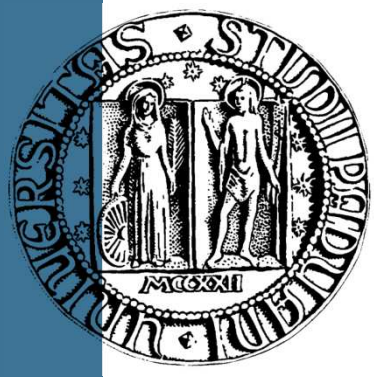
BCIs and mobile robots for neurological rehabilitation

practical applications of remote control

Remote Control of Mobile Robots Applied in Non-Invasive BCI for Disabled Users Afflicted by Motor Neurons Diseases



Author: Luigi Criveller

Supervisor: Emanuele Menegatti

BCIS AND MOBILE ROBOTS FOR NEUROLOGICAL REHABILITATION
practical applications of remote control

Padova.
February, 2010

BCIs and mobile robots for neurological rehabilitation

practical applications of remote control

Remote Control of Mobile Robots Applied in Non-Invasive BCI for Disabled Users Afflicted by Motor Neurons Diseases

by

LUIGI CRIVELLER
Department of Information Engineering,
University of Padova, Italy

with supervision of

MENEGATTI EMANUELE
Department of Information Engineering,
University of Padova, Italy

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

The physiological site of the sixth Chakra, the Ajna, is located in the center of the forehead. It is symbolized by an eye - the so-called third eye, the inner eye, or the eye of the mind. A lotus with only two petals, it is visualized as a deep indigo blue. This is the center of visual, psychic and intuitive perception- the place where we store our memories, perceive our dreams, and imagine our future.

Its element is light, a higher, faster vibration than that of sound, the least dense and most versatile of any element anyone can encounters. Traveling at speeds beyond comprehension, light in all its splendor allows to perceive the world in an infinite display of pattern. And anytime the world is viewed, it must be remembered that it is not objects that are seen but reflected light.

When the third-eye is opened, a new and completely different dimension of reality is revealed to the practitioner and users.

Preface

Brain-Computer Interface (BCI) research deals with establishing communication pathways between the brain and external devices. A BCI system enables control of devices or communication with other persons, only through cerebral activity, without using muscles. Because they don't depend on neuromuscular control, BCIs can provide communication and control for people with devastating neuromuscular disorders, such as amyotrophic lateral sclerosis, brainstem stroke, cerebral palsy, and spinal cord injury. BCI research and development aims to enable these users, who might be unable even to breathe or move their eyes, to convey their wishes to caregivers, use word-processing programs and other software, or control a robotic arm or a neuroprosthesis: from this point of view, at least in theory, there are no limitation for type of device a BCI system can control.

So, BCIs can be designed for communication or control applications. For communication applications, the BCI output drives a device such as a word processor or speech synthesizer, allowing the user to use language (The P300 speller is one example of a BCI-based communication device). For control applications, the BCI output drives a device such as a cursor, robotic arm, or wheelchair that the user can move. BCI-based control applications include multidimensional cursor movement, robotic arm movement, or even mobile robots movements.

Control applications can be based on goal selection. which means the BCI simply indicates the desired outcome, and downstream hardware and software handle the continuous kinematic control that achieves the outcome. Goal selection is also known as inverse kinematic control because the specific control parameters are computed from knowledge of the goal. Goal selection is much less demanding in terms of the complexity and rate of the control signals the BCI must provide. This would, of course, require a downstream device with detailed and continually updated knowledge of the environment.

It appears that is possible to control a mobile robotic device in remote, using a BCI goal-based, and so it is. Joining the low complexity of a goal-based system for detecting a suitable destination and the entertainment level of a mobile robot armed with a camera for real-time video feedback, patients are allowed to overcome, at least partially, their physical disorders and to interact with real world with a certain degree of freedom. Padova university's "Mobile Robotics" laboratories recent obtained a non-invasive BCI system that cover quite that characteristics: NEVRAROS. This system presents to patient a captivating, but simple graphical interface which provides specific visual stimuli for destinations selection, and the patient answers to such that stimuli with a specific EEG amplitude alteration. NEVRAROS acquires brain signals, extracts key features from them, and translates the features into goal-commands for remote mobile robot. Goal-commands are then converted in low level commands for navigation. In the meantime, the web camera mounted on the robot send a video stream to patient's display, so he can "lives" the entire navigation task.

This thesis represents a first well-organized report of NEVRAROS project. A concise excursus of base concepts concerning neurological rehabilitation, BCI systems, mobile robots and a brief characterization of NEVRAROS project's goals and achievements are presented in Chapter 1. Chapter 2 presents NEVRAROS system in details, starting from system's design and implementation and presenting system's architecture. Chapter 3 is used for showing system evaluation, both from performances and usability point of view. Description of result in testing NEVRAROS over patients will be presented as well. Finally, Chapter 4 summarizes evaluation results and proposes a nucleus of ideas for future works and developments. References can be found in Chapter 5.

Acknowledgments

This work is the result of inspirations and contributions from many teachers and students at the Department of Information Engineering of University of Padova (DEI).

I would like to thank Emanuele Menegatti, Teacher in Robotic and Chief Director in Autonomous Robotics Research, that make this experience so rich and stimulating by sharing his knowledge and resources with me. It is his work and perseverance that enabled me to achieve the end of this long journey by presenting such these results.

A very valuable and direct support and contribution for this work came from my current collaboration with the San Camillo IRCCS Hospital. I would like to thank: Franco Piccione, for technologies and resources in Neurological Rehabilitation; Stefano Silvoni for his contribution to EEG signal classification and system communication; Mauro Marchetti for his contribution to psychological contest and features in neurological rehabilitation; Marianna Cavinato for continuous psychological support and for her non-trivial medical support.

Also I would like to thank the Sensibilab team from Politecnico di Milano: their continuous support and development of BCI++ became the technological background for proceeding in this work. A special thank goes to Paolo Perego, for the continuous collaboration in setting up BCI++ system for my special needs.

This work takes inspiration and contributions by several fields, psychology and neuro-engineering included: for all those people who, directly or indirectly, support me with their specific knowledge goes a great thank.

It has been a pleasure to work with Autonomous Mobile Robots Laboratory team. Thanks to Marco Mina and Matteo Danieletto for their careful and valuable suggestion for solving several problems connected both to TCP/IP connection, C++ programming and Video management.

Special thanks to Cristina Fornasier for carefully correcting the text files and to Giorgia Criveller for her effort on the cover design.

List of Figures

Table of Contents

1 - Concepts Overview

BCI systems encompasses a broad range of knowledge derived from several disciplines: system anatomy and physiology, robotic, information technology, psychology and more. The major challenge for a researcher in BCI systems is to integrate the diverse information obtained from this disciplines into a coherent understanding structure.

First of all, a concrete set of medical concepts are needed, especially neurological concepts. The principal goal of BCI work is to enable people with neural pathways that have been damaged by amputation, trauma or diseases to better function and control their environment, through either reanimation of paralyzed limbs or control of robot devices. Although no one engineer would ever be asked to know exactly specific information concerning Human Nervous System, Motor Neuron Circuits, Human Motor Control or Motor Neuron Diseases, a general overview is necessary for better understanding both the medical contest where most clinical cases resides and the neurological challenges a BCI system can or cannot deal with.

Second, a solid background in recent BCI research is obviously needed. The concept of a direct Brain–Computer Interface has emerged over the last three decades of research as a promising alternative to existing interface methods. BCI research is a multidisciplinary field and as a result, there have been several varied approaches to the design of BCIs reported over the last three decades. Some notes concerning these approaches are needed for better understand design's choices in NEVRAROS project.

Moreover robotics and mobile robots in particular, are presented here, in very simple form and characterization, enabling the reader to understand why researchers choose a mobile robot instead other technologies as remote device to control.

The Human Nervous System

Neuroscience encompasses a broad range of questions about how nervous systems are organized, and how they function to generate behavior. These questions can be explored using the analytical tools of genetics, molecular and cell biology, systems anatomy and physiology, behavioral biology, and psychology. The major challenge studying neuroscience is to integrate the diverse knowledge derived from these various levels of analysis into a more or less coherent understanding of brain structure and function. Many of the issues that have been explored successfully concern how the principal cells of any nervous system, neurons and glia, perform their basic functions in anatomical, electrophysiological, and molecular terms. The varieties of neurons and supporting glial cells that have been identified are assembled into ensembles called neural circuits, and these circuits are the

primary components of neural systems that process specific types of information. Neural systems comprise neurons and circuits in a number of discrete anatomical locations in the brain. These systems subserve one of three general functions. Sensory systems represent information about the state of the organism and its environment, motor systems organize and generate actions; and associational systems link the sensory and motor sides of the nervous system, providing the basis for “higher-order” functions such as perception, attention, cognition, emotions, rational thinking, and other complex brain functions that lie at the core of understanding human beings, their history and their future.

The Neuron

The central nervous system monitors and controls the entire body by its peripheral divisions, which are distributed to all the muscles, organs, and tissues. The central nervous system is protected by fluid-filled membranes, the meninges, and surrounded by the bony skull and vertebrae.

The basic conducting element in the nervous system is the neuron. A neuron has a cell body, dendrite, and axon (Figure 1.1). The cell body contains many of the organelles vital to maintain the cells structure and function, and is considered the tropic center of the nerve cell. The dendrites extend from the cell body and increase the receptive surface of the neuron providing an elaborate arborization; thus, dendrites are the primary target for synaptic input from other neurons. The axon leaves the cell body and connects to other cells. Axons are covered by a membrane called myelin that insulates the axons from the fluids in the central nervous system. The site of contact between the axon of one nerve cell and the dendrites and cell body of another neuron is the synapse. The electrical event that carries signals over neurons is called the action potential, which is a self-regenerating wave of electrical activity that propagates from its point of initiation at the cell body to the terminus of the axon where synaptic contacts are made. The chemical and electrical process by which the information encoded by action potentials is passed on at synaptic contacts to the next cell in a pathway is called synaptic transmission. In the central nervous system, the nerve cells are supported by glia and blood vessels; in the peripheral nervous system, they are supported by satellite cells, fibroblasts, Schwann cells, and blood vessels.

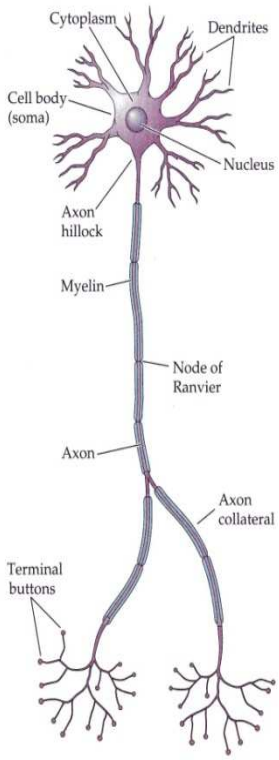


FIGURE 1.1. Diagram of a nerve cell.

Neurons are organized into neural circuits that process specific kinds of information and provide the foundation of sensation, perception and behavior. Although the arrangement of neural circuits varies greatly according to the function being served, some features are characteristic of all such ensembles. Considering the direction of information flow in any particular circuit, neurons are divided into three basic categories: (1) nerve cells that carry information toward the brain or spinal cord are called afferent neurons; (2) nerve cells that carry information away from the brain or spinal cord are called efferent neurons; (3) interneurons, the vast majority of the neurons in the central nervous system. They only participate in the local aspects of a circuit, based on the short distances over which their axons extend. These three functional classes are the basic constituents of all neural circuits.

The areas in the central nervous system that contain high numbers of neuronal cell bodies are called gray matter and the regions that contain primarily axons are called white matter. Neurons are organized into ganglia, nuclei, or layered cortices, basing of their location in human body.

The Nervous System

When considered together, circuits that process similar types of information comprise neural systems that serve broader behavioral purposes. The most general functional distinction divides such collections into sensory systems that acquire and process information from the environment, and motor systems that respond to such information by generating movements and other behavior. In addition to these broad functional distinctions, neuroscientists and neurologists have conventionally divided the vertebrate nervous system anatomically into central and peripheral components. The CNS, central nervous system (brain and spinal cord) is surrounded by fluid-filled membranes and housed in either the bony skull or vertebrae. In contrast, the PNS, peripheral nervous system, that brings information from and to the central nervous system lacks a bony covering but is protected by the fascia, skin, muscles, and organs where it distributes. Sensory information enters the central nervous system through the afferent divisions of the peripheral nerves. Peripheral nerves are found everywhere in the body: skin, muscles, organs, and glands. Peripheral nerves originate from either the spinal cord or brain.

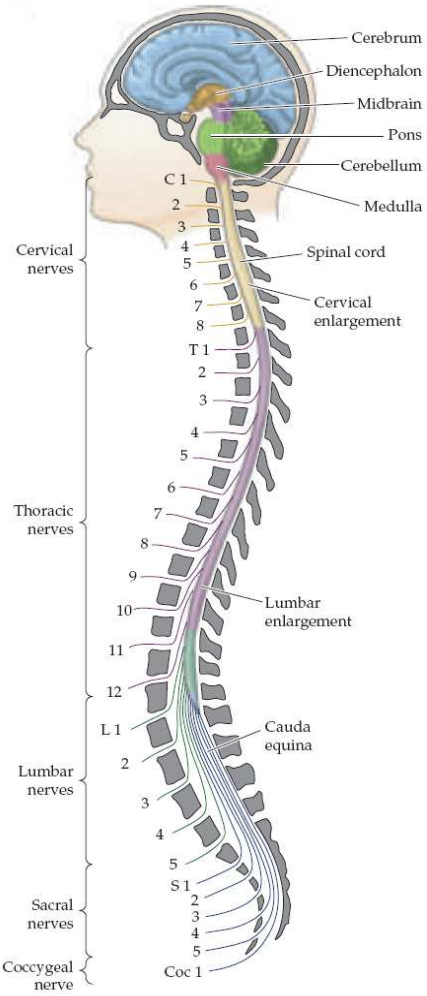


FIGURE 1.2. The subdivisions and components of the central nervous system. (Note that the position of the brackets on the left side of the figure refers to the vertebrae, not the spinal segments.)

The central nervous system (Figure 1.2) consists of the spinal cord and brain (brain stem, cerebellum, diencephalons, and cerebrum). The organization of the gray matter varies in each of these regions. Attached to all of the 32 segments of the spinal cord and the brain stem are sensory ganglia (which are simply local accumulations) that form the first link in the sensory system and bring the sensory information into the central nervous system (thus, they are typical PNS elements) . Motor axons exit from each of the 32 segments of the spinal cord and all levels of the brain stem and connect the central nervous system to all muscles and organs in the body. In the spinal cord, much of the brain stem, and diencephalon, the neurons are organized into nuclei, local accumulations of neurons having roughly similar connections and functions; in the superior colliculus of the brain stem, cerebellum, and cerebrum, the neurons are organized anatomically into layers, or cortex, and functionally into vertical columns.

Neuroanatomical Terminology

Describing the organization of any neural system requires a rudimentary understanding of anatomical terminology. Anterior and posterior indicate front and back (head and tail); rostral and caudal, toward the head and tail; dorsal and ventral, top and bottom (back and belly); and medial and lateral, at the midline or to the side (Figure 1.3). The proper assignment of the anatomical axes dictates the standard planes for histological sections or live images used to study the internal anatomy of the brain. Horizontal sections (also referred to as axial or transverse sections) are taken parallel to the rostral-caudal axis of the brain; thus, in an individual standing upright, such sections are parallel to the ground. Sections taken in the plane dividing the two hemispheres are

sagittal, and can be further categorized as midsagittal and parasagittal, according to whether the section is near the midline (midsagittal) or more lateral (parasagittal). Sections in the plane of the face are called coronal or frontal. Different terms are usually used to refer to sections of the spinal cord. The plane of section orthogonal to the long axis of the cord is called transverse, whereas sections parallel to the long axis of the cord are called longitudinal. In a transverse section through the human spinal cord, the dorsal and ventral axes and the anterior and posterior axes indicate the same directions.

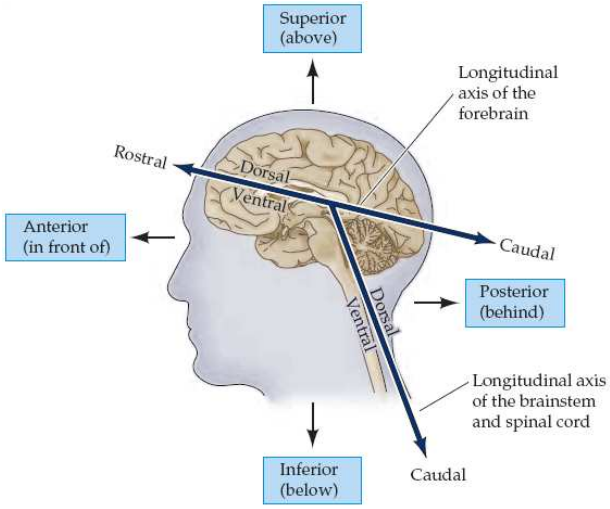


FIGURE 1.3. A flexure in the long axis of the nervous system arose as humans evolved upright posture, leading to an approximately 120° angle between the long axis of the brainstem and that of the forebrain. The terms anterior, posterior, superior, and inferior refer to the long axis of the body, which is straight. Therefore, these terms indicate the same direction for both the forebrain and the brainstem. In contrast, the terms dorsal, ventral, rostral, and caudal refer to the long axis of the central nervous system. The dorsal direction is toward the back for the brainstem and spinal cord, but toward the top of the head for the forebrain. The opposite direction is ventral. The rostral direction is toward the top of the head for the brainstem and spinal cord, but toward the face for the forebrain. The opposite direction is caudal.

Central Nervous System

The central nervous system (defined as the brain and spinal cord) is usually considered to have seven basic parts: the spinal cord, the medulla, the pons, the cerebellum, the midbrain, the diencephalon, and the cerebral hemispheres (Figure 1.4). Running through all of these subdivisons are fluid-filled spaces called ventricles. The medulla, pons, and midbrain are collectively called the brainstem and they surround the 4th ventricle (medulla and pons) and cerebral aqueduct (midbrain). The diencephalon and cerebral hemispheres are collectively called the forebrain, and they enclose the 3rd and lateral ventricles, respectively.

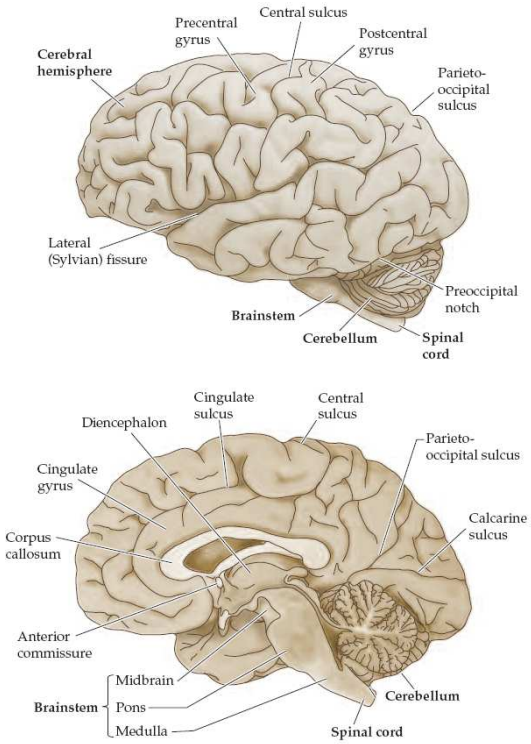


FIGURE 1.4. Gross anatomy of the nervous central system

The spinal cord is that portion of the central nervous system that lies in the vertebral canal from the upper border of the atlas to the lower border of the first lumbar vertebrae in the adult. The spinal cord has 32 segments divided into five regions (cervical, thoracic, lumbar, sacral, and coccygeal) and these regions innervates specific regions in the neck and upper extremity (cervical segments), thorax and abdomen (thoracic levels), anterior leg and thigh (lumbar segments), and buttock and posterior leg and thigh (lumbar segments). This ordered relationship between the spinal cord and body produces a somatotopic organization throughout the central nervous system. The spinal cord is organized into columns of gray and white matter, with the gray matter centrally placed and surrounded by the white matter. The white matter of the spinal cord is divided into three

columns: anterior, posterior, and lateral. The pathways interconnecting the spinal cord and brain are found in these columns.

The brainstem is a conduit for several major tracts in the central nervous system that relay sensory information from the spinal cord and brainstem to the forebrain, or relay motor commands from forebrain back to motor neurons in the brainstem and spinal cord. The brainstem contains numerous additional nuclei that are involved in important functions including the control of heart rate, respiration, blood pressure, and level of consciousness.

The cerebellum is essential for the coordination and planning of movements as well as learning motor tasks and storing that information.

The diencephalon stands as the great waystation between the brain stem and cerebral cortex, as all the major ascending pathways terminate here. The diencephalon consists of the following divisions: thalamus, hypothalamus, epithalamus, and subthalamus. The thalamus forms the largest division of the diencephalon and it is divided into several groupings of nuclei. All of the ascending pathways terminate in thalamic nuclei and then their information is projected onto their respective region of the cerebral cortex. The epithalamus is a small zone with functions similar to the hypothalamus. The hypothalamus is the smallest subdivision of the diencephalon and is found inferiorly in the third ventricle. However, because it functions as the “head ganglion” in the autonomic nervous system, it might well be the most important portion and emotional center of the diencephalon. The subthalamus, found below the thalamus, is an important subcortical region in the basal ganglia. The functions of this region are to integrate sensory and motor information and to begin to interpret these data according to the perceptions of the emotional areas in the brain.

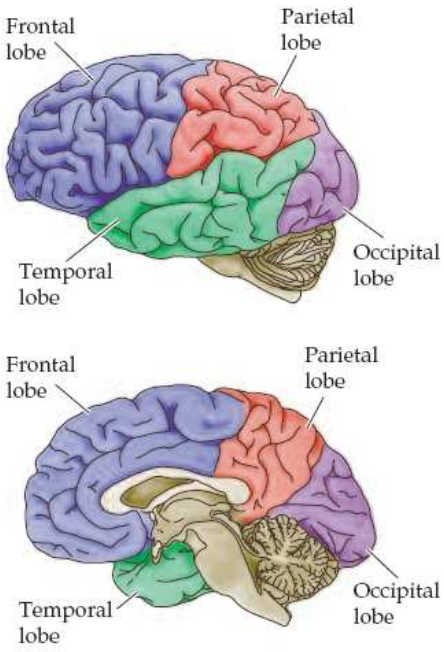


FIGURE 1.5. Lobes division of forebrain.

The cerebrum, forming the bulk of the brain and thus of the central nervous system, consists of a left and a right hemisphere containing the cortical gray matter, white matter, and basal nuclei. In each hemisphere, we find four lobes: the rostral frontal lobe, the middle parietal lobe, and the posterior occipital lobe with the inferiorly located temporal lobe (Figure 1.5). The cerebral cortex consists of a corrugated surface, the cortical gray matter, which is laminated and has six layers, is broken up into numerous gyri, separated by narrow spaces or grooves, the sulci. The 12 to 15 billion cortical neurons are found in the gray cortical mantle.

The left cerebral hemisphere is dominant for functions including speech, initiation of movement, emotions, and artistic abilities. Note that each hemisphere provides motor controls to the opposite side of the body. This is because the motor pathway from each cerebral cortex crosses in the transition between the spinal cord and medulla; consequently, the sensory fibers also cross over to the opposite side of the body. In over 90 % of the human population, movement initiates from the left hemisphere, making it dominant for initiation of movement.

The cerebral cortex includes motor, sensory, auditory, and visual regions. In addition, broad areas are involved with multimodal integration, which combines sensory and motor with an emotional content to determine how to respond in any situation. These emotional or limbic areas occupy much of the temporal and frontal lobes. Human beings also use language extensively and much of the frontal–parietal–occipital–temporal regions that abut the lateral sulcus in the left hemisphere undertake these functions. Similar areas of the right hemisphere are devoted to visual–spatial integration.

The axons entering or leaving the cerebral hemispheres form three distinctive groups of fibers: associational, commissural, and subcortical. (1) Associational Fibers. These type of fibers provides the integrative circuitry for movement, language, memory, and emotions. They are distinguished in: short associational fibers that form the bulk of local connections within a hemisphere, and long associational fibers that interconnect diverse areas in a hemisphere, providing multimodal association. (2) Commissural Fibers. These fibers interconnect areas in the contralateral hemispheres and permit learning and memory in one hemisphere to be shared with the other. The bulk of the frontal, parietal, occipital, and temporal lobes are interconnected by the corpus callosum. (3) Subcortical Fibers. This category of fibers includes fiber bundles reaching the cortex from subcortical areas, and axons leaving the cortex and connecting to subcortical nuclei.

Motor Control and Motor Neurons Disorders

Movements, whether voluntary or involuntary, are produced by spatial and temporal patterns of muscular contractions orchestrated by the brain and spinal cord. Analysis of these circuits is fundamental to an understanding of both normal behavior and the etiology of a variety of neurological disorders. The brainstem and spinal cord circuitry make elementary reflex movements possible, as well as the circuits that organize the intricate patterns of neural activity responsible for more complex motor acts. More deeply, all movements produced by the skeletal musculature are initiated by "lower" motor neurons in the spinal cord and brainstem that directly innervate skeletal muscles; the innervation of visceral smooth muscles is separately organized by the autonomic divisions of the visceral motor system. The lower motor neurons are controlled directly by local circuits within the spinal cord and brainstem that coordinate individual muscle groups, and indirectly by "upper" motor neurons in higher centers that regulate those local circuits, thus enabling and coordinating complex sequences of movements (Figure 1.6). Especially important are circuits in the basal ganglia and cerebellum that regulate the upper motor neurons, ensuring that movements are performed with spatial and temporal precision. Specific disorders of movement often signify damage to a particular brain region. For example, clinically important and intensively studied neurodegenerative disorders such as Parkinson's disease, Huntington's disease, and Amyotrophic Lateral Sclerosis result from pathological changes in different parts of the motor system.

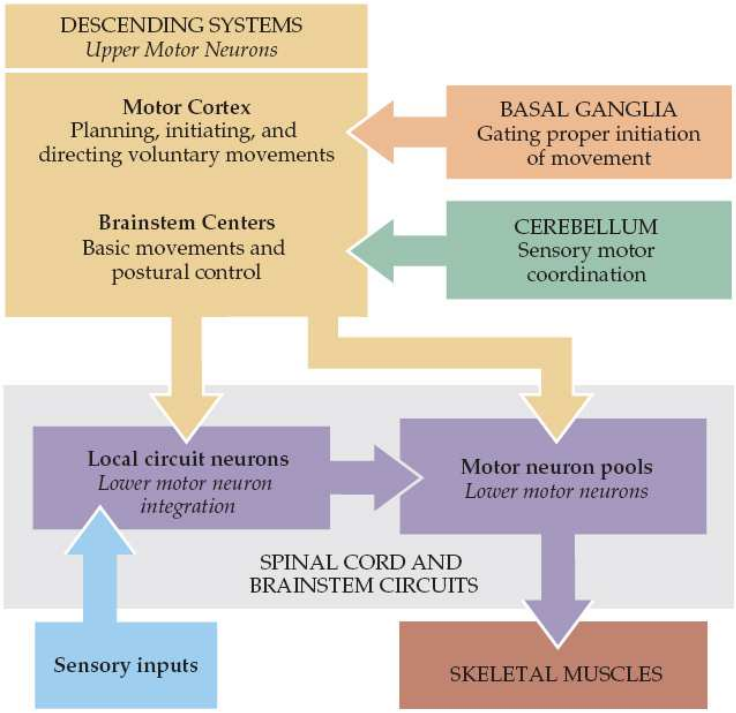


FIGURE 1.6. Overall organization of neural structures involved in the control of movement. Four systems—local spinal cord and brainstem circuits, descending modulatory pathways, the cerebellum, and the basal ganglia— make essential and distinct contributions to motor control.

Lower Motor Neuron Circuits and Motor Control

Skeletal muscle contraction is initiated by lower motor neurons in the spinal cord and brainstem. The cell bodies of the lower neurons are located in the spinal cord gray matter and in the motor nuclei of the cranial nerves in the brainstem. These neurons (also called α motor neurons) send axons directly to skeletal muscles via the ventral roots and spinal peripheral nerves, or via cranial nerves in the case of the brainstem nuclei. The spatial and temporal patterns of activation of lower motor neurons are determined primarily by local circuits located within the spinal cord and brainstem. Descending pathways from higher centers comprise the axons of upper motor neurons and modulate the activity of lower motor neurons by influencing this local circuitry. The cell bodies of upper motor neurons are located either in the cortex or in brainstem centers. The axons of the upper motor neurons typically contact the local circuit neurons in the brainstem and spinal cord, which, via relatively short axons, contact in turn the appropriate combinations of lower motor neurons. The local circuit neurons also receive direct input from sensory neurons, thus mediating important sensory motor reflexes that operate at the level of the brainstem and spinal cord. Lower motor neurons, therefore, are the final common pathway for transmitting neural information from a variety of sources to the skeletal muscles.

Four distinct but highly interactive motor subsystems (local circuits in the spinal cord and brainstem, descending upper motor neuron pathways that control these circuits, the basal ganglia, and the cerebellum) all make essential contributions to motor control. Alpha motor neurons located in the spinal cord and in the cranial nerve nuclei in the brainstem directly link the nervous system and muscles, with each motor neuron and its associated muscle fibers constituting a functional entity called the motor unit. Motor units vary in size, amount of tension produced, speed of contraction, and degree of fatigability. Graded increases in muscle tension are mediated by both the orderly recruitment of different types of motor units and an increase in motor neuron firing frequency. Local circuitry involving sensory inputs, local circuit neurons, and α and γ motor neurons are especially important in the reflexive control of muscle activity. The stretch reflex is a monosynaptic circuit with connections between sensory fibers arising from muscle spindles and the α motor neurons that innervate the same or synergistic muscles. Gamma motor neurons regulate the gain of the stretch reflex by adjusting the level of tension in the intrafusal muscle fibers of the muscle spindle. This mechanism sets the baseline level of activity in α motor neurons and helps to regulate muscle length and tone. Other reflex circuits provide feedback control of muscle tension and mediate essential functions such as the rapid withdrawal of limbs from painful stimuli. Much of the spatial coordination and timing of muscle activation required for complex rhythmic movements such as locomotion are provided by specialized local circuits called central pattern generators.

Upper Motor Neuron Control of Brainstem and Spinal Cord

The axons of upper motor neurons descend from higher centers to influence the local circuits in the brainstem and spinal cord that organize movements by coordinating the activity of lower motor neurons. The sources of these upper motor neuron pathways include several brainstem centers and a number of cortical areas in the frontal lobe. The motor control centers in the brainstem are especially important in ongoing postural control. Each center has a distinct influence. Two of these centers, the vestibular nuclear complex and the reticular formation, have widespread effects on body position. Another brainstem center, the red nucleus, controls movements of the arms; also in the brainstem, the superior colliculus contains upper motor neurons that initiate orienting movements of the head and eyes. The motor and “premotor” areas of the frontal lobe, in contrast, are responsible for the planning and precise control of complex sequences of voluntary movements. Most upper

motor neurons, regardless of their source, influence the generation of movements by directly affecting the activity of the local circuits in the brainstem and spinal cord. Upper motor neurons in the cortex also control movement indirectly, via pathways that project to the brainstem motor control centers, which, in turn, project to the local organizing circuits in the brainstem and cord. A major function of these indirect pathways is to maintain the body's posture during cortically initiated voluntary movements.

Two sets of upper motor neuron pathways make distinct contributions to the control of the local circuitry in the brainstem and spinal cord. One set originates from neurons in brainstem centers (primarily the reticular formation and the vestibular nuclei) and is responsible for postural regulation. The reticular formation is especially important in feedforward control of posture (that is, movements that occur in anticipation of changes in body stability). In contrast, the neurons in the vestibular nuclei that project to the spinal cord are especially important in feedback postural mechanisms (i.e., in producing movements that are generated in response to sensory signals that indicate an existing postural disturbance). The other major upper motor neuron pathway originates from the frontal lobe and includes projections from the primary motor cortex and the nearby premotor areas. The premotor cortices are responsible for planning and selecting movements, whereas the primary motor cortex is responsible for their execution. The motor cortex influences movements directly by contacting lower motor neurons and local circuit neurons in the spinal cord and brainstem, and indirectly by innervating neurons in brainstem centers that in turn project to lower motor neurons and circuits. Although the brainstem pathways can independently organize gross motor control, direct projections from the motor cortex to local circuit neurons in the brainstem and spinal cord are essential for the fine, fractionated movements of the distal parts of the limbs, the tongue, and face.

Modulation of Movement by Basal Ganglia and Cerebellum

In contrast to the components of the motor system that harbor upper motor neurons, the basal ganglia and cerebellum do not project directly to either the local circuit or lower motor neurons; instead, they influence movement by regulating the activity of upper motor neurons. The term basal ganglia refers to a large and functionally diverse set of nuclei that lie deep within the cerebral hemispheres. The motor components of the basal ganglia effectively make a subcortical loop that links most areas of the cortex with upper motor neurons in the primary motor and premotor cortex and in the brainstem. The neurons in this loop respond in anticipation of and during movements, and their effects on upper motor neurons are required for the normal course of voluntary movements. When one of these components of the basal ganglia or associated structures is compromised, the patient cannot switch smoothly between commands that initiate a movement and those that terminate the movement.

The efferent cells of the cerebellum influence movements by modifying the activity patterns of the upper motor neurons. In fact, the cerebellum sends prominent projections to virtually all upper motor neurons. Thus, much like the basal ganglia, the cerebellum is part of a vast loop that receives projections from and sends projections back to the cerebral cortex and brainstem. The primary function of the cerebellum is to detect the difference, or "motor error," between an intended movement and the actual movement, and, through its projections to the upper motor neurons, to reduce the error. These corrections can be made both during the course of the movement and as a form of motor learning when the correction is stored. When this feedback loop is damaged, as occurs in many cerebellar diseases, the afflicted individuals make persistent movement errors whose specific character depends on the location of the damage.

Motor Neurons Diseases

The motor neuron diseases (MNDs) are a group of progressive neurological disorders that destroy cells that control essential muscle activity such as speaking, walking, breathing, and swallowing. Normally, messages from upper motor neurons are transmitted to lower motor neurons and from them to particular muscles. When there are disruptions in these signals, the result can be gradual muscle weakening, wasting away, and uncontrollable twitching. Eventually, the ability to control voluntary movement can be lost. MNDs may be inherited or acquired, and they occur in all age groups. In adults, symptoms often appear after age 40. In children, particularly in inherited or familial forms of the disease, symptoms can be present at birth or appear before the child learns to walk. Further and detailed informations about these. The causes of sporadic (non-inherited) MNDs are not known, but environmental, toxic, viral, or genetic factors may be implicated. Common MNDs include amyotrophic lateral sclerosis (ALS), spinal muscular atrophy, primary lateral sclerosis, and progressive muscular atrophy.

Amyotrophic lateral sclerosis (ASL) is a progressive neurodegenerative disease caused by the degeneration of α motor neurons and upper motor neurons in the motor cortex. The disorder causes muscle weakness and atrophy throughout the body as both the upper and lower motor neurons degenerate, ceasing to send messages to muscles. Unable to function, the muscles gradually weaken, develop twitches and eventually atrophy because of that denervation. The patient may ultimately lose the ability to initiate and control all voluntary movement. Cognitive function is generally spared except in certain situations such as when ALS is associated with frontotemporal dementia. Sensory nerves generally remain functional. In the majority of cases the disease does not impair a patient's mind, personality, intelligence, or memory. Nor does it affect a person's ability to see, smell, taste, hear, or feel touch.

Primary lateral sclerosis (PLS) is a rare neuromuscular disease characterized by progressive muscle weakness in the voluntary muscles. It affects upper motor neurons only. Symptoms may include difficulty with balance, weakness and stiffness in the legs, and clumsiness. Other symptoms may include spasticity in the hands, feet, or legs; foot dragging, and speech problems due to involvement of the facial muscles. The disorder usually begins in the legs, but it may also start in the tongue or the hands. The disease progresses gradually over a number of years, or even decades. In PLS, there is no evidence of the degeneration of spinal motor neurons or atrophy that occurs in amyotrophic lateral sclerosis.

Spinal Muscular Atrophy (SMA) is a neuromuscular disease resulting in progressive muscular atrophy and weakness. The clinical spectrum of SMA ranges from early infant death to normal adult life with only mild weakness. In all of its forms, the primary feature of SMA is muscle weakness, accompanied by atrophy of muscle. This is the result of denervation, or loss of the signal to contract, that is transmitted from the spinal cord. This is normally transmitted from motor neurons in the spinal cord to muscle via the motor neuron's axon, but either the motor neuron with its axon, or the axon itself, is lost in all forms of SMA.

Progressive muscular atrophy (PMA) is a rare subtype of amyotrophic lateral sclerosis (ALS) which affects only the lower motor neurons.]This is in contrast to the most common form of ALS/MND, amyotrophic lateral sclerosis, which affects both the upper and lower motor neurons, or another rare form of ALS/MND, primary lateral sclerosis, which affects only the upper motor neurons. The distinction is important because PMA is associated with a better prognosis than classical ALS/MND.

Neural Signaling

The brain is remarkably adept at acquiring, coordinating, and disseminating information about the body and its environment. Such information must be processed within milliseconds, yet it also can be stored away as memories that endure for years. Neurons within the central and peripheral nervous systems perform these functions by generating sophisticated electrical and chemical signals. This chapter describes these signals and how they are produced. It explains how one type of electrical signal, the action potential, allows information to travel along the length of a nerve cell. It also briefly shows how other types of signals, both electrical and chemical, are generated at synaptic connections between nerve cells. Synapses permit information transfer by interconnecting neurons to form the circuitry on which neural processing depends. Finally it describes common methods and approaches for detecting and acquiring brain activities information.

Neural Signals and Transmission

Nerve cells generate electrical signals that transmit information. Although neurons are not intrinsically good conductors of electricity, they have evolved elaborate mechanisms for generating these signals based on the flow of ions across their plasma membranes, and transmit them to other cells by means of synaptic connections. These signals ultimately depend on changes in the resting electrical potential, a negative potential generate within the neuron, across the neuronal membrane. A resting potential occurs because nerve cell membranes are permeable to one or more ion species subject to an electrochemical gradient. More specifically, a negative membrane potential at rest results from a net efflux of K^+ across neuronal membranes that are predominantly permeable to K^+. In contrast, an action potential occurs when a transient rise in Na^+ permeability allows a net flow of Na^+ in the opposite direction across the membrane that is now predominantly permeable to Na^+. The action potential transiently abolishes the negative resting potential and makes the transmembrane potential positive. Action potentials are propagated along the length of axons and are the fundamental signal that carries information from one place to another in the nervous system. The brief rise in membrane Na^+ permeability is followed by a secondary, transient rise in membrane K^+ permeability that repolarizes the neuronal membrane and produces a brief undershoot of the action potential. As a result of these processes, the membrane is depolarized in an all-or-none fashion during an action potential. When these active permeability changes subside, the membrane potential returns to its resting level because of the high resting membrane permeability to K^+.

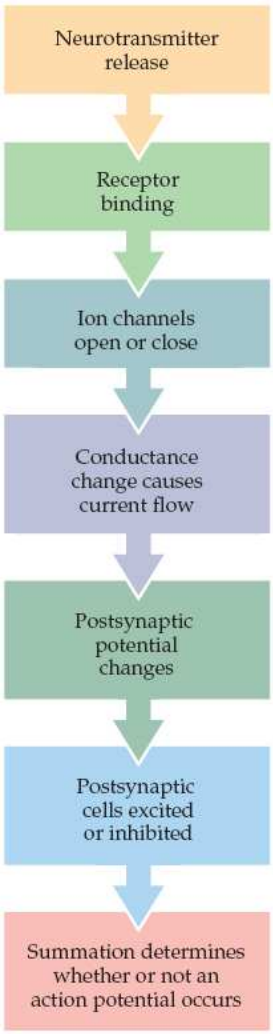


FIGURE 1.7. Events from neurotransmitter release to postsynaptic excitation or inhibition.

Synapses communicate the information carried by action potentials from one neuron to the next in neural circuits. Synaptic communication is made possible by synapses, the functional contacts between neurons. Two different types of synapse, electrical and chemical, can be distinguished on the basis of their mechanism of transmission. At electrical synapses, current flows through gap junctions, which are specialized membrane channels that connect two cells. In contrast, chemical synapses enable cell-

to-cell communication via the secretion of neurotransmitters; these chemical agents released by the presynaptic neurons produce secondary current flow in postsynaptic neurons by activating specific receptor molecules.

Measuring Brain Electrical Activity

Brain activity measurement methods can be classified according to their invasiveness. The quality of the acquired signals usually increases with the invasiveness of the method since with invasive techniques the probe is closer to the source. Quality of non-invasive methods signals is poor since the skull acts as an attenuator of neural signals, thus filtering out high frequencies and lowering signal-to-noise ratio (SNR). Another way to classify the different available methods to measure the activity of the brain is the nature of the recording, namely neuronal or vascular (metabolic) activity. Neuronal activity can be measured in the range of milliseconds whereas the temporal resolution of vascular activity is much lower, it lies in the range of seconds

Single unit recordings The electrophysiological activity (action potentials) from a single or a reduced population of neurons can be recorded using this method. Electrodes (or micro-electrodes with a tip size of a few μm for single neuron recordings) are directly implanted in the cortex. Due to its high invasiveness, this technique is mainly used on animals (monkeys). This technique provides the best temporal and spatial resolution but beside the risk of the invasive approach, electrodes induce scars in the tissue so that quality of recordings decreases over time and neuronal tissue necrosis can follow electrode implantation.

Magnetic resonance imaging (MRI) is primarily a medical imaging technique to visualize detailed internal structure and limited function of the body (Figure 1.8). MRI provides great contrast between the different soft tissues of the body, making it especially useful in neurological imaging. The body is largely composed of water molecules which each contain two hydrogen nuclei or protons. When a person goes inside the powerful magnetic field of the scanner, the magnetic moments of these protons align with the direction of the field. A radio frequency electromagnetic field is then briefly turned on, causing the protons to alter their alignment relative to the field. When this field is turned off the protons return to the original magnetization alignment. These alignment changes create a signal which can be detected by the scanner. The frequency at which the protons resonate depends on the strength of the magnetic field. The position of protons in the body can be determined by applying additional magnetic fields during the scan which allows an image of the body to be built up. These are created by turning gradient coils on and off which creates the knocking sounds heard during an MR scan. Diseased tissue, such as tumors, can be detected because the protons in different tissues return to their equilibrium state at different rates. By changing the parameters on the scanner this effect is used to create contrast between different types of body tissue. Contrast agents may be injected intravenously to enhance the appearance of blood vessels, tumors or inflammation. MRI uses no ionizing radiation and is generally a very safe procedure.

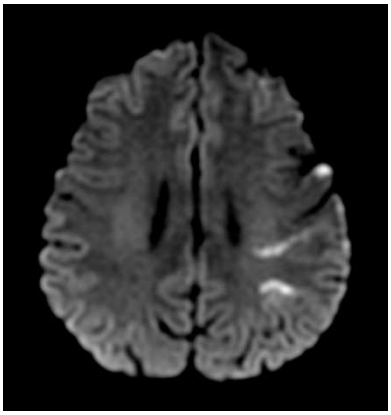


FIGURE 1.8. Horizontal plane MRI brain image.

Magnetoencephalography (MEG) is an imaging technique used to measure the magnetic fields produced by electrical activity in the brain via extremely sensitive devices such as superconducting quantum interference devices (SQUIDs). The MEG signals derive from the net effect of ionic currents flowing in the dendrites of neurons during synaptic transmission. In accordance with Maxwell's equations, any electrical current will produce an orthogonally oriented magnetic field. It is this field which is measured with MEG. The net currents can be thought of as current dipoles which are currents defined to have an associated position, orientation, and magnitude, but

no spatial extent. According to the right-hand rule, a current dipole gives rise to a magnetic field that flows around the axis of its vector component.

Positron Emission Tomography (PET) Three dimensional maps of functional processes in the brain can be obtained with this nuclear medical imaging technique (Figure 1.9). A short-lived radioactive tracer isotope is integrated into a metabolically active molecule, typically sugar, and is injected into the blood circulation. This radioactive isotope decays by emitting a positron, the antimatter counterpart of an electron. Therefore it is possible to measure metabolic activity of a brain area by detecting positron emission. Due to the relatively slow coupling between neuronal activity and metabolism (neurovascular coupling), the temporal resolution of this technique is usually low, typically lying in the range of seconds. This technique also presents a small risk for the subject since radioactive compounds are injected in the blood circulation.

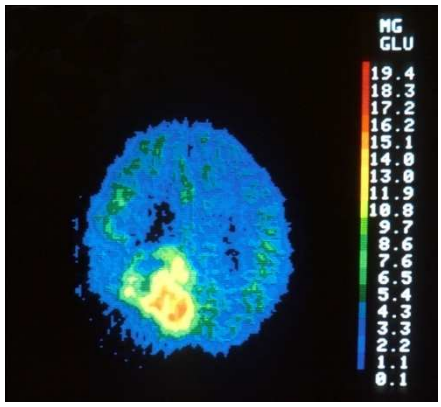


FIGURE 1.9. Horizontal plane PET brain image.

The Electrocorticogram (ECoG) is a technique for recording electrical potentials in the brain. In a surgical procedure an array of electrodes, typically an 8×8 grid, is placed on the cortex surface. After the implantation, signals which are generated by the same mechanisms as the EEG can be measured. However, effects of volume conduction are less visible in the ECoG, i.e. the signals are less spatially blurred than EEG signals. Further advantages are that ECoG signals are barely contaminated with muscle or eye artifacts and that activity in frequencies up to about 100 Hz can be easily observed.

Near Infrared Spectroscopy (NIRS) This method uses the interaction of the near infrared region of the electromagnetic field spectrum (from about 1000 nm to 2500 nm) with biological materials that show a relatively good transparency in this wavelength. Oxygenated and deoxygenated haemoglobin have different optical properties. As for fMRI, since blood oxygenation is correlated with neuronal activity, differences in optical response can be used to measure brain activity. Due to the neurovascular coupling, this technique has a low temporal resolution and so far its spatial resolution is poor.

Electroencephalography (EEG) is the recording of electrical activity along the scalp produced by the firing of neurons within the brain (Figure 1.10). In clinical contexts, EEG refers to the recording of the brain's spontaneous electrical activity over a short period of time, usually 20–40 minutes, as recorded from multiple electrodes placed on the scalp. The electric potentials generated by single neurons are far too small to be picked by EEG. EEG activity therefore always reflects the summation of the synchronous activity of thousands or millions of neurons that have similar spatial orientation, radial to the scalp. Currents that are tangential to the scalp are not picked up by the EEG. Because voltage fields fall off with the fourth power of the radius, activity from deep sources is more difficult to detect than currents near the skull. Scalp EEG activity shows oscillations at a variety of frequencies. Several of these oscillations have characteristic frequency ranges, spatial distributions and are associated with different states of brain. These oscillations represent synchronized activity over a network of neurons.

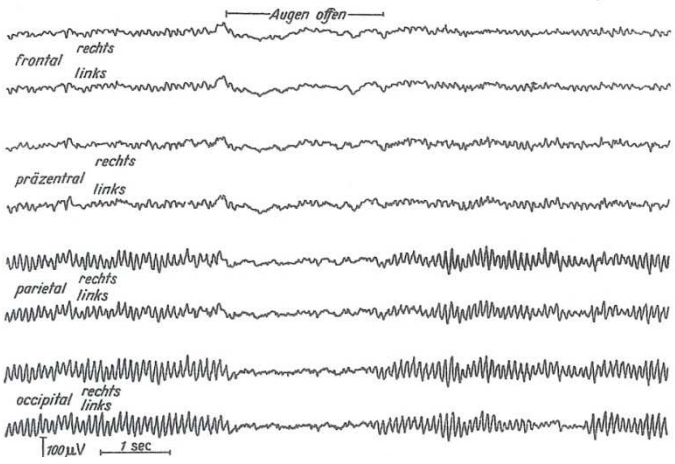


FIGURE 1.10. Normal EEG acquisition.

EEG recordings usually present rhythmical patterns. EEG waves can be classified according to different brain functions, but the terminology is imprecise and sometimes abused because traditionally brain waves were classified on the basis of visual inspection and not using precise frequency analysis. (1) Delta is the lowest frequency range, below 4 Hz. It is typical of infants and is present in deep sleep and in some organic brain diseases. (2) Theta is the frequency range from 4 to 8 Hz and is associated with drowsiness, childhood, adolescence and young adulthood. This EEG frequency can sometimes be produced by hyperventilation. Theta waves can be seen during hypnagogic states such as trances, hypnosis, deep day dreams, lucid dreaming and light sleep and the preconscious state just upon waking, and just before falling asleep. (3) Alpha is the frequency range from 8 to 12 Hz. It is characteristic of a relaxed, alert state of consciousness. For alpha rhythms to arise, usually the eyes need to be closed. Alpha attenuates with drowsiness and open eyes, and typically come from the occipital (visual) cortex. The alpha rhythm is usually characterized by rounded or sinusoidal wave forms. However, a sizable minority of individuals have sharp alpha configuration. In such cases, the negative component appears to be sharp and the positive component appears to be rounded, similar to the wave morphology of rolandic mu rhythm. An alpha-like normal variant called Mu is sometimes seen over the motor cortex (central scalp) and attenuates with movement, or rather with the intention to move. (4) Beta is the frequency range from 12 to 30 Hz. Low amplitude beta with multiple and varying frequencies is often associated with active, busy or anxious thinking and active concentration. Rhythmic beta with a dominant set of frequencies is associated with various pathologies and drug effects. (5) Gamma is the frequency range from approximately 30 to 100 Hz. Gamma rhythms may be involved in higher mental activity, including perception, problem solving, fear, and consciousness.

Brain Computer Interfaces System

A Brain Computer Interface, is a device that provides the brain with a new, non-muscular communication and control channel. The purpose of a BCI is to identify the user's intention by observing and analyzing brain activity without relying on signals from muscles or peripheral nerves. The concept of a BCI has emerged over the last three decades of research as a promising alternative to existing interface methods. Researchers produced a wide number of books, papers and documents with detailed informations about these interface techniques. In other words, a BCI allows users to act on their environment by using only brain activity, without using peripheral nerves and muscles. Goal of BCI research is to develop systems that allow disabled users to communicate with other persons, to control artificial limbs, or to control their environment. To achieve this goal, many aspects of BCI systems are currently being investigated. Research areas include evaluation of invasive and noninvasive technologies to measure brain activity, evaluation of control signals (i.e. patterns of brain activity that can be used for communication), development of algorithms for translation of brain signals into computer commands, and the development of new BCI applications. So, the ultimate goal of this research is to create a specialized interface that will allow an individual with severe motor disabilities to have effective control of such those devices. This type of interface would increase an individual's independence, leading to an improved quality of life and reduced social costs.

Concepts and Classification

BCI definition covers a wide class of systems which interface with the central nervous system. A general model of a BCI system is the following. Note that no mentions are made both concerning how brain activities are acquired and which set of devices the system can controls.

An online data-processing system controls devices in real time and provides feedback to the user. Providing feedback as fast and accurately as possible is critical. Any unnecessary noise or delay is adverse to the quality of the feedback and hinders users' abilities to train their brain patterns. To generate the control signal, the BCI must extract and classify signals features from user's brain. Every feature extraction method has its own hyperparameters. Data preprocessing is important to remove the influence of technical artifacts and non-brain activity such as the electrical signals caused by eye movements or facial muscles. A BCI uses offline analysis not only to estimate a reliable classifier, but to tune the processing steps' hyperparameters. It can also compare and optimize different features or groups of features, as well as various classifiers and spatial filters, offline. The features extracted are translated into logical device independent control signals, which are translate into semantic control signals that are appropriated for a particular type of device. The semantic control signals are then translated into physical control signals that are used within the device, whose dynamic state represents, together with what displayed, if necessary, by a control display, the feedback for the user (Figure 1.11).

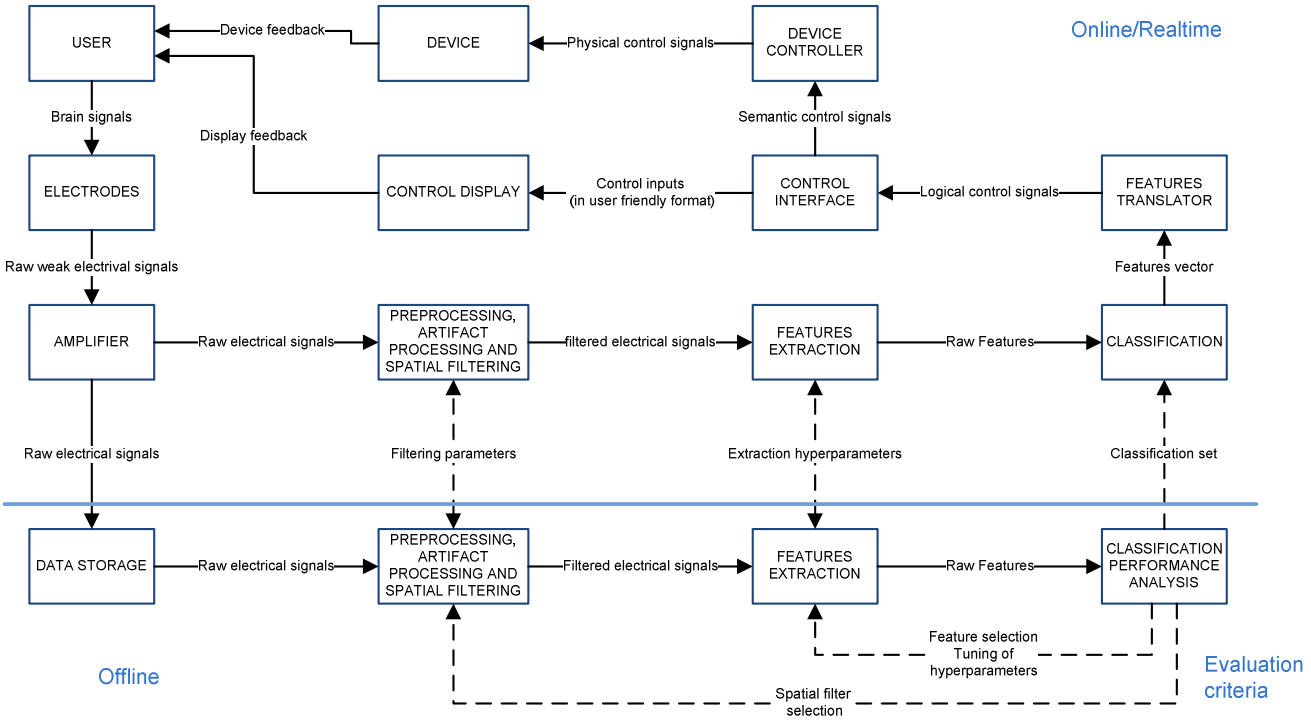


FIGURE 1.11. Proposed functional model for a typical brain-computer interface. A BCI uses an online data-processing system to control devices in real time and provide feedback to the user; it uses offline analysis to train the feature classifier and optimize the various data-processing steps.

General classification of BCI systems divides them into two different groups, depending on method used for acquiring raw neurological signals from the brain. We have, therefore, a distinction between Invasive BCI (IBCI) and Non-Invasive BCI (NIBCI). Invasive BCIs are implanted directly into the grey matter of the brain during neurosurgery. As they rest in the grey matter, invasive devices produce the highest quality signals of BCI devices but are prone to scar-tissue build-up, causing the signal to become weaker or even lost as the body reacts to a foreign object in the brain. Non-invasive BCI uses electrodes implanted outside the skull, on the cranial skin, and measures the overall brain electrical activities. As we said above, different methods to measure brain activity can

be used in a BCI. The characteristics of the methods we reviewed are summarized. As can be seen, each method has its own advantages and disadvantages and hence so far no method of choice exists.

Neurophysiologic Signals

Because of its portability, most non-invasive brain-computer interfaces use electroencephalogram signals, by means EEG method is used for measuring brain activity. Main brain activity signals source is hence a huge cortical neurons set synchronous activity. Two fundamentally different approaches exist to classify such this activity, based on how brain activity take place. In the first approach subjects perceive a set of stimuli displayed by the BCI system and can control their brain activity by focusing onto one specific stimulus (evoked BCI). The changes in neurophysiologic signals resulting from perception and processing of stimuli are termed event-related potentials (ERPs). Event-related potentials are, in principle, easy to pick up with scalp electrodes. The necessity of external stimulation does, however, restrict the applicability of evoked potentials to a limited range of tasks. In the second approach users control their brain activity by concentrating on a specific exogenous mental task (spontaneous BCI). In this approach feedback signals are often used to let subjects learn the production of easily detectable patterns of neurophysiologic signals. Both evoked and spontaneous approaches present a set of patterns used by BCI researchers during their experiments.

Slow cortical potentials (SCPs) (spontaneous) are slow voltage shifts in the EEG occurring in the frequency range 1-2 Hz. Negative SCPs correspond to a general decrease in cortical excitability. Positive SCPs correspond to a general increase in cortical excitability. Through feedback training subjects can learn to voluntarily control their SCPs. The voluntary production of negative and positive SCPs has been exploited in one of the earliest BCI systems for disabled subjects: in their pioneering work, Birbaumer et al. showed that patients suffering from ASL can use a BCI to control a spelling device and to communicate with their environment.

Event-related changes of ongoing EEG activity in specific frequency bands (spontaneous). Detected rhythmic activity usually within 8-12 Hz, often mixed with a β component (around 20 Hz) and observed over primary sensory or motor cortical areas. Event-related desynchronization (ERD) defines an amplitude decrease of a rhythmic component, whereas event-related synchronization (ERS) characterizes an amplitude increase.

Motor-related potentials (MRPs) (spontaneous). The events to which MRPs are related are the preparation or imagination of movements. MRPs are slow negative potentials, observable over the sensorimotor cortex before movement onset or during movement imagination. Since the sensorimotor cortex has a somatotopic organization the body part that will be moved, or for which a movement is imagined, can be inferred from the location of greatest amplitude of the MRP.

Steady State Evoked Potentials (SSEP) (evoked). Stimulus are presented repetitively at high rate, so that relevant neuronal structures are prevented to return to their resting states. The amplitude of the SSEP is increased at the frequency of the modulation of stimulus. Dominant location depends on type of SSEP: VEP (visual), AEP (auditory), SEP (somatosensory). Steady-State Visual Evoked Potentials (SSVEPs) are oscillations observable at occipital electrodes, induced by repetitive visual stimulation. Stimulation at a certain frequency leads to oscillations at the same frequency and at harmonics and subharmonics of the stimulation frequency. In a BCI, SSVEPs are used by simultaneously displaying several stimuli flickering at different frequencies. Users can select one stimulus by focusing on it, which leads to increased amplitude in the frequency bands corresponding to the flickering frequency of the stimulus.

Evoked Potentials (EPs) (evoked). Evoked potentials can be seen as a specific kind of ERP generated directly in response to external stimulus such as visual evoked potentials (VEP) and auditory evoked potentials (AEP). Reactions to stimuli or events lead to variations of the electrical activity of specific brain areas and the resulting EEG traces exhibit modifications called potentials. In the case of external stimuli, for example discrete visual feedback, the precise time of the stimulus is known. It's therefore possible to extract averages of the stimulus locked response of the brain. Time-locked averages allow elimination of random noise while keeping track of the ERP components. When the precise time of the stimulus is not available, it's much more complicated to extract ERP from the ongoing EEG. In this case, a specific action of the subject related to the nature of the stimulus can be used as trigger. In any case, the main challenge is the detection of ERP no more in averages of many single trials, but directly at the level of the single trial.

Visual Evoked Potentials Waveforms

Many BCI systems use a functional approach based on external stimulation. This stimulation typically affects those sensory areas that are more refined in humans: visual area, auditory area and somatosensory area. The visual and auditory areas in particular are subject specific research in the field of BCI systems, and this strategic choice is a direct consequence of the primary objective of such systems: freeing the subject's communication and interaction with the environment from the neuromuscular control, and using the sole point of brain activity as reference. In this sense, the auditory area and especially the visual area are favorite than somatosensory area, for they are more suitable for the collection of information from the environment.

Auditory evoked potentials are usually caused by tones, such as harmonics or impulsive sounds. Basing of their latency of occurrence from stimulation, the EPs are divided in short, medium and long latency (Figure 1.12). Short latency EPs include: (1) SP potential (Summating Potential) and AP potential (Action Potential) occurring in the first 2.5 ms by the cochlea and auditorium nerve, and (2) brainstem auditory evoked potentials (BSAEP) reflecting the response of brainstem stimulation during the first 12 ms. Average latency EPs include a series of peaks of positive and negative wave occurring between 12 and 50 ms of stimulation, but exact timing location is still on discussion. Long latency EPs take place between 50 and 250 ms after stimulation, and consist of four major wave peaks, namely P50, N100, P150, N200, on the basis of polarity and peak latency. Their origin is due to cortical area of the brain. Evoked potentials are also found more than 250 ms latency, but their nature is more related to the cognitive context of the stimulus.

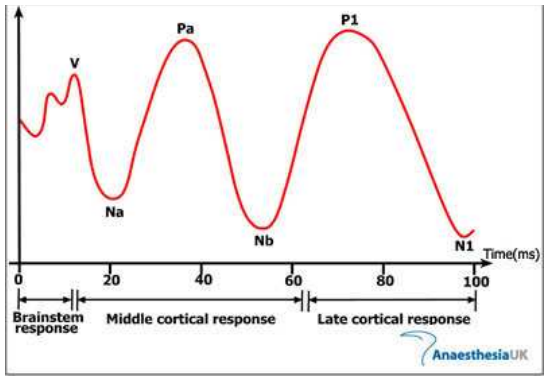


FIGURE 1.12. Schematic representation of the AEP

VEPs waveform depends by the temporal frequency of stimulus. With high-frequency stimulation (approximately f> 6 Hz.) the waveform becomes equivalent to a sinusoid, resulting in potential steady state. With low-frequency stimulation (approximately f <2 Hz), the waveform forms a discrete number of deviations, or transients potential, of short duration and occurrence at the sudden change in brain activity. VEPs are characterized by latency, amplitude and waveform affected by aging of the subject. The standard description of VEPs refers to the typical response of an adult aged 18 to 60 years. The peak latency of visual EPs indicates the elapsed time from onset of stimulus to the maximum point of excursion or deflation. Using pattern reversal stimulation, the visual EPs found are: N75 (negative bias, approximate 75ms latency), P100 (positive bias, approximate 100ms latency) and N135 (negative bias, approximate 135ms latency). Using stimulation onset-

offset, three potentials are obtained: C1 (positive polarization approximate 75ms latency), C2 (negative bias, approximate 125ms latency) and C3 (positive bias, approximate 150ms latency). Stimulating with light, six potential are evoked, whose latency is much more approximate than for other types of stimulation (Figure 1.13); N1 (negative bias, approximate 40ms latency), P1 (positive bias, approximate 60ms latency), N2 (negative polarization, latency Approximate 90ms) P2 (positive bias, approximate latency 120ms), N3 (negative bias, approximate latency 150ms), P3 (positive bias, approximate 300ms latency).

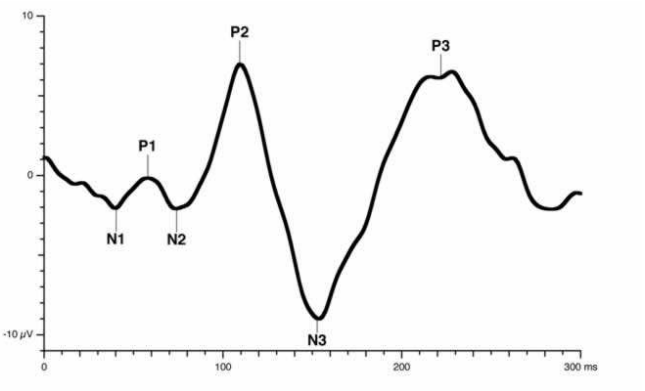


FIGURE 1.13. Schematic representation of the VEP using light stimulation approach.

Enhancing BCI Systems with Robotic Devices

As described above, BCI systems allow to map patients brain activities signals, and transform them into logical signals for commanding external devices. If we consider the mapping task of perception and processing of stimuli (ERPs), we have hence an artificial source of stimuli that leads patients to discriminate their brain activities between each stimulus. On more, if we allow BCI system to use results from stimulation for creating new sequences of stimuli, in example using a mobile robot armed with a camera and using the camera video stream as new context for patients stimulation, new advantages point immediately up. First of all, patients can perceive a change in the real world as result of their tasks using BCI system. That necessarily leads to an improve in patients attention, avoiding descendent parable in locked-in status. Second, robot as external device helps patients in communicating with real world. BCI systems normally work over virtual environments, and patients are only able to interact with virtual objects, or go all over virtual paths. That is different from doing the same with real objects or real paths, indeed. Besides, there are no spatial limits in using remote robot to control via BCI system. With opportune linking systems, patients can be easily be connected with remote robot placed far away, in different places of the same building, or in different buildings in the same city, or in different buildings in different cities and countries. From this point of view, Internet provides a well implemented linking system. Advantages in using a BCI system, say, in example, placed in Italy, for commanding a remote robot in an England art museum are trivial.

FIGURE 1.14. A Rovio mobile robot screenshot.

Goals

NEVRAROS project was born as result of a strict collaboration between the Dipartimento di Ingegneria dell'Informazione, from Università degli Studi di Padova, and Ospedale IRCCS San Camillo, from Venezia. Such that collaboration started with previous works both in non-Invasive BCI's research and remote controlled autonomous mobile robot's research.

NEVRAROS system presents to patient a captivating, but simple graphical interface which provides specific visual stimuli for destinations selection, and the patient answers to such that stimuli with a specific EEG amplitude alteration. NEVRAROS acquires brain signals, extracts key features from them, and translates the features into goal-commands for remote mobile robot. Goal-commands are then converted in low level commands for navigation. In the meantime, the web camera armed on the robot send a video stream to patient's display, so he can "lives" the entire navigation task.

Initial meetings and brainstorming sessions whit F. Piccione, E. Menegatti, S. Silvoni, M. Marchetti and M. Cavinato showed how this project would have to utilize technologies and devices already in use at San Camillo's laboratories. Moreover, this project concerns about information and medical technologies for medical rehabilitation. Since users of the system to be build will be strictly humans, the use of human-like vision for the on-board camera armed on the robotic device is hence essential.

Objectives

Major objectives of this project are listed below:

- Enhance patient's attention and reaction level by using a BCI system, which assembles a simple-to-use but highly appealing user interface.
- Reduce patient's effort-in-using level for the BCI system by implementing a goal-based user interface.
- Implement a BCI systems which allows to command a mobile autonomous remote robot as external device.
- Use on-board robot camera real-time video stream (which represents the final output of converting brain activities from last stimulation into device commands) as user's visual feedback and reference for next stimulation.

Requirements

Major requirements of this project are listed below:

- Use of non-Invasive BCI system.
- Use EEG as method to measure brain electrical activities.
- Use the positive deflection in the EEG P300 as neurophysiologic signal to analyze and classify.
- Use of P300-based classification algorithm in use at San Camillo.
- Use of mobile robot with human-like vision camera.
- Use of computers and other technologies, such as amplifiers, electrodes, computers, screens and similar, that are already in use at San Camillo.

Related Works

The impetus behind research into the establishment of communications pathways between the brain and external devices, or brain-computer interfaces (BCI), can be traced back to studies conducted in the 1970s postulating algorithms that correlated the firing patterns of motor cortex neurons with specific muscular responses [1]. In the intervening decades, advances in computer and sensor technologies[2], [3], component miniaturization, and materials biocompatibility [4], as well as our ever-improving understanding of the human central nervous system [5], [6], [7], [8], have served to accelerate research into the development of truly effective BCI systems [9], [10], [11], [12], [13].

The majority invasive BCI science pools are placed in North America. The feasibility of this direct-BCI approach has been demonstrated over the past eight years, beginning in animals and more recently progressing to humans. In 1999, Chapin et al. trained rats to position a robot arm to obtain water by pressing a lever by recording simultaneously motor cortex neurons activities [14]. One year after, Wessberg et al. used signals derived from the rat motor cortex for controlling one-dimensional movements of a robot arm [15]. In 2002, Taylor et al. proposed a direct cortical control of 3D neuroprosthetic devices [16]. In 2003, Carmena et al. demonstrate that primates can learn to reach and grasp virtual objects by controlling a robot arm through a closed-loop brain-machine interface [17]. 2006 preliminary studies made by Hocberg et al. show initial results for a tetraplegic human using an Invasive BCI piloted neuromotor prostheses [19].

The majority non-Invasive BCI science pools are placed in Europe. A variety of studies over the past 15 years have shown that the scalp-recorded electroencephalogram can be used as the basis for a brain-computer interface. Non-Invasive BCI can provide an alternative method of communication and control for those severely affected individuals. There have been a number of BCI communication systems that have been designed to demonstrate proof of principle. These are based on a variety of neural features such as slow cortical potentials, motor potentials, event-related synchronizations and desynchronizations, steady-state evoked potentials, and P300. These systems have generally used surface-recorded EEG. In 1999 Birbaumer and colleagues trained individuals to modify SCPs based on feedback and used this paradigm for BCI-based communication [20]. Mason and Birch in 2000, have used motor-related potentials as the basis of a BCI-communication system [21]. In 1991 Wolpaw et al. showed that ERD-related phenomena could be used for a BCI based on a two-target, cursor-movement task [22]. The mu rhythm has also been used for tasks involving multiple targets in one dimension by McFarland et al. in 2003 [23], answering questions by Miner et al. in 1998 [24], two-dimensional cursor movement by Wolpaw and McFarland in 1994 and 2004 [25], [26], spelling devices by both Blankertz et al. in 2006 [27] and Scherer et al. in 2004 [28], and control of an orthosis by Muller-Putz et al. in 2005 [29]. Middendorf et al. (2000) used a SSVEP-based system to allow users to select one of two virtual buttons flashing at different rates on a computer screen [30]. Muller-Putz et al. (2005) used an SSVEP-based system to allow users to select one of four flashing lights on a video screen [31]. Cheng et al. (2002) used a SSVEP-based system to allow users to select one of 12 buttons flashing at different rates on a computer screen [32]. In 2004, E. Lalor et al. proposed a non-invasive system that used SSVEPs as control signals for commanding playing mechanisms for an immersive 3D game [33]. Sellers and Donchin (2006) showed that both users without motor impairments and users with ALS were able to use the P300-based, single-stimulus system using either auditory or visual presentations [34]. In 2008, F. Piccione et al. proposed a non-Invasive BCI system based on visually-evoked brain stimulation, that used P300 ERPs as control signals for commanding the motion of a cursor in a 2D graphic environment [35]. In 2008, C. J. Bell et al. proposed a non-Invasive BCI system based on visually-evoked brain stimulation, that used P300 ERPs as control signals for commanding a partially autonomous humanoid robot to perform complex tasks such as

walking to specific locations and picking up desired objects [36]. Vaughan et al. (2006) describe the daily use of an in-home P300 system by an individual with ALS. This system consists of a reduced set of electrodes, a portable amplifier, and a laptop computer [37].

Early studies and preliminary experiments in commanding a mobile robot using a P300 ERPs-based non-Invasive BCI were performed by L. Tonin et al. [38] in 2008. Results with non-invasive BCI system based on spontaneous brain activity, that makes possible the continuous control of a mobile robot in a house-like environment were obtained in 2004 by José R. Millan et al. [39]. An electroencephalogram-based control of a mobile robot was performed by Kazuo Tanaka in 2005 [40]. Design and online experiments of a self-paced online brain-computer interface (BCI) for controlling a simulated robot in an indoor environment were made by Tao Geng in 2010 [41]. In 2008 a teleoperated museum robotic guide using a P300-based-like selection interface was made by Chello [42]. Tonin and Menegatti showed first experimental results integrating a P300-based BCI with an holonomic mobile robot [43].

2 - NEVRAROS System

Accomplishment of NEVRAROS system takes over twelve months of research. Such that system comes from strict collaboration of several researches, such as engineers, medics and psychologists, which contributed supporting with all their specific knowledge. Not surprising NEVRAROS project management leaded in creating several prototypes, each ones was tested and controlled by each of team players. After each testing session new features were added (and also many bugs were fixed, with jointly collaboration with Politecnico di Milano's researchers), ending with a an hardware and software system able to meet all that constraints, requirements and goals presented in last chapter.

Life Cycle of this project covers main phases of almost all engineering projects. Using an evolutionary process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model, NEVRAROS creation passed through analysis, design, implementation and testing phases for each spiral of the evolutionary iteration. Prime results of analysis phase was already discussed in previous chapter, highlighting most important goals and prerequisites. Main aspect of design phase cover grand characterization of model, showing in high - level intuitive and visual form what the system is able to do, which are the main component that compose the system and how every component is eventually interconnected to others. Implementation phase concerns about how such those details exploited in design phase are achieved in concrete. Eventually test phase is responsible for checking and controlling system features and characteristics compliance with requisites and goals.

Overview

NEVRAROS is a system proposed as a solution in neurological rehabilitation for patients affected by severe and irreversible neuro-motor diseases; such those pathologies removed indeed the traditional opportunities for communication and iteration with the outside world through the traditional channels of communication and iteration offered by the human body: we are talking about use of speech and use of limbs, basis of human movements and communication.

With this system, patient is able to achieve some iteration functions and communication with the surrounding environment again, using brain activities as signal control. Although present researches and technologies are not quite good enough to restore even a fraction of that independence level showed a healthy person, the possibility of control small navigation tasks, or express simple key concepts, is a concrete positive result that provides new

hope of integration for a whole category of patients otherwise destined, as unfortunately demonstrated by the medical statistics, to a premature death not because of the disease which mainly suffer, but rather by the action of secondary, quite tractable, conditions and diseases, not diagnosed by medical personnel because of the lack of communication by the patient.

Goal-based destination selection

The chief objective of this system is to ensure the navigation. Inability in using and controlling voluntary muscles of the limbs for the management of movements, is in fact primary dysfunction that outcome from motor neuron diseases. Current medical and surgical knowledge is unable to restore damages and disturbances of that kind, and so offering a system to view and navigate the outside world has great importance and significance. Although concurrent studies on BCI show many important results in controlling vehicles suitable to human being, their complexity, both in terms of implementation and usage by patient, makes them hard tools to practice with (non-Invasive BCI for manage prosthetic objects requires the ability to control certain pattern of brain signals that can be achieved by healthy patients only after months or even years of training, and time of learning for disabled patients are even longer). The idea of developing a remote robotic eye instead, allowing patients to view what the robot perceives and command it in the early stages of navigation decision, is a concept which, though simpler and less performance than other solutions, can be easily realized in practice, using systems and methods that require little or marginal effort of learning, and extremely limited training time.

FIGURE 2.1. complex non-Invasive BCI system for commanding a robotic wheelchair.

Let's consider for an moment the key concepts about movements. Abstracting movement concepts from a human-like prospective, and idealizing the hypothetical movements device in a non specified mobile device, it is easily demonstrable that user is able to control that device in navigating into 2D surfaces (i.e. a flat surface in a 3D environment), using only six high-level commands: four of them are related to selecting movement direction (right, left, forward, backward); the other two are used for actually impress the movement through selected direction (go, stop). With a little more fantasy, the number of high-level command can be reduced to four without reducing navigation freedom: the two command for impress the movements can be absorbed by the four commands for direction selection as follow: user select one of the directional commands for starting movement in that direction, and then select opposite movement direction command for stopping movement (i.e. "left" command device to navigate in left direction if device is in "stop" status, while stop the device if it is in "right" navigation status). Such these schemes are simple but complete, if high-level commands can be imparted to device in real time by the user (imagine what could happen if exist a large hole somewhere in the movement direction, and user isn't quick enough to command a "stop" to

FIGURE 2.2. Controlling a mobile device with six high-level commands: four directional arrows for direction decision, and two movements buttons for starting and stopping movement

the mobile device). A solution of this kind is hence not acceptable enough, especially considering that device's users will be patients with neurological diseases.

FIGURE 2.3. Controlling a mobile device with four high-level destination commands: three select new destinations, one for going back to last destination

Overtaking such this problem is possible, if a little restriction in user movement freedom is acceptable. Previous solution give to user complete freedom in navigating into 2D surface, so user is able to manually select his destination and favorite path for reach it. Restricting available destination into a finite set, and pre-preparing properly path for navigate from a destination to another offer to user a chance to control mobile device without worrying about tight deadlines in controlling navigation: once he select favorite destination, automatic routines will select secure path independently, and device will reach such that destination with no further effort. Control a mobile device with such this solution requires four high-level commands again: one of them is "backward to last destination", and the other three are " forward to next destination", one for each remaining directions.

Classification Process

Strictly related to navigation, another important goal of NEVRAROS system is to translate the intent of a subject directly into control commands for mobile device. A significant challenge in designing a BCI is to balance the technological complexity of interpreting the user's brain signals with the amount of user training required for successful operation of the interface.

The BCI scenario involves two possibly adaptive parts, the user and the system. The operant conditioning approach uses a fixed translation algorithm to generate a feedback signal from EEG. Users are not equipped with a mental strategy they should use. Rather, they are instructed to watch a feedback signal and to find out how to voluntarily control it. Successful operation is reinforced by a reward stimulus. In such BCI systems, the adaptation of the user is crucial and typically requires extensive training. On the other hand, machine learning techniques allow us to fit many parameters of a general translation algorithm to the specific characteristics of the user's brain signals. This is done by a statistical analysis of a calibration measurement in which the subject performs well-defined mental acts such as imagined. Here in principle no adaptation of the user is required, but it can be expected that users will adapt their behavior during feedback operation. The idea of the machine learning approach is that a flexible adaptation of the system relieves a good amount of the learning load from the subject.

NEVRAROS system is somewhere between those extremes. It uses a machine learning technique for specific waves classification in brain signal, which are generated voluntary from user as consequence of a cognitive effort from visual stimulation. Machine learning technique adopted falls into supervised learning techniques, or deducing a specific function from training data. The task of the supervised learner, or classifier, is to predict the value of the function for any valid input object after having seen a number of training examples. The training data consist of pairs of input object and desired output. The output of the function can predict a class label of the input object. To achieve this, the learner has to generalize from the presented data to unseen situations in a "mathematically reasonable" way. Supervised learning generates a global model that maps input objects to desired outputs. In order to solve the given problem of supervised learning some steps are involved:

- Determine the type of training examples. Before doing anything else it must be decided what kind of data is to be used as an example.
- Gathering a training set. The training set needs to be characteristic of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.
- Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should be large enough to accurately predict the output.
- Determine the structure of the learned function and corresponding learning algorithm.
- Complete the design. The learning algorithm is run on the gathered training set. Parameters of the learning algorithm may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation. After parameter adjustment and learning, the performance of the algorithm may be measured on a test set that is separate from the training set.

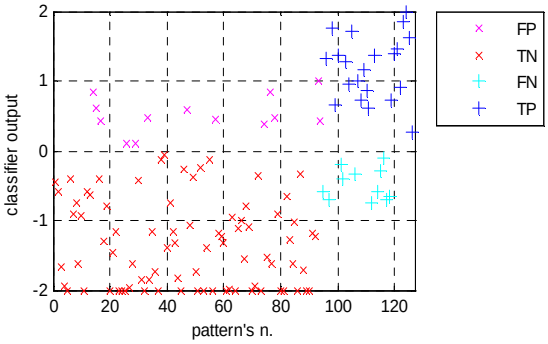


FIGURE 2.4. A classification result after a test phase. Different color correspond to different classes of classification, namely FP (detection of P300 with no real P300 occurrence), TN (no detection of P300 with real P300 occurrence), FN (no detection of P300 and no real P300 occurrence), TP (detection of P300 and real P300 occurrence),

The classification problem that NEVRAROS system manages is to properly classify specific temporal subsets of brain signal (epochs), namely recognize for each input if a specific wave form (P300 waveform) occurs. The learning algorithm runs on a training set of P300 wave forms generated by patient brain activity, and algorithms parameters are then optimized. Eventually, tuned algorithm is used for properly classifying new incoming P300 waveforms from patient in test phase. In order to achieve such that classification, NEVRAROS provides two different execution modes: one is for learning phase and one is for testing phase. In learning mode, patient train classifier providing a set of epochs for training set. In testing mode, adjustment of learning algorithm are tested over patient. More concerning both classification method and classification algorithm and NEVRAROS functional modalities will be revealed afterwards.

System Overview

Previous chapter already presents main concepts concerning BCI systems and overall architecture. NEVRAROS functional model recall functional model for a typical brain-computer interface: patient, or user, is connected to the system by a set of electrodes properly implanted outside the skull, on the cranial skin. Brain activity is hence detected and transmitted to an opportune amplifier as electrical signal. Once signal is powered up to opportune level, enhanced signal is transmitted to the classifier for classification. In the mean time an external analogical blink sensor acquire interval time of stimulation provided by user interface, and transmit to classifier (via amplifier, again) a pseudo-periodic square wave, which indicates when stimulations take place (high voltage level means visual stimulation is ongoing, low voltage level represents otherwise). When the classifier receives both

signals from blink sensor and electrodes, starts to classifier the P300 waves resident into brain signal using square wave as selector of the time period to analyze. Once classification is finished, results are transmitted to interface manager which updates user interface status (providing new stimuli or calling required module) and sends, if necessary, high level commands of navigation to the mobile devices. During all its execution lifetime, the mobile device sends back to the interface manager a video stream that is displayed into user interface screen.

Figure 2.5 shows an high-level diagram of components for NEVRAROS system. Note that blink sensor and user receive the same visual stimulation, but user respond to stimulation creating a new set of waves to be classified, while blink sensor generate a pseudo-periodic square wave used as reference for selecting the proper segment of signal to analyze. The same function provided by blink sensor can be implemented via software by the interface manager too.

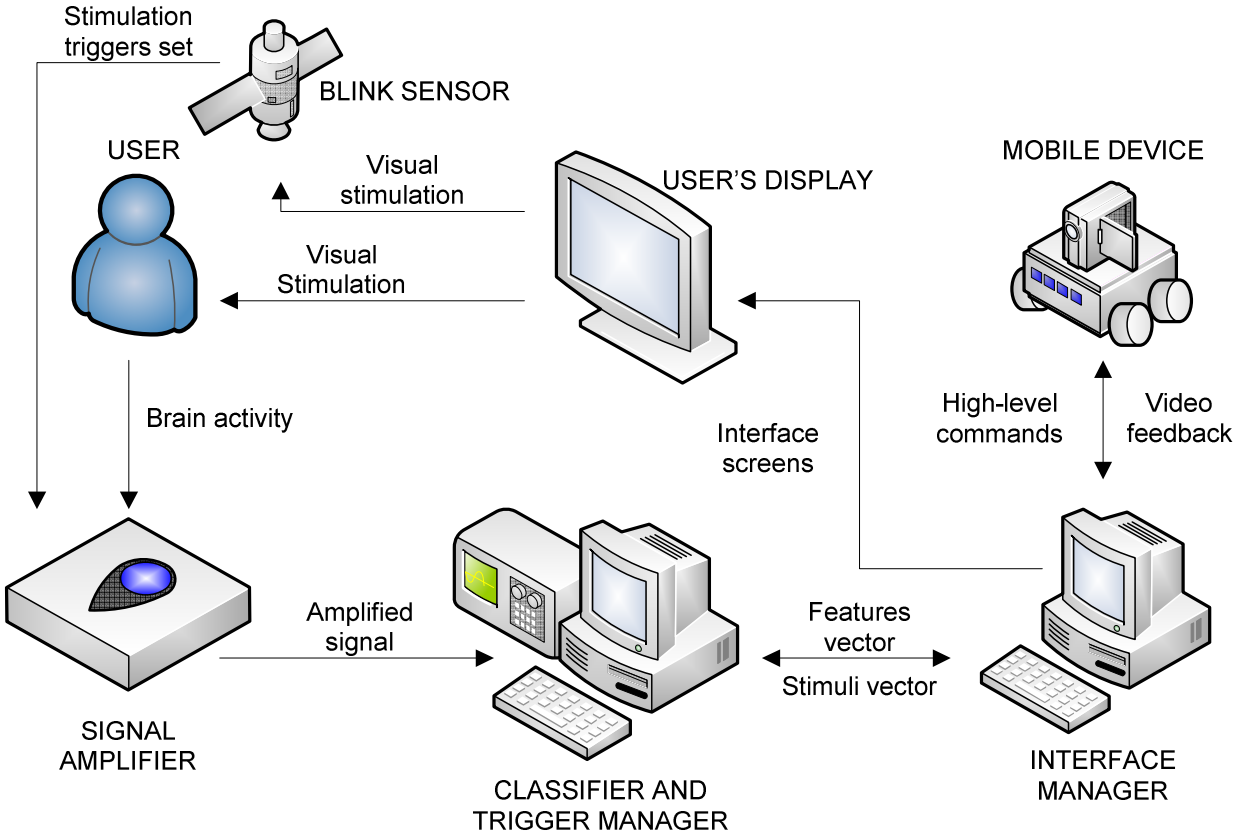


FIGURE 2.5. Proposed functional model for NEVRAROS..

When inner code calls opportune method for creating and show a stimulus, interface manager sends an acknowledgment signal to the classifier, waiting for classification as reply. Motivation for using two different but equivalents methods to communicate triggers of classification are not trivial: the use of software for sending triggers is more compact and require less devices, but delays can occur during transmission, because acknowledgment signal is a socket generate by a method which flows through the network that connect classifier and interface manager. Such those delays are not exactly determinable, and can vary in a little range. The use of external device allows quantification of delays by simple confronting arrival time of square waves with arrival times of acknowledgment sockets. Once delay is quantified, and the classifier is set up for managing it, blink sensor can be removed.

Features Overview

Main features of NEVRAROS system are listed below. Such this list do not cover components-specific features, but only overall features that the entire system provides to users.

- Properly connection between each system component. All components connection are properly tuned-up ensuring fast data transmission and with no data lose or corruption. In particular, communication link between classifier and interface manager is hard real-time validated by exhaustive simulation and testing for guarantee stimuli triggers and signal from brain activity are synchronized within an affordable time range.
- ICA-based Matlab classification algorithm. A proprietary, stable and validated algorithm for brain signal classification is mounted and ready for specific P300 waves-based use.
- goal-based selection algorithm with up to 4 different destinations selectable. A proprietary, stable and validated algorithm for interpret user's will and decision among selectable destinations.
- User interface modules manager for enhanced real time experience. A properly module manager administrates interface execution, choosing appropriate module each time without external support.
- User interface module for goal-based navigation. Patient is able to choose between four different destination each time of selection. Each destination corresponds to specific concrete object or places in real environment.
- User interface module for video feedback from mobile device. Patient is able to view the video stream from mobile device directly into user interface.
- User interface module for path and environments. Patient is able to select several environment for navigation, depending on where the mobile device is and if the location of the mobile device is already registered.
- User interface module for tuning up built-in user interface features. Specific settings are available allowing system to run into different modalities. Learning mode and testing mode are also available for user learning and testing sessions.
- Enhanced code management for fast adding new mobile devices or navigation environments. New locations and mobile devices can be easily imported or removed without affecting performances of the system.
- External analogical device for stimuli blinking feedback. An external blink sensor is included for system testing phases. New stimulation algorithms can be implemented, and blink device is directly connected to amplifier for controlling time delay between user interface visual stimuli instructions for classifier and patient brain response.

NEVRAROS Components

In order to better understand how NEVRAROS works, a brief overview of used technologies is useful. Note that some of the technologies presented below, even if perfectly functioning for the scope of this project, are obsolete. Improvements can surely be achieved selecting State-of Arts technologies, especially those devices concerning medical and neurological equipment.

Electrode cup and amplifiers

One of the keys to recording good EEG signals is the type of electrodes used. Electrodes that make the best contact with a subject's scalp and contain materials that most readily conduct EEG signals provide the best EEG recordings. Some of the types of electrodes available include:

FIGURE 2.6. Different types of electrodes. Ospedale San Camillo provides a full set of electrodes

- Reusable disks. These electrodes can be placed close to the scalp, even in a region with hair because they are small. A small amount of conducting gel needs to be used under each disk. The electrodes are held in place by a washable elastic head band. Disks are usually made of tin, silver, or gold. They can be cleaned with soap and water or Cidex.
- EEG Caps with disks. Different styles of caps are available with different numbers and types of electrodes. Some caps are available for use with replaceable disks and leads. Gel is injected under each disk through a hole in the back of the disk. Since the disks on a region of the scalp covered with hair cannot be placed as close to the scalp as individual disc electrodes, a greater amount of conducting gel needs to be injected under each. After its use, more time is required to clean the cap and its electrodes, as well as the hair of the subject.
- Adhesive Gel Electrodes. These are the same disposable silver/silver chloride electrodes used to record ECGs and EMGs, and they can be used with the same snap leads used for recording those signals. These electrodes are an inexpensive solution for recording from regions of the scalp without hair. They cannot be placed close to the scalp in regions with hair, since the adhesive pad around the electrode would attach to hair and not the scalp.
- Subdermal Needles. These are sterilized, single-use needles that are placed under the skin. Needles are available with permanently attached wire leads, where the whole assembly is discarded, or sockets that are attached to lead wires with matching plugs. Also, human subjects and, in some situations, regulatory committees need to approve the use of these electrodes before they are used.

Once electrodes are set up, the amplifier gets input signal directly from an examinee head. Amplitude of brain potentials measured directly on a scalp is about 100uV, and its frequency range is not strictly defined, we can say that typically most of spectrum energy is between 0.2Hz and 20Hz.

Beside the scalp potential, on amplifier entrance there is a polarization voltage (connection between scalp and electrode plate becomes a little battery with voltage of about 100mV, that voltage is called polarization voltage, it's always DC voltage – it makes it easy to remove it later in circuit).

On the amplifier entrance there is also a noise that is inducted in electrode wires (especially from electrical wires – 50/60Hz, but also from any other electromagnetic source). So first part of EEG amplifier is a passive RC filter that filters high frequency components.

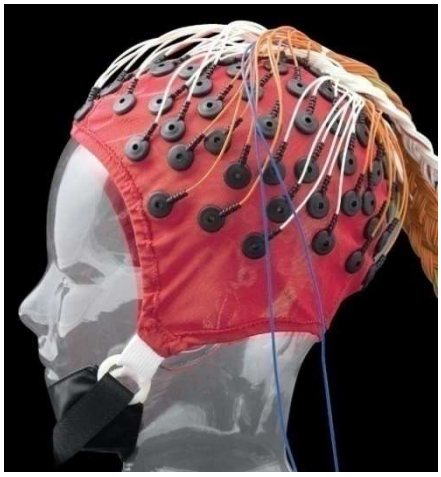


FIGURE 2.7. Placement of electrodes is a long and boring process that can be temporally reduced with using an electrode cap or some pre-assembled devices.

Next part is protection circuit that protects both, examinee from short circuit in device, and the device from connecting it to some power source instead of to brain.

After protection circuit there is an instrumentation amplifier. Instrumentation amplifier amplifies signal difference and rejects input signals common to both input leads. This is very important because noise is pretty the same on both instrumentation amplifier input leads (electrodes and electrode cables are very close so noise influence is same on both of them), so, due to its ability to reject input signals common to both input, it will reject the noise. Brain potentials are different on each electrode so instrumentation amplifier will amplify brain potentials because it amplifies signals difference. Finally, amplified brain potentials are digitalized.

IRCCS San Camillo Hospital provided a set of reusable disks made of silver, and a EEG and ERP amplifier from Compumedics Neuroscan.

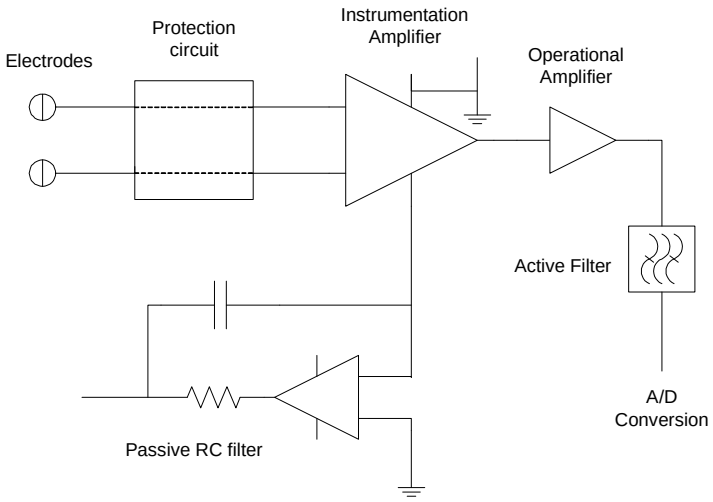


FIGURE 2.8. Circuit block scheme of typical EEG amplifier.

SCAN

The SCAN Acquisition software serves as the interface to the Neuroscan amplifier. SCAN acquisition provides a multitude of recording options which are saved in unique files that can be recalled for each experiment, ensuring that each individual data set is acquired with the same parameters. Even with the numerous options for acquiring the data the software is straightforward and simple to use. The SCAN Analysis software is a comprehensive tool for processing and analyzing EEG and ERP data. The latest advancements included a programming-based batch processing language, a PCA/ICA filter toolbox and EKG and Blink reduction tools. The SCAN system is divided primarily into two modules: ACQUIRE (for acquisition of data) and EDIT (for analysis of data). Many of the modules are self-contained, preset programs (with some variable parameters), although the Gentask module

allows you to create your own stimulus presentation sequence. The SCAN programs are accessed from the Program Launcher.

- ACQUIRE module. The ACQUIRE module is used for recording EEG and EP data. It records the data primarily in three formats - continuous stream (appears as scrolling EEG-like record), discrete epochs (stores series of discrete EP epochs), and averaged files. There are advantages and disadvantages to each acquisition type. In most instances you will want to take advantage of SCAN's ability to record the entire raw data file (continuous mode) - as opposed to storing only the epochs or averaged EP data. This allows to perform any number of offline analyses, while still having access to the original data.
- EDIT module. The EDIT module is used for transforming the data files in a number of ways, including offline filtering, re-referencing, baseline correction, editing the recordings for eye movement and other types of artifact, and manual review of individual sweeps. Spectral analysis (forward and backward FFT), coherence, mean frequency, global field power and filtering are among the types of analyses that may be calculated offline. 2D and 3D Mapping and 2D Cartooning are options in the EDIT program.
- WAVEBOARD program. The Waveboard is a program that is useful for displaying multiple waveforms from multiple files, and measuring points and differences between points on the waveforms.

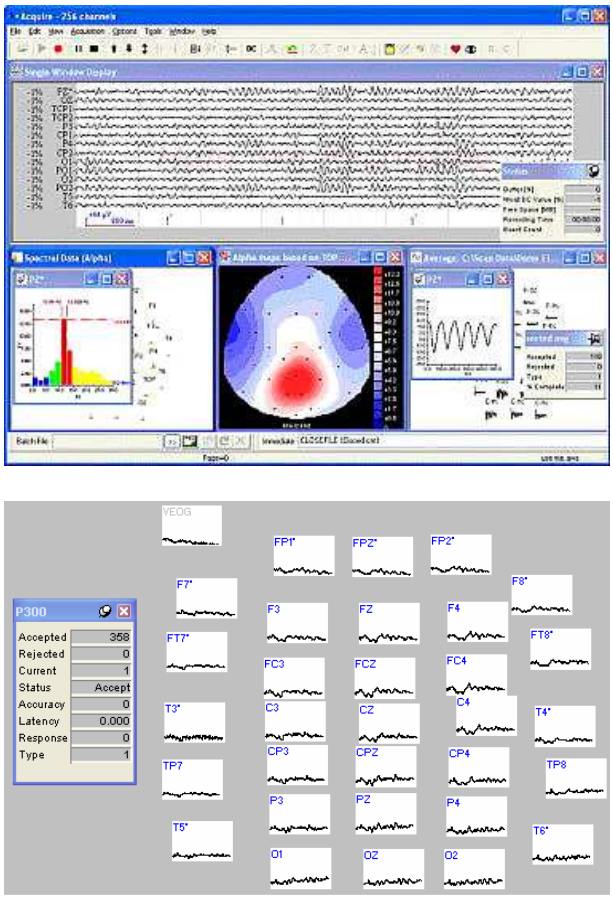


FIGURE 2.9. Screenshots from SCAN software. a) main window of ACQUIRE module, with EEG raw signal plotted and diagrams of brain activities. b) demo of 32-channel signal acquisition

HIM

The Hardware Interface Module (HIM) is a software designed in order to provide a solid structure for the acquisition, storage and visualization of the signal. HIM is an open-source software and was written in C++ language using cross-platform wxWidgets library (the actual build is for Microsoft Windows® only). HIM has a core block which handles all the task that are common to every protocols and which allow the loading of plug-ins. These plug-ins are compiled as .dll file and contain the algorithms that the user develops. Thanks to this approach there is a solid base platform, which will simplify the development of updates without rebuilding everything, and will make possible to share applications and algorithms without recompiling them. The HIM software supports now several kind of instruments for signal acquisition; some are real, other are virtual and are useful for debug and simulation purposes.

FIGURE 2.10. Main windows of HIM. Starting from the left, user can choose the device for signal acquisition, starting and stopping HIM execution, selects triggers and display plotted results

The compatible devices are: Kimera II (Sensibilab prototype); G.Mobilab (G.Tec – Austria); Neuroscan (Compumedics); Brain Products. With the interactive file player it is possible to load previously recorded signal from a file and play it at the same sample frequency. The instrument simulator can be used in order to load pieces of dataset; with this virtual device the operator can select a piece of acquisition in order to simulate a classification or a rest phase. The Hardware Interface Module can communicate with the supported devices both via Bluetooth® and TCP/IP.

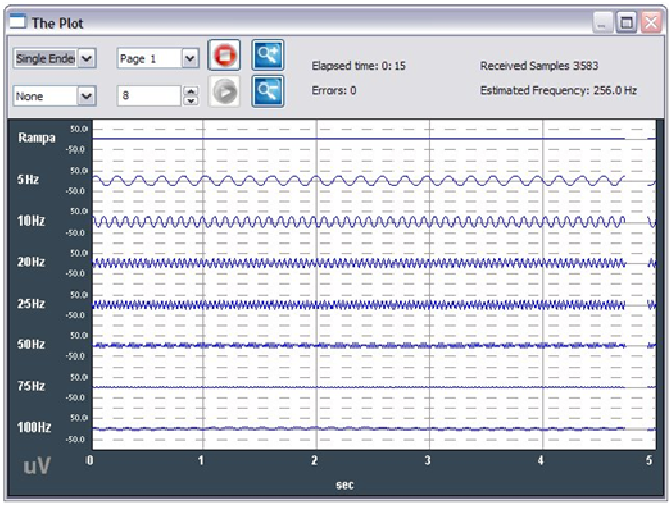


FIGURE 2.11. HIM plotted results of signal analysis and classification

HIM was designed both for laboratory purposes and daily life application: the use with double monitors enabled PC can increase the usability of this software. In Figure the main windows of the Hardware Interface Software is shown. The main window is used to select and connect the acquisition instrument, to activate data storage and to select the algorithm for signal processing. There are also six button to insert some operator activated signal trigger for general purpose applications. Two additional windows can be activated by pressing on specific buttons: one window is for signal visualization, the second allows to view the feedback calculated by the algorithm. These windows have a simple interface that allows to control the signal acquisition and feedback. Through the use of intuitive commands in the plot window, HIM allows also an elaboration of the signal (denoising, filtering, amplification…) to improve the signal view; it is possible to select the use of some simple spatial filters or online time domain filtering.

The basic class also handles all the operation related to data buffer management, the performance monitoring and, if necessary, the communication with Matlab®. With HIM a Visual C++ 2005 Project Wizard is also provided in order to assist developers during the creation of new algorithm classes. The wizard is based on a basic algorithm class which provides some functions that can be very important in the laboratory use of the system (eg. plot of the algorithm output). The result is a .dll file that has to be copied into the HIM working directory.

FIGURE 2.12. IRRLicht, BCI++, Sensibilabs, and Polimi Logos

AENIMA

The Graphic User Interface is a flexible tool developed in order to simplify the implementation of new operating protocols for laboratory testing or BCI based user applications. This module, named AEnima, is an independent application and was written in C++ language using a multiplatform graphics engine, in order to provide a more realistic and challenging experience to the user and guarantee versatility and efficiency in application development. The two modules, HIM and AEnima, are connected via TCP/IP and are thought to run on two different computer, as well is possible to run on the same one. The Figure shows the structure of AEnima. The core of the system is an open source high performance realtime 3D C++ graphic engine which allow the easy and fast creation of immersive environment for BCI applications and protocols with an high level of multimedia contents. The open source graphics engine (Irrlicht) supports both OpengGL and DirectX (ver. 8 and 9) library so that can be used also on computer with limited performance. Irrlicht has also its own rendering software; this allow to implement 2D and 3D mesh/models and use complex animations using high level functions to simplify the creation of complex virtual environments.

FIGURE 2.13. AEnima luncher.

The Audio Engine offers an high level set of functions which allow the reproduction and management of sound effects and audio files in different formats (eg. WAV, MP3, OGG). This engine allows also for positional and 3D audio experience which can be a useful in order to develop protocols and paradigms with auditory stimulation (ASSR) or feedback. The Event Manager module is dedicated to the management of the incoming and out coming messages from and to the Hardware Interface Module through the socket module. This module manages also keyboard and mouse events thus allowing the easy interaction with AEnima by the operator. The Socket module is dedicated to the management of the TCP/IP socket connection with the Hardware Interface Module server. The communication is made by means of proprietary message, named BCI_Message, so to efforts the stability of the communication.

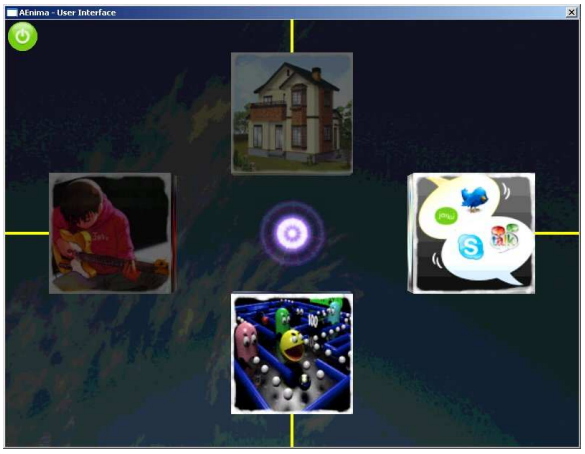


FIGURE 2.14. Aenima window screenshot during a SVEEP simulation

The protocol dealer module is an important part of the software that coordinates the execution of protocols and user applications. The system presents also two stimulation modules; the first one is a stimulation module that sends message via USB in order to control SSVEP external simulator. The second one is a software module that can send command via TCP/IP to a FES controller. A specific software module was also implemented in order to provide an application layer with an home automation system.

In order to easily develop a new protocol, AEnima has a dedicated Visual C++ Project Wizard that allows to implement a working code in few simple steps. The new protocol can be added to AEnima as a plug-in using

dynamic library loading (DLL) guaranteeing rapid and stand-alone update and easy redistribution of new protocols and applications. In order to manage a large amount of protocols and to modify start-up options, a dedicated launcher tool (AEnima launcher) has been developed. With this application you can directly run a protocol or create a shortcut with preset start-up options for a certain protocol.

Blink sensor

The NEVRAROS BCI system uses a stimulation paradigm that exploits the evoked potential by visual stimulation to generate and classify a particular waveform (P300). Considering that the time parameters of generation and classification belong to an order of a few tens of milliseconds, the temporal accuracy in terms of synchronization between evocating and classifying istant must be ensured. No ensuring that synchronization leads to a wrong selection of the time interval jeopardizing the classification process.

Non synchronization may depends on factors such as organization and management of the network connection between the components, organization and management of computational procedures for the generation of stimuli and classification of EEG traces, organization and management of computational procedures for sending and receiving data between the components.

Concerning the two latter, no substantial changes to components code for eliminate or at least check the source of delays is feasible in terms of practicality, without deep modifications of components. Experimental data show a deterministic delay (constant, and therefore manageable) of 300 ms plus a random delay that falls within a time interval of 60 ms acquisition window for the classification on the generation of stimulation. Not being able to ensure greater precision turning this software, a solution has been identified by using an external analog electrical component that, on one hand, acquires the timing of stimulation to the actual time of generation, and the other provides a waveform significantly in parallel with EEG traces received by component classification.

FIGURE 2.15. Blink sensor. The black cap store the photodiode and assure protection against external artifacts.

Specifically, Blink sensor is a craft electrical device created in order to detect visual stimulation offered by the user interface. This device is composed by a photodiode, which observes light emission changing frequencies, and an electrical circuit that transform the information given by the photodiode in a suitable electrical signal for the amplifier. Blink sensor is an analogical device, and it can distinguish between only two light emission levels. Low level is set to zero (equivalent to no significant light emission detected), while a potentiometer allows user to manually select suitable range for high level. A led is also provided for external visual feedback, and works as follow: a solid green represents an high light emission, while no reaction represent no light emission. The device is battery powered, and is provided with suitable external cables terminating with a jack that transmit output signal, for connecting it to the amplifier headbox.

NEVRAROS Display Interface

NEVRAROS display interface represents the core of all the system. It is an Aenima dependent dynamic library loading which contains algorithms and code for user stimulation, mobile device control, video feedback from mobile device control and trigger generator for HIM. It is written in C++ language, for complete integration with Aenima framework. Through using this Display Interface, user can select suitable mobile device to command, select preferred destination and "live" the telepresence experience.

NEVRAROS Display Interface can be divided into four logical modules:

- Primary module: Starting module that control the overall subsystem. Primary module calls Supervisor module for setting up specific parameters of selection and navigation, and manages Selection and Navigation module execution and activity. It also loads all necessary resources and graphical elements of the interface.
- Selection module: module that control the destination selection paradigm. A central ring containing a cursor is presented to patient. In each edge of the ring a directional arrow represents one of the four possible destinations achievable at each iteration. Near each arrow a small picture representing the destination is also available. The four arrows blink in casual sequence, and their blinking represents the visual stimulation for the patient. He must express his cognitive act of concentration every time the arrow indicating the destination he decided to reach blinks. Each time a cognitive act of concentration is recorded and connected to a particular blink, the cursor moves for one step in the corresponding arrow's direction. Once one of the edges is reached by the cursor, the Interface choose that destination as selected destination, sends to mobile device opportune commands for reach it in the real environment and calls Navigation module for execution. Since visual stimuli are limited in number, reaching no target determines no Navigation module calling nor commands to mobile device sending. In such this case, Selection module wait for a little time and then ask for a new selection session.
- Navigation module: module that control the navigation process of the mobile device and presents to user the video feedback in coming. A central window shows the

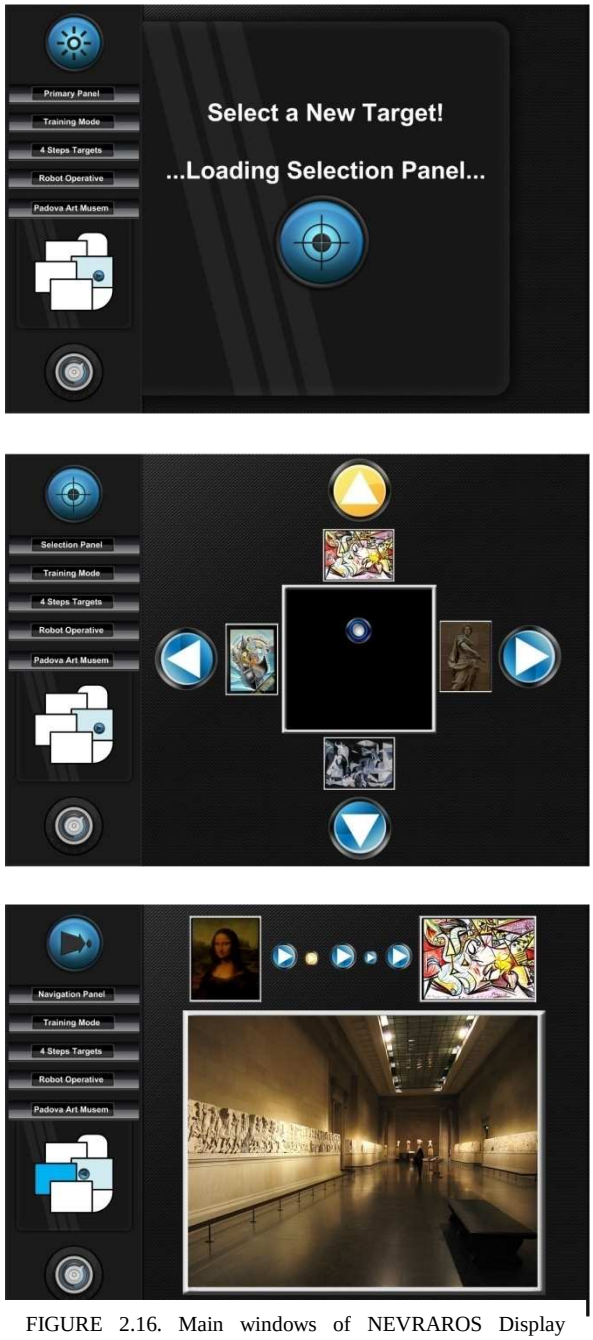


FIGURE 2.16. Main windows of NEVRAROS Display Interface. a) Primary module, which also load graphic elements such as mini map of the overall environment. b) Selection module. Upper arrow is blinking for stimulating patient, and central cursor is moving for reaching desired destination. c) Navigation module. While mobile device is reaching selected destination, a video stream is presented to user, as well as an intuitive indication of path selected (the room where the target resides is highlighted) and the overall progress of the path (little arrows on top indicate percentage of path already crossed.

video stream from the mobile device. Upper indicators shows overall progress in the path. Once mobile device reach selected destination, a binary choice is offered to user, with the same paradigm of previous selection task. Here user can decide if reach a new target or else take a look again to current destination achieved. Once such this decision is made, Navigation module calls back Selection module for a new selection session.

- Supervisor module: this module is hidden from user interface, and is visible only from secondary screen dedicated to system supervisor. Here supervisor can tune the interface up for the navigation session, selecting which modality use for next session (learning, testing or real navigation), how many elementary steps the central cursor must do for reach external edges, and more. From here a blinking square is also selectable, if use of Blink sensor is expected.

In order to easily add new paths and environments to navigate into, a dedicated manager control periodically which environments are available. As for the environments, mobile devices are controlled too.

Available robots

The project is developed connecting a mobile robot to BCI system. Mobile robots laboratories from Università degli Studi di Padova and IRCCS San Camillo Hospital provided two robots for research.

Fred is a an holonomic robot produced by "Team Artisti Veneti" and builded with hexagonal structure and three omnidirectional wheels. In this way the robot is able to move to any position in the plane without rotate itself. Odonometry is controlled by motor board connected to on board computer trought serial port. The robot is also fit with a framegrabber for video acquisition and an audio board and WiFi connectivity. The robot also has an omnidirectional camera with an hyperbolic mirror.

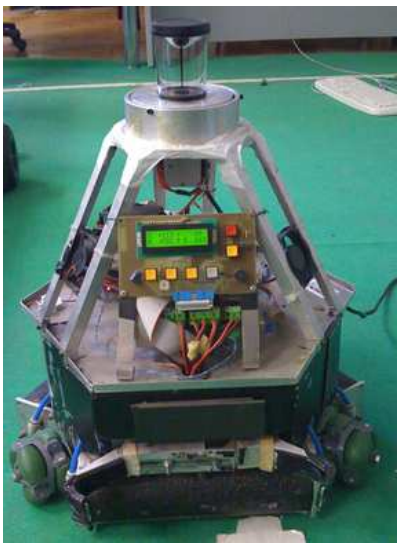


Figure 2.18. Team Artisti Veneti Fred holonomic robot

Figure 2.17. WowWee Rovio holonomic robot

WowWee Rovio$_{TM}$ is a mobile wireless IP camera with a three-wheeled drive system. Rovio is equipped with an IR sensor on the front for basic obstacle avoidance. Rovio also has a NorthStar II sensor (also known as the TrueTrackTM sensor in WowWee terms). This sensor enables the robot to self-navigate as it follows pre-programmed paths. The NSII (NorthStar II) system reads the two IR spots projected onto the ceiling by the TrueTrack Beacon integrated into the Rovio docking base or projected by a Rovio TrueTrack standalone beacon (Room Beacon). The data acquired from the NS2 sensor provides an x- and y- coordinate and theta as well some other useful information.

NEVRAROS Implementation

Let's consider some more NEVRAROS functional details; previously a general system description from components point of view was described. Now the system is presented from activities point of view. A general scenario for system behavior can be describe as follow. Main actors and object of NEVRAROS system can be listed as:

- (System) User. Healthy or disabled patient which is going to use NEVRAROS.
- (System) Supervisor. External technician supporting patient in system setting up. (necessary in preliminary phases, and optional in mature system utilization)
- Aenima. User interface and communication with HIM and mobile device manager module.
- HIM. Classifier, amplifier and communication with Aenima manager module.
- Robot. mobile device and communication with Aenima manager module.
- Blink Capt. blink sensor and communication with HIM manager module.

First of all, Supervisor implants electrodes set on the User cranial skin, and connects electrodes to amplifier via amplifier headbox. Once electrodes are set up properly, Supervisor execute HIM and Aenima. Concerning about HIM, Supervisor selects patient's profile and load its own data set (if exist) into classifier. In the meantime, he also set Aenima up (selecting parameters for defining if the system is in learning or testing mode, how long session will be and more). In the meantime, robot is also initialized, ready to follow User's instructions. Blink Capt. is also used, but only in first-time testing, for controlling time synchronization between different modules. With the overall system ready, Aenima is started, and it offers several visual stimulations to the User. The latter reacts to stimulations with an EEG alteration, that is acquired and analyzed by HIM (and the classifier within it). Results of such that classification is a message sent to Aenima, describing if a positive classification occurred. Consequently, Aenima decides on sending to the Robot high level commands for movement in real environment. The visual feedback is finally shown to User.

Preparing the electrode cup and setting up amplifiers

The International 10–20 System of Electrode Placement is the most widely used method to describe the location of scalp electrodes. The 10–20 system is based on the relationship between the location of an electrode and the underlying area of cerebral cortex. Each site has a letter (to identify the lobe) and a number or another letter to identify the hemisphere location. The letters used are: "F"-Frontal lobe, "T"-Temporal lobe, "C"-Central lobe, "P"-Parietal lobe, "O"-Occipital lobe. Even numbers (2,4,6,8) refer to the right hemisphere and odd numbers (1,3,5,7) refer to the left hemisphere. "Z" refers to an electrode placed on the midline. The smaller the number, the closer the position to the midline. "Fp" stands for Front polar. "Nasion" is the point between the forehead and nose. "Inion" is the bump at the back of the skull. In the 10–20 system, centimeters (cm) are the unit of measure. 10% and 20% refer to interelectrode distance derived from three main measurements:

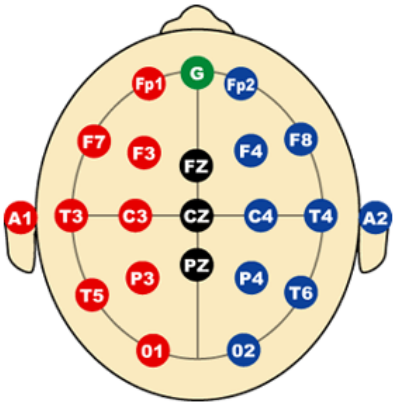


Figure 2.19. Site organization in 10-20 system.

nasion–inion, preauricular points and circumference of the head. Neutral/Ground: A scalp electrode affixed to the midline forehead or other relatively "neutral" site. The neutral (or ground) electrode is important as it is used by the system to reduce the effect of external interference

This procedure should be left to those who have developed such skills or EEG technicians, to ensure the safety of the patient. Correct placement of electrodes is important and elimination of artifacts is a worthwhile challenge. In order to obtain good tracings in the EEG, with little interference of artifact, the contact of the electrode to the skin must be optimized. This is done by scrubbing the skin to remove dead or dry cells that do not conduct electricity well, by scratching the skin to minimize motion potentials in the skin, by defatting the skin to permit the electrode to grip the skin and by using conductive gel.

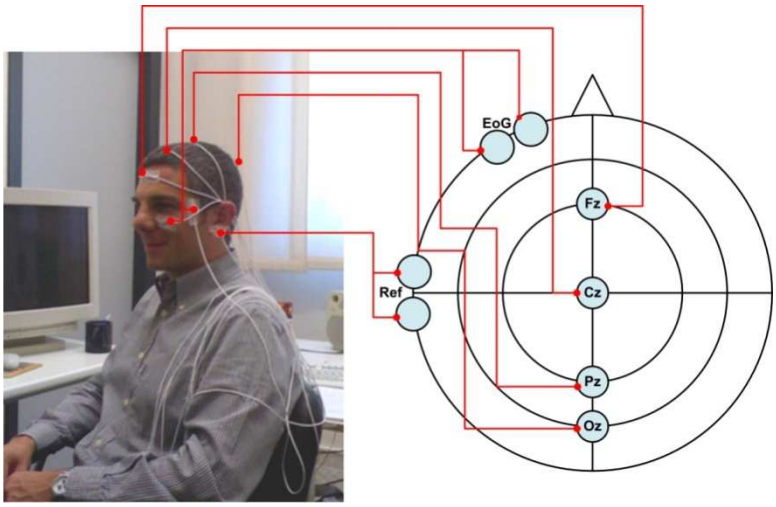


Figure 2.20. Positioning of electrodes into expected sites.

For the scope of P300 classification, 4 sites are enough for signal acquisition. Midline is choose for equalize different hemisphere contribute to resulting signal. Fp1 site is also included for acquiring eyes muscles activity and removing it from brain signal.

Once electrodes are correctly placed, signal acquisition should begin. First of all SCAN software for Neuroscan amplifier must be opened, and each electrode impedance must be checked. This procedure allows to determinate whereas an electrode has interference problems. Once all electrodes shows impedance minor than 28KOhms, Signal acquisition can starts with SCAN support. The result is a suitable set of powered-up signals for HIM classifier.

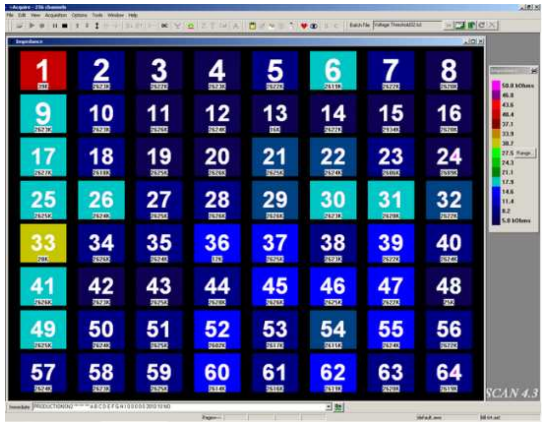


Figure 2.21. Checking electrodes impedance by SCAN Acquire tool.

Exporting classifier

In order to understand whether a P300 pattern has been generated by the visual stimulus, a Support Vector Machine was developed. Generally speaking, the Support Vector Machine implements the following idea: it maps the input vector x into a high-dimensional feature space Z through some non-linear mapping K, chosen a priori. In this space, a hyper plane is constructed. This hyper plane, in our case, separates the P300 patterns from the non-P300 patterns. The core of a SVM classifier is the kernel function

$$K(x) \rightarrow Z$$

One of the most used kernel functions, as in our experimental sessions, is the radial basis kernel

$$K(x) = e^{-x^2}$$

Using the SVM classifier, the following issues have been observed:

- A fast learning rate: typically a few seconds are sufficient to learn the training set.
- Quite coherent results between the off-line training, the testing phase and the real-time phase.
- Good numerical stability.

The raw signal, acquired by Neuroscan, follows four processing steps: first, all the signals recorded by scalp electrodes are processed in order to obtain a set of independent components. Since the locations of the brain that generate ERP cannot be determined easily by the scalp recordings (resolution problem), many algorithms have been studied in order to separate each signal in a set of independent sources (i.e., originating from different areas). One of the most promising algorithms is the socalled Independent Component Analysis (ICA). ICA determines what spatially fixed and temporally independent component activations compose an observed time-varying response, without attempting to directly specify where in the brain these activations arise. Practically the problem that ICA solves, is to recover sources from their instantaneous mixture without any previous knowledge of the sources and the mixing channel. Differently from Principal Component Analysis PCA that finds components that are uncorrelated, ICA is a much stronger criterion because it is based on statistical moments of a higher order, so ICA requires more than the uncorrelatedness of the components. The most general case can be so characterized: we consider n unknown sources signals $s_i(t)$, i = 1,…n, which are mutually independent, and we model the sensor's output as

$$s(t) = Ax(t)$$

where A is an unknown non-singular mixing matrix, $x(t)= [x_1(t), \ldots x_n(t)]^T$, $s(t)= [s_1(t), \ldots s_n(t)]^T$. With no knowledge of the source signals and the mixing matrix, we want to recover the original signals from the observed signals x(t) by the following linear transformation:

$$y(t) = Wx(t)$$

where $y(t)= [y_1(t), \ldots y_n(t)]^T$ and W is the un-mixing matrix. Of course it is impossible to find the original sources without ambiguity, because they are not identifiable in a strictly statistical sense. However, up to some permutation, it is possible to obtain $c_i s_i(t)$ where c_i are unknown non-zero scalar factors. In order to separate the components, ICA works on a learning algorithm that minimizes the dependency between the output components: such a dependency is measured by the Kullback-Leibler divergence between the joint and the product of the marginal distributions of the output:

$$D(W) = \int p(y) \log \frac{p(y)}{\prod_{a=1}^{n} p_a(y_a)} dy$$

Where $p_a(y_a)$ is the marginal probability density function (pdf). To perform this, some hypotheses are implicit and a training algorithm is needed to find the right un-mixing matrix W. The hypotheses are the following:

1. The signals recorded from the electrodes are an instantaneous mix of n statistically independent sources. This implies that the coefficients of the mixing matrix A are linear and time-independent. From a physiological point of view this is equivalent to saying that the sum of the electrical potentials coming from different areas of the brain on the scalp electrodes, is linear. To be more precise, it is not the result of non-linear distortion or temporal convolution of the sources.
2. The number of sources n does not exceed the number of electrodes. In physiology this means that the areas involved are stable and in a defined number.

3. The sources and the mixing process are stationary, they don't change their statistical properties in time.

The first hypothesis is well confirmed in literature. The second hypothesis doesn't represent a problem because we can take as many sources as we want (in theory at least), in order to have the number of sources smaller than the number of electrodes. The third one is generally not verified but we can overcome this problem if we choose a time interval, small enough to consider with a good approximation, the signal stationary. According to these considerations we can apply the independent component analysis computation.

Connecting HIM and AENIMA

We already said Aenima and HIM are both products of Sensibilab. They are built in order to communicate each other via TCP/IP connection. AEnima main program is a multi-thread software. There is a main thread used to run a loop which control and update the graphics, and evaluate the active protocol. The secondary thread is used to control socket: the socket thread allow to manage the communication via TCP/IP between AEnima and the Hardware Interface Module. The communication is managed by HIM (the server), so HIM must be started before AEnima to instantiate the socket connection. Both socket and thread are described and implemented into the SocketComm and SocketThread classes. A snipped from that two classes headers files shows main functions and methods provided.

```
class SocketComm: public SocketComm_base
{
	public:
		SocketComm(void);
		~SocketComm(void);
		Function to create the socket,
		int Open(char *ipAddress, int SockPort);
		Function to connect the software with the created socket.*/
		void Connect(void);
		This function is used to send data to the server.*/
		int SendData(BCIMessage *myDataIn);
		This function closes the socket connection.*/
		void Close(void);
		Function to receive data from server.*/
		int RecvData(BCIMessage *myDataIn);
	};
}

struct ThreadDataStruct
{
	int   GoOn;
	SocketComm mySock;
	BCIMessage bufDataIn;
	LedDriver   myLed;
	IrrlichtDevice* myDevice;
	SEvent      event;
};

DWORD WINAPI RecvThread( LPVOID lpParam );
DWORD WINAPI SendThread( LPVOID lpParam );
void CreateRecvThread(int *isWindowActive, SocketComm *mySock, LedDriver *myLed,
IrrlichtDevice *myDevice);
void GetIncomingData(char *buffer);
void ReleaseSocketThread(void);
void StopRecvThread(void);
```

```
void StartRecvThread(void);
```

AEnima and Him communicate using specific message called BCI_Message**.** BCI_Message has the structure shown below:

```
typedef struct BCIMessage
{
	int Kind;
	int Value;
	int Buffer[16];
	int Check;
}BCIMessage;
```

Kind indicates which message was transmitted/received. *Value* and *buffer[16]* are two variables which can be used to send data. You can send a single number using value or an array of max 16 values using buffer. Creation of new BCIMessage is achieved by calling the function ComposeMessage; This function automatically sets the message's structure and calculate the error checking variable:

```
BCIMessage ComposeMessage(int Kind, int Value, int *pBuffer,int BuffSize)
```

Messages are defined in a small set of integer values, each one representing a specific message:

```
	//Instrument control
	const int	START_ACQ			=	101;
	const int	STOP_ACQ			=	102;
	const int	CONNECTION_COMPLETED		=	103;
	//Triggers event
	const int	GRAPH_TRIG			=	201;
	//Messages from algorithm to AEnima
	const int	CLASSIFICATION			=	301;
	const int	FEEDBACK			=	302;
	const int	SET_PARAMS			=	303;
	//Messagese from AEnima to algorithm
	const int	RESET_ALGORITHM			=	401;
	const int	REQUEST_PARAMS			=	402;
	//Generic messages
	const int	BCI_OK				=	501;
	const int	BCI_ERROR			=	502;
```

START_ACQ is sent by AEnima to HIM to indicate that the protocol is starting and request to activate the acquisition and registration of signals.

STOP_ACQ is sent by AEnima to HIM to inform that the protocol is stopping and to deactivate the signals acquisition.

CONNECTION_COMPLETED is the message sent by HIM to AEnima in response to START_ACQ message. HIM sends this message only if the system is ready for an acquisition. If Hardware Interface Module is not ready, AEnima doesn't receive this message and protocol isn't activated.

GRAPH_TRIG message is sent by AEnima to HIM and can be used to trigger signals acquired by HIM depending on event occurred in the graphics user interface.

CLASSIFICATION message is sent by HIM to AEnima when on the Hardware Interface Module an algorithm for the classification of the signal is active. Classification message is also used by AEnima with arrow keys and false SockeEvent so that it's possible to test protocol without HIM connection.

FEEDBACK message, like the classification one, is sent by HIM when an algorithm is active.

REQUEST_PARAMS is used by AEnima to request some parameters from Hardware Interface Module.

SET_PARAMS is the response from HIM to REQUEST_PARAMS message. This message is sent only if algorithm is active.

RESET_ALGORITHMS message is sent by AEnima to reset the active algorithm on HIM.

BCI_OK and *BCI_ERROR* are additional messages for different uses.

User Protocol - Class dependences and class hierarchy

User Protocol is the creative core of the overall system. It offers to the User a graphic interface where stimulations take place, targets are achieved and visual feedback is shown from the robot mobile webcam. User protocol consists of several classes and libraries:

- class Nevraros: this is the main class which implements VEP stimulations, sends and receives information from/to the robotic device and manages and controls classification information from/to HIM and other modules. It has strong dependency with class ProtocolMngr. It uses most of ProtocolMngr overloaded methods for creating a graphic interface in a 3D Irrlicht-based virtual environment, and it sends messages to HIM using defined BCIMessage objects.
- class IGraphicEngine: this class provides static methods for controlling and modifying graphic elements created within Nevraros class. It uses methods provided by Irrlicht graphic engine for create SceneNode (or else, 3D graphic objects) and control them in the 3D environment.
- class ILocations: this class provides static methods for creating and controlling locations and targets. Here dependencies between different targets are defined. Briefly, each target is characterized according to goal-based approach. Every target object has a defined numbers of neighbors that will be displayed once the target is reached.
- class IStim: this class provides static methods for controlling and creating randomized stimuli, check whenever a collision occurred has some particular interest (target or non target).
- library MRPT: this is an external library used for implementing TCP/IP and HTTP connection with the robotic device. Mobile Robot Programming Toolkit (MRPT) provides C++ developers an extensive, portable and well-tested set of libraries and applications which cover the most common data structures and algorithms employed in a number of mobile robotics research areas: localization, Simultaneous Localization and Mapping (SLAM), computer vision and motion planning (obstacle avoidance). Key points in the design of MRPT are efficiency and reusability of code. The libraries include classes for easily managing 3D(6D) geometry, probability density functions (pdfs) over many predefined variables (points and poses, landmarks, maps), Bayesian inference (Kalman filters, particle filters), image processing, path planning and obstacle avoidance, 3D visualization of all kind of maps (points, occupancy grids, landmarks,...), etc.
- class RovioCommander: this class provides high level commands for controlling Rovio mobile Robot via HTTP connection. Methods for acquiring video streams and reports on Rovio status and position are also implemented here.

A useful UML-like class dependency diagram shows how different classes work together within NEVRAROS software. Note that return type and type parameter are omitted.

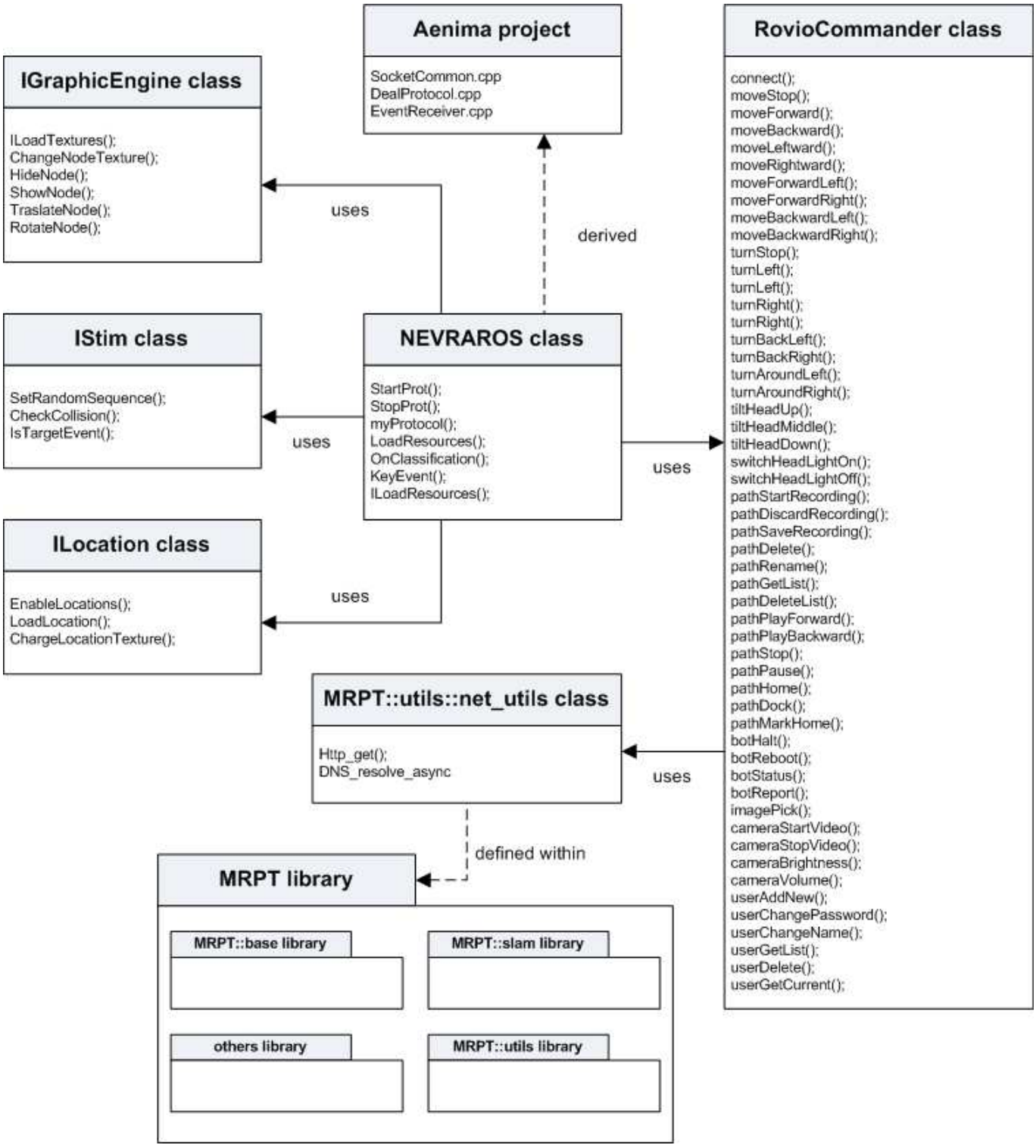


Figure 2.22. Class and libraries organization of NEVRAROS system. NEVRAROS class is the main core of the overall system. It uses IGraphicEngine, IStim, ILocation, RovioCommander classes for serve to patient high level GUI. Note that NEVRAROS uses Aenima SocketCommon class for connecting via TCP/IP mode with HIM, while indirectly use MRPT::utils::net_utils methods for connecting via HTTP mode with Rovio remote mobile robot. MRPT library is huge, and only few high-interesting parts are shown in the diagram. Aenima organization is partially shown too. Many MRPT libraries contains methods and approach for high and low level robotic processes, and it will be useful for future works enhancing NEVRAROS system. RovioCommander methods are created following WowWee directives and approaches presented in Rovio API specifications, V. 1.3.

User Protocol - Virtual Graphic Environment

The graphic environment is provided by Aenima using Irrlicht enigne. When a new Aenima protocol project is created, a Visual C++ Studio Wizard application created autonomously a 3D environment, which is populated and controlled within the project. The Irrlicht Engine is a cross-platform high performance realtime 3D engine written in C++. It features a powerful high level API for creating complete 3D and 2D applications such as games or scientific visualizations. It comes with an excellent documentation and integrates all state-of-the-art features for visual representation such as dynamic shadows, particle systems, character animation, indoor and outdoor technology, and collision detection.

The basic environment consists in an empty 3D space, visible by a central-focus camera point. Environment is populated by creating new 3D objects. Note that because of user interface only needs 2D environment, every 3D object is hence created setting to zero z-axis dimension. All object is referenced by an unique ISceneNode variable: selection of different nodes is provided by assigning to them unique names. Once an object is created and initialized with 3D environment parameters, texture, position and blending options can also be tuned.

```
mySceneNode = AEngine->smgr->addCubeSceneNode(146.0f,0,-1,
            core::vector3df(0,0,0),
            core::vector3df(0,0,0),
            core::vector3df(1.34f,1.0f,0.0f));
mySceneNode->
      setMaterialTexture(0, AEngine->driver->getTexture(texture relative path));
mySceneNode->
      setPosition(AEngine->camera->getTarget() + core::vector3df(3D position));
mySceneNode->
      getMaterial(0).EmissiveColor = video::SColor(0,255,255,255);
mySceneNode->
      setVisible(true/false);
 mySceneNode->
      setName(Object unique name);
```

Objects behavior is controlled thanks to IGraphicEngine class. It provides a core set of methods for moving, changing texture, setting position and setting visibility of graphic objects. Each node is called using its unique name. Note that Irrlicht strategies ask for direct calling the selected object with mySceneNode= AEngine->smgr->getSceneNodeFromName(aName). Once this call is done, mySceneNode will refer to the same object untill a new call of mySceneNode= AEngine->smgr->getSceneNodeFromName(aName).

```
void ChangeNodeTexture(const c8* nodeName, const c8* aTexture)
{
      mySceneNode= AEngine->smgr->getSceneNodeFromName(nodeName);
      mySceneNode->setMaterialTexture(0, AEngine->driver->getTexture(aTexture));
}

void HideNode(const c8* nodeName)
{
      mySceneNode= AEngine->smgr->getSceneNodeFromName(nodeName);
      mySceneNode->setVisible(false);
}

void ShowNode(const c8* nodeName)
{
      mySceneNode= AEngine->smgr->getSceneNodeFromName(nodeName);
      mySceneNode->setVisible(true);
}
```

```
void TraslateNode(const c8* nodeName, float x, float y, float z, int time)
{
	mySceneNode= AEngine->smgr->getSceneNodeFromName(nodeName);
	AEngine->movement = AEngine->smgr->createFlyStraightAnimator((mySceneNode-
	>getPosition()),(mySceneNode-
	>getPosition()+core::vector3df((f32)x,(f32)y,(f32)z)),time,false);
	mySceneNode->addAnimator(AEngine->movement);
}

void RotateNode(const c8* nodeName, float xrotationPerSecond, float
yrotationPerSecond, float zrotationPerSecond)
{
	mySceneNode= AEngine->smgr->getSceneNodeFromName(nodeName);
	AEngine->movement = AEngine->smgr-
	>createRotationAnimator(core::vector3df((f32)xrotationPerSecond,(f32)yrotationP
	erSecond,(f32)zrotationPerSecond));
	mySceneNode->addAnimator(AEngine->movement);
 }
```

User Protocol - Local Location Management

We will see later in this chapter how paths and location will be managed by Rovio robot. As paths and target are managed in real environment, so it has to be in the virtual graphic environment. ILocation class was implemented for achieving this goal. Anytime NEVRAROS system starts is execution, user is asked to choose an environment of experiment, both in Training/Testing and in Free user mode. Selecting different location involve that different targets will be available. Each target set is build according to goal-selection approach, so the user won't be asked to manually drive cursor/robot for the path selected, but only define the target he want to reach. Anytime a target is reached, a new set of neighbors (or else, a new set of target directly reachable from the current position) will be displayed. EnableLocation() is the method that loads positional information concerning the set of target selected. Such that information are to be previously stored within the method in implementation activity. At current time no external ways to add new location are ready for use. Selection of different location is provided by LoadLocation() method. Particular attention is needed in populating such locations when we want to create a real location (which means that location will be some location where a Rovio robot is operating). Synchronization in path and targets both for Rovio management and local management must manually be achieved.

User Protocol - Rovio Management

This section talks about using Rovio as robotic device for the presented BCI system. A brief overview of WowWee Rovio was presented previously, and now we expand such that presentation in order to underline most important hardware and software concepts for managing such that robotic device.

Rovio’s embedded intelligence is powered by a main processor (Marvell "PXA270M"), with clock frequency TBD (there is a 24MHz crystal) and a main Memory of 8MB RAM (2MB flash). Rovio acts like a web server. Using the WowWee provided software, th computer's browser connects to that web server and the Rovio provides web pages for the user interface. The ARM processor runs the web server, webcam, media streaming and general robot control. It runs the open-source "eCos" operating system. The media streaming server is based on a variant of the "spook" media streaming server. It uses HTTP protocol to transmit commands and send back to application

suitable ack signals, and it uses RTSP protocol to stream video and audio from the robot webcam and microphone. The RAM memory can be directly accessed using special URLs, so high level programming is possible using special URLs from a PC program (either a standalone program, or special JavaScript inside a webpage). Most of the functionality is performed in .CGI scripts, some with URL args or http request POSTed args.

HTTP and RTSP protocols.

The Hypertext Transfer Protocol (HTTP) is an Application Layer protocol for distributed, collaborative, hypermedia information systems.

HTTP is a request-response protocol standard for client-server computing. In HTTP, a web browser, for example, acts as a client, while an application running on a computer hosting the web site acts as a server. The client submits HTTP requests to the responding server by sending messages to it. The server, which stores content (or resources) such as HTML files and images, or generates such content on the fly, sends messages back to the client in response. These returned messages may contain the content requested by the client or may contain other kinds of response indications. A client is also referred to as a user agent (or 'UA' for short). In between the client and server there may be several intermediaries, such as proxies, web caches or gateways. In such a case, the client communicates with the server indirectly, and only converses directly with the first intermediary in the chain. A server may be called the origin server to reflect the fact that this is where content ultimately originates from.

HTTP is not constrained in principle to using TCP/IP, although this is its most popular implementation platform. Indeed HTTP can be "implemented on top of any other protocol on the Internet, or on other networks." HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used. Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs), or, more specifically, Uniform Resource Locators (URLs) using the http or https URI schemes.

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request. It establishes a Transmission Control Protocol (TCP) connection to a particular port on a host (typically port 80). An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested resource, an error message, or some other information. HTTP defines nine methods indicating the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

- HEAD: Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.
- GET: Requests a representation of the specified resource.
- POST: Submits data to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.
- PUT: Uploads a representation of the specified resource.
- DELETE: Deletes the specified resource.
- TRACE: Echoes back the received request, so that a client can see what changes or additions have been made by intermediate servers.
- OPTIONS: Returns the HTTP methods that the server supports for specified URL.

- CONNECT: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.
- PATCH: Is used to apply partial modifications to a resource.

The Real Time Streaming Protocol (RTSP) protocol is used for transferring real-time multimedia data (for example, audio and video) between a server and a client. It is a streaming protocol; this means RTSP attempts to facilitate scenarios in which the multimedia data is being simultaneously transferred and rendered (that is, video is displayed and audio is played). RTSP typically uses a TCP connection for control of the streaming media session, although it is also possible to use UDP for this purpose. The entity that sends the RTSP request that initiates the session is referred as the client, and the entity that responds to that request is referred to as the server. Typically, the multimedia data flows from the server to the client. RTSP also allows multimedia data to flow in the opposite direction. However, the extensions defined in this specification were not designed for such scenarios. Clients can send RTSP requests to the server requesting information on content before a session is established. The information that the server returns is formatted by using a syntax called Session Description Protocol (SDP). Clients use RTSP requests to control the session and to request that the server perform actions, such as starting or stopping the flow of multimedia data. Each request has a corresponding RTSP response that is sent in the opposite direction. Servers can also send RTSP requests to clients, for example, to inform them that the session state has changed. If TCP is used to exchange RTSP requests and responses, the multimedia data can also be transferred over the same TCP connection. Otherwise, the multimedia data is transferred over UDP.

The RTSP protocol has similarities to HTTP, but RTSP adds new requests. While HTTP is stateless, RTSP is a stateful protocol. A session identifier is used to keep track of sessions when needed; thus, no permanent TCP connection is required. RTSP messages are sent from client to server, although some exceptions exist where the server will send to the client. Presented here are the basic RTSP requests. Some typical HTTP requests, like the OPTIONS request, are also available. The default transport layer port number is 554.

- OPTIONS: this request returns the request types the server will accept.
- DESCRIBE: this request includes an RTSP URL (rtsp://...), and the type of reply data that can be handled. Among other things, the presentation description lists the media streams controlled with the aggregate URL. In the typical case, there is one media stream each for audio and video.
- SETUP: this request specifies how a single media stream must be transported. This must be done before a PLAY request is sent. The request contains the media stream URL and a transport specifier. This specifier typically includes a local port for receiving RTP data (audio or video), and another for RTCP data (meta information). The server reply usually confirms the chosen parameters, and fills in the missing parts, such as the server's chosen ports. Each media stream must be configured using SETUP before an aggregate play request may be sent.
- PLAY: this request will cause one or all media streams to be played. Play requests can be stacked by sending multiple PLAY requests. The URL may be the aggregate URL (to play all media streams), or a single media stream URL (to play only that stream). A range can be specified. If no range is specified, the stream is played from the beginning and plays to the end, or, if the stream is paused, it is resumed at the point it was paused.
- PAUSE: this request temporarily halts one or all media streams, so it can later be resumed with a PLAY request. The request contains an aggregate or media stream URL. When to pause can be specified with a range parameter. The range parameter can be left out to pause immediately.
- RECORD: this request can be used to send a stream to the server for storage.
- TEARDOWN: this request is used to terminate the session. It stops all media streams and frees all session related data on the server.

CGI and .CGI scripts.

The Common Gateway Interface (CGI) is a standard that defines how webserver software can delegate the generation of webpages to a text-based application. Such applications are known as CGI scripts; they can be written in any programming language, although scripting languages are often used.

he task of a webserver is to respond to requests for webpages issued by clients (usually web browsers) by analyzing the content of the request (including the URL, the HTTP 'method', request headers and any message body), determining appropriate actions to take and creating an appropriate document to send in response, then returning that to the client.

If the request is just to GET a file that exists on disk, the server can just return the file's contents. Alternatively, the document's content may need to be composed on the fly, and other actions such as updating a database, may be required. One way of achieving this is for a console application to handle the request and compute the returned document's contents, and tell the web server to use that console application. CGI specifies which information is communicated between the webserver and such a console application, and how.

The webserver software will invoke the console application as a command. CGI defines how information about the request (such as the URL, request headers etc) is passed to the command in the form of arguments and environment variables. The application then writes the output document to standard output. CGI also defines how the application should pass back extra information about the output (such as the MIME type and other response headers).

A CGI script is a program that is stored on the remote web server and executed on the web server in response to a request from a user. A CGI script file is written in a programming language which can be either:

- Compiled to run on the server.
- Interpreted by an interpreter on the server.

Examples of languages used to write CGI scripts include C, C++, Ada (Compiled languages) and perl, JCL (Interpreted languages). The CGI script is executed when an anchor tag <A ... > or an image tag <IMG ...> refers to the CGI script file rather than a normal file. The determination of whether this is a CGI script file or just an HTML file is made on the physical placement of the file on the server. Remember, the script file is placed on the same machine on which the web server runs and not on your local machine. Usually this placement is in the remote web servers cgi-bin directory. However the exact location of this directory on the server machine is determined by the web administrator for that machine. This placement and control of the cgi-bin directory is determined by the web administrator to prevent security problems, that could occur if arbitrary programs where allowed to be executed by anybody accessing the machine.

Rovio's APIs specification shows type format of .cgi scripts used for enabling its direct control and management (See Appendix A for complete reference).

Mobile Robots Programming Toolkit.

Mobile Robot Programming Toolkit (MRPT) provides C++ developers an extensive, portable and well-tested set of libraries and applications which cover the most common data structures and algorithms employed in a number of mobile robotics research areas: localization, Simultaneous Localization and Mapping (SLAM), computer vision and motion planning (obstacle avoidance).

Key points in the design of MRPT are efficiency and reusability of code. The libraries include classes for easily managing 3D(6D) geometry, probability density functions (pdfs) over many predefined variables (points and

poses, landmarks, maps), Bayesian inference (Kalman filters, particle filters), image processing, path planning and obstacle avoidance, 3D visualization of all kind of maps (points, occupancy grids, landmarks,...), etc.

he MRPT consists of a set of C++ libraries and a number of ready-to-use applications. Figure 31 shows a dependence graph of the currently existing libraries in MRPT.

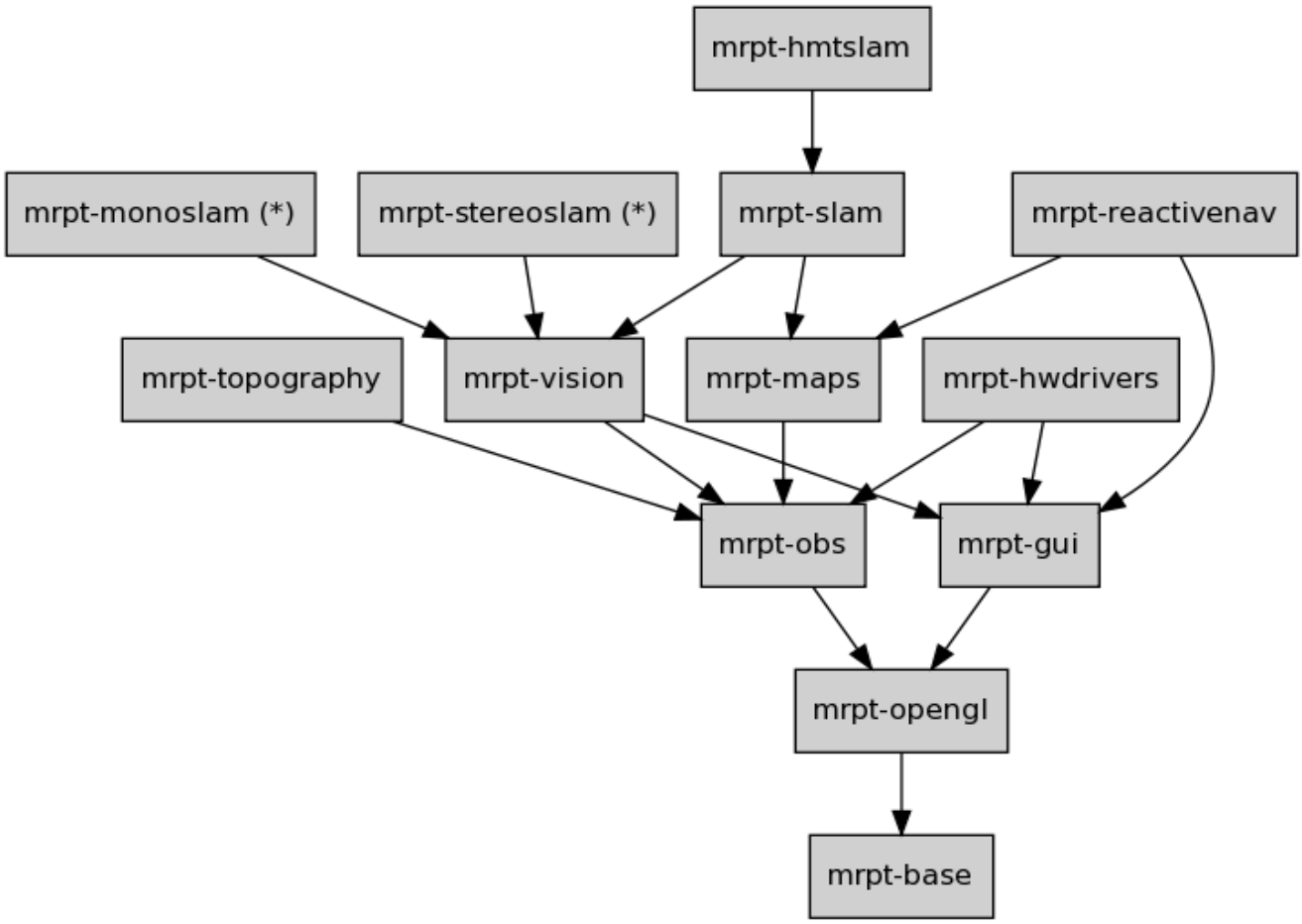


Figure 2.23. The MRPT libraries dependence graph.

Such those libraries suite perfectly for controlling and connecting Aenima to a Rovio robot. MRPT provides an HTTP connection class (mrpt::utils::net_utils) with useful methods for implement an HTTP request. Here a snipped concerning net_utils header:

```
#ifndef  MRPT_NET_UTILS_H
#define  MRPT_NET_UTILS_H

#include <mrpt/utils/CClientTCPSocket.h>
#include <mrpt/utils/CServerTCPSocket.h>

namespace mrpt
{     namespace utils
      {     /** A set of useful routines for networking.   */
            namespace net
            {     using std::string;
                  /** Possible returns from a HTTP request.  */
                  enum ERRORCODE_HTTP     {     erOk = 0,
                                                erBadURL,
                                                erCouldntConnect,
```

```
                                        erNotFound,
                                        erOtherHTTPError
                                   };
              /** Perform an HTTP GET operation (version for retrieving the data
              as a vector_byte)
              ERRORCODE_HTTP BASE_IMPEXP
              http_get(
                    const string      &url,
                    vector_byte       &out_content,
                    string                  &out_errormsg,
                    int                     port = 80,
                    const string      &auth_user = string(),
                    const string      &auth_pass = string(),
                    int                     *out_http_responsecode = NULL,
                    mrpt::utils::TParameters<string>  *extra_headers = NULL,
                    mrpt::utils::TParameters<string>  *out_headers = NULL,
                    int   timeout_ms = 1000
                    );

              /** Perform an HTTP GET operation (version for retrieving the data
              as text)
              ERRORCODE_HTTP  BASE_IMPEXP
              http_get(
                    const string      &url,
                    string                  &out_content,
                    string                  &out_errormsg,
                    int                     port = 80,
                    const string      &auth_user = string(),
                    const string      &auth_pass = string(),
                    int                     *out_http_responsecode = NULL,
                    mrpt::utils::TParameters<string>  *extra_headers = NULL,
                    mrpt::utils::TParameters<string>  *out_headers = NULL,
                    int   timeout_ms = 1000
                    );

              /** Resolve a server address by its name, returning its IP address
              as a string - This method has a timeout for the maximum time to
              wait for the DNS server.
              bool DNS_resolve_async(
                    const std::string &server_name,
                    std::string  &out_ip,
                    const unsigned int timeout_ms = 3000
                    );
        } // End of namespace
    } // End of namespace
} // end of namespace
#endif
```

As you can see from the #include declarations, HTTP connection is performed using TCP/IP Sockets. Using such this class it is easy to perform suitable HTTP requests to Rovio including in the URL invocation of .cgi scripts for robot control:

```
string response, errormsg;
string command =  format("http://%s/rev.cgi?Cmd=nav&action=1",ipAddress.c_str());
http_get(command, response, errormsg, portNumber, userName, password);
```

where, in the above example, rev.cgi?Cmd=nav&action=value is the correct syntax for generate a report from libNS module that provides Rovio’s current status.

RovioCommander class

RovioCommander is the class implemented for connecting and controlling the Rovio Robot. It uses MRPT library and Rovio's API for creating an HTTP connection between Rovio robot and Aenima, and perform HTTP request in the suitable format explained in the previous chapters. RovioCommander supports simple drive commands, user credential management, video streaming and Rovio simple localizations.

RovioCommander constructor asks for including Rovio user credential. Such that information are stored in suitable private variables, and they are used for authenticate user accessing Rovio robot. Credential values are stored offline and locally, so no consistence control is performed when creating a new RovioCommander object (which means that is perfectly legal create a new RovioCommander object with not in-Rovio registered credentials). Credentials manipulation can be performed in two ways:

- locally, with a set of methods that manage locally the credentials value (no control of value's integrity is performed with Rovio's credentials stored inline).
- globally, with a set of methods that direct cooperate with Rovio robot and manage credentials stored in Rovio memory (this approach is available only after Rovio connection initialization, see paragraph below)

Here the two sets of methods declaration.

```
//Local user Credentials and HTTP connections basic settings modifiers
void setUserName(string aUserName);
void setPassword(string aPassword);
void setIpAddress(string aIPAddress);
void setPortNumber(int aPortNumber);

//Local user Credentials and HTTP connections basic settings acceders
string getUserName(void);
string getPassword(void);
string getIpAddress(void);
int getPortNumber(void);

//Methods for user mangement
void userAddNew(const string &username, const string &password);
void userChangePassword(const string &username, const string &password);
void userChangeName(const string &oldUsername, const string &password,
      const string &newUsername,);
void userGetList(const string &userList);
void userDelete(const string &username);
void userGetCurrent(const string &username);
```

In order to establish a suitable connection between system and robotic device, a wakeRovioUp(void) method is provided. It simply controls that connection with Rovio robot is established and check for errors:

```
int RovioCommander::wakeRovioUp(void)
{
      string response, errormsg;
      string command =
      format("http://%s/rev.cgi?Cmd=nav&action=1",ipAddress.c_str());
      http_get(command, response, errormsg, portNumber, userName, password);
      if (!response.empty())
      {
            cout<<"[RovioManager::wakeRovioUp] Response:\n"<<response<<endl;
            connectionEstabilished = true;
      }
      if (!errormsg.empty())
```

```
	{
		cout<<"Error initializing Rovio: \n"<<errormsg<<endl;
		return 1;
	}
	return 0;
}
```

Basic robot movements are provided by a suitable set of methods that traces Rovio APIs specification provided by WowWee company. Each basic movement is selected by inserting an opportune value within .cgi command. In the snipped below, we can see a prototype of all movements methods implemented. aMovement integer variable represent the numeric value that distinguishs different kind of movement. For each movement is also defined a speed value, that represent how fast the movement is done. The value "18" in action parameter within .cgi command identify we're performing a basic movement command. The value we'll insert in drive parameter will set the precise kind of movement.

```
string response, errormsg;
if(connectionEstabilished)
{
	string command =
	format("http://%s/rev.cgi?Cmd=nav&action=18&drive=%i&speed=%i",
	ipAddress.c_str(), aMovement, aSpeed);
	http_get(command, response, errormsg, portNumber, userName, password);
	if(errormsg.empty())
	{
		cout<<"[RovioManager::manualDrive("<<aMovement<<")]
		Response:\n"<<response<<endl;
		return 0;
	}
	else
	{
		cout<<"Error moving Rovio: \n"<<errormsg<<endl;
		return 1;
	}
}
else
{
	cout<<"Rovio is not initialized yet."<<endl;
	return 1;
}
```

Rovio provides useful methods for creating, storing and modifying paths. All paths information are stored online, in Rovio flash memory, so they are usable by any user connecting to it. This allows NEVRAROS system to ignore path management, in sense no information about movements to perform for executed a certain path are controlled by it. NEVRAROS only store a reference name for each path available on Rovio, and perform suitable request with the selected name once a target is reached and the robot need to be moved. In this way, it will be theorically possible to forget anything about basic movement and manual drive: using Rovio web application all the paths could be created and stored in Rovio flash memory. Once this is done, name references would be inserted within ILocation class, and NEVRAROS would just call for selected path once a target is reached in virtual environment. In practice such this approach is not yet implemented. For first experiments, Rovio moved within a simple ring with only four targets, so direct requesting basic 4-dimensional movement is far simpler.

Video streaming from Rovio webcam is controlled within RovioCommander class too. Basicly request to Rovio for a video streaming from its webcam does not differ from requesting any other action: request is invokated through a .cgi command; only difference is that a RTSP request is sent, instead of an HTTP request. So, the .cgi script for it has the form of

Rtsp://xxx.xxx.xxx.xxx/webcam

Where xxx.xxx.xxx.xxx of course represent the IP adress of the Rovio robot. As Rovio APIs declares, the command return a video streaming that it could be possibly seen within a Web Browser. For playing the video stream within NEVRAROS System a little more effort must be used. First of all, both Irrlicht nor BCI++ offer an immediate solution for directly screen the video streaming. Grabbing frame per frame and display them as images at same imagerate than framerate is the only available solution, at the moment. Hence, the video streaming from RTSP must be correctly grabbed and controlled by an opportune video controller that convert the video stream in a temporary format suitable for extracting single frames. For this scope, FFmpeg libraries were used. FFmpeg is a free software / open source project that produces libraries and programs for handling multimedia data and publishes them under the GNU Lesser General Public License or GNU General Public License (depending on which options are enabled). The most notable parts of FFmpeg are libavcodec, an audio/video codec library used by several other projects, libavformat, an audio/video container mux and demux library, and the FFmpeg command line program for transcoding multimedia files. An opportune wrapper for C++ language ensure complete usability of FFmpeg library within the project.

Because of video streaming follows its own life cycle, independently from Nevraros main life cycle, a dedicated thread is created. It control, acquire and grabs frames from video streaming once the method retrieve_video() is invoked. It continues its operations until it is stopped manually by stop_video(). During thread life cycle, frames can be extracted calling getNextImageSync(CObservationImagePtr& lastImage), a method that obtains images in synchronous way (ensuring real time feedback from Rovio camera).

```
bool RovioCommander::retrieve_video()
{
	if(m_videoThread.isClear())
	{
		m_videothread_initialized_done  = false;
		m_videothread_initialized_error = false;
		m_videothread_must_exit         = false;
		m_videothread_finished          = false;

		m_videoThread = createThreadFromObjectMethod
				(this,&mrpt::hwdrivers::CRovio::thread_video);
		while (!m_videothread_initialized_done) {
			mrpt::system::sleep(10);
		}
		if (m_videothread_initialized_error)
		{
			m_videoThread.clear();
			return false;
		}
		else return true;
	}
	else return true;
}

bool RovioCommander::stop_video()
{
	bool was_already_stop = true;
	m_videothread_must_exit = true;
	if (isVideoStreamming())
	{
		joinThread(m_videoThread);
		was_already_stop = false;
	}
	m_videoThread.clear();
```

```
	return !was_already_stop;
}

bool RovioCommander::getNextImageSync(CObservationImagePtr& lastImage )
{
	if (!isVideoStreamming())
		return false;
	{
		mrpt::synch::CCriticalSectionLocker cs( &buffer_img_cs );
		if(!buffer_img)
			return false;
		lastImage = buffer_img;
	}
	return true;
 }
```

Once getNextImageSync(CObservationImagePtr& lastImage) is called, CObservationImagePtr lastImage variable contains a frame grabbed from video streaming. CObservationImage is a MRPT provided container that encapsules an image from a camera, whose relative pose to robot is also stored. Within it the image is referenced by a ICamera object, whose APIs provided a method for saving it as a JPEG file in local hard drive. At this point is easy to perform a Nevraros::ChangeNodeTexture(const c8* nodeName, const c8* aTexture) for loading the image as texture of the SceneNode that represent the video feedback in the virtual environment (of course this method must be invoked framerate times per second).

Another way to obtain a pseudo video frame is by using asynchronous method captureImageAsync(CImage & picture, bool rectified) which stores an image grabbed directly from Rovio webcam in a CImage object. this method must be invoked framerate times per second just like the latter.

```
bool RovioCommander::captureImageAsync( CImage & picture, bool rectified)
{
	try
	{
		vector_byte resp;
		string errormsg;
		string MF=format("http://%s/Jpeg/CamImg[0000].jpg",options.IP.c_str());
		http_get (MF, resp, errormsg, 80, options.user, options.password);

		CMemoryStream stream( &resp[0],  resp.size()  );
		picture.loadFromStreamAsJPEG(stream);
		if( rectified )
		picture.rectifyImageInPlace(options.cameraParams.intrinsicParams,
		options.cameraParams.getDistortionParamsAsVector() );
		picture.saveToFile("0000.jpg");
		return true;
	}
	catch(std::exception &e)
	{
		cerr << e.what() << endl;
		return false;
	}
 }
```

As you can see, this method do not uses RTPS connection, but obtain a pick from Rovio webcam performing an HTTP request. Even if the two methods both provide what is needed, the use of the video stream, even if a little more complex is more desirable.

User Protocol - Online Path Management

Rovio features provide interesting way for navigation. Using path management options, Rovio is able to detect is position in real environment and navigate into it following precise paths. Basically it uses an Infrared signaling beacon for self localization. The TrueTrack Room Beacon provided with Rovio can provide coverage for one room or open area that is approximately 20-25 feet (6-7.5 meters) in diameter. With the TrueTrack™ Navigation System it is possible to store waypoints - one click will automatically navigate Rovio to the preprogrammed point - and TrueTrack Room Beacons can expand the range of these waypoints. Rovio and TrueTrack™ Navigation System use a NorthStar detector for checking their exact position. The NorthStar detector uses triangulation to measure position and heading in relation to IR light spots that can be projected onto the ceiling (or other visible surface). Because each IR light spot has a unique signature, the detector can instantly and unambiguously localize. Because the NorthStar detector directly measures position and heading, a localization result is intrinsically robust. A NorthStar-enabled product does not require prior training or mapping to measure its position.

Even if RovioCommander class provides methods for creating and managing paths, it is far simpler to use the Web-based WowWee application for creating new paths, and only requesting of walking trough already created paths can be demanded to Nevraros system directly. All the paths created are defined in relation with Home TrueTrack Beacon position, so if dock is moved, or setted for the first time, a Save Home operation must be performed.

Once a paths set is created, all paths name must be saved and migrated into Nevraros system. Each path correspond to a navigation action the robot will performs any time the correspondent target is reached. Using the methods provide by RovioCommander, is easy to implement a basic path attuator:

```
void pathStartRecording(void);
void pathDiscardRecording(void);
void pathSaveRecording(const string &aPathName);
void pathDelete(const string &pathName);
void pathRename(const string &oldPathName, const string &newPathName);
void pathGetList(const string &pathList);
void pathDeleteList(void);
void pathPlayForward(const string &pathName);
void pathPlayBackward(const string &pathName);
void pathStop(void);
void pathPause(void);
void pathHome(void);
void pathDock(void);
void pathMarkHome(void);
```

Let consider the easiest example, where in a certain environment a Rovio charging Dock is provided. In the room four targets are placed, plus a reference to a starting position. The robot is allowed to navigate from initial position to one of the four targets, and then it comes back to initial position autonomously. Anytime it need to recharge its battery, or when experiments are finished, it cames back to charging Dock position and docks (Figure 2.24).

Using Rovio Web-application the five paths are created and stored with name PathX, where X is the number of the path. Also a Mark to Home position is done. The Robot starts its execution from charging Dock position.

```
//Initializing Rovio
RovioCommander myRovio;
myRovio.wakeRovioUp();
myRovio.pathMarkHome();
myRovio.pathPlayForward(Path0);
...
switch(targetReached)
```

```
{
	case 1:
		//When target 01 is reached with stimulation paradigm
		myRovio. pathPlayForward(Path1);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
		wait(5);
		myRovio. pathPlayBackward(Path1);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
	break;
	case 2:
		//When target 02 is reached with stimulation paradigm
		myRovio. pathPlayForward(Path2);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
		wait(5);
		myRovio. pathPlayBackward(Path2);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
	break;
	case 3:
		//When target 03 is reached with stimulation paradigm
		myRovio. pathPlayForward(Path3);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
		wait(5);
		myRovio. pathPlayBackward(Path3);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
	break;
	case 4:
		//When target 04 is reached with stimulation paradigm
		myRovio. pathPlayForward(Path4);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
		wait(5);
		myRovio. pathPlayBackward(Path1);
		while(myRovio.botStatus()=="executing path")
		{
			wait(1);
		}
	break;
}
...
//Way back home and dock
myRovio.pathPlayBackward(Path0);
```

More about paths in real experiments will be said in next chapter. One important thing to remember is that every path, measuration and similar is performed by Rovio referring to current Dock position. So, if charging dock is moved, targets must be moved too (or equivalenty, dock must be restored to its initial position). That happens because TrueTrack Beacon use signaling feedback from IR source to near walls/floor (if a GPS localization would be available, there will be no problems).

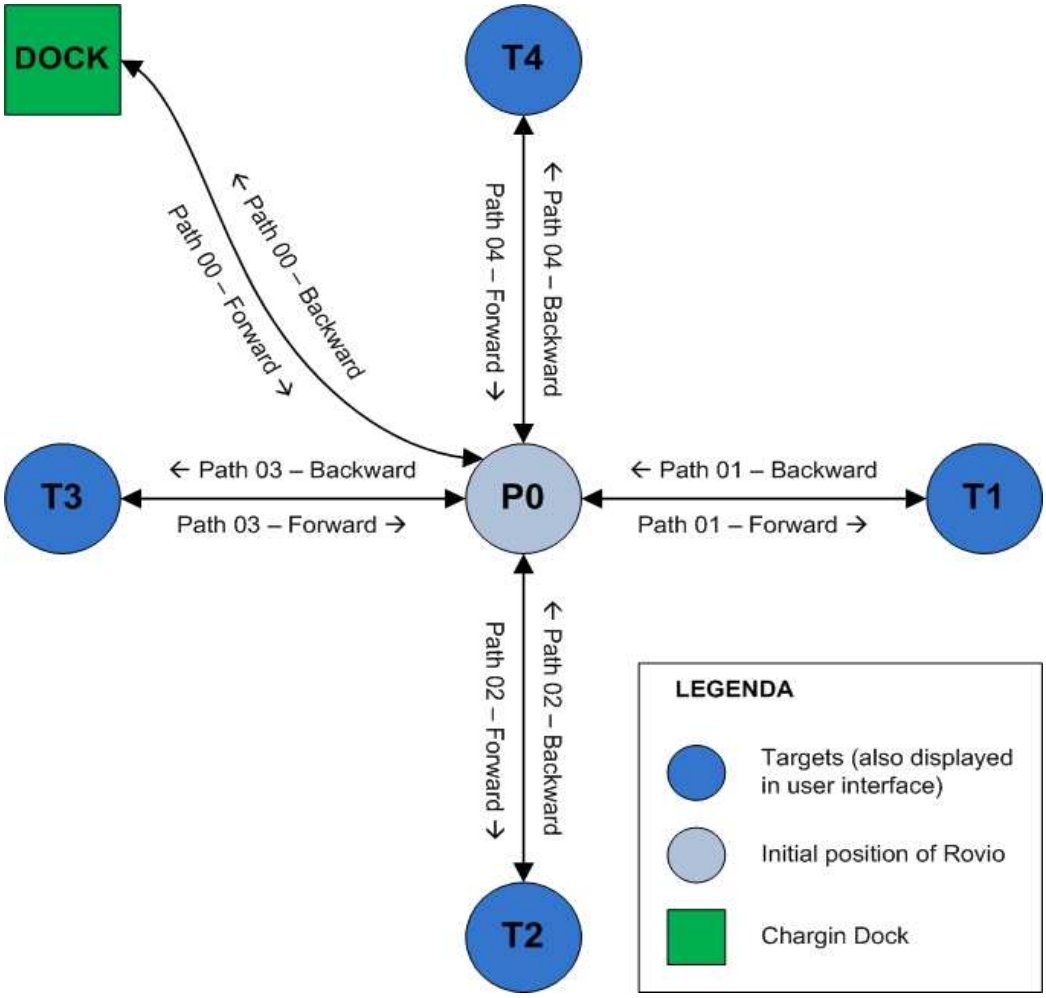


Figure 2.24. (a) A simple real environment schematization for Rovio path management. (b) Path Management form on Rovio Web-based application. Up to 256 paths can be created and stored in Rovio Flash memory. (c) Rovio and its docking station.

User Protocol - Nevraros Main Class

Nevraros class combines all the features presented in previous chapter for providing VEP stimulations, sending and receiving information from/to the robotic device and managing and controlling classification information from/to HIM and other modules. It is a derived class from ProtocolMngr class, and it contains all the data memebers of ProtocolMngr class.

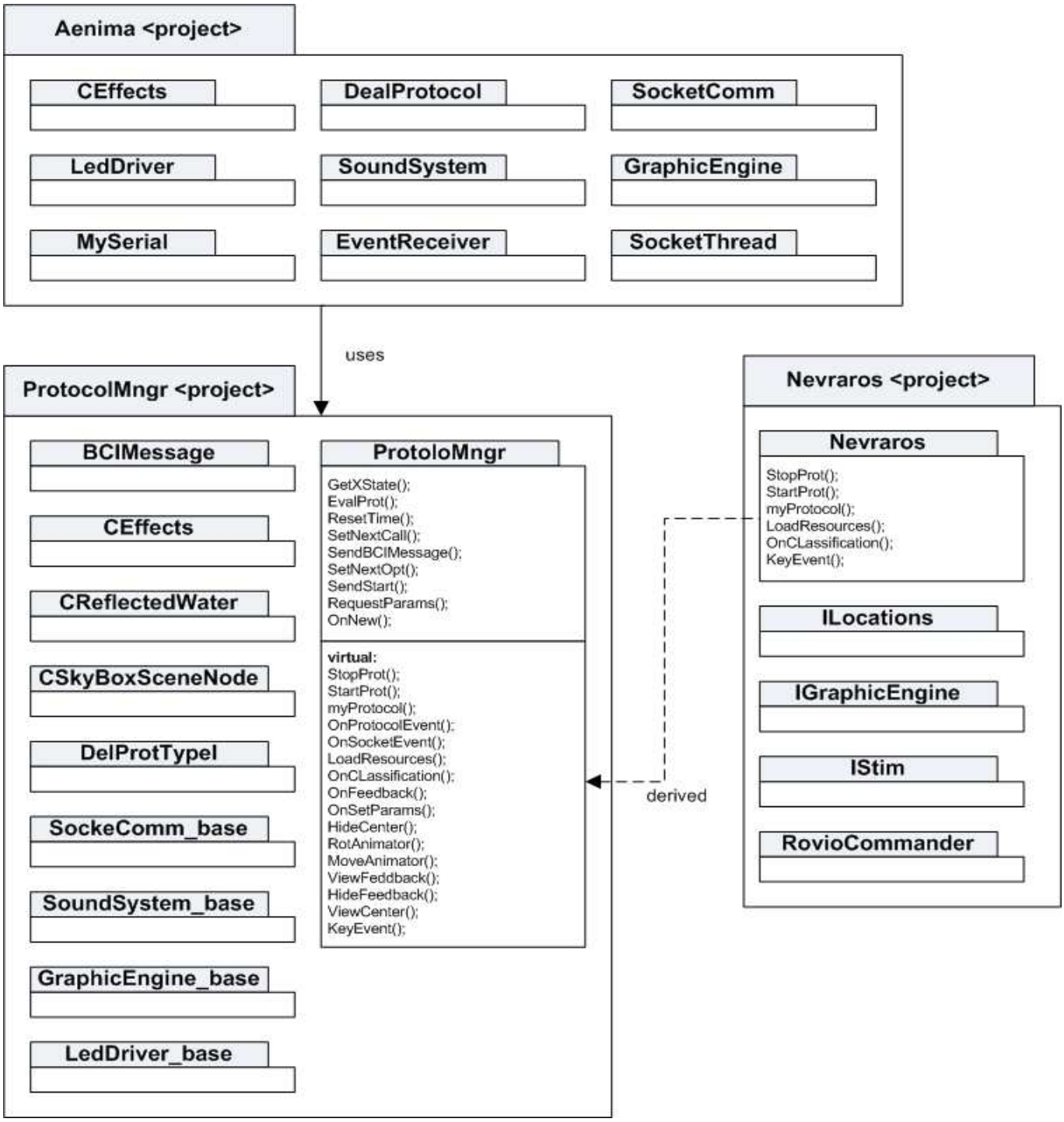


Figure 2.25. Dependences between Aenima project, ProtocolMngr project and Nevraros project. Nevraros main class is derived from ProtocolMngr main class. When Aenima starts its execution, it select which derived class (among all the protocols implemented) to use with ProtocolMngr.

ProtocolMngr class contains methods for more than P300-based P300 stimulation, and so not all provided methods are useful. Nevraros needs only six base methods:

```
class Nevraros: public ProtocolMngr
{
public:
      Nevraros(GraphicEngine_base *Engine, SoundSystem_base *Sound, SocketComm_base*
      pSocket, LedDriver_base *myLed, bool startup, DealProtType *pParent);
      ~Nevraros(void);

      bool StartProt(bool SendSocketMsg);
      bool StopProt(bool SendSocketMsg);
      void myProtocol(int myTime);
      void LoadResources(void);
      void OnClassification(BCIMessage *SocketMsg);
      void KeyEvent(const SEvent &event);
 ...
 }
```

StartProt() and StopProt() are used for (trivial) starting and stopping current protocol. They invoke same methods from PrtocolMngr class and send a message via Socket to Aenima for starting session.

```
bool Nevraros::StartProt(bool SendSocketMsg)
{
      ...
      return ProtocolMngr::StartProt(SendSocketMsg);
}

bool Nevraros::StopProt(bool SendSocketMsg)
{
      ...
      return ProtocolMngr::StopProt(SendSocketMsg);
}
```

LoadResources() is the method that provides creation of graphic elements within 3D space offered by base graphic constructor in Aenima. Here all graphic objects are listed and initialized. When Nevraros class constructor is invoked, OnNew() method from ProtocolMngr is called. Within it LoadResources is then invoked.

```
Nevraros::Nevraros(GraphicEngine_base *Engine, SoundSystem_base *Sound,
SocketComm_base* pSocket, LedDriver_base *myLed, bool startup, DealProtType *pParent)
                  :ProtocolMngr(Engine, Sound, pSocket, myLed, startup, pParent)
{
      OnNew();
      AEngine->device->setWindowCaption(L"AEnima - Nevraros");
      srand((unsigned)time(0));
      ShowNode(INTROSCREEN);
      ILoadTextures();
}
...
void ProtocolMngr::OnNew(void)
{
      ...
      cout<<"LOADING BUILT-IN RESOURCES..."<<endl;
      AEngine->LoadBuiltInResources();
      cout<<"...done!"<<endl;
      cout<<"LOADING PROTOCOL-SPECIFIC RESOURCES..."<<endl;
      LoadResources();
      cout<<"...done!"<<endl;
      ...
}
```

KeyEvent() contains keyboard management, and here keyboard keys actions and operations can be scheduled.

```
void Nevraros::KeyEvent(const irr::SEvent &event)
{
	if (event.EventType == irr::EET_KEY_INPUT_EVENT && !event.KeyInput.PressedDown)
	{
		switch (event.KeyInput.Key)
		{
			case irr::KEY_KEY_V:
				...
				break;
			case irr::KEY_KEY_I:
				...
				break;
			case irr::KEY_KEY_J:
				...
		}
	}
}
```

KeyEvent() contains keyboard management, and there keyboard keys actions and operations can be scheduled. MyProtocol() and OnClassification() contains code for running stimulation and performing action whenever a Classification from HIM takes place.

Once the protocol is started, a quick configuration process is to be done: System's Supervisor has to select overa small group of settings in order to:

Figure 2.26. Tuning User Protocol up.

- Decide which type of experiment has to be executed (learning, testing, free-user). Protocol mode selection is represented by an icon that blink when a valid selection is performed.
- Decide the center-to-target distance, or else the minimum number of true positives reactions to a target stimulation in order to achieved the target. Distance selection is represented by an icon that blink when a valid selection is performed.
- Decide how many sessions have to be executed (this setting is available only on learning and testing mode). Number of sessions selection is represented by an icon that blink when a valid selection is performed.
- Decide how many stimulation the system provide to user for each session (again, available only on learning and testing mode). Stimulation selection is represented by an icon that blink when a valid selection is performed.
- Decide if the Blinker has to be used. Blinker selection is represented by an icon that blink when a valid selection is performed.
- Decide the (possibly in real environment) location of the experiment, or else the location where the robot actually is. Location selection is represented by an icon that blink when a valid selection is performed.

Once tuning is done, stimulation window will appear. Main execution is represented by a Giga loop that runs continuatively, switching by different cases: each case correspond to a logical action of the interface, such as preparing a stimulation, moving the central ball and so on. Switching from a case to another can be timely

delayed, so precise temporization of different events can take place. Initially SetRandomSequence(targets) is invoked, so an opportune array of random stimulations is created. This 1 dimensional stimuli array consists of integer values stored in random order and repeated randomly. Available values are in the set of: {LEFT = 1, RIGHT = 8, DOWN = 16, UP = 32}. A pointer will scan the overall array from first to last element. When stimuli pointer scan a new element, suitable operation for displaying correspondent stimulus is done.

```
switch (stimulations[stim_pointer])
{
	case UP:
	trig_value=UP;
	if (IsTargetEvent(false,autoclass)) trig_value |= TARGET_EVENT;
	SendBCIMessage(GRAPH_TRIG,trig_value);
	ChangeNodeTexture(UPARROW,"../Media/Nevraros/Interface/arrow_yu.png");
	break;
	case DOWN:
	trig_value=DOWN;
	if (IsTargetEvent(false,autoclass)) trig_value |= TARGET_EVENT;
	SendBCIMessage(GRAPH_TRIG,trig_value);
	ChangeNodeTexture(DOWNARROW,"../Media/Nevraros/Interface/arrow_yd.png");
	break;
	case LEFT:
	trig_value=LEFT;
	if (IsTargetEvent(false,autoclass)) trig_value |= TARGET_EVENT;
	SendBCIMessage(GRAPH_TRIG,trig_value);
	ChangeNodeTexture(LEFTARROW,"../Media/Nevraros/Interface/arrow_yl.png");
	break;
	case RIGHT:
	trig_value=RIGHT;
	if (IsTargetEvent(false,autoclass)) trig_value |= TARGET_EVENT;
	SendBCIMessage(GRAPH_TRIG,trig_value);
	ChangeNodeTexture(RIGHTARROW,"../Media/Nevraros/Interface/arrow_yr.png");
	break;
	default:
	break;
 }
```

Each time a stimulation action happens, a BCIMessage is sent to HIM, with the value of the stimulation (UP, DOWN, LEFT, RIGHT, NONE) and a reference for distinguish between target and non target stimulations.

HIM will compare the information obtained from Interface BCIMessage with information obtained from the classifier, and send back to the Interface another BCIMessage with the action to do. BCIMessages from HIM are managed in the function OnClassification(BCIMessage *SocketMsg).

```
void Nevraros::OnClassification (BCIMessage *SocketMsg)
{
	if (isRunning==false) return;
	if(P300)
	{
		switch (SocketMsg->Value)
		{
			case UP:
				if(stimulations[stim_pointer]==UP)
				{	TraslateNode(CURSOR,0,step,0,500);
					up_movs++;
					movs_used++;}
			break;
			case DOWN:
				if(stimulations[stim_pointer]==DOWN)
				{	TraslateNode(CURSOR,0,-step,0,500);
					down_movs++;
```

```
                        movs_used++;}
                  break;
                  case LEFT:
                        if(stimulations[stim_pointer]==LEFT)
                        {     TraslateNode(CURSOR,-step,0,0,500);
                              left_movs++;
                              movs_used++;}
                  break;
                  case RIGHT:
                        if(stimulations[stim_pointer]==RIGHT)
                        {     TraslateNode(CURSOR,step,0,0,500);
                              right_movs++;
                              movs_used++;}
                  break;
                  case NONE:
                        none_movs++;
                        movs_used++;
                  break;
            }
            P300 = false;
      }
}
```

As you can see, CURSOR SceneNode is moved only when a valid classification has occurred. Every time a movement is occurred, a collision checker control:

- If a border of the ring is met (which means a target is achieved) or
- If there exists more stimulations

When first condition is met, the protocol control which target the cursor has achieved and act in consequences of that (starting robot session or anything else); when the second condition is met, it start with another stimulation epoch. When no conditions are met at all, the protocol warns the user that stimulations are over and ends the session.

Once a target is reached, supposing we're in free-user mode, the robotic device is started and commanded to meet the destination that the target represent. The stimulation module is temperately hidden, and the navigation module pops up. The Rovio robot is initialized as showed in previous paragraph, and a stream of frame appears in an opportune Box.

3 - System Evaluation

This chapter covers the long way of validating and testing implemented system. When concerning with medical software, evaluating it means both valuate is efficiency and efficacy in terms of computer-way-of-view and in terms of performances when used among patients. A software system that is not robust and efficient of course is not a suitable solution for commercial, or what else, use; moreover, a software system that is robust and efficient but reports no good evaluation in patient usability will have no future too. Due to this considerations, we distinguish system evaluation in (a) evaluation of system performances in terms of bugs, crashes, algorithmic problems, implementation problems, efficacy and efficiency of execution and more, and (b) usefulness and efficiency in qualitative results obtained by user experimentation. Concerning point (a) several test and benchmarks for validating and evaluating system software/hardware performances has been done. Concerning point (b) experimental usage among healthy and injured patients took place (or are under implementation).

Validation tests

Validation tests on system performances were maid together with system development. We distinguish validation tests in (a) hardware benchmarks and (b) software performance and usability. For the first, we must note Nevraros system consists in several components, which must cooperate in efficient way. Concerning the latter, we focus our evaluation tests in timing performances of system modules and data transmission.

Hardware benchmarks

First tests took place executing the overall system in a single pc. We tested components efficiency using a 3 years old computer provided by San Camillo Hospital with follow basic features: RAM (512 MB, DDR2), Processor (AMD Athlon 64 single core, 2.4Ghz), Video card (nVidia NForce3, 128MB dedicated memory). We immediately note that efficiency result were far away from desirable. Aenima module required high percentage of video memory for creating and maintaining 3D Irrlicht-based virtual environment, and HIM and SCAN modules required high percentage of processor usage for acquire and classifying raw data from electrodes. Use of at least two pcs is hence required. In second tests round two pcs, with comparable features were used: we demanded data acquisition and classification to pc1, while the user interface and robot control start from pc2. The two pcs

initially transmitted data and information through Hospital-dedicated LAN, but we abandoned that transmission solution in short time. Average transmission time of small data packets across company LAN was around 80 to 120 ms, far too big values for our scopes: we need high speed data transmission in order to synchronize time instants of VEP and acquisition window on raw brain activity data. A dedicated switch from D-Link company solved most of problems. Using a dedicated LAN assured time transmission of 65Kb data packet in less than 2 ms.

Video monitor played a relevant role in transmission delays too. Refresh time of monitors cannot be ignored, because it can leads to apparently unsolvable delays. Every monitor has its own refresh rate (both CRT and LCD). It represent how many times a second the video screen is updated. Since common commercial CRT monitors have 50-75Hz refresh rate and LCD monitors have 100-120Hz refresh rate, in worst case happening (a stimulation take place immediately after a refresh action), we can have 8 to 20 ms of delay. San Camillo Hospital provided a LCD monitor with 100Hz refresh rate, which leads to maximum delay of 8,4 ms.

Software performances

Software performances are mainly measured by temporal parameters. In particular, we are searching for quantitative execution time of Nevraros system under precise conditions. Due to well defined medical needs, a set of time requirements must be validated. Also assuring a fast time-to executing period needed is sign of good quality of software. Using VEP as stimulation approach implies that user attention is always focused on video screen. Long-time concentrating is without doubts a tiring activity, and directly reflects usability of software system.

Basing on medical knowledge concerning P300 waveform and its time constraints, we can see a minimum 2s interval per epoch is needed. Further reduction of this time interval cause an aliasing phenomena, with all the consequences of noise and values alteration. Experimental tests show that incrementing epoch interval over 2s do not produce evident benefits.

Patient attention in using the system is also strictly related to timing constraints. Considering a simple case of stimulation session with 4 steps center-to-target distance and 4 possible targets, and a medium quality classifier with about 75% of average success classification, we need a total of 96 stimuli for assure a target will be reached. That means 96 epochs of stimulation, or else 192 seconds of pure stimulation (that is more than 3 minutes). Normally a learning/testing experiment request 6 to 8 sessions to be performed sequentially, which means 20 to 27 minutes of concentration (18 to 14 of pure concentration and 5 to 10 seconds of pause between each session). Maintaining the brain focused on specific objects for such this time is a tiring activity, and we will see in next chapter that tiredness directly influence quality of brain signals. When system is used in free-user mode, once a session is finished, patients must relax for a few time before starting a new stimulation session, or they would not be able to generate P300 peaks correctly (and also they would not use the system for more than little periods).

On the other side, the system must provide a fast way for communicating and controlling external devices. If a patient needs, let say, 15 minutes of work for commanding the robotic device to a single target, its interest will decrease rapidly due to indifference.

As result to this two problems, we set the follow timing constraints. The system start its execution and is ready to provide stimulation to user in less than 90 seconds (this time period is assured if system supervisor is ready to tune the system up in few seconds. See chapter 2 for "tuning system up" operations"). The system provides a set of VEP consequently for a time of no more than 4:08 minutes (longest case with 124 epochs). In learning/testing mode, the intra-session time is less than 10 seconds. In free-user mode, the intra-session (or else, the time

between choosing a target and choosing a new target) is not less than a quarter of stimulation time. Using standard 96-epochs set of stimulation we have hence a minimum of 46 seconds of intra-session period. If we consider that no more of 4 seconds are required for switching from selection module to navigation module (and same time is required for switching back to selection module from navigation module), we hence have 38 seconds of minimum video feedback length.

Experimentation

Experimentation took places in San Camillo laboratories. There use tests were performed on healthy patients. Testing Nevraros on unhealthy patients is a process not yet achieved, but still on working. We distinguish experimentation in (a) check classifier quality and (b) test the overall system with free-user mode sessions. For the first, several classifier were created (with both algorithmic and bounds differences), and we tested them on two healthy subjects for defining the best by-result classifier. Concerning the latter, last adjustments are in progress for prepare a suitable controlled location where the robot can act and navigate under control of patient.

Classifier Quality

Here the results of testing classifier quality over two healthy patients. For Patient01, we have six training session where classifier were populated. Next 8 testing sessions were performed.

Patient01 results:

BCI-skill	**1 healthy subject (mean±std)**
Classification accuracy (performance %)	74.3±9.5
Transfer bit rate (bit/min)	7.38±5.11
Percentage of sessions successfully completed (%)	75
Training Number of Stimuli (TNS)	175
Performance trend (%/session)	2.57
Weakness index (%)	0.0±0.0
Robustness index (%)	88.8±21.0

FIGURE 3.1. BCI classifier significative parameters for 8 testing sessions. Of that sessions set, 4 were performed using cp300q2 classifier, and 4 using cp300q3.

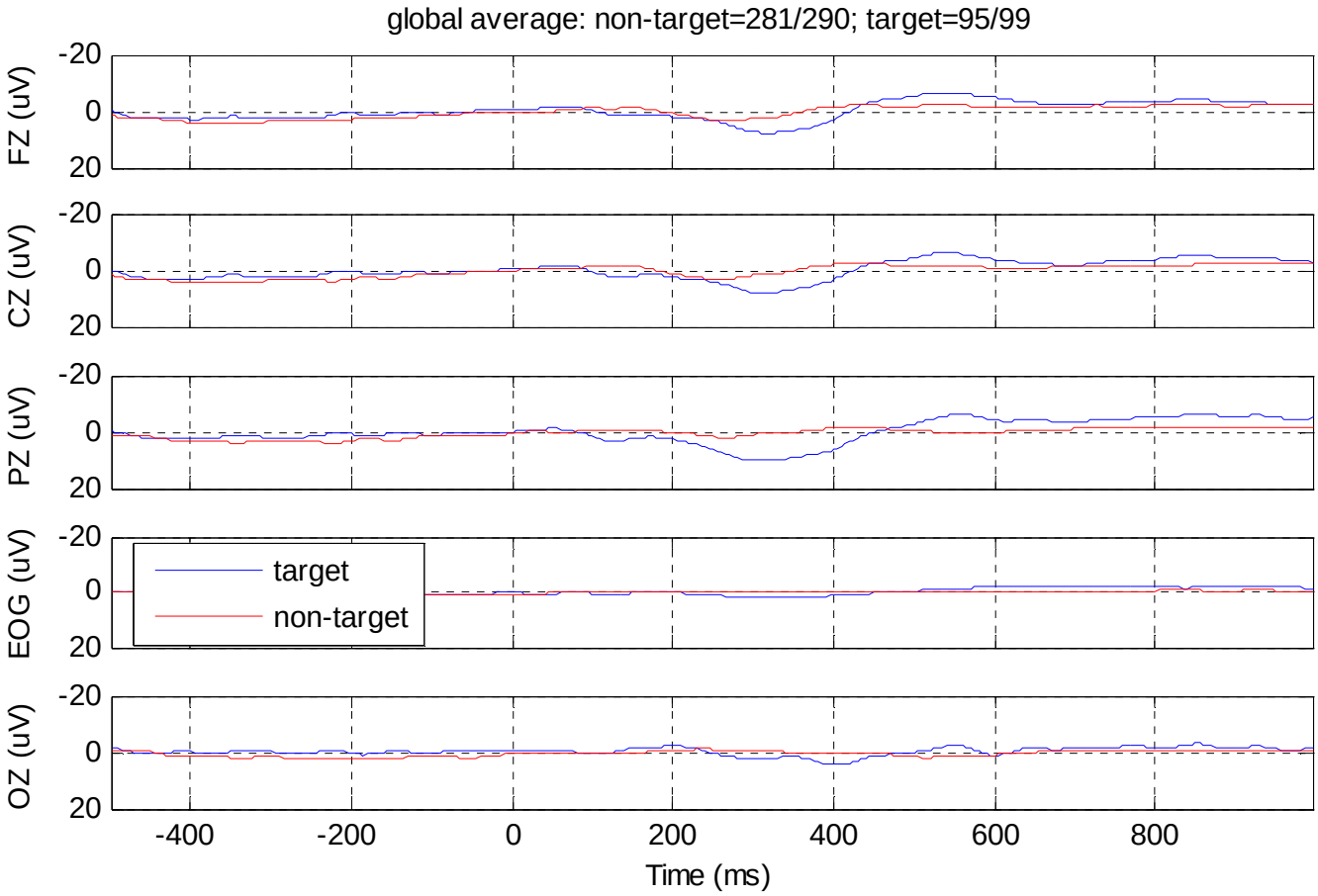
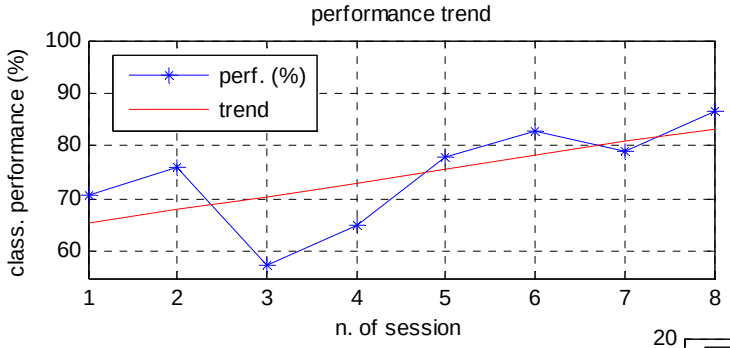


FIGURE 3.2. Traces average representation. Blue lines stands for brain activity with VEP stimulation (target). Frontal, Central and Parietal electrodes show the N3 and P3 peaks.



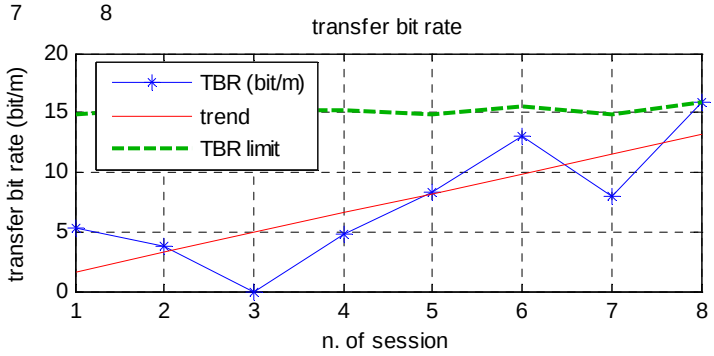


FIGURE 3.3. (a) classification performance trend and (b) transfer bit rate trend. Examining classification performance trend we can see classifier enhances its ability performing more testing sessions. This happens because of classifier uses also training traces for population improvement. As classification performances grows, also bit rate transfer do it too. Comparing the two blue waveforms we can see similar plot.

Results show that second-half sessions set lead to better classification process. In fact, for first 4 testing sessions we used a certain type of classification, while in second 4 training sessions a new type of classifier was introduced. Below a synthetic results report concerning last 4 training sessions solo (with new type of classifier) show the difference between average report.

BCI-skill	**1 healthy subject (mean±std)**
Classification accuracy (performance %)	81.5±4.0
Transfer bit rate (bit/min)	11.28±3.81
Percentage of sessions successfully completed (%)	100
Performance trend (%/session)	2.29
Weakness index (%)	0.0±0.0
Robustness index (%)	100.0±0.0
Chance level probability	s=100% with p=0.0039

FIGURE 3.4. BCI classifier significative parameters for last 4 testing sessions. Better results are obtained if confronting result for average 8 training session report.

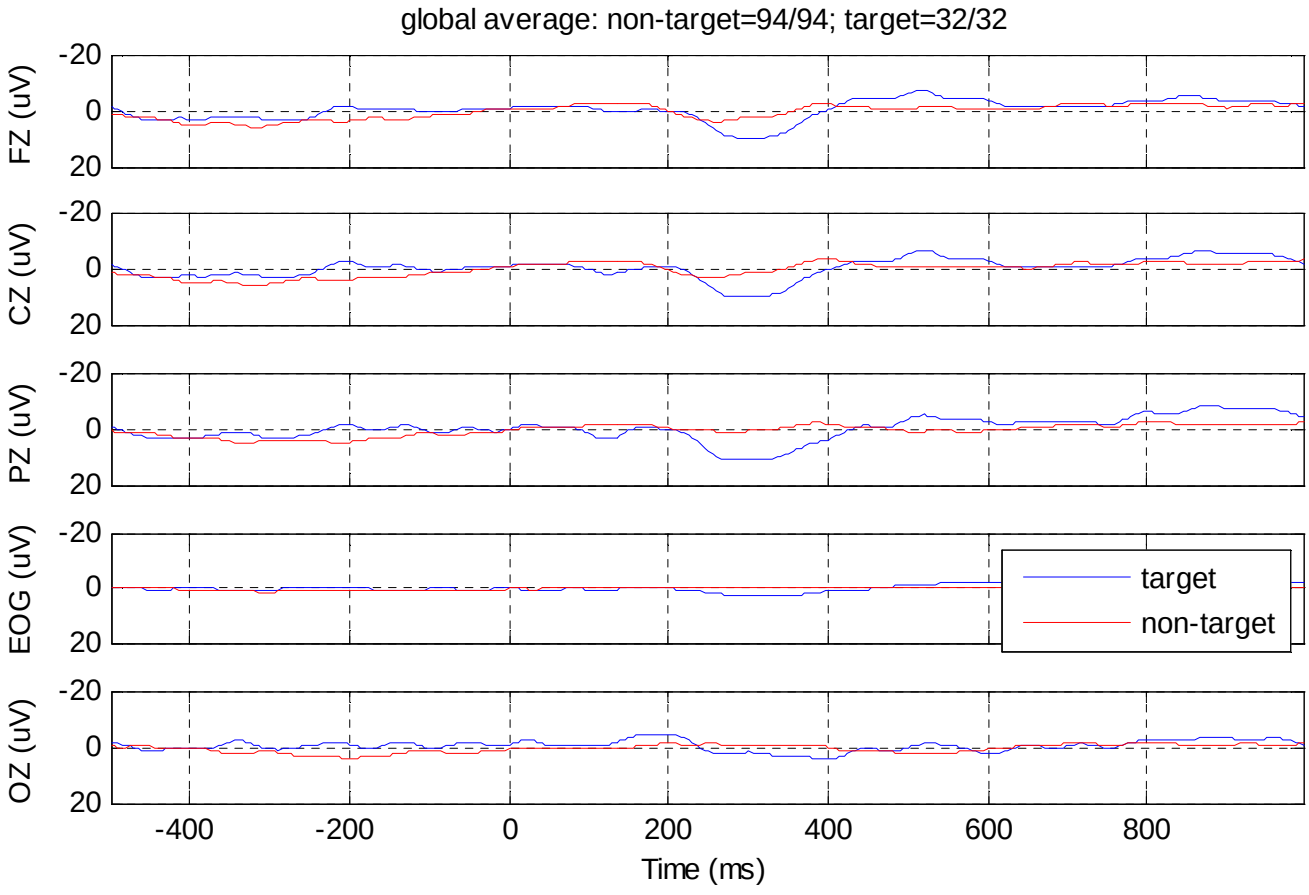


FIGURE 3.5. Traces average representation. Blue lines stands for brain activity with VEP stimulation (target). Frontal, Central and Parietal electrodes show the N3 and P3 peaks.

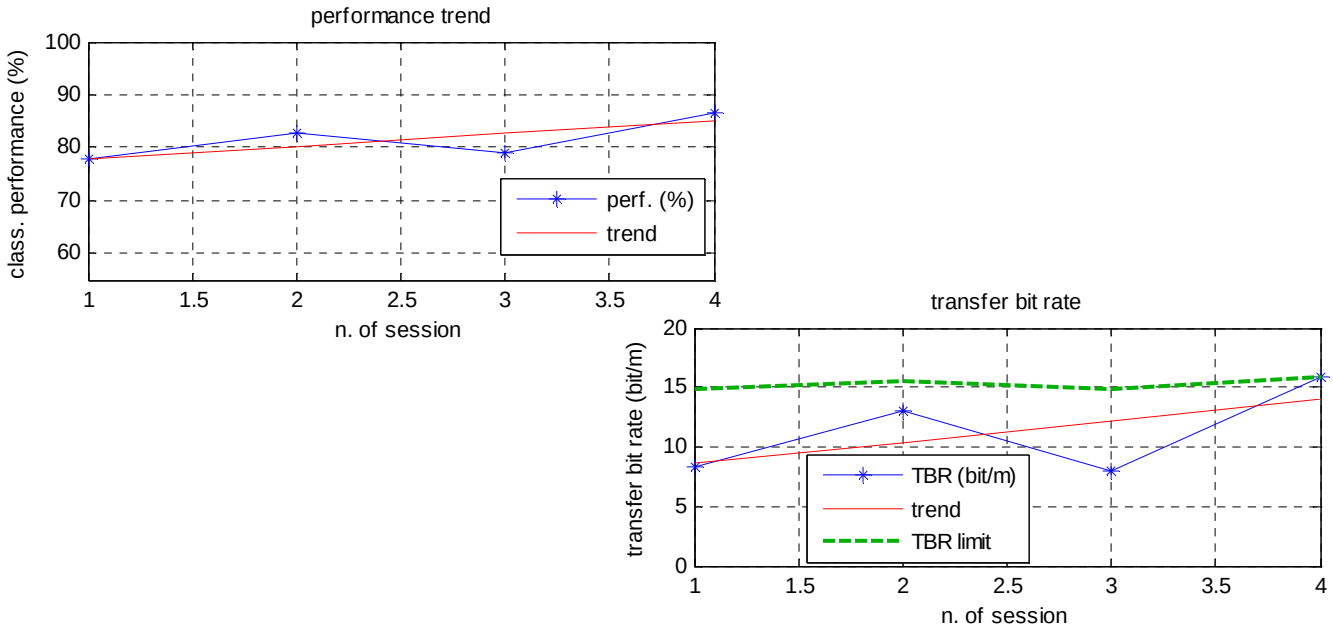


FIGURE 3.6. (a) classification performance trend and (b) transfer bit rate trend. Examining classification performance trend we can see classifier enhances its ability performing more testing sessions. This happens because of classifier uses also training traces for population improvement. As classification performances grows, also bit rate transfer do it too. Comparing the two blue waveforms we can see similar plot.

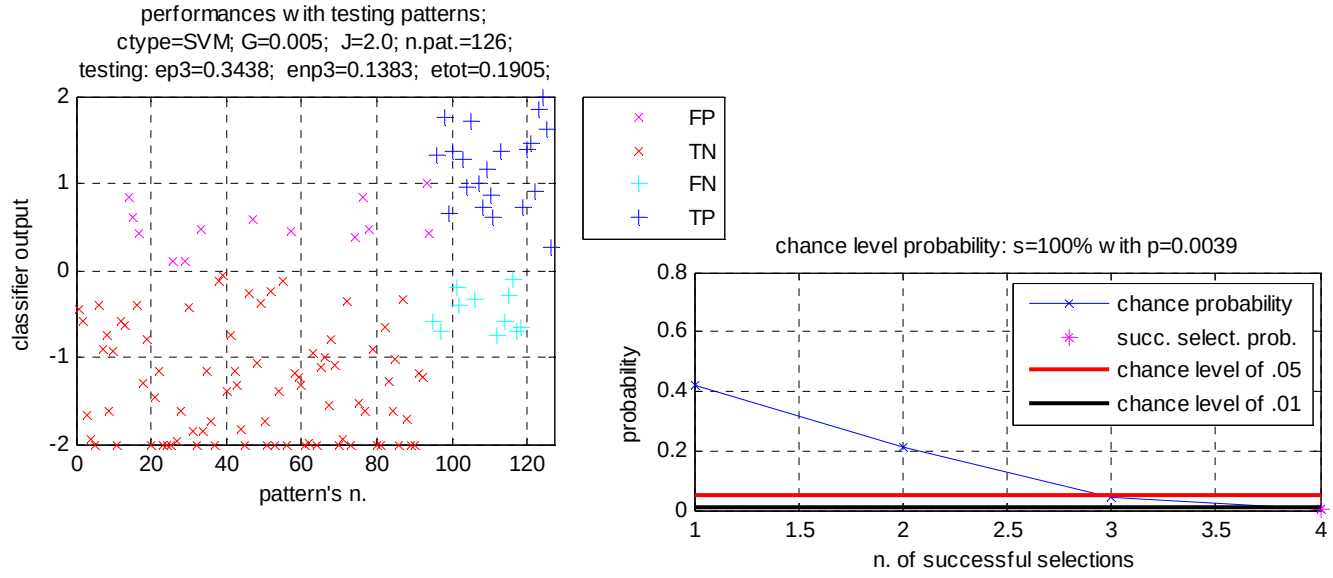


FIGURE 3.7. (a) classification output representation and (b) chance level probability for last 4 training sessions.

Patient02 results:

BCI-skill	1 healthy subject (mean±std)
Classification accuracy (performance %)	78.5±6.6
Transfer bit rate (bit/min)	7.99±4.51
Percentage of sessions successfully completed (%)	87.5
Training Number of Stimuli (TNS)	231
Performance trend (%/session)	-1.58
Weakness index (%)	0.0±0.0
Robustness index (%)	85.0±35.1
Chance level probability	s=87.5% with p=0.0004

FIGURE 3.8. BCI classifier significative parameters for 8 testing sessions. All testing sessions were performed with best classifier.

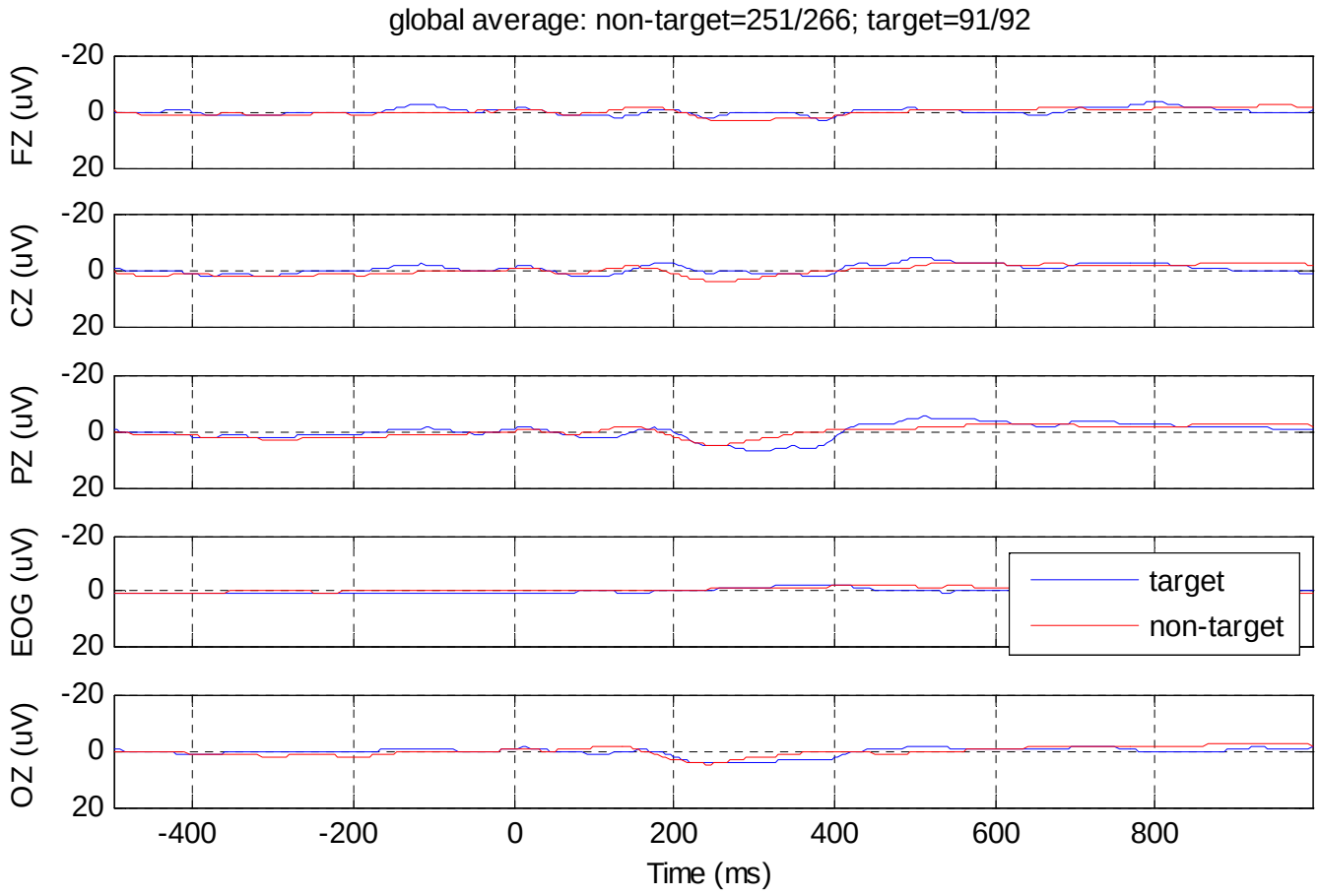


FIGURE 3.9. Traces average representation. Blue lines stands for brain activity with VEP stimulation (target).

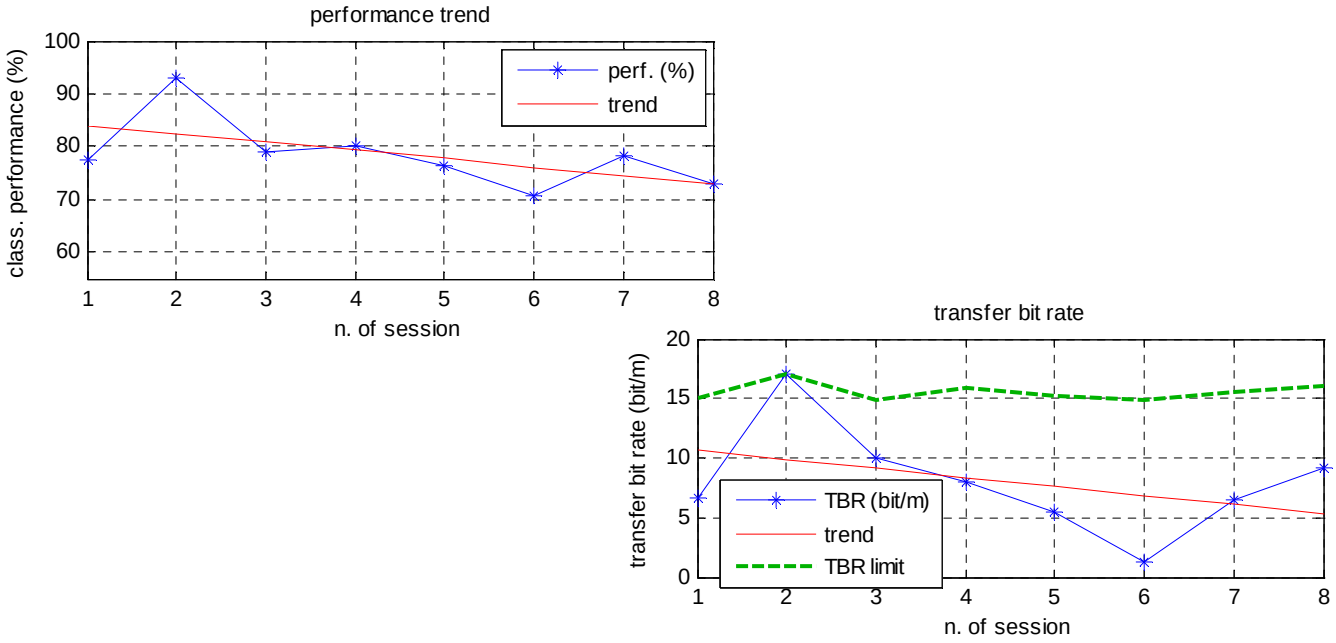


FIGURE 3.10. (a) classification performance trend and (b) transfer bit rate trend. Examining classification performance trend we can see classifier reduces its ability performing more testing sessions. This is probably due to patient fatigue or not ready mental state, or maybe some self-maid artifacts introduced some extra noise enhancing classification difficulty.

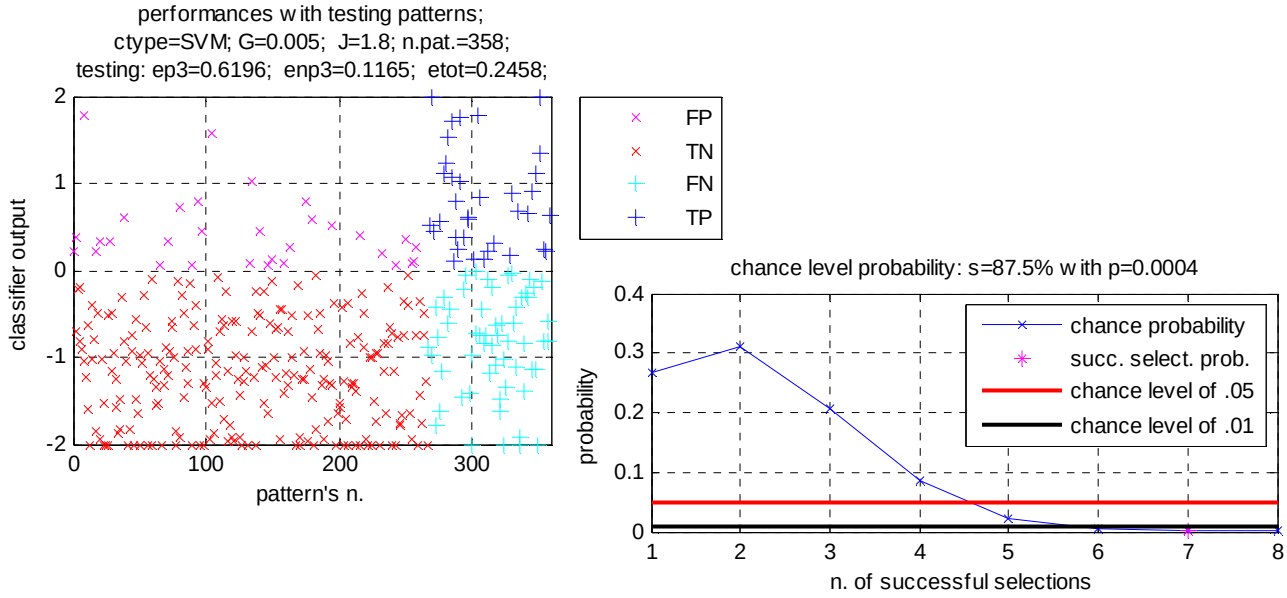


FIGURE 3.11. (a) classification output representation and (b) chance level probability for last 4 training sessions.

System test

An experimental test of overall system is actually under construction. The laboratory of San Camillo dedicated to BCI studies will be the environment where the Rovio robot will navigate. Five targets plus home location will be available. Patient, supervisor and all computer supplies will also be placed in the same room (Rovio moving is noiseless, so there will be no problems). Below an environmental diagram shows where the targets are to be placed.

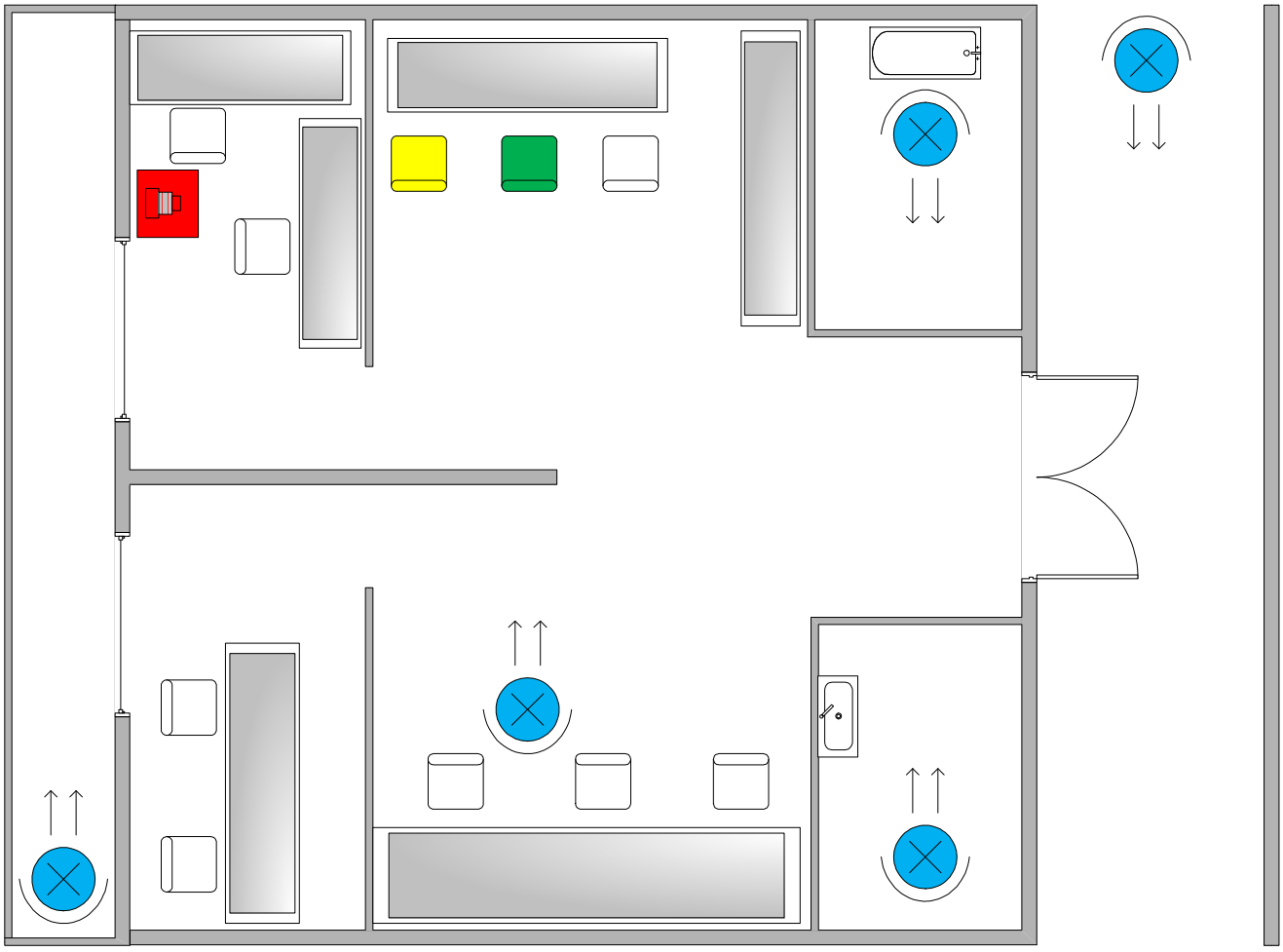


FIGURE 3.12. Environmental diagram of up-to-be overall experiment location. Red object stands for Rovio Charging Dock. Green chair represents patient and Display monitor position within environment. Yellow chair represents place for system supervisor. There also pc with classifier, amplifier and medical equipment will be placed. Blue circles represent available targets. All doors and windows will be open for the overall experiment.

Using Rovio paths management features, 12 different paths will be created and stored in Rovio flash memory. This is the minimum number of path for providing 4 different reachable targets from every target position. Note that thanks to bi-directional paths walking, we are able to use same path both for going from target A to target B and for coming back.

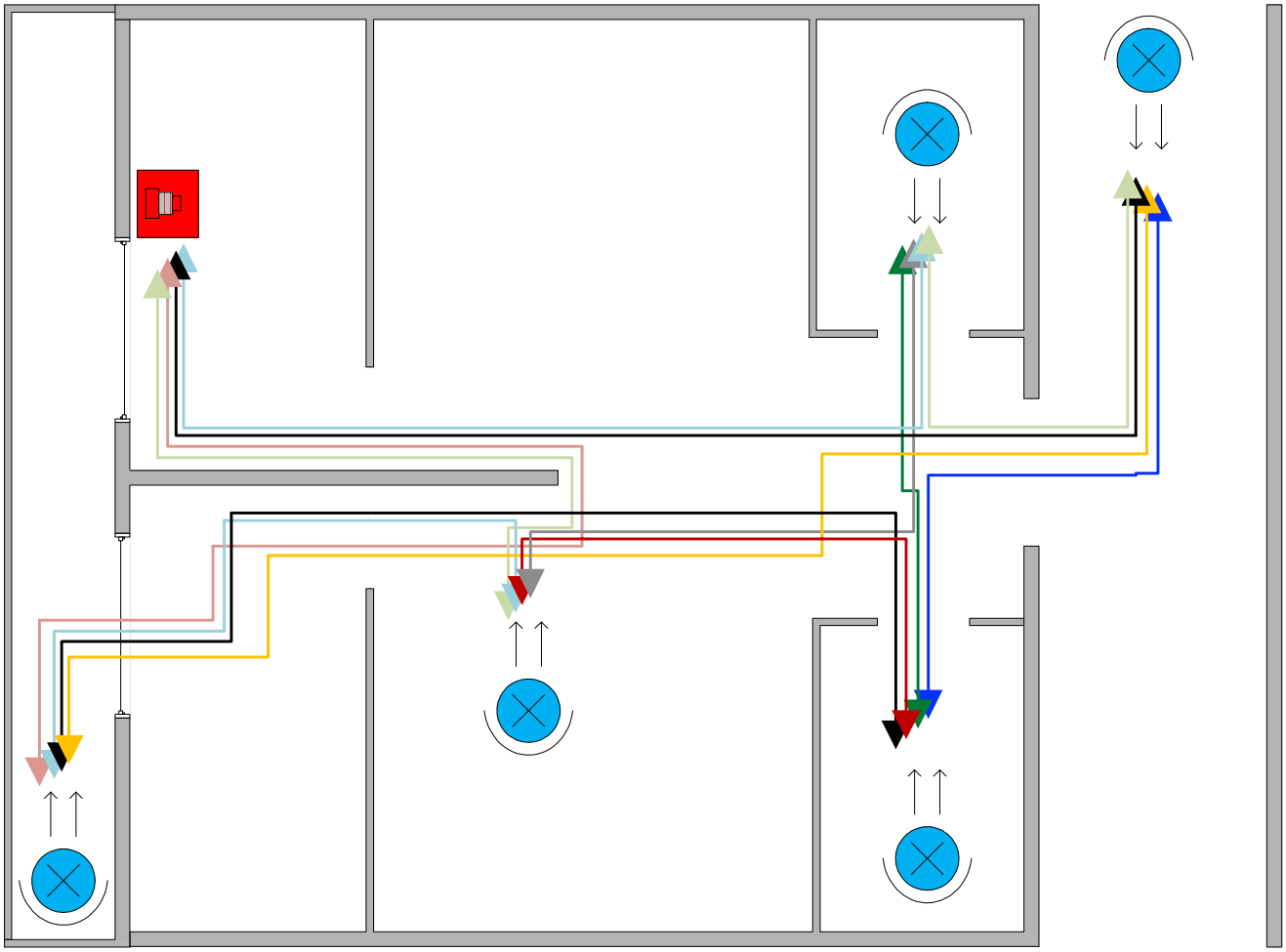


FIGURE 3.13. Path design for connecting environment targets. Each target is start point for reach other 4 different targets. Each path can be walked in both directions. Hence we have (targets*target reachable)/2= 12 different paths.

Path creation will be done by using web-based Rovio application. Targets will be identified by a suitable image to display during selection activities, and a red cross painted on the real environment ground will indicate the exact point where the robot will stop for reaching the desired target.

Note that before starting free-user mode experiment, some learning and testing session are needed for populating and validating the classifier. A minimum of 8 (better 16) learning sessions are needed, and also a minimum of 8 testing sessions.

Discussion

Primary goal of this project was to implements a concrete and functional system for acquiring, classifying and interpret brain activities for commanding a mobile device in real environment. Obtained results show how this goal is gettable by using technologies listed in previous chapters. BCI to robotic device link was created, and a suitable system for codifying and transmitting commands was also obtained. More can be done for enhancing such that system. First of all, MRPT libraries allows further development for device localization, perception, planning and navigation. Interesting results could be achieved by introducing additional modules for extracting environmental information from frame images obtained by robot onboard camera. It could be amusing recognize targets exact position by analyzing images features. Also, pre-compiled paths needs great effort both for create them and maintain them. Introducing a sensor-feedback target localization system where each target is provided with signaling sensor for detection and the robot creates its own path for achieving a target signaling its presence would be of great impact.

Visual Evoked Potentials approach were choose for its intrinsic easy-to-generate distinctive wave peaks. P300 peaks where also used because of San Camillo Hospital researchers made accurate studies concerning its mathematical model and classification algorithms. The necessity of external stimulation does, however, restrict the applicability of evoked potentials to a limited range of tasks. In a future view, a more natural and suitable alternative for interaction is to analyze components associated with spontaneous “intentional” mental activity. This is particularly the case when controlling robotics devices: subjects’ attention must be focused on driving and not on external stimuli.

A critical issue is how to improve the robustness of BCIs with the goal of making it a more practical and reliable technology. A first avenue of research is online adaptation of the interface to the user to keep the BCI constantly tuned to its owner. The point here is that, as subjects gain experience, they develop new capabilities and change their brain activity patterns. In addition, brain signals change naturally over time. In particular, this is the case from a session (with which data the classifier is trained) to the next (where the classifier is applied). Thus, online learning can be used to adapt the classifier throughout its use and keep it tuned to drifts in the signals it is receiving in each session.

Goal-based selection targets has been proved as a robust way of commanding remote devices. User can focus on target destination without wasting energies in continuous control for each basic movement. Limitations on this approach are related to VEP flashing-based stimulation paradigm. Since targets image are to be displayed on the screen, and they have to be placed at the greatest distance possible, a limited number of targets can be displayed in the same time. New approaches for add more targets on the screen have to be studied for increasing system usability and speed of selection.

References

[1] S. Sutton, M. Braren, J. Zubin, E. R. John. Evoked-Potential Correlates of Stimulus Uncertainty. Science 26, Vol. 150. no. 3700. 1965.

[2] J. J. Burmeister, K. Moxon, G.A. Gerhardt. *Ceramic-based multisite microelectrodes for electrochemical recordings*. Anal Chem. Vol. 72. 2000.

[3] G. Ensell, D.J. Banks, P.R. Richards, W. Balachandran, D.J. Ewins. *Silicon-based microelectrodes for neurophysiology, micromachined from silicon-on-insulator wafers*. Med. Biol. Eng. Comput. Vol. 38, Issue 2. 2000.

[4] K. Yoshida, W. Jensen, P. Norlin, M. Kindlundh, and U.G. Hofmann. *Characterization of silicon microelectrodes.* from the EU VSAMUEL project. Biomediinische Technik 2001.

[5] Duane E. Haines. *Neuroanatomy, atlas of structures sections systems, 6th edition*. The Point Lippincott Williams & Wilkins. 2004.

[6] M. Mumenthaler, H. Mattle. *Neurology, 4th edition*. Georg Thieme Verlag Stuttgart. New York. 2004.

[7] D. Purves, G. J. Augustine, D. Fitzpatrick, W. C. Hall, A. Lamantia, J. O. Mcnamara, M. Williams. *Neuroscience, 3th edition*. Sinauer Associates, Inc. Publishers. Sunderland, Massachusetts. S.A. 2004.

[8] S. Jacobson, E. M. Marcus. *Neuroanatomy for the Neuroscientist*. Springer. 2008.

[9] N. Birbaumer. *Brain-Computer Interfaces research, Coming of age*. Clinical Neurophysiology, Vol. 117, No. 3. 2006.

[10] T. W. Berger, John K. Chapin, Greg A. Gerhardt, Dennis J. McFarland, José C. Principe, Walid V. Soussou, Dawn M. Taylor, Patrick A. Tresco. *Brain-Computer Interfaces, an international assessment of research and developments trends*. Springer. 2008.

[11] J.R. Wolpaw, N. Birbaumer, D. McFarland, G.Pfurtscheller, T. Vaughan. Computer-interfaces for communication and control. Clinical Neurophysiology. Vol. 113. 2002.

[12] J. J. Vidal. *Toward direct brain-computer communication*. Annual review of biophysics and bioengineering. Vol. 2. 1975.

[13] Mikhail A. Lebedev, Miguel A.L. Nicolelis. *Brain–machine interfaces: past, present and future*. Trends in Neurosciences. Vol. 29, Issue 9. 2006.

[14] J.K. Chapin, R.A. Markowitz, K.A. Moxon, M.A.L. Nicolelis. *Direct real-time control of a robot arm using signals derived from neuronal population recordings in motor cortex*. *Nature Neuroscience. Vol. 2. 1999.*

[15] J. Wessberg, C.R. Stambaugh, J.D. Kralik, P.D. Beck, M. Laubach, J.K. Chapin, J. Kim, S.J. Biggs, M.A. Srinivasan, M.A. Nicolelis. *Real-time prediction of hand trajectory by ensembles of cortical neurons in primates.* Nature. Vol. 208. 2000.

[16] D. M. Taylor, S.I. Tillery, A.B. Schwartz. *Direct cortical control of 3D neuroprosthetic devices*. Science. Vol. 296. 2002.

[17] J.M. Carmena, M.A. Lebedev, R.E. Crist, J.E. O'Doherty, D.M. Santucci, D.F. Dimitrov, P.G. Patil, C.S. Henriquez, M.A. Nicolelis. *Learning to control a brain-machine interface for reaching and grasping by primates.* PLoS Biology. Vol. 1. 2003.

[18] J.P. Donoghue. Connecting cortex to machines: recent advances in brain interfaces. *Nature Neuroscienc. S*upp. Vol. 5. 2002.

[19] L. R. Hochberg, M.D. Serruya, G.M. Friehs, J.A. Mukand, M. Saleh, A.H. Caplan, A. Branner, D. Chen, R.D. Penn, J.P. Donoghue. *Neuronal ensemble control of prosthetic devices by a human with tetraplegia.* Nature. Vol. 442. 2006.

[20] N. Birbaumer, N. Ghanayim, T. Hinterberger, I. Iversen, B. Kotchoubey, A. Kubler, J. Perlmouter, E. Taub, H. Flor. *A spelling device for the paralyzed*. Nature. Vol. 398. 1999.

[21] S. G. Mason, G.E. Birch. *A brain-controlled switch for asynchronious control applications*. IEEE Trans. Biomed. Eng. Vol. 47. 2000.

[22] J. R. Wolpaw, D.J. McFarland, G.W. Neat, and C.A. Forneris. *An EEG-based brain-computer interface for cursor control.* Electroencephalography and Clinical Neurophysiology 78:252–259. 1991.

[23] McFarland, D.J., W.A. Sarnacki, and J.R. Wolpaw. *Brain-computer interface (BCI) operation: optimizing information transfer rates*. Biological Psychology 63:237–251. 2003.

[24] Miner, L.A., D.J. McFarland, and J.R. Wolpaw. 1998. *Answering questions with an electroencephalogram-based brain computer interface.* Arch. Phys. Med. Rehabil. 79:1029–1033.

[25] Wolpaw, J.R. and D.J. McFarland. 1994. *Multichannel EEG-based brain-computer communication.* Electroencephalogr. Clin. Neurophysiol. 90:444–449.

[26] Wolpaw, J.R. and D.J. McFarland. 2004. *Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans*. PNAS 101:17849–17854.

[27] Blankertz, B., G. Dornhege, M. Krauledat, K-R. Muller, V. Kunzmann, F. Losch, and G. Curio. 2006. *The Berlin braincomputer interface: EEG-based communication without subject training.* IEEE Trans. Neural Syst. Rehab. Eng. 14:147–152.

[28] Scherer, R., G.R. Muller, C. Neuper, B. Graimann, and G. Pfurtscheller. 2004. *An asynchronously controlled EEG-based virtual keyboard: Improvement of the spelling rate*. IEEE Trans. Biomed. Eng. 51:979–984.

[29] Muller-Putz, G.R., R. Scherer, G. Pfurtscheller, and R. Rupp. 2005. *EEG-based neuroprothesis control: a step towards clinical practice.* Neuroscience Letters 382:169–174.

[30] Middendorf, M., G. McMillian, G. Calhoun, and K. Jones. 2000. Brain-computer interfaces based on the steady-state visual-evoked response. IEEE Trans. Rehab. Eng. 8:211–214.

[31] Muller-Putz, G.R., R. Scherer, C. Brauneils, and G. Pfurtscheller. 2005. Steady-state visual evoked potential (SSVEP)-based communication: impact of harmonic frequency components. J. Neural Eng. 2:123–130.

[32] Cheng, M., X. Gao, S. Gao, and D. Xu. 2002. Design and implementation of a brain-computer interface with high transfer rates. IEEE Trans. Biomed. Eng. 49:1181–1186.

[33] E. Lalor1, S. P. Kelly, C. Finucane, R. Burke, R. B. Reilly, G. McDarby. Brain Computer Interface based on the Steady-State VEP for Immersive Gaming Control. Workshop Proceedings of International BCI Conference. Graz. 2004.

[34] Sellers, E.W., and E. Donchin. 2006. A P300-based brain-computer interface: Initial tests by ALS patients. Clinical Neurophysiology 117:538–548.

[35] F. Piccione, C. Volpato, M. Marchetti, K. Priftis, A. Merico, M. Cavinato, G. Sorarù, A. Palmieri, L. Tonin, S. Silvoni. Amyotrophic Lateral Sclerosis patients are able to direct a computer screen cursor using a P300-based BCI. Proceedings of 4th International Brain-Computer Interface Workshop and Training Course. Graz. 2008.

[36] Christian J. Bell, Pradeep Shenoy, Rawichote Chalodhorn, Rajesh P N Rao. Control of a humanoid robot by a noninvasive brain–computer interface in humans. Journal of Neural Engineering. Vol. 5. 2008.

[37] Vaughan, T.M., D.J. McFarland, G. Schalk, W.A. Sarnacki, D.J. Krusienski, E.W. Sellers, and J.R. Wolpaw. 2006. The Wadsworth BCI research and development program: at home with BCI. IEEE Trans. Neural Syst. Rehab. Eng. 14:229–233.

[38] L.Tonin, F.Piccione, A.Merico, L.Piron, C. Volpato, K.Priftis, M.Marchetti, M.Cavinato, E. Menegatti. Connecting a mobile robot to a brain computer interface. Workshop Proceedings of International Conference on Simulation, Modeling and Programming for Autonomous Robots. Venice. 2008.

[39] J. del R. Millán, F. Renkens, J. Mouriño, W. Gerstner. Non-Invasive Brain-Actuated Control of a Mobile Robot by Human EEG. IEEE Trans. on Biomedical Engineering, Vol. 51. 2004.

[40] Kazuo Tanaka, Kazuyuki Matsunaga, Shigeki Hori. Electroencephalogram-based control of a mobile robot. Electrical Engineering in Japan Volume 152 Issue 3, Pages 39 - 46. 2005.

[41] Tao Geng, John Q. Gan, Huosheng Hu. A self-paced online BCI for mobile robot control. International Journal of Advanced Mechatronic Systems 2010 - Vol. 2, No.1/2 pp. 28 - 35. 2010.

[42] A. Chella, E. Pagello, E. Menegatti, K. Prifitis et al. A BCI teleoperated museum robotic guide. Presented at 2009 International Conference on Complex, Intelligent and Software Intensive Systems. 2009.

[43] L. Tonin, E. Menegatti. Integrazione di un sistema BCI ed un robot olonomo. Padova. 2008.

[44] S.G. Mason, G.E. Birch. A General Framework for BCI Design. IEEE Transaction of Neural Systems and Rehabilitation engineering. Vol. 11, No. 1, March. 2003.

[45] F. Angeleri, S. Butler et al. Analysis of the Electrical Activity of the Brain. Wiley Blackwell Publisher. 1996.

[46] P. Perego et al. A Home Automation Interface for BCI application validated with SSVEP protocol. Presented at 4th Internationa Brain-Computer Interface Workshop 2008, Graz. 2008.

[47] E. W. Sellers, E. Donchin. A P300-based brain-computer interface: Initial tests by ALS patients. IEEE Transactions of Clinical Neurophysiology. Vol. 117. 2006.

[48] F.Piccione, K. Priftis et al. Amyotrophic Lateral Sclerosis patients are able to direct a computer screen cursor using a P300-based BCI. Proceedings of 4th International Brain-Computer Interface Workshop and Training Course. Graz. 2008.

[49] U. Hoffmann, J.M. Vesin, T. Ebrahimi, K. Diserens. An efficient P300-based brain-computer interface for disabled subjects. Journal of Neuroscience Methods. Vol. 167, No. 1. 2008.

[50] H. Hoffmann. Bayesian Machine Learning Applied in a BCI for disabled users. Lausanne, EPFL Editor. 2007.

[51] M. Marchetti. Brain-Computer Interfaces, confronto di tre interfacce guidate dalla P300. Università degli Studi di Padova Editore. 2007.

[52] N. Birbaumer, J.R. Wolpaw et al. Brain–Computer Communication Unlocking the locked-in. Institute of Medical Psychology and Behavioral Neurobiology, University of Tübingen, German. 2001.

[53] Y. Wang, X. Gao. Brain–Computer Interfaces Based on Visual Evoked Potentials. IEEE Engineering in Medicine and Biology Magazine. Vol. 27. No. 5. 2008.

[54] J.R. Wolpaw, N. Birbaumer, D. McFarland, G.Pfurtscheller, T. Vaughan. Computer-interfaces for communication and control. Clinical Neurophysiology. Vol. 113. 2002.

[55] D.J. McFarland, W.A. Sarnacki, and J.R. Wolpaw. Electroencephalographic (EEG) Control of Three-Dimensional Movement. Society for Neuroscience Abstract. 2008.

[56] I. Robinson, M. Hunter. Motor Neuron Diseases. Routledge Publisher. 1998.

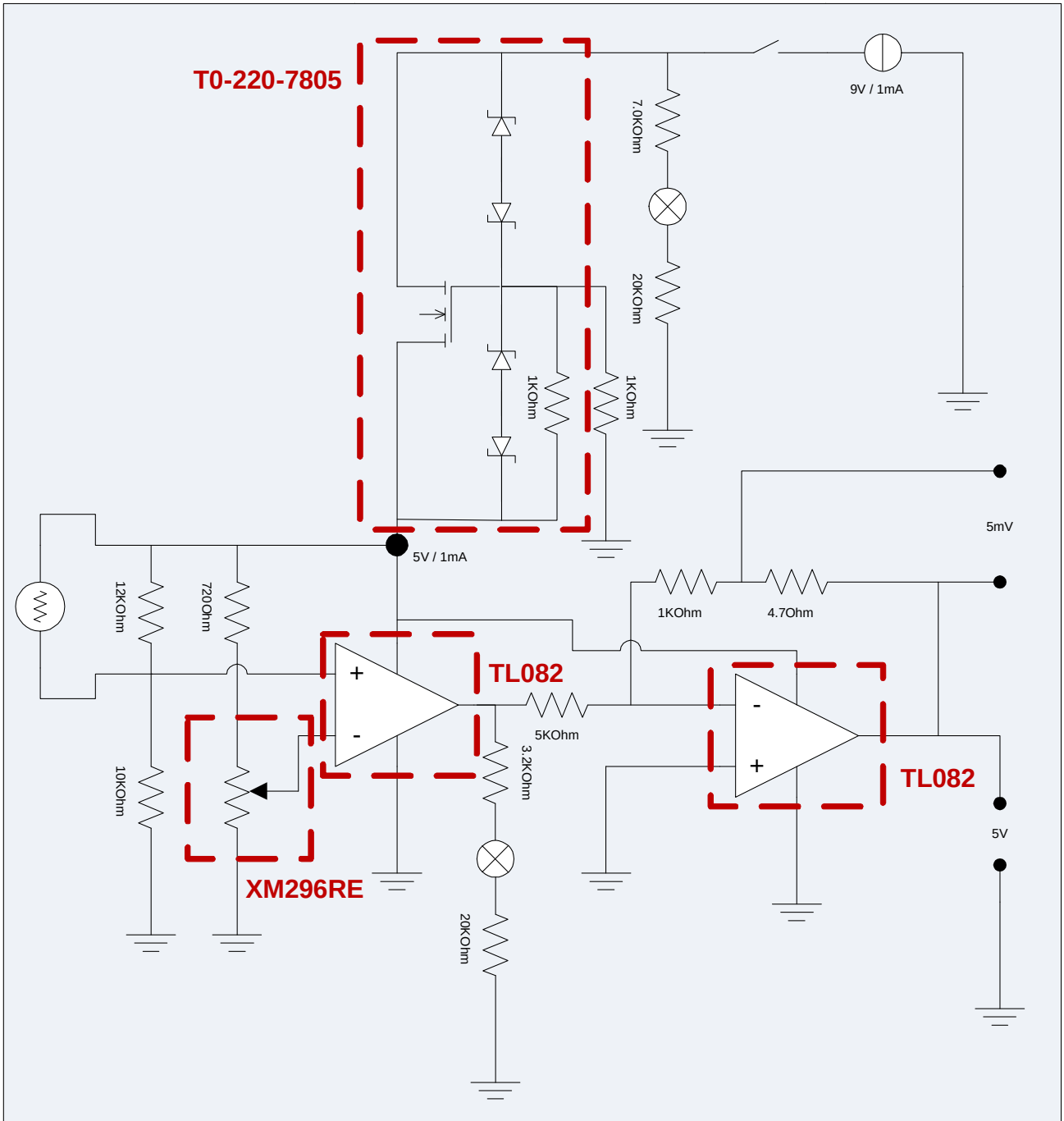
[57] F. Piccione, F. Giorgi, P. Tonin, K. Priftis, S. Giove, S. Silvoni, G. Palmas, F. Beverina. P300-based brain computer interface: Reliability and performance in healthy and paralised participants. Clinical Neurophysiology. Vol. 117, No. 3. 2006.

[58] G. Pfurtscheller, G. R. Müller-Putz et al. Rehabilitation with Brain-Computer Interface Systems. IEEE Computer Society. 2008.

[59] M. Tebaldi, E. Menegatti. Sistemi di Brain Computer Interface utilizzabili in robotica. Padova. 2007.

[60] Bruce H. Dobkin. The Clinical Science of Neurologic Rehabilitation, 2nd edition. Oxford University Press. 2003.

Appendix A - Schematics



Blinker	V 1.0	LC Group Inc. - LC Hardware	
Schematics. 15/12/2009		Criveller Luigi	Marco Gottardo

XMR296RE Datasheet

Electrical and Mechanical Specifications

Resistance Range
Standard: 500 ohms through 1 megohms.
Special: For non-standard resistance values, please contact CTS.

Resistance Tolerance
Standard: ±20%
Special: Tighter tolerances available in most resistance values.

Power Rating, Watts
0.15 watts @ 50°C derated to no load @ 85°C for linear curve.
0.1 watts @ 50°C derated to no load @ 85°C for regular non-linear curve.

Voltage Rating
Across end terminals:
Standard 250 VDC; Special: 350 VDC load not to exceed wattage ratings at given ambient temperature.

Angle of Rotation
Total Rotation: 300° ± 5°
Effective Electrical Rotation: 280° (approximate)

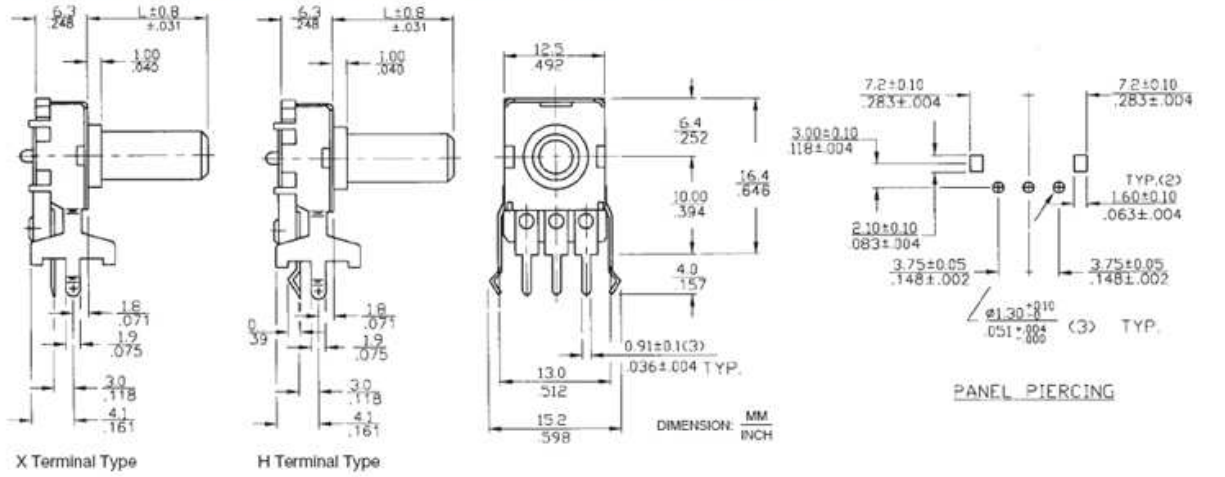
Rotational Torque
20 – 216 gcm

Stop Strength
5" lbs (5.76 kg-cm) minimum stop strength.

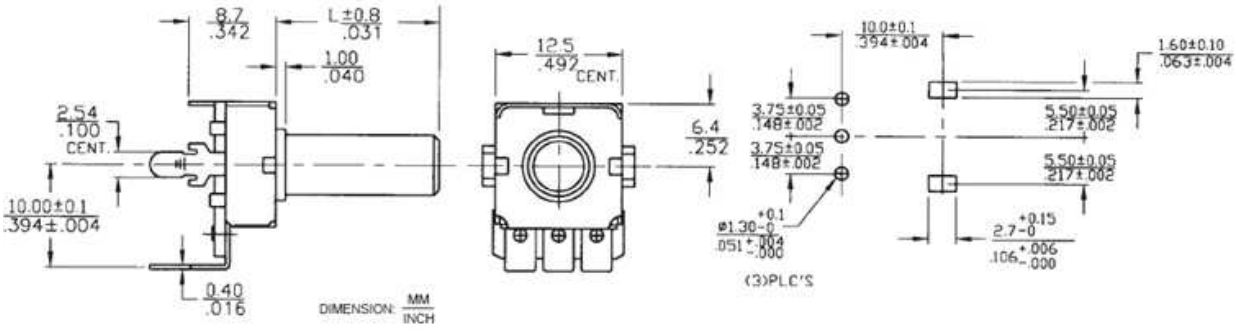
Mounting Information
P.C. board mounting
Type "X": Control shaft parallel to P.C. board.
Type "U": Control shaft vertical to P.C. board.

Detent
50% detent available.

XM 296RE Molded Shaft, Control Shaft Parallel to P.C. Board



UM 296RE Molded Shaft, Control Shaft Vertical to P.C. Board



TL082 Datasheet

General Description

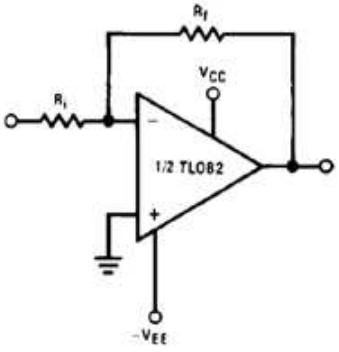
These devices are low cost, high speed, dual JFET input operational amplifiers with an internally trimmed input offset voltage (BI-FET II™ technology). They require low supply current yet maintain a large gain bandwidth product and fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents. The TL082 is pin compatible with the standard LM1558 allowing designers to immediately upgrade the overall performance of existing LM1558 and most LM358 designs.

These amplifiers may be used in applications such as high speed integrators, fast D/A converters, sample and hold circuits and many other circuits requiring low input offset voltage, low input bias current, high input impedance, high slew rate and wide bandwidth. The devices also exhibit low noise and offset voltage drift.

Features

■ Internally trimmed offset voltage	15 mV
■ Low input bias current	50 pA
■ Low input noise voltage	16nV/√ Hz
■ Low input noise current	0.01 pA/√ Hz
■ Wide gain bandwidth	4 MHz
■ High slew rate	13 V/μs
■ Low supply current	3.6 mA
■ High input impedance	$10^{12}\Omega$
■ Low total harmonic distortion A_V = 10, R_L = 10k, V_O = 20 Vp − p, BW = 20 Hz−20 kHz	<0.02%
■ Low 1/f noise corner	50 Hz
■ Fast settling time to 0.01%	2 μs

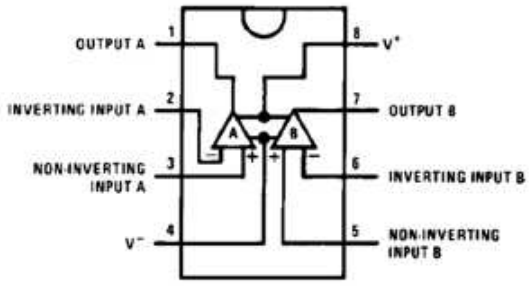
Typical Connection



TL/H/8357–1

Connection Diagram

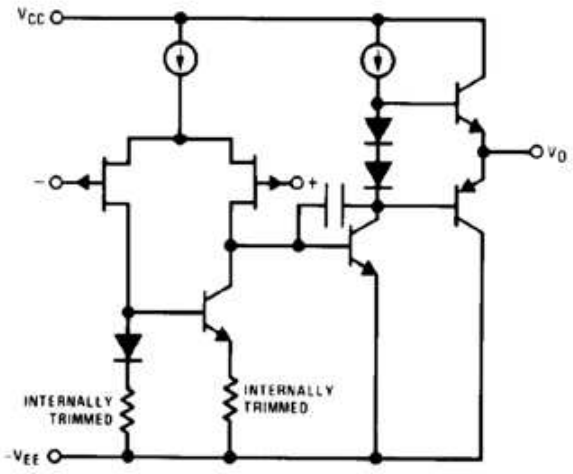
DIP/SO Package (Top View)



TL/H/8357–3

Order Number TL082CM or TL082CP
See NS Package Number M08A or N08E

Simplified Schematic



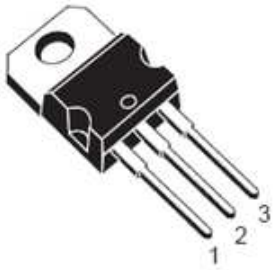
TL/H/8357–2

BI-FET II™ is a trademark of National Semiconductor Corp.

T0-220-7805 Datasheet

TYPE	V_{CES}	$V_{CE(sat)}$	I_C
STGP20NB37LZ	CLAMPED	< 2.0 V	20 A

- POLYSILICON GATE VOLTAGE DRIVEN
- LOW THRESHOLD VOLTAGE
- LOW ON-VOLTAGE DROP
- HIGH CURRENT CAPABILITY
- HIGH VOLTAGE CLAMPING FEATURE



TO-220

DESCRIPTION

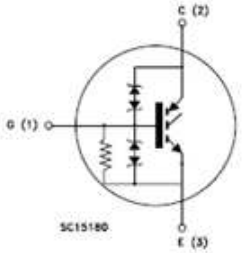
Using the latest high voltage technology based on patented strip layout, STMicroelectronics has designed an advanced family of IGBTs with outstanding performances.

The built in collector-gate zener exhibits a very precise active clamping while the gate-emitter zener supplies an ESD protection.

APPLICATIONS

- AUTOMOTIVE IGNITION

INTERNAL SCHEMATIC DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{CES}	Collector-Emitter Voltage (V_{GS} = 0)	CLAMPED	V
V_{ECR}	Reverse Battery Protection	20	V
V_{GE}	Gate-Emitter Voltage	CLAMPED	V
I_C	Collector Current (continuous) at T_c = 25 °C	40	A
I_C	Collector Current (continuous) at T_c = 100 °C	30	A
I_{CM}(•)	Collector Current (pulsed)	80	A
E_{AS}	Single Pulse Energy Tc = 25 °C	700	mJ
P_{tot}	Total Dissipation at T_c = 25 °C	150	W
	Derating Factor	1	W/°C
E_{SD}	ESD (Human Body Model)	4	KV
T_{stg}	Storage Temperature	-65 to 175	°C
T_j	Max. Operating Junction Temperature	175	°C

(•) Pulse width limited by safe operating area

Appendix B - Rovio APIs

CGI Commands Specification

Category	CGI Commands	Description
Movement	Rev.cgi	Refer to Movement Command table
Camera Control	GetData.cgi	Get MJPEG
	GetImage *	Get Image
	Video Streaming *	Stream Video
	ChangeResolution.cgi	Change the resolutions of camera's images.
	ChangeCompressRatio.cgi	Change the quality setting of camera's images.
	ChangeFramerate.cgi	Change the frame rate setting of camera's images.
	ChangeBrightness.cgi	Change the brightness of camera's images.
	ChangeSpeakerVolume.cgi	Change the Speaker Volume setting of IP_Cam.
	ChangeMicVolume.cgi	Change the Mic Volume setting of IP_Cam.
	SetCamera.cgi	Change camera sensor's settings
	GetCamera.cgi	Get camera sensor's settings
User Management	GetMyself.cgi	Get the username who sent this HTTP request.
	SetUser.cgi	Add a user or change the password for existed user.
	DelUser.cgi	Delete a user account.
	GetUser.cgi	Get the users list of IP Camera.
	SetUserCheck.cgi	Enable or disable user authorization check.
Time	SetTime.cgi (not tested)	Set time zone and time.
	GetTime.cgi	Get current IP Camera's time zone and time.
	SetLogo.cgi	Set a logo string on the image.
	GetLogo.cgi	Get a logo string on the image.
Network	SetIP.cgi	Tell IP Camera how to set an initial IP.
	GetIP.cgi	Get IP settings.

	SetWlan.cgi	Change settings for wireless LAN.
	GetWlan.cgi	Get settings for wireless LAN.
	SetDDNS.cgi	Set DDNS using dyndns.org / no-ip / dnsomatic service
	GetDDNS.cgi SetMac.cgi	Get DDNS setting Set Mac address
	GetMac.cgi	Get Mac address
Http Server	SetHttp.cgi	Set the parameters for HTTP server (Currently only TCP port).
	GetHttp.cgi	Get HTTP server's settings.
Mail	SetMail.cgi GetMail.cgi	Configure email for sending IPCam images. Get email for sending IPCam images.
	SendMail.cgi	Send an email with IPCam images.
Other	SetName.cgi	Set name of the camera.
	GetName.cgi	Get camera's name.
	GetStatus.cgi	Get run-time status of Rovio.
	GetLog.cgi	Get IP Camera's system logs information.
	GetVer.cgi	Get IP Camera's version.
	SetFactoryDefault.cgi	Change all settings to factory-default.
	Reboot.cgi	Reboot IP Camera.
	GetData.cgi	Get images/status with "multipart/x-mixed-replace" mime-type.
	GetAudio.cgi	Send audio to server and playback at the server
	SetMediaFormat.cgi	Set media format
	GetMediaFormat.cgi Upload.cgi	Get media format Upload firmware image (*.bin)
	Cmd.cgi	Use this command to combine several commands to a single http request,

Movements Command Specification

Action number	Function Name	Description
1 2	GetReport() StartRecoding()	Generates report of current status Start recording a path.
3	AbortRecording()	Terminates recording a path
4	StopRecording(string PathName)	Stop recording and store the path
5	Deletepath(string PathName)	Delete specific path
6	GetPathList()	Return stored paths

7	PlayPathForward(string PathName)	Replay a stored path from closest point to the end
8	PlayPathBackward (string PathName)	Replay a stored path from closest point to the beginning
9	StopPlaying()	Stop playing a path
10	PausePlaying()	Pause playing a path
11	RenamePath(string OldPathName, string NewPathName)	Rename the path name
12	GoHome()	Drive to home location without docking
13	GoHomeAndDock()	Drive to home location with docking
14	UpdateHomePosition()	Update home location
15	SetTuningParameters()	Set homing, docking and driving parameters
16	GetTuningParameters()	Return homing, docking and driving parameters
17	ResetNavStateMachine()	Stop and reset to idle
18	ManualDrive()	Accepts manual driving commands
19	RESERVED TestCommand()	RESERVED
20	GetMCUReport()	Return MCU report
21	ClearAllPaths()	Delete all paths
22	GetStatus()	Return navigation status
23	SaveParameter(long index, long value)	Stores robot parameters
24	ReadParameter(long index)	Return robot parameters
25	GetLibNSVersion()	Return libNS and NS sensor versions
26	EmailImage(string email)	Email current image / set an action (in path recording mode)
27	ResetHomeLocation()	Clear home location

Response Code Command Table

Code	Error Code	Description
0	SUCCESS	CGI command successful
1	FAILURE	CGI command general failure
2	ROBOT_BUSY	Robot is executing autonomous function
3	FEATURE_NOT_IMPLEMENTED	CGI command not implemented
4	UNKNOWN_CGI_ACTION	CGI nav command: unknown action requested
5	NO_NS_SIGNAL	No NS signal available
6	NO_EMPTY_PATH_AVAILABLE	Path memory is full
7	FAILED_TO_READ_PATH	Failed to read FLASH
8	PATH_BASEADDRESS_NOT_INITIALIZED	FLASH error

9	PATH_NOT_FOUND	No path with such name
10	PATH_NAME_NOT_SPECIFIED	Path name parameter is missing
11	NOT_RECORDING_PATH	Save path command received while not in recording mode
12	FLASH_NOT_INITIALIZED	Flash subsystem failure
13	FAILED_TO_DELETE_PATH	Flash operation failed
14	FAILED_TO_READ_FROM_FLASH	Flash operation failed
15	FAILED_TO_WRITE_TO_FLASH	Flash operation failed
16	FLASH_NOT_READY	Flash failed
17	NO_MEMORY_AVAILABLE	NA
18	NO_MCU_PORT_AVAILABLE	NA
19	NO_NS_PORT_AVAILABLE	NA
20	NS_PACKET_CHECKSUM_ERROR	NA
21	NS_UART_READ_ERROR	NA
22	PARAMETER_OUTOFRANGE	One or more parameters are out of expected range
23	NO_PARAMETER	One or more parameters are missing