UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Tesi di laurea

# "Inertial Odometry for Lower Limb Exoskeleton Control"


Relatore: Prof. Tortora Stefano

Laureando: Simion Riccardo

Correlatori: Dott. Trombin Edoardo


ANNO ACCADEMICO 2023 – 2024

Data di laurea 27/09/2024

## ABSTRACT

This thesis examines the development of a tracking system for a lower limb exoskeleton using inertial odometry techniques. The system utilizes inertial sensors, such as accelerometers and gyroscopes, to monitor and manage the device's movements precisely and efficiently. The Robot Operating System (ROS) framework has been employed for system integration and management, facilitating modular development and communication between various modules. The proposed approach aims to improve the stability, precision, and reliability of the exoskeleton control system, representing a significant advancement in wearable robotics and offering potential applications in motor rehabilitation and recovery.



## SOMMARIO

Il presente elaborato esamina lo sviluppo di un sistema di tracciamento per un esoscheletro degli arti inferiori utilizzando tecniche di odometria inerziale. Il sistema utilizza sensori inerziali, come accelerometri e giroscopi, per monitorare e gestire i movimenti del dispositivo in modo preciso ed efficiente. Il framework Robot Operating System (ROS) è stato impiegato per l'integrazione e la gestione del sistema, facilitando lo sviluppo modulare e la comunicazione tra i vari componenti. L'approccio proposto mira a migliorare la stabilità, la precisione e l'affidabilità del sistema di controllo dell'esoscheletro, rappresentando un significativo progresso nella robotica indossabile e offrendo potenziali applicazioni nella riabilitazione e nel recupero motorio.

# TABLE OF CONTENTS

# Introduction

## 1.1 Exoskeletons and LLEs

An exoskeleton is a wearable robot that enhances, support and assist human limbs in daily tasks. It is a device that augments human performance by increasing strength, endurance, and other physical capabilities of healthy or able-bodied individuals [1].

Exoskeletons can be classified as either active or passive. Passive devices differ from active ones because they do not contain actuators; instead, they rely on natural human movement to generate the energy needed to operate the device which is achieved through materials, springs, or dampers. This can lead to increased stress and fatigue, but also can provide mechanical benefits and protection to the user. An active exoskeleton, on the other hand, is powered by actuators such as electric motors, pneumatics, hydraulics, or a combination of multiple technologies to enhance human strength and reduce the body's energy consumption [2, 3].

Exoskeletons find applications in various fields such as in clinical environments for rehabilitation, restoration of muscular weakness and augmentation or enhancement of intact operators, for example, in the military field. That said, there are three fundamental categories of exoskeletons: rehabilitation exoskeletons, augmentation exoskeletons, and assistive exoskeletons [4].

Rehabilitation exoskeletons, which HAL exoskeleton shown in Figure 1 is a realistic example, are designed to follow a specific treatment or therapy for the wearer during which it is assumed that the patient partially or totally regains his mobility and must be useful for a wide range of patients considering their physical differences (disease, weight, height, gender). Rehabilitation therapy with robots ensures a greater number of repetitive movements. This allows that the proprioceptive input coming from limb movements stimulates the brain and spinal cord neuroplasticity, which is the key for restoring the mobility of affected limbs in cases of some neuromuscular disorders. For this reason, rehabilitation exoskeletons often encourage the user's physical recovery by gradually reducing assistance as the user improves [5]. Augmentation exoskeletons, such as the XOS2 in Figure 2, are mostly used by healthy users to reduce the (metabolic) effort required for tasks or

to enhance his physical abilities, making the user able to perform tasks that normally would be incapable of.

*Figure 1: HAL for rehabilitation*

*Figure 2: Raytheon Sarcos XOS2 for military porpouse*

Assistive exoskeletons mainly focus on users that have permanently lost their mobility. This type of exoskeletons often requires a high level of personalization to the user's needs, which is very expensive and challenging to satisfy in the real world. The fact that the specialized equipment is not available outside the laboratory and everyday walking occurs at different speeds and with varying durations makes providing beneficial assistance in the real world very difficult [6].

When discussing exoskeletons, the focus will be on Lower Limb Exoskeletons (LLEs). LLEs are mechatronic devices which are applied to the external surface of the body and can provide ergonomic structural support, allowing lower limb movements. LLEs have been developed since the 1960s to enable spinal cord injury (SCI) patients to walk, by regaining their locomotion or recovering their gait. The initial design started with orthoses, such as the Hip Guidance Orthosis (HGO) shown in Figure 3, which required much physical effort on the part of the patient and the loss of a lot of energy in attempting to walk [7].

*Figure 3: Hip Guidance Orthosis (HGO)*

Nowadays, LLEs combine elements of a mechanical structure, sensor, controller and actuator. Speaking of actuators, LLEs can be actuated in three different ways: active actuation, passive actuation, hybrid actuation. Active actuators comprise electric, pneumatic and hydraulic actuators. Passive actuators comprise non-powered components or elastic components such as springs, which can store energy and are based on the principles of gravity for balance. Passive LLEs depend solely on the physical effort of the patient and allow for very slow walking speed. Hybrid actuators are a combination of the two [8].

Currently, there are two main types of LLEs: the ones for full mobilization, and the ones for partial assistance. Full mobilization LLEs are designed to move the legs of people suffering from a severe loss of motor control or motor disorders, typically in people with SCI. The actuators must have a high torque capability because they provide the entire torque required for the movement.

Such devices are available commercially since 2011, when the ReWalk in Figure 4 (ReWalk Robotics, Israel) was released on the market.

*Figure 4: ReWalk Exoskeleton*

They could be developed quickly because their control strategy can be simply positioning control over time. There is no need to collaborate with an existing voluntary movement of the legs, because there is none (or it is very weak) and thus the user's legs are assumed to be passive. The start of the gait is often triggered by the upper body movements or buttons pressed by the fingers, which is simple to implement.

Partial assistance devices are generally lighter, targeting various less severe handicaps. These could be the loss of stamina because of aging, the loss of strength or coordination because of incomplete spinal cord injury SCI, stroke, neurodegenerative diseases, etc. These devices can also assist the gait of healthy people, which can be useful for endurance augmentation purposes. This is more challenging because the device must assist more than it is hindering its user, given the complex nature of the interaction with the user [9].

Understanding this difference in the end goals, there is a lot of commonalities between the two applications in terms of the techniques used for control.

## 1.2 Control of an Exoskeleton

From the control perspective, the main challenge for gait assistance is to contribute to the intended movement, where effective collaboration can be interpreted in different ways, depending on the context and application. In general, for partial assistance it would mean synergy in forces or torques between the user and the device, and for full mobilization it would be coordination between the movements of the exoskeleton and those of the user's upper body. Many strategies are used to identify the user's intent and apply an appropriate torque or motion accordingly [10].

A study conducted by Tucker et al. [11] presents a generalized framework (Figure 5) with a three-level hierarchy control structure.

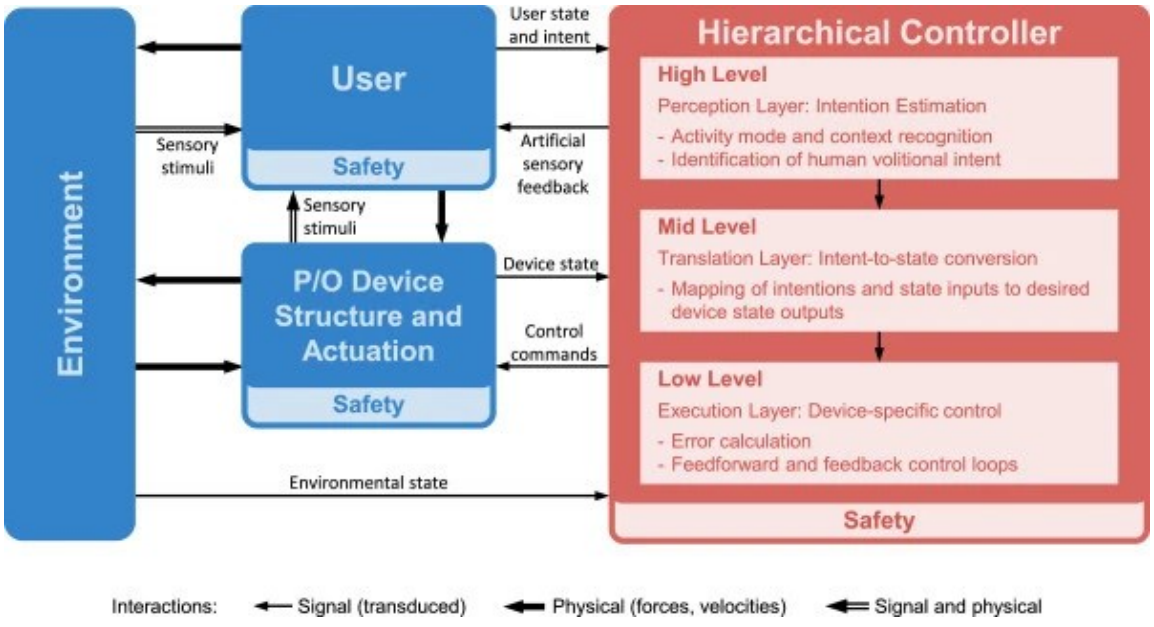



*Figure 5: Tucker framework representation*

The proposed framework illustrates the physical and signal-level interactions between a powered lower limb prosthetic or orthotic (P/O) device, a user, and his environment. The arrows indicate the exchange of power and information between the various components of the P/O ecosystem. A hierarchical control structure is implemented, with the estimation of the user's locomotive intent taking place at the high-level, translation of the user's intent to a desired device state at the mid-level, and a device-specific controller responsible for realizing the desired device state at the low-level. Safety mechanisms underly all aspects of P/O design, including those which are mechanically passive and those which are actively controlled.

- High-level: the high-level control determines the general behaviour of the exoskeleton. The controller must perceive the user's locomotive intent, allowing the switch between different operating modes, depending on the desired type of activity. A reliable high-level control is crucial for the usability of exoskeletons for people in real-world situations and everyday life, where the inputs can come from both the user and the environment. There are several high-level control strategies:
    - Manual User Input (MUI): The exoskeleton is directly controlled by the user thanks to input devices such as buttons, as in the CUHK-EXO exoskeleton [12], or voice commands, as in the Vanderbilt Powered Orthosis [13]. MUI is commonly used for complete SCI patients, because no input can be obtained from lower limbs.
    - Brain-Computer Interface (BCI): Electrodes measure the user's brain activity to determine the mode of operation. Currently the most common method used among the different brain signal recording methods [14] is electroencephalography (EEG) since it is non-invasive and easier to use.
    - Movements recognition (MOV): The controller's behaviour is entirely dependent on the user's movements or intent to move. The special feature of this method is that is not

required any cognitive load or direct input from the user, making the interaction natural and intuitive. Generally [15] for this method a machine learning or fuzzy logic algorithm are involved to process Inertial Measurement Data (IMU) and joint sensors.

- Mid-level: The mid-level controller allows to translate the user's motion intentions from the high level to the desired device sequence for tracking by a low-level controller. It is at this level of control that the user's state within the gait cycle is determined, and a control law applied. It may have the form of a position/velocity, torque, impedance, or admittance controller.
- Low-level: The low-level controller acts as a specific control layer device from which is derived the actuator that tracks the state in the mid-level. Actuators used in robotics are generally direct-current motors that are current-driven, and the field of wearable robotics is no exception. The target current is determined depending on the type of low-level controller. This level serves as the force, torque and position or angle of the exoskeleton joint.

## 1.3 Purpose and Structure of the Thesis

**Purpose**

This thesis provides an overview of the results obtained during my time at the IAS Laboratory in the Department of Information Engineering at the University of Padova. The aim of the thesis is to determine the movements and, consequently, the position of the exoskeleton over time using inertial odometry techniques. To achieve this, I was provided with a .bag file containing a recorded session of the exoskeleton in use (ALICE). By applying the techniques presented in the following chapters, I was able to extract its movements. I used ROS framework, as it facilitates the acquisition, processing, and publication of the data necessary to obtain the exoskeleton's pose and position.

**Structure**

In the first chapter, I presented exoskeletons in general and their specific employment in the medical field, focusing on the LLEs. After that I reported the standard for exoskeletons control, analysing the different levels of the control hierarchy.

In the next chapters, the topics will be organized as follows. In the second chapter I present the ALICE Open-Source Exoskeleton, which is the one used in the laboratory, describing its features. Afterwords, I describe the IMU, which is the type of sensors used to capture the motion and the orientation of the exoskeleton and the techniques applied to the sensors data to obtain exoskeleton’s pose and position over time through IMU integration. Lastly, I present the ROS framework, and the algorithms/methods used. In the third chapter I report the experiments I run to validate the methods used. In the last chapter, I discuss the outcomes of the experiments, focusing on possible limitations and future applications.

# Materials and Methods

## 2.1 ALICE Open-Source Exoskeleton

The first version of ALICE Exoskeleton was developed in honour of Cybathlon 2016 [16]. To make the project available to developing countries, a simplified alternative which would require only easy-to-find materials and low construction budget was designed.

The exoskeleton combines hardware, software and operation conditions focused on reducing complexity while achieving the minimum required functionality for previously unattended patients. ALICE has been successfully tested with patients exhibiting different medical conditions, such as Muscular Dystrophy(MD), Cerebral Palsy (CP), Spinal Muscular Atrophy (SMA), and Cephalic Disorder (CD). Short term clinical usage has been proved to benefit patient positive impacts in blood circulation and internal organ functioning, and to improve the mental well-being of patients [17].

The exoskeleton used consists of 3 links: pelvis, femur and tibia, each of which includes one electric actuator. The first version of ALICE includes 4 active degrees of freedom (DOFs): hip flexion / extension and knee flexion / extension in both legs. It also has a passive DOF: dorsal / plantar flexion of the ankle. The exoskeleton is powered by automotive DC 26Nm actuators [18], and a significant portion of parts is manufactured on PLA(Polylactic Acid) [19] and commercial 3D-Printing composites.

ALICE is adjustable for adult patients with femur and tibia lengths between 35 cm to 50 cm and pelvic width from 29 cm to 40 cm [20]. ALICE can be adhered directly to the pelvis and has two adjustable elements that adapt to the femur and tibia respectively, as shown in Figure 6, allowing the user a good wearability.

*Figure 6:*
*CAD of the exoskeleton robot for rehabilitation, ALICE*

Each joint described above has the Range of Motion (ROM) shown in Figure 7 and found in the literature [21] .

| Joint | Action | ROM |
|---|---|---|
| Hip | Extension/Flexion | −30°/120° |
| | Abduction/Adduction | −50°/30° |
| Knee | Flexion/Extension | −120°/0° |
| Ankle | Plantar/Dorsal Flexion | −40°/30° |

*Figure 7: ROM of different joints*

## 2.2 IMU and IMU Integration

An Inertial Measurement Unit is a solid-state sensor capable of measuring motion. An IMU is in fact composed of 3 different components, as can be seen in Figure 8: accelerometer, gyroscope and magnetometer (optional) which respectively allow to obtain the linear acceleration (the rate of change in velocity), the angular rate (change in angular velocity), and the magnetic field force along the three axes X, Y and Z and provides movement data. The combination of these three sensors provides a complete picture of the orientation and movement of an object in space.

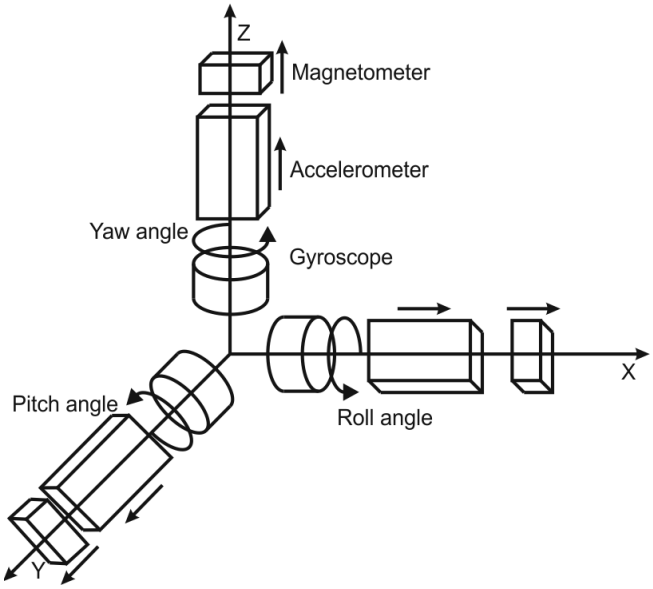



*Figure 8: 3-axis IMU representation*

- Accelerometer:

  Accelerometers, Figure 9, are electro-mechanical devices that detect static or dynamic acceleration forces. Static forces include gravity, while dynamic forces can include vibrations and movements [22]. As said above, the accelerometers used in IMUs measure acceleration on three different axes.


*Figure 9:MEMS Accelerometer (most widely used on the market)*

The use of accelerometers for kinematic analysis of limb movement has been explored by numerous researchers [23]. If the acceleration of a segment is known, with a single mathematical integration it is possible to obtain its velocity and with a second integration its position, provided that both position and speed are known at a certain point in the measurement period. However, these requirements, combined with the 'drift' that the accelerometer often undergoes, have prevented it from entering a widespread use for this purpose. If the limb rotates and changes position, as is usually the case, gyroscopes are required.

- Gyroscope:

  Gyroscopes, Figure 10, are electro-mechanical devices that allow to measure angular velocity, which indicates the speed of rotation. Gyros are essential for determining correct orientation in autonomous devices, such as LLEs.

  Triple-axis gyroscopes measure rotation around the three orthogonal axis X, Y and Z.

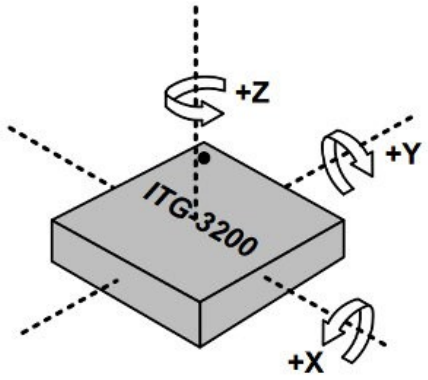



*Figure 10: Triple-Axis Digital-Output Gyroscope - ITG-3200*

The angular orientation (attitude) of the object can be determined by integrating its angular velocity over time. Like accelerometers, gyroscopes also suffer from drift, which can lead to errors in orientation over time.

- Magnetometer:

  Magnetometers, Figure 11, measure earth's magnetic field along the three axes. They are not used often as they can be distorted by metal or electronic objects in the surrounding environment, and it is difficult to get a correct calibration.


*Figure 11: Triple Axis Compass Magnetometer Sensor Module*

Combining a three-axis accelerometer and a three-axis gyroscope results in an IMU sensor of 6 DOFs. An IMU with 9 DOFs can be achieved by adding the magnetometer.

**IMU Integration**

The term IMU integration refers to the process of obtaining the position and orientation over time of a moving object from the raw data of an IMU. As mentioned above, to obtain the position of the device from the output of the accelerometer it is necessary to make two mathematical integrations. Similarly, the output of the gyroscope is integrated to get the angular orientation.

When a large amount of data is processed over time, errors can accumulate producing a totally uncontrolled variation from the true values. This error is called **drift**. For example, accelerometers are susceptible to noise and biases that, when integrated twice to obtain position, can lead to large cumulative errors and even a small bias in the gyroscope can result in a large error in orientation after a long period [24]. The drift is one of the primary challenges in using IMUs for accurate motion tracking. Sensor fusion techniques, such as complementary filters, Kalman

filters or Madgwick filters, are often employed to correct these errors by combining data from multiple sensors (E.g. GPS) [25]. In the paragraph 2.4, I will delve deeper into IMU integration algorithms, which are central to the work presented in this paper.

## 2.3 ROS

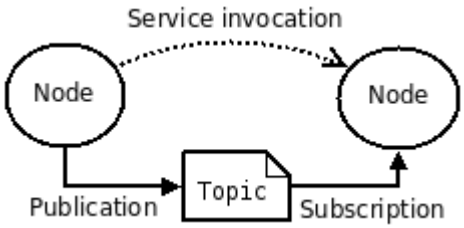
ROS (Robot Operating System) is an open-source operating system, a framework for robotic development, which provides a wide range of libraries, tools and conventions that simplify the development of applications for robots. ROS is a crucial component because it ensures that the various elements of a robotic system communicate with each other, resulting in efficient data exchange between software and hardware components. ROS currently only works on Unix-based platforms [26]. In fact, I used Ubuntu 20.04. The core ROS system, along with useful tools and libraries are regularly released as a ROS Distribution. This distribution is similar to a Linux distribution and provides a set of compatible software for others to use and build upon. The ROS distribution I used is ROS Noetic.

ROS has some important concepts [27]:

- Packages: packages allow software organization in ROS. A ROS package may contain ROS runtime processes named “nodes”, libraries, datasets, configuration files. A package is the smallest entity that is possible to build, run or release; its purpose is to encapsulate a specific functionality. This allows for a structured and modular approach that makes it easy to re-use, maintain and share code.
- Nodes: nodes are processes that perform computation in ROS. A ROS node must be written with the use of a ROS client library, such as *roscpp* (for C++) or *rospy* (for Python). Nodes communicates with each other using messages.
- Master: the Master provides a naming service for ROS. It helps nodes to find and communicate with each other. The Master keeps a record of active nodes and communications between them.
- Messages: communication in ROS is via messages exchanged between the different nodes. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported,

as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).

- Topics: topics are the communication channels used by nodes to exchange messages in ROS. In fact, messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by publishing it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will subscribe to the appropriate topic.
- Services: the publish/subscribe model used for the messages is not appropriate for request/reply interactions, which are often required in a distributed system. Request/reply is done via services. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply.
- Bags: bags are files used to record and reproduce messages exchanged between nodes within a ROS system. These files contain a time record of the data of messages sent and received while running a ROS system. Bags are an important mechanism for storing data, such as sensor data, that can be difficult to collect but is necessary for developing and testing algorithms.




The ROS Master acts as a nameservice in the ROS Computation Graph. It stores topics and services registration information for ROS nodes. Nodes communicate with the Master to report their registration information. As these nodes communicate with the Master, they can receive information about other registered nodes and make connections as appropriate. The Master will also make callbacks to these nodes when this registration information changes, which allows nodes to dynamically create connections as new nodes are run.

Nodes connect to other nodes directly; the Master only provides lookup information. Nodes that subscribe to a topic will request connections from nodes that publish that topic and will establish that connection over an agreed upon connection protocol.

## 2.4 Algorithms/Methods used

In this section, I am going to delve into the methods employed to determine the position of the exoskeleton using inertial odometry techniques.

The first step in the process involved a detailed analysis of the data recorded in the '.bag' file provided by the IAS Laboratory. This file contained the raw sensor data captured thanks to the Intel RealSense D435i camera, during a session where the ALICE exoskeleton was in use.

The Intel RealSense camera D435i combines D435 standard camera with the addition of an inertial measurement unit. The IMU, which detects movement and rotation in 6 degrees of freedom, combines accelerometers and gyroscopes to track rotation and movement across three axes [28].

The focus was on extracting relevant information, specifically the linear acceleration and angular velocity, which are crucial for determining the movement of the exoskeleton.

Using the ROS command '*rosbag info'*:

```
~/catkin_ws/src/tesi$ rosbag info rectangle_path.bag
```

I was able to obtain the information related to the characteristics of the .bag file, as shown in Figure 12:

```
path:         rectangle_path.bag
version:      2.0
duration:     44.7s
start:        Aug 26 2024 17:15:08.71 (1724685308.71)
end:          Aug 26 2024 17:15:53.42 (1724685353.42)
size:         1.6 GB
messages:     12944
compression:  bz2 [1341/1341 chunks; 45.83%]
uncompressed: 3.5 GB @ 79.1 MB/s
compressed:   1.6 GB @ 36.3 MB/s (45.83%)
types:        realsense2_camera/IMUInfo  [a02adb3a99530b11ba18a16f40f9512a]
              realsense2_camera/Metadata [4966ca002be16ee67fe4dbfb2f354787]
              sensor_msgs/CameraInfo     [c9a58c1b0b154e0e6da7578cb991d214]
              sensor_msgs/Image          [060021388200f6f0f447d0fcd9c64743]
              sensor_msgs/Imu            [6a62c6daae103f4ff57a132d6f95cec2]
topics:       /camera/accel/imu_info         1 msg     : realsense2_camera/IMUInfo
              /camera/color/camera_info   1341 msgs    : sensor_msgs/CameraInfo
              /camera/color/image_raw     1340 msgs    : sensor_msgs/Image
              /camera/color/metadata      1340 msgs    : realsense2_camera/Metadata
              /camera/gyro/imu_info          1 msg     : realsense2_camera/IMUInfo
              /camera/imu                 8921 msgs    : sensor_msgs/Imu
```

*Figure 12: result of rosbag info*

As can be seen from the image presented above, the messages of interest are published on the ROS topic /camera/imu. At this point, it was essential to study their nature. To do so, I used '*rostopic echo /camera/imu'* while the recording from the bag file was played using '*rosbag play rectangle_path.bag*', thus printing the data over time. I organized them into a .csv file to facilitate a more straightforward analysis:

```
~/catkin_ws/src/tesi$ rostopic echo -b rectangle_path.bag -p /camera/imu > imu_data.csv
```

The '.csv' file below shows the results of the steps performed. It can be observed in Figure 13 that at each time instant, the linear acceleration along the three axes x, y, z and the angular velocities also along x, y, z are measured.

| | A | B | C | D | E | R | S | T | U | V | W | X | Y | Z | AA | AB | AC | AD | AE | AF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | %time | field.header.seq | field.header.stamp | field.header.frame_id | field | field.angular_velocity.x | field.angular_velocity.y | field.angular_velocity.z | field | field | field | field | field | field | field | field | field | field.linear_acceleration.x | field.linear_acceleration.y | field.linear_acceleration.z |
| 2 | 1,71225095453762E+018 | 0 | 1,71225095453182E+018 | camera_imu_optical_frame | 0.0 | -0.06283184885978699 | -0.003490658476948739 | 0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2808844107666901 | -4.770470095017506 | -8.483767000875378 |
| 3 | 1,71225095453763E+018 | 1 | 1,71225095453683E+018 | camera_imu_optical_frame | 0.0 | -0.06632251292467117 | 0.001745329238474369 | 0.001745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2660394645667233 | -4.775418400392224 | -8.681699697076974 |
| 4 | 1,71225095454744E+018 | 2 | 1,71225095454184E+018 | camera_imu_optical_frame | 0.0 | -0.05759586393833160 | 0.0 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2421353148018497 | -4.775838375091553 | -8.834363858130292 |
| 5 | 1,71225095454745E+018 | 3 | 1,71225095454685E+018 | camera_imu_optical_frame | 0.0 | -0.04537855833768844 | 0.001745329238474369 | -0.008726646192371847 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2173949495509765 | -4.775838375091553 | -8.982805824078235 |
| 6 | 1,71225095455756E+018 | 4 | 1,71225095455186E+018 | camera_imu_optical_frame | 0.0 | -0.02792526781558990 | 0.006981316953897476 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2065086367482928 | -4.794313602496983 | -9.011172943164706 |
| 7 | 1,71225095455764E+018 | 5 | 1,71225095455687E+018 | camera_imu_optical_frame | 0.0 | -0.005235987715423107 | 0.008726646192371845 | -0.006981316953897476 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1966106258817649 | -4.814109632963063 | -9.030968973630786 |
| 8 | 1,71225095456771E+018 | 6 | 1,71225095456188E+018 | camera_imu_optical_frame | 0.0 | 0.010471975430846214 | 0.013962633907794952 | -0.012217304669320583 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1772973655519208 | -4.819773805465246 | -8.933037381525008 |
| 9 | 1,71225095456772E+018 | 7 | 1,71225095456689E+018 | camera_imu_optical_frame | 0.0 | 0.03839724510908127 | 0.012217304669320583 | -0.012217304669320583 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1575036645539793 | -4.824722223195514 | -8.829120606918936 |
| 10 | 1,71225095458041E+018 | 8 | 1,71225095457719E+018 | camera_imu_optical_frame | 0.0 | 0.057595863938331604 | 0.013962633907794952 | -0.012217304669320583 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1281166403994307 | -4.820073255995531 | -8.634053123449402 |
| 11 | 1,71225095458041E+018 | 9 | 1,71225095457691E+018 | camera_imu_optical_frame | 0.0 | 0.06632251292467117 | 0.013962633907794952 | -0.010471975430846214 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.0984282203705359 | -4.815125194717384 | -8.43613019293469 |
| 12 | 1,71225095458749E+018 | 10 | 1,71225095458192E+018 | camera_imu_optical_frame | 0.0 | 0.07155849784612656 | 0.012217304669320583 | -0.013962633907794952 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1078418218600162 | -4.829727871210323 | -8.24309967224884 |
| 13 | 1,71225095459706E+018 | 11 | 1,71225095458693E+018 | camera_imu_optical_frame | 0.0 | 0.06457718461751938 | 0.010471975430846214 | -0.010471975430846214 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.117768832849209 | -4.844425449008505 | -8.050754505888086 |
| 14 | 1,71225095459707E+018 | 12 | 1,71225095459193E+018 | camera_imu_optical_frame | 0.0 | 0.057595863938331604 | 0.013962633907794952 | -0.008726646192371847 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1326130686568777 | -4.834529309346752 | -7.966637318763196 |
| 15 | 1,71225095460708E+018 | 13 | 1,71225095459694E+018 | camera_imu_optical_frame | 0.0 | 0.03490658476948738 | 0.003490658476948738 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1472203324110582 | -4.824268620199536 | -7.885872709484591 |
| 16 | 1,71225095460709E+018 | 14 | 1,71225095460195E+018 | camera_imu_optical_frame | 0.0 | 0.012217304669320583 | 0.008726646192371845 | -0.006981316953897474 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1521678159671053 | -4.79953124189677 | -7.940295182398054 |
| 17 | 1,71225095461705E+018 | 15 | 1,71225095460696E+018 | camera_imu_optical_frame | 0.0 | -0.006981316953897474 | 0.008726646192371845 | -0.012217304669320583 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1573270781207054 | -4.774786671643355 | -7.999571312548717 |
| 18 | 1,71225095461706E+018 | 16 | 1,71225095461197E+018 | camera_imu_optical_frame | 0.0 | -0.02792526781558990 | 0.006981316953897476 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.167225588261780 | -4.750404034013499 | -8.167845969906331 |
| 19 | 1,71225095462711E+018 | 17 | 1,71225095461698E+018 | camera_imu_optical_frame | 0.0 | -0.03665191307663917 | 0.010471975430846214 | -0.008726646192371847 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1777245911904625 | -4.726202761274 | -8.336689826777341 |
| 20 | 1,71225095462711E+018 | 18 | 1,71225095462199E+018 | camera_imu_optical_frame | 0.0 | -0.04712389037013054 | 0.003490658476948738 | -0.010471975430846214 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1975154742348385 | -4.716307338907538 | -8.514807910539451 |
| 21 | 1,71225095463687E+018 | 19 | 1,71225095462700E+018 | camera_imu_optical_frame | 0.0 | -0.05235987901687622 | 0.0 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2165283240134303 | -4.707582956318714 | -8.680660764809892 |
| 22 | 1,71225095463688E+018 | 20 | 1,71225095463201E+018 | camera_imu_optical_frame | 0.0 | -0.04712389037013054 | 0.005235987715423107 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.226427071910864 | -4.715232327466876 | -8.812394925513939 |
| 23 | 1,71225095464686E+018 | 21 | 1,71225095463702E+018 | camera_imu_optical_frame | 0.0 | -0.03665191307663917 | 0.005235987715423107 | -0.008726646192371847 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2353595942258834 | -4.718445017209814 | -9.92646201827653 |
| 24 | 1,71225095464689E+018 | 22 | 1,71225095464203E+018 | camera_imu_optical_frame | 0.0 | -0.02094395086169243 | 0.0 | -0.013962633907794952 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2353595942258834 | -4.733292403243464 | -8.951207661665944 |
| 25 | 1,71225095465682E+018 | 23 | 1,71225095464704E+018 | camera_imu_optical_frame | 0.0 | -0.005235987715423107 | 0.001745329238474369 | -0.008726646192371847 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2324956084089124 | -4.748709626635051 | -8.961055909483914 |
| 26 | 1,71225095465683E+018 | 24 | 1,71225095465205E+018 | camera_imu_optical_frame | 0.0 | 0.010471975430846214 | 0.0 | -0.010471975430846214 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2077505676382424 | -4.768505657101131 | -8.857126749536995 |
| 27 | 1,71225095466667E+018 | 25 | 1,71225095465705E+018 | camera_imu_optical_frame | 0.0 | 0.031415924429893494 | 0.0 | -0.006981316953897474 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1843378755678308 | -4.790947571424745 | -8.744609199217727 |
| 28 | 1,71225095466671E+018 | 26 | 1,71225095466206E+018 | camera_imu_optical_frame | 0.0 | 0.043633230030536665 | 0.003490658476948738 | -0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1694901103562109 | -4.83954155126082 | -8.576334300208185 |
| 29 | 1,71225095467664E+018 | 27 | 1,71225095466707E+018 | camera_imu_optical_frame | 0.0 | 0.05061454698443413 | 0.0 | -0.001745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1538905987054792 | -4.867867897618242 | -8.41411577902508 |
| 30 | 1,71225095467664E+018 | 28 | 1,71225095467208E+018 | camera_imu_optical_frame | 0.0 | 0.05235987901687622 | 0.001745329238474369 | -0.001745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1340949534725653 | -4.892612603951348 | -8.285444557393344 |
| 31 | 1,71225095468674E+018 | 29 | 1,71225095467709E+018 | camera_imu_optical_frame | 0.0 | 0.04712389037013054 | 0.001745329238474369 | -0.001745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1185244998914675 | -4.909752869381527 | -8.167765030212168 |
| 32 | 1,71225095468675E+018 | 30 | 1,71225095468211E+018 | camera_imu_optical_frame | 0.0 | 0.036651913076639175 | 0.005235987715423107 | 0.001745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1234737609287252 | -4.889955629270271 | -8.103424302071084 |
| 33 | 1,71225095469666E+018 | 31 | 1,71225095468711E+018 | camera_imu_optical_frame | 0.0 | 0.01745329238474369 | 0.003490658476948738 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1284229312775996 | -4.869222318075932 | -8.061560383240648 |
| 34 | 1,71225095469666E+018 | 32 | 1,71225095469212E+018 | camera_imu_optical_frame | 0.0 | 0.0 | 0.003490658476948738 | 0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1333707094441144 | -4.844483157332984 | -8.115987018041348 |
| 35 | 1,71225095470632E+018 | 33 | 1,71225095469713E+018 | camera_imu_optical_frame | 0.0 | -0.012217304669320583 | 0.008726646192371845 | 0.003490658476948738 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1434460141033662 | -4.820769603804925 | -8.189897479443413 |
| 36 | 1,71225095470633E+018 | 34 | 1,71225095470214E+018 | camera_imu_optical_frame | 0.0 | -0.031415924429893494 | 0.005235987715423107 | 0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1731379917615497 | -4.800974982110694 | -8.33835714215014 |
| 37 | 1,71225095471655E+018 | 35 | 1,71225095470715E+018 | camera_imu_optical_frame | 0.0 | -0.04014257341623306 | 0.003490658476948738 | 0.003490658476948738 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1983655311378948 | -4.784528738931292 | -8.484587592245726 |
| 38 | 1,71225095471651E+018 | 36 | 1,71225095471215E+018 | camera_imu_optical_frame | 0.0 | -0.04363323003053665 | 0.006981316953897476 | 0.001745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2082624021554564 | -4.779580314701194 | -8.623143951912267 |
| 39 | 1,71225095472646E+018 | 37 | 1,71225095471716E+018 | camera_imu_optical_frame | 0.0 | -0.04363323003053665 | 0.005235987715423107 | 0.005235987715423107 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2133191039913656 | -4.77704556636059 | -8.738783346298973 |
| 40 | 1,71225095472649E+018 | 38 | 1,71225095472217E+018 | camera_imu_optical_frame | 0.0 | -0.03665191307663917 | 0.006981316953897476 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2034359815740151 | -4.781993519491624 | -8.78331492447857 |
| 41 | 1,71225095473612E+018 | 39 | 1,71225095472718E+018 | camera_imu_optical_frame | 0.0 | -0.02792526781558990 | 0.003490658476948738 | -0.003490658476948738 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.1974309318662768 | -4.793432649191978 | -8.810986561101833 |
| 42 | 1,71225095473613E+018 | 40 | 1,71225095473219E+018 | camera_imu_optical_frame | 0.0 | -0.012217304669320583 | 0.006981316953897476 | -0.006981316953897474 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2023793730160205 | -4.823123207574723 | -8.791192855513335 |
| 43 | 1,71225095474639E+018 | 41 | 1,71225095473722E+018 | camera_imu_optical_frame | 0.0 | -0.001745329238474369 | 0.005235987715423107 | -0.010471975430846214 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2087179299166907 | -4.844484806060791 | -8.756114591069005 |
| 44 | 1,71225095474639E+018 | 42 | 1,71225095474221E+018 | camera_imu_optical_frame | 0.0 | 0.010471975430846214 | 0.003490658476948738 | -0.012217304669320583 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2186129005302809 | -4.844484806060791 | -8.681901943100419 |
| 45 | 1,71225095475622E+018 | 43 | 1,71225095474722E+018 | camera_imu_optical_frame | 0.0 | 0.02094395086169243 | 0.001745329238474369 | -0.01745329238474369 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2211165977857917 | -4.84152724633910 | -8.597318194309048 |
| 46 | 1,71225095475623E+018 | 44 | 1,71225095475223E+018 | camera_imu_optical_frame | 0.0 | 0.026179939050843811 | 0.0 | -0.019198622554540634 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.2062712949917976 | -4.831630397878912 | -8.488452851246887 |
| 47 | 1,71225095476588E+018 | 45 | 1,71225095475723E+018 | camera_imu_optical_frame | 0.0 | 0.027925267815589905 | -0.001745329238474369 | -0.015709622149467447 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | -0.191424920564813 2 | -4.826440893409232 | -8.382718348551215 |

*Figure 13: '.csv' file obtained from the '.bag' file*

With the linear acceleration and angular velocity obtained in an easily accessible format, it is possible to perform the integration operations explained in the previous paragraphs to determine the position at each instant, and thus the path travelled by the exoskeleton.

IMU INTEGRATION

To achieve this, the initial approach used was a "brute force" method, where I performed direct and exclusive integrations using specific MATLAB functions on the data obtained from the IMU, which are necessary to determine the position:

```
% CSV file upload keeping the original column names.
%readtable permit to load data from the CSV file into a MATLAB table
data = readtable('C:\Users\simio\Downloads\imu_data.csv', 'VariableNamingRule', 'preserve');


% Renames columns to ensure compatibility with MATLAB.
% matlab.lang.makeValidName permit to change column names
% to make them compatible with MATLAB variable naming rules.
% Useful if the original names contain invalid characters, spaces, or are too long.
data.Properties.VariableNames = matlab.lang.makeValidName(data.Properties.VariableNames);

% Finds the column name for timestamps
timestamp_col = data.Properties.VariableNames{contains(data.Properties.VariableNames, 'time')};

% Converting timestamps to Seconds
timestamps = data.(timestamp_col) * 1e-9;

% Sorts data by increasing timestamp
% (useful in case there have been errors in the conversion to the .csv file).
% Sorting data by timestamps is essential for proper time analysis.
% This ensures that data is processed in the correct order and allows for easy integration.
[timestamps, idx] = sort(timestamps);
data_sorted = data(idx, :);

% Extracting the correct data using the column names updated after
% change the name to make them valid
accel_x = data_sorted.('field_linear_acceleration_x');
accel_y = data_sorted.('field_linear_acceleration_y');
accel_z = data_sorted.('field_linear_acceleration_z');
gyro_x = data_sorted.('field_angular_velocity_x');
gyro_y = data_sorted.('field_angular_velocity_y');
gyro_z = data_sorted.('field_angular_velocity_z');

% Acceleration correction along Z-axis.
% The acceleration along the Z-axis is corrected by adding 9.81 m/s²
% to remove the effect of gravity. In the context of an IMU sensor,
% the Z-axis usually measures both the acceleration due to the movement of the device
% and the constant acceleration caused by gravity.
% Since the focus is on achieving only the acceleration due to motion,
% the gravitational acceleration has been balanced.
accel_z = accel_z + 9.81;

% As explained in the previous paragraphs, I obtain the speed by integrating
% Linear Acceleration
velocity_x = cumtrapz(timestamps, accel_x);
velocity_y = cumtrapz(timestamps, accel_y);
velocity_z = cumtrapz(timestamps, accel_z);

% And finally I get the position by integrating the speed first obtained
position_x = cumtrapz(timestamps, velocity_x);
position_y = cumtrapz(timestamps, velocity_y);
position_z = cumtrapz(timestamps, velocity_z);

% 3D route tracking
figure;
plot3(position_x, position_y, position_z, 'LineWidth', 2);
title('Path estimation');
xlabel('Position X (m)');
ylabel('Position Y (m)');
zlabel('Position Z (m)');
grid on;
```

Figure 14 represents the estimated trajectory of the sensor in terms of position along the X and Y axes, obtained through the integration of the measured accelerations. It is evident how the drift has caused a significant error over time in subsequent measurements, as explained later in Chapter 3, where it is detailed that the actual path recorded in the '.bag' file is a regular rectangular trajectory of 3 meters by 1.4 meters.

The first method performs direct integration of linear accelerations to obtain velocity and then integrates velocity to get position. This approach does not account for biases or offsets in the acceleration data, which can affect the accuracy of the position calculation.

Additionally, no operations have been performed on the angular velocities, which could be integrated to obtain the IMU's orientation at each time instant. The orientation is crucial for accurately reconstructing the actual path, as the IMU's linear acceleration data is affected by its orientation in space. Without taking orientation into account, the calculated trajectory may not accurately reflect the real-world movement, since the IMU's frame of reference might change during the motion.

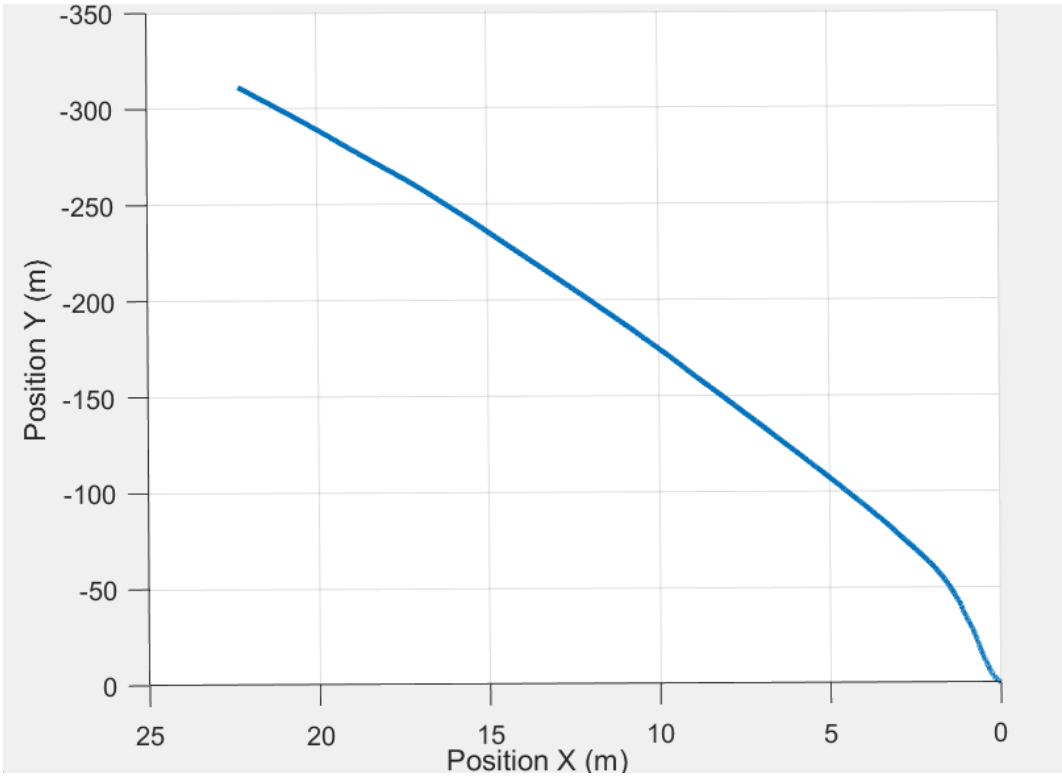



*Figure 14: "brute force" result*

In the second approach, I tried to enhance the accuracy and reliability of position estimation using several methods.

First, I removed the bias from the acceleration data. By calculating and subtracting the mean value of each acceleration component, the systematic errors due to sensor bias are significantly reduced. This allows more precise subsequent integrations of acceleration, yielding more accurate velocity and position estimates.

```
42
43      mean_accel_x = mean(accel_x);
44      mean_accel_y = mean(accel_y);
45      mean_accel_z = mean(accel_z);
46
47      accel_x = accel_x - mean_accel_x;
48      accel_y = accel_y - mean_accel_y;
49      accel_z = (accel_z + 9.81)-mean_accel_z;
50
```

Next, a low-pass filter is applied to the accelerometer data before integration. This filtering process helps to remove high-frequency noise that can distort the measurements and lead to inaccurate results. By focusing on the meaningful movement data and ignoring irrelevant noise, the method achieves smoother and more precise trajectory estimates.

```
51
52      cutoff = 2.5; % Cutoff frequency in Hz
53      accel_x_filt = lowpass(accel_x, cutoff, fs);
54      accel_y_filt = lowpass(accel_y, cutoff, fs);
55      accel_z_filt = lowpass(accel_z, cutoff, fs);
56
```

Another crucial improvement is the implementation of drift correction for the velocity data. Over time, small errors can accumulate during the integration of acceleration, leading to velocity drift. Using MATLAB's *detrend* function, the velocity is corrected to remove this drift, ensuring that the integrated position remains consistent and reliable over time.

```
62
63      velocity_x = detrend(velocity_x, 'constant');
64      velocity_y = detrend(velocity_y, 'constant');
65      velocity_z = detrend(velocity_z, 'constant');
66
```

Figure 15 illustrates the estimated position after applying the second data filtering approach. Unlike the first approach, this estimate accounts for bias errors, providing a more accurate trajectory and reducing the effect of drift. The improvements shown above make the second method more robust and accurate than the initial "brute force" approach, since the trajectory obtained is much more similar to the recorded one. While the first method simply performed direct integrations of raw data, this enhanced version applies critical corrections and filters, laying the groundwork for even more refined methods in the future. Despite its limitations, the first method remains useful as a reference point to compare the impact of these enhancements.

A more detailed analysis will be carried out in the following chapter, where the two methods will be compared.

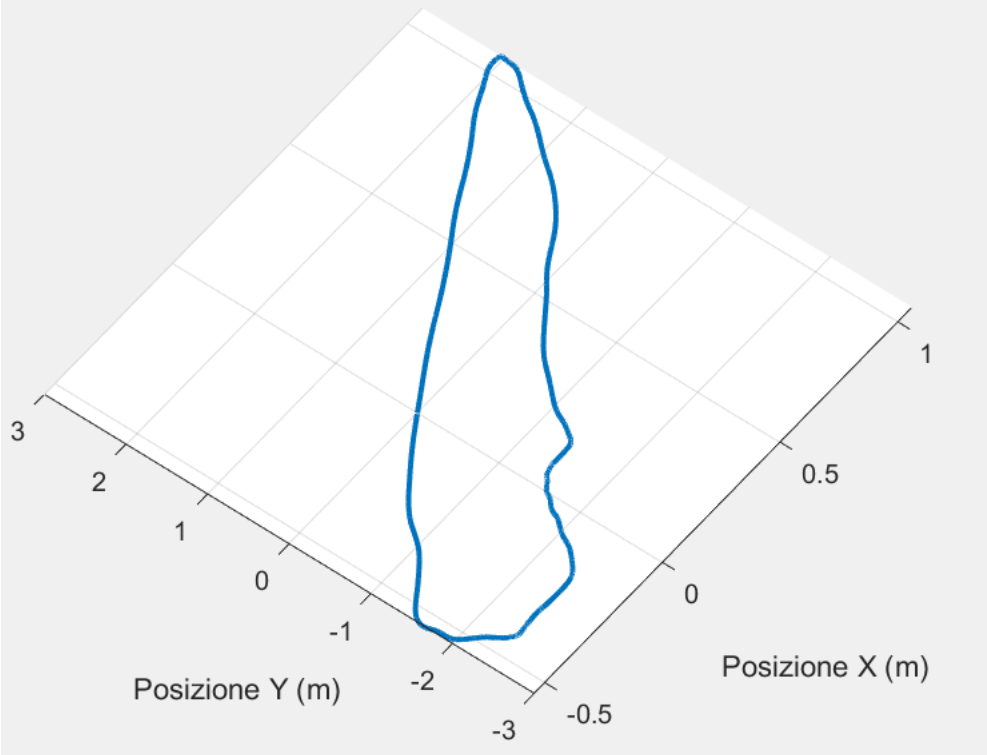



*Figure 15: result obtained using filtered data and bias/drift correction*

# Experiment and Results

## 3.1 Validation of Algorithms/Methods

To validate the results obtained, the provided .bag file contains a recording where a rectangular path of 3m x 1.4m was followed. This known reference path allows for direct comparison between the estimated trajectory from the IMU data and the actual recorded movement. By overlaying the results from the IMU-based calculations with the path from the .bag file, discrepancies can be identified and analysed.

## 3.2 Results

As can be noticed from the graph obtained in the “brute force” method, the direct integration of raw linear acceleration data without the any preprocessing steps, such as filters or bias correction, led to a considerable accumulation of errors over time. This resulted in a significant drift in the estimated trajectory, which can be clearly observed in the plot. The path, in fact, looks exaggerated, reaching unrealistic scales, with the drift making the estimated path completely deviate from the actual movement.

The enhancements in the second approach yielded a trajectory much closer to the real path followed by the exoskeleton. In the image to be considered, the estimated path now roughly follows the expected rectangular shape. While some distortions are still visible, particularly in the overall curvature of the path, the scaling of the position is now much more accurate, with the coordinates more closely matching the real-world dimensions of 3m x 1.4m, the result obtained has in fact dimensions of 3,9m x 1,6m. Despite the improvements, deviations from the exact rectangular path are still visible. These residual errors stem from factors such as IMU sensor configuration and readings and minor noise not eliminated by the filter.

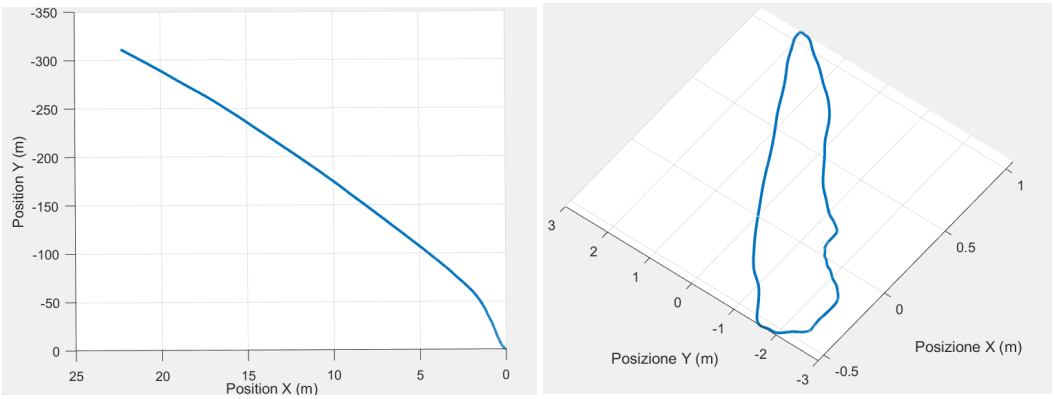



*Figure16: comparison between the two approaches*

The comparison between the two approaches highlights the significant improvement achieved by incorporating bias correction, filtering, and drift correction techniques. In addition, it has been shown that raw data integration alone is not enough for accurate position estimation and how easy it is to incur drift when working with IMU data.

To improve the results obtained from the current IMU-based position estimation, integrating additional sensors such as GPS could be highly beneficial [29]. The IMU alone, while providing detailed information about linear acceleration and angular velocity, is prone to accumulating drift over time due to the integration of noisy sensor data, as shown before.

# Conclusions

## 4.1 Contributions to the work

In this thesis, I presented some contributions to the field of inertial odometry for exoskeleton control. One of the key advancements is the development of an enhanced methodology for integrating data from inertial sensors, specifically accelerometers and gyroscopes, to estimate the position and orientation of a lower limb exoskeleton. By employing techniques such as filtering and drift correction, the approach proposed significantly improves the accuracy of trajectory estimation compared to conventional brute-force integration methods, which are prone to cumulative drift errors. The refined method has led to a more stable and reliable estimation of the exoskeleton's movement, addressing a critical challenge in wearable robotics.

Additionally, this work leverages the Robot Operating System (ROS) framework to facilitate the acquisition, processing, and analysis of IMU data. The use of ROS enables efficient modular development and real-time communication between different components of the system, ensuring seamless integration of the sensor data into the control algorithms. This contributed to a robust system architecture that supports both experimentation and future scalability.

The methodology was experimentally validated using a known rectangular path (3m x 1.4m), providing a clear benchmark for comparing the estimated trajectory against the actual movement. The results demonstrate that the improvements in filtering and drift correction significantly reduced errors and provided a much closer approximation of the real-world path, compared to the initial methods.

Ultimately, this thesis contributes to the advancement of wearable robotics, particularly in the context of lower limb exoskeletons used for rehabilitation and mobility assistance. The improved inertial odometry techniques developed in this work offer a pathway towards more accurate and efficient control systems, which could enhance the performance of exoskeletons in real-world scenarios, improving both user experience and therapeutic outcomes.

## 4.2 Limitations

The most prominent challenge I found is the inherent susceptibility of inertial measurement units (IMUs) to drift over time. Despite the implementation of filtering techniques and drift correction, small errors in the accelerometer and gyroscope data accumulate, particularly over extended periods of use. This limitation impacts the long-term accuracy of the estimated trajectory, as the system lacks external references, such as GPS or visual markers, to further constrain or correct the position.

Additionally, the current method relies solely on data from a single IMU, which limits the precision of the motion estimation. The integration of multiple IMUs or additional sensors could potentially improve the accuracy by providing more data points and enabling more robust sensor fusion techniques. Another limitation is that the system assumes a relatively controlled environment with consistent and predictable movement patterns. Most advanced exoskeletons still fail to provide the real-time adaptability and flexibility presented by humans when confronted with natural environments, in fact more dynamic or unstructured real-world environments, sensor noise, unexpected movements, or external factors could introduce variability that would affect the accuracy of the estimated trajectory.

The computational complexity of the implemented methods, though manageable in a controlled lab setting, may pose a challenge for real-time applications on embedded systems with limited processing power. Optimization of the algorithms may be necessary to ensure scalability and usability in practical, on-device scenarios.

Lastly, it is also important to consider the economic point of view of using an exoskeleton, as they are still very expensive to use and the credibility and acceptability gap in patients is still not high enough.

## 4.3. Future applications

Inertial odometry techniques implemented in this work have the potential to contribute to a wide range of future applications, for example integration of the enhanced odometry system into lower limb exoskeletons designed for rehabilitation and assistance therapies. The system could be used to track the user's movements in real-time, aiding patients going through rehab from spinal cord injuries, strokes or other motor impairments, potentially accelerating the rehabilitation process by adapting to the user’s progress and providing enhanced support for everyday activities.

Additional sensor modalities such as cameras or GPS can be integrated to enhance the accuracy of the system, allowing it to perform reliably in more complex and dynamic environments. This would open possibilities for outdoor use, where environmental factors such as uneven terrain or variable lighting conditions could be better accounted for. Furthermore, as advances in sensor technology and machine learning techniques continue to evolve, there is potential to incorporate adaptive algorithms that refine the control of the exoskeleton in real-time, based on continuous feedback from the user's movements and the surrounding environment.

**BIBLIOGRAPHY**


[1] *State of the art and future directions for lower limb robotic exoskeletons*. (2017, February 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/7393837

[2] *Industrial exoskeletons: What you're not hearing -- Occupational health & Safety*. (2018b, October 1). Occupational Health & Safety. https://ohsonline.com/articles/2018/10/01/industrial-exoskeletons-what-youre-not-hearing.aspx

[3] Roboticsbiz. (2021, October 29). *Active vs. passive exoskeletons explained*. Roboticsbiz. https://roboticsbiz.com/active-vs-passive-exoskeletons-explained/#:~:text=Now%2C%20exoskeletons%20are%20generally%20classified%20as%20active%20and,human%20strength%20or%20reduce%20the%20body%E2%80%99s%20energy%20consumption.

[4] Lee, H., Ferguson, P. W., & Rosen, J. (2020). Lower Limb Exoskeleton Systems—Overview. In *Elsevier eBooks* (pp. 207–229). https://doi.org/10.1016/b978-0-12-814659-0.00011-4

[5] Emken, J. L., Benitez, R., & Reinkensmeyer, D. J. (2007). Human-robot cooperative movement training: Learning a novel sensory motor transformation during walking with robotic assistance-as-needed. *Journal of NeuroEngineering and Rehabilitation*, *4*(1). https://doi.org/10.1186/1743-0003-4-8

[6] Slade, P., Kochenderfer, M. J., Delp, S. L., & Collins, S. H. (2022). Personalizing exoskeleton assistance while walking in the real world. *Nature*, *610*(7931), 277–282. https://doi.org/10.1038/s41586-022-05191-1

[7] Major, R. E., Stallard, J., & Rose, G. K. (1981). The dynamics of walking using the hip guidance orthosis (hgo) with crutches. *Prosthetics and Orthotics International*, *5*(1), 19–22. https://doi.org/10.3109/03093648109146224

[8] Aliman, N., Ramli, R., & Haris, S. M. (2017). Design and development of lower limb exoskeletons: A survey. *Robotics and Autonomous Systems*, *95*, 102–116. https://doi.org/10.1016/j.robot.2017.05.013

[9] Baud, R., Manzoori, A. R., Ijspeert, A., & Bouri, M. (2021). Review of control strategies for lower-limb exoskeletons to assist gait. *Journal of*

*Neuroengineering and Rehabilitation*, *18*(1). https://doi.org/10.1186/s12984-021-00906-3

[10] *GAIT prediction and assist control of lower limb exoskeleton based on inertia measurement unit*. (2022, September 23). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9963096

[11] Tucker, M. R., Olivier, J., Pagel, A., Bleuler, H., Bouri, M., Lambercy, O., Del R Millán, J., Riener, R., Vallery, H., & Gassert, R. (2015*). Control strategies for active lower extremity prosthetics and orthotics: a review.* Journal of Neuroengineering and Rehabilitation, 12(1), 1. https://doi.org/10.1186/1743-0003-12-1

[12] *Reference Joint Trajectories generation of CUHK-EXO exoskeleton for system balance in walking assistance*. (2019). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/document/8667291

[13] *Preliminary evaluation of a powered lower limb orthosis to aid walking in paraplegic individuals*. (2011, December 1). IEEE Journals & Magazine | IEEE Xplore. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6033046

[14] Pino, A., Tovar, N., Barria, P., Baleta, K., Múnera, M., & Cifuentes, C. A. (2021). Brain–Computer interface for controlling Lower-Limb exoskeletons. In *Springer eBooks* (pp. 237–258). https://doi.org/10.1007/978-3-030-79630-3_9

[15] Baud, R., Manzoori, A. R., Ijspeert, A., & Bouri, M. (2021b). Review of control strategies for lower-limb exoskeletons to assist gait. *Journal of NeuroEngineering and Rehabilitation*, *18*(1). https://doi.org/10.1186/s12984-021-00906-3

[16] *This is CYBATHLON*. (n.d.). CYBATHLON ETH Zürich. https://cybathlon.ethz.ch/en/cybathlon

[17] *Alice Open Source Exoskeleton 2021 update*. (n.d.). Hackaday.io. https://hackaday.io/project/181850-alice-open-source-exoskeleton-2021-update

[18] AM Equipment. (2024, August 7). *226 Series DC Gear Motor - AM Equipment*. https://www.amequipment.com/shop/230-series-dc-gear-wiper-motor-4yyt5-9pm67-mhpm3-f3xjn/

[19] Flynt, J. (2017, November 10). *Polylactic acid (PLA): the environment-friendly plastic*. 3D Insider. https://3dinsider.com/what-is-pla/

[20] M. Cardona and C. E. García Cena, "Musculoskeletal modeling as a tool for biomechanical analysis of normal and pathological gait", *VIII Latin American Conference on Biomedical Engineering and XLII National Conference on Biomedical Engineering. CLAIB 2019. IFMBE Proceedings*, vol. 75, pp. 955-963, October 2019.

[21] Arnold, E. M., Ward, S. R., Lieber, R. L., & Delp, S. L. (2009). A model of the lower limb for analysis of human movement. *Annals of Biomedical Engineering*, *38*(2), 269–279. https://doi.org/10.1007/s10439-009-9852-5

[22] Memarian, F., Rahmani, S., Yousefzadeh, M., & Latifi, M. (2019b). Wearable technologies in sportswear. In *Elsevier eBooks* (pp. 123–160). https://doi.org/10.1016/b978-0-08-102582-6.00004-6

[23] Alipio, M., & Villena, M. L. (2023). Intelligent wearable devices and biosensors for monitoring cattle health conditions: A review and classification. *Smart Health*, *27*, 100369. https://doi.org/10.1016/j.smhl.2022.100369

[24] J. Woodman, O. J. W. (2007, August). *An introduction to inertial navigation*. University of Cambridge. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.html

[25] Baker, B. B. & DigiKey's North American Editors. (2018, January 30). *Apply Sensor Fusion to Accelerometers and Gyroscopes*. Digikey. https://www.digikey.it/en/articles/apply-sensor-fusion-to-accelerometers-and-gyroscopes?msockid=135d25c1c5fa6b4414a931cec4916a0b

[26] *ROS/Introduction - ROS Wiki*. (n.d.). https://wiki.ros.org/ROS/Introduction

[27] *ROS/Concepts - ROS Wiki*. (n.d.). https://wiki.ros.org/ROS/Concepts

[28] Intel RealSense. (2024, July 3). *Depth Camera D435i – Intel® RealSenseTM depth and tracking cameras*. Intel® RealSense™ Depth and Tracking Cameras. https://www.intelrealsense.com/depth-camera-d435i/

[29] Jamil, F., Iqbal, N., Ahmad, S., & Kim, D. (2020). Toward accurate position estimation using learning to prediction algorithm in indoor navigation. *Sensors*, *20*(16), 4410. https://doi.org/10.3390/s20164410