptive constraints have been investigated as a tool to optimally adjust the provided level of assistance, according to some knowledge of the task, hardware or user. In the literature, Hidden Markov Models have been deployed for run-time identification of whether the user wanted to intentionally leave a guidance constraint to circumvent circular obstacles placed along the path. In this work we present the evaluation of two classification methodologies for the runtime recognition of intentional and unintentional violations of ACs, exploiting interaction force signals measured at the tool tip of a haptic manipulator: continuous Hidden Markov Models (HMM) and feedforward Neural Networks (NN). Both approaches have been deployed for path-following and target-reaching tasks.

During cooperative assistance, intentional violation of ACs takes place whenever the current action of the user is in disagreement with the purpose of the constraint, typically resulting from environmental sensing limitations of the robotic system. In this case, the constraint is felt as a hindrance, resulting in disturbing interaction forces at the tool tip. Unintentional violations occur when the user shares the purpose of the constraint and accidental errors in the task execution are made anyway. Active constraints can have one of two purposes: *Guidance constraints* are enforced to guide the motion of the tool along a specified trajectory. *Regional constraints* are enforced to bound the motion of the tool into certain safe regions. Both types of constraints were considered in this work and modeled with a planar geometry, according to a conventional viscoelastic constraint model.

The measured force signals were deployed to train and validate three different classifiers: a continuous HMM, based on the magnitude of the force signal; a feedforward NN (StNN) based on seven statistical features (e.g. mean, variance, energy) extracted from the measured force signal; a feedforward NN (SpNN) based on spectral features, yielding the energy distribution of the force signal across different levels of its Wavelet decomposition.

The Phantom Omni (Sensable Technologies, Inc.) haptic device was used during assisted cooperative tasks. The active constraint controller was implemented on Matlab/Simulink R2014b platform. Visual feedback of the task execution was provided through a screen. To evaluate task-independent properties of the classification methods proposed, two sets of tasks were considered:

- Following task. The user was asked to move as accurately as possible along different 2D spline-based paths, assisted by a guidance constraint. He/she was asked to overcome any circular obstacle placed along the path by acting against the constraint;
- Reaching task. The user had to place the pointer as accurately as possible on several equally spaced targets, which lay within a forbidden region bounded by a regional constraint with 50% probability.

We asked 12 subjects to perform 10 trials for each of the two tasks. The interaction force between the tool tip and the constraint was recorded, segmented to extract non-null interactions ($\boldsymbol{f} > 0$), and automatically labeled as “intentional” or “unintentional” violations according to the known positions of obstacles/region boundaries. Two distinct datasets were built from all users across all trials for supervised learning, each furtherly split into a training dataset (50%) and a validation dataset (50%). The three classifiers were applied to each task dataset. The performance of the classification methods were computed in terms of sensitivity (Se) and specificity (Sp), yielding the Receiver Operating Characteristics (ROC) curves over time for each classifier. A classification

threshold was applied on the continuous output of the models. As we believed that, for safety reasons, it would be desirable to have a classifier characterized by a high specificity level, with high capability of detecting and rejecting unintentional violations, the discretization threshold was computed on the basis of a specificity level set by the operator. The sensitivity profile was consequently calculated.
Results showed that, in general, the classification performances of each methodology increased as the violation time widened: the ROC curves asymptotically tend to the ideal classifier. However, the minimum interval time necessary to reach a required performance was different. In general, StNN methods are faster than HMM, that, in turn are faster than SpNN. For path-following tasks, each classifier is able to detect intentional violations with a sensitivity level exceeding 90% within 1s after the violation has started. For target-reaching tasks, similar performances was achieved for StNN and HMM, while SpNN reaches similar performances after 2s. Future work will investigate methods to exploit the continuous output feature of the classification methods proposed in order to optimally modulate the provided assistance according to the probability of the user’s intention classification within surgical cooperative tasks.

1 Introduction

1.1 Robotics in Surgery: a general Overview

Over the last few years, robotic manipulators have been appearing in operating rooms as a valuable help to assist the surgeon during surgical interventions by enhancing the performances in terms of precision, accuracy and speed of execution. The reasons as to why this phenomenon is taking place is to be sought in the rapid development of new technologies and knowledge base in the industrial robotic field. Surgical applications of robotic devices have become feasible and safe thanks to the constant improvements in mechanical design, kinematics, control algorithms and programming that, consequently, enhanced the overall robot-assisted performances in comparison to the classic approach (Howe, 1999). Moreover, robotic researchers have worked to enhance the capabilities of the robot to deal with evolving environments, through the introduction of the concepts of adaptability and autonomy. The first characteristics refers to the capability of the robot to properly respond to dynamic workspace conditions, on the basis of sensory information; the second characteristics refers to the ability of the manipulator to carry out a given task without human supervision. With respect to surgical applications, these added features have been deployed for image processing, spatial reasoning, path planning, real-time sensing and real-time control. The State-of-the-Art provides two meaningful definitions of both "industrial robot" and "surgical robot". According to the Robot Institute of America, a robot is a "reprogrammable multifunctional manipulator, designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks. In (Davies, 2000) a robot is defined as a "reprogrammable computer-controlled mechanical device equipped with sensors and actuators”. Conversely, surgical robots are defined as “powered computer-controlled manipulators with artificial sensing that can be programmed to move and position tools in order to carry out a range of surgical tasks” (Davies, 2000).

The introduction of robotic assistants into the operating theatre does not imply the complete replacement of the human operator, but rather the establishment of a man-machine synergy, in which the manipulator is meant to “assist” the surgeon during the execution of complex tasks, thus “extending and enhancing human capabilities and dexterity” (Howe, 1999). This approach, based on the establishment of a close “cooperation” between the surgeon and the robotic device, was preferred to the classic approach used in industrial fields, where “fully autonomous” robots were used in many varied applications. Even though autonomous devices might offer many advantages over humans, such as resistance to harmful conditions, untiring strength and superior accuracy and precision (Bowyer, 2014), there exist some major bottlenecks that make them unsuitable for surgical applications:

- Autonomous systems are not able to deal with “highly unstructured” environments, where objects within the robot’s workspace are partially unknown due to sensing or data processing limitations. Any decision-making process will be based on incomplete or inaccurate information (Bowyer, 2014);
- Autonomous robots have a limited knowledge base concerning the task that they are executing. A robot may be programmed to perform one set of motions very accurately, but without a comprehensive understanding of the task’s purpose and environment. They are, thus, less likely to respond correctly to unusual or unexpected events (Bowyer, 2014);
- Autonomous robots are unsuitable for some applications where the culpability for damage or harm is unclear: there often needs to be a

distinction regarding who is responsible for the safety of people and objects within the robot's workspace (Bowyer, 2014).

The introduction of cooperative approaches has been the best solution the tackle the three underlined drawbacks, which are unavoidable when using fully autonomous devices into the surgical framework. It is important to highlight how the success of the human-robot cooperation is based on the fact that both the robotic assistant and the human operator provide complementary advantages and tend to make up for each other weaknesses. On one hand, the manipulator provides enhanced precision, accuracy, dexterity and, in general, is able to use

	SURGEON	ROBOT
PROS	1. Task versatility; 2. Judgement experience; 3. Hand-eye coordination; 4. Dexterity at mm/cm scale; 5. Integrate qualitative data; 6. Flexible and adaptable.	1. Good geometric accuracy; 2. Stable and untiring; 3. Can use diverse sensors; 4. May be sterilized; 5. Can be designed for a wide range of scales; 6. Optimized for particular environments.
CONS	1. Tremors; 2. Fatigue; 3. Limited geometric accuracy; 4. Limited sterility; 5. Low dexterity outside natural scale; 6.Susceptible to radiation and infections.	1. Poor judgement; 2. Expensive; 3. Cumbersome and large; 4. Not versatile; 5. Inability to process qualitative information; 6. Technology in flux.

Table 1.1 Strengths and weaknesses of the surgeon and the robotic manipulator during cooperative tasks.

detailed *quantitative* information (Howe, 1999), like 3D imaging data and intra-surgical sensory data. On the other hand, humans are superior at integrating different sources of *qualitative* information and consequently exercise judgement. They are characterized by excellent hand-eye spatial coordination and have a finely developed sense of touch. Table 1.1 summarizes the advantages and disadvantages of human and robot capabilities.

1.2 History and applications

Fully autonomous robotic manipulators first appeared in the industrial field in the early 1960s. However, the first attempts to introduce these devices into the surgical scenario are much more recent, as they are dated back to the mid-1980s. In 1985, Y. S. Kwoh used a standard industrial device to hold a fixture next to the patient’s head, in order to locate a biopsy probe for neurosurgery applications (Kwoh, 1988). The robot deployed was a “Puma 560”, a serial manipulator characterized by 6 DoF (Figure 1.1). This manipulator was chosen after a careful evaluation of the industrial devices available on the market, and was driven by three main criteria: dexterity, accuracy and reliability while operating in surgical environments. The 3D coordinates of the target points on the brain surface were computed by means of a stereotactic frame coupled with a CT scanner and then they were located in the manipulator’s reference frame. Once the Puma had

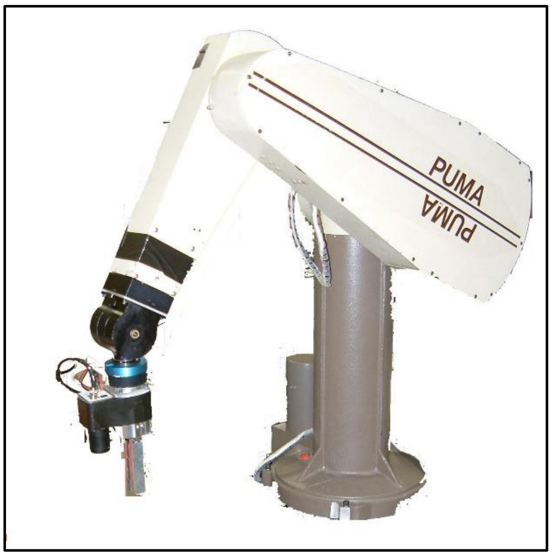


Figure 1.1 Industrial Robotic Manipulator "Puma 560"

reached the target position, the power was removed, while the surgeon used the fixtures to orientate the drills and the biopsy probes, inserted manually into the skull. An overview of the system is shown in Figure 1.2.

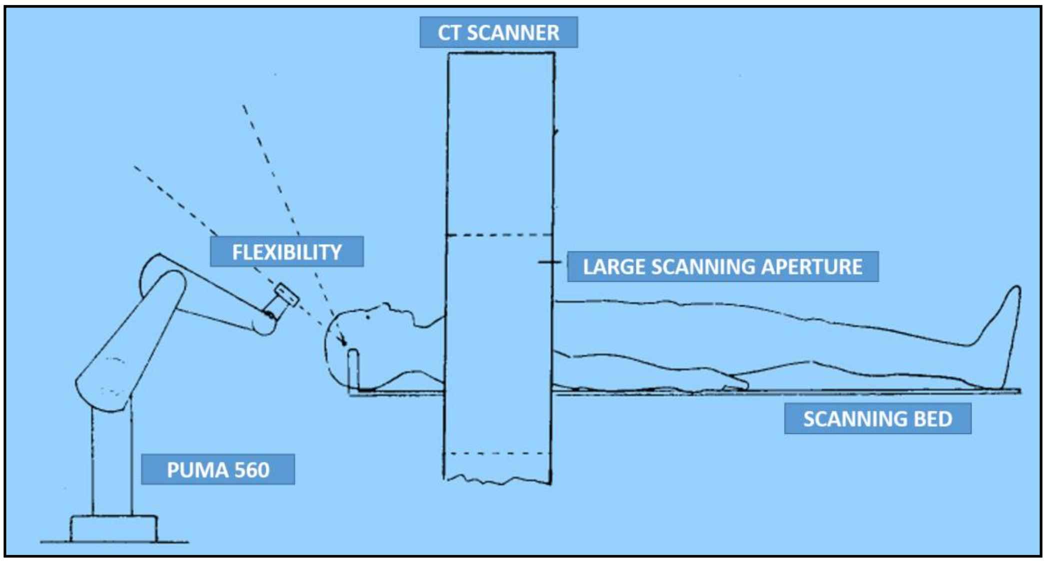


Figure 1.2 Overview of the surgical system deployed for brain biopsies. The CT scanner provides the correct spatial relationship between the imaging reference frame and the reference frame of the robotic manipulator (Kwoh, 1988).

In parallel with the researches carried out by Y. S. Kwoh, R. H. Taylor was implementing a surgical scenario that involved an industrial robot system for hip replacement surgery in dogs (Taylor, 1989). A “Scara” robotic manipulator (Figure 1.3) was used to hold in place a rotating cutter, which reamed out the proximal femur to take the femoral stem of a prosthetic implant for a total hip replacement. After a number of studies, the robotic system was deployed as a “veterinarian robot” for replacing hips of pet dogs under the supervision of veterinarian surgeons (Davies, 2000).

In the early 1990s, after a number of studies focused on the introduction of industrial robots into the surgical scenario, a new trend in research was elicited by the shared concerns about the low safety standards that characterized the use

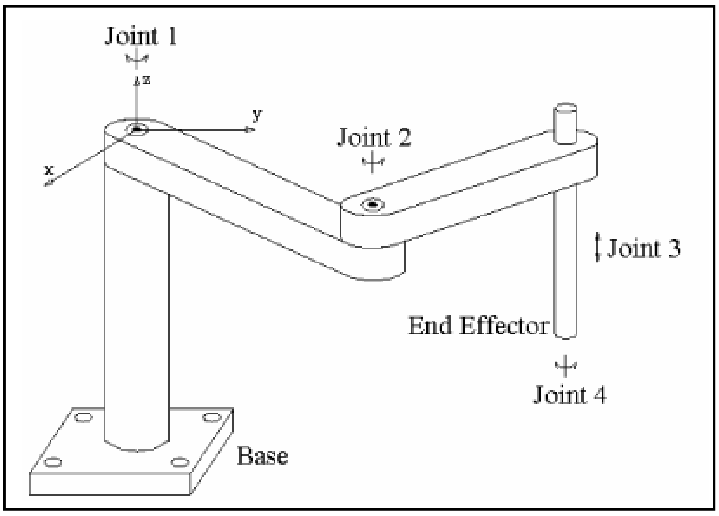


Figure 1.3 Selective Compliance Assembly Robot Arm (SCARA). SCARA is an industrial robot with four parallel rotational axes and four degrees of freedom.

of industrial manipulators close of human operators. In general, industrial robots were designed to be deployed in general-purpose tasks, and were thus characterized by wide ranges of motion, that made them inherently less safe than special-purpose mechanisms, whose ranges of motion and forces were set specifically for a given task. “Special-purpose” manipulators have gained credibility thanks to the fact that they can be safely applied and controlled with limited ranges of force and position; in addition, a dedicated system has the possibility to run customized software (Davies, 1991). The first special-purpose device was conceived in the late 1980s and was clinically tested in April 1991. The robot was characterized by a tailored kinematic system, designed to remove prostatic adenomas: this was the very first time that a manipulator had been used automatically to remove tissue from patients (Davies, 2000). Since that time, a second-generation prostate robot, called “*Probot*” has been developed at Imperial College (Ng, 1993).

As from the mid-1990s, trends in surgical robotics have been devoted towards two main goals:

- From the technology points of view, cooperative robots have been preferred over autonomous manipulators: instead of the complete replacement of the surgeon, the robot becomes an extension on his/her hand (Dogangil, 2010), making up for the limitations of human performances such as tremor, fatigue and limited dexterity at sub millimeter scales (Table 1.1).
- On the application point of view, the trend flows toward "*Minimally Invasive Surgery*" (MIS). This approach represents one of the newest trends in surgical robotics and is characterized by far less invasive procedures compared to the open ones: laparoscopic devices are inserted through small incisions in the tissue and the operation is externally supervised by means of image guidance, provided by endoscopic cameras. MIS approaches results in fewer post-operative complications, shorter recovery times and, sometimes, outpatient treatments for previously longer procedures (Dogangil, 2010).

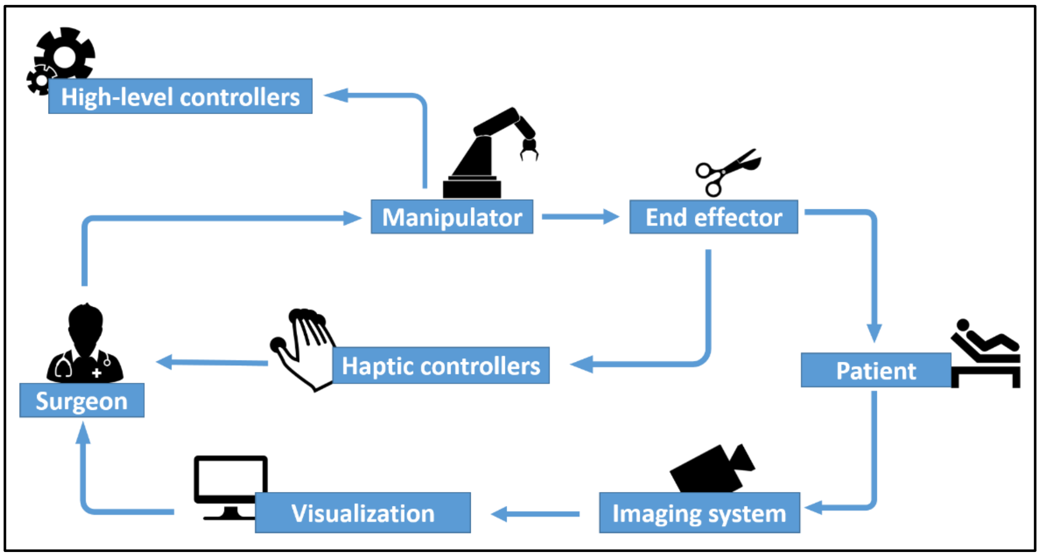


Figure 1.4 Building blocks of a general surgical robotic system. The surgeon supervises the robot's activity by means of the haptic feedback and the visual feedback provided by the imaging system.

Any successful surgical robotic system consists of some fundamental building blocks that operate in synergy (Figure 1.4): the surgical manipulator, the imaging system, the visualization equipment, the high-level controllers, the end-effector tools, the haptic controller devices and, most importantly, the surgeon (Dogangil, 2010). It is important to note how the surgeon is kept within the control loop: he/she supervises the robot's actions with the help of augmented reality provided by the haptic and visual feedbacks. The growing acceptance of these surgical robotic systems into the operating rooms is witnessed by the many commercial devices that are currently available on the market and that apply to a wide range of clinical scenarios:

- *Neurosurgery* (Varma, 2006), (Coste-Manière, 2005), (Sutherland, 2003);
- *Eye surgery* (Jensen, 1997), (Ergeneman, 2008);
- *Thoracic and cardiac surgery* (Häcker, 2005), (Boehm, 2000);
- *Orthopedic surgery* (Kazanzides, 1995), (Lavallee, 1995);
- *Urologic surgery* (Krieger, 2005), (Salcudean, 2008);
- *Tumors radiation* (Adler, 1997).

Extensive reviews of currently available medical robotic system are reported in (Howe, 1990), (Dogangil, 2010), (Camarillo, 2004) and (Curley, 2005). The following sections are devoted to the description of some meaningful examples of surgical robotic applications.

1.2.1 Neurosurgery

Neurosurgery was one of the first surgical specialties that deployed "image-guided" techniques to perform the operation. In the preoperative phase, the imaging system computes a 3D model of the patient's brain, obtained through the processing of multiple CT or MRI images. The target points (e.g., intra-cortical lesions or stimulation points) are then identified within the volumetric model and their coordinates are computed with respect to the image reference frame.

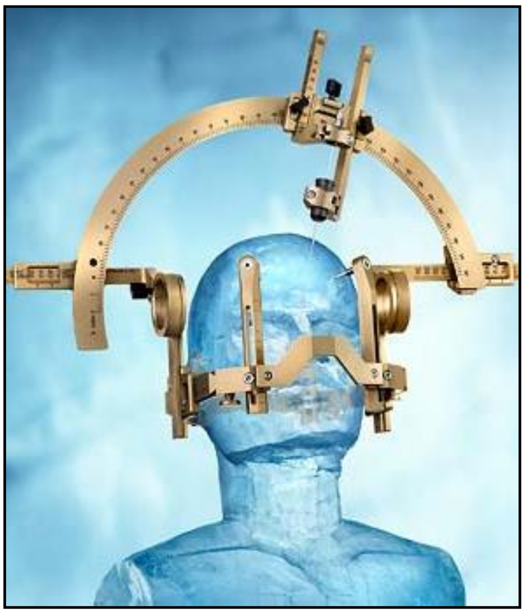


Figure 1.5 Stereotactic frame for the mapping procedures of the target points within the skull.

A stereotactic head frame (Figure 1.5) attached to the patient's skull provides the correct registration matrix to obtain the coordinates of the target points in the reference frame of the robotic system. The location of the target points is consequently used to compute the optimal operation plan in terms of degree of alignment, orientation and insertion of the end-effector tool to the desired point in the brain (Dogangil, 2010).

One of the major drawbacks of image-guided neurosurgery is the so-called "brain shift" phenomenon, that alters the relationship between the preoperative image data and the current anatomy of the patient. To overcome these problems two solution have been proposed (Dogangil, 2010):

- Integrate deformable templates for nonrigid registration, based on biomechanical models of the soft tissue;
- Integrate intra-operative imaging for a continuous monitoring of the patient's anatomy and instrumentation; this requires that the manipulator be compatible with the imaging modality and space constraints;

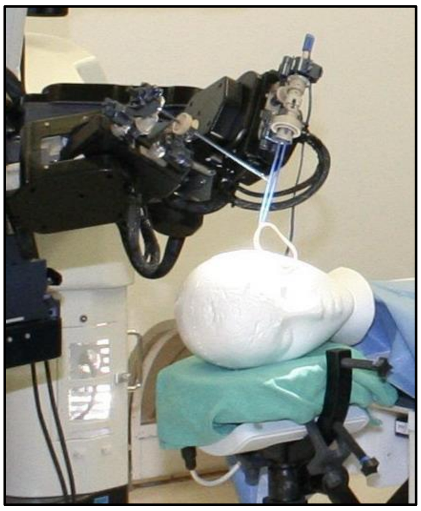


Figure 1.6 NeuroArm surgical robot.

Two commercially available neuro-robotic systems are the “NeuroArm” and “CyberKnife” systems. The “NeuroArm” (Figure 1.6) (Figure 1.7) is a MRI-compatible system designed for stereotaxy and microsurgery, including manipulation and cutting of soft tissues, dissection of tissue planes, suturing, electro-cauterize, aspirate and irrigate (Dogangil, 2010). The working principle is based on teleoperation, in which two remote slave manipulators replicate the hand movements of the master manipulator directly controlled by the operator. In turn, the surgeon supervises the execution of the task by means of stereoscopic visual feedback, 3D MRI displays and haptic force feedback provided by multi-axial force sensors mounted on the end-effector.

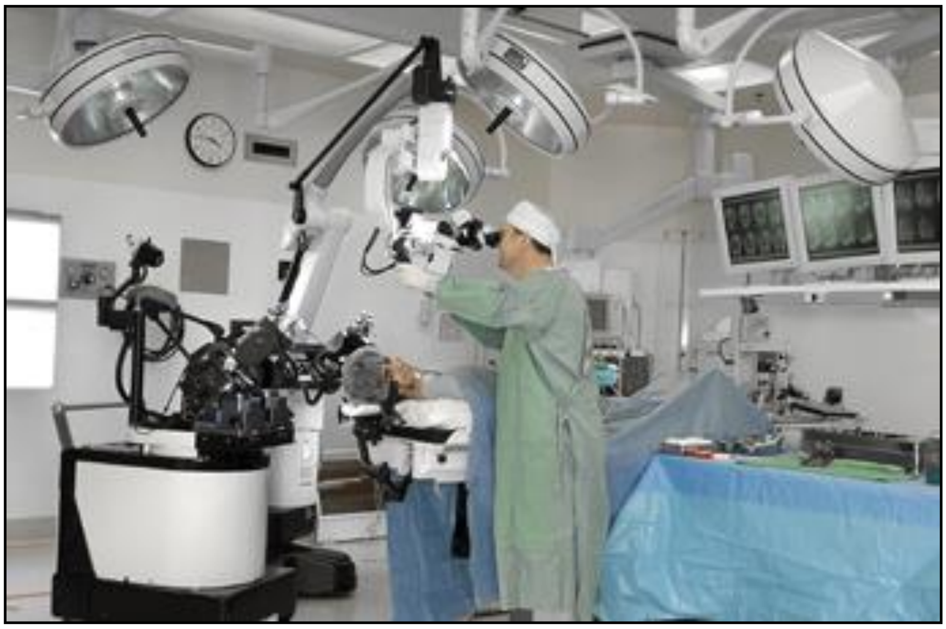


Figure 1.7 Overview the of NeuroArm robotic system.

The “CyberKnife” system (Figure 1.8) is a non-invasive robotic manipulator designed for radiosurgery. It consists of a compact linear accelerator mounted on a 6-DOF robotic arm that can be used to radiate a variety of tumors. During the preoperative phase, CT images are used for the path-planning of the system. During the intra-operative phase a set of X-ray cameras, coupled with flat panel image detectors, monitors the movements of the patient and consequently, unlike stereotactic systems, compensates for them. Recently, the accuracy of the treatment has been enhanced with the introduction of a respiratory tracking system, which allows the “CyberKnife” to track lesions that move with breathing and follow them in real time for more precise treatments (Dogangil, 2010).

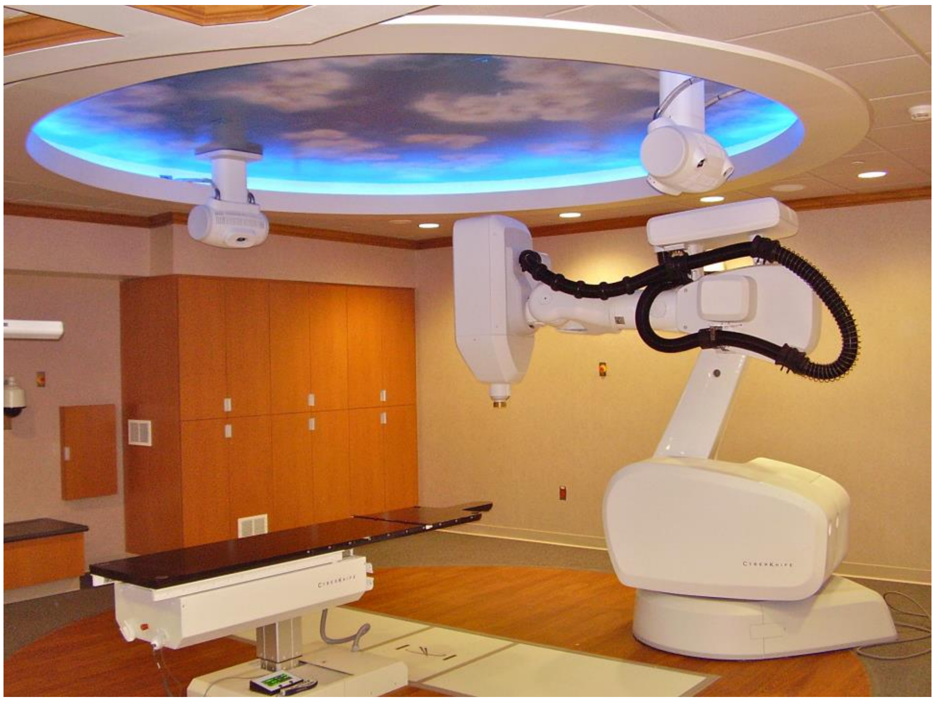


Figure 1.8 “CyberKnife” System for tumor radiation.

1.2.2 Cardiac and thoracic surgery

Thoracic and cardiac surgery, and in general abdominal surgery, is one of the fastest growing arenas of use for robotic telemanipulation (Curley, 2005). Over the last decade, the trend in the design and implementation of these robotic systems has flowed toward a laparoscopic methodology, that meets the requirements of “Minimally Invasive Surgery” (MIS) approach. The use of laparoscopes, as opposed to the classic open procedure, provides benefits such as fewer post-operative complications, shorter recovery times and reduced tissue scars. The execution of the surgical intervention is performed through small incisions in the abdominal or thoracic walls (Figure 1.9). The surgical tools are inserted through such incisions toward the target point within the abdomen, along

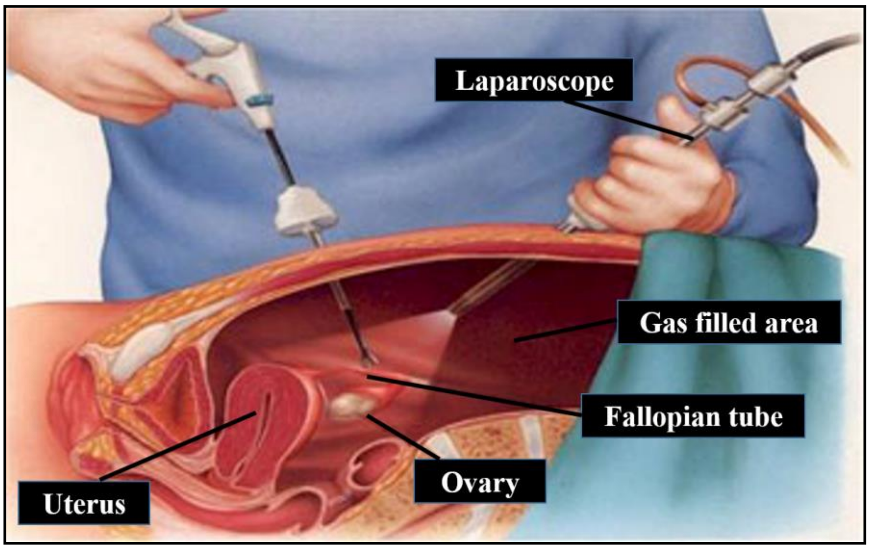


Figure 1.10 Cross-sectional view of gynecological laparoscopic procedure.

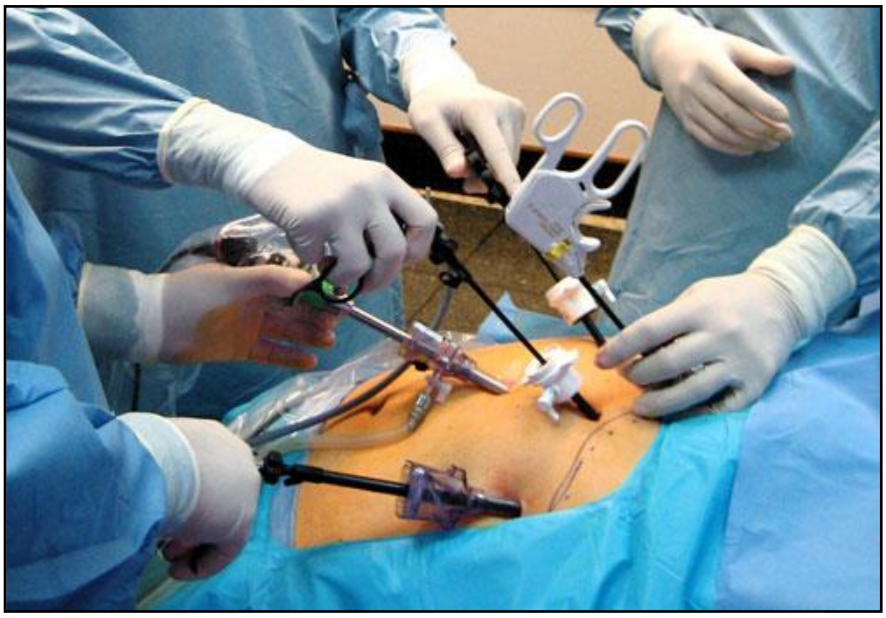


Figure 1.9 Laparoscopic tools inserted through skin incisions in the abdomen.

with an endoscopic camera for visual feedback to the surgeon (Figure 1.10). The development of laparoscopic manipulators, despite its benefits for the patients, has introduced a number of significant challenges to the surgeon (Curley, 2005):

- Loss of haptic feedback. The operator is, thus, unable to perceive the interaction forces between the tool tip and the tissue;
- Limitation in the range of motion of the instrumentation (generally four degrees of freedom against the hand and wrist's seven) ;
- The rise of the so-called "fulcrum effect", that forces the operator to make counterintuitive movements with the end effectors;
- Amplification of the physiological tremor at the end effectors;
- Limited dexterity;
- Impaired depth and field of vision without using dual-camera 3D systems;
- Steep learning curve.

One of the most successful robotic manipulator for minimally invasive laparoscopic surgery is the "da Vinci" system. It is a master-slave robotic system that was initially designed for closed chest cardiac surgery (Boehm, 2000), although many more application have been explored, like visceral surgery, gynecology and urology surgery. The "Da Vinci" system is composed of two elements: the surgeon and patient sides. The surgeon side (Figure 1.11) component is the control console that consists of a binocular viewer for stereoscopic vision, finger-

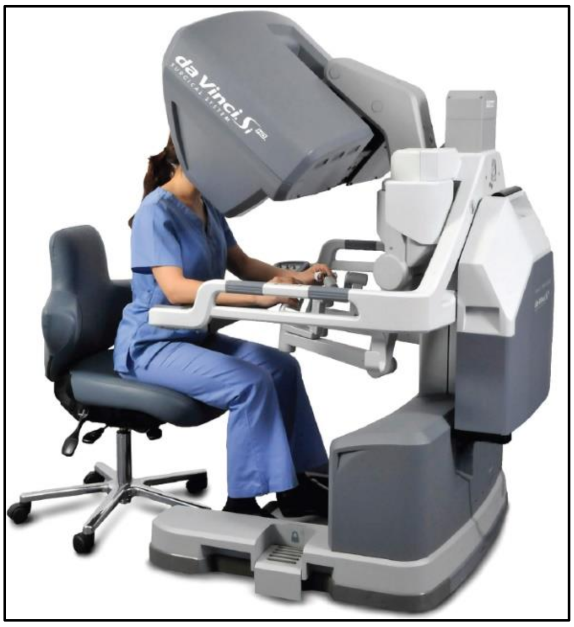
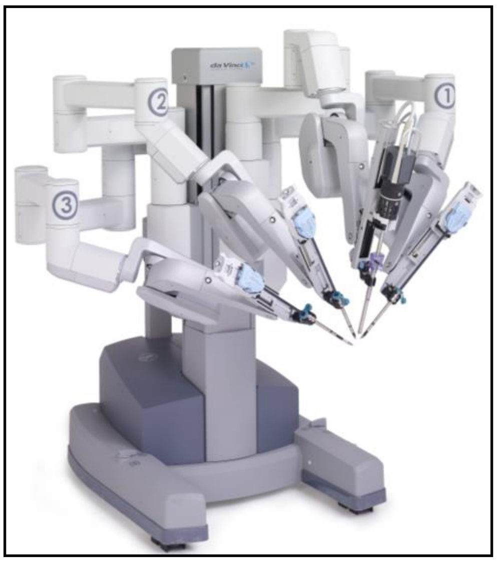


Figure 1.11 "Da Vinci" System: control console of the surgeon



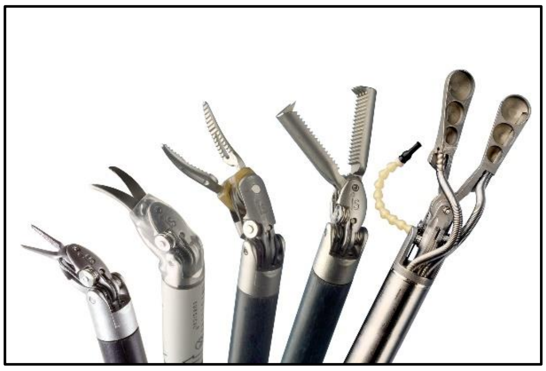


Figure 1.13 "Da Vinci" System: examples of laparoscopic tools for tissue manipulation

Figure 1.12 "Da Vinci" System: Slave manipulator with three robotic arms for surgical tools and one robotic arm for the endoscopic camera.

held controllers through which the operator remotely manipulates the robotic arms, foot control pedals for camera focus and orientation, clutch and diathermy, and the central computer. The patient side component (Figure 1.12) (Figure 1.13) is composed of four articulated arms, one dedicated to the camera, which provide image magnification from 2X to 10X, and three for instruments. The surgeon controls any two arms at a time. The system is designed to merge the advantages of freehand movements, used in open surgery, the benefits of a minimally invasive approach and the increased precision and accuracy that arise from motion scaling, whereby large motions of the master device are scaled down proportionally to produce small motions of the slave.

1.2.3 Orthopedic surgery

Along with neurosurgical applications, orthopedics was one of the first areas in which robot applications were developed (Howe, 1999). The fundamental difference existing between orthopedics and other type of surgery (e.g., neuorosurgey, thoracic surgery, eye surgery) is the profound diversity of the biomehcanics behavior of the target tissues. While soft tissue requires complex biomechanics model to implement reliable robotic navigation, bones are relatevely easy to manipulate, as the amount of deformation is negligigle during surgical procedures. This property results in simpler image-guided techniques to implement the problems arising from mismatches between the preoperative and intraoperative plans are overcome. Orthopedics application that have received the greatest attention are hip and knee replacement, and spinal fusion (Howe, 1999).

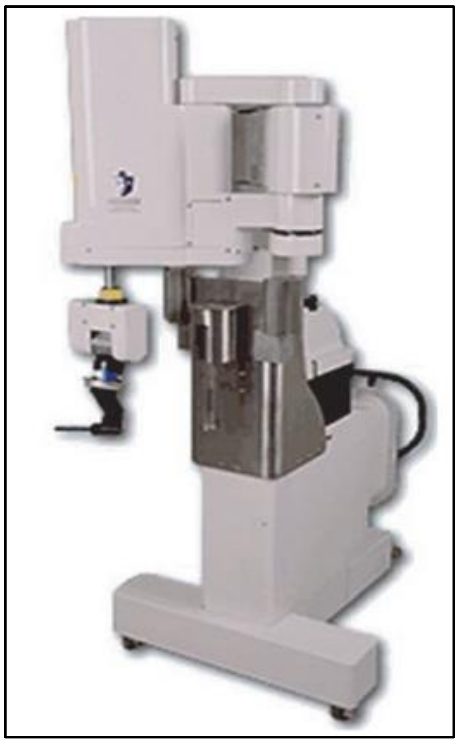
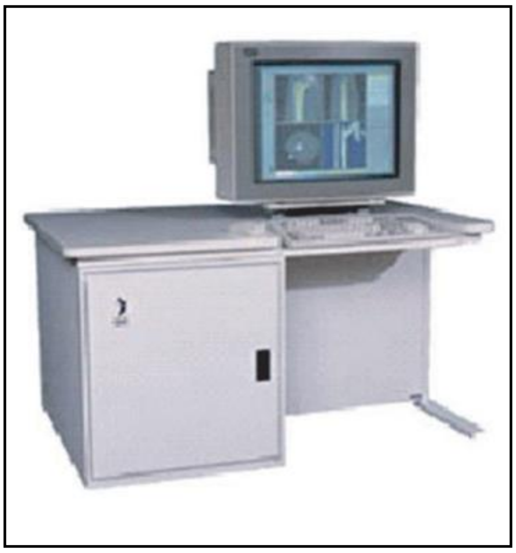


Figure 1.14 "Robodoc" System.

An example of orthopedic manipulator for hip surgery replacement (Kazanzides, 1995) is the “Robodoc” system (Figure 1.14), that was developed to improve the performances while forming the femural cavity to host the hip prosthesis. In comparison to the manual procedure, in which the surgeon cuts the cavity by forcing hand-help broaches and reamers into the femur, the robotic system provided a less rough and uneven internal surface. The tip of the manipulator is equipped with a high-speed rotary cutter that precisely reams out the femoral cavity, adjusting its dimesions according to the stem of the implant. In addition to increased accuracy in the formation of the cavity, the implant size and placement can be chosen on the basis of preoperative CT images. A separate 3D planning workstation, called “Orthodoc” (Figure 1.15), computes a 3-dimensional model of the patient hip and the surgeon can manually



adjusts the position and orientation of the prosthesis until satified. The resulting femoral cavity is then dislayed and the sequence of robot motions are automatically planned (Howe, 1999).

Figure 1.15 "Orthodoc"system for path-planning of the orthopedic surgical procedure.

1.3 Surgical Robot Taxonomy

Over the past 200 years, a wide variety of robots for surgical applications have been designed, developed and clinically used. Several authors (Camarillo, 2004) have proposed to organize the different varieties of surgical manipulators into a taxonomies. Among others, the most widely shared classification is based on the role that each device has within the operating room. Three discrete categories have been identified:

- *Passive role manipulators*. The role of the robot is limited and its involvement is related to only low risk tasks. An example of passive manipulator has been discussed in section 1.2.1, in which an industrial Puma 560 (Figure 1.1), was deployed to firmly hold in position a fixture newt to the head of the patient, in order to allow for the surgeon safely insert needles for brain biopsy. Further applications are endoscopic holders (e.g., AESOP (Sackier, 1994)), whose role is to maintain endoscopic tools in a steady position during laparoscopic procedures.

- *Restricted role manipulators*. In this case, the robot is involved in higher-risk tasks, but its role is still restricted to essential portions of the procedure. Examples are cooperative manipulators and mechanical stabilizers that filter out the noise due to hand tremor.
- *Active role manipulators*. The robot is intimately involved in the task procedure and carries high responsibility and risk (Camarillo, 2004). Usually, autonomous robots compute preoperative plans without human intervention. Examples of autonomous manipulators include the “CyberKnife” system for radiosurgery (Section 1.2.1) and the “Robodoc” system for hip replacement surgery (Seciont 1.2.3).

Although autonomous robots might appear superior, it should be noted that their deployment requires significant human supervision, often resulting in an

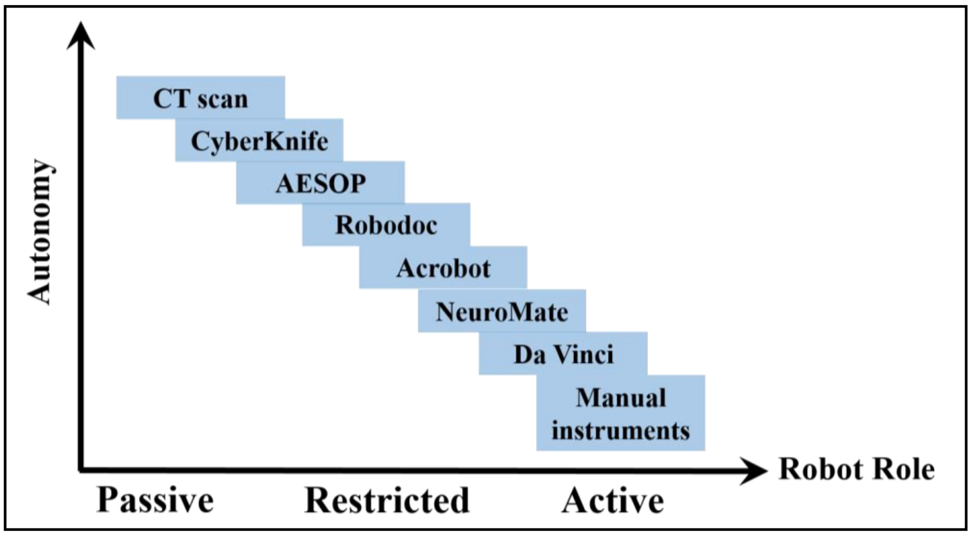


Figure 1.16 Tradeoff between the procedural role and autonomy. Procedural role indicates the level of responsibility and involvement the robot has with the patient during a procedure. Robot role scales up with greater duration, scope, invasiveness, and risk, which decreases the level of autonomy in current systems (Camarillo, 2004).

increased burden on the surgeon. This is an important trade-off (Figure 1.16) to evaluate when choosing the proper surgical system for a given application.

1.4 Active Constraints

The introduction of robotic manipulators into the surgical field, despite all the advantages deriving from enhanced precision, accuracy and dexterity, has raised high concerns about the safety of the human operators and the patients, that, during surgical procedures, share their workspace with the robot. Before the first studies by Y. S. Kwoh in 1985 (Kwoh, 1988), who used a "PUMA 560" industrial manipulator as a fixture holder to improve the accuracy during probes insertion for brain biopsies, fully autonomous systems were designed to be deployed in the industrial field, at different stages of the manufacturing processes. Their main application included welding, painting, assembly, pick and place, and testing. All tasks were accomplished with high precision, accuracy and speed. The human operator was in charge of the supervision of the process, but did not share the workspace of the manipulator. The success of autonomous robots in the industrial field relies on the fact that the manufacturing process is the sum of repetitive simple tasks. Hence, the robot can be programmed to execute each task very precisely, accurately, and with higher ranges of speed with respect to the human performance. Conversely, a surgical scenario involves inherently many degrees of variability, both with respect to the types of task to be executed, and with respect to their execution time and sequence. Despite all the advantages, there exists a limit in the capability of autonomous manipulators to perceive and evaluate their environment, making them unsuitable to deal with highly unstructured processes. These processes involve incomplete or inaccurate sensory information about the surrounding objects and thus, put some limitations in the reliability of any decision-making process of the autonomous device (Bowyer, 2014). To overcome these limitations, the concept of "human-robot cooperation" was introduced, in which the robotic assistant intelligently regulates the motion of the human user (Bowyer, 2014), making up for each other weaknesses (Table 1.1) This shared control concept was first envisaged by Rosenberg in the early 1990s: "virtual fixtures", as he conceived them, are defined as "abstract sensory information overlaid on top of reflected sensory feedback from a remote environment" (Rosenberg, December 1993). The function of virtual fixtures is to enhance operator performances by allowing precision to exceed natural human abilities, while reducing mental and physical workload associated to the task (Rosenberg, September 1993). Rosenberg conceived these virtual tools as a "ruler" (Rosenberg, September 1993), which can greatly improve the performances of a straight line-drawing task. The use of the ruler reduces the mental processing, speeds up the execution of the task and allows for a much better outcome. Without this tool, the drawing is a manual task that requires constant visual supervision and hand-eye coordination (Rosenberg, September 1993). Similarly to "virtual fixtures", "active constraints" (ACs), are high-level control algorithms that can be used to assist a human in man-machine cooperative tasks. In (Abbott, 2007), ACs are defined as "software-generated force and position signals applied to a human operator in order to improve the safety, accuracy and speed of the robot-assisted task. They capitalize on both the accuracy of the robotic system and the intelligence of the human operator". In (Bowyer, 2014), ACs are defined as "collaborative control strategies which can be used in human manipulation tasks to improve or assist by anisotropically regulating the motion. Throughout operation, the robot controller monitors tool motion, and analyzes it with respect to preplanned trajectories and known restricted regions. The active constraint controller then attenuates or nullifies any user command that will cause the manipulator to digress from a plan, or enter a forbidden region" (Bowyer, 2014).

One of the first research systems that led to the design and implementation of a robotic device, with an additional active constraints controller, is the “Acrobot” system (Figure 1.17), developed at Imperial College London as from the late

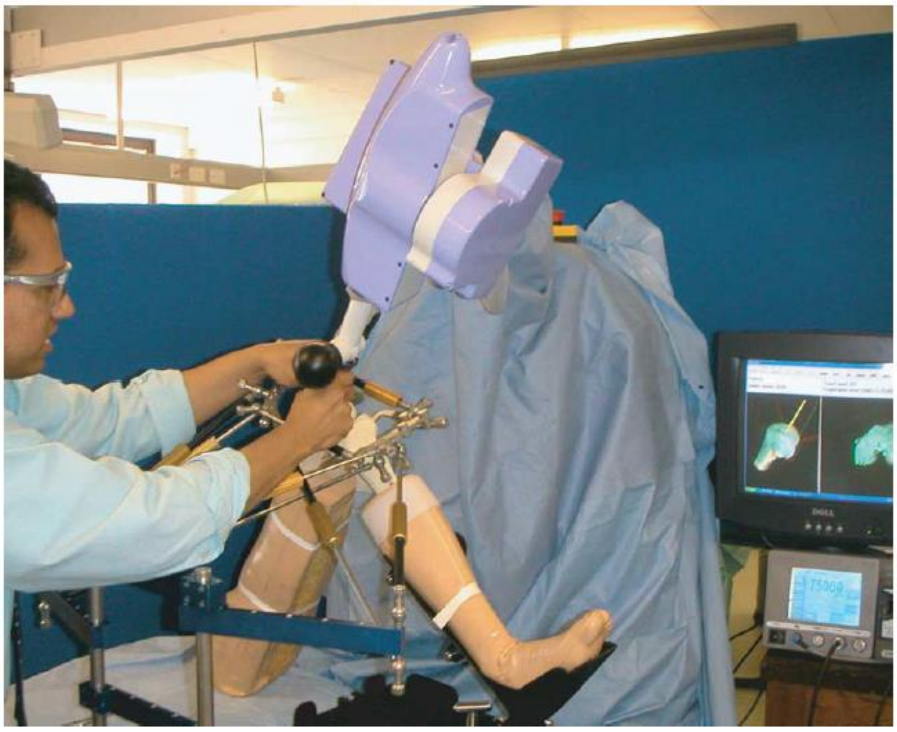


Figure 1.17 "Acrobot" System developed at Imperail College, for Unicodylar Knee Arthroplasty (UKA).

1990s for unicondylar knee arthroplasty (Davies, 2006). Such a system is a “hands-on manipulator in which the surgeon holds a force-controlled handle that is located near the tip of the robot” (Davies, 2006). The hands-on approach, which results in a direct physical human-robot interaction, provides reliable haptic feedback without the need for force sensors: the surgeon is then able to feel the difference when cutting either hard or soft tissues. The pre-operative CT-based planner defines the regions of the workspace within which the robot can move and cut without causing damages or harm. If the operator tries to enter any forbidden regions of the robot workspace, the controller actively constraints the motion within safe positions, ensuring that complex cut surfaces are accurately achieved, and that critical features are preserved. The performances of the "Acrobot" system was tested in a randomized, double-blind (patient and evaluator) trial, and the results were compared to the conventional surgery. The evaluation criteria were based on the absolute values of alignment error in valgus and varus directions. Such errors were computed for both the femoral and the tibial components. Results showed that the "Acrobot" system reduces the risk of inaccurate outcomes in arthroplasty. Moreover, a significant enhancement in the post-operative improvement was observed (Davies, 2006).

1.4.1 Generalized Active Constraint Framework

A correct and extensive discussion about active constraints, their use and application requires the establishment of a generalized framework. As discussed in (Bowyer, 2014), this framework is composed of three main processes (Figure 1.18):

- *Constraint definition;*
- *Constraint evaluation;*
- *Constraint enforcement.*

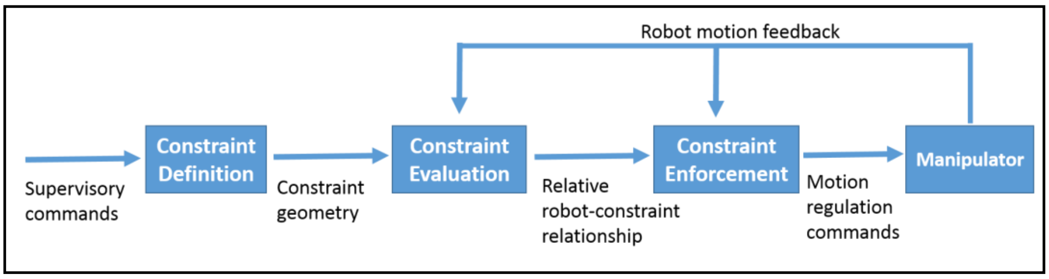


Figure 1.18 Generalized Active Constraints Framework (Bowyer, 2014).

The definition of an active constraint refers to the geometry that describes the constraint itself. This first process can be carried out either via human supervision or by using automatic algorithms. Once the geometry has been defined, it has to be evaluated: the constraint geometry is compared to the pose of the manipulator to assess the reciprocal compatibility. The last step is the constraint enforcement, namely the conversion of the relative robot-constraint configuration into input commands, which in turn will regulate the motion of the human user.

1.4.1.1 Constraint Definition Methods

The definition step is carried out by computing the spatial geometry that describes the features of the constraint. Although in literature there are some methodologies to automatically generate constraint geometries on the basis of medical images, the most widely used approach is the *apriori* definition of such geometries, via a separate process that is often supervised by a human operator (Bowyer, 2014). Constraint definition methods span from simple points and lines, to complex surfaces and volumes and will be briefly described and discussed. A comprehensive review can be found in (Bowyer, 2014).

Point Constraints.

A point-like geometry (Figure 1.19) is a rather simple constraint definition method based on a closed-form solution that efficiently represents either 3D Cartesian points, or configuration and orientation poses. This approach has been widely used in literature for tasks requiring accurate and precise tool positioning.

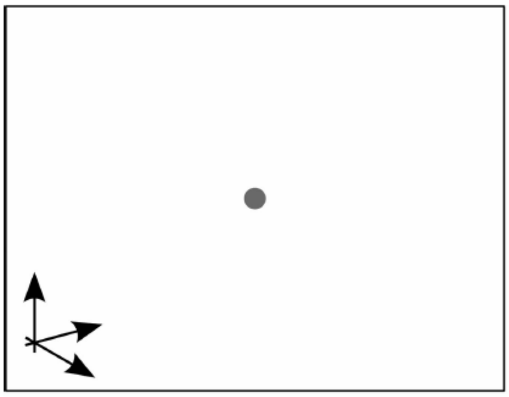


Figure 1.19 Point Constraint representation (Bowyer, 2014)

Linear Constraints.

Linear constraints define paths in the 3D space with a variable degree of complexity, ranging from straight lines, to sinusoids, to spline-based curves. Straight lines represent the simplest geometry, have a closed-form solution, but are not suitable to describe complex trajectories. On the contrary, spline-based geometries are suitable to deal with a higher degree of complexity, but typically suffer from the absence of any closed-form solution, which results in an increased memory demand.

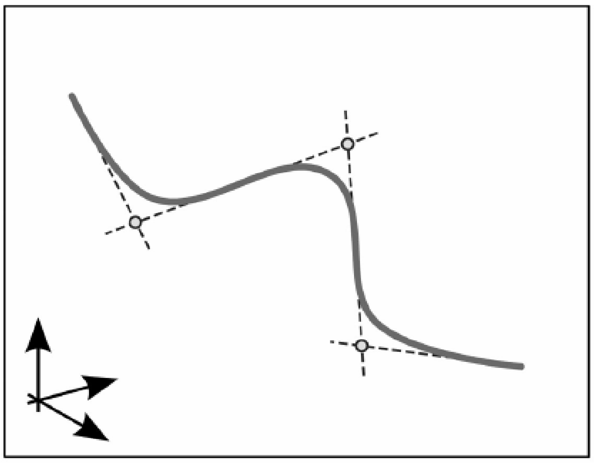


Figure 1.20 Linear Constraint representation (Bowyer, 2014)

Surface Constraints.

Surface constraints are used to divide the robot's workspace into separate subspaces. Like linear constraints, there exist different methodologies to

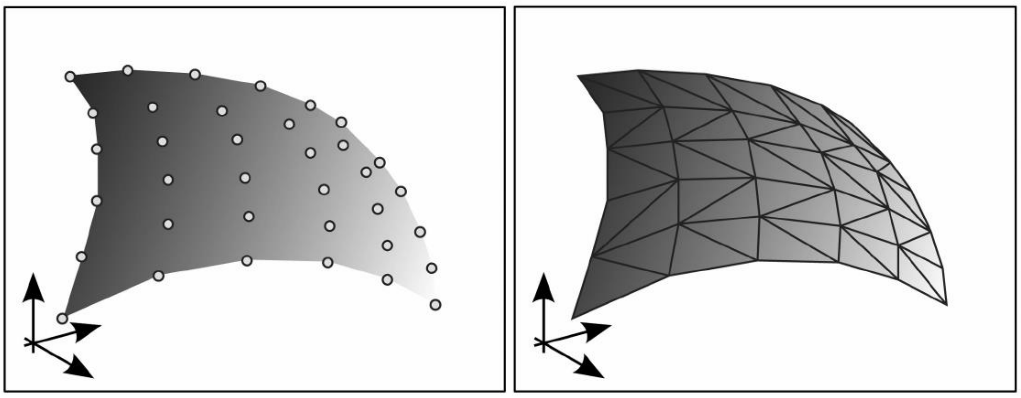


Figure 1.21 Surface Constraint representation. On the right, is it shown a surface defined as a cloud of points. On the left, it is shown a surface defined by a polygonal mesh (Bowyer, 2014).

implement the relative geometries of such surfaces. The simplest approach, limited to a low range of complexity is the segmentation of the surface into a subset of hyperplanes. Higher degrees of complexity require the adoption of more sophisticated methods, such as “polygonal mesh” and “point cloud” constraints (Figure 1.21). Thanks to their great flexibility, polygonal meshes are useful when the application requires the precise reconstruction of real-world surfaces. Unfortunately, despite their advantage, this type of constraint is difficult to construct, evaluate and store. On the other hand, “point cloud” constraints, being defined by sampling a set of Cartesian points from the surface are more easily constructed and stored.

Volumetric constraints.
Volumetric constraints (Figure 1.22) are defined by means of analytical geometric primitives, like spheres, cylinders, cones and cubes. Thanks to their closed-form solution that provide efficient computation, volumetric constraint have been proposed as an alternative to describe complex spaces as a combination of primitives.

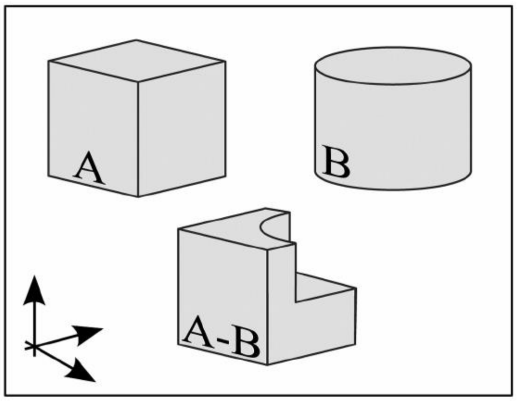


Figure 1.22 Constraints represented by volumetric primitives (Bowyer, 2014).

1.4.1.2 Constraint Evaluation Methods

The constraint evaluation phase is required to assess to relative configuration between the robot pose and the geometry of the constraint previously defined. In particular, we are interested in the proximity function between the end-effector and the constraint, in order to decide on the necessity and direction of the anisotropy. The majority of the evaluation methods are based on simple closed-form representations of the constraint. In a few other applications two key methods are deployed: *spatial partitioning and bounding volume hierarchies; feature-tracking algorithms* (Bowyer, 2014).

1.4.1.3 Constraint Enforcement Methods

In the constraint enforcement phase, the evaluated constraint geometry is converted into proper motion/force commands to give as input to the manipulator. In literature, a wide range of enforcement methods is proposed, each one having different features that makes it suitable for some applications over others. In the following, a brief overview of such methodologies is discussed. A comprehensive review can be found in (Bowyer, 2014).

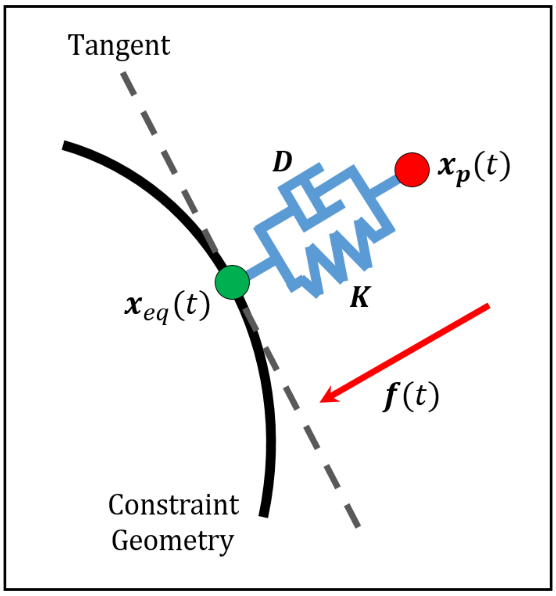


Figure 1.23 Enforcement method: simple function of constraint proximity.

Simple Functions of Constraint Proximity.

The simplest method described in the literature in based on a distance function (e.g., closest-point) between the constraint geometry and the robot's end-effector. The constraint force vector is modeled as a dynamic model, usually truncated to the first order. The mechanical behavior is equal to a spring-damper system (1st order, Figure 1.23), described by the following equation:

$$\boldsymbol{f} = K(\boldsymbol{x}_p - \boldsymbol{x}_{eq}) + D(\dot{\boldsymbol{x}}_p - \dot{\boldsymbol{x}}_{eq}) \qquad \text{(Eq. 1.1)}$$

where K, D are the stiffness and damping parameters of the system, $\boldsymbol{x}_p$is the 3D tool tip position and $\boldsymbol{x}_{eq}$ is the closest point on the constraint geometry.

Proxy and Linkage Simulation.

Proxies are virtual objects characterized by a given dynamic behavior. During the execution of the task, the proxy is virtually attached to the tool tip of the manipulator and the linkage is usually modeled as an elastic or viscoelastic model. When the robot enters forbidden regions, the proxy is blocked by the constraint surface, giving rise to a haptic force that encourages the user to leave such restricted region.

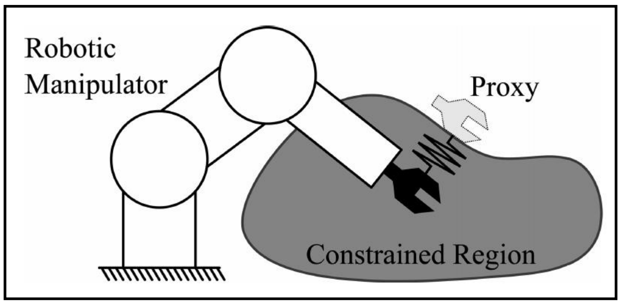


Figure 1.24 Proxy-based constraint. The linkage model is modeled as a spring (Bowyer, 2014).

Nonenergy Storing Constraints.

The main drawback of the previously discussed enforcement methods is that the effect of potential energy storing during the violation of forbidden regions. A sudden release of such stored energy may cause unwanted outcomes, like system instability and thus, harmful conditions for the operator. To overcome this problem in (Kikuuwe, 2008) a “simulated plasticity” model, based on Coulomb friction, was proposed as a way to dissipate energy during the tip-constraint interaction.

Potential fields.

Potential fields are one of the most widely used methodologies for real-time collision avoidance in robotic systems. Each point of the robot’s workspace s assigned a potential value, on the basis of the spatial distribution of both targets and obstacles. Each target is characterized by a low potential value, representing potential wells; obstacles are characterized by high potential values, representing a maximum of the potential function. The force acting on the end effector is proportional to the negative gradient of the potential field and encourages the manipulator to move toward minimum points (e.g., targets), while avoiding obstacles.

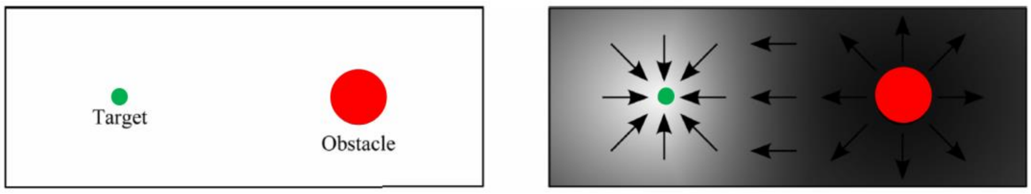


Figure 1.25 Constraints enforces as potential fields. The target acts as an attraction point, while the obstacle acts as a repulsion point (Bowyer, 2014).

1.5 Research problem

1.5.1 Adaptive Active Constraints

In the vast majority of robotic applications, active constraints geometries are computed via a separate process that precedes the actual execution of the task. In surgical applications, constraints are computed during the preoperative phase and are subsequently enforced during the intraoperative phase. The reliability and the exactness of the spatial distribution of such constraints relies on the hypothesis that the intraoperative scenario can be modeled via its preoperative counterpart: the environment is, thus, considered enough stable to deploy “static” active constraints. On the other hand, there is a number of applications, documented on literature (Ren, 2008), (Ryden, 2012), (Kwoh, 2013), that require the use of “dynamic” fixture as a consequence of the time evolution of the geometry that describes the surgical environment. Dynamic virtual fixture are currently used in "beating heart" surgery, where the natural contraction of the heart is tracked through the use of MRI-based imaging systems and the shape of the constraint is consequently updated. Minimally Invasive Surgery (MIS) applications involve the use of snake robot as endoscopic probe (Figure 1.26). In (Kwoh, 2013) a methodology is proposed to optimize the manipulator joint configuration on the basis of the actual anatomical constraints that are computed using real-time

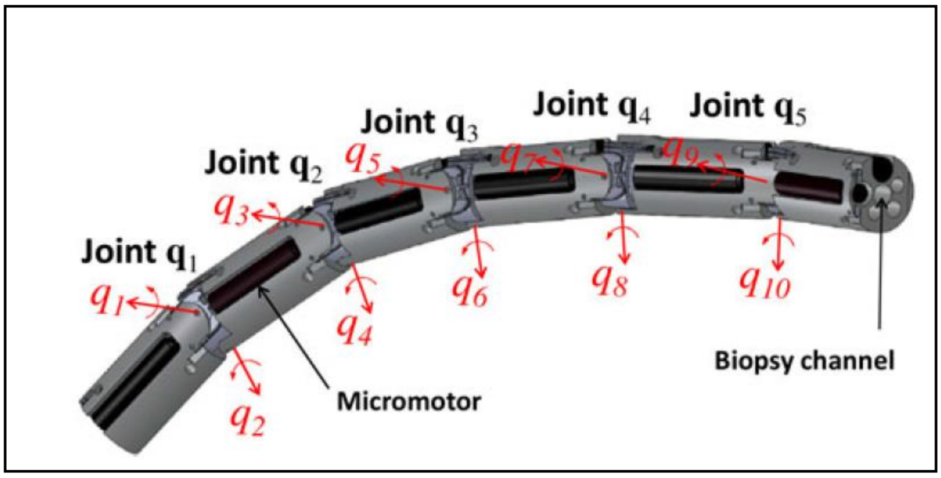


Figure 1.26 Snake robot deployed as endoscopic probe. The joint configuration is optimized on the basis of the evolution of soft tissues geometry (Kwoh, 2013).

proximity queries. In other applications, it might be useful to adapt the geometry of the constraint, or the assistance level provided by the haptic feedback, on the basis of the classification and interpretation of user commands. During the execution of a task, the presence of unwanted obstacles or unplanned target points might induce the human operator to intentionally move the end-effector against the constraint, as a result of changed environmental conditions. In this case, it would be beneficial to detect such intentional action and, consequently, update either the geometry or the stiffness of the constraint, in order to adapt to the new, unpredictable scenario. In (Passenberg, 2011), the authors investigate whether interaction forces can be used to distinguish between scenarios where human and assistant agree and where they disagree. A simple experimental design was implemented, consisting of two scenarios: a maze without obstacles (SC1) and a maze with obstacles (SC2) (Figure 1.27).

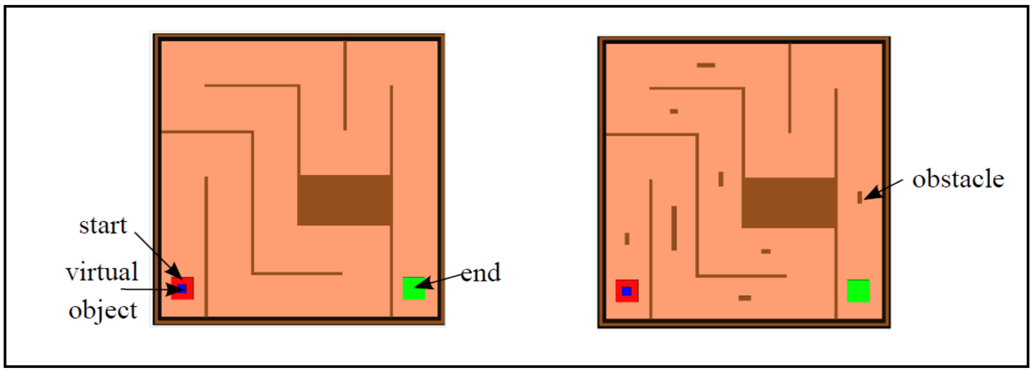


Figure 1.27 Virtual scenarios deployed for the experiments. On the left, it is shown a virtual maze without obstacles. On the right, it is shown a virtual maze with obstacles (Passenberg, 2011).

Each user was asked to avoid contact with walls and obstacles while moving as quickly as possible. The haptic feedback helped the user to move the virtual object along the desired path, from "start" to "end". However, in the second scenario, the presence of obstacles induced the user to act perpendicularly against the constraint to circumvent the obstacles. Results showed that the interaction force is a good estimator of the agreement level between the user and the assistant. In SC1, subjects did not act strongly against the constraint, as it was helpful. On the contrary, in SC2, higher forces were recorded, as the subject had to intentionally violate the constraint to avoid obstacles.

In (Li, 2003), authors propose a real-time method, based on Hidden Markov models, to detect and classify any change in the user intention, on the basis of the measured interaction forces between the constraint and the tool tip. The model was trained to recognize three actions: idling, following a path, and avoiding an obstacle. Results showed that the average accuracy of real-time recognition continuous among all subjects exceeded 90%.

1.5.2 Research Question

As discussed in the previous section, in a number of surgical applications, it might be beneficial to detect and interpret the user’s commands in order to provide some kind of adaptation in the assistance level provided by the active constraints. In general, active constraints help the operator in the execution of cooperative tasks by anisotropically regulating the tool motion, and enhancing the precision and accuracy of movements. However, when dealing with unstructured and/or dynamic environments, some events might occur that the robot controller is not able to properly interpret. For example, the user might want to depart from a constraint to circumvent unpredicted obstacles or to reach unplanned targets that lie within forbidden regions. In these cases, the constraint is perceived as a hindrance for the execution of the task and, consequently, its geometry and/or haptic strength should be modified accordingly. In order to modulate the assistance level, it is necessary to correctly detect and identify the current user’s intention on the basis of some kind of information (e.g. sensory information).
In this work, we decided to identify two rather general types of user’s actions:

- *Intentional Violation of active constraints;*
- *Unintentional Violation of active constraint.*

Intentional violations take place whenever the current action of the user is in disagreement with the purpose of the constraint, typically resulting from environmental sensing limitations of the robotic system. In this case, the constraint is felt as a hindrance, resulting in disturbing interaction forces at the tool tip. On the other hand, unintentional violations occur when the user shares the purpose of the constraint and accidental errors in the task execution are made. If we were able to reliably detect the occurrence of intentional violations, it would consequently possible to adapt the assistance level provided by the constraint (e.g. remove the constraint or weaken its magnitude).

In this project, we decided to investigate whether it is possible to distinguish between intentional and unintentional violations of active constraints, on the basis of the interaction force signals. To answer this question we trained and validated three different binary classifiers:

1. A classifier based on a continuous Hidden Markov Model with Gaussian output;
2. A classifier based on a feedforward Neural Network trained on the basis of statistical parameters describing the force signals;
3. A classifier based on a feedforward Neural Network trained on the basis of the energy distribution of the force signal across the different levels of the Wavelet decomposition.

As the classification should provide real-time adaptation of the assistance level, each model was evaluated in terms of:

- Minimum interval time required, after the violation has started, to obtain reliable classification of the intentions. In other words, we wonder which is the minimum amount of information, in terms of force signal, to correctly distinguish intentional violations from unintentional violations.
- Penetration depth into the constraint;
- Evolution of the classification performances over time.

2 Materials and Methods

In this chapter, we present the methodology and the experimental design adopted to research methods for the classification of “intentional” and “unintentional” violations of active constraints. The performance of Hidden Markov Models, taken from the literature, and Neural Networks, first proposed in this work, are investigated within this experimental framework. The chapter will be organized in four sections:

1. *Active Constraints*. In this section we discuss the definition, the evaluation and the enforcement methods we chose for the core active constraint formulation upon which the classifier was built.
2. *Experimental Design*. In this section we discuss the methodologies adopted to perform the experiments from which classifiable data was generated, describing the hardware and software setups, and the experimental protocol.
3. *Classification Methods.* In this section we discuss how we implemented Neural Network (NN)-based classifiers, and Hidden Markov Models (HMM)-based classifiers that operated in the experimentally produced active constraint violation data;
4. *Data Analysis*. In this section we discuss the performance indices chosen to evaluate the classification capabilities of each classifier.

2.1 Active Constraints

Active constraints are high-level control algorithms that are deployed in man-machine cooperative tasks to improve the performance of the task execution in terms of accuracy, precision and speed, while relying upon the constant supervision of the operator. As explained in the previous chapter, and widely discussed in (Bowyer, 2014), a generalized framework is introduced, composed of three main step: constraint definition, in which the geometry is described; constraint evaluation, in which the relative constraint-robot position are assessed; constraint enforcement, in which the geometry is transformed into position/force commands. Although constraints can be classified on the basis of their geometry and/or enforcement law, we chose to follow an alternative classification, proposed

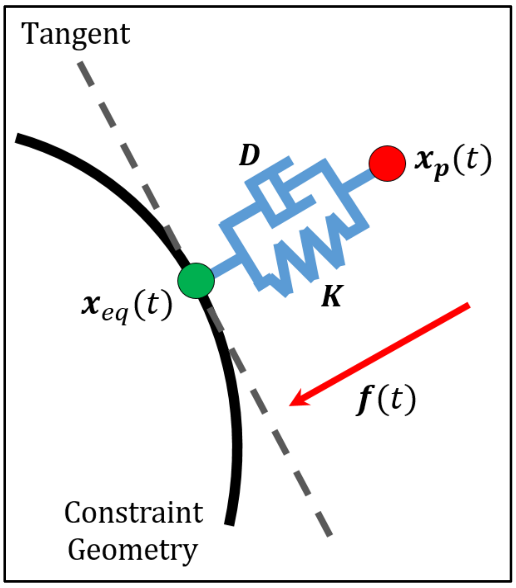


Figure 2.1 First order dynamic model for the calculation of the haptic force provided by the active constraint.

in (Bowyer, 2014) and (Abbott, 2007), that is based on the purpose of the constraint, namely which is its function with respect to the task to be executed. Two main classes have been identified:

- *Guidance Constraints*;
- *Regional Constraints*.

Guidance constraint help the operator to move along a preplanned trajectory, while regional constraints prevent the user from entering forbidden region of the workspace. We chose an enforcement methodology, based on a simple proximity function and a 1st order dynamic model (e.g., spring-damper model, Figure 2.1):

$$\boldsymbol{f}(t) = K\left(\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t)\right) - D\left(\dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}_{eq}(t)\right) \qquad \text{(Eq. 2.1)}$$

where $\boldsymbol{f}$ is the feedback force, $\boldsymbol{x}(t)$ is the Cartesian position of the end-effector, $\boldsymbol{x}_{eq}(t)$ is the closest point on the constraint geometry to the end-effector, and K, D are the stiffness and damping parameters of the active constraint, respectively. The computation of the $\boldsymbol{x}_{eq}(t)$ is carried out through a real-time closest-point algorithm, based on the Euclidean norm. This approach has been discussed in section 1.4.1.1.

2.1.1 Guidance Constraints

Guidance constraints assist the user in moving the robot manipulator along desired paths or surfaces in the workspace (Abbott, 2007). The enforcement of guidance constraints requires the definition of a set of reference directions that are used to describe the assistance level provided by the haptic feedback. In general, such directions are computed with respect to the local reference frame on the trajectory, yielding a perpendicular ($\boldsymbol{\pi}$) and a tangential ($\boldsymbol{\tau}$) direction of motion (Figure 2.2). The corresponding amount of perpendicular haptic force

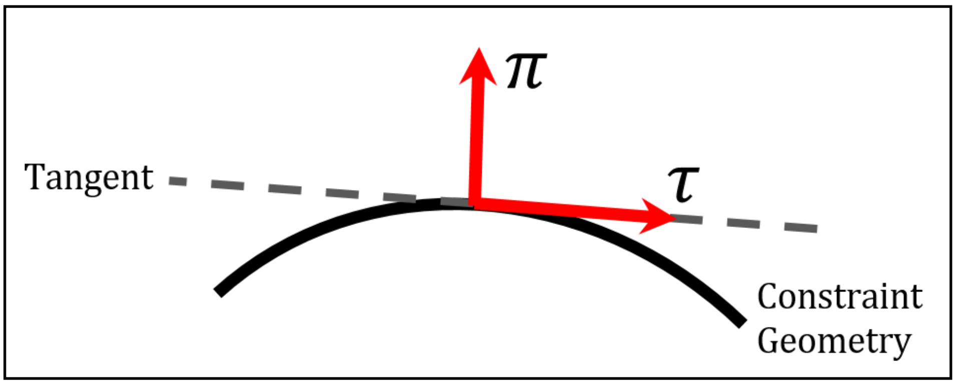
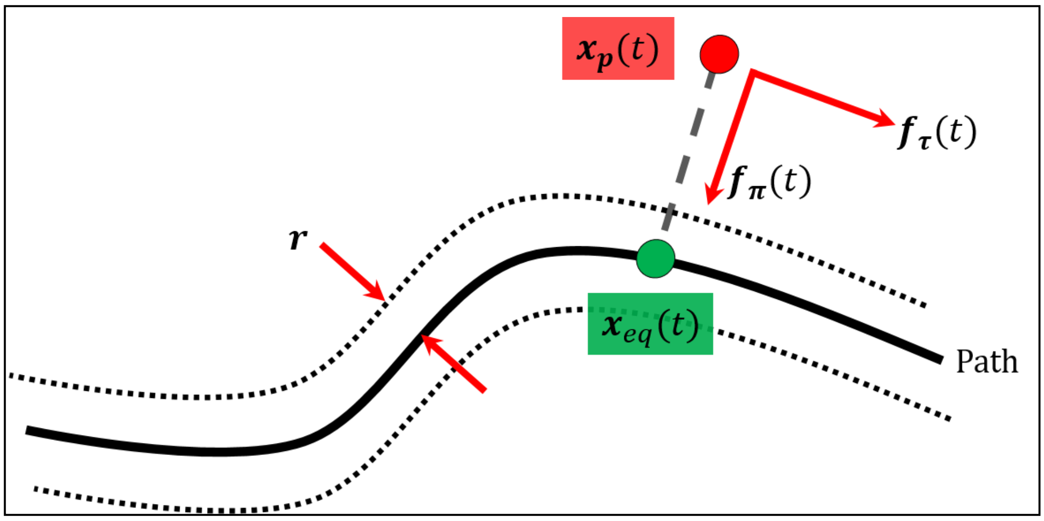


Figure 2.2 Local reference frame located onto the trajectory, yielding a perpendicular $\boldsymbol{\pi}$ and a tangential $\boldsymbol{\tau}$ direction.

compensates for any deviation of the robot from the path. The tangential component of the haptic force encourages the operator to move along the path and thus it identifies the preferred direction of motion. The assistance level provided is proportional to the magnitude of the force. The perpendicular component of the force $\boldsymbol{f}_{\boldsymbol{\pi}}$ can be defined such that a percentage of perpendicular deviation is admitted; namely, the constraint actively helps the operator when a certain



$\boldsymbol{x}_{\boldsymbol{p}}(t)$
$\boldsymbol{f}_{\boldsymbol{\tau}}(t)$
$\boldsymbol{f}_{\boldsymbol{\pi}}(t)$
$\boldsymbol{r}$
$\boldsymbol{x}_{\boldsymbol{eq}}(t)$
Path

Figure 2.3 Guidance Constraints define a tube-shaped region (dotted lines), whose longitudinal axis is the correct path (black line). Within this region, the haptic force is zero, as a certain degree of perpendicular error is permitted.

threshold is overcome. The value that sets this degree of deviation error, defines a tube-shaped region, whose longitudinal axis corresponds to the desired path (Figure 2.3). The mathematical relationship that describes the guidance constraint enforcement is the following:

$$\boldsymbol{f} = \boldsymbol{f}_{\boldsymbol{\tau}} + \boldsymbol{f}_{\boldsymbol{\pi}} \qquad \text{(Eq. 2.2)}$$

Where $\boldsymbol{f}_{\tau}$ and $\boldsymbol{f}_{\pi}$ are the tangential and perpendicular components of the haptic force. These components are defined as:

$$\boldsymbol{f}_\pi(t) = K_\pi\big(\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t) - r\boldsymbol{\pi}\big) + D_\pi\left(\dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}_{eq}(t)\right) \qquad \text{(Eq. 2.3)}$$

$$\boldsymbol{f}_\tau(t) = K_\tau\big(\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t) - r\boldsymbol{\tau}\big) + D_\tau\left(\dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}_{eq}(t)\right) \qquad \text{(Eq. 2.4)}$$

If $|\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t)| < r$.

$\boldsymbol{x}(t)$ is the Cartesian position of the end-effector; $\boldsymbol{x}_{eq}(t)$ is the closest point on the constraint geometry to the end-effector; r is the variable that represent degree of perpendicular error allowed; $\boldsymbol{\pi}$ is the unit vector that identifies the orthogonal direction; K_π, D_π and K_τ, D_τ are the stiffness and damping parameters of the system in both directions, respectively.

An example of guidance constraints enforcement can be found in (Burghart, 1999). In this study, guidance constraints are used to improve the outcomes of craniofacial surgical intervention, by restricting the surgeon's hands motion while guiding a surgical saw attached to the manipulator. In the planning step, the path is split into segments, around which, two concentric cylinders are constructed. Within the inner cylinder the surgeon is allowed to guide the saw without constraints. As soon as he/she moves the tool into the outer cylinder, an increasing force has to be applied to continue into the desired direction. Finally, if the surgeon tries to push the tool further beyond the outer cylinder, the robot prohibits any further movement of the surgical saw into the forbidden zone.

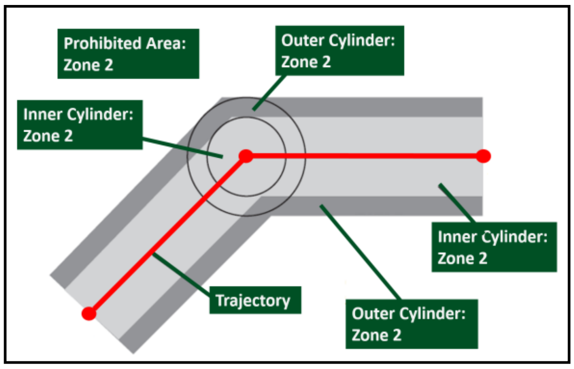


Figure 2.4 Guidance constraint enforcement for enhanced craniofacial surgery.

2.1.2 Regional Constraints

Regional constraints, also called “forbidden-region” constraints, prevent the robot form entering into restricted regions of the workspace. They have an on/off nature, such that they have no effect on the robot when it is outside of the forbidden region (Abbott, 2007). In general, the haptic force generated by the presence of the constraint acts perpendicularly to the surface (Figure 2.5). Unlike guidance constraints, there is not a tangential component of the force, because no preferred directions of motion are defined. Similarly to guidance constraints, we chose a 1st order dynamic model as enforcement method:

$$\boldsymbol{f}(t) = K\left(\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t)\right) - D(\dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}_{eq}(t)) \qquad \text{(Eq. 2.5)}$$

$\boldsymbol{x}(t)$ is the Cartesian position of the end-effector, $\boldsymbol{x}_{eq}(t)$ is the closest point on the constraint geometry to the end-effector, and K, D are the stiffness and damping parameters of the system, respectively.

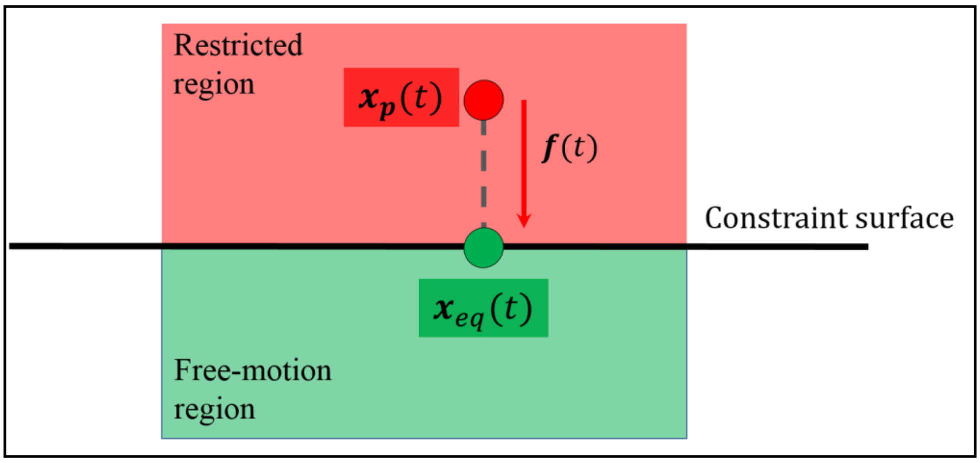


Figure 2.5 Forbidden-region constraints enforcement. The haptic force acts perpendicularly to the constraint surface, encouraging the human operator to place the tool tip safely into the safe region.

An example of regional constraint deployment is discussed in (Park, 2001): medical images were used to align virtual planes between an artery that requires dissection and the chest wall to which it was attached. The tool was constrained within a safe region, enabling the surgeon was assisted in accurate resection of the artery, for sternotomies or bypass graft procedures (Park, 2001).

2.2 Experimental Design

In this section, we discuss the architecture of the overall system that composes the experimental design used to generate classifiable constraint violation data:

- The *software setup*, that is composed of several Graphical User Interfaces (GUIs) to set, control, and visualize the tasks;
- The *hardware setup*, that is composed of a control console, a display to provide visual feedback during the execution of the task, and the haptic interface;
- The *experimental protocol*, that describes the guidelines followed to create two subsets of ten path-following and target-reaching tasks, and the instructions for the execution of the tasks, customized to obtain reliable and reproducible results;

2.2.1 Software Setup

The software setup is composed of three main sections: the “settings” interface, the control system and the “task” interface. In the first section it is possible to choose and customize the task to be executed, according to the goal of the project; in the second section a closed-loop system is implemented for the runtime control of the haptic interface; the last section is characterized by a graphical interface that is shown to the subjects while executing the task.

2.2.1.1 Settings GUI

The creation of customized tasks is accomplished by means of dedicated Graphical User Interfaces (GUIs), which allow the operator to set all the meaningful parameters according to the goal of the project. Three interfaces were programmed:

1. "*Workspace settings*" GUI. This interface is designed to set the workspace boundaries and the required safety limitations;
2. "*Path-following setting*" GUI. This interface is designed to customize the "path-following" tasks to produce data for the evaluation of classification performance in the case of "guidance" constraints;
3. "*Target-reaching setting*" GUI. This interface is designed to customize the "target-reaching" tasks to produce data for the evaluation of the classification performances in the case of "forbidden region" constraints.

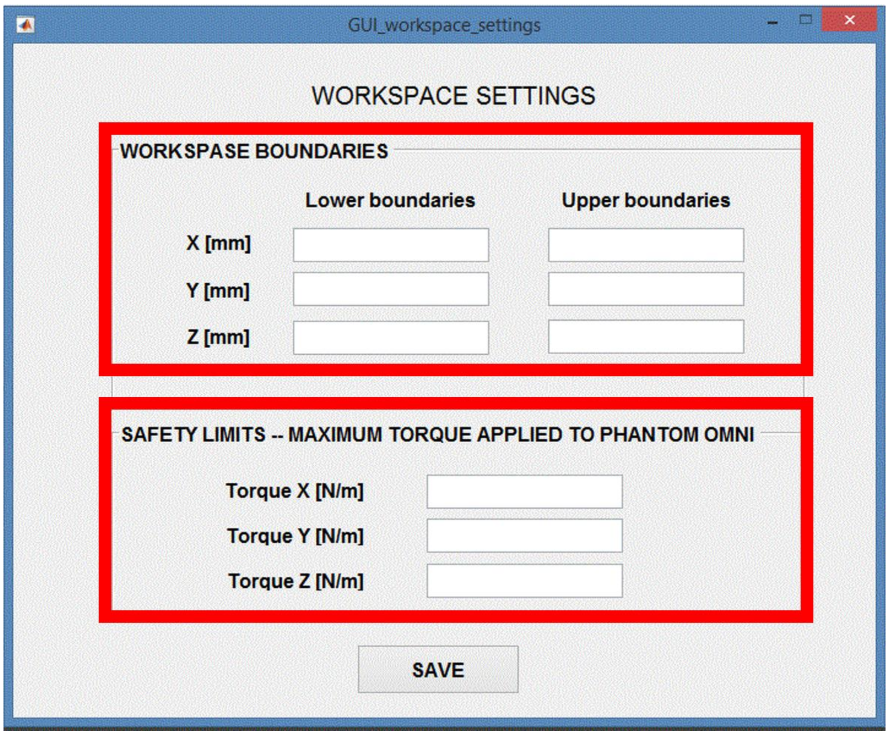


Figure 2.6 "Workspace settings" GUI.

2.2.1.1.1 “Workspace settings” GUI

The “*Workspace Settings*” GUI (Figure 2.6) is designed to set the workspace boundaries and the safety limits. Workspace boundaries prevent the haptic interface from assuming any joint configuration close to singularities, while safety limitations represent the maximum torque that can be applied to the joints of the manipulator. The user can set up to nine parameters:

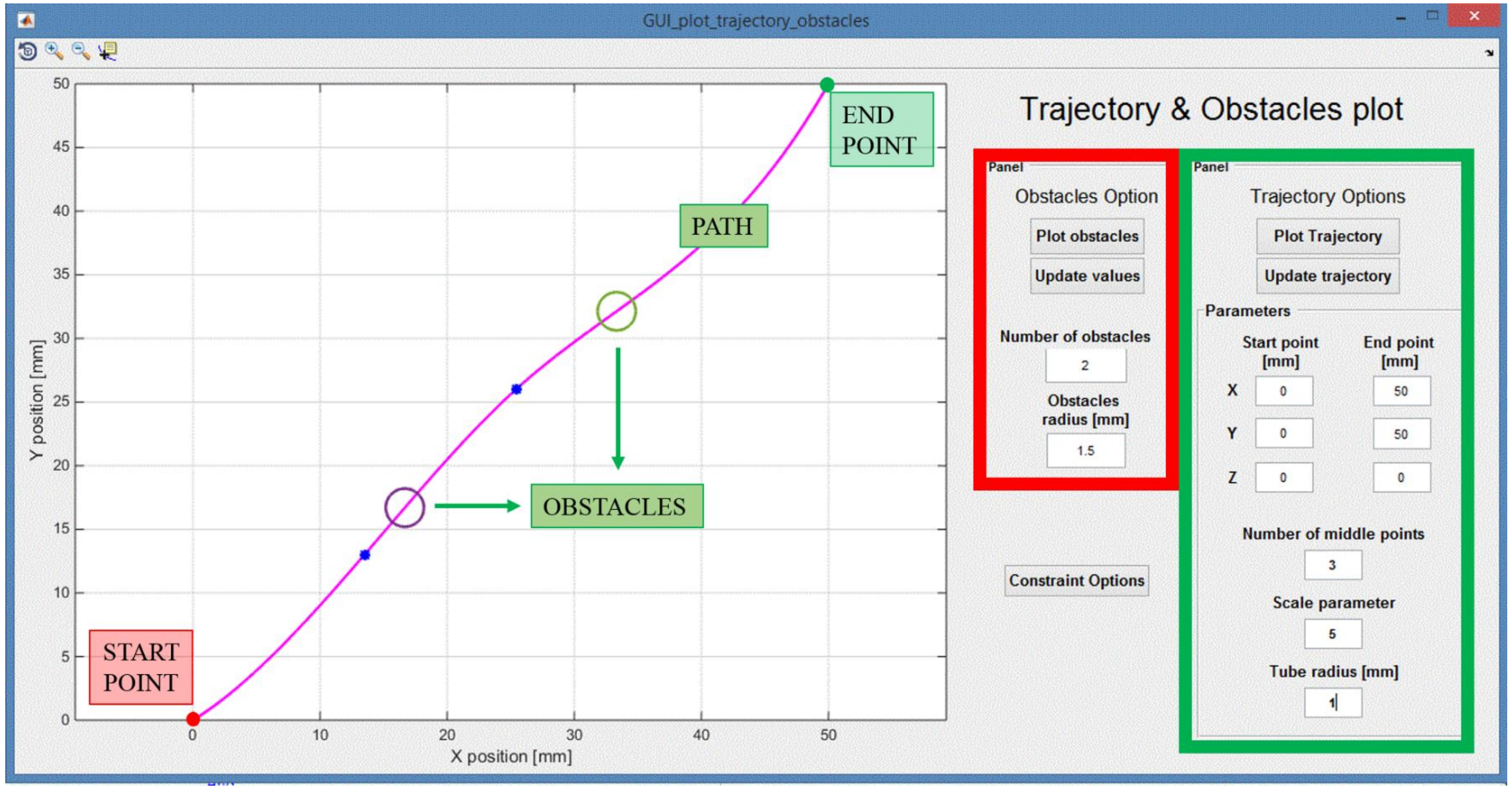
- Six parameters set the position boundaries along the three Cartesian directions(X, Y, Z). These limitation are chosen in order to avoid singularities that occur when the configuration is close to the physical limits of the haptic interface;
- The maximum torques that can be safely enforced to each actuator of the haptic master.

2.2.1.1.2 “Path-Following settings” GUI

The “Path-following settings” Graphical User Interface (Figure 2.7) is designed to set the geometrical parameters that characterize the “path-following” tasks, for the evaluation of guidance constraints.

The interface is composed of several subpanels, by which it is possible to set different parameters:

- Update the coordinates (X, Y, Z) of the boundary points of the trajectory, labelled in Figure 2.7 as “Start point” and “End point”;
- Set two parameters, “number of middle points” and “scale”, which allow for the introduction of some degree of variability in the definition of the trajectory. The first variable is related to the number of intermediate, equally spaced points that are sampled along the straight line linking the “Start” and the “End” points. The second variable introduces a degree of variability in the positioning of these points by scattering them around,



Trajectory & Obstacles plot

Number of obstacles: 2

Obstacles radius [mm]: 1.5

Parameters

	Start point [mm]	End point [mm]
X	0	50
Y	0	50
Z	0	0

Number of middle points: 3

Scale parameter: 5

Tube radius [mm]: 1

Figure 2.7 "Path-following settings" GUI for the customization of "path-following" tasks. In figure the trajectory, its boundary points “start” and “End”, and the circular obstacles that are placed along the path. The features of the task can be updated through the two subpanels on the right hand side. The red-framed panel allows for the modification of the radius and number of obstacles. The green-framed panel allows for the modification of the trajectory characteristics.

according to a 2D uniform statistical distribution. These points are then used to calculate the trajectory according to a spline-based interpolation (Figure 2.8);

- Set the number and radius of obstacles to lay along the path. Obstacles are depicted in (Figure 2.7) as circles;

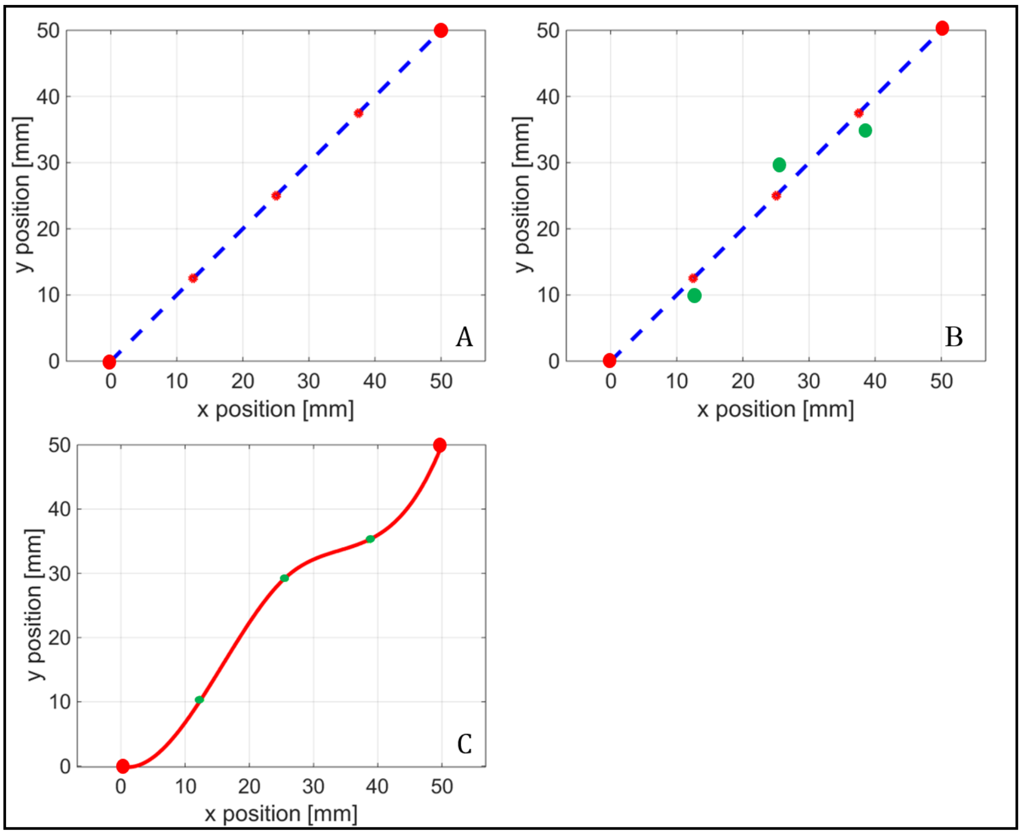


Figure 2.8 Trajectory computation. A number of intermediate points is sampled along the straight line linking the boundary points (red points, A). These points are scattered around according to a uniform statistical distribution (green points, B). The trajectory is the result of a spline-based interpolation (red line, C).

- Set the radius r (Eq. 2.3) of the guidance constraint, namely the distance to exceed when leaving the path to feel the effect of the constraint;
- "Plot" window, in which the effects of the intermediate updates of the parameters are displayed.

- “Plot” and “Update” buttons to control the setting process.

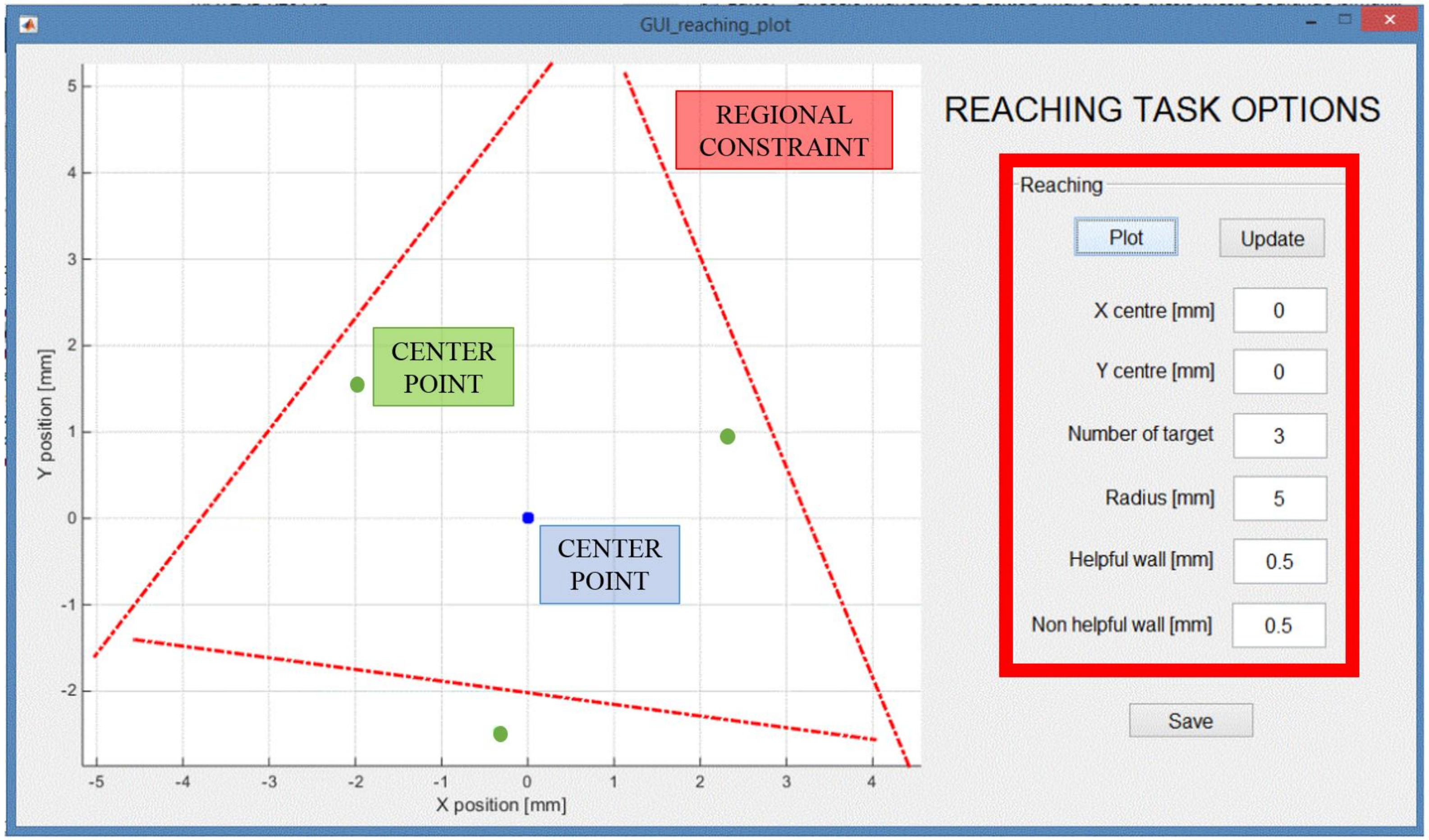
2.2.1.1.3 Target-Reaching settings GUI

The second Graphical User Interface allows the user to customize the “target-reaching” task. Figure 2.9 shows an overview of the window through which it is possible to modify the following elements:

- The coordinates (X, Y) of the center point. The updated values are typed in the “X center” and “Y center” editable textboxes;
- The number of targets, that characterizes the task, can be updated in the “number of targets” editable textbox;
- The dispersion radius of the targets is the measure of the distance between the center point and each target. Its value is typed in the “radius” editable textbox and identifies a circumference from which the coordinates of the targets are sampled. The targets are equally spaced from each other, i.e.

$$\Delta\alpha = \frac{2\pi}{n} \tag{Eq. 2.6}$$

  where $\Delta\alpha$ is the angular distance and n is the total number of targets.
- The distance of the regional constraints enforced. Each target is associated with one regional constraint. Each fixture can either be helpful or unhelpful: in the first case, the fixture is placed beyond the target, effectively helping the user accomplishing the task without hindering it. In the second case, the fixture lies before its associated target, hindering the reaching goal. For any number n of targets, the nature of the constraint is statistically distributed with probability of 50%. Both parameters can be set in the “Helpful Wall” and “Not Helpful Wall” editable textboxes;
- “Plot”, “Update” and “Save” buttons to manage the setting process.



REACHING TASK OPTIONS

Reaching

X centre [mm]	0
Y centre [mm]	0
Number of target	3
Radius [mm]	5
Helpful wall [mm]	0.5
Non helpful wall [mm]	0.5

Figure 2.9 "Target-reaching settings" GUI. This GUI allows for the customization of the target-reaching task for the evaluation of forbien-region

2.2.1.2 Control System

The system that supports the execution of the tasks implements a Cartesian control that, based on position data of the tool tip of the manipulator, calculates the amount of haptic force required to help the user to accomplish the task. A scheme of the control system is shown in Figure 2.10.

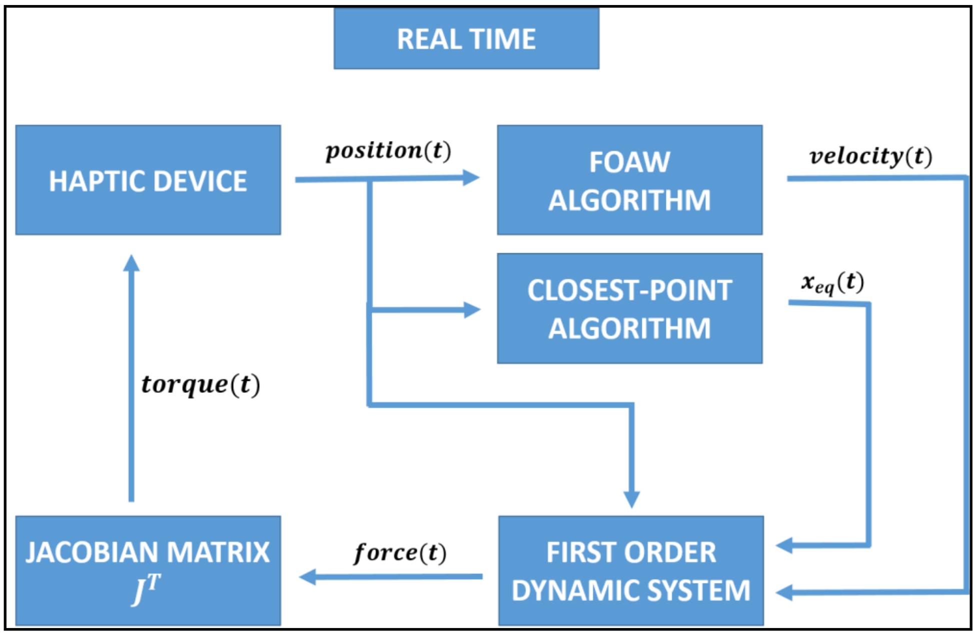


Figure 2.10 Overview of the control system.

The Cartesian control system is composed by the following blocks:

- The haptic interface;
- A derivative step that calculates the Cartesian velocity of the tool tip by applying a First Order Adaptive Windowing on the raw position data. Such an approach, though introducing a time delay, provides an intrinsic reduction of the noise affecting the position measurement and an improved controller stability;
- A block that embeds a closest-point algorithm to calculate the value of the parameter $x_{eq}(t)$ (Eq. 2.1), necessary to modulate the direction and

magnitude of the feedback force. Its value is computed by applying a runtime proximity function based on the Euclidean norm between the current position of the tool tip, and the closest point lying on the constraint.

- A block that calculates the modulus and the direction of the Cartesian feedback force that has to be applied to the tool tip of the manipulator;
- A block that implements the runtime calculation of the Jacobian matrix J^T, based on the current values of the joints position, read out from the encoders of the master. The matrix is used to calculate the values of the torques commands to feed back to the haptic interface;
- A real time block that computes the relationship between the simulation clock and the CPU clock.

2.2.1.2.1 FOAW algorithm for velocity calculation

The calculation of the Cartesian velocity at the end-effector of the haptic interface was carried out by implementing a First Order Adaptive Windowing on raw position data (Janabi-Sharifi, 2000). The choice to implement such a methodology is due to the fact that the haptic device was characterized by a noisy velocity signal that caused high frequency vibration, influencing the stability of the system.

Let $x(t)$ be a position signal that is sampled with period T. The measurement of the position $x(t)$ is corrupted by noise due to quantization (encoders, digital converters), or to other sources. The measurement model is the sum of the real position value $x(t)$ and some uncorrelated noise, i.e.

$$y(t) = x(t) + c(t) \qquad \text{(Eq. 2.7)}$$

where $y(t)$ is the measured position, $x(t)$ is the true position, and $c(t)$ is the noise that corrupts the signal. In the absence of additional information, the error can be considered to have a zero mean bounded uniform distribution (e.g. the case of pure quantization).

The problem is to find a good estimator $\hat{v}(t)$ of the true velocity $v(t)$,

$$v(t) = \frac{dx(t)}{dt} \qquad \text{(Eq. 2.8)}$$

A good estimator should be able to both reject the noise and minimize calculation delay that could compromise the stability of the closed-loop system.

One of the classical approaches is the Finite Difference Method (FDM), based on the Euler approximation of the derivative operator, yielding the following estimate:

$$\hat{v}(t) = \frac{y_k - y_{k-1}}{T} = \frac{x_k - x_{k-1}}{T} + \frac{c_k - c_{k-1}}{T} \qquad \text{(Eq. 2.9)}$$

Where k is the current time instant and T is the sample time. This method asymptotically breaks down at high sampling rates, when high time resolution is needed for feedback control. As the sampling time T becomes smaller, the position increments decrease, but the noise component does not and is correspondingly amplified. The First Order Adaptive Window algorithm introduces an implicit down-sampling step, by online adjusting the length n of the window within which to apply the Finite Difference method. This operation is equivalent to averaging the last n velocity estimates,$\hat{v}_k$, $\hat{v}_{k-1}$,…, $\hat{v}_{k-n}$, with $\hat{v}_i$ obtained from the FD method, i.e.

$$\hat{v}_k = \frac{1}{n}\sum_{j=0}^{n-1} \hat{v}_{k-j} = \frac{y_k - y_{k-n}}{nT} \qquad \text{(Eq. 2.10)}$$

As the application of large windows introduces time delay that might compromise the stability of the system, the width of the window should be properly adjusted according to the characteristics of the measured position signal $y(t)$: large windows should be applied at slow velocities and short windows should be applied at faster velocities. Noise reduction and precision put a lower bound on the window size, while reliability provides an upper limit for the window length. In other words, a criterion should be established to determine whether the slope of a straight line reliably approximates the derivative of a signal between two samples $x_k,\ x_{k-n}$. The selected criterion is then used to find the longest window which satisfies the accuracy requirement, solving a min–max problem.

The FOAW method is based on finding a window of length n, where $n = \max\{1, 2, 3, \dots\}$, such that

$$|y_{k-i} - \bar{y}_{k-i}| \leq d, \ \ \forall i \in \{1, 2, \dots, n\} \qquad \text{(Eq. 2.11)}$$

where y_{k-i} is the measured signal at time instant $(k - i)$; $\bar{y}_{k-i}$ is the approximation of y_{k-i} calculated by means of the straight line that linearly interpolates the boundary points of the window y_k and y_{k-n}; d is the peak of the noise superimposed to the signal, i.e.

$$d = ||c_k||_\infty \quad \forall k \qquad \text{(Eq. 2.12)}$$

The steps of the algorithm are the following (Janabi-Sharifi, 2000):

1. Set $i = lower\ bound$;
2. Calculate the straight line that interpolates the points y_k, y_{k-i};
3. Check whether the line passes through all the points inside the window, within the uncertainty band of each point.

4. If so, and if $i < upper\ bound$, set $i = i + 1$ and GOTO step 3. Else return the last estimate $n = i$.

After the optimal window size has been found, the slope of the straight line gives the estimation of the velocity $\hat{v}_k$. To improve the robustness of the estimation a Least Squares approximation was use instead of the slope value of the interpolating straight line, giving the following estimate:

$$\bar{v}_k = \frac{n \sum_{i=0}^{n} y_{k-i} - 2 \sum_{i=0}^{n} i\, y_{k-i}}{Tn(n+1)(n+2)/6} \qquad \text{(Eq. 2.13)}$$

The application of the Least Squares provides additional smoothing, preventing undesired overshoots due t fast changes in the window size.

For the control system implemented the length of the window was bounded within the interval $[50, 100]$ samples, while the noise norm of the position data was set to $d = 0.6mm$.

2.2.1.2.2 Closest-point algorithm

The computation of the haptic force to help the user enhancing the task performances requires some knowledge about the relative geometry between the current position of the tool tip $\boldsymbol{x}(t)$ and the constraint geometry. Such computation is carried out by means of a proximity function based on the runtime calculation of the Euclidean norm:

$$d(t) = \sqrt{[x_p(t) - x_c(t)]^2 + [y_p(t) - y_c(t)]^2} \qquad \text{(Eq. 2.14)}$$

where d is the Euclidean norm, (x_p, y_p) are the 2D Cartesian coordinates of the tool tip and (x_c, y_c) are the Cartesian coordinates of the generic point belonging to the enforced constraint. The closest-point algorithm computes the coordinates (x_{eq}, y_{eq}) that minimize the value of the Euclidean norm:

$$\left(\boldsymbol{x}_{eq}(t), \boldsymbol{y}_{eq}(t)\right) = \underset{x_c, y_c}{\operatorname{argmin}}(d(t)) \qquad \text{(Eq. 2.15)}$$

At each time step, $\boldsymbol{x}_{eq}(t), \boldsymbol{y}_{eq}(t)$ are computed by comparing the current tip position (x_c, y_c) to the discretized points that describe the constraint geometry, and used to provide the right amount of feedback force.

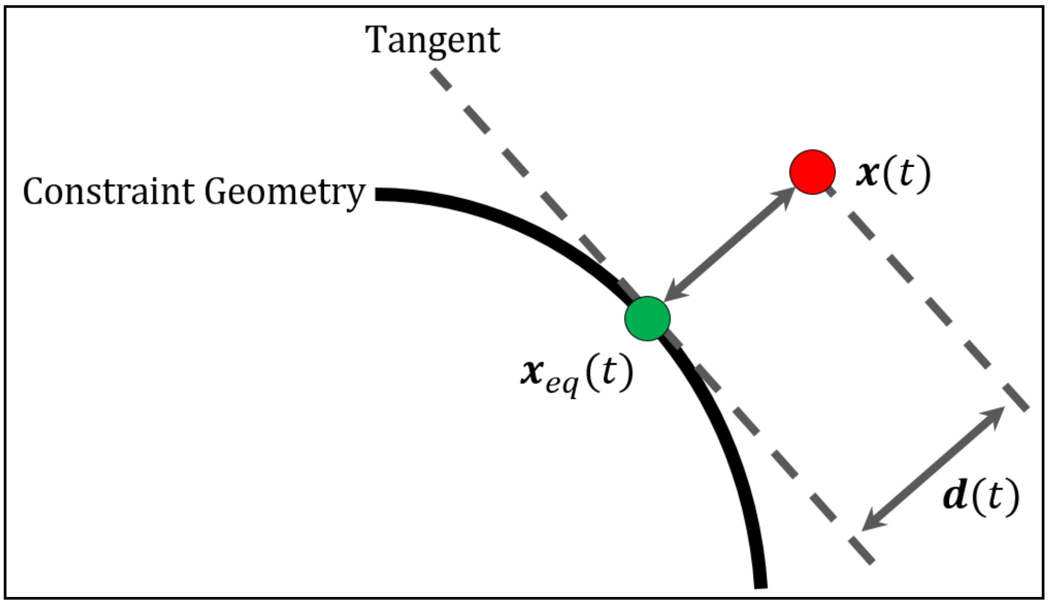


Figure 2.11 Closest-Point algorithm for the calculation of the magnitude of the feedback force: $\boldsymbol{x}_{\boldsymbol{eq}}\ (\boldsymbol{t})$ is the point that belongs to the constraint geometry, that minimizes the distance $\boldsymbol{d}(\boldsymbol{t})$.

As shown in Figure 2.11, the point that satisfies (Eq. 2.14) identifies a tangent to the constraint profile that is perpendicular to the straight line passing through such point and the point that marks the current position of the tool tip.

2.2.1.2.3 1st Order Dynamic Model

The computation of the magnitude and direction of the feedback force to apply to the haptic master is based on the choice of a "constraint enforcement" method. As described in section 1.4.1.3, there is a wide variety of methods described in literature for enforcing active constraints, each one having attributes that lend themselves to certain applications over others (Bowyer, 2014). According to the goal of this project, a "Simple Function of Constraint Proximity" was selected, as it met all the requirements of the system, such as the permission of small magnitude penetration while relying on a simple representation of the constraint itself. The behavior of the active constraint is a linear model with derivative terms, that yields a 1st order differential equation that represents a mechanical spring-damper system.

The mechanical behavior of *guidance constraints* is described by equations (Eq. 2.2) (Eq. 2.3) (Eq. 2.4): the total haptic force is given by the sum of a perpendicular component $\boldsymbol{f_\pi}$ that corrects for any deviations from the path, and the parallel component $\boldsymbol{f}_\tau$ that encourages the user to move along the path. In our application, the parallel component was set equal to zero:

$$\boldsymbol{f}_\tau = 0 \qquad \text{(Eq. 2.16)}$$

$$\dot{\boldsymbol{x}}_{eq}(t) = 0 \qquad \text{(Eq. 2.17)}$$

The total haptic force is thus defined as:

$$\boldsymbol{f}(t) = K\big(\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t) - r\boldsymbol{\pi}\big) + D\dot{\boldsymbol{x}}(t) \qquad \text{(Eq. 2.18)}$$

where $\boldsymbol{x}(t)$ is the position of the tool tip, $\boldsymbol{x}_{eq}(t)$ is its closest point that belongs to the constraint geometry, K and D are the stiffness and the damping parameters of the system.

Regional Constraints are described by equation (Eq. 2.5). By applying (Eq. 2.20), the mechanical behavior of this type of constraint is defined as:

$$\boldsymbol{f}(t) = K\left(\boldsymbol{x}(t) - \boldsymbol{x}_{eq}(t)\right) + D\dot{\boldsymbol{x}}(t) \qquad \text{(Eq. 2.19)}$$

where $\boldsymbol{x}(t)$ is the position of the tool tip, $\boldsymbol{x}_{eq}(t)$ is its closest point that belongs to the constraint geometry, K and D are the stiffness and the damping parameters of the system.

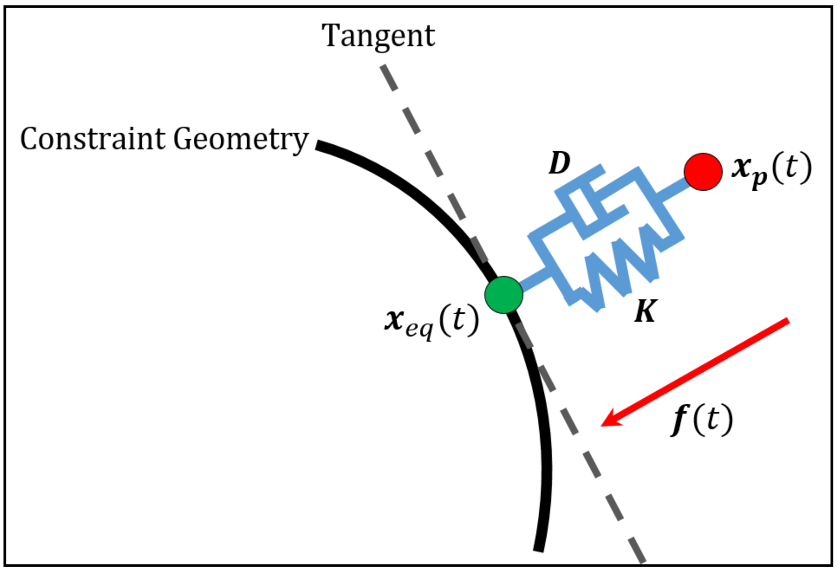


Figure 2.12 Spring-damper mechanical model used to represent the behavior of the constraint.

The value and sign of the stiffness K are important to determine the dynamic behavior of the system: any change in the magnitude of the stiffness affects how strongly the constraint acts upon the haptic master: an increase in its value requires exerting a higher force to reach the same penetration depth. The sign of the stiffness determines whether the behavior of the fixture is "attractive" or "repulsive": in the first case, motion toward the constraint is encouraged, while in the second case the tool tip is pushed away from it. A representation of both scenarios is shown in Figure 2.13.

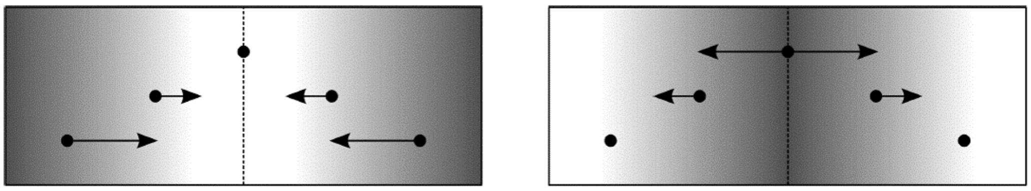


Figure 2.13 Attractive (left) and repulsive (right) constraints.

In our system, a positive value of stiffness was set, yielding a compound attractive behavior.

2.2.1.2.4 Jacobian Matrix

The Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function. Suppose that:

$$F : \mathbb{R}^n \rightarrow \mathbb{R}^m \qquad \text{(Eq. 2.20)}$$

is a function which takes as input the vector $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ and produces as output the vector $\boldsymbol{F}(\boldsymbol{x}) \in \mathbb{R}^{\boldsymbol{m}}$. Then the Jacobian matrix **J** of **F** is a mxn matrix defined as:

$$J = \frac{d\boldsymbol{F}}{dx} \qquad \text{(Eq. 2.21)}$$

Its matrix representation is the following:

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix} \tag{Eq. 2.22}$$

In the field of robotics the Jacobian matrix relates joint velocities to Cartesian velocities of the end effector of a generic robotic manipulator (Craig, 2005). Its value is dependent on the current angular position of the joints, yielding at each time step the geometrical relationship between the joint and Cartesian spaces. Thus, the dimensionality of the Jacobian matrix is related to the geometrical characteristics of the robotic manipulator: the number of rows equals the number of degrees of freedom in the Cartesian space being considered; the number of columns is equal to the number of joints of the device. In the general case in which the robot is characterized by six Degrees of Freedom and k joints, the Jacobian matrix has $(6 \times k)$ dimension, yielding the following relationship:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = J \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_k \end{bmatrix} \tag{Eq. 2.23}$$

where v_x, v_y and v_z are the Cartesian components of the linear velocity of the tool tip, ω_x, ω_y and ω_z are the Cartesian components of the angular velocity, and $\dot{q}_1$,…, $\dot{q}_k$ are the joint velocities, either linear or angular, depending on the nature of the joint, prismatic or rotational, respectively. The Jacobian matrix is not only an important parameter for kinematic transformations, but also when dealing with dynamic variables. As shown in Figure 2.14, it is possible to establish a direct proportionality relationship between the Cartesian force acting on the end effector and the torque acting on each joint whose compound action results in such a force. The relationship is the following:

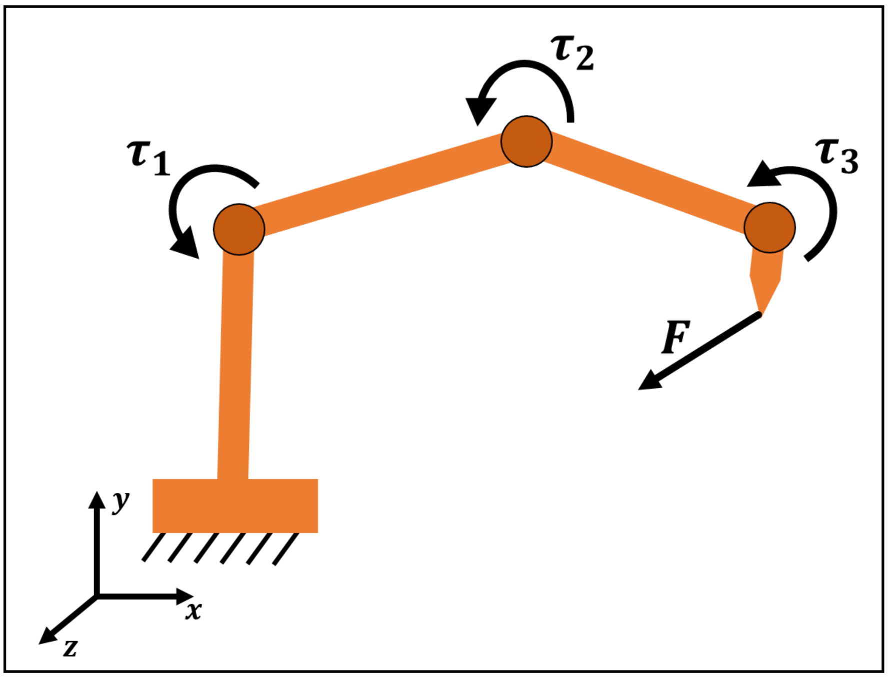


Figure 2.14 Representation of the relationship of the Cartesian force felt at the end-effector of the manipulator and the torques applied to its joints.

$$\boldsymbol{\tau} = J^T \boldsymbol{F} \qquad \text{(Eq. 2.24)}$$

where $\boldsymbol{\tau}$ is the torque applied at joints level, $\boldsymbol{J}$ is the Jacobian matrix and $\boldsymbol{F}$ is the resulting Cartesian force felt at the tip of the haptic device. In our system, a custom block calculates the runtime value of $\boldsymbol{J}$, that is then used to obtain the torques vector corresponding to the Cartesian force provided by the active constraint.

2.2.1.2.5 Real Time Block

The real time block is the element of the control system that gives as output the relationship between the simulation clock and the CPU clock. According to this

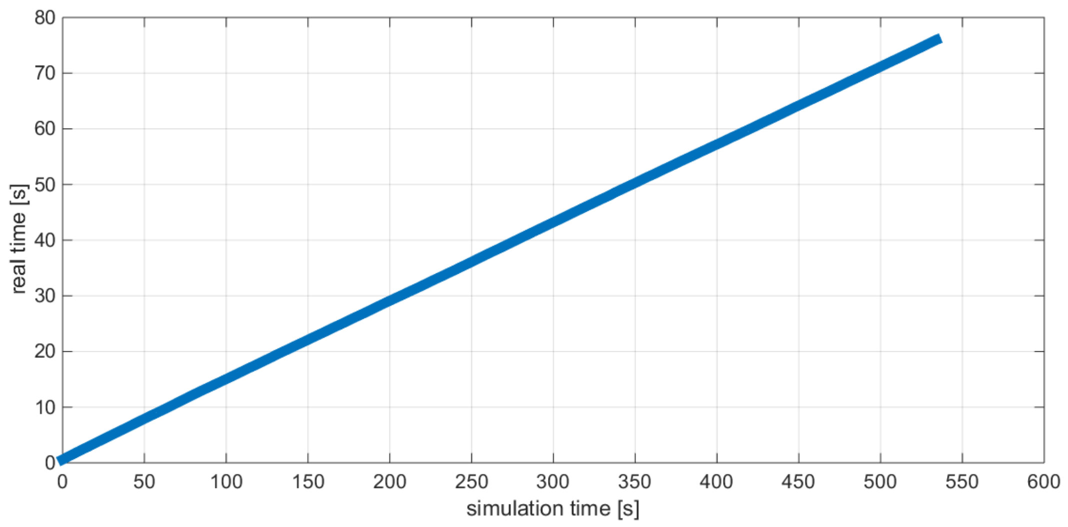


Figure 2.15 Linear relationship between the simulation time and the CPU time.

relationship, it is possible to calculate the real sampling interval of the signals recorded. As shown in Figure 2.15, such relationship is characterized by a highly linear behavior. The estimation of the linear transform is carried out by using a linear Least Squares approach that aims at computing the equation of the straight line that best approximates the data. Given a linear overdetermined system:

$$A\boldsymbol{x} = b \qquad \text{(Eq. 2.25)}$$

where A is the system matrix, $\boldsymbol{x}$ is the vector of the unknown parameters and b is the vector of constant values, the linear Least Squares solution is the following:

$$\overline{\boldsymbol{x}} = (A^T A)^{-1} A^T b \qquad \text{(Eq. 2.26)}$$

In our specific case, the application of this method provides the estimation of the angular coefficient m and offset q of the generic straight line:

$$y_{real} = mx_{sim} + q \qquad \text{(Eq. 2.27)}$$

where x_{sim} is the vector containing the simulation time and y_{real} is the vector containing the CPU time. By arranging all the known and unknown parameters into the system, we obtain:

$$\begin{bmatrix} x_{sim}(1) & 1 \\ \vdots & \vdots \\ x_{sim}(n) & 1 \end{bmatrix} \begin{bmatrix} m \\ q \end{bmatrix} = \begin{bmatrix} y_{real}(1) \\ \vdots \\ y_{real}(n) \end{bmatrix} \qquad \text{(Eq. 2.28)}$$

The output of the algorithm are the unknown m and q. The value of the angular coefficient is further exploited to compute the value of the real sample time, given the simulation sample time:

$$\Delta T_{real} = m\Delta T_{sim} \qquad \text{(Eq. 2.29)}$$

2.2.2 Hardware setup

The hardware setup is composed of three main elements:

- The *control console*, by which the operator supervises the execution of the trials (Figure 2.17);
- The *display*, that provides visual feedback while the subject is either following a trajectory or reaching a target (Figure 2.17);
- The *haptic interface* Phantom OMNI® (Sensable Technologies).

An overview of the system is shown in Figure 2.17. and Figure 2.16

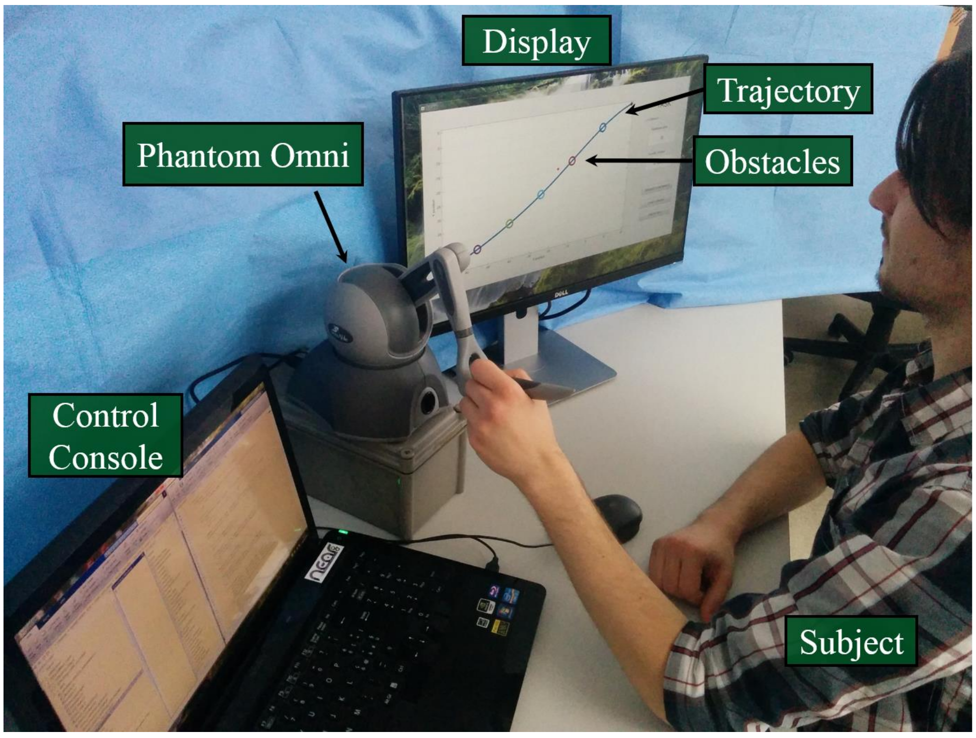


Figure 2.17 The Hardware Setup is composed of three blocks: the control console, the display and the haptic interface.

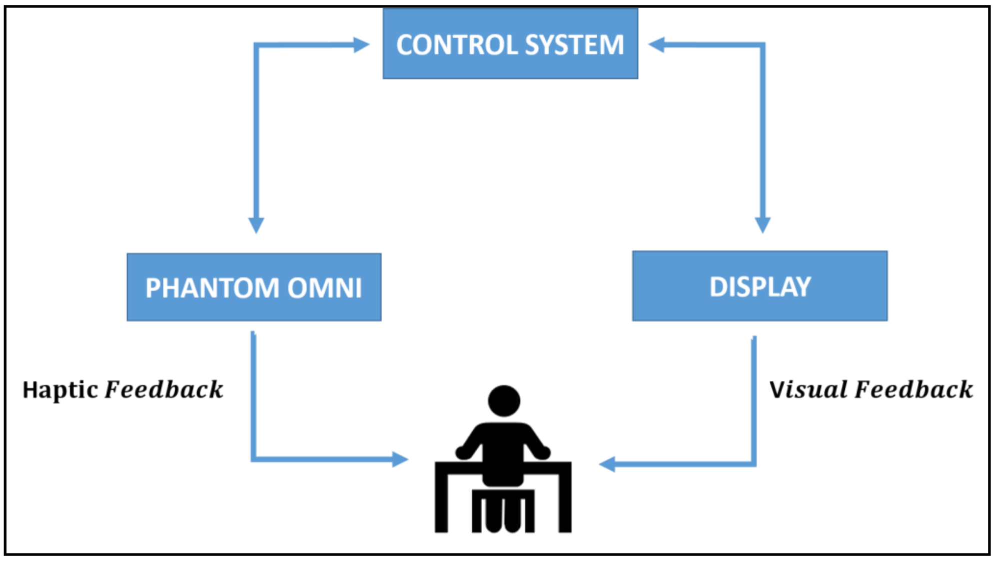


Figure 2.16 Scheme of the Hardware Setup.

2.2.2.1 Haptic Interface: Phantom OMNI® haptic device.

The Phantom OMNI® device, by SensAble Technologies, was chosen as haptic device.

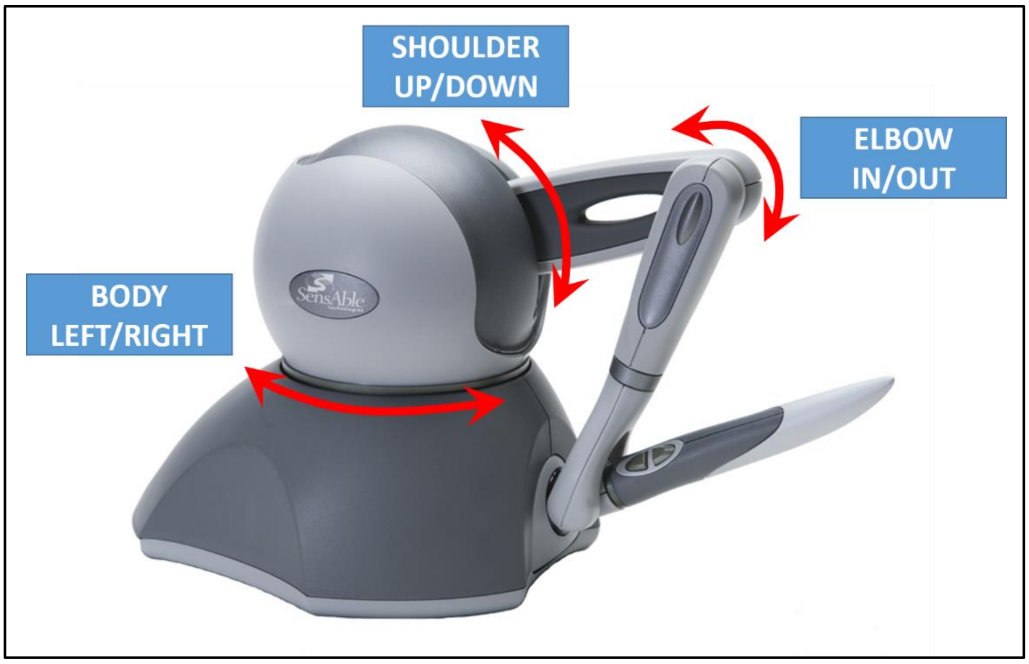


Figure 2.18 Phantom OMNI® haptic device. It is characterized by six degrees of freedom positional sensing and three torque actuators that control the angular position of the body, the shoulder and the elbow of the device.

The device is characterized by the following technical specifications:

- Dexterous Serial manipulator design;
- Portable design and compact footprint for workplace flexibility;
- Compact workspace for ease-of-use;
- Stylus-docking inkwell for automatic calibration;
- Six degree-of-freedom positional sensing;
- Nominal position resolution $res < 0.055mm$.

2.2.3 Experimental Protocol

The goal of this project is the investigation of the constraint violation classification performance of two different algorithms: Hidden Markov Models and Neural Networks. Both classifiers are trained to distinguish between "intentional" and "unintentional" violation of active constraints. Intentional violations occur when the enforced constraint does not share the purpose of the human operator and thus the constraint is felt as a hindrance, resulting in disturbing forces at the tool tip. On the other hand, unintentional violations are unavoidable due to intrinsic "biological" errors in the positioning of the tool tip. We set up a simple experimental protocol, composed of two sections:

- *Task Design.* In this section will be presented the strategies adopted for the conception of the tasks and their virtual enforcement;
- *Task Execution.* In this section will be discussed the guidelines followed when participants were asked to execute the tasks;

2.2.3.1 Task Design

We decided to separately assess the classification performances for tasks characterized by either "guidance constraints" or "forbidden-region" constraints. Hence, it was decided to implement two sets of tasks:

- One subset composed of 10 "path-following" tasks;
- One subset composed of 10 "target-reaching" tasks.

It was decided to introduce a degree of variability across tasks of the same type to evaluate how the performances of both HMM-based and NN-based classifiers were affected.

In the first set of "path-following" tasks, the variability was introduced by changing three factors:

- the number and size of obstacles;
- the length and shape of the trajectory;

- the width of the tube-shaped region within which a deviation error is permitted without any corrective force.

In the second set of "target-reaching" tasks, a degree of variability was introduced by changing:

- the number of target points;
- the distance between the center point and each target;
- the strength of the "forbidden region" constraint.

2.2.3.2 Task execution

We asked 12 subject, aged between 20 and 30, to perform the total set of 20 tasks each, ten path-following tasks and ten target-reaching tasks. To guarantee reliability and reproducibility of the experiments the following guidelines were followed:

1. The subject is asked which hand he/she prefers using to grasp the stylus of the Phantom Omni; the relative position between the display and the haptic interface is consequently adjusted;
2. The subject sits and adjusts the height of the chair such that his/her elbow can be positioned comfortably on the table;
3. The subject is asked to line up his/her elbow with the centerline of the haptic manipulator;
4. The robot workspace is defined as follows:
   i. $x \in (-90; +90)mm$;
   ii. $y \in (-70; +05)mm$;
5. The subject is asked, pivoting on the elbow, to freely move the stylus, and set the elbow-robot distance such that he/she can reach the workspace boundaries without shift the elbow;

Path-following protocol. Once the subject is set into the correct position he/she is asked to execute the first set of following-tasks:

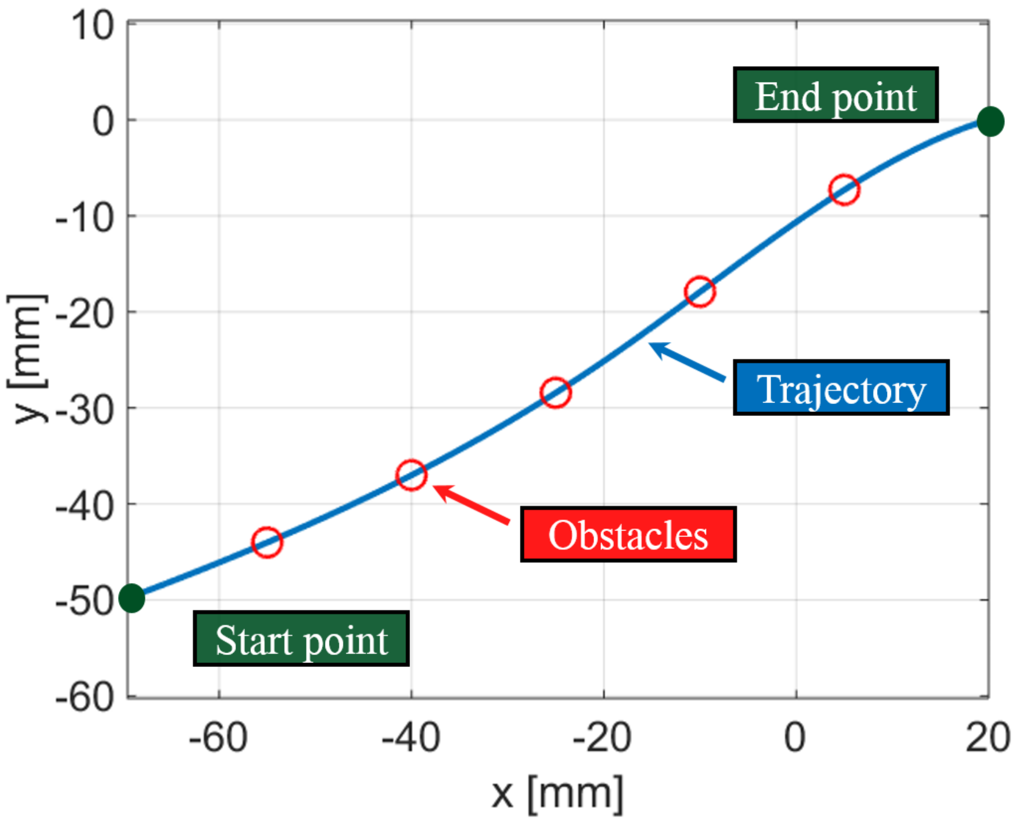


Figure 2.19 Example of "path-following" task. It is displayed the trajectory (blue line), and the circular obstacles (red circles) placed along the path.

1. The trajectory is visualized on the screen, together with the circular obstacles (Figure 2.19); a pointer identifies the real-time position of the tool-tip of the end-effector with respect to the path and obstacles;
2. For each task, the subject is instructed to "follow the displayed trajectory as accurately as possible, relying both on the visual and haptic feedback. The circular obstacles must be externally circumvented, either sides, as accurately as possible, trying not to violate the inner area. The execution time is not important".

3. The subject is free to choose one of the two boundary points of the path as “Start point”. The pointer is located onto this point and consequently the constraints are enforced;
4. The execution is stopped when the opposite point, the “End point”, is reached.

Target-reaching protocol. After the subject has performed all ten path-following tasks, he/she can rest to recover from muscular fatigue. After a period of about five minutes, the target-reaching tasks are executed:

1. The targets are visualized on the screen, marked as red points. The subject is asked to position the pointer that identifies the tool tip, on the center point, marked with a blue point.

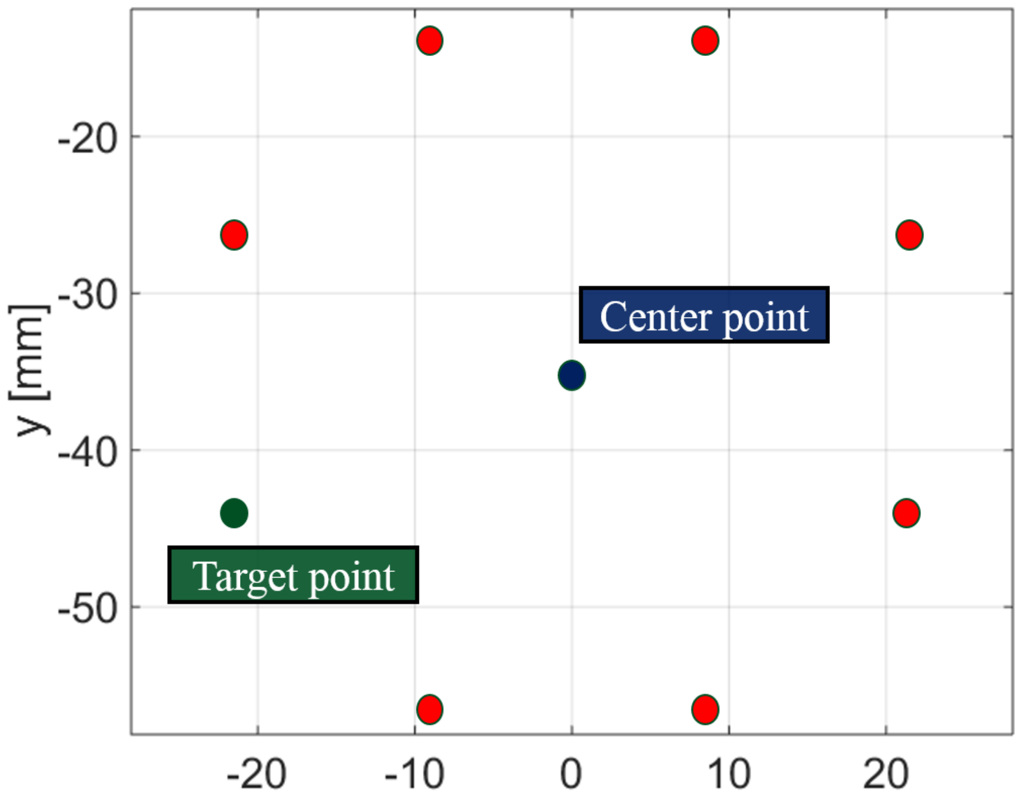


Figure 2.20 Example of "target-reaching" task. The subject is asked to position the tool tip, as accurately as possible, onto the current target point, marked with a green point.

2. The constraints are enforced, and one of the target become green. The subject is asked to reach the target and stop the pointer on the green marker as accurately as possible (Figure 2.20). Once the target is reached, he/she waits for about three seconds and then returns to the center point;
3. The target point is updated and the user repeats the reaching action toward the new target.
4. The task finishes once all the targets have been reached.

2.3 Classification Methods

In machine learning theory, classification refers to the problem of identifying to which set of categories the actual observation belongs. The observation is a vector of measured/computed features describing the "object" (e.g. a signal) that we want to place into the correct class or category. In general, classification methods are built upon two main steps: the training phase and the validation phase. The training phase is the first step of the procedure, during which the classifier learns to recognize and distinguish the variety of classes that compose the training set. According to classification theory, three main training approaches exist (Hertz, 1991):

- **Supervised learning,** in which the training process is done by comparing the actual output of the network with known correct answers (learning with a teacher).
- **Reinforcement learning,** in which the network knows whether the output is correct or not;
- **Unsupervised learning,** in which the training process is carried out without any teacher: the network is expected to create categories on the basis of the correlations of the input data, and to produce output signals corresponding to the input categories.

The second step is the validation phase, during which the classifier is used to categorize new observations, different from those used for the training phase.
In the following sections, two classification approaches are discussed: Hidden Markov Models (Li, 2006) and Neural Networks. The first approach is an unsupervised method based upon a probabilistic model of the observation data, which computes the likelihood that the current observation vector belongs to a given statistical distribution. The second approach is a supervised methodology based on a nonlinear mapping function g between a n-dimensional input space and a m-dimensional output space. The input space is composed of the observation vectors, while the output space dimension is equal to the number of classes we want the input data be sorted.

2.3.1 Markov Models

Markov Models (MM) were first introduced and studied in the late 1960s and early 1970s as a statistical method to study and model real-world processes. The reason as to why Markov Models have become increasingly popular in the last several years is mainly due to the fact that MM are based on a strong mathematical structure. Hence, they can form the theoretical basis for use in a wide range of applications, such as on-line handwriting recognition, speech and gesture recognition, language modeling, motion video analysis and tracking, protein/gene sequence alignment, stock price prediction.
The conception of Markov models was inspired by the fact that real-world processes generally produce observable outputs, which can be characterized as signals. These signals can either be discrete (e.g., characters from a finite alphabet), or continuous (e.g., temperature measurement). The goal of Markov Models is to characterize such signals in terms of signal models, namely to build a mathematical structure that is able to explain the time evolution of the

observed/measured phenomenon. The advantages that arise from having a reliable model are the following:

- A model provides the basis for a theoretical description of a signal processing system that can be used to process the signal to obtain a desired output;
- The model can provide a description of the signal source without directly observing it;
- Models are generally powerful tools in practical applications, like prediction, recognition and identification systems.

MMs are based on a statistical approach, which is built on the hypothesis that the observable signal is the result of a parametric random process. The process parameters can be estimated/determined in a well-defined manner.

2.3.1.1 Markov Chains

The simplest implementation of Markov Models are Markov Chains (Rabiner, 1989). A Markov Chain is a discrete system that is described at any time as being in one of the set of N distinct states $S_1, S_2, \dots, S_N$. At each time increment $t = 1, 2, 3 \dots$, the system undergoes a change of state, according to a set of probabilities associated with each state. In Figure 2.12 is shown a 3-state Markov Chain with transition probabilities associated to states A, B and C. Markov Chains are characterized by the following probabilistic description:

$$P\big(q_t = S_j \big| q_{t-1} = S_i, q_{t-2} = S_k \dots\big) = P(q_t = j | q_{t-1} = S_i) \qquad \text{(Eq. 2.30)}$$

where q_t is the state at time t and S_j identifies the actual state. This probabilistic description implies that the state at time t is a function of only the state at time $t - 1$, yielding a first-order MM.

The transition probabilities matrix A is defined as:

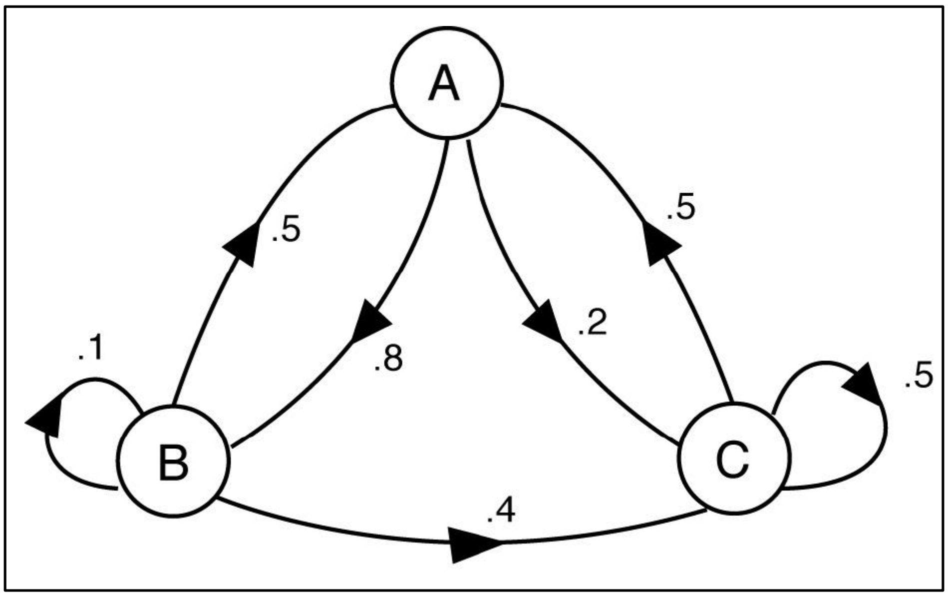


Figure 2.21 Example of Markov Chain characterized by three states A, B and C.

$$a_{ij} = P(q_t = j | q_{t-1} = S_i) \qquad 1 \leq i, j \leq N \qquad \text{(Eq. 2.31)}$$

The elements of the matrix A obey the following stochastic constraints:

$$a_{ij} \geq 0 \qquad \text{(Eq. 2.32)}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \text{(Eq. 2.33)}$$

If the measured output is the set of states q_t, the model is called “observable”. Otherwise, if the observation is the result of a probabilistic function of the state, the model is called “Hidden MM” (HMM).

2.3.1.2 Hidden Markov Models (HMMs)

HMMs are discrete models where the current state q_t is not directly observable, because the state is "hidden": what we measure as output is a sequence of observations produced by a stochastic function of the state, i.e.

$$\boldsymbol{O} = \boldsymbol{O_1}\ \boldsymbol{O_2}\ \boldsymbol{O_3} \dots \boldsymbol{O_T} \qquad \text{(Eq. 2.34)}$$

HHMs are described by five parameters (Rabiner, 1989):

- N is the number of states of the model;
- M is the number of distinct observation symbols (the discrete alphabet size);
- The state transition probability distribution $A = \{a_{ij}\}$;
- The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where

$$\boldsymbol{O} = \boldsymbol{O_1}\ \boldsymbol{O_2}\ \boldsymbol{O_3} \dots \boldsymbol{O_T} \qquad \text{(Eq. 2.35)}$$

$$b_j(k) = P(v_k\ at\ t | q_t = S_j) \qquad \begin{matrix} 1 \le j \le N \\ 1 \le k \le M \end{matrix} \qquad \text{(Eq. 2.36)}$$

- The initial state distribution $\pi = \{\pi_i\}$, where

$$\pi_j = P(q_1 = S_i) \qquad 1 \le i \le N \qquad \text{(Eq. 2.37)}$$

HMMs are generally associated with three basic problems (Rabiner, 1989):

- Given an observation sequence and a model $\lambda = (A, B, \pi)$, how do we compute $P(O|\lambda)$, the probability of the observation sequence given the model? Such problem is solved by using the “forward-backward” algorithm (Rabiner, 1989), (Devijer, 1985);

- Given an observation sequence and a model $\lambda = (A, B, \pi)$, how do we compute the corresponding state sequence that best explains the observations? Such problem is solved by using the “Viterbi” algorithm (Rabiner, 1989) (Forney Jr, 1973);
- How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$? To solve this problem an iterative procedure is necessary, that locally maximizes the quantity $P(O|\lambda)$, such as “Baum-Welch” method and gradient techniques (Rabiner, 1989).

2.3.1.3 Continuous HMMs

The HMM described so far is suitable when dealing with discrete observations, for which an observation probability B to infer some knowledge about the process. However, for some applications the measured system output is continuous signal. Although this problem can be overcome by discretizing the signal, it is preferred to introduce a probability density function (pdf) to model the state-observation relationship. The most general representation of the pdf is a finite mixture if the form:

$$b_j(\boldsymbol{O}) = \sum_{m=1}^{M} c_{jm}\,\mathfrak{N}[\boldsymbol{O}, \mu_{jm}, U_{jm}] \qquad 1 \leq j \leq N \qquad \text{(Eq. 2.38)}$$

where $\boldsymbol{O}$ is the vector being modeled, c_{jm} is the mixture coefficient for the mth mixture in state j and $\mathfrak{N}$ is any log-concave or elliptically symmetric density, (e.g., Gaussian density) with mean vector $\boldsymbol{\mu}_{jm}$ and covariance matrix $\boldsymbol{U}_{jm}$ for the mth mixture component in state j. This relationship can be used to approximate, arbitrarily closely, any finite, continuous density function. The mixture gains satisfy the stochastic constraint:

$$\sum_{m=1}^{M} c_{jm} = 1 \qquad 1 \geq j \geq N \qquad \text{(Eq. 2.39)}$$

$$c_{mj} \geq 0 \qquad 1 \leq j \leq N, 1 \leq m \leq M \qquad \text{(Eq. 2.40)}$$

According to these constraints, the pdf is properly normalized such that

$$\int_{-\infty}^{+\infty} b_j(x)dx = 1 \qquad 1 \leq j \leq N \qquad \text{(Eq. 2.41)}$$

The re-estimation process yield the updated values of the mixture coefficients, the mean vector and the covariance matrix for each mixture (Liporace, 1982).

2.3.2 Neural Networks

Neural Networks (NN) were first introduced as an alternative computational paradigm to the usual one (based on programmed instruction sequences) which was introduce by Von Neumann and has been used as the basis of almost all machine computation to date. NNs studies were inspired by knowledge from neuroscience and by the wide range possibility of making artificial computing networks (Hertz, 1991). In particular, the structure and functioning of a general Neural Network is inspired to the structure and functions of a biological neuron, shown in Figure 2.22.

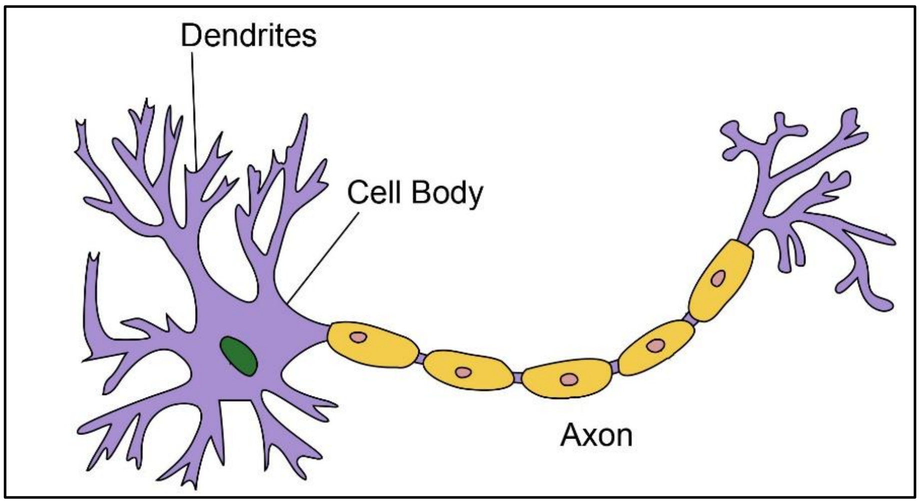


Figure 2.22 Scheme of a biological neuron.

The general neuron structure is characterized by the following elements:

- A tree-like network of fibers called “dendrites”, that gather in incoming information;
- A “cell body” or “soma”, where the nucleus is located and where the dendrites convey the information which is then processed;
- A single long fiber originating from the soma, called “axon”, which eventually branches into strands and sub-strands. The axon outputs the processed information to further neurons by means of the synaptic junctions.

The transmission of the signal from one cell to another is a complex chemical process whose effect is to raise or lower the electrical potential inside the body of the receiving cell. If this potential reaches a threshold, an action potential is sent down the axon: the cell has fired. In 1943, McCulloch and Pitts proposed a simple model of a neuron as a binary threshold unit that computes a weighted sum of its inputs and outputs “1” or “0” according to whether this sum is above or below a given threshold:

$$y_i(t+1) = \varphi\left(\sum_j w_{ij}x_j((t) - \mu_i)\right) \quad \text{(Eq. 2.42)}$$

or in vector notation,

$$\boldsymbol{y} = \varphi(w\boldsymbol{x} - \boldsymbol{\mu}) \quad \text{(Eq. 2.43)}$$

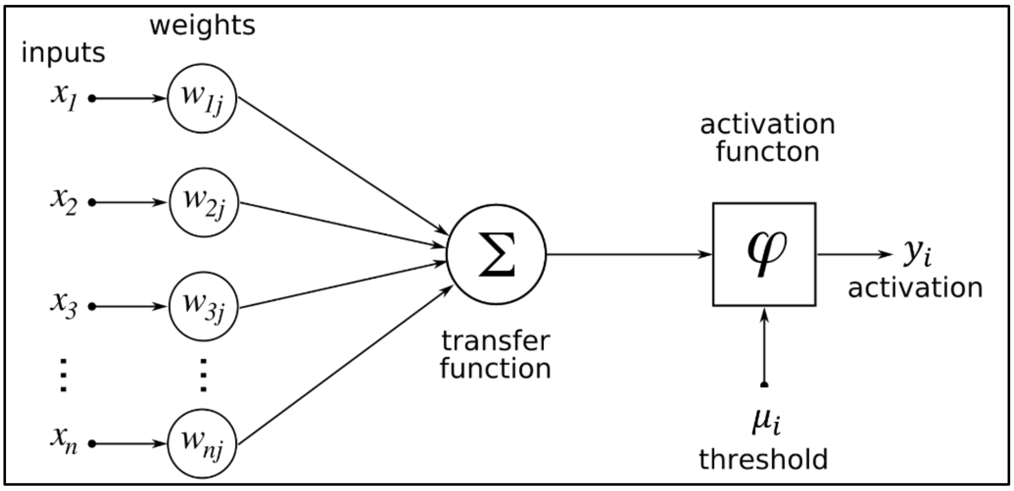


Figure 2.23 McCulloch Pitts unit.

Where y_i represents the state of neuron i as *firing* $(y = 1)$ or *not firing* $(y = 0)$, t is the discrete time and $\Theta(x)$ is the Heaviside function defined as:

$$\varphi(\boldsymbol{x}) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & if\ x \leq 0 \end{cases} \quad \text{(Eq. 2.44)}$$

The weights w_{ij} represent the strength of the synaptic junction that connects neuron j to neuron i. Depending on the nature of such connection, weights might assume either positive or negative values, corresponding to “excitatory” or “inhibitory” synapse, respectively. The cell-specific parameter μ_i is the threshold value for unit i, that sets the value that has to be overcome to have an input signal $y_i(t) \neq 0$. An overview of the McCulloch-Pitts structure is shown in Figure 2.23. Though a McCulloch-Pitts unit is a computationally powerful device, its structure is much simpler in comparison to the biological neurons that are characterized by the following features:

- They are not threshold entities, but respond in a continuous way (graded response);
- They might perform *nonlinear* summation of inputs;
- They produce a train of impulses, not a simple output level;
- The firing delay is not fixed, but varies stochastically;
- Multiple neurons acts asynchronously with respect to each other;
- The amount of transmitter may vary unpredictably.

A generalization of the McCulloch-Pitts equation that can embed some of these features is the following:

$$y_i(t+1) = g\left(\sum_j w_{ij} x_j(t) - \mu_i\right) \qquad \text{(Eq. 2.45)}$$

where $g(\cdot)$ is a more general nonlinear continuous function, called “activation” or “state” function.

2.3.2.1 Perceptron Units

The fundamental McCulloch-Pitts unit is the building block of the so-called “Rosenblatt Perceptron”. A perceptron is defined as a layered feed-forward structure, composed by a set of input terminals, one or more intermediate layers of units, and a final output layer where the result of the computation is read off (Hertz, 1991). The structure of a feed-forward network is defined such that each neuron belonging to the layer k can only convey the information to any of the neurons belonging to the next layer $k + 1$. No backward propagation nor communication with neurons belonging to the same layer is permitted. An overview of the structure of the perceptron is shown in Figure 2.24.

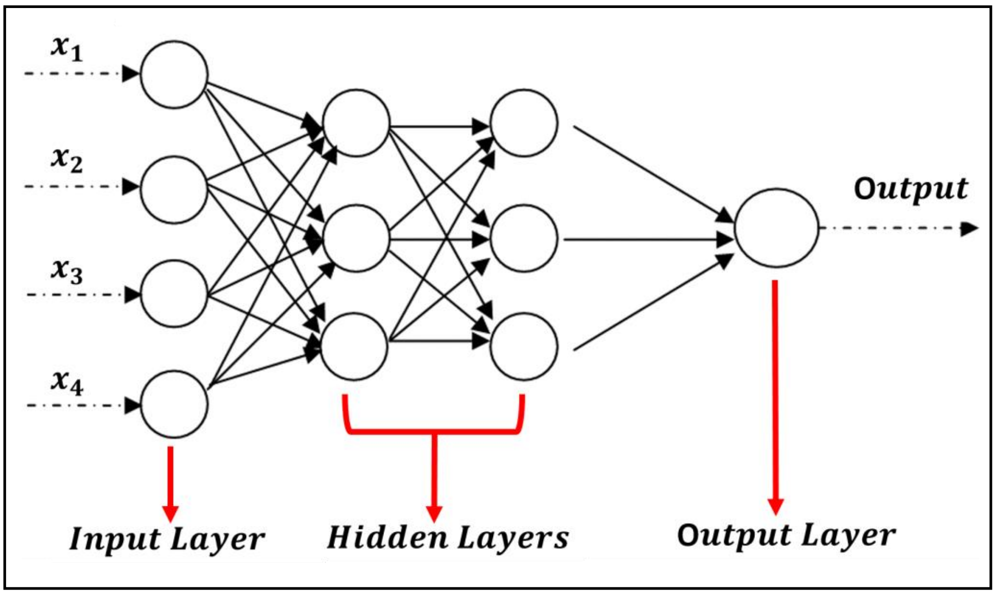


Figure 2.24 Scheme of the Perceptron Unit.

The structure of a perceptron is defined by the following elements:

- Number of inputs N;
- Number of outputs M;
- Number of Hidden layers and number of neurons to locate in each layer;
- Topology of connections among different layers;
- Activation functions that characterize each neuron.

The choice of the proper activation function has to be defined on the basis of the specific goal of the network. In general, perceptron is required to execute an association task, that can be cast in the form of asking for a particular output pattern ξ_i^μ in response to a given input pattern ξ_k^μ. Hence, we want that the actual output pattern O_k^μ has to be equal to the target (desired) pattern ξ_i^μ

$$O_k^{\mu} = \xi_i^{\mu} \qquad \forall k, i \qquad \text{(Eq. 2.46)}$$

Such association between the input and the output spaces requires that the network learn which is the optimal mapping function between these spaces. The mapping function can either perform a "linear" or "nonlinear" transformation, depending on the characteristics of the input data.

The most common activation functions $g(\cdot)$ are shown in Figure 2.25

- $y = sgn(x)$;
- $y = x$;
- $y = \tanh(x)$;
- $y = 1/(1+e^{-x})$;

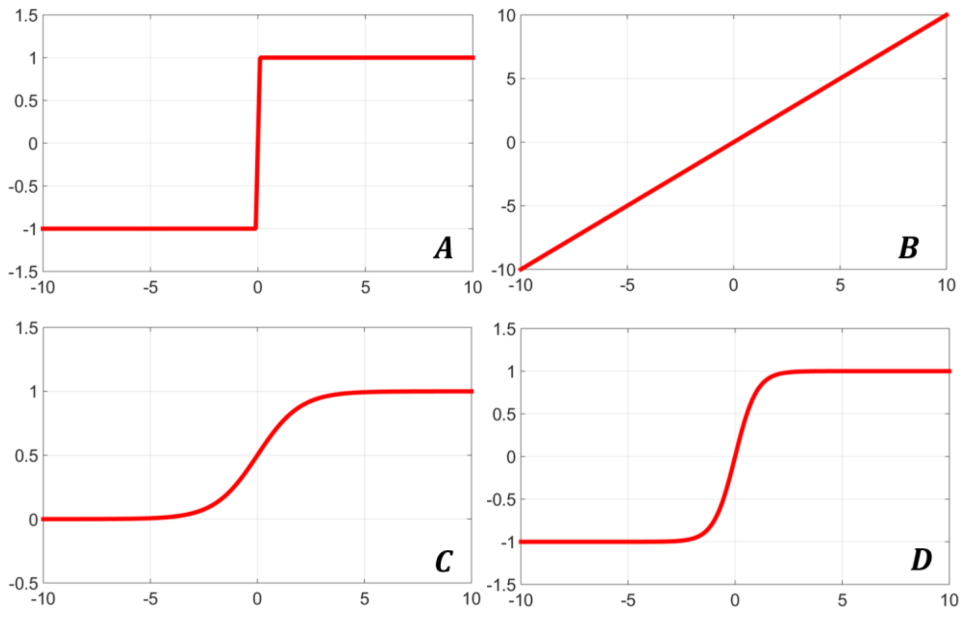


Figure 2.25 Common activation functions: sgn function (A), linear function (B), sigmoidal function (C) and hyperbolic tangent (D).

2.3.2.2 Neural Network Input Spaces

The training process of neural networks is aimed at finding the "local" optimal mapping function between the input and output spaces:

$$f: \mathbb{R}^m \rightarrow \mathbb{R}^n \qquad \text{(Eq. 2.47)}$$

Where m is the dimension of the input space and n is the dimension of the output space. The dimensionality of both spaces represents the number of input and output neurons, respectively. In general $n < m$. The choice of the dimension of the output space depends on the particular mapping function that the network has to perform. For example, in a binary classification function, the output space is characterized by a dimension $n = 1$, yielding one single neuron that outputs the result of the classification $y = \pm 1$. In prototyping tasks, the dimension n matches the number of prototypes that the network should learn to detect and classify. On the other hand, the input space is composed of the variables deemed meaningful for the optimal training of the network.

2.3.2.2.1 Statistical Input Space

A statistical parameter is a numerical characteristics of a population or a statistical model. In statistical inference, parameters are often unknown and have to be inferred on the basis of the observation of a random sample taken from the population of interest. Given a continuous time signal $s(t)$, a general discrete sequence of N samples, that represents a random observation of such signal, can be expressed as:

$$p = \{p_n\} \qquad n = 1, \ldots, N \qquad \text{(Eq. 2.48)}$$

Starting from a finite sequence of samples, some statistical parameters are defined to describe the characteristics of the continuous signal (Phinyomark, 2012):

- Arithmetic mean μ;
- Variance σ^2;
- Integral Value IV;
- Energy E;
- Maximum Value MV;
- Waveform Length WL;
- Average Amplitude Change AAC.

Arithmetic mean μ. The arithmetic mean μ is an index of central tendency that represents the center value of a given random variable. The statistical estimator is defined as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_n \qquad \text{(Eq. 2.49)}$$

where N is the number of samples of the observation vector. This statistical estimator is an "unbiased estimator" of the population mean.
Variance σ^2. The variance σ^2 is an index of dispersion that represents how far a set of numbers is spread out. A little value means that the observations tend to be very close to the mean value. The statistical estimator of the variance is defined:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(p_n - \mu)^2 \qquad \text{(Eq. 2.50)}$$

where μ is the mean value, and N is the number of samples. This statistical estimator is an "unbiased estimator" of the population variance.

Integral Value IV. The integral value is defined as the sum of the samples of the observation vector, i.e.

$$IV = \sum_{i=1}^{N} p_n \qquad \text{(Eq. 2.51)}$$

Energy E. The energy of a finite sequence of data is defined as the sum of the square value of the observation vector, i.e.

$$E = \sum_{i=1}^{N} (p_n)^2 \qquad \text{(Eq. 2.52)}$$

Maximum Value MV. The maximum value is the sample with the absolute highest value, i.e.

$$ML = \underset{n}{\operatorname{argmax}}(p_n) \qquad \text{(Eq. 2.53)}$$

Waveform Length WL. The waveform length is the index that estimates the length of the time profile of the finite sequence. Hence, it is defined as:

$$WL = \sum_{i=1}^{N-1} |p_{n+1} - p_n| \qquad \text{(Eq. 2.54)}$$

Average Amplitude Change AAC. The average amplitude change is an estimator proportional to the derivative of the signal, defined as:

$$AAC = \frac{1}{N-1} \sum_{i=1}^{N-1} |p_{n+1} - p_n| \qquad \text{(Eq. 2.55)}$$

The seven parameters above mentioned are the descriptors of a general observation vector constituted by finite sequence of data withdrawn from a continuous signal. These parameters are computed from the time evolution of the signal. In the next section, a frequency-domain approach will be discussed.

2.3.2.2.2 Frequency Input Space

Frequency analysis is an alternative methodology to time analysis when studying the characteristics of stochastic and deterministic signals. Frequency analysis, called also “spectral” analysis, performs the decomposition of a given signal into its basic frequency components. Given a continuous time signal $s(t)$, the simplest mapping operator from time domain to frequency domain is the Fourier Transform (FT), that yields the decomposition of the signal $s(t)$ into an orthonormal space constituted by sine and cosine functions given as (Nayak, 2011):

$$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-j2\pi ft} dt \qquad \text{(Eq. 2.56)}$$

where $s(t)$ is the time signal, f is the frequency variable and $S(f)$ is the Fourier Transform of the signal. The FT is a complex-valued function of the frequency, whose absolute value represents the amount of that frequency present in the original function and whose complex argument is the phase offset of the basic sinusoid in that frequency. The main drawback of the Fourier Transform is that the time information is lost, as the quantity $S(t)$ does not depend on t (Nayak, 2011). For this reason, FT is a powerful technique for stationary signals, where the characteristics do not change with time. For non-stationary signals, the spectral content changes with time and hence time-averaged amplitude spectrum found by using Fourier Transform is inadequate to track the changes in the signal magnitude, frequency or phase. The first attempt to introduce an explicit time-dependence is the Short Time Fourier Transform (STFT), the result of repeatedly multiplying the signal with shifted short time windows and performing the FT on it. The STFT of a signal $s(t)$ is defined as:

$$S(\tau, f) = \int_{-\infty}^{+\infty} s(t)w(t-\tau)e^{-j2\pi ft}dt \qquad \text{(Eq. 2.57)}$$

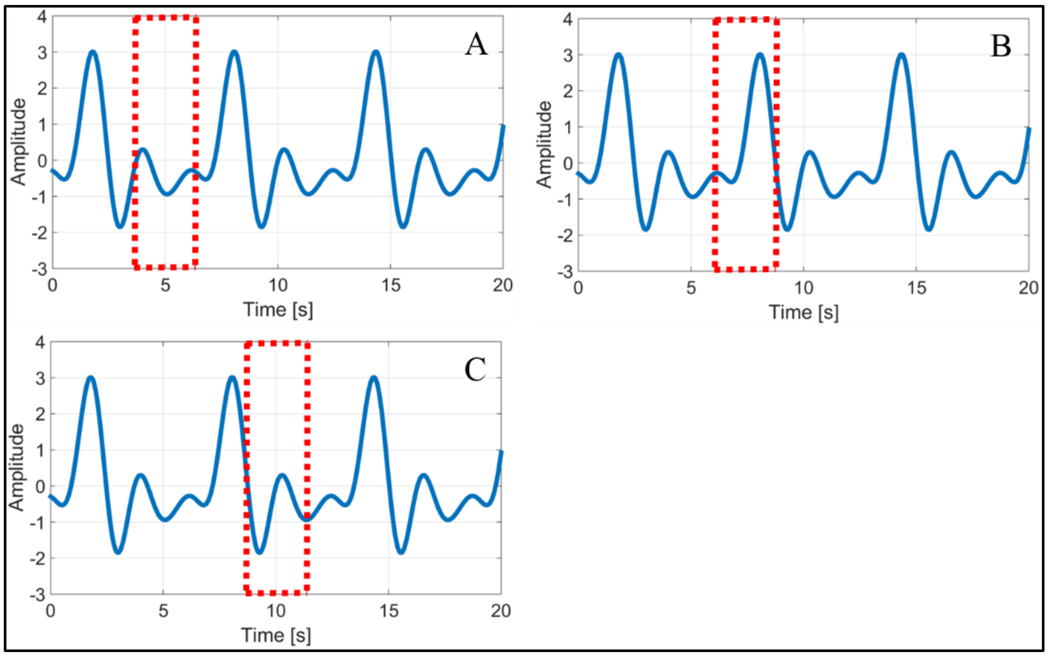


Figure 2.26 Application of Short Time Fourier Transform. The original signal is windowed and the local spectrum is calculated.

Where $w(t)$ is the windowing function, whose purpose is to dissect the signal to smaller segments, where the segments of the signal are assumed to be stationary. The resulting spectrum of this window is the "local spectrum". Figure 2.26 shows a scheme of the STFT method. The STFT represent a compromise between time and frequency views of a given signal. It provides some information about both when and at what frequency a signal event occurs. However, the reliability of the information is strictly linked to the size of the window. Due to the fact that constant a window is used in STFT approach, all parts of the signal are analyzed with the same resolution. This constitutes the main drawback of the method, because at high frequencies good time resolution is needed, while at low frequencies a good frequency resolution is essential. The frequency resolution is proportional to the bandwidth of the windowing function, while time resolution is proportional to the length of the window. Consequently, a short window is needed for good time resolution, and a long window is needed for good frequency resolution. This limitation is due to the Heisenberg-Gabor inequality (Nayak, 2011):

$$\Delta \mathrm{t} \cdot \Delta \mathrm{f} \geq c_w \qquad \text{(Eq. 2.58)}$$

where Δt and Δf are the time and frequency resolution, respectively. c_w is a constant that is dependent on the type of windowing function used. To overcome all the problems associated with Fourier analysis, a new mapping function based on Wavelets Transform is introduced. Like the Fourier Transform, the Continuous Wavelet Transform (CWT) uses inner products to measure the similarity between a signal $s(t)$ and an analyzing function. Unlike, the FT and STFT, that exploit sine and cosine functions to decompose the signal, this approach deploys wavelets ψ, that are waveforms with limited duration in time, average value of zero and nonzero norm. Table 2.1 shows a comparison of the three analyzing functions used in Fourier Transform, Short Time Fourier Transform and Wavelet Transform, respectively.

The principle of CWT is to compare the signal to shifted and scaled (compressed or stretched) versions of a “mother” wavelet. This operation yields a transformation $W(t, f)$ that embeds both time and frequency information, which are related to shifting and scaling operations, respectively.

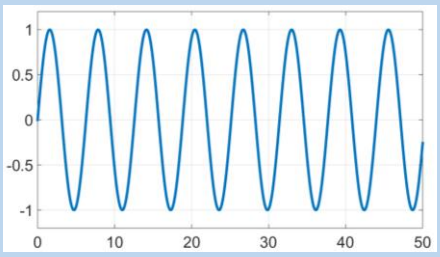
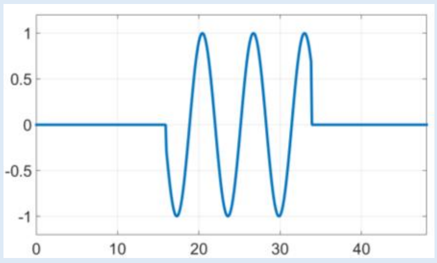
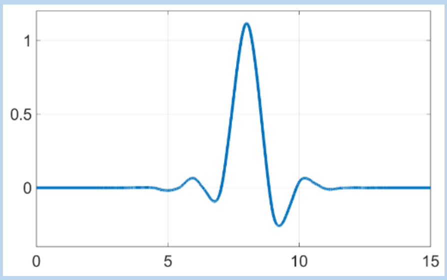
	Analyzing function	Plot
FOURIER TRASFORM	Complex Exponential $e^{-j2\pi ft}$	
SHORT TIME FOURIER TRANSFORM	Windowed complex Exponential $w(t)e^{-j2\pi ft}$	
WAVELET TRANSFORM	Wavelet function $\psi(t,f)$	

Table 2.1. Comparison of the different mother functions used for spectral analysis, in the case of Fourier Transform, Short Time Fourier Transform and Wavelet Transform.

Given a mother Wavelet ψ, the Continuous Wavelet Transform is defined as:

$$W(a,b) = \int_{-\infty}^{+\infty} s(t) W_{a.b}^{*}(t) dt \qquad \text{(Eq. 2.59)}$$

where $s(t)$ is the time signal, a is the scaling parameter, b is the translation parameter and $W^*_{a.b}(t)$ is the dilation and translation of the mother Wavelet:

$$W(a,b) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \qquad \text{(Eq. 2.60)}$$

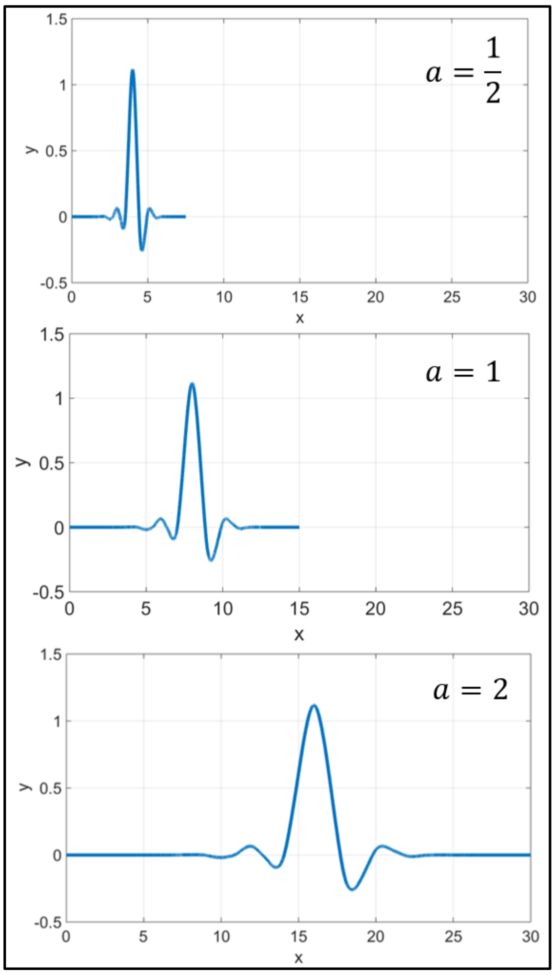


Figure 2.27 Effect of the scaling factor on the mother Wavelet.

As aforementioned, the parameters a and b control the amount of scaling and shifting of the mother wavelet $W(a,b)$, respectively. Hence, the parameter a is related to the frequency content of the signal, while the parameter b is related to its time information. In Figure 2.27 is shown the effect of changing the value of the parameter a. The smaller the scale factor, the more compressed the wavelet. Conversely, the larger the scale factor, the more stretched the wavelet. Therefore, the frequency support of each wavelet is modulated accordingly. Stretching the wavelet in time causes its support in the frequency domain to shrink. In addition to shrinking the frequency support, the center frequency of the wavelet shifts toward lower frequencies. The effect of the modulation of the parameter b is shown in Figure 2.28. By changing the value of

b, the analyzing wavelet undergoes a shift along the time axis, yielding. Consequently, it is superimposed onto different segments of the signal $s(t)$, yielding the time information. In the CWT, both parameters assume continuous values. In particular:

$$a\epsilon\mathbb{R}^+, \qquad b\epsilon\mathbb{R} \qquad \text{(Eq. 2.61)}$$

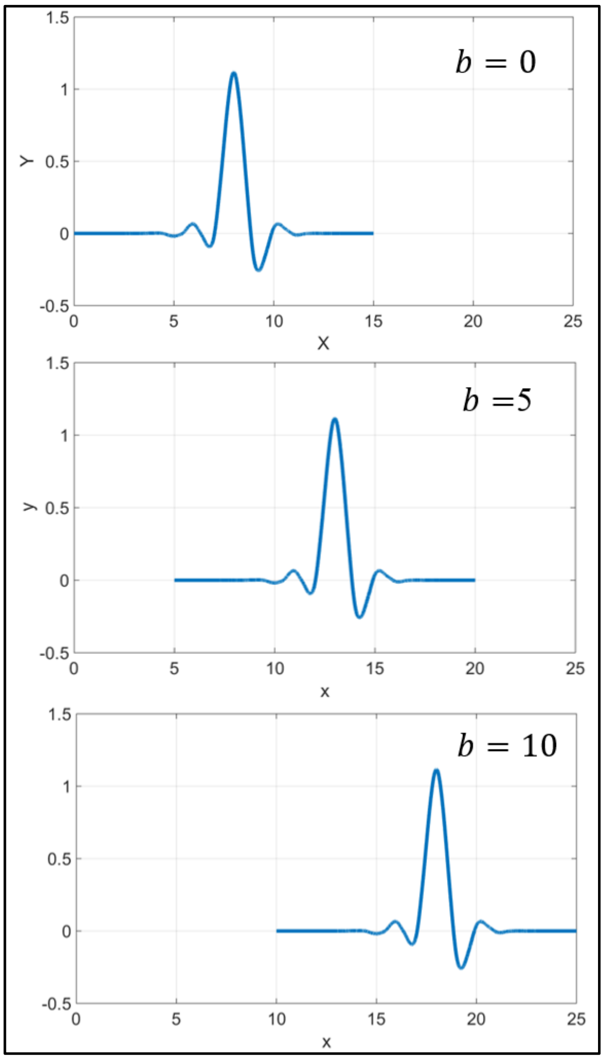


Figure 2.28 Effect of the shifting factor on the mother Wavelet.

In (Burrus, 1998) it was demonstrated that the Continuous Wavelet Transform provides a redundant representation of the signal in the sense that the entire support of $W(a, b)$ need not be used to correctly recover the original signal $s(t)$. The minimum amount of information that is required to correctly reconstruct the original signal $s(t)$ is obtained by down-sampling the CWT at “dyadic” intervals. This corresponds to the evaluation of the CWT at discrete values of the scaling and shifting parameters a, b, such that:

$$a_j = 2^j \qquad j = 1, 2, \dots \infty \qquad \text{(Eq. 2.62)}$$

$$b_{j,k} = 2^j k \qquad k = -\infty, \dots -1, -2, 0, 1, 2, \dots \infty \qquad \text{(Eq. 2.63)}$$

This operation corresponds to applying a down-sampling procedure to the CWT yielding the Discrete Wavelet Transform (DTW) of the signal $s(t)$, defined as:

$$W(a,b) = \int_{-\infty}^{+\infty} s(t) \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j k}{2^j}\right) dt \qquad \text{(Eq. 2.64)}$$

Graphically, the down-sampling effect corresponds to the superimposition of a dyadic grid to the CWT.

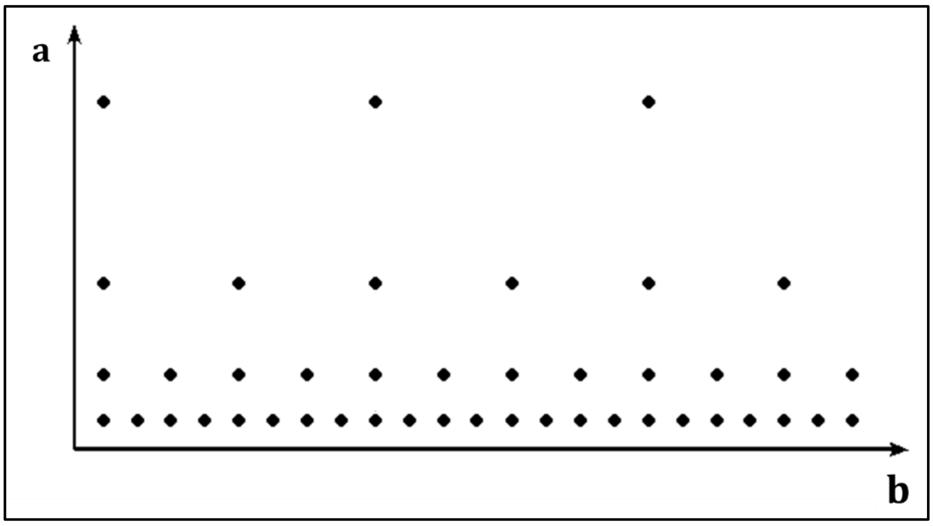


Figure 2.29 2D representation of the dyadic grid that yields the optimal sampling of the CWT to preserve all the information of the original signal.

The Discrete Wavelet Transform has the necessary information for the correct reconstruction of the original signal by means of the Inverse Discrete Wavelet Transform (IDWT), defined as:

$$s(t) = \sum_{k=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} d_j(k)\, 2^{j/2} \psi(2^j t - k) \qquad \text{(Eq. 2.65)}$$

where $d_j(t)$ are the Wavelet coefficients, that can be interpreted as the degree of correlation between the $s(t)$ and the mother Wavelet for a fixed pair $(\bar{a}, \bar{b})$.
The Discrete Wavelet Transform can be interpreted as an iterative filtering technique (Burrus, 1998): the computation of the wavelet coefficients $d_j(k)$ at a fixed scale a is the result of the following filtering operation

$$W(a, b) = s(t) * \psi(-b/a) \qquad \text{(Eq. 2.66)}$$

where ψ has a band-limited spectrum, so the filtering operation is a band-pass filter. This is the fundamental property that characterizes the Fast Wavelet Transform (FWT) algorithm, which was first developed by Mallat in 1988. The algorithm yields the discrete coefficients by recursively applying a band-pass filter with halved frequency support. The output of the Mallat algorithm is the Wavelet decomposition tree of the original signal $s(t)$. At each iterative step, the band-pass filtering operation splits up the input signal into a low-frequency component and a high-frequency component, that are usually referred to as "approximation" (cAn) and "detail" (cDn) coefficients, respectively. The scheme of the algorithm is shown in FIG. The first step of the algorithm is to apply a band-pass filter corresponding to the mother wavelet with scale factor $a = 0$ to the signal $s(t)$, thus obtaining the 1st level Wavelet decomposition coefficients cA1 and cD1. At each successive step, the frequency support of the actual band-pass filter is halved, due to the increase in the scale factor with the power of 2. This filter is then applied to the approximation coefficients obtained at the previous iteration. The maximum number of iterations that outputs a correct Wavelet decomposition tree is linked to the length of the signal $s(t)$ and to its sampling frequency (Burrus, 1998).

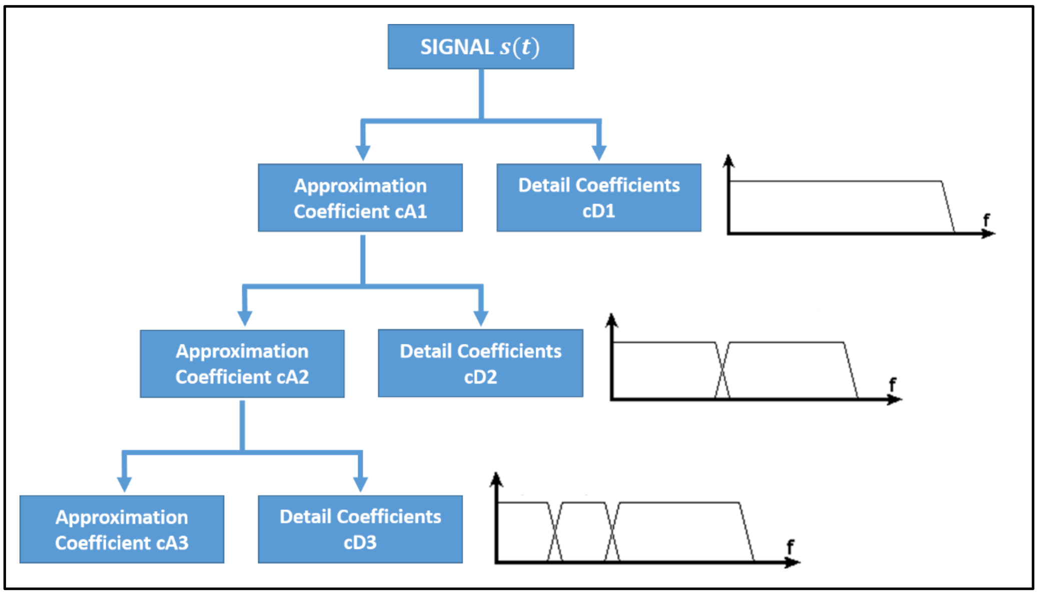

SIGNAL $s(t)$
Approximation Coefficient cA1
Detail Coefficients cD1
f
Approximation Coefficient cA2
Detail Coefficients cD2
f
Approximation Coefficient cA3
Detail Coefficients cD3
f

Figure 2.30 Graphical representation of the Wavelet Decomposition algorithm.

An important property of the DWT is that the energy of the signal $s(t)$ is distributed across the decomposition tree without loss of information (Pittner, 1999). Let N represent the decomposition level be applied to a finite sequence of data $p = \{p_n\}$. The energy of the sequence of data E_0 is defined as:

$$E_0 = \sum_{i=1}^{N} (p_n)^2 \qquad \text{(Eq. 2.67)}$$

cA_k and cD_k represent the "approximation" and "detail" sequences at level k of the Wavelet tree. The distribution of energy is formalized as follows:

$$E_0 = E_N^{cA} + \sum_{k=1}^{N} E_k^{cD} \qquad \text{(Eq. 2.68)}$$

where E_N^{cA} is the energy content of the approximation sequence of level N, and E_k^{cD} is the energy content of the detail sequences of level $k = 1, \dots, N$. For each decomposition level k, the amount of energy is given by:

$$E_k(\%) = 100 \cdot \frac{E_k}{E_0} \qquad \text{(Eq. 2.69)}$$

2.4 Data Analysis

The Data Analysis steps are related to signal processing and training phase of the HMM-based and NN-based algorithms. The main steps are the following:

1. *Recording* of the Cartesian Force **f**(t) that describes the evolution of the interaction between the tool tip of the haptic device and the active constraint;
2. *Segmentation* of the force signal on the basis of the existence of interaction force $\boldsymbol{f} > 0$.
3. *Labelling.* The segments obtained are automatically labeled as "intentional " or ""unintentional violations;
4. *Dataset creation.* The segments are sorted into different datasets;
5. *Training Phase.* Both HHM-based and NN-based classifiers are trained to distinguish between "intentional" and "unintentional" violations;
6. *Validation Phase.* The trained classifiers are used to classify the observations of the validation sets;
7. *Evaluation.* The performance of each method is assessed on the basis of the associated confusion matrixes and ROC curves.

Recording. During the execution of the task, the Cartesian components of the force vector **f**(t) are recorded. The z-component of the force is set to zero as the task is performed on a 2D plane:

$$\boldsymbol{f}(t) = \begin{bmatrix} f_x(t) \\ f_y(t) \\ 0 \end{bmatrix} \qquad \text{(Eq. 2.70)}$$

The modulus of the force is computed as:

$$\bar{f}(t) = \sqrt{{f_x}^2(t) + {f_y}^2(t)} \qquad \text{(Eq. 2.71)}$$

Segmentation. For each task, the modulus of the force $\bar{f}(t)$ is split into segments (Figure 2.31): Each segment represents one single interaction between the tip and the constraint. The segments are represented by a nonzero value of the interaction force:

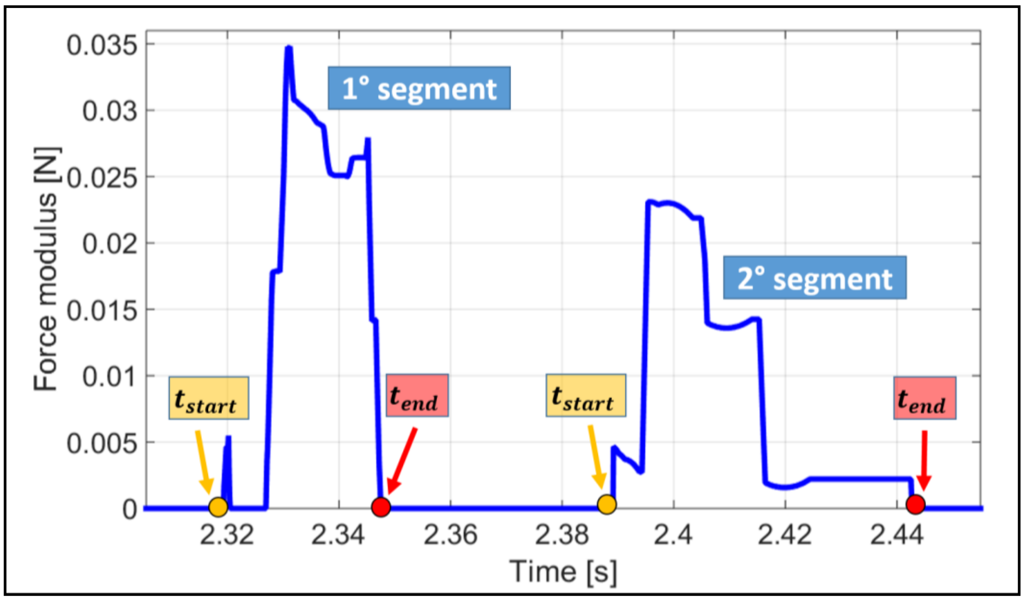


Figure 2.31 Example of two consecutive force segments, separated by a row of zero samples.

$$\bar{f}(t) > 0 \qquad \text{(Eq. 2.72)}$$

Segments are interspersed by force samples equal to zero, meaning that, in that time interval, the subject, has not violated the constraint. The condition that sets the boundaries of each segment are the following:

$$\bar{f}(t)\begin{cases} = 0, & t_{start} - 50ms < t < t_{start} \\ > 0, & t_{start} \leq t \leq t_{end} \\ = 0, & t_{end} < t < t_{end} + 50ms \end{cases} \qquad \text{(Eq. 2.73)}$$

where t_{start} is the time instant in which the interaction begins, t_{end} is the time instant where the interaction ends, 50ms is the time required separate two consecutive segments.

Automatic labelling. To build up a proper classifier, it is necessary to train and validate the model. During the training phase, the model learns to correctly classify the input observations of the training set. In particular, Neural networks are characterized by a "supervised" learning process: the weights and biases of the network are updated on the basis of the error between the desired output and the current output (Eq. 2.46). Hence, supervised learning requires the classification results to be known *a-priori*. Moreover, this information is important during the validation phase, as the assessment of the classifier performance is based on the comparison between the output result and the true value of the classification. To obtain such *apriori* knowledge, an automatic labelling algorithm was set up, thus avoiding either manual labelling of each interaction or asking the subject to explicitly tell his/her will to act against the constraint. The automatic labelling algorithm was based on the Cartesian position of the tool tip of the haptic interface. For the "path-following" task, the position $p(t)$ of the end effector is compared to the Cartesian coordinates of the center of each circular obstacles placed along the trajectory. For the “target-reaching” task, the position $p(t)$ is compared to the Cartesian position of each target point. The values were assigned such that

$$Intentional\ Violation \rightarrow +1$$
$$Unintentional\ violation \rightarrow -1$$

Dataset creation. To separately assess the classification performances in the case of both “guidance” and “forbidden-region” constraints, two different datasets were created. Each dataset was furtherly split into a training set and a validation set: the segments were randomly split and the observations were uniformly distributed such that the number of “intentional” and “unintentional” violations was evenly split between the training and the validation phases. Hence, a total of four datasets was obtained.

Training phase. During the training phase, the algorithm learns to distinguish between “intentional” and “unintentional” violations. Three training processes were carried out:

1. HMM-based classifier (Section 2.3.1.3);
2. NN-based classifier with statistical inputs (StNN) (Section 2.3.2.2.1);
3. NN-based classifier with spectral inputs (SpNN) (Section 2.3.2.2.2);

The HMM-based classifier is composed of two parallel Hidden Markov Models, each one separately trained either on “intentional” or on “unintentional” violations (Figure 2.32). The training set is constituted by observation vectors extracted from the modulus of the interaction force signal $\bar{f}(t)$. To train the model it was hypothesized that the observations could be explained as a random realization of a mixture of Gaussian distributions, characterized by different mean μ and

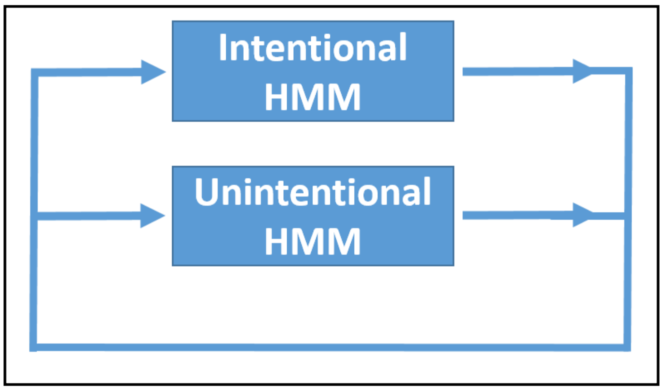


Figure 2.32 HMM-based model.

standard deviation σ. The best number of Gaussian distributions was computed by means of an optimization algorithm: it was found that nine Gaussians bells provided the best interpolation of the input data. Consequently, the number of states was set equal to the number of Gaussian bells of the mixture, and the training process was executed.

The Neural Network classifiers were trained on the basis of two different input features: statistical features (Section 2.3.2.2.1) and spectral features (Section 2.3.2.2.2), both extracted from the segments of the interaction force signal $\bar{f}(t)$. Two classifier were obtained: a statistical-based NN (StNN), and a spectral-based NN (SpNN).

The structure of the StNN is the following:

- 7 input neurons. As discussed in section 2.3.2.2.1, seven statistical features have been taken into account: mean, variance, maximum point, energy, integral value, waveform length and average amplitude change. Each statistical parameter was computed for each segment;
- 1 hidden layer, composed of 15 neurons;
- 1 output neuron, that yields the classification results (± 1).

The structure of the SpNN is the following:

- 10 input neurons. As discussed in section 2.3.2.2.2, the Wavelet decomposition algorithm, down to level N, is applied to each force segment, and the distribution of the signal energy across the different Wavelet levels (Eq. 2.69) is computed (a Daubechies 'db1' was chosen) .

- 1 hidden layer, composed by 30 neurons;
- 1 output neuron, that yields the results of the classification (± 1).

For both networks "hyperbolic tangent" (tanh) activation functions were chosen to bound the ouput in the interval $[-1, +1]$, and the training phase was performed on the basis of the "Levenberg-Marquardt" algorithm.

Validation phase. During the validation phase, the models are deployed to classify the observations of the validation set. In both HMM and NN methods, we want to assess which is the minimum time required, after the violation has started, to obtain reliable classification results. The fundamental hypothesis is that, during the very first phase of the interaction, the information extracted is not sufficient to detect intentional violations, as the magnitude range of the force signal $\bar{f}(t)$ is equal to the one characterizing unintentional violations. However, after a certain amount of time, the magnitude of the interaction force increases as the operator wants to either circumvent an obstacle or reach a target. The minimum time

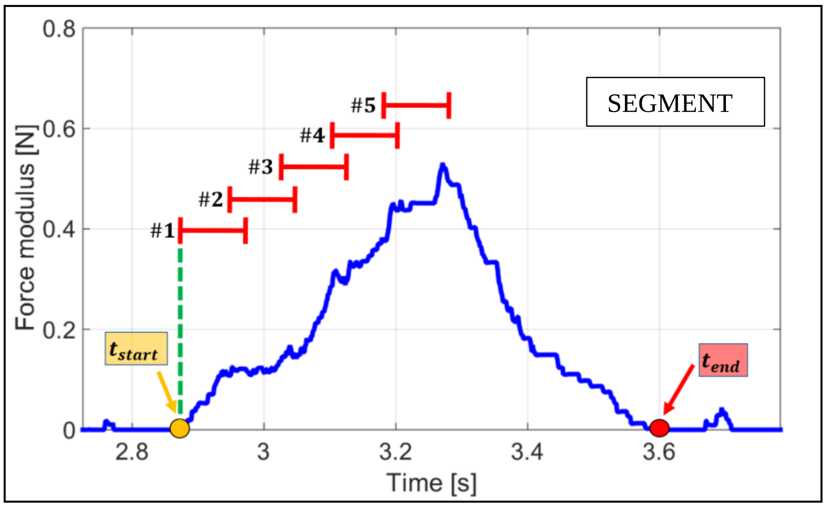


Figure 2.33 Application of the HMM-based classifier on a force segment. The observation buffers (#1, #2, …) are shifted in time to simulate the real-time application.

required to yield a reliable detection of intentional events is the research question. Both models were applied separately on each segment.

The HMM-based model (Figure 2.32) was validated by giving as observation vector a series of buffers (50 samples) shifted in time, to simulate the real-time application (Figure 2.33). The model outputs two likelihood functions, ϕ_{int} and ϕ_{unint}, related to both the intentional-HMM and unintentional-HMM. The likelihood score represents the probability that the current buffer, which constitutes the observation vector, belongs either to the first or the second model. We expect that, for intentional violations, the likelihood related to the intentional-HMM will be significantly higher than the likelihood associated to the unintentional-HMM.

The NN-based models were validated by feeding the network with the time evolution of either statistical or spectral features. For the StNN, statistical features were computed iteratively over time, following the general relationship:

$$\alpha(t) = f(\beta(t-1), s(t), N) \qquad \text{(Eq. 2.74)}$$

With $t_{start} \leq t \leq t_{end}$

$\alpha(t)$ is any statistical parameter at the current time instant, $\beta(t-1)$ is any statistical parameter computed at the previous step, $s(t)$ is the current value sampled from the force segment, N is the actual number of samples since the violation has started. The definition of the seven statistical parameters in section 2.3.2.2.1 is modified as follows.

Arithmetic Mean $\mu(t)$.

$$\mu(t) = \frac{(N-1)\mu(t-1) + s(t)}{N} \qquad \text{(Eq. 2.75)}$$

Integral Value $\mathrm{IV}(t)$.

$$\mathrm{IV(t) = IV(t-1) + s(t);} \qquad \text{(Eq. 2.76)}$$

Energy $\mathrm{E(t)}$

$$\mathrm{E(t) = E(t-1) + s^2(t)} \qquad \text{(Eq. 2.77)}$$

Variance $\sigma^2(t)$

$$\sigma^2(\mathrm{t}) = \frac{\mathrm{E(t) - N\mu^2(t)}}{\mathrm{N-1}} \qquad \text{(Eq. 2.78)}$$

Maximum Value $MV(t)$

$$\mathrm{MV(t) = max(MV(t-1)}, s\mathrm{(t))} \qquad \text{(Eq. 2.79)}$$

Waveform Length $WL(t)$

$$\mathrm{WL(t) = WL(t-1) + |s(t) - s(t-1)|} \qquad \text{(Eq. 2.80)}$$

Average Amplitude Change $AAC(t)$

$$AAC(t) = \frac{WL(t)}{N} \qquad \text{(Eq. 2.81)}$$

The SpNN model was validated by feeding the network with the time evolution of the amount energy of distributed across different frequency bands. As discussed in section 2.3.2.2.2, the Wavelet decomposition is a linear operator with respect to the energy of the signal to decompose, i.e.:

$$E_0 = E_N^{cA} + \sum_{k=1}^{N} E_k^{cD} \qquad \text{(Eq. 2.82)}$$

Where E_0 is the energy of the force segment, E_N^{cA} is the energy content of the approximation sequence of level N, and E_k^{cD} is the energy content of the detail sequences of level $k = 1, \dots, N$. The energy distribution was calculated by extracting from each force segment a buffer of samples of increasing length (Figure 2.34). The Wavelet decomposition was computed, yielding the energy distribution;

$$E_k(\%) = 100 \cdot \frac{E_k}{E_0} \qquad \text{(Eq. 2.83)}$$

For $k = 1, \dots N + 1$.

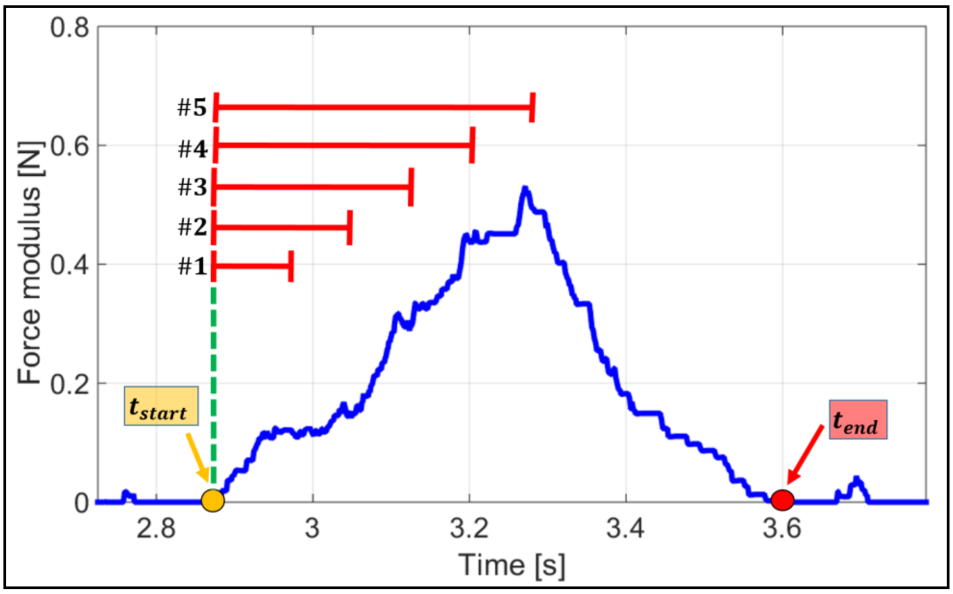


Figure 2.34 Extraction of buffers of increasing length from a force segment.

Evaluation Phase. We chose to evaluate the performances of the classifiers at each time step in the discrete time evolution, through the construction of the confusion matrix (Vercellis, 2006), which yields the amount of correctly classified and misclassified observations. The confusion matrix is characterized by four parameters:

1. *True Positive* TP, is the amount of intentional violations correctly classified;
2. *True Negative* TN, is the amount of unintentional violations correctly classified;
3. *False Negative* FP, is the amount of misclassified intentional violations;
4. *False Positive* FP, is the amount of misclassified unintentional violations;

The confusion matrix is the following:

		PREDICTIONS	
		-1	+1
OBSERVATIONS	-1	TN	FP
	+1	FN	TP

Table 2.2 Scheme of a general Confusion Matrix.

These four values that constitute the confusion matrix are used to define two indexes of performance: Sensitivity and Specificity. The sensitivity is the percentage of intentional violations correctly classified:

$$Se = \frac{TP}{TP + FN} \tag{Eq. 2.84}$$

The specificity is the percentage of unintentional violations correctly classified:

$$Sp = \frac{TN}{TN + FP} \qquad \text{(Eq. 2.85)}$$

Both in the case of HMM-based and NN-based model, the output function is not a binary classification, but a continuous function. In the first case, the likelihood function can assume values in $\mathbb{R}$. The neural network output values are bounded in the interval $[-1, +1]$, as a consequence to the choice of "tanh" as activation functions. To fill the confusion matrix, a further discretization step must be introduced, by applying a threshold to the continuous output. By tuning this threshold, the values of specificity and sensitivity will vary accordingly: an increase in the sensitivity of the model leads to a decrease in the sensitivity and vice versa.

We judged that, in surgical applications, it would be advisable to have a higher specificity level and accept a lower degree of sensitivity. This choice has been done on the basis of safety reasons: in a real surgical scenario, the detection of intentional violations should trigger the adaptation and/or removal of the assistance level provided by the active constraint. Thus, a high percentage of "false positives", namely a high number of misclassified unintentional violations, would mean that, in many cases, the adaption is triggered even though the surgeon is not acting against the constraint. On the other hand, a high specificity level ensures that a high percentage of unintentional violations is reliably detected and rejected. Consequently, the sensitivity level decreases and the detection of intentional violations is less reliable. However, the system is safer as any misclassification of intentional actions has no effects on the adaptation level.

This reason lead us to compute the discretization threshold on the basis of a given specificity level: the sensitivity index is consequently computed.

For the HMM-based model, the threshold is compared with the absolute difference $\Delta\phi$ in the likelihood scores:

$$\Delta\phi = |\phi_{int} - \phi_{unint}| \qquad \text{(Eq. 2.86)}$$

If this difference overcomes a threshold Γ, the current observation is classified as intentional.

For the NN-based the threshold Γ is compared to the current output. If the output value overcomes the threshold, the observation is classified as intentional.

Once the sensitivity and specificity levels, for each classifier the ROC (Receiver Operating Characteristics) curves (Vercellis, 2006) are obtained. The ROC curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (Se) against the false positive rate ($1 - Sp$) at various threshold settings (Figure 2.35).

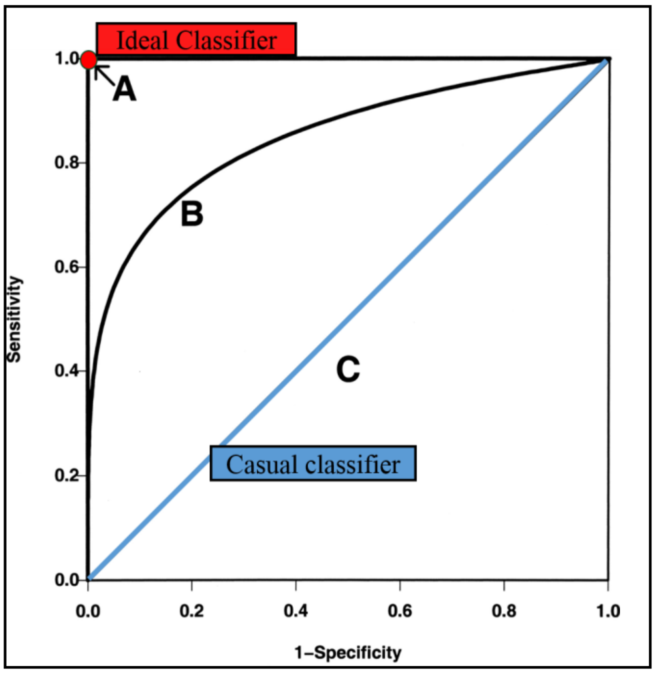


Figure 2.35 ROC curves. The curve "C" represents the casual classifier, "B" is the general classifier, "A" is the ideal classifier, represented by the point A(0, 1).

3 Results

In this chapter, the experimental results are presented. Separate discussions are carried out for both “path-following” and “target-reaching” cases. For each set of tasks, three classification models have been applied and discussed:

1. Hidden Markov Model;
2. Neural network based on statistical features (StNN);
3. Neural network based on spectral features (SpNN);

For each classification model, we show:

1. The time evolution of the ROC curves, to assess how the performance of the classifier evolves over time;
2. A hypothetical surgical application in which some specificity levels are set, and the corresponding sensitivity profiles over time are obtained. As discussed in section 2.4, for safety reasons it would be preferable to deploy a classifier characterized by high specificity levels. For this reason, we have decided to evaluate each model for five specificity levels: 80, 90, 95, 97.5 and 99%. The sensitivity profiles over time are subsequently plotted and evaluated in terms of minimum time interval required to exceed a sensitivity threshold of 90% and the corresponding penetration depth value, that indicates how deeply the tool tip has been pushed into the constraint.

3.1 Path-following task

A total of 120 “path-following” tasks were executed by 12 subjects. In Figure 3.1 is shown an example of the task execution. The red dotted line represents the

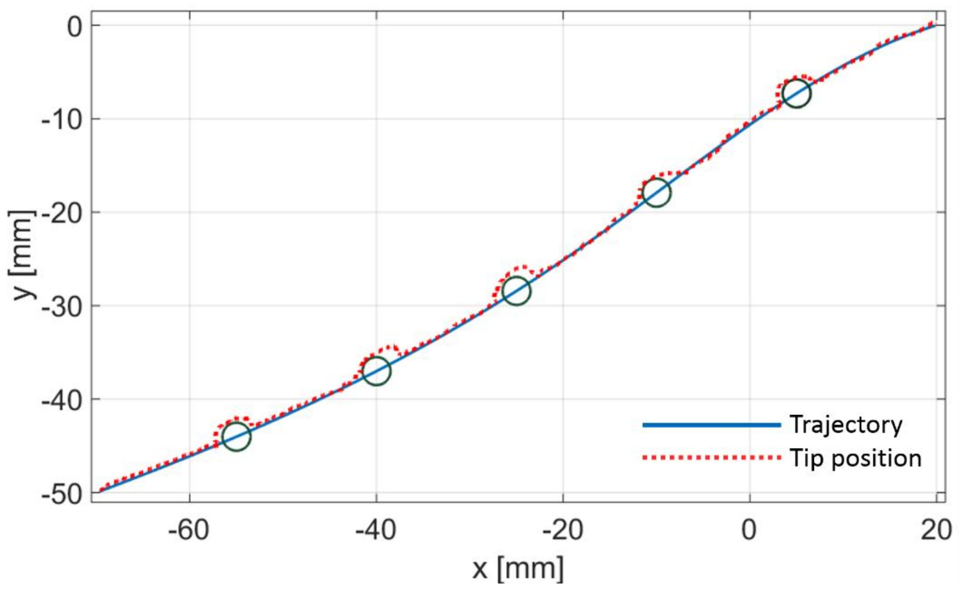


Figure 3.1 Example of "path-following" task execution. The blue line represents the path that the subject is asked to follow. The dotted red line represents the actual trajectory followed by the tip of the haptic device.

Cartesian trajectory described by the tool tip. In correspondence to the circular obstacles, the subject voluntarily acts against the constraint, thus departing from

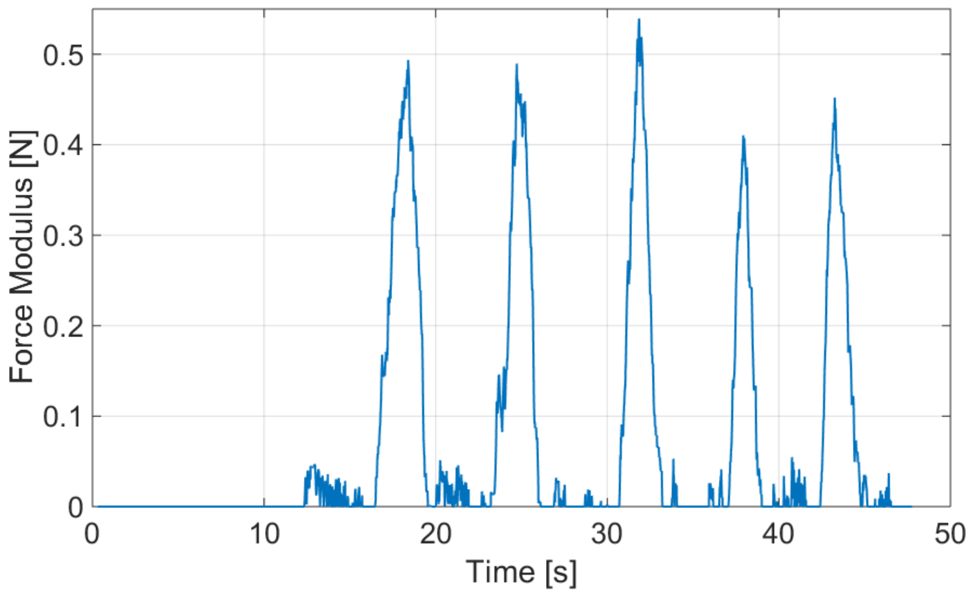


Figure 3.2 Modulus of the interaction force.

the path (blue line). In Figure 3.2 is shown the time profile of the interaction force corresponding to the task shown in Figure 3.1. In Figure 3.3 are displayed the Cartesian components of the force signal. In both plots, five force peaks are clearly distinguishable: these peaks represent the intentional violations due to the will to circumvent the obstacles.

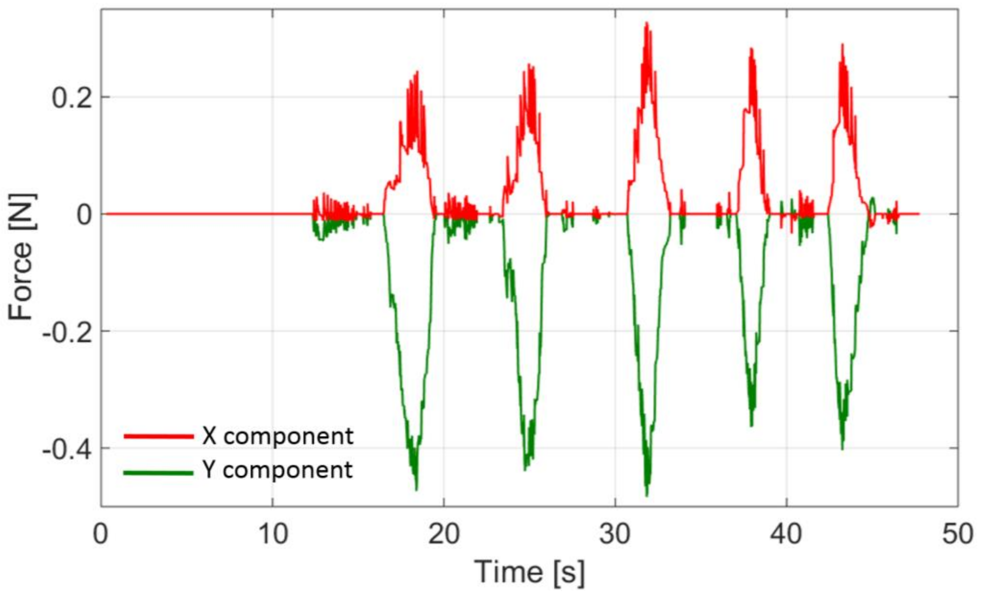


Figure 3.3 Cartesian components of the interaction force.

For each subset of 120 tasks, either “path-following” or “target-reaching”, we defined two compound parameters describing the average performance across subjects, both in terms of penetration depth and time:

- The *average penetration peak*, APP. It indicates the average maximum penetration depth of the tool tip into the constraint;
- The *average time peak*, ATP. It indicates the average time required to reach the maximum penetration depth APP.

For “path-following” tasks $APP = 1.28mm$, while $ATP = 1.1s$ (Table 3.1). It means that each subject, on average, pushes into the constraint up a penetration deepness of $1.28mm$. The average time required to reach this penetration level is $1.1s$.

Path-following task	
Average Penetration Peak $\boldsymbol{APP}$	$1.28mm$
Average Time Peak $\boldsymbol{ATP}$	$1.1s$

Table 3.1 Average penetration peak and average time peak, for "path-following" tasks.

The average penetration peak and the average time peak represent a discriminatory point between two distinct phases of the violation: the rising and descending phases of the force signal (Figure 3.2). In the first phase, the user actively pushes against the constraint to circumvent an obstacle. During the second phase, after the penetration peak has been overcome, the user passively goes back to the path under the action of the feedback force. A good classifier should be able to detect intentional violations during the rising phase of the force signal, such that the assistance level is adjusted according to the current will of the user. High sensitivity levels provided during the descending phase might not be useful, as the assistance level is provided too late with respect to the user's current action.

3.1.1 HMM-based classifier

The HMM-based classifier yields a classification output every 35ms. For each segment, the model outputs two likelihood functions ϕ_{int} and ϕ_{unint}, related to the probability that the current observation belongs either to the intentional-HMM or unintentional-HHM. In Figure is shown a comparison between the likelihood functions both is case of intentional and unintentional violations. For unintentional violations, there is not a significant distinction between the likelihood functions ϕ_{int} and ϕ_{unint}: this is due to the fact that the distribution of the force signal can either be interpreted as an unintentional violation or the tail of an intentional violation. On the other hand, for intentional violations, the

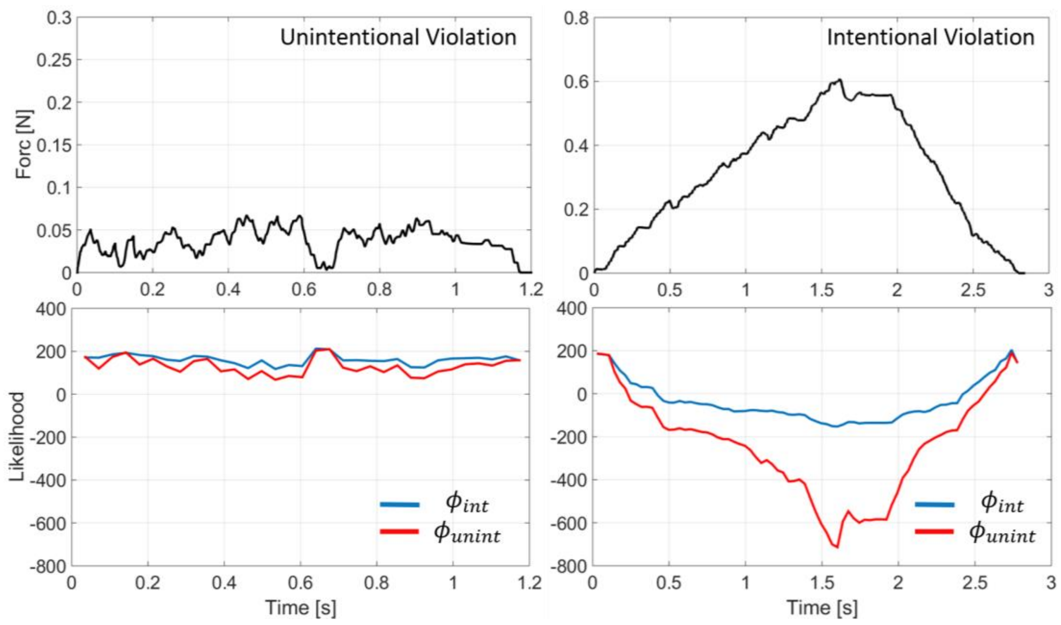


Figure 3.4 Likelihood functions both in cases of intentional and unintentional violations. In the case of intentional violations, the likelihood of the unintentional HMM (red line) decrease after the violation has started, reaching its minimum after 0.5s.

difference in the likelihood functions reaches significant values after 500ms: after this time, the likelihood of the unintentional model decreases, reaching a minimum point as the force signal reaches its maximum point. Conversely, the likelihood function of the intentional model is high, meaning that the violation has been correctly detected. The performance of the HMM-based classifier is visualized by means of the ROC curves, shown in Figure 3.6.

The Receiver Operating Characteristics (ROC) curves evolve over time: in the interval $[100, 1000]ms$, the area under the curves increases, yielding better classification performance: during the first time instants of the violation, the $100ms$-ROC (red curve) is near the bisector, indicating poor classification results. However, as the time interval increases, the ROC curves tend to the ideal classifier, represented by the point $P(0, 1)$.

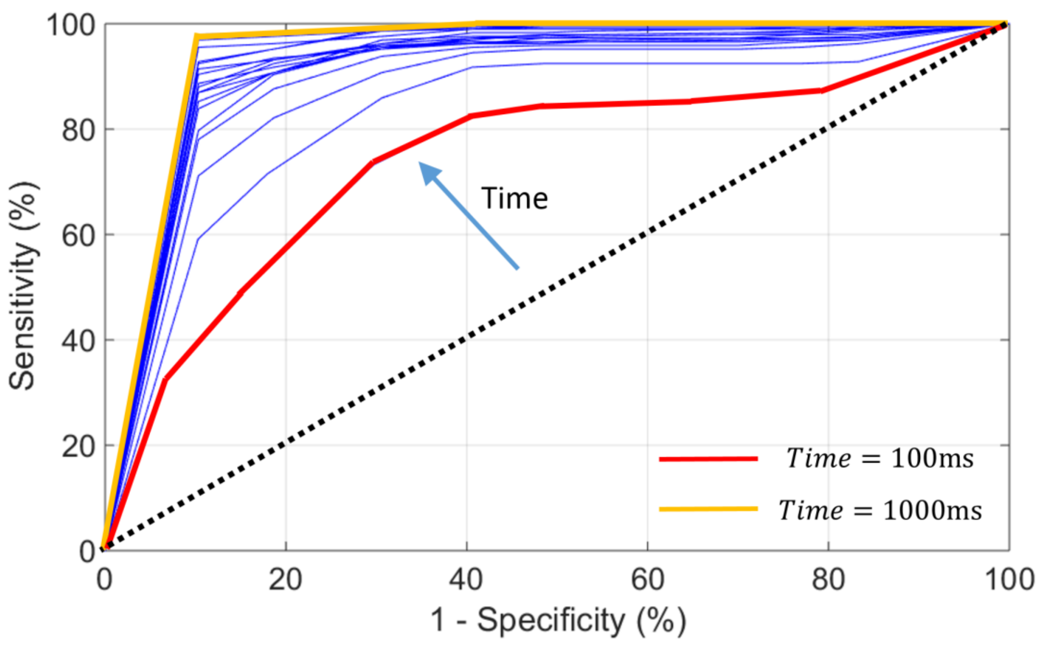


Figure 3.6 Evolution of the ROC curves over time.

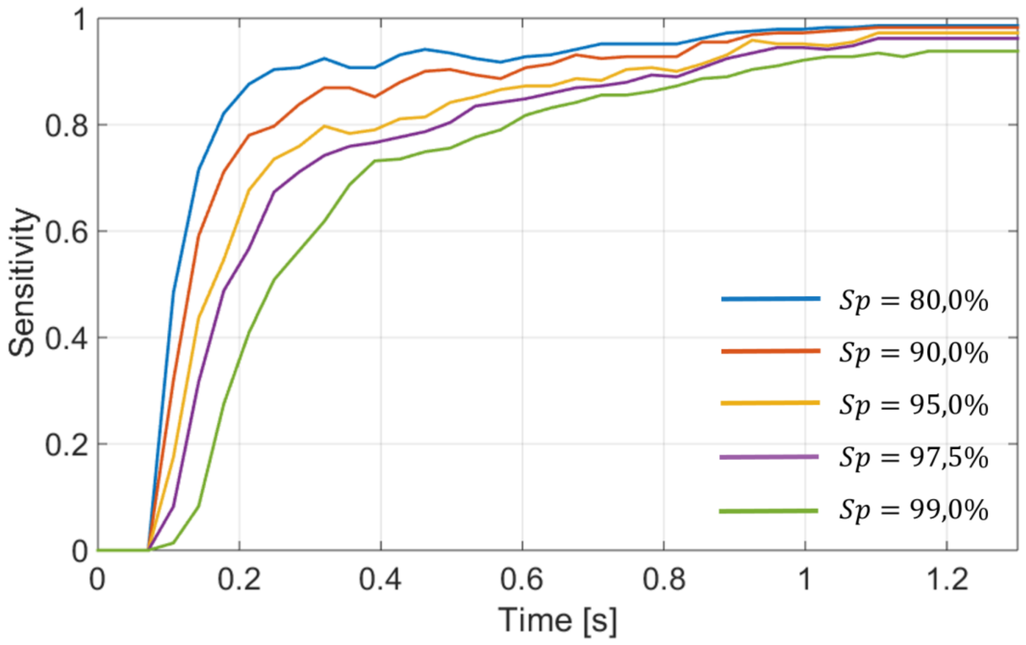


Figure 3.5 Sensitivity profiles over time, given five specificity levels.

In Figure 3.5 are shown five sensitivity profiles over time, corresponding to five specificity levels: 80%, 90%, 95%, 97.5% and 99%. In general, an increase in the required specificity level corresponds to a shift in time of the coupled sensitivity profile: the minimum interval time necessary to overcome a sensitivity level $Sp = 90\%$ widens, meaning that the classifier has slower detection times. Conversely, a decrease in the specificity level, down to $Sp = 80\%$, yields a faster model, which is able to detect 90% of intentional violations within $250ms$ after the interaction has started.

In Table 3.2 the time-sensitivity values corresponding to Figure 3.5 are displayed.

SPECIFICITY	**TIME [s] ($Sp > 90\%$)**	**VIOLATION [mm]**
80%	0.284	0.48
90%	0.604	0.94
95%	0.747	1.01
97,5%	0.854	1.18
99%	0.924	1.23

Table 3.2 Time and penetration depth values associated with different specificity levels.

It is clear how, requiring higher performances in terms of specificity, has a delay effect on the corresponding sensitivity profiles. However, the HMM-based classifier is characterized by sensitivity levels exceeding 90% within $1s$ after the violation has started, for all the five specificity level set. The time information is associated with of amount of penetration depth into the constraint. By comparing the values in Table 3.2 with the "average penetration peak" and "average time peak" values, from Table 3.1, it is clear how, in all five cases, the required sensitivity level is reached before either the average penetration peak or average time peak are overcome. This means that the HMM-based classifier is able to detect intentional violations during the rising profile of the force, while the subject is actively pushing against the constraint to circumvent the obstacle.

3.1.2 NN-based classifier

3.1.2.1 Statistical Classifier - StNN model

The StNN model, is a Neural Network-based classifier trained on the basis of seven statistical features, extracted from the force segments (section 2.3.2.2.1) of the interaction force. In Figure 3.7 are shown two examples of the output profile of the neural network, both in case of unintentional and intentional violations. In the first case (Figure 3.7, left panels), the output of the network asymptotically tends to -1: hence, the unintentional violation is correctly detected. In the second case (Figure 3.7, right panels), the network output shows a sigmoidal profile ranging from -1 to $+1$. This means that, during the first milliseconds of interaction, the network is not able to distinguish the intention. However, as the violation goes on, the output level increases, reaching +1 within 1s after the

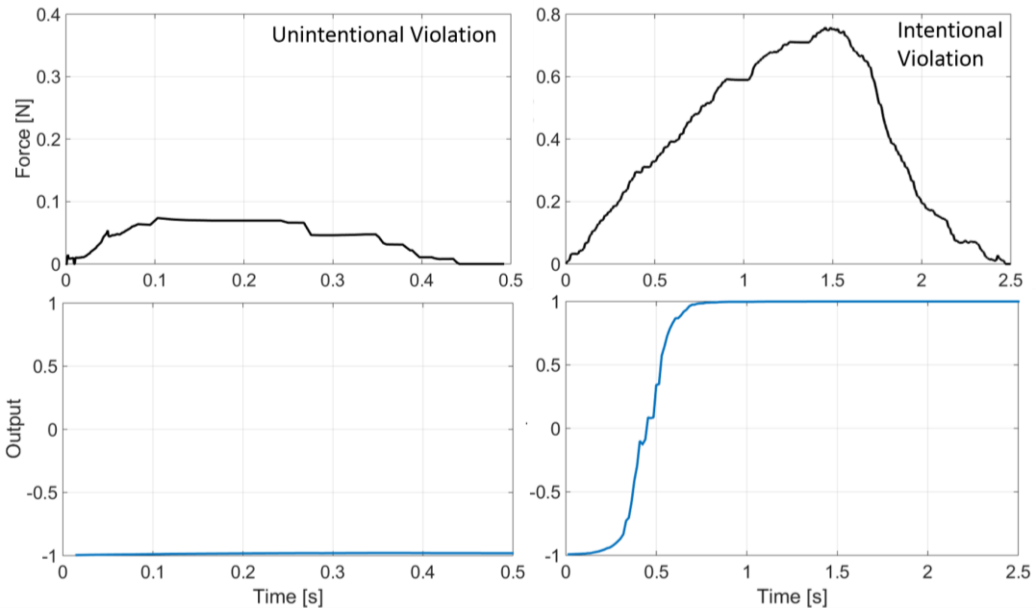


Figure 3.7 Example of the neural network output, both in case of intentional and unintentional violations.

violation has started. The performances of the StNN model is visualized in Figure 3.8. The ROC curves show a similar time profile with respect to the HMM-based classifier. As the violation time widens, the area under the ROC curves increases,

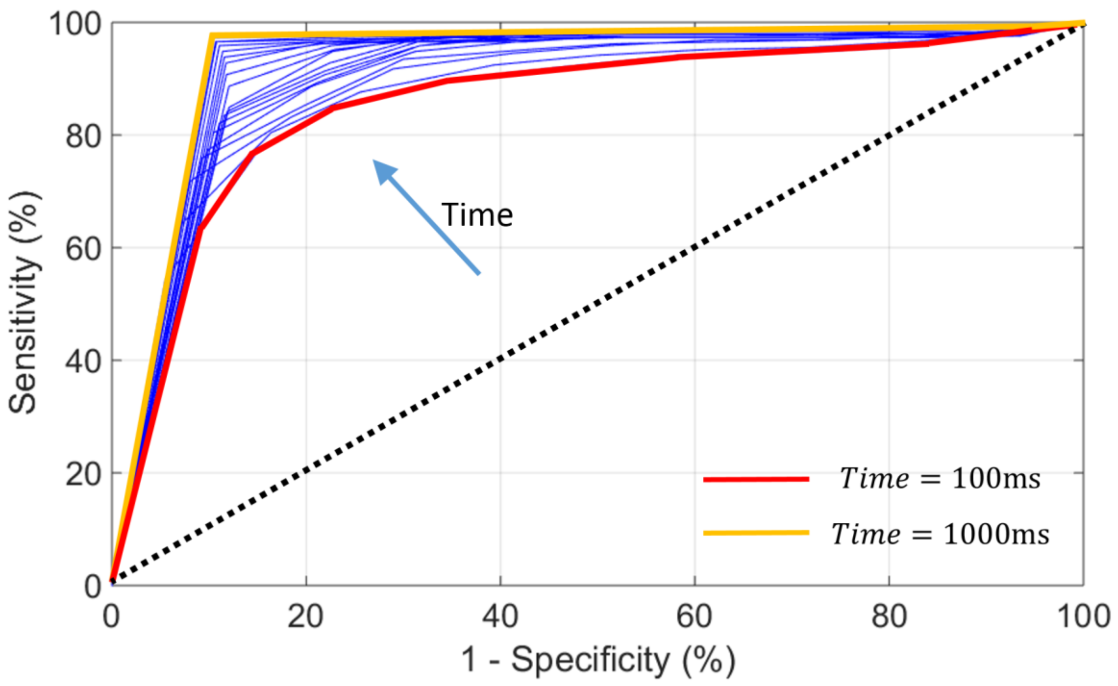


Figure 3.8 Time evolution of the ROC curves for the StNN model.

yielding better classification results. Unlike for HMM model, the 100ms-ROC curve is significantly distinguishable from the bisector, meaning that the StNN classifier provides a good classification performance within the first $100ms$ of interaction.

Figure 3.9 shows the five sensitivity profiles over time, computed for five specificity levels of 80, 90, 95, 97.5 and 99%. Each curve is characterized by a sigmoidal profile with a rising slope that depends on the specificity level set. For specificity levels up to 97.5%, the sensitivity index exceeds 90% value within $900ms$ after the intentional violation has started. On the contrary, a specificity of 99% decreases the slope of the sensitivity that exceeds the threshold after$1.1s$.

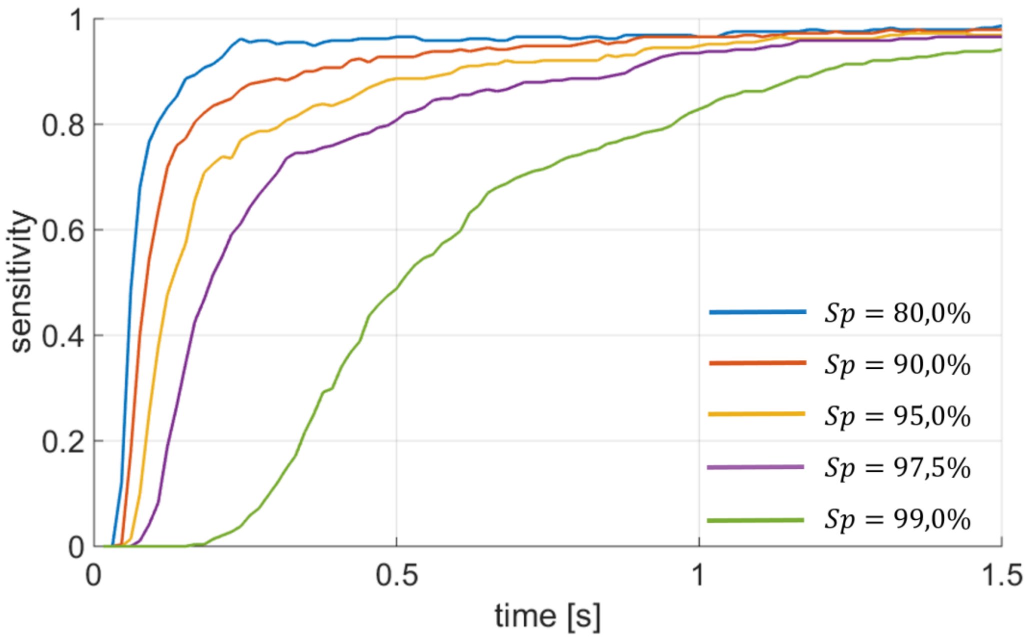


Figure 3.9 Sensitivity profiles over time for the StNN classifier.

In Table 3.3 are displayed the time-sensitivity pairs associated with the curves displayed in Figure 3.9. Similarly to the HMM-based classifier, the sensitivity profiles exceed 90% before either the average penetration peak or average time peak are overcome (Table 3.3), meaning that the classifier detects intentional events during the rising phase of the force, when the user is pushing to go past the obstacle.

SPECIFICITY	TIME [s] ($Sp > 90\%$)	VIOLATION [mm]
80%	0.172	0.31
90%	0.346	0.57
95%	0.599	0.93
97,5%	0.893	1.21
99%	1.197	1.27

Table 3.3 Time and penetration depth values associated with different specificity levels.

3.1.2.2 Spectral Classifier – SpNN

The SpNN model was trained on the basis of the energy distribution across different frequency bands of each force segments, decomposed through the Discrete Wavelet transform algorithm. The time performances of the classifier are displayed in Figure 3.10. The ROC curves of the model have a similar time evolution with respect to both the HMM-based and StNN methods: the classification performance improves as the violation time increases. However, by analyzing the $100ms$-ROC curve (red curve), it is possible to note that, by setting a high specificity level ($Sp > 85\%$), we obtain poor performances in terms of sensitivity level (below 20%). This means that, during the first milliseconds of violation, the model is able to detect less than 20% of intentional interactions. This behavior characterizes all the ROC curves of the time interval considered, up to $1s$ (yellow curve), meaning that the classifier has a stricter trade-off at high specificity values. On the contrary, low specificity levels ($Sp < 50\%$) yield

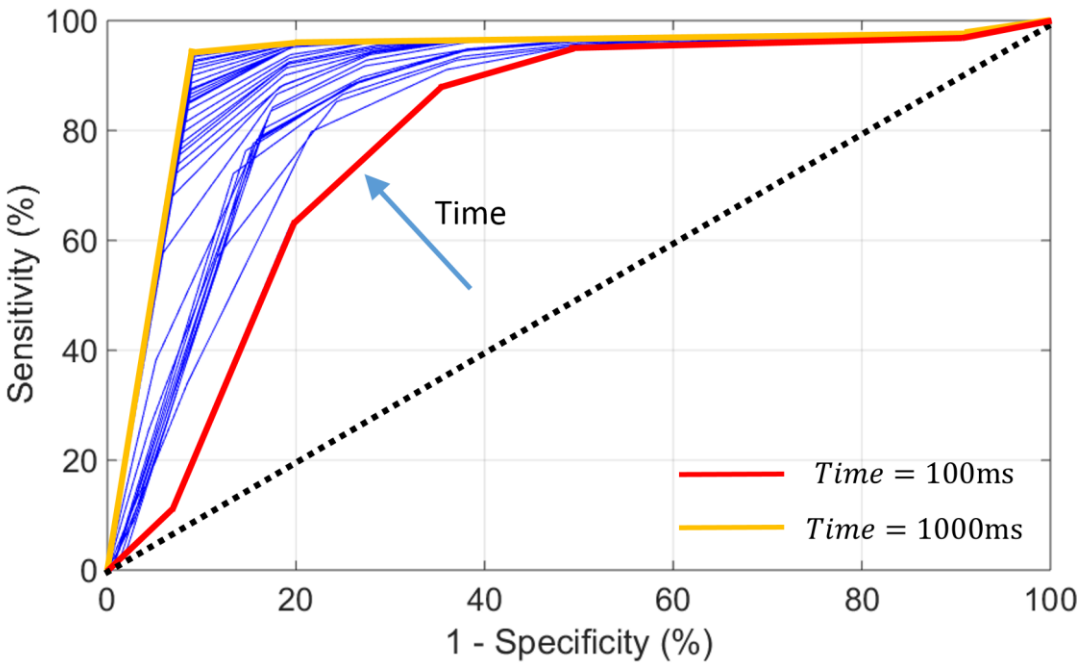


Figure 3.10 Evolution of the ROC curves for SpNN classifier.

corresponding sensitivity indexes exceeding 90%, across all the time span taken into account $[0.1, 1]s$.

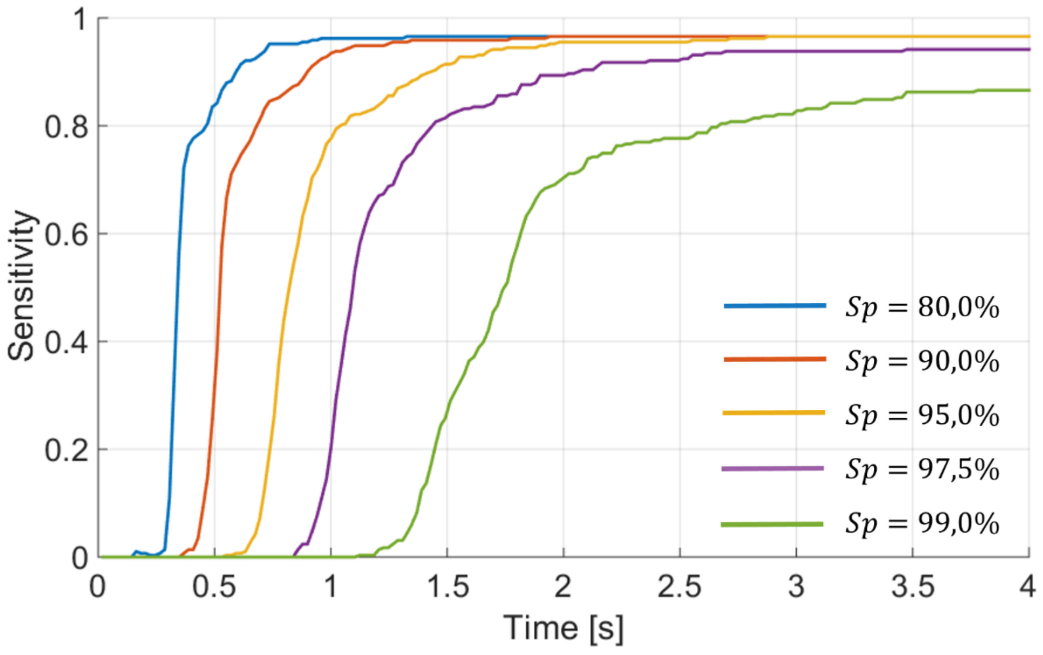


Figure 3.11 Sensitivity profiles over time for the SpNN model.

In Figure 3.11 are plotted the time profiles of the sensitivity index. In Table 3.4 the relative time-penetration values are listed.

By comparing the performances of the SpNN model, both with respect to the HMM and StNN models, we notice that the spectral classifier is characterized by worse classification performances. As displayed in Table 3.4, for specificity levels $Sp = 95$, 97.5 and 99%, the minimum time required to reach a sensitivity level exceeding $Se = 90\%$ is higher than the average time peak value, $ATP = 1100ms$ (values flagged with *). This means that the model detects intentional violations during the descending phase of the force signal, after the violation peak has occurred and the user is almost past the obstacle. In particular, by setting a specificity level $Sp = 99\%$, the corresponding sensitivity never exceed 90%, asymptotically tending to $Se = 86.6\%$, after a time of $t = 3.5s$.

SPECIFICITY	TIME [s] $(Sp > 90\%)$	VIOLATION [mm]
80%	0.582	0.91
90%	0.895	1.21
95%	1.420*	1.194*
97,5%	2.049*	0.73*
99%	3.580* $(Se = 86.6\%)$	0.11*

Table 3.4 Time-penetration values corresponding to sensitivity level of 90%. The values marked with *, correspond to specificity levels that have not met the requirements.

3.2 Reaching task

12 subjects executed 120 “target-reaching” tasks. As in the previous section 3.1, we define two compound parameters that describe the average behavior of the subjects, in terms of maximum average penetration peak APP, and average time peak ATP. For “target-reaching” tasks, the value are listed in Table 3.5.

Target-reaching task	
Average Penetration Peak $\boldsymbol{APP}$	$0.75mm$
Average Time Peak $\boldsymbol{ATP}$	$2.03s$

Table 3.5 Average penetration peak and average penetration time for "target-reaching" tasks.

In Figure 3.12 is shown an example of the execution of the task. The dashed red line represents the position of the tool tip of the manipulator, while the green markers represent the equally distributed target points that each user is meant to reach.

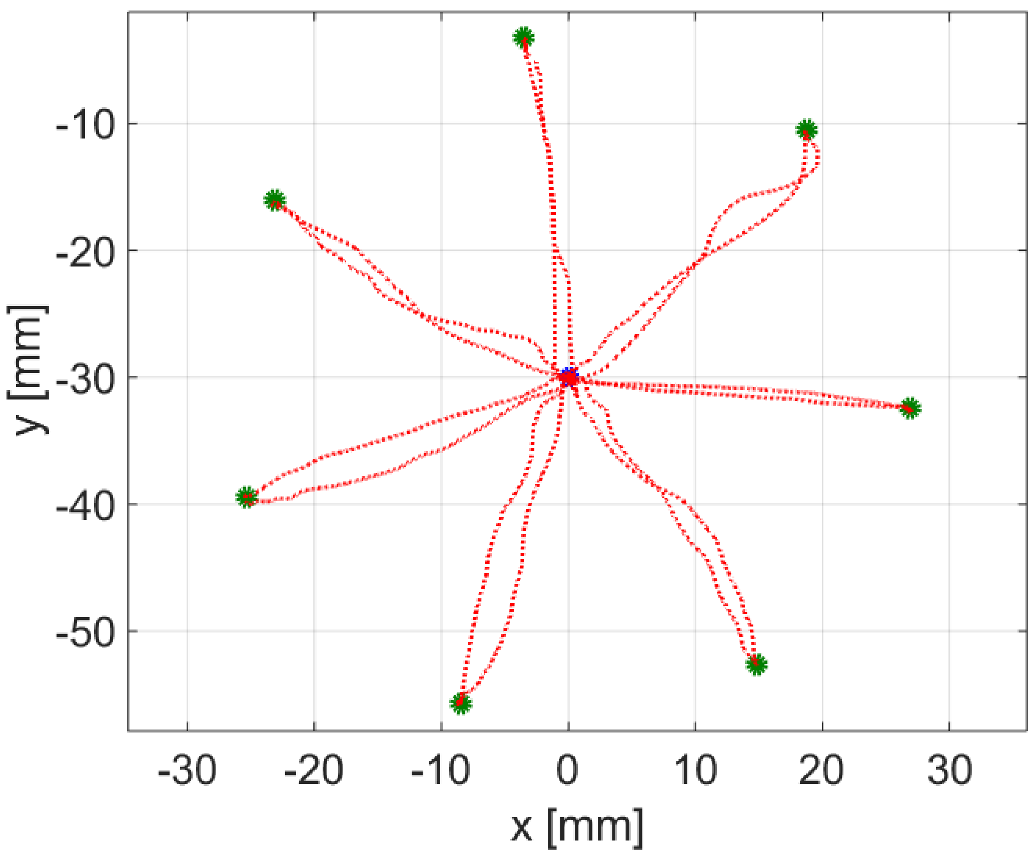


Figure 3.12 Execution of a target-reaching task. The dotted red line represents the path of the tool tip of the haptic device, while the green points represent the targets that the user is asked to reach.

In Figure 3.13 is plotted the corresponding magnitude of the interaction force while the user performs the reaching movements to position the tool tip onto each green point.

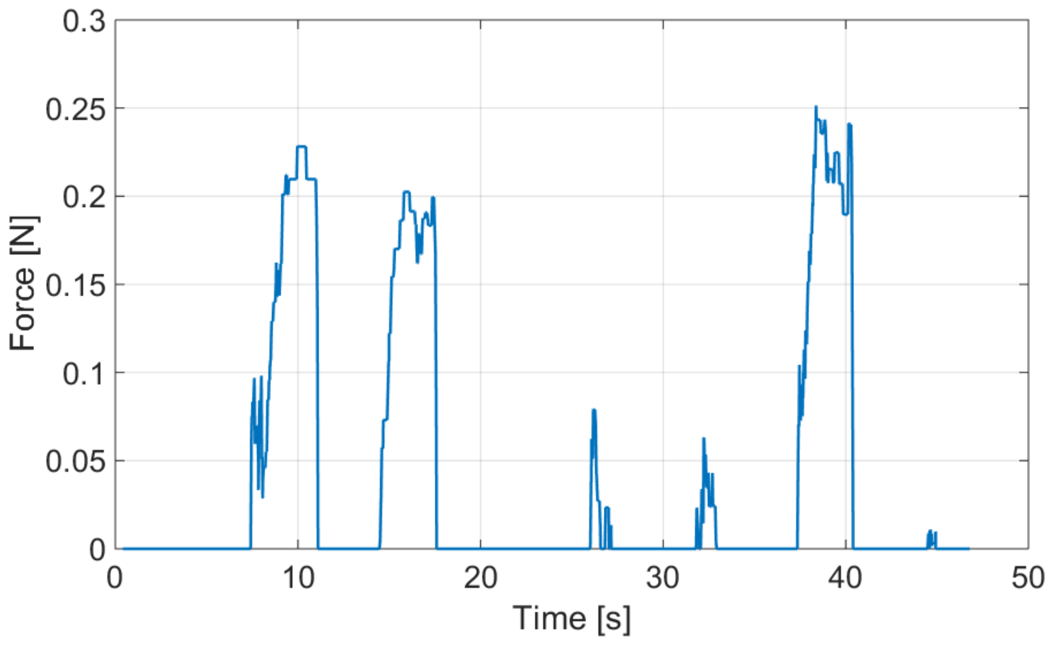


Figure 3.13 Modulus of the interaction force for the reaching task.

3.2.1 HMM-based classifier

The HMM-based classifier yields a classification output every 35ms. An example of the two likelihood functions ϕ_{int} and ϕ_{unint}, is shown in Figure 3.15, both in the case of intentional and unintentional violation. Similarly to the path-following case, the HMM model outputs two very similar likelihood functions when analyzing unintentional segments (left panels). This is due to the fact that observation vectors extracted from unintentional segments might be misidentified as the tails of intentional segments, yielding a very poor distinction of likelihood functions associated to the HMM models. Conversely, a significant distinction of the likelihood levels is detected in the case of intentional violation, (right panels) after a time $t > 500ms$. The distinction become clearer as the modulus of the interaction force reaches a minimum level that of magnitude, which allows for the correct detection of the event.

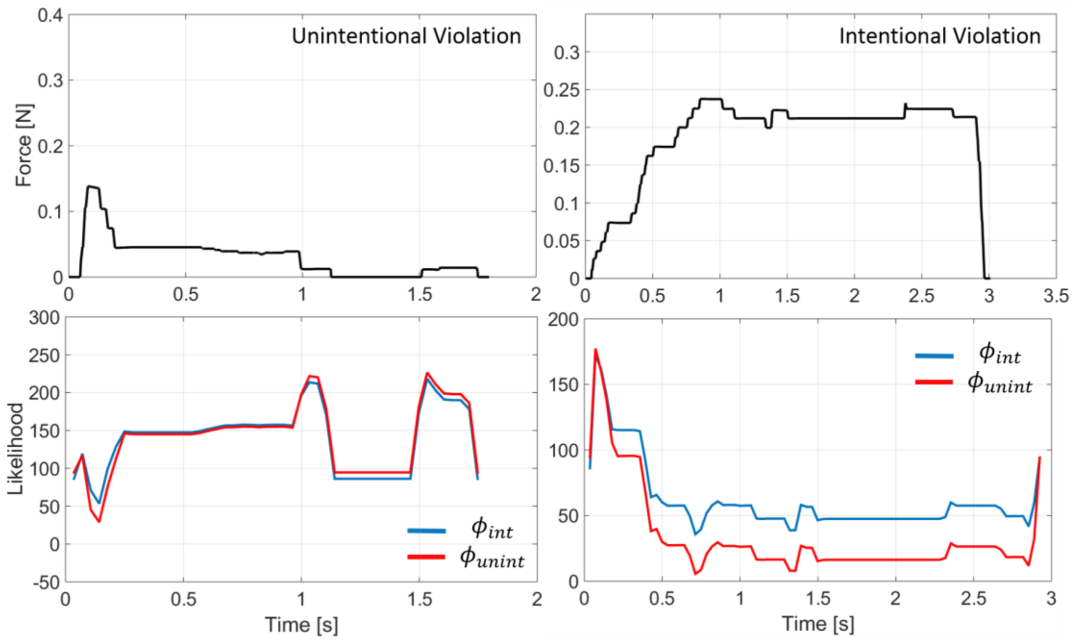


Figure 3.15 Likelihood functions read off from the model, both for intentional (right panels) and unintentional (left panels) violations.

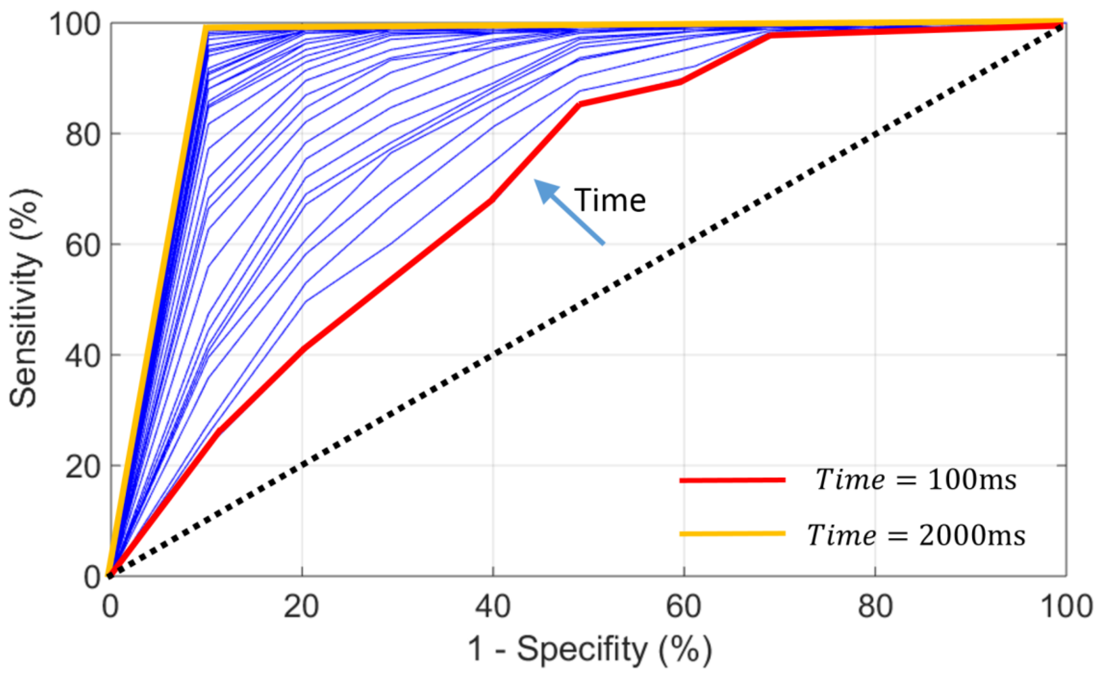


Figure 3.14 Time evolution of the ROC curve associated to the HMM classifier.

The classification performance of the HMM-based model is shown in Figure 3.14. The time evolution of the different ROC curves is evaluated within the time interval $[0.1, 2]s$. During the first milliseconds of violation (red curve), the classifier is characterized by poor performances: the $100ms$-ROC curve lies closely to the bisector: hence, high sensitivity levels are obtained by accepting low specificity values, $Sp < 40\%$. By setting the specificity index $Sp > 80\%$, the corresponding sensitivity index decreases down to $Se < 40\%$. By widening the interaction time, better performance are achieved at the expenses of slower classification results. Around $2s$ (yellow curve) after the violation has started, the ROC curve asymptotically tend to the ideal classifier, yielding sensitivity levels exceeding 90%, on top of specificity levels $Sp > 90\%$.

In Figure 3.16 are plotted the sensitivity profile of the HMM-based classifier, corresponding the set specificity values of 80, 90, 95, 97.5, 99%.

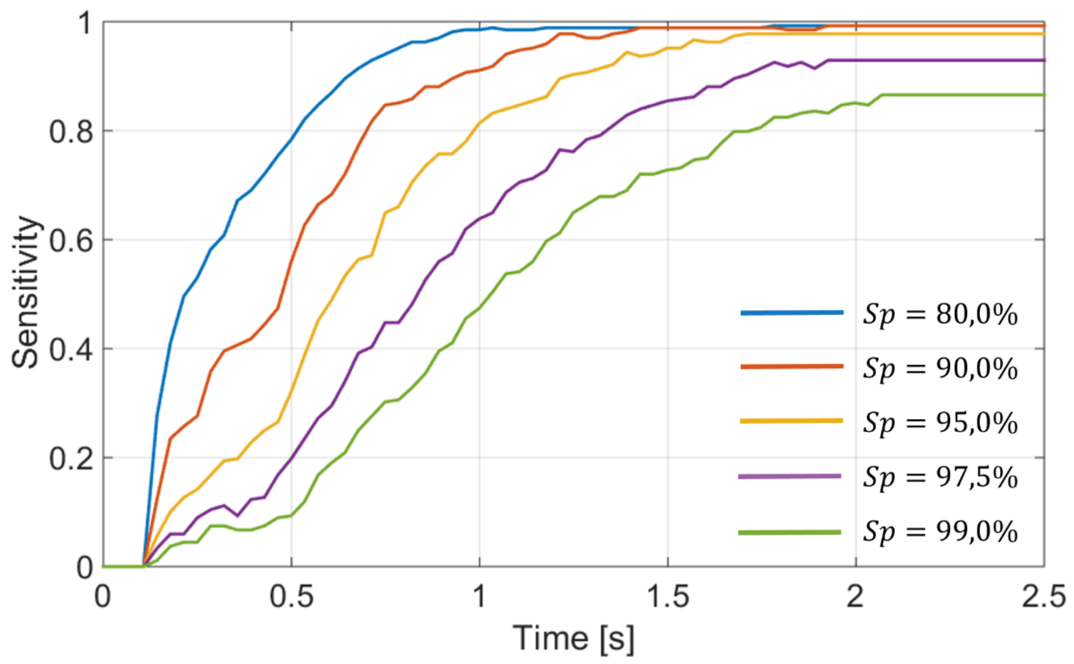


Figure 3.16 Sensitivity profiles over time, given five specificity levels.

In Table 3.6 are displayed the corresponding time-violation parameters.

SPECIFICITY	**TIME [s]** $(Sp > 90\%)$	**VIOLATION [mm]**
80%	0.682	0.47
90%	0.968	0.55
95%	1.253	0.62
97,5%	1.788	0.67
99%	2.068* $(Se = 86.2\%)$	0.71*

Table 3.6 Time-penetration values corresponding to five specificity levels.

By comparing the time-penetration values with the "average penetration peak" and the "average time peak" values, listed in Table 3.5, we can see how the HMM-based classifier is able, in four cases, to reliably detect intentional violations with a sensitivity exceeding 90%, before maximum penetration depth $APP = 0.75mm$ is reached. On the other hand, if we set a specificity level $Sp = 99\%$, the sensitivity exceed the threshold during the descending phase of the force, after the penetration peak has occurred. This event is clear also by comparing the time instant in which $Se > 90\%$, $t = 2.068s$, with the average time peak value $ATP = 2.03s$.

3.2.2 NN-based classifier

3.2.2.1 Statistical Classifier – StNN

The statistical classifier has been applied to classify the interaction segments extracted from the target-reaching task. An example of the network output is displayed in Figure 3.17.

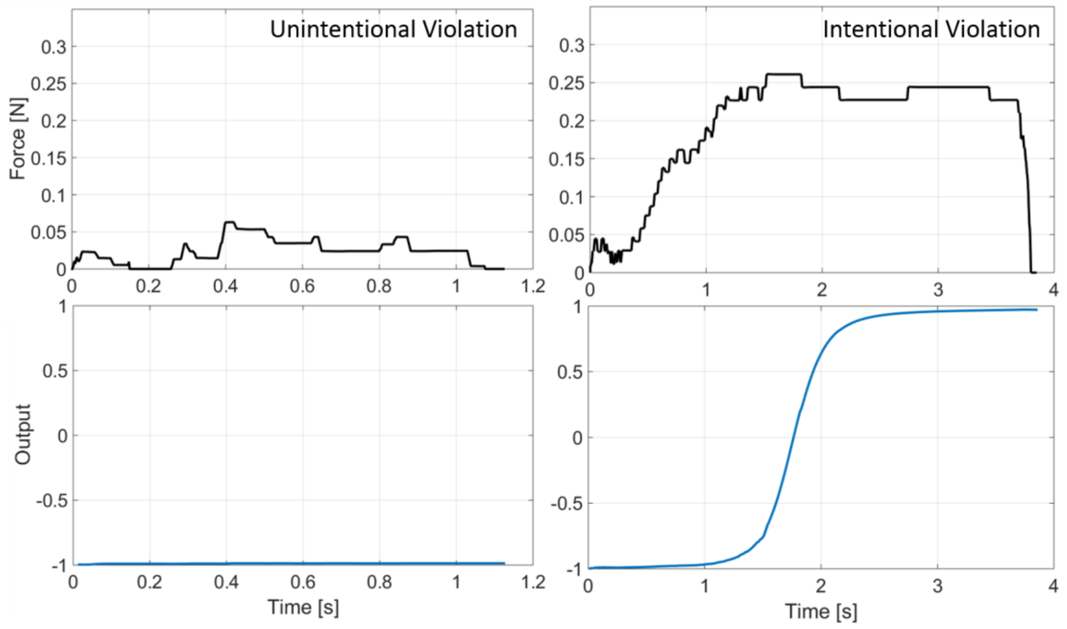


Figure 3.17 Classification outputs for the statistical classifier. On the left panels is shown an example of unintentional violation, while in the right panel is shown an example of intentional violation

Like in the "path-following" case (Figure 3.7), the StNN model has a similar behavior: for unintentional interactions (left panels), the output asymptotically tends to -1; on the contrary, for intentional violations (right panels), the output asymptotically tends to +1 with a sigmodal profile.

The ROC curves associated to the StNN classifier are visualized in Figure 3.18. During the first instants of the violation, the classifier is characterized by a poor performance, as the $100ms$-ROC curve lies closely to the bisector. In addition, unlike the HMM model (Figure 3.14), for low specificity levels $Sp < 30\%$ the curve is almost superimposed to the bisector meaning that, the deployment of the classifier after $100ms$ time after the violation has started provides unreliable results for any specificity level. By widening the time interval, the ROC curve tends to the ideal classifier: after $2s$ the classifier is able to detect intentional violations with a sensitivity exceeding 95%, on top of a specificity level $Sp > 90\%$.

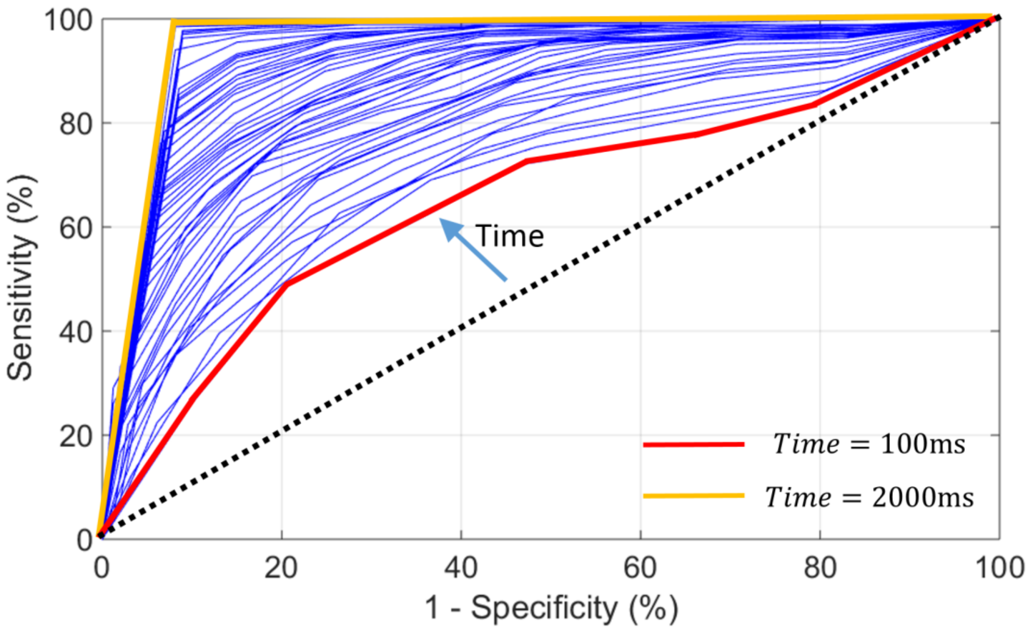


Figure 3.18 ROC curve of the statistical classifier, for target-reaching tasks.

In Figure 3.19 are plotted he sensitivity profiles over time, corresponding to five specificity levels $Sp =$ 90, 80, 95, 97.5 and 99%. Like in the previous models, the different profiles of the sensitivity undergo a shift in time, resulting from an increase in the specificity required. In general, all the five profiles analyzed asymptotically reach a sensitivity level exceeding 97% after a time $t = 2s$. In particular, the 99%-curve (green curve) is characterized by a much lower slope, meaning that the classifier has been slowed down, due to the strict requirements we put on the specificity index: hence, the curve exceeds the 90% level after $1.7s$.

In Table 3.7 are listed the associated time-penetration values for different specificity levels. By comparing these values with the average penetration and time peaks (Table 3.5), we see how, in all cases analyzed, the sensitivity exceeds 90% value before the penetration peak value is reached. Hence, intentional interactions are detected during the rising phase of the force.

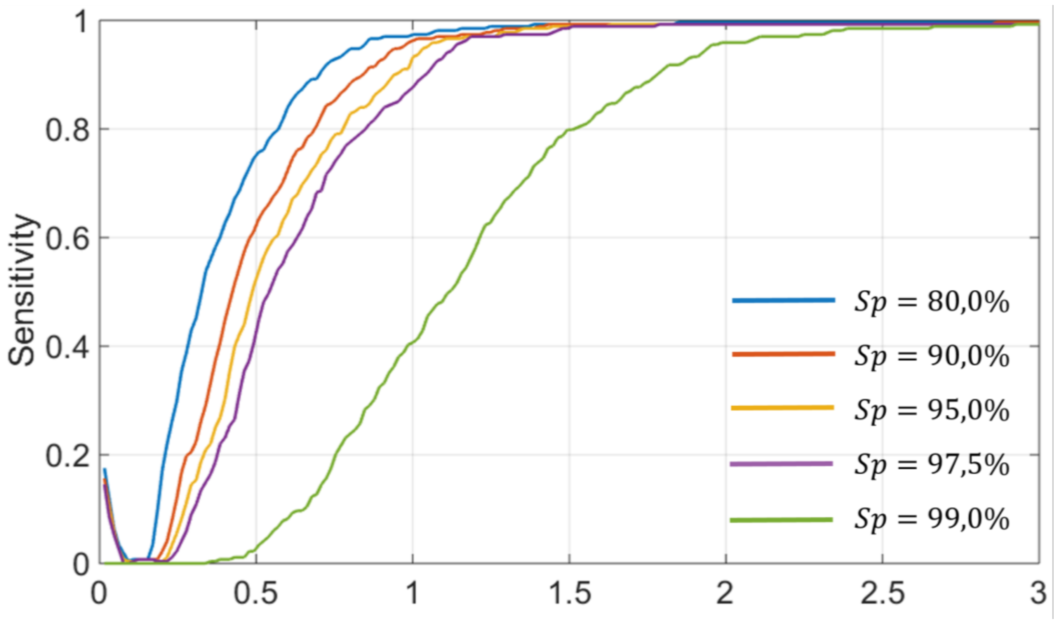


Figure 3.19 Sensitivity profiles corresponding to five specificity levels

SPECIFICITY	TIME [s] ($Sp > 90\%$)	VIOLATION [mm]
80%	0.699	0.47
90%	0.837	0.52
95%	0.946	0.55
97,5%	1.040	0.57
99%	1.778	0.67

Table 3.7 Time-penetration values related to the five sensitivity profiles.

3.2.2.2 Spectral Classifier – SpNN

The spectral classifier has been trained on the basis of the energy distribution across the different levels of the Wavelet decomposition. In general the SpNN model is characterized by a worse performance, both compared to the HMM and StNN classifies. The ROC curves of the spectral classifier are plotted in Figure 3.20.

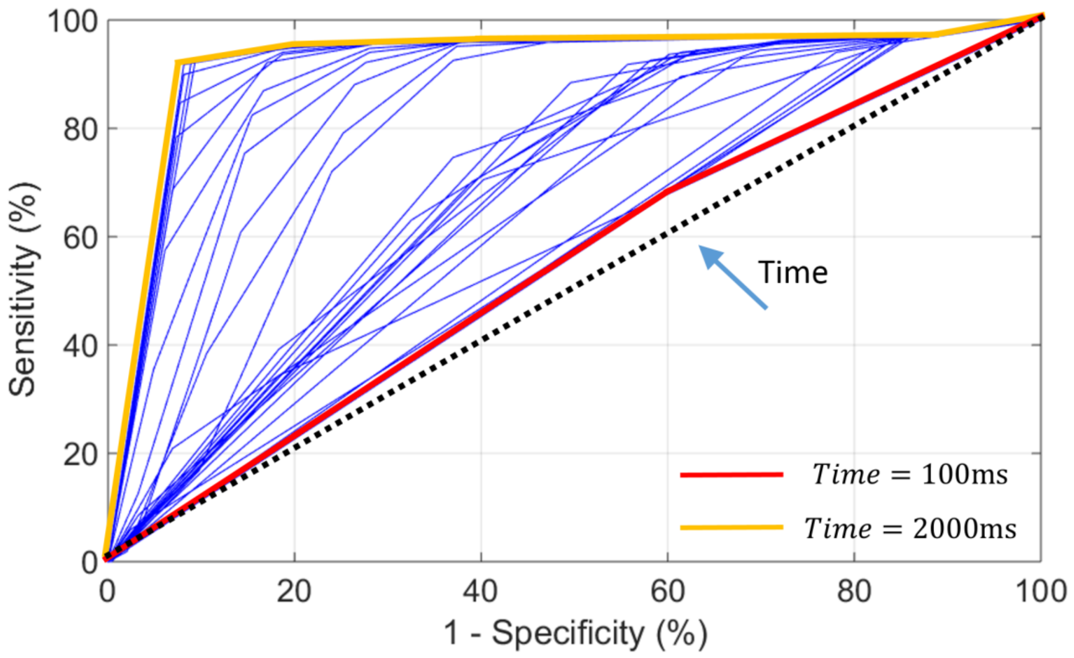


Figure 3.20 ROC curves describing the performances of the SpNN classifier.

During the first instants of interaction, the classifier is not able to output reliable results: the $100ms$-ROC curve (red curve) is almost superimposed to the bisector (black dotted line), meaning that the model performances are comparable to those of a random classifier. However, similarly to the other cases, the classification performances improve as the time interval widens. Within two seconds of interaction (yellow line), the classifier is characterized by a sensitivity index exceeding 90%, coupled with a specificity level around $Sp = 90\%$.

In Figure 3.21 are shown the sensitivity profiles corresponding to specificity levels of 80, 90, 95, 97.5 and 99%.

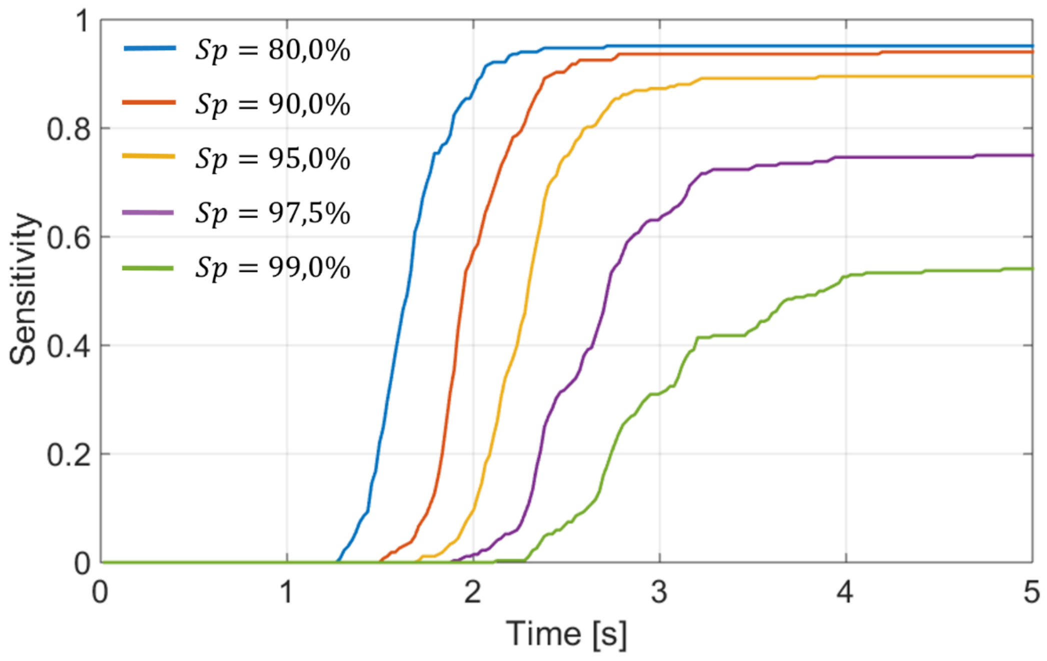


Figure 3.21 Sensitivity profiles for different specificity levels.

SPECIFICITY	TIME [s] ($Sp > 90\%$)	VIOLATION [mm]
80%	2.051	0.68*
90%	2.426	0.64*
95%	3.220* $Se = 89\%$	0.40*
97,5%	3.942* $Se = 75\%$	0.14*
99%	4.115* $Se = 53\%$	0.10*

Table 3.8 Time-penetration values corresponding to different sensitivity profiles.

Correspondingly, the time-violation pairs are listed in Table. By comparing these values with the average penetration peak and average time peak values (Tabe 3.5), it is clear how the SpNN method applied to target-reaching tasks is not able to meet the requirements we desire. For all the five specificity levels analyzed, the sensitivity exceeds the 90% threshold after the penetration has occurred. This means that the method is not capable of yielding reliable detection of intentional events during the rising phase of the force signal. A specificity level $Sp = 80\%$, yields a correspondent sensitivity profile that overcomes the threshold $30ms$ after the peak has occurred. A further increase in the specificity index slows down the classifier. In addition, if we require a specificity level $Sp > 95\%$, the corresponding sensitivity level never overcomes the threshold, asymptotically tending to lower values.

Discussions and future works

The objective of this work is framed within the context of adaptive active constraints for cooperative tasks. As discussed in section 1.5.1, there is a number of surgical applications, like beating heart surgery and endoscopic procedures, which require that the geometry of the constraint will be constantly updated over time, in order to adapt to the geometry of the soft tissue. In other cases, it might be useful to update the constraint definition on the basis of the interpretation of the user's commands and adjust the assistance level provided, accordingly. The latter case is the one we chose to investigate. Our work focused on the investigation of whether intentional and unintentional violations of active constraints might be distinguished on the basis of the interaction force signal. Intentional violations occur whenever the current action of the human operator is in contrast with the purpose of the constraint. This conflicting situation results in significant interaction forces at the tool tip of the robotic manipulator, as the constraint hinders the execution of the task. On the other hand, unintentional violations occur whenever the constraint shares the purpose of the human operator, bur inherent errors are made anyway. We believed that, the detection of intentional violations should, in turn, be used to trigger the modulation the assistance level provided by the constraint, which might be removed, allowing for a complete free motion around the workspace, or its stiffness might be proportionally reduced. We proposed several detection methods based on the definition of three binary classifiers:

- A continuous HMM-based classifier, whose training set is composed by observation vectors extracted directly from the interaction force signal;
- A NN-based classifier, trained on the basis of statistical features that represent the time evolution of the force signal;

- A NN-based classifier, trained on the basis of spectral features, extracted from the Wavelet decomposition of the force segments.

We decided to investigate the classification performance of each methodology in two different cases:

- “Path-following” tasks. In this case, the subject is asked to manipulate the haptic device to follow a 2D trajectory displayed on a screen. Along the path, a number of circular obstacles is placed, such that the subject is forced to intentionally depart from the “guidance constraint”, to externally circumvent them.
- “Target-reaching” tasks. In this case, the subject is asked to manipulate the haptic device to place the pointer onto a number of equally spaced target points. For some of these points, the “regional constraint” is wrongly enforced and the target lies a forbidden region, forcing the subject to intentionally act against the constraint to accomplish the reaching task.

We built a set of 20 experimental tasks, from which classifiable data was generated: 10 “path-following” and 10 “target-reaching” tasks. We asked 12 subjects, aged between 20 and 30, to perform 20 trails each. The experimental setup was composed of three main elements: the haptic interface “Phantom Omni”, a display to provide visual feedback, and the control console, from which the operator supervised the execution of the trials. 240 tasks were executed and two distinct dataset were built, in order to separately evaluate the classification results in case of both “guidance” and “forbidden region” constraints. Both dataset were composed of force signals measured at the tool tip of the haptic device, thus describing the magnitude of the interaction with the constraint. The recorded signals were split into segments, according to the existence of the interaction ($\boldsymbol{f} > 0$) and were automatically labelled as “intentional” ($+1$) or “unintentional” (-1) on the basis of the current position of the tool tip with respect to the geometry of the constraints and/or the obstacles. The labelling process was necessary for the subsequent supervised learning of the NN-based classifiers and for the validation phases.

The classification performance of each methodology was evaluated in terms of:

- Time evolution of the Receiver Operating Characteristics (ROC) curve;
- Minimum amount of time t_{min} required, after the violation has started, to reliably apply the classifier and detect intentional violations with a sensitivity level exceeding $Se = 90\%$;
- Penetration depth associated to the time t_{min}.

In general, it was observed that the time evolution of the ROC curves has a similar profile across the different classifiers: as expected, during the first instants of the interaction (within $200ms$), the amount of information recorded is not sufficient to distinguish between the two kinds of interaction with high sensitivity. However, as we widen the interval time, the ROC curves asymptotically tend to the ideal classifier, that is represented by the point (0, 1). The proper time instant in which to apply the classifier has to be chosen on the basis of the specific requirements of the application. As discussed in section 2.4, we believe that, for surgical procedures, it might be advisable to deploy classification methods characterized by a high specificity level. This choice is inherently safer, as the classifier is able to detect and reject unintentional violations, whose misclassification might lead to unwanted adaptation of the assistance level. On the contrary, misclassification of intentional violations, hence a low sensitivity index, has no effect on the nature of the constraints. It is clear how, for runtime identification of intentional violations, the value of the sensitivity index is dependent on the time window we choose in order to apply the classifier: we expect that, given a certain specificity level, if we increase the observation window, we will consequently obtain a significant increase in the corresponding sensitivity level. This is due to the fact that the amount of input information becomes more meaningful as the intentional violation is taking place.

These considerations lead us to the evaluation of the sensitivity profiles over time, for each classification methodology. We set five different specificity indexes $Sp = 80$, 90, 95, 97.5 and 99%. Consequently, we computed the time evolution of the sensitivity and evaluated the time instant in which this index exceeded a threshold of 90%. Results showed that, for all the classification methodologies investigated, there exists a trade-off between the specificity level we require, and the slope of the associated sensitivity profile. Higher specificity requirements lead to a decrease in the slope of the associated sensitivity profile. Consequently, the overall classification process is slowed down, because a wider time window is necessary to exceed the 90% threshold. Vice versa, a decrease in the specificity level has the effect of speeding up the classification process, as the slope of the sensitivity is increased and the threshold is overcome within a narrower time interval. As discussed in section 2.4, separate evaluations were carried out for "guidance" and "forbidden region" constraints and in both cases, the performance of the three classifiers were investigated. A comparison across different methodologies showed that in general, for a given specificity level, the statistical classifier (StNN) is characterized by a faster increase in the sensitivity profile with respect to HMM-based method, while the spectral (SpNN) classifier yields the slowest performances. This means that, during the first time instants after the interaction has started, the statistical classifier, on average, is able to detect intentional violations with higher sensitivity with respect to the other methods. An additional feature that we investigated is whether the classification methods are able to correctly distinguish the intentions during the rising phase of the force signal, when the user is actively pushing against the constraint to either circumvent an obstacle or to reach a target. The identification of the intention after the penetration peak has occurred might not be useful, as the adaptation of the assistance level is provided too late. In general, for $Sp < 97.5\%$, the StNN and the HMM classifiers are able to detect intentional violations before the occurrence of the penetration peak. On the contrary, the spectral classifier shows poor performance when high specificity levels are required. For “path-following” tasks, the SpNN classifier is able to detect intentional violations during the rising phase of the force for specificity levels $Sp < 95\%$. Conversely, for “target-reaching” tasks, the classifier yields reliable results after the penetration peak has occurred, for all the specificity levels investigated.

In general, the results of our work showed that the deployment of the interaction force signal (or features extracted from it) for the training phase of different classification methods is a good choice if we want to detect the current user’s intention to violate an active constraint. However, we propose some improvements in the experimental that overcome the current limitations of the work:

- It should be advisable to investigate how the classification outcomes are affected when the experimental trails are defined in a 3-dimensional environment;
- Consequently, the hardware setup should be upgraded to provide stereoscopic vision to the subject;
- The software setup, including the control system and the classification algorithms should be implemented in C++ language, to obtain improved stability of the control system and decreased computation time for the classification methods;
- As in this work we chose to test non-expert subject, it should be advisable to investigate whether an experimental protocol including expert subjects (e.g., surgeons) might affect the results of the classification.

Moreover, future work might be aimed at investigating possible methods to exploit the continuous output feature of the classification methods proposed, in order to optimally modulate the provided assistance according to the probability of the user’s intention classification within surgical manipulation tasks.

References

Abbott, J. J., Marayong, P., & Okamura, A. M. (2007). Haptic virtual fixtures for robot-assisted manipulation. In Robotics research (pp. 49-64). Springer Berlin Heidelberg.

Adler Jr, J. R., Chang, S. D., Murphy, M. J., Doty, J., Geis, P., & Hancock, S. L. (1997). The Cyberknife: a frameless robotic system for radiosurgery. Stereotactic and functional neurosurgery, 69(1-4), 124-128.

Boehm, D. H., Reichenspurner, H., Detter, C., Arnold, M., Gulbins, H., Meiser, B., & Reichart, B. (2000). Clinical use of a computer-enhanced surgical robotic system for endoscopic coronary artery bypass grafting on the beating heart. The Thoracic and cardiovascular surgeon, 48(4), 198-202.

Bowyer, S. A., Davies, B. L., & Rodriguez y Baena, F. (2014). Active constraints/virtual fixtures: A survey. Robotics, IEEE Transactions on, 30(1), 138-157.

Burghart, C., Keitel, J., Hassfeld, S., Rembold, U., & Woern, H. (1999, June). Robot controlled osteotomy in craniofacial surgery. In Proc. of the 1st Internat. Workshop on Haptic Devices in Medical Applications.

Burrus, C. S., Gopinath, R. A., & Guo, H. (1998). Introduction to wavelets and wavelet transforms (Vol. 998). New Jersey: Prentice hall.

Camarillo, D. B., Krummel, T. M., & Salisbury, J. K. (2004). Robotic technology in surgery: past, present, and future. The American Journal of Surgery, 188(4), 2-15.

Coste-Manière, È., Olender, D., Kilby, W., & Schulz, R. A. (2005). Robotic whole body stereotactic radiosurgery: clinical advantages of the CyberKnife® integrated system. The International Journal of Medical Robotics and Computer Assisted Surgery, 1(2), 28-39.

Craig, J. J. (2005). Introduction to robotics: mechanics and control (Vol. 3). Upper Saddle River: Pearson Prentice Hall.

Curley, K. C. (2005). An overview of the current state and uses of surgical robots. Operative Techniques in General Surgery, 7(4), 155-164.

Davies, B. (2000). A review of robotics in surgery. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 214(1), 129-140.

Davies, B. L., Hibberd, R. D., Ng, W. S., Timoney, A. G., & Wickham, J. E. A. (1991, June). A surgeon robot for prostatectomies. In Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on (pp. 871-875). IEEE.

Davies, B., Jakopec, M., Harris, S. J., Rodriguez y Baena, F., Barrett, A., Evangelidis, A., ... & Cobb, J. (2006). Active-constraint robotics for surgery. Proceedings of the IEEE, 94(9), 1696-1704.

Devijver, P. A. (1985). Baum's forward-backward algorithm revisited. Pattern Recognition Letters, 3(6), 369-373.

Dogangil, G., Davies, B. L., & y Baena, F. R. (2010). A review of medical robotics for minimally invasive soft tissue surgery. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 224(5), 653-679.

Ergeneman, O., Dogangil, G., Kummer, M. P., Abbott, J. J., Nazeeruddin, M. K., & Nelson, B. J. (2008). A magnetically controlled wireless optical oxygen sensor for intraocular measurements. Sensors Journal, IEEE, 8(1), 29-37.

Forney Jr, G. D. (1973). The viterbi algorithm. Proceedings of the IEEE, 61(3), 268-278.

Häcker, A., Chauhan, S., Peters, K., Hildenbrand, R., Marlinghaus, E., Alken, P., & Michel, M. S. (2005). Multiple high-intensity focused ultrasound probes for kidney-tissue ablation. Journal of Endourology, 19(8), 1036-1040.

Hertz, J. (1991). Introduction to the theory of neural computation (Vol. 1). Basic Books.

Howe, R. D., & Matsuoka, Y. (1999). Robotics for surgery. Annual Review of Biomedical Engineering, 1(1), 211-240.

Janabi-Sharifi, F., Hayward, V., & Chen, C. S. (2000). Discrete-time adaptive windowing for velocity estimation. Control Systems Technology, IEEE Transactions on, 8(6), 1003-1009.

Jensen, P. S., Grace, K. W., Attariwala, R., Colgate, J. E., & Glucksberg, M. R. (1997). Toward robot-assisted vascular microsurgery in the retina. Graefe's archive for clinical and experimental ophthalmology, 235(11), 696-701.

Kazanzides, P., Mittelstadt, B. D., Musits, B. L., Bargar, W. L., Zuhars, J. F., Williamson, B., ... & Carbone, E. J. (1995). An integrated system for cementless hip replacement. Engineering in Medicine and Biology Magazine, IEEE, 14(3), 307-313.

Kikuuwe, R., Takesue, N., & Fujimoto, H. (2008). A control framework to generate nonenergy-storing virtual fixtures: use of simulated plasticity. Robotics, IEEE Transactions on, 24(4), 781-793.

Krieger, A., Susil, R. C., Ménard, C., Coleman, J. A., Fichtinger, G., Atalar, E., & Whitcomb, L. L. (2005). Design of a novel MRI compatible manipulator for image guided prostate interventions. Biomedical Engineering, IEEE Transactions on, 52(2), 306-313.

Kwoh, Y. S., Hou, J., Jonckheere, E. A., & Hayati, S. (1988). A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery. Biomedical Engineering, IEEE Transactions on, 35(2), 153-160.

Kwok, K. W., Tsoi, K. H., Vitiello, V., Clark, J., Chow, G. C., Luk, W., & Yang, G. Z. (2013). Dimensionality reduction in controlling articulated snake robot for endoscopy under dynamic active constraints. Robotics, IEEE Transactions on, 29(1), 15-31.

Lavallee, S., Sautot, P., Troccaz, J., Cinquin, P. H., & Merloz, P. H. (1995). Computer-assisted spine surgery: a technique for accurate transpedicular screw fixation using CT data and a 3-D optical localizer. Computer Aided Surgery, 1(1), 65-73.

Li, M., & Okamura, A. M. (2003, March). Recognition of operator motions for real-time assistance using virtual fixtures. In Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2003. HAPTICS 2003. Proceedings. 11th Symposium on (pp. 125-131). IEEE.

Liporace, L. (1982). Maximum likelihood estimation for multivariate observations of Markov sources. Information Theory, IEEE Transactions on, 28(5), 729-734.

Nayak, M., & Panigrahi, B. S. (2011). Advanced Signal Processing Techniques for Feature Extraction in Data Mining. International Journal of Computer Applications, 19(9), 30-37.

Ng, W. S., Davies, B. L., Hibberd, R. D., & Timoney, A. G. (1993). A first hand experience in transurethral resection of the prostate. IEEE Med Bio Soc Magazine, 120-125.

Park, S., Howe, R. D., & Torchiana, D. F. (2001, January). Virtual fixtures for robotic cardiac surgery. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001 (pp. 1419-1420). Springer Berlin Heidelberg.

Passenberg, C., Groten, R., Peer, A., & Buss, M. (2011, June). Towards real-time haptic assistance adaptation optimizing task performance and human effort. In World Haptics Conference (WHC), 2011 IEEE (pp. 155-160). IEEE.

Phinyomark, A., Nuidod, A., Phukpattaranont, P., & Limsakul, C. (2012). Feature extraction and reduction of wavelet transform coefficients for EMG pattern classification. Elektronika ir Elektrotechnika, 122(6), 27-32.

Pittner, S., & Kamarthi, S. V. (1999). Feature extraction from wavelet coefficients for pattern recognition tasks. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 21(1), 83-88.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.

Ren, J., Patel, R. V., McIsaac, K. A., Guiraudon, G., & Peters, T. M. (2008). Dynamic 3-D virtual fixtures for minimally invasive beating heart procedures. Medical Imaging, IEEE Transactions on, 27(8), 1061-1070.

Rosenberg, L. B. (1993, September). Virtual fixtures: Perceptual tools for telerobotic manipulation. In Virtual Reality Annual International Symposium, 1993., 1993 IEEE (pp. 76-82). IEEE.

Rosenberg, L. B. (1993, December). Virtual fixtures as tools to enhance operator performance in telepresence environments. In Optical Tools for Manufacturing and Advanced Automation (pp. 10-21). International Society for Optics and Photonics.

Ryden, F., & Chizeck, H. J. (2012, October). Forbidden-region virtual fixtures from streaming point clouds: Remotely touching and protecting a beating heart. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on (pp. 3308-3313). IEEE.

Sackier, J. M., & Wang, Y. (1994). Robotically assisted laparoscopic surgery. Surgical endoscopy, 8(1), 63-66.

Salcudean, S. E., Prananta, T. D., Morris, W. J., & Spadinger, I. (2008, May). A robotic needle guide for prostate brachytherapy. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on (pp. 2975-2981). IEEE.

Sutherland, G. R., McBeth, P. B., & Louw, D. F. (2003, June). NeuroArm: an MR compatible robot for microsurgery. In International congress series (Vol. 1256, pp. 504-508). Elsevier.

Taylor, R. H., Paul, H. A., Mittelstadt, B. D., Glassman, E., Musits, B. L., & Bargar, W. L. (1989, November). Robotic total hip replacement surgery in dogs. In Engineering in Medicine and Biology Society, 1989. Images of the Twenty-First Century., Proceedings of the Annual International Conference of the IEEE Engineering in (pp. 887-889). IEEE.

Varma, T. R. K., & Eldridge, P. (2006). Use of the NeuroMate stereotactic robot in a frameless mode for functional neurosurgery. The International Journal of Medical Robotics and Computer Assisted Surgery, 2(2), 107-113.

Vercellis, C. (2006). Business Intelligence: modelli matematici e sistemi per le decisioni. McGraw-Hill.