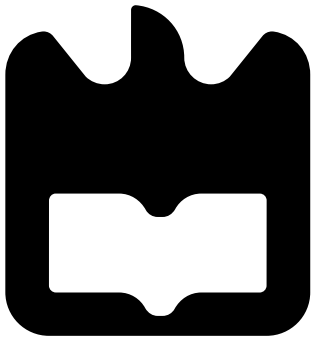
João Carlos Pimentel Fidalgo Peixoto

Integração de Dados Visuais e Inerciais Para o Equilíbrio de Um Robô Humanóide

Visual and Inertial Data Integration to Assist Humanoid Balance



João Carlos Pimentel Fidalgo Peixoto

Integração de Dados Visuais e Inerciais Para o Equilíbrio de Um Robô Humanóide

Visual and Inertial Data Integration to Assist Humanoid Balance

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requesitos necessários á obtenão do grau de Meste em Engenharia Mecânica, realizada sob a orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro e sob co-orientação de Filipe Miguel Teixeira Pereira da Silva, Professor Auxiliar do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor António Manuel Ferreira Mendes Lopes
Professor Auxiliar da Universidade do Porto - Faculdade de Engenharia

Professor Doutor Vítor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

I would like to take this opportunity to acknowledge my family, friends and professors, who stood and stand by me, not only this passing year, but since I started this new adventure known as college. In particular, I thank my parents who have always been supportive, both mentally and financially. My family (blood or not) who encouraged me all this time, as well as my friends. Without them this project wouldn't have come to fruition.

I cannot forget my tutors who celebrated my successes and encouraged me when things weren't going so well. I would especially like to thank Dr. Victor Santos for sharing with me his passion and knowledge about computer programming and always acknowledging both my effort and quality as a beginner programmer. Of course, I would also like to thank my fellow colleagues in LAR, especially Jorge Almeida, who taught me all I know about ROS (even if I still don't know much).

I would like to thank my girlfriend for being my safe haven, for enduring my bad mood and all my hours of work. Without any of these people, I don't think I would have gotten through all of this.

Last but not least, I would like to thank Motofil for providing the FANUC arm robot, without which, this work would have been impossible.

Palavras-chave PHUA, Humanoíde, IMU, Sensores Inerciais, Acelerómetros, Giroscópios, Visão, SIFT, SURF, Filtro de Kalman, Equilíbrio

Resumo Esta dissertação aborda o problema que consiste na medição do movimento da cabeça de um robot humanóide fundindo dados inerciais e visuais, com o objetivo de obter o output que melhor descreve o movimento da cabeça do humanóide. O seu principal objectivo é perceber e desenvolver um algoritmo usando o Filtro de Kalman, que irá fundir ambas as fontes de dados com o propósito de obter uma nova fonte de informação com um maior grau de confiança. Para cumprir os objectivos, um modelo da cabeça do humanóide, juntamente com as câmaras e os sensores inerciais, vão ser movidos na ponta de um braço robótico industrial, que é usado como grupo de controle (ground truth) no que toca à posição angular. Pontos-chave nos frames obtidos através da câmara, são extraídos e usados para calcular a diferença na posição angular que ocorreu entre frames, que vão mais tarde, juntamente com os dados inerciais obtidos de giroscópios, servir de input a um modelo de um Filtro de Kalman.

Uma vez que este dissertação assenta em ferramentas como o Filtro de Kalman, que tem como propósito unir dados de origens diferentes, é essencial que se conheçam os tipos de dados e ferramentas que irão ser utilizados. Assim, várias experiências foram desenvolvidas e estudadas com o intuito de desenvolver o conhecimento nessas matérias. Adicionalmente, erros foram acrescentados aos dados, artificialmente, com o objectivo de emular sensores sensíveis a ruído. No entanto, o sistema continua a ter uma performance positiva.

Keywords PHUA, Humanoid, IMU, Inertial Sensors, Accelerometers, Gyroscopes, Vision, SIFT, SURF, Kalman Filter, Balance

Abstract

This thesis addresses the problem of measuring a humanoid robot head motion by merging inertial and visual data, in order to obtain an output that will describe the head motion of the robot. Its primary goal is the understanding and development of an algorithm using the Kalman Filter tool, which will merge inertial and visual data, resulting in a more reliable source of information. To accomplish this, a model of a humanoid robot head, including a camera and inertial sensors, are moved on the tip of an industrial robot's arm which is used as ground truth for angular position. Visual features are extracted from the camera images and used to calculate angular displacement and velocity of the camera, which is then merged with angular velocities from a gyroscope and fed into a Kalman Filter, in order to obtain an output.

Since this thesis is expected to merge two different kinds of data using the Kalman Filter tool, the need to understand both types of data arises, as well as the way the Kalman Filter operates. Therefore, many experiments were developed and studied with the intent of deepening the knowledge on those matters. The results are quite interesting. Additionally, errors are introduced artificially into the data to emulate noisy sensors, and the system still performs very well.

Contents

List of Figures

List of Tables

Acronyms

AMPM	Angular Momentum induced inverted Pendulum Model
COG	Center of Gravity
CoM	Center of Mass
CoP	Center of Pressure
DEM	Departamento de Engenharia Mecânica
DETI	Departamento de Electrónica, Telecomunicações, e Informática
DoG	Difference of Gaussian
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
FRI	Foot Rotation Indicator
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KF	Kalman Filter
LAR	Laboratório de Automação e Robótica
PHUA	Projeto Humanóide da Universidade de Aveiro
ROS	Robot Operating System
ZMP	Zero Momentum Point

Chapter 1

Introduction

"Building a robot that has legs and walks around is a very expensive proposition. Mother Nature has created many wonderful things but one thing we do have that nature doesn't is the wheel, a continuous rotating joint, and tracks, so we need to make use of inventions to make things simpler". [1]

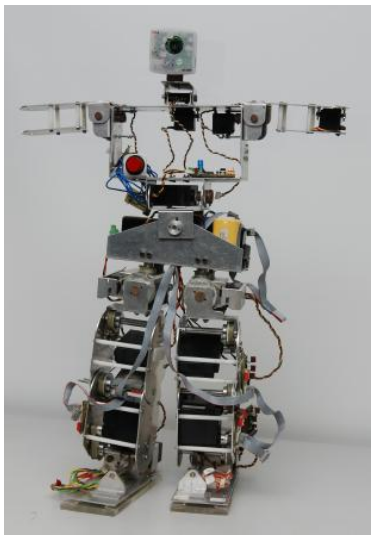
Colin Angle, the CEO and founder of iRobots, considered by many the pioneer of the home robot market, claim that it is impractical and very expensive to build a robot with legs. Not only because of the legs themselves, but also because of all the dynamics that are implied in the balance of the robot. He says that, if we have wheels, there is no need to use legs for locomotion, unless it is for entertainment.

It is true that, if wheels can be used instead of legs, they should be used. However, there are cases where the understanding of locomotion, using legs, can be deemed as important. If a humanoid robot exists, which behaves very similar to the human being, then it is possible to learn something from them. Even if that doesn't happen, all the knowledge associated with the creation of legs for robots, though it may be obsolete for robot locomotion, may have other fields of application. Looking at robotic legs as the future for the disabled people, or their application in the military market, may bring the opportunity for a hugely profitable situation. All that is needed to do before that, is walking the path to knowledge and understanding.

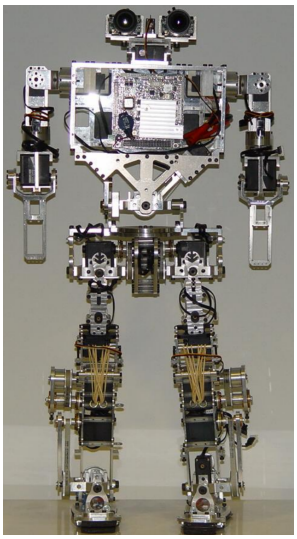
1.1 Background/Motivation

With the intention of creating a low cost platform for students to learn new skills in the area of programming and robotics, the Department of Mechanical Engineering (DEM) and the Department of Electronics, Telecommunications and Informatics (DETI) started on LAR (Automation and Robotics Laboratory) the Humanoid Project of the University of Aveiro (PHUA). PHUA has begun in 2004 [2] with its first design (Figure 1.1a). Over the years, many students worked on this project, ending up with a 6Kg, 667mm and 27 DOF (Degrees of Freedom) (Figure 1.1b) humanoid platform. These 27 DOF are distributed by the robot: 2 in the feet, 12 in the legs, 3 in the trunk, 8 in the arms and 2 in the neck.

The humanoid has eight pressure sensors under its feet [3] (four in each) and a firewire camera placed on its head. Regarding the matter of learning by demonstration, the PHUA has a haptic interface using force feedback for teleoperation [4]. There is also a built in IMU (Inertial Measurement Unit) platform consisting of accelerometers, gyroscopes and magnetometers [5]. This project hopes to contribute by combining the firewire camera information



(a) First Humanoid Platform [6]



(b) Latest Humanoid Platform [7]

Figure 1.1: PHUA Platforms

with the IMU data.

1.2 Problem Description

Concerning the problem, there is a humanoid robot that doesn't walk. Before "learning" how to walk, it needs to "learn" how to balance. Thus, the necessity to develop an algorithm that would give the robot the ability to balance itself. In order to achieve this goal, there are numerous ways to proceed, each of them having their own difficulties. Essentially, this problem can be approached from two perspectives, bottom-up or top-down. The first one relies on COP [8] (center of pressure), where load cells are placed on the feet, in order to understand where the pressure on the robot is mainly at (in space). For instance, if the left foot of the robot applies more pressure than the right one, there is a high probability that the robot is falling to its left. This method has already been implemented [3]. However, the robot does not yet have the ability to balance itself. The approach used in this dissertation, will be the top-down perspective. The idea is to use the head to understand how the robot is oriented in space. Having the orientation of the robot, understanding how it is relative to the environment and correcting its position will become a possibility. In order to complete this approach, inertial and visual data will be used and merged together to better understand the dynamics of motion of the robot. The main problems encountered were essentially the accuracy of the inertial data and its processing, as well as the feature recognition in the visual data and consequent accuracy. Another problem presented is the ability to convert the features recognized into motion descriptors, so they can be merged with the data obtained from the sensors. In the end, it will be possible to look at a feature and track its location in the image. Comparing the location of matched features in different frames with the data obtained from the sensors and merging them into a dynamic equation of the robots movement, will describe how the robot behaves in real-time and how to send it commands, in order for

it not to fall.

1.3 Objectives

Given all that has been said so far, the main objective of this thesis is to be able to merge different sets of data in order to obtain a more accurate description of the robot angular motion, namely in the head. For that, there is a need to put the sensors to the test and see how reliable they are. After this, there is a need to understand what kind of features are possible to recognize using vision and how they can be incorporated in the project. After the two aforementioned tasks are completed, it is needed to know how to integrate them in a single system. For that, the Kalman Filter tool will be used. The idea is to use both data in a single system of variables, so that they can filter the data of each other. What is meant with this is that the inertial and visual data will be used in order to clearly know what is the behavior of the robot's head and its orientation regarding gravity.

1.4 State of the Art

1.4.1 Humanoid Balance

In order to better understand humanoid balance, several researchers conducted experiments in which external forces were applied to humans and their response observed [9–11]. According to Horak et al. [10], it is useful to disrupt the equilibrium and record the behavioral reaction to this disruption. The concept of postural strategies emerged from these studies, as investigators tried to describe the solutions presented by the human body, in terms of movement patterns, joint torques and contact forces. Two distinct postural responses strategies were predicted [12] and described [13], the *ankle strategy* and the *hip strategy*. Later, a *stepping strategy* was also predicted [14–16]. The *ankle strategy* is normally used when the subject suffers a perturbation in its CoM (center of mass), on a flat support area. This perturbation is relatively weak and is mostly absorbed by ankle joint, resulting in an unchanged upper body. It is ideal for tasks that require upright posture and makes the humanoid resemble a single-segment inverted pendulum. The *hip strategy* is used in the presence of a larger perturbation. The hip and knee joints are used, and the whole body helps absorbing the impact. The subject resembles a double-segment inverted pendulum divided at the hip. This strategy is best applied when a rapid shift in the CoM is required. The *stepping strategy* is used for perturbations that are to strong. It will result in the body stepping out one or more steps and is characterized by asymmetrical loading and unloading of the legs, to move the base of support under the falling CoM. Notice that in this case, since moving the CoM is not an option, the support area (feet) try to move under it, in order to regain balance. This is mainly used when the perturbations are so large, that balancing in-place is not possible.

Kuo [11] claims that standing human subjects, when instructed to not move their feet, will use movements in the sagital plane in order to regain balance. For small disturbances [17] the subjects tend to use the *ankle strategy* and for larger disturbances (which place their center of mass near the perimeter of the foot support) *hip strategy* is used. These observation led Kuo to conclude that a balance algorithm couldn't be based on a linear model, since the responses for large perturbations are not amplified responses to small disturbances [13], but rather a completely different movement strategy. The greater the perturbation the more reliance in the

hip strategy, which also is more effective in stabilizing the center of mass. Allum et. al. [17], postulated that the desire to stabilize the head, may be a decisive factor when choosing the proper movement strategy. In addiction, McCollum and Leen [18] concluded that the *ankle strategy* (which behaves like a two-segment inverted pendulum) provides greater chance of stability than the *ankle strategy* (which behaves like a one-segment inverted pendulum) when facing transmission delays, since the first has a longer time constant.

1.4.2 Perception

For these strategies to work, balance measures need to be found. In Robotics, there are several balance measures that can be used, for example, Zero Momentum Point (ZMP) [19] or Foot Rotation Indicator (FRI) [20]. ZMP is defined [22, 23] as the point in which the effects of all forces may be replaced by a unique force, and at which all the angular moments are equal to zero. In other words, ZMP is the point in which, the ground reaction force occurs, when considering all the forces acting on the humanoid. ZMP may also be known as CoP (center of pressure). This measure is widely used in humanoid projects, including the highly successful Honda Asimo [21]. It has been discovered that the location of the CoP is roughly proportional to the magnitude of the torque at the ankle and in order to aid in balance, it must be away from the edges of the foot, otherwise the foot will begin to rotate. Center of pressure serves as a useful ground reference point as it has physical meaning and can be used as a measure of stability.

The FRI is an indicator of postural instability. Opposed to ZMP, FRI may leave the support area, and when it does, its location indicates the direction and magnitude of imminent angular forces, acting on the foot. It can be defined as the point where the ground reaction would have to act to keep the foot stationary. If FRI remains inside the support area, then the humanoid is not suffering from instability, regardless of the position of its CoM, and the farthest way this point is from its support area, the greater the angular momentum felt in the foot and therefore, the greater the instability. This point may be used in order to plan a strategy for the robot to act, since the location of the same reveals the overall dynamics the mechanism suffered. Notice that ZMP is where the ground reaction force acts, regardless of the state of stability of the robots, which is why it is always located inside the support area of the foot (a reaction exists between the foot and the ground). It is important to notice that, even though FRI may leave the supported area, it is related to the foot rotation phenomenon, and therefore, can only be applied during the single support phase of a humanoid.

Goswami & Kallem [24], introduced a paper where a new balance measure was proposed, with the objective of generalizing previous concepts such as the ones mentioned above. The paper studied the mechanics of rotation stability, and was focused on $\dot{H}_G$, which is the rate of change of centroidal angular momentum of a robot. According to this article, the stability information is contained within this criteria. This paper is based upon the fundamental principle that the resultant external moment on a system, computed at its CoM, is equal to the rate of change of its centroidal angular momentum ($\dot{H}_G$) [25]. Goswami & Kallem concluded that the criteria proposed could be sucessfully used for unbalance measure and that lost of balance implies $\dot{H}_G \neq 0$

Feedback methods using the Angular Momentum inducing inverted Pendulum Model (AMPM), have been proposed and proven successful. According to Komura et. al. [26], AMPM has the ability to calculate the angular momentum created by the ground reaction force in real-time. Thus, allowing the calculation of the counteracting motion needed, in

real-time as well. Presented in the article as well, is a new criteria, which can be applied to humanoids during gait (i.e., walking), called *distortion of inertia*, which is based on the difference of the moment of inertia between the current posture and the expected posture (during gait). Using this criteria, will allow to estimate in real time the motion needed to counteract external perturbations and then gradually moving the humanoid to the original gait motion. In a way, this criteria operates such as a feedback.

1.4.3 Merging Data

In order to obtain the balance measures, sensors are needed. Inertial (only) and visual (only) data are widely used, however, there are situation in which, using one type of data only, may cause the cost of acquisition of the sensors to be expensive, a different methodology was tried, which consist in the fusion of different sources of data in order to obtain the same reliability at a lower cost (or higher performance at the same cost). A popular fusion consist in inertial-visual data merging, since the quality of inertial data is affected by temperature, gravity, drifts and biases, none of which affect the imaging data. Several works have been using this approach [27, 28], implementing a Kalman Filter [29] and Extended Kalman Fitler [30] in order to obtain the fusion. The measurements are used as inputs for the Kalman Filter tool, in which they may be analyzed all at the same time affecting the entire state variable matrix, or in other cases, the data is analyzed individually and only the state variables that depend on the source of data are updated.

Jones and Soatto [27], presented a detailed article, in which they presented a model to estimate motion from monocular visual and inertial measurements, as well as an integrated approach to *"loop-closure"*, which consist in the recognition of previously-seen locations. In order to fuse both types of data, equations that described how the different kinds of measures related between each other, as well with the dynamics of the system were created. These equations were later fed into an EKF (Extended Kalman Filter).

According to Liu, C. & Prior, S. [31], almost all the visual-inertial fusion algorithms rely on nonlinear Kalman Filter. The paper tries to merge the accelerometer data with the Kalman Filter as well, using however, two similar models (one for each source type). The equations proposed for the process model describes the laws of motion for position and velocity, where it is assumed that the acceleration is constant and equal to zero. However, in the velocity equation, it is subtracted the magnitude of the sensor measurement error, times the the error in direction (obtained from comparing the current source of data, with the opposing source). The state variable matrix will only update the values that depend of the analyzed data.

Other approach exist. Furgale, P., Barfoot, T. & Sibley, G. presented a paper [32], in which a continuous-time approach was proposed for dealing with high-rate sensors, like IMU units. The paper proposes to use basis functions as a state representation, opposing the usage of discrete data. Not only this method may allow a smother trajectory with fewer variables, when comparing to time discrete data, but at the same time it facilitates the fusion of different kinds of sensors. Since the state representation is continuous, the measurements just need to be inserted in it. The paper evaluated the approach with the problem of determining the rotation and translation between an IMU and camera that are rigidly mounted to a sensor head. The results validated this approach, taking around 26 seconds to test a dataset corresponding to 140 seconds, which implies its promise regarding realtime experiments.

Chapter 2

Experimental Tools and Setup

Since this work is based partially on inertial sensors, it is mandatory to understand some concepts, namely what they are, how they work and most of all, what they can measure. The sensors being used, described on pag. 8, have nine DOF (Degrees of Freedom) which represent three sets of measurements on three different axis. In order to create and run the experiments, the idea of using the humanoid was discarded, since it lacked a reliable ground truth or even the ability to repeat experiments. To overcome this problem, a FANUC robot (pag. 11) was used, which allowed the development of experimental trials that could be repeated and measured, obtaining a reliable ground truth in the process. It will also be presented other tools used during the elaboration of this dissertation such as ROS (pag. 10), which allowed the development of software.

2.1 INS

In order to control and monitor the orientation and trajectory of a given object, a system is used, which consists of a set of inertial devices that are able to quantify linear and angular motion. This system is known as INS [33] (Inertial Navigation System) and includes accelerometers, gyroscopes and magnetometers.

Nowadays, this kind of devices is used in a variety of areas, including but not limited to, military use, naval and aerospacial navigation, videogames augmentation, among others. Thanks to this, the devices have undergone an extensive evolution, decreasing their size and price, making them accessible to the common person. This is why they are being widely used in many different projects and areas.

In INS system there are some variables that can be measured directly, such as linear acceleration, angular velocity and magnetic field, while others can be measured indirectly, integrating the values obtained like linear velocity, linear displacement and angular displacement.

2.1.1 Accelerometers

The main goal of the accelerometer is to measure the forces applied to it. It is not possible to measure the acceleration of a body in free fall using an accelerometer, since there is no sense of weight. However, if looking at the measures of a resting accelerometer, it is observed, a constant measure of acceleration due to the gravity (weight) felt by the sensor. (This effect

was widely observed during the experimental phase of this thesis.)

The accelerometers are profusely used, namely in positioning systems, inclinometers, vibration sensors, among others. The most common example is its utilization in smartphones, which have the ability to adjust the display accordingly to its orientation. These sensors are based in the second law of Newton.

$$F = m.a \qquad [\text{N}] \quad (2.1)$$

An accelerometer converts the felt force into a measurable quantity (voltage). An accelerometer doesn't measure (directly) the acceleration, but rather the force applied to it. This is why the acceleration of a body in free fall cannot measured and why the acceleration is measured in resting sensors.

2.1.2 Gyroscopes

The main goal of the gyroscope is to measure angular velocity (in one or more axis). This device is composed by a suspense rotor in a support formed by two articulated circles with Cardan joints. It consists, essentially, of a free wheel (or more), which can turn in any direction and which opposes any attempt to change the original direction of its axis. (When we spin a bicycle wheel, while in the air, it offers resistance when we try to change its axis direction.)

Thanks to this effect, the gyroscope can be used as a reference for orientation, but not to position, i.e., a gyroscope cannot detect linear movement. For this reason, it's optimal to use both accelerometers and gyroscopes, since the first is able to detect linear movement and the second angular movement. The most common use of this type of device can be observed in the areal navigation. However, some companies (Nintendo, Sony and Microsoft) have begun to use this kind of sensor in order to increase the videogame experience, converting the user's movements into commands.

2.1.3 Magnetometers

The main objective of this sensor is to measure the magnitude and direction of magnetic fields. Normally they are used in geophysics studies related with the Earths magnetic field. In the inertial sensors used in this project there are magnetometers. However due to the magnetic fields created by the electric motors, their measures are useless.

2.1.4 Sensors Used

PHUA has two distinct IMU models: one "RAZOR 9DOF - SEN 10736" and eight "POLOLU - MiniMU9DOF v2", making a total of nine sensors.

RAZOR 9DOF - SEN 10736 [35]:

- 3DOF accelerometer - ADXL345:
  - operating until ± 16g;
  - 13 bit resolution;
  - operating frequency of 3.2 kHz.
- 3DOF gyroscope - ITG3200:
  - operating until ± 2000/s;
  - 16 bit resolution;
  - operating frequency of 30 kHz.
- 3DOF magnetometer - HMC5883L:
  - operating until ± 8 gauss;
  - 12 bit resolution;
  - operating frequency of 400 kHz.
- ATmega328P incorporated micro-controller;
- RS232 communication.

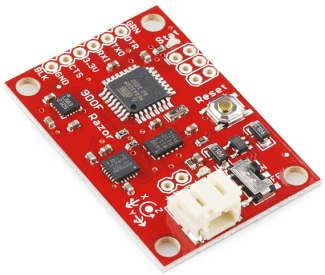


Figure 2.1: RAZOR 9DOF - SEN10736.

POLOLU - MinIMU9DOF v2 [36]:

- 3DOF accelerometer -LSM303DLHC:
  - operating until ± 16g;
  - 12 bit resolution;
  - operating frequency of 400 kHz.
- 3DOF gyroscope - L3GD20:
  - operating until ± 2000/s;
  - 16 bit resolution;
  - operating frequency of 0.76 kHz.
- 3DOF magnetometer - LSM303DLHC:
  - operating until ± 8 gauss;
  - 12 bit resolution;
  - operating frequency of 400 kHz.
- I^2C communication.

Figure 2.2: POLOLU - MinIMU9DOF v2.

2.2 ROS

ROS [37] (Robotic Operating System) is a tool designed for software development. This evolution is associated with its modular structure, which allows the division of a complex project into smaller and simpler packages. Each package can contain one or more executables, known as nodes. If it is needed to launch more than one node to receive some kind of useful output, it is possible to create a launch file, which will launch the nodes needed, with the arguments defined by the user. ROS is an open source tool, so many useful samples and packages created by third parties are already available for download.

ROS is programmed in either C++ or Python. In this project, the language chosen was C++. ROS has tools for processing and compiling code and uses makefiles to generate the executables, being this another advantage of ROS.

2.2.1 Topics and ROSBag

Another great advantage of ROS is the existence of topics. Topics allow easy communication between two nodes. The node that will send the message needs only to publish in a given topic (even if there is no one "listening"). The node(s) that need to receive messages, only have to subscribe to the corresponding topic in order to "listen" to the publisher. In ROS there is a large amount of topic types, ranging from strings to numerical values (integers, floats, ...) or even images. It is possible to create our own kind of topic using ROS tools. Every time some information is published in a topic, the subscribed node(s) will incur into a predefined callback and execute a given function. This way, it is possible for the listeners to operate in a *blank* while cycle, being interrupted when new information arrives.

Let's say that a node (listener), which receives strings from another node (publisher), needs to print the strings in the terminal. In this case, the main body of the listener is an infinite while cycle which has nothing in it. However, this listener has a callback that is called every time a new information arrives at the topic. It is in this callback that the *printf* function is located. After executing the callback, the listener goes back to his infinite while cycle.

There is also a useful tool in ROS called ROSBag, which allows the interception of messages sent by nodes into topics and respective recording in a *".bag"* file. This way it is possible to re-utilize this data in the future. Among other informations, ROSBag records the time in nanoseconds. It is possible to use the *".bag"* file again to simulate a node or convert it to a *".csv"* file, so it can be read by matlab or excel.

Using the previous example, let's say that the strings sent by the publisher sends, contain the temperature values in a room, taken every thirty seconds. It is highly unlikely that in our lifetime we will ever record those exact temperatures again (except in a controlled environment). If it is needed to simulate what happened on the day that the publisher recorded the data, it is possible, using the *".bag"* file.

2.2.2 Useful Codes in ROS

In table 2.1 it can be found some of the most useful codes and commands for ROS.

Code	Description
catkin_init_workspace	begins a workspace (where we create packages)
catkin_make	compiles the workspace
rospack find *package*	finds the path of a package
roscd *package*	navigates to the package
catkin_create_pkg *package*	creates a package
roscore	initializes ROS environment
rosnode list	lists the executing nodes
rosrun *package node*	runs the node in the package
roslaunch *package launch*	launches the launch file of a package
rostopic list	lists the topics in execution

Table 2.1: ROS Commands.

2.3 Industrial Manipulator FANUC

The FANUC manipulator is a robotic arm used in the industry, due to its easy of program and for its application in the most various tasks. The FANUC 200iB [38] has a 6 DOF, having six rotation joints. The biggest advantage of this robot is its precision. Due to this fact we are able to perform and reproduce testing trials with high repeatability rates, as well as acquiring extremely reliable data from its end-effector. The industrial manipulator is able to return, among other things, its position (in either world coordinates or joint coordinates) and orientation of its end-effector. With this data it is possible to compare the output obtained from the industrial manipulator and what the sensors are measuring. Thus, obtaining a highly reliable ground truth, that will serve as base for the future experiments. The experimental setup is as shown in the Figure 2.3 and Figure 2.4. A piece was designed in order to aggregate all the components into a single platform (Appendix B). Therefore, even if only one inertial sensor is needed for this dissertation, the possibility of adding more sensors for future experiments exists.

Furthermore, for easier communication between the computer and the industrial manipulator, a ROS node was created, which is called *fanuc_control*. Essentially, the purpose of this node is to send instructions to the robot and receive answers from it as well. The *fanuc_control* node uses TCP/IP communication in order to exchange messages with a program within the industrial manipulator called ROBCOMM [39]. Currently, the *fanuc_control* can send some messages to the FANUC robot. However, they need to be compiled beforehand. This node was used throughout the work in order to run trials and receive information related to the industrial manipulator end-effector, which later on served as ground truth.

The inputs and outputs of the node are as seen in Figure 2.5. As can be seen, the node receives inputs from both the inertial and visual sensors. After receiving them, they are published again. The reason for this was to format "*/topic_raw_data*" and "*/topic_filtered_imu*" in order to be compatible with *.csv* files. The reason for the node to subscribe to "*/camera/image_rect_color*", is to be able to publish the timestamps regarding the frames, which will be useful information later.

In Figure 2.6, it can be seen a basic grafcet of the node behavior. First, the node initializes, the ROS environment and then the publishers (in order to output data). After this, the node constructs a class called "*dataReceiver*". This class is responsible for the conversion

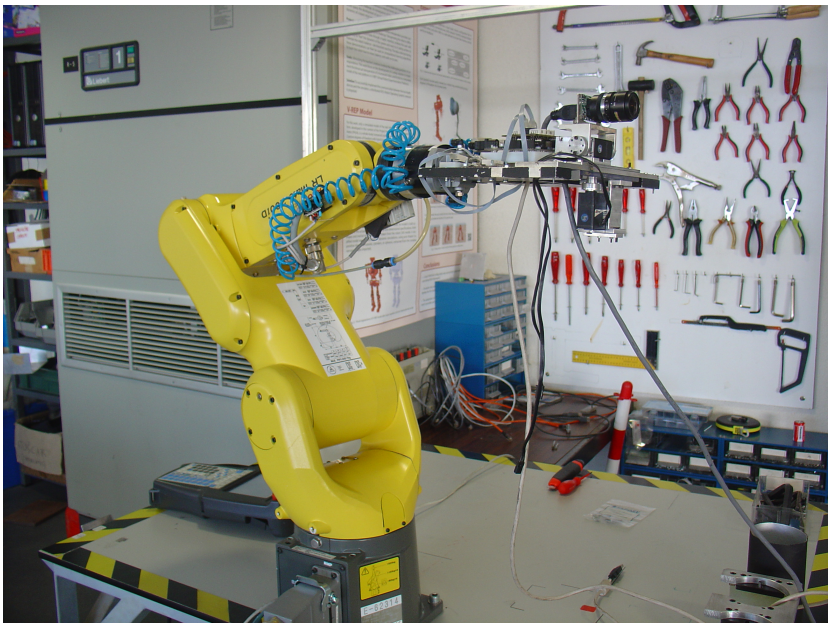
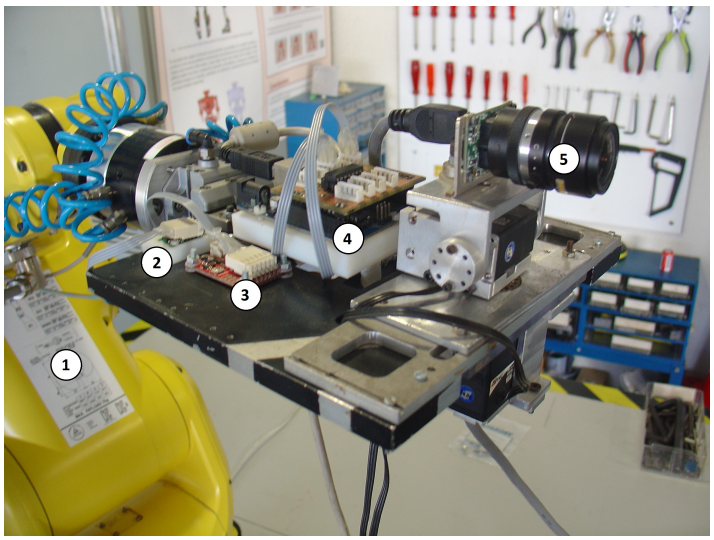


Figure 2.3: Experimental Setup.



1. FANUC manipulator
2. POLULU-minIMU 9DOF v2
3. RAZOR 9DOF - SEN 10736
4. Arduino UNO R3
5. Firefly MV-03MTC - Point-grey

Figure 2.4: Experimental Setup Detailed View.

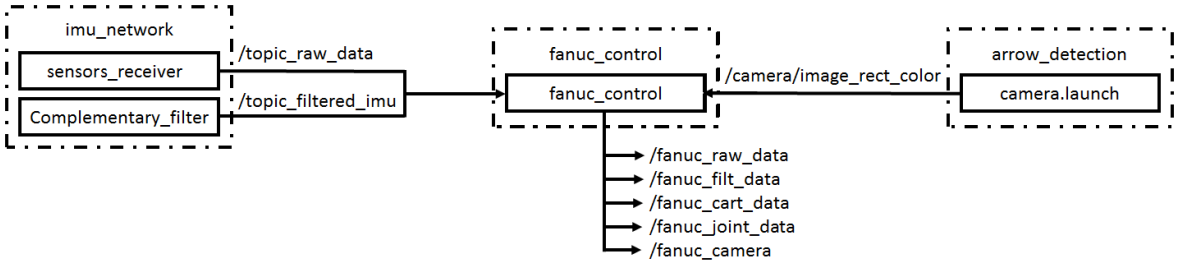


Figure 2.5: "*fanuc_control*" Inputs and Outputs.

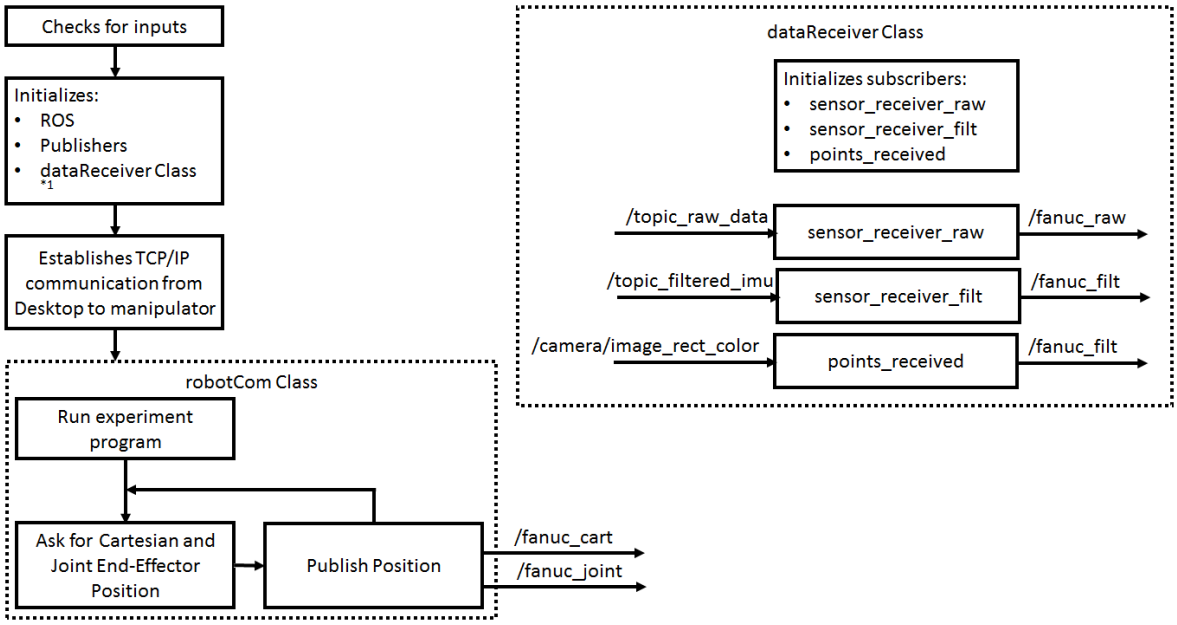


Figure 2.6: "*fanuc_control*" simple block diagram.

of the inertial data into a *.csv* format and the recording of the frames timestamps. This class is based on interrupts, so each time that new data arrives, the callback is activated and the function is executed. After initializing "*dataReceiver*", a TCP/IP connection will be established between the Desktop and the industrial manipulator. After, the node constructs the class "*robotCom*", which will be responsible for the Desktop-manipulator communication. This class will command the experiment program, present in the manipulator console, to run, while recording and publishing, the end-effector cartesian and joint coordinates. Currently, the fanuc_control node has a help menu and a debug system in order to help future development.

2.4 inertial_correct_cam

In order to facilitate the recognition of features in the real world, a package in ROS called *"inertial_correct_cam"* was developed, which has a node with the same name. The node evaluates the measurements obtained from an accelerometer (which is attached to the camera) and rotates an image according to the received data. This node has a class named *"rotateImage"*, which has three functions inside. Two of these function are callbacks for ROS topics (subscribers) and are called *"sensor_receiver"*, which receives the data from the topic *"topic_raw_data"* from the node *"sensors_receiver"* from the package *"imu_network"* and *"rotate_frame"*, which rotates a frame obtained from the topic *"/camera/image_rect_color"*, which is created after the launch file from the package *"arrow_detection"* is initialized. The third function is called *"calculate_angles"*, which calculates the orientation of the sensor using the accelerometers. The node consists of a script which initializes this class and then enters in a while cycle, waiting to receive new data from the topics [*ros::spin()*].

Figure 2.7 is a simple block diagram of this script. After the initialization of *rotateImage*, the script waits for inertial and visual data. When visual data arrives into *rotate_frame*, it

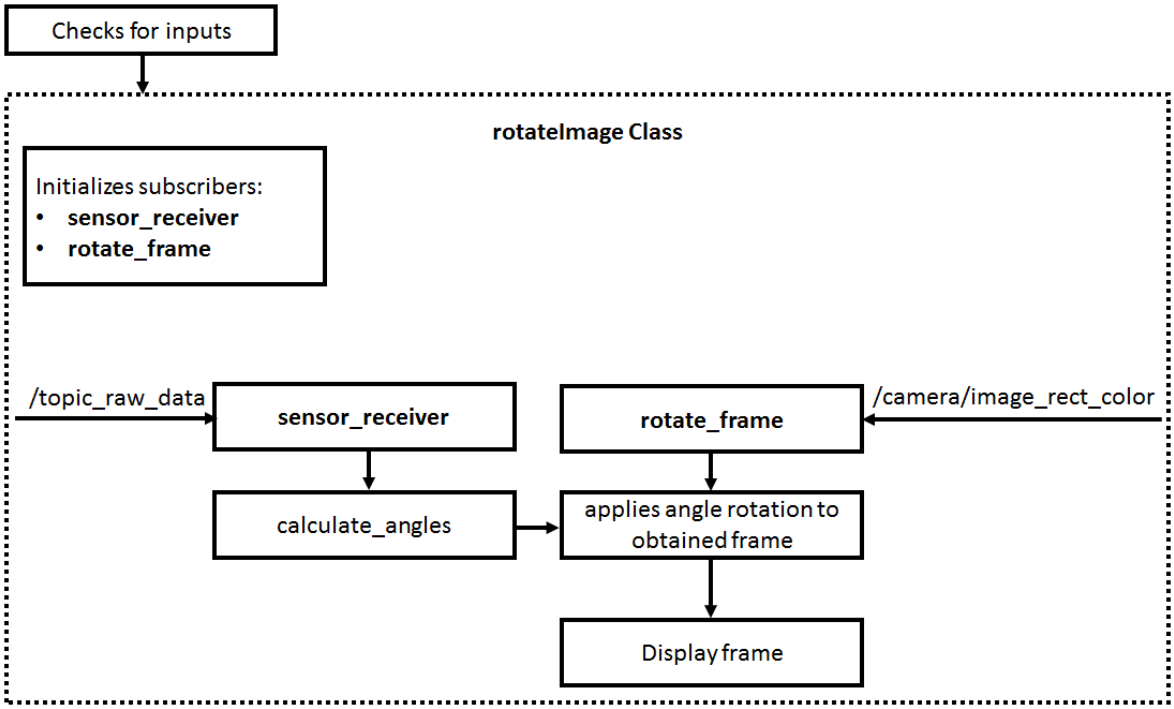


Figure 2.7: *"inertial_correct_cam"* simple block diagram.

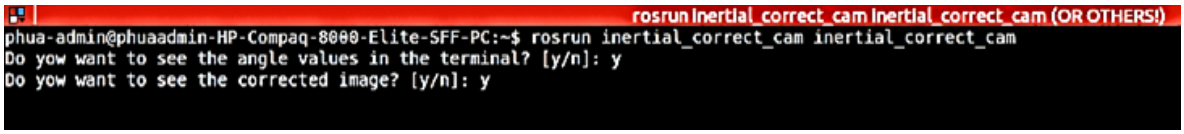


Figure 2.8: When inertial_correct_cam is ran it asks the user if he wants to see the angle values and the output image or not.

is rotated by a value, set by *sensor_receiveir()*, which has received the accelerometers data, and transformed it into an angle value, with the help of *calculate_angles()*. This last function, calculates the angle based on the accelerometers, as described in page 28.

It can be seen, in Figure 2.9 that the sensor seems to always be aligned with the gravity vector, since the image rotates with the same angle that the sensor does. Figure 2.10 was obtained after attaching the sensor to the back of the camera. It can be seen that the image obtained from the camera appears to be normal to the gravity vector as well, i.e., even thought the camera was at about 45°angle with the floor, the image was corrected to be presented as if the camera was making a 0°angle with it. In theory, this will diminish the difficulty of feature recognition.

As mentioned, this idea was one of the first that came to mind and it seemed that it was actually merging two different sets of data (the inertial data was used to correct the visual data). However, this script came to be unnecessary, as other ideas replaced this obsolete one (at that time, there was no awareness of Scale and Rotation Invariant Features, which will be talked about later). Notice that even though this concept seems to improve the output of the visual data, it does not, making it fact more difficult to use, since now it can't be known for sure how the visual data describes the real world. Furthermore, no information

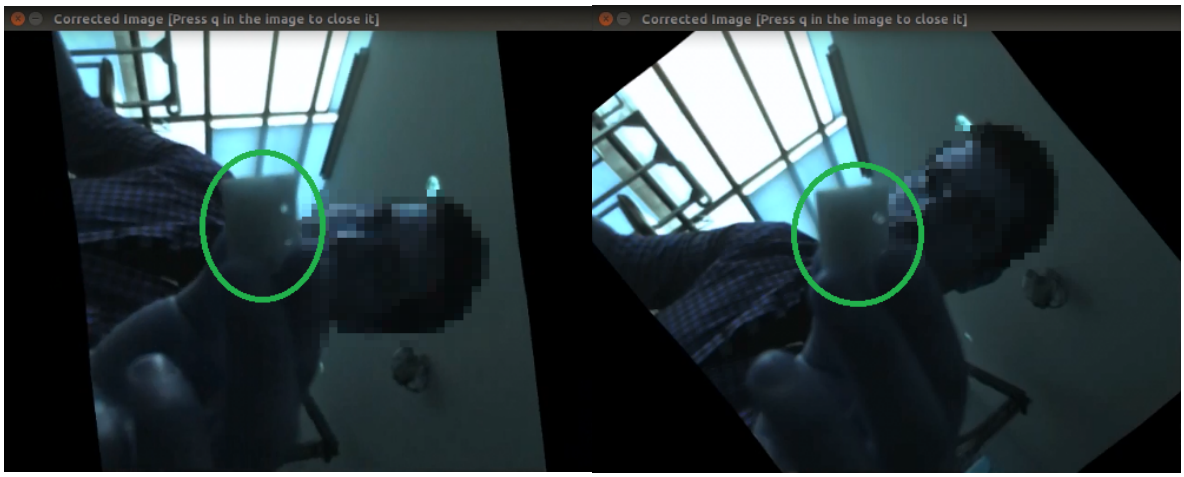


Figure 2.9: Output from *"inertial_correct_cam"*.

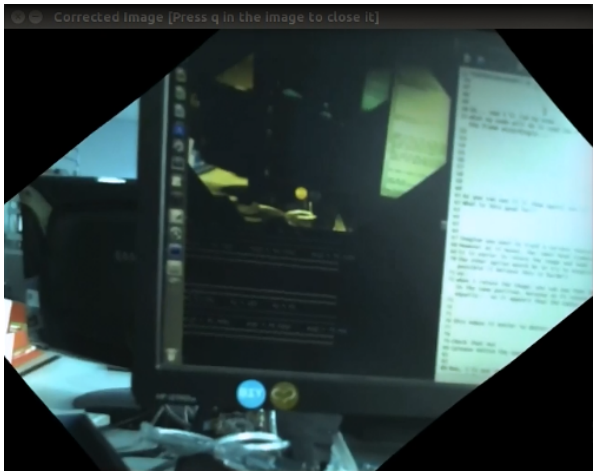


Figure 2.10: Sensor attached to camera - *"inertial_correct_cam"* output.

was added or filtered in any set of data. Nevertheless, this script may be useful for other purposes than our own and it gave me some experience working on the ROS environment, making it a deprecated but necessary step in solving this problem (data merging).

Chapter 3

Experiments With Inertial Sensors

This chapter explains the main experiments done using the inertial sensors, with the objective of trying to understand its outputs and how they can be used in order to balance the robot. Essentially, as was mentioned before, the main focus of study will be on accelerometers and gyroscopes. It is presented in this chapter the analyzes of these sensors data in order to obtain linear and angular position values.

3.1 Accelerometers Output

3.1.1 Finding Linear Position

The first tested subjects were the accelerometers, because it was thought that by trying to obtain the linear position, velocity and acceleration of the robot, valuable knowledge of the dynamics of the humanoid would be gained. As it was said in section 2.1.1 on page 7, the accelerometers record a constant acceleration value, even when they are on a table. This acceleration is the acceleration of gravity, which needs to be eliminated in order to obtain the true acceleration of the sensor. This can be done using the formula: [40]

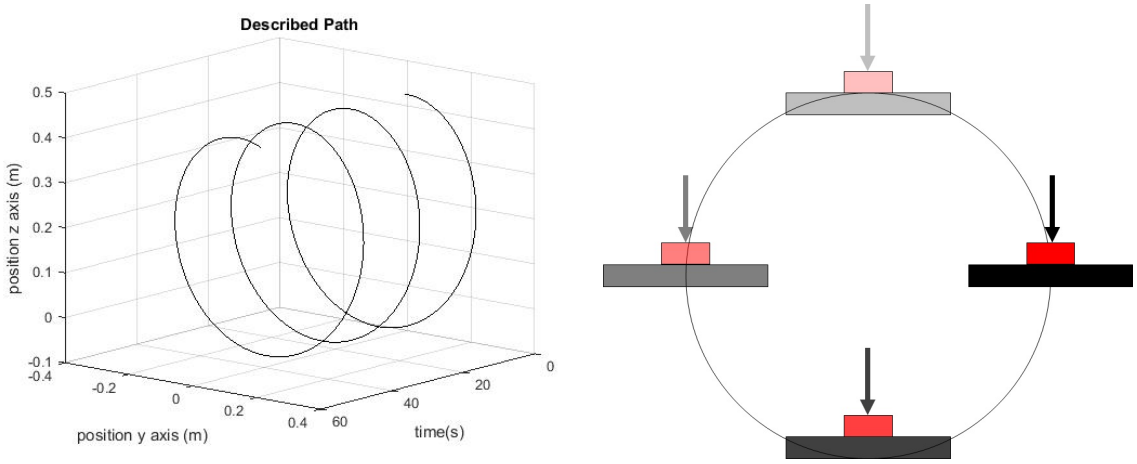
$$a_B = a_m - R_I^B \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} [m.s{-}2] \tag{3.1}$$

Where:

a_B relative acceleration
a_m absolute acceleration
R_I^B rotation matrix of the sensors (defines orientation)
g is the acceleration of gravity ($9.81 m.s^{-2}$)

In the first experiments this equation was considered, but the knowledge to obtain the orientation of the sensors was nonexistent, so in order to facilitate the calculations, all the paths described by the sensor had the same orientation, making it easier to remove the acceleration of gravity. The path for the first accelerometers experiment is described in Figure 3.1.

Using this path, the force of gravity was always applied to the accelerometer that measures forces in the z-axis, thus making it easy to remove that force.



(a) Path of the Sensor - Isometric.

(b) Path of the Sensor - YZ Plane.

Figure 3.1: Path of the Sensor - Accelerometers Evaluation.

Some corrections were needed in order for all the data to be in accordance with SI. The first thing is the accelerometers data. The accelerometers return a value between -4096 and 4096 in which $1G = 255$. In order to obtain the accelerometers in $m.s^{-2}$ their values have to multiply by ${}^{9.81}/_{255}$. The time-stamp in ROS is presented in nanoseconds, so to convert it to seconds, it is needed to multiply every time value by 10^{-9}. Furthermore, to have the begin of the experiment at $t = 0$, the first measurement of time was subtracted from all the other time measurements. The data obtained from the industrial manipulator is in mm, so it needs to be divided by 1000. To summarize:

Ground Truth Corrections:

$$t_{groundtruth_k} = (t_{fanuc_k} - t_{fanuc_0}).10^{-9}[s] \quad (3.2)$$
$$x_{groundtruth_k} = x_{fanuc_k}.10^{-3}[m] \quad (3.3)$$
$$y_{groundtruth_k} = y_{fanuc_k}.10^{-3}[m] \quad (3.4)$$
$$z_{groundtruth_k} = z_{fanuc_k}.10^{-3}[m] \quad (3.5)$$

Inertial Corrections:

$$t_{inertial_k} = (t_{sensor_k} - t_{sensor_0}).10^{-9}[s] \quad (3.6)$$
$$\ddot{x}_{inertial_k} = \ddot{x}_{sensor_k}.{}^{9.81}/_{255}[m.s^{-2}] \quad (3.7)$$
$$\ddot{y}_{inertial_k} = \ddot{y}_{sensor_k}.{}^{9.81}/_{255}[m.s^{-2}] \quad (3.8)$$
$$\ddot{z}_{inertial_k} = \ddot{z}_{sensor_k}.{}^{9.81}/_{255}[m.s^{-2}] \quad (3.9)$$

Where:

t is time
the subscript $groundtruth$ is the treated data, obtained from the FANUC robot
the subscript $fanuc$ is the untreated FANUC robot data

the subscript *inertial* is the treated data, obtained from the inertial sensors
the subscript *sensor* is the untreated inertial sensor data
k is the instant in time
x, y, z are the linear position in the axis mentioned
$\ddot{x}, \ddot{y}, \ddot{z}$ are the linear acceleration in the axis mentioned

Knowing the linear acceleration, an integration is needed to obtain the linear velocity. Integrating once more will result in the linear position. All that is needed, is the initial velocity and position. The position is known (it is given by the industrial manipulator), but the velocity is not. Nonetheless, an estimation for the initial velocity, may be done using the first two measurements of position, which are given by the FANUC manipulator. Another way to obtain the initial velocity is to start the experiment in a resting position, meaning that the initial velocity will be equal to zero. Since we have discrete data, the best way to calculate the linear velocity and position is using formula 3.10 and 3.11. The outputs were as represented in Figures 3.2 to 3.6.

$$v_{k+1} = v_k + \Delta t a_k [m.s{-}2] \tag{3.10}$$
$$p_{k+1} = p_k + \Delta t v_k [m] \tag{3.11}$$

Where:

a_k is the acceleration in instant k
v_k is the velocity in instant k
p_k is the position in instant k
k is the time iteration

Figure 3.2 and 3.3 represent the data obtained directly from the nodes used. There are some conclusion that can be obtained. Looking at Figure 3.2 and Figure 3.4, it seems that they both display the same data, meaning that there is no treatment of the accelerometers data in the node *"complementary_filter"*. Looking at Figure 3.3 and Figure 3.5 the same conclusion emerges for the gyroscopes data. This suggests that *"complementary_filter"* is doing nothing in treating the accelerometers or gyroscopes outputs (as it was expected to do). This fact made me (wrongly) ignore this node for a long time.

Having the data obtained from the inertial sensors and using the formulas 3.10 and 3.11, the computed velocity and position were as seen in Figure 3.7 to Figure 3.9:

Observing the computed velocity and position some conclusions can be reached. First of all, since the sensors describe a circular motion in the y0z plane, it was expected that the position in the x-axis (Figure 3.7) would be constant. However, there is an error of $100m$.. The velocity and acceleration should both be zero too, but that is not the case. It can be observed that the noise is disrupting the results. In Figure 3.8 and Figure 3.9 the same conclusion can be reached. It was expected that the position would describe a oscillatory like curve. That is not the case. The only conclusion is that the use of accelerometers to calculate the linear motion of the robot can't be utilized, since the sensors are highly susceptible to noise.

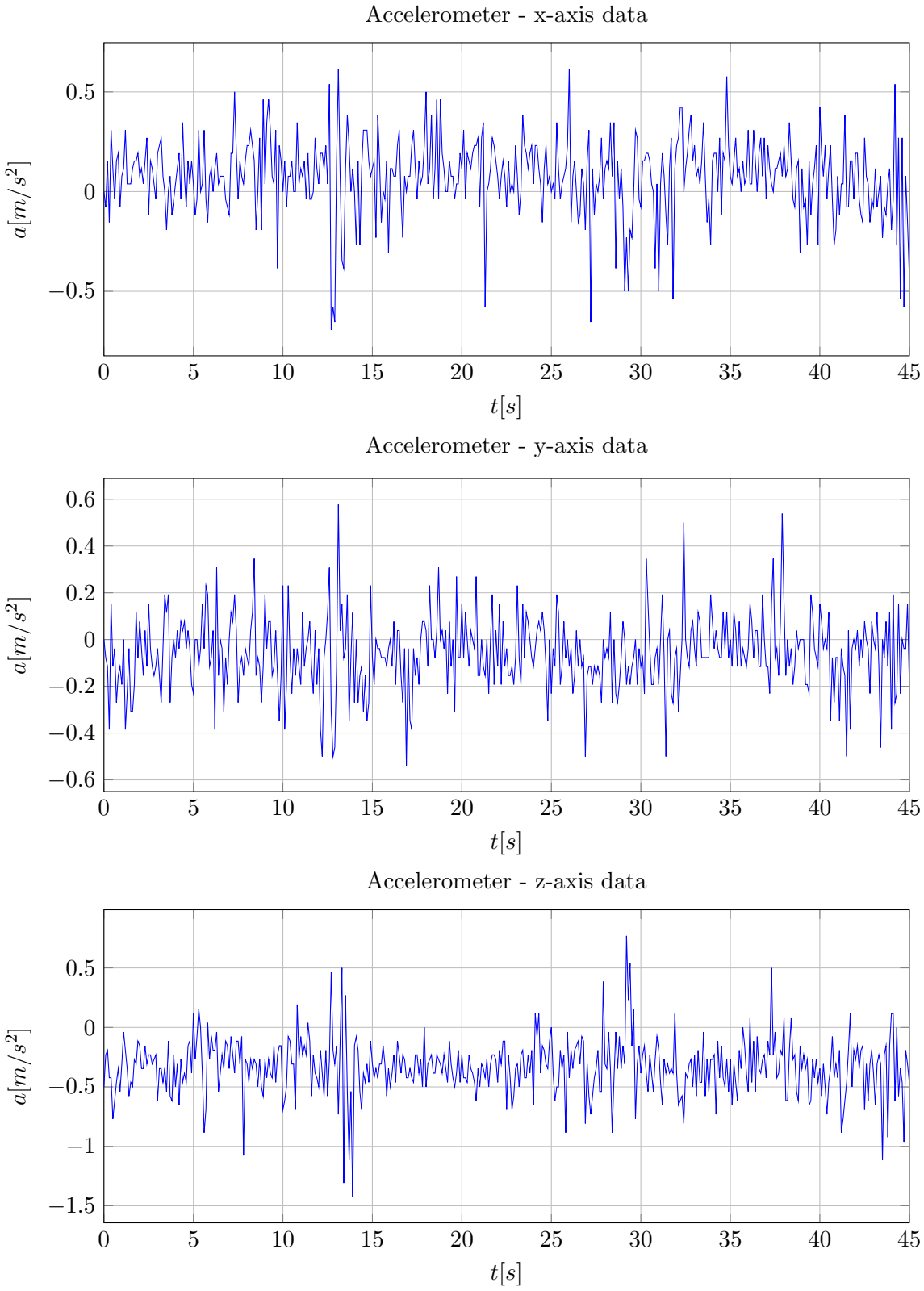


Figure 3.2: Accelerometer Data - Experiment #1

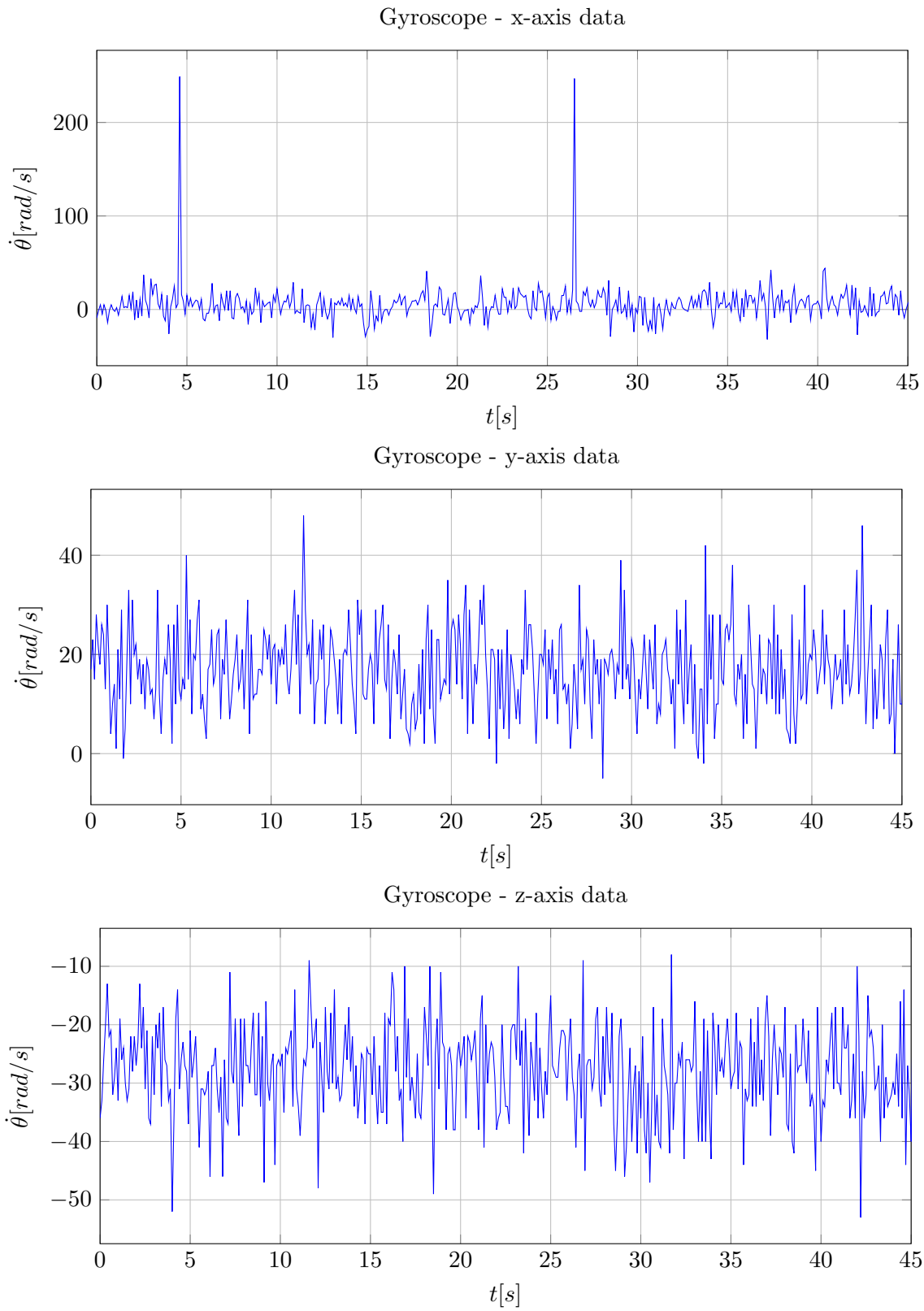


Figure 3.3: Gyroscope Data - Experiment #1

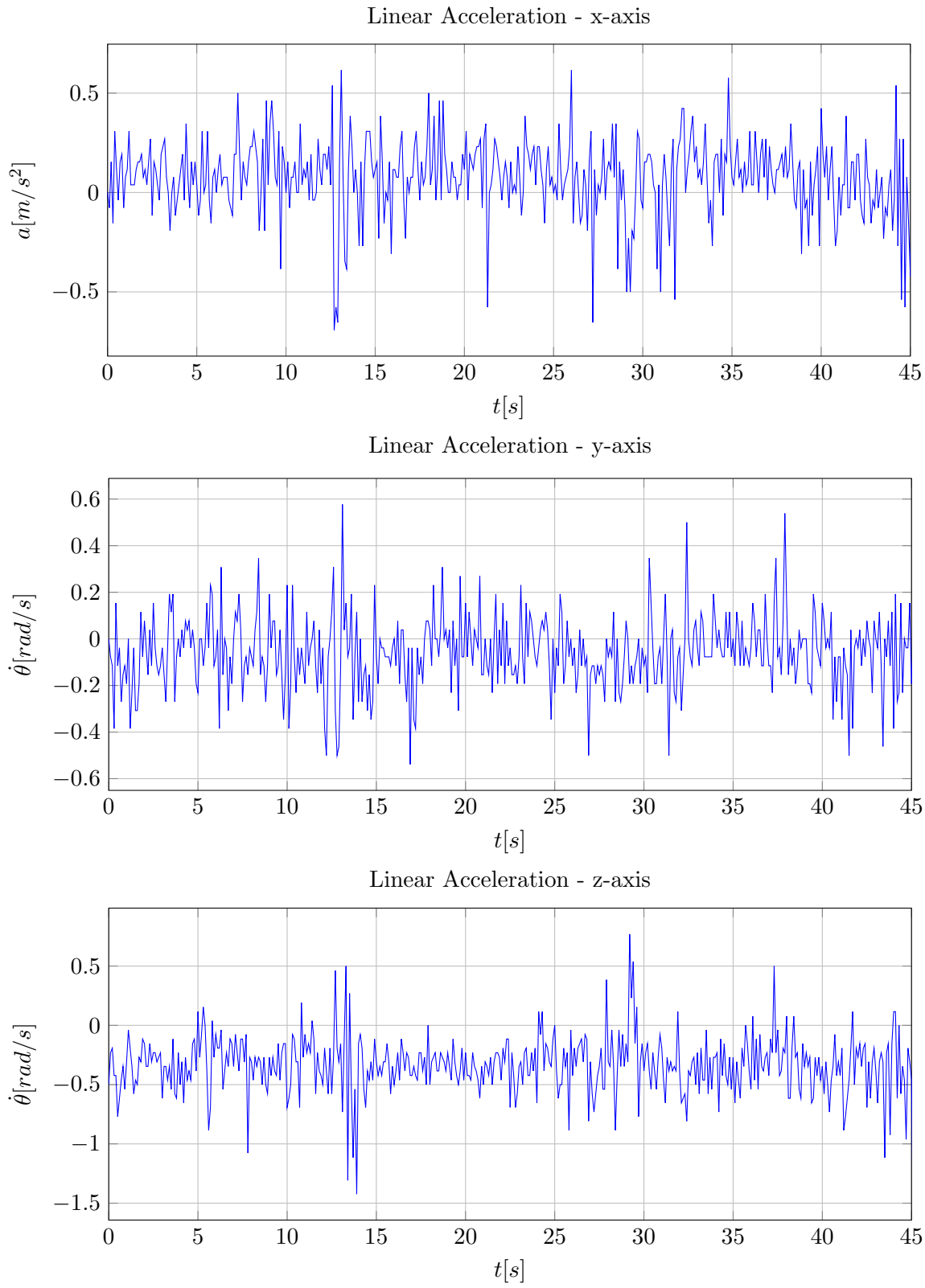


Figure 3.4: Linear Acceleration Data - Experiment #1

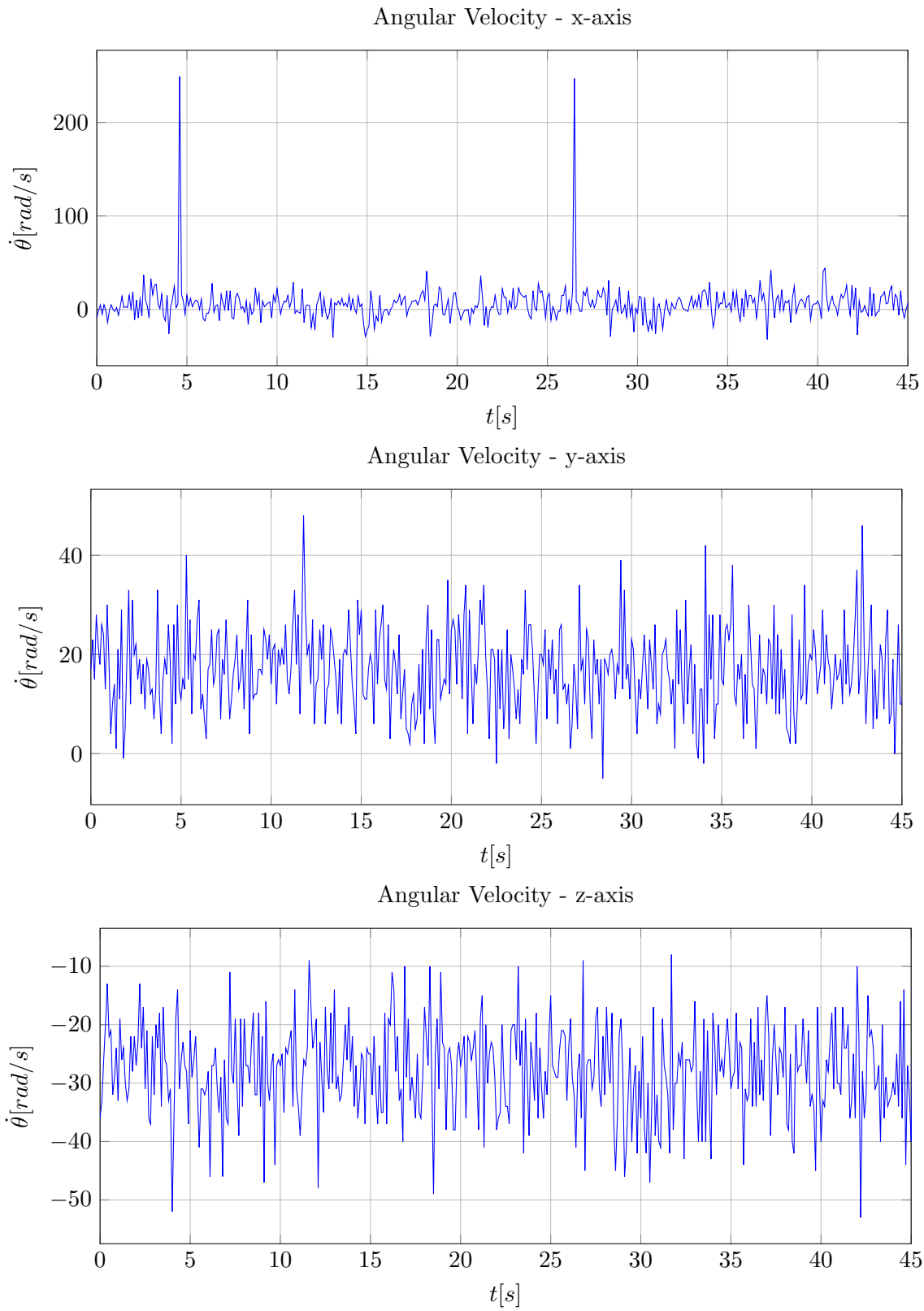


Figure 3.5: Angular Velocity Data - Experiment #1

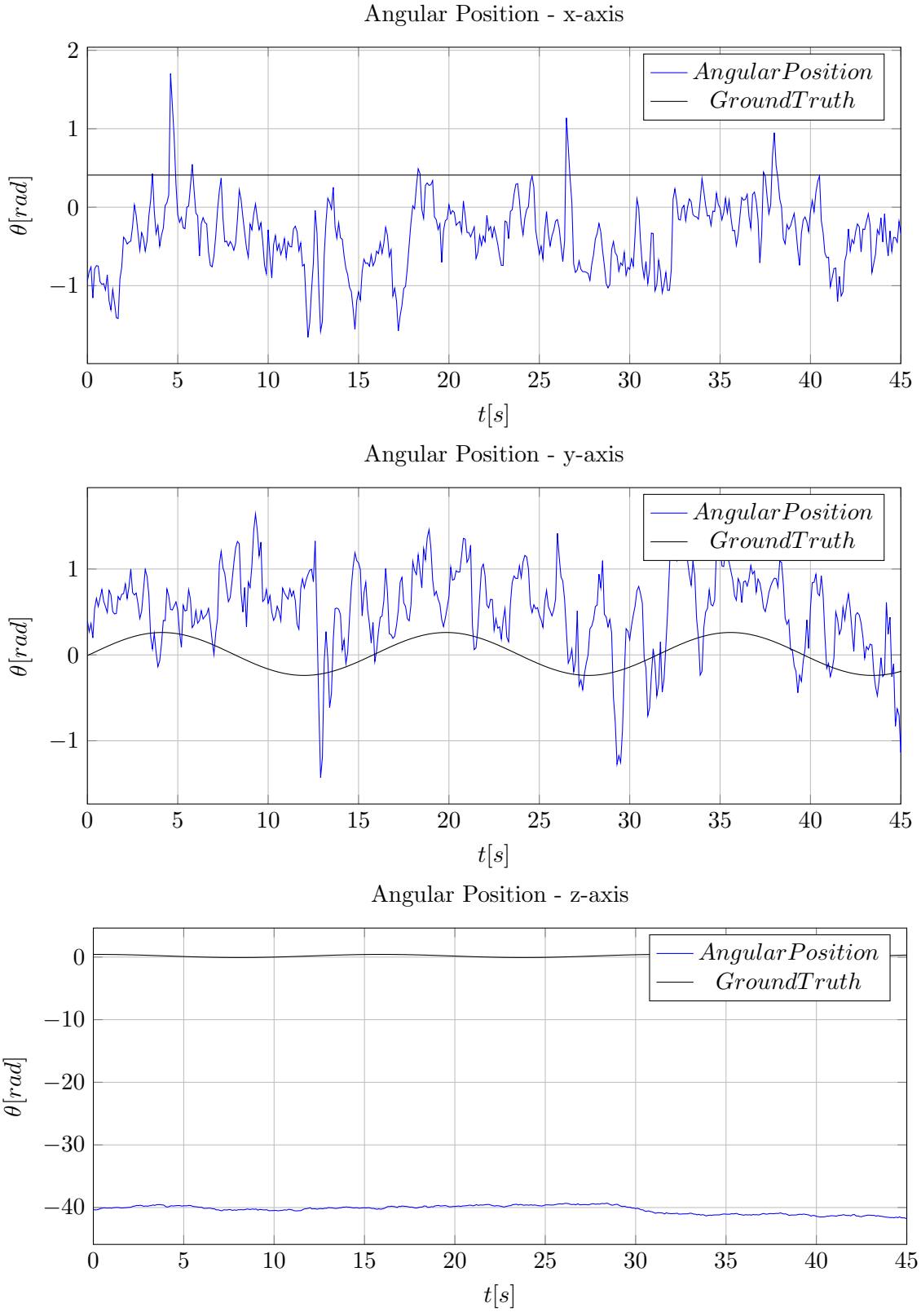


Figure 3.6: Angular Position Data - Experiment #1

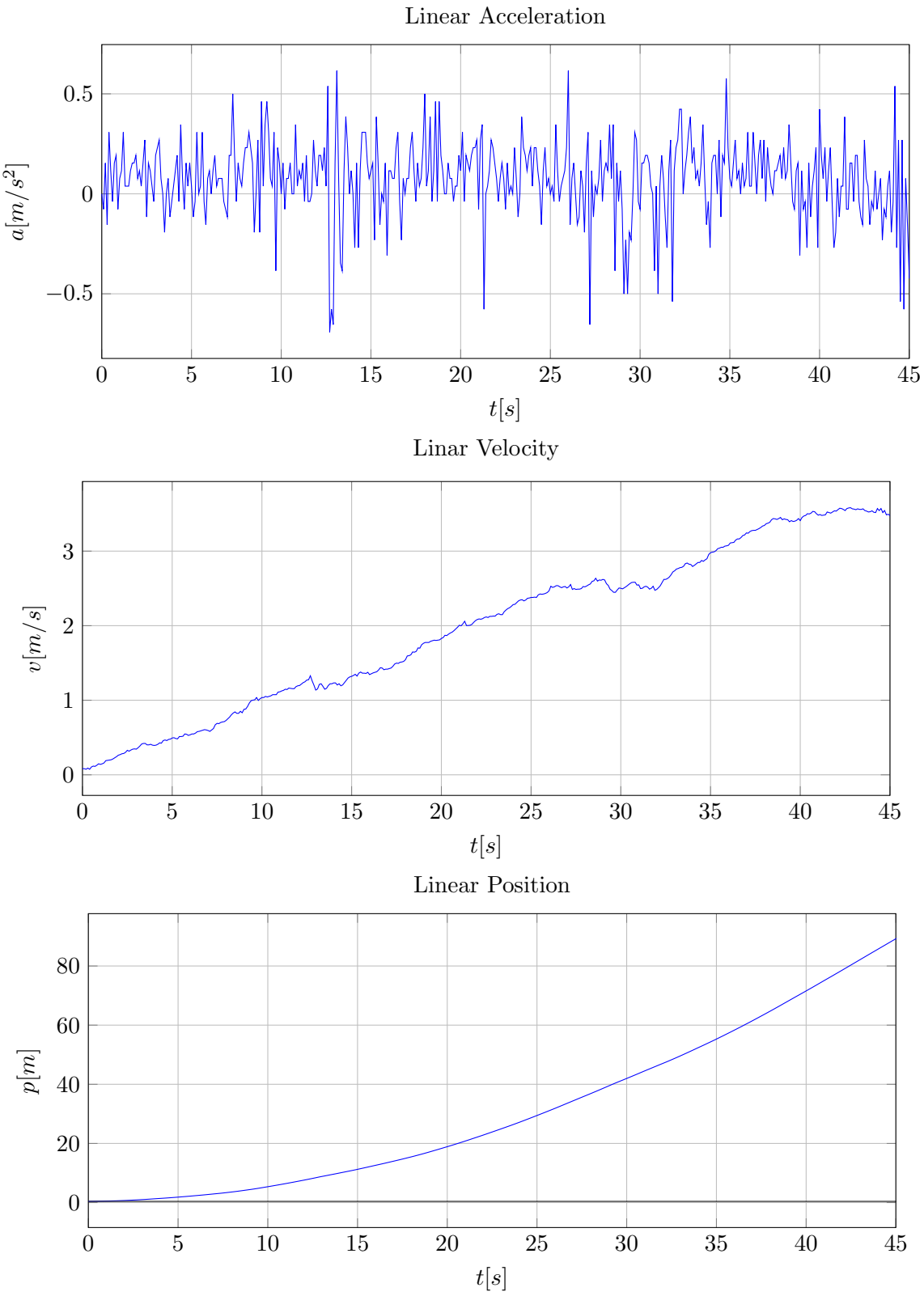


Figure 3.7: Integrated Accelerometer data - X-Axis

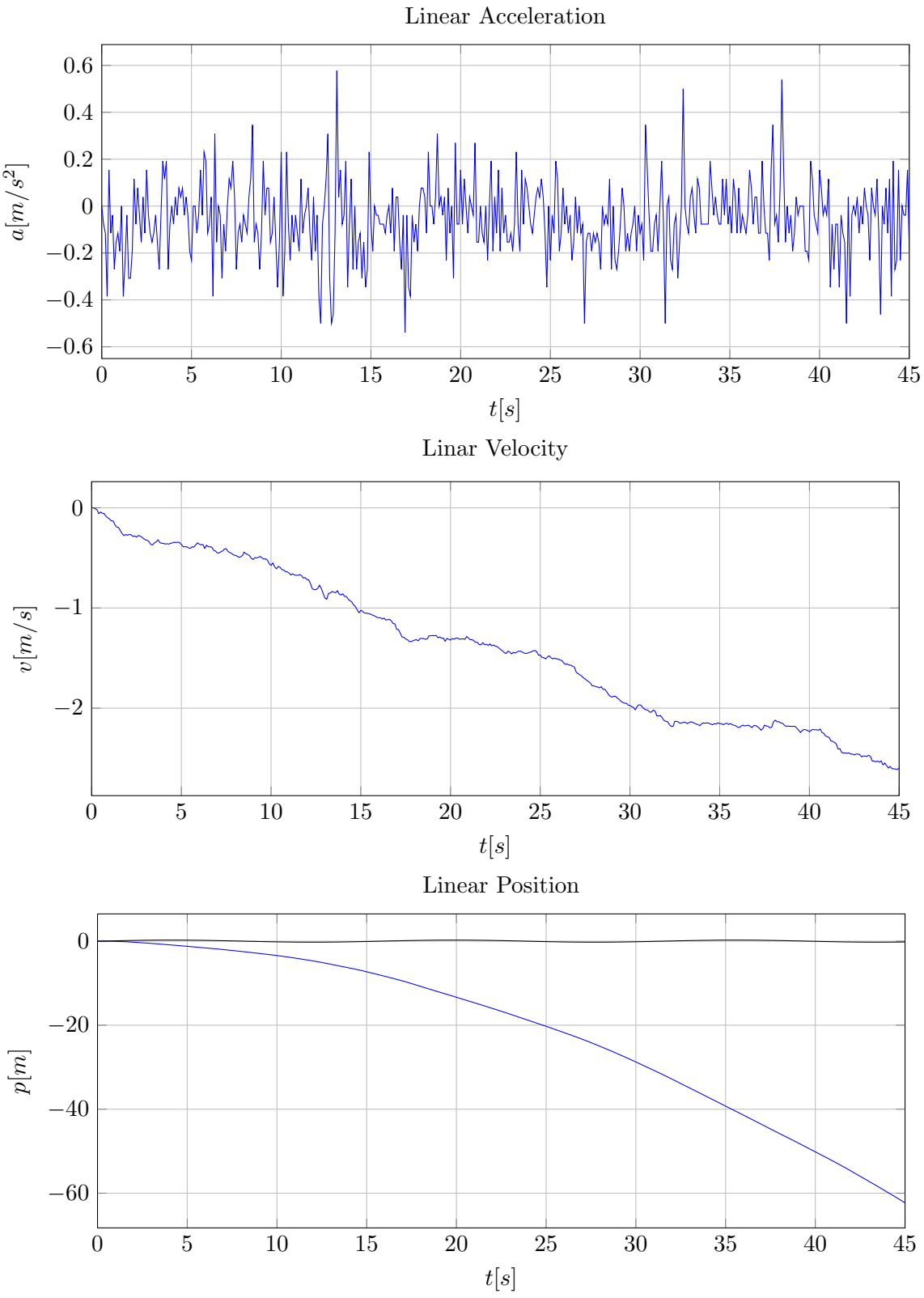


Figure 3.8: Integrated Accelerometer data - Y-Axis

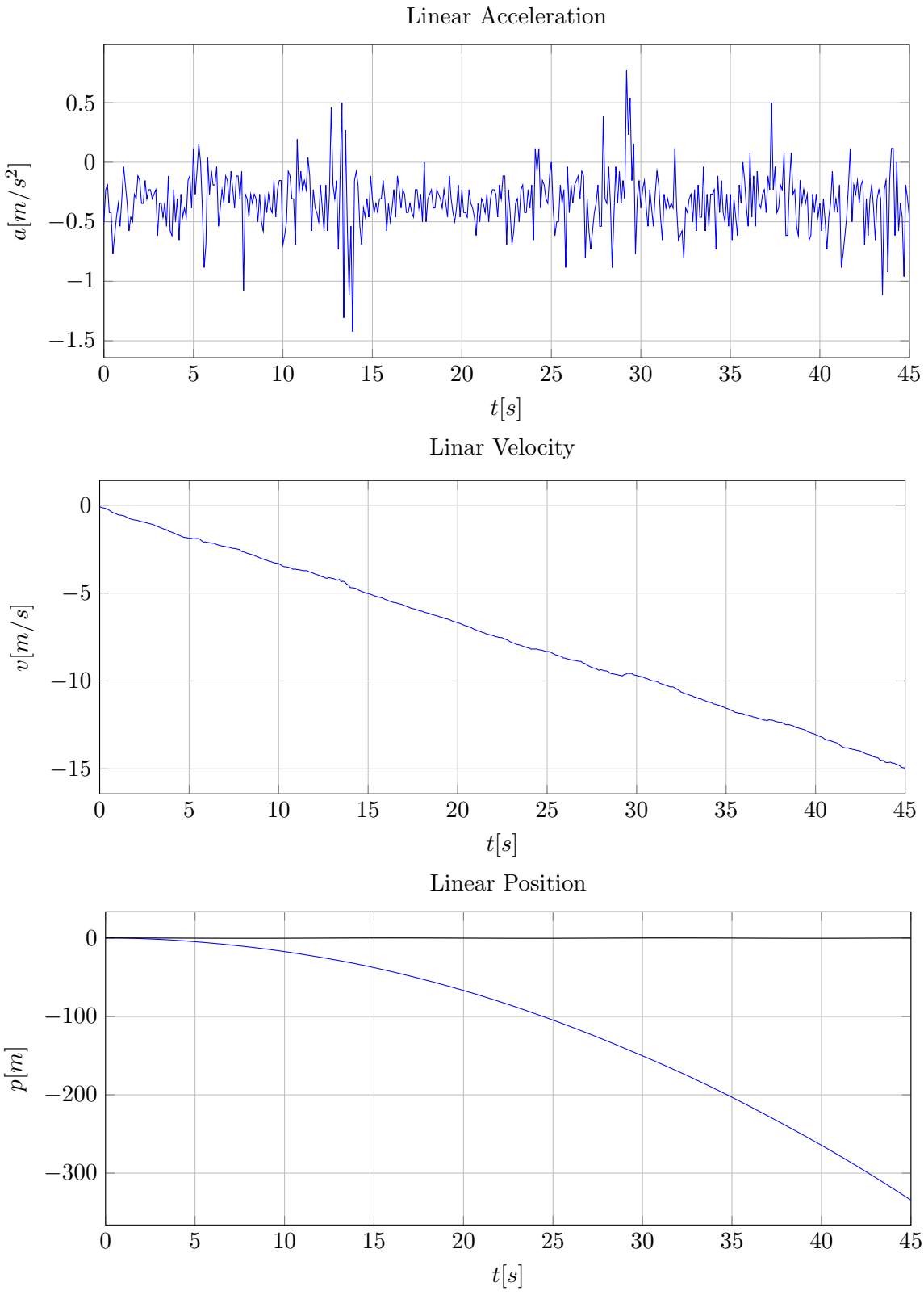


Figure 3.9: Integrated Accelerometer data - Z-Axis

Figure 3.10: 3D analyses of orientation finding using accelerometers.

3.1.2 Finding Orientation

After arriving to the conclusion that it wouldn't be possible to use the accelerometers to estimate the linear position, it was tried to use them as inclinometers. The reason accelerometers were thought to be a good way to find orientation is that, opposed to the gyroscopes, which calculate the orientation values based on the previous data (which causes an increasing error), the inclinometers would calculate the orientation using the gravity. This would possibly decrease the error in the orientation calculation. This idea would also be proven wrong. This time, instead of presenting the experiments that were done (which offer limited conclusion) a visual and mathematical explanation as why it can't be done will be presented. The initial idea was to define the sensors such as seen in Figure 3.10:

Theoretically, knowing the force applied by gravity in z, comparing it to the gravity acceleration and then looking at the sign of y, the rotation in the x-axis is found.

$$orientation = \arccos\left(\frac{z_{measure}}{9.81}\right)[^\circ] \tag{3.12}$$

In equation 3.12 the final result would be 45° or -45° depending on the sign of y. This process can be applied to the rest of the examples as well. For y axis rotation, the equation would be the same and the sign of x-axis would define the rotation direction. However, looking at the z rotation, the problem with this approach appears. Imagine that the gravity force is coincident with the z-axis. In that case, the rotation in the z-axis would always give the same result in the remaining axis, independently of its position in the xy plane. So, using this definition would only achieve the measurement of two angles at any given time.

After realizing this, a search for a mathematical expression begun. Defining the rotation in x-axis as roll (Φ), y-axis as pitch (θ) and z-axis as yaw (ψ) (Figure 3.10), it is possible to calculate these angle rotations using:

$$\Phi = \arctan\left(\frac{R_{z,y}}{R_{z,z}}\right)[^\circ] \tag{3.13}$$

$$\theta = \arctan\left(\frac{-R_{z,x}}{\sqrt{R_{z,y}^2 + R_{z,z}^2}}\right)[^\circ] \tag{3.14}$$

$$\psi = \arctan\left(\frac{R_{y,x}}{R_{x,x}}\right)[^\circ] \tag{3.15}$$

Where R is a matrix which defines how two different coordinates referentials are related. In this case the referential belonging to the inertial sensors and the one belonging to the ground truth, in which the gravity vector and the z-axis are coincident.

$$R = \begin{bmatrix} R_{x,x} & R_{x,y} & R_{x,z} \\ R_{y,x} & R_{y,y} & R_{y,z} \\ R_{z,x} & R_{z,y} & R_{z,z} \end{bmatrix} \tag{3.16}$$

$R_{z,x}$ is the measure of the force of gravity in the x-axis of the sensor, $R_{z,y}$ is the measure of gravity in the y-axis and $R_{z,z}$ the measure of gravity in the z-axis. In other words, this values can be measured, so Φ and θ can be calculated. This way of calculating angles is much more interesting since the arctan function accepts all real numbers, while arccos only accepts values between [-1;1].

However, when trying to calculate ψ (using the accelerometers), since there is no constant force coming other than gravity, which leads to $R_{x,x}$ and $R_{y,x}$, being equal to zero. In other words, since there aren't any forces that actuate on the inertial sensors that could be used as references (except gravity), the values of $R_{x,x}$ and $R_{y,x}$ are zero, which will lead to an unknown value of Ψ.

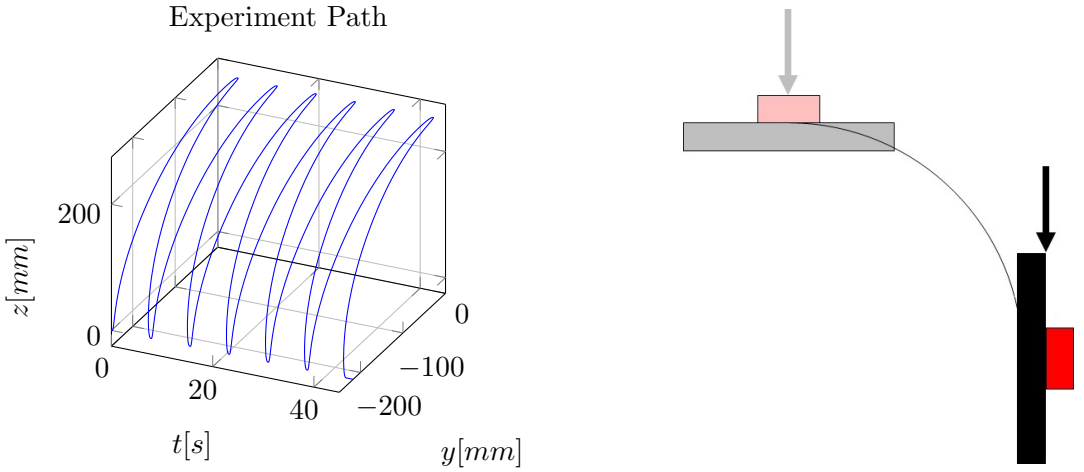
Mathematically, this is the explanation for why it is not possible to use the accelerometers as inclinometers to find the angles of rotation in all axis and therefore its orientation. There is another problem associated with this, which is that, every time an acceleration that it's not given by gravity is applied to the sensors, our angles will suffer an error. Using accelerometers as inclinometers is only advisable with static objects.

3.2 Gyroscopes Output

After the conclusion that the accelerometers would not give either the orientation nor the linear position of the humanoid, attention was turned to the gyroscopes. The idea was the same, but this time the angular velocity would be integrated to obtain the angular displacement.

In order to test that, the path of the experiment (Figure 3.1) had to be remodeled, since no rotation occurs in it. For that, the path represented in Figure 3.11 was created.

In this experiment, the Euler angles of the industrial manipulator end-effector (w, p, r), where $w = \Phi$, $p = \theta$ and $r = \psi$ were used as ground truth. To confirm that, the Euler angles can accurately describe the rotation of the FANUC robot, a simulation was created, where an object was rotated, using w, p, r, obtained from its end-effector. The Euler angles value



(a) Path of the Sensor - Isometric View.

(b) Path of the Sensor - YZ Plane.

Figure 3.11: Path of the Sensor - Gyroscopes Evaluation.

was represented as well. In w and r it can be seen that they both are constant at $-90°$, although having some peaks (Figure 3.14). These peaks are simultaneous in both angles and occurred due to the FANUC's euler angle definition, i.e., it can be seen that they appear when $p = -90°$. This happens because if the rotation in the y-axis continued, the industrial manipulator would assume that $w = -180°$ and $r = 0°$. On the other hand, it can be seen that p oscillates between $0°$ and $-90°$ which is consistent with the experiment done.

Using formula 3.11 from page 19, applied to the gyroscope angular velocity, the angular orientation should be obtained. This formula was applied to the data obtained directly from the sensors and to the data obtained from node *"complementary_filter"*. The reason for using both sets of data is to confirm the theory that the inertial data doesn't change after being treated by *"complementary_filter"*. If the data is even slightly different, the results will end up being very different. If the data is exactly the same, so will be the output of this calculation. The results were as presented in Figure 3.12 and Figure 3.13.

At this point there is no doubt that, both sets of data are the same. On the other hand, it can be seen that the results obtained are not that bad. Looking at the y-axis angular position, it can be sees that it behaves like it should with the exception of an error that is being accumulated through time. This conclusion is valid to both x and z-axis as well.

Two conclusions were reached:

1. There is no point in using the *"complementary_filter"* node since it is not treating the data.

2. A way has to be found to treat the data, in order to eliminate the gain in the integrated angular velocity (Figure 3.12)

Even though *"complementary_filter"* is not treating the raw data from the accelerometers nor the gyroscopes, there is a third output in this node, the angular position. Converting the angular position obtained from the *"complementary_filter"* node from radians to degrees gave the results presented in Figure 3.14.

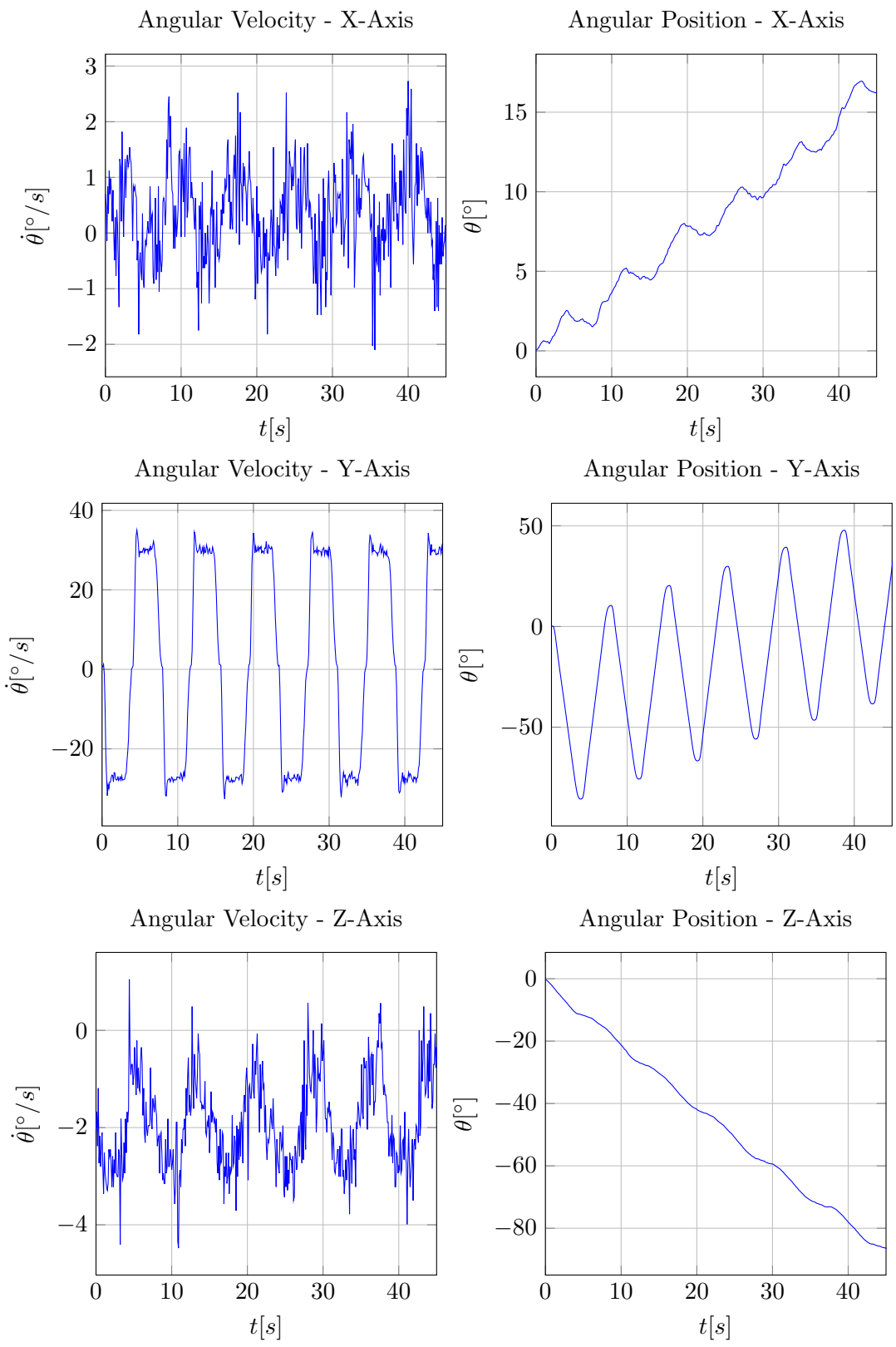


Figure 3.12: Angular velocity and position (integration of velocity), obtained from the gyroscopes.

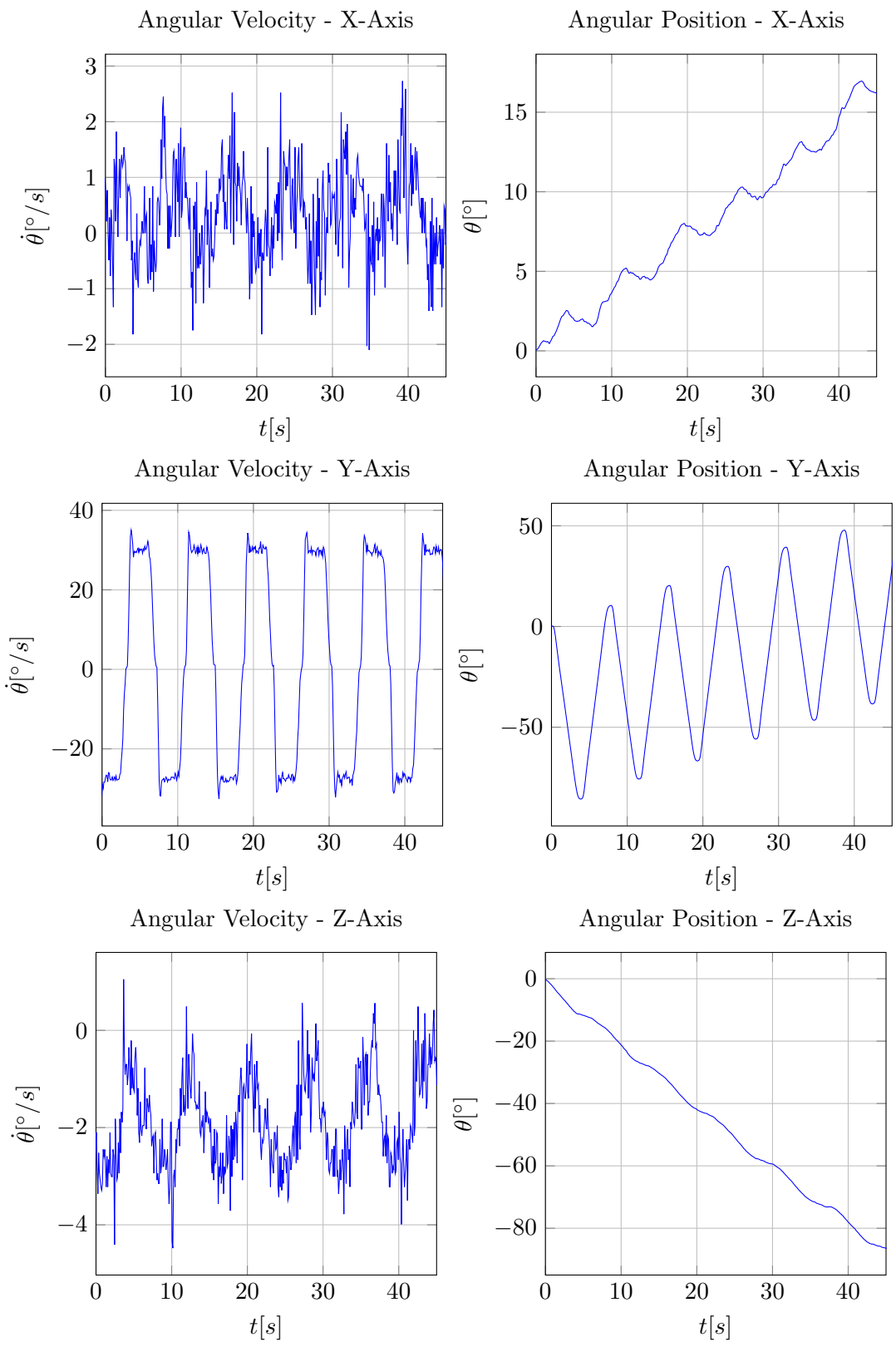


Figure 3.13: Angular velocity and position (integration of velocity), obtained from node *complementary_filter*.

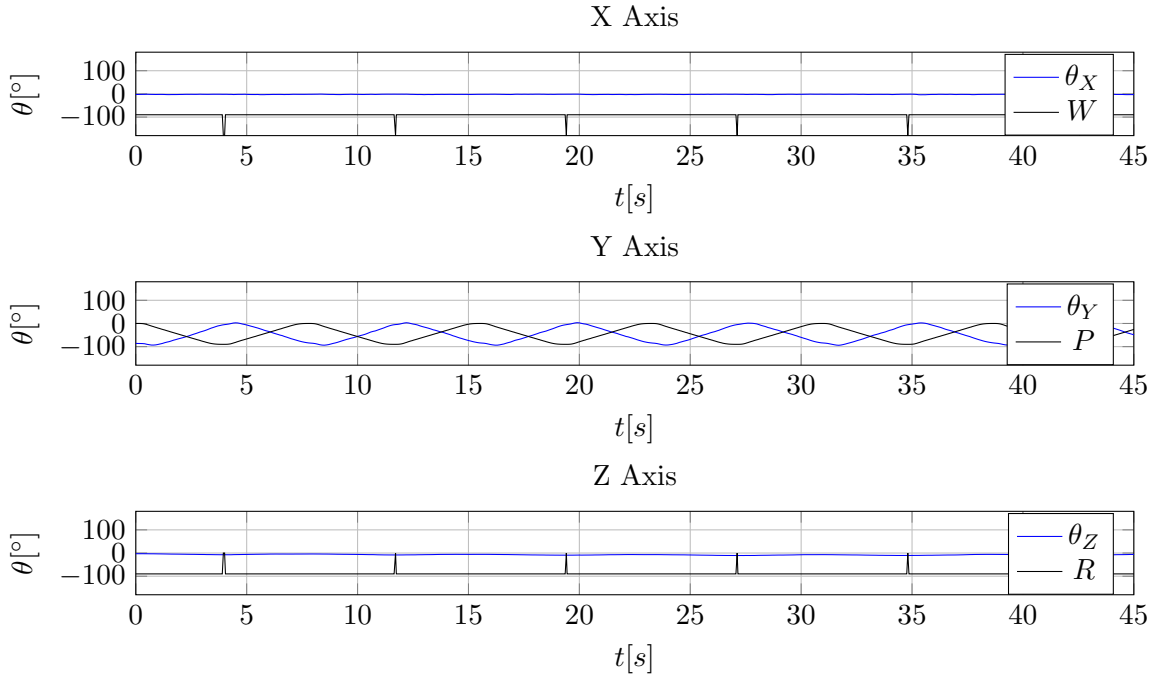


Figure 3.14: Angular Position VS Ground Truth (before adjustments).

In Figure 3.14, θ represents the angular position calculated by "complementary_filter" and W, P, R the obtained Euler angles from the FANUC manipulator. Looking at the θ_x and θ_z, it can be seen that there seems to be no variation in the angle (as it would be expected). In θ_y, the angle values are between $0°$ and $-90°$, which is consistent with the experiment. After some corrections in order to match the reference frames of both sets of data, the results were as shown in Figure 3.15.

It can be seen, in Figure 3.15, that the black line (ground truth) and the blue line (data obtained from complementary_filter) are coincident. This suggests that the orientation of the humanoid can be obtained, using the "complementary_filter" angular position. In order to do that, it is needed to understand how to place the sensors in the humanoid and run some tests in order for all the reference frames in the sensors to make sense in the robots perspective. It can also be seen that there are some errors in the angular position measures. One way to solve this problem, is by using a Kalman Filter, described further.

In summary, this chapter described the practical experiments using the IMU. Some conclusion were reached. The accelerometers were found to be unreliable regarding the tracking of linear position, since they are highly susceptible to noise. That noise is in turn amplified when the measurements are integrated in order to obtain the linear position. Using the acceleramoters as inclinometers is an alternative, since they are able to measure its orientation regarding the gravity vector. However, the orientation given by the accelerometers is only reliable if the system is not experiencing any force, except the force of gravity. At the same time, the inclinometer is limited to the knowledge of the sensor in only two angles. Notice that any transformation that occur with the device in a plane that's perpendicular to the gravity vector is unnoticed by the inclinometer. For example, if the force of gravity is felt only in the z-axis, it can be deduced that the z-axis is pointing up (or down) in a vector that

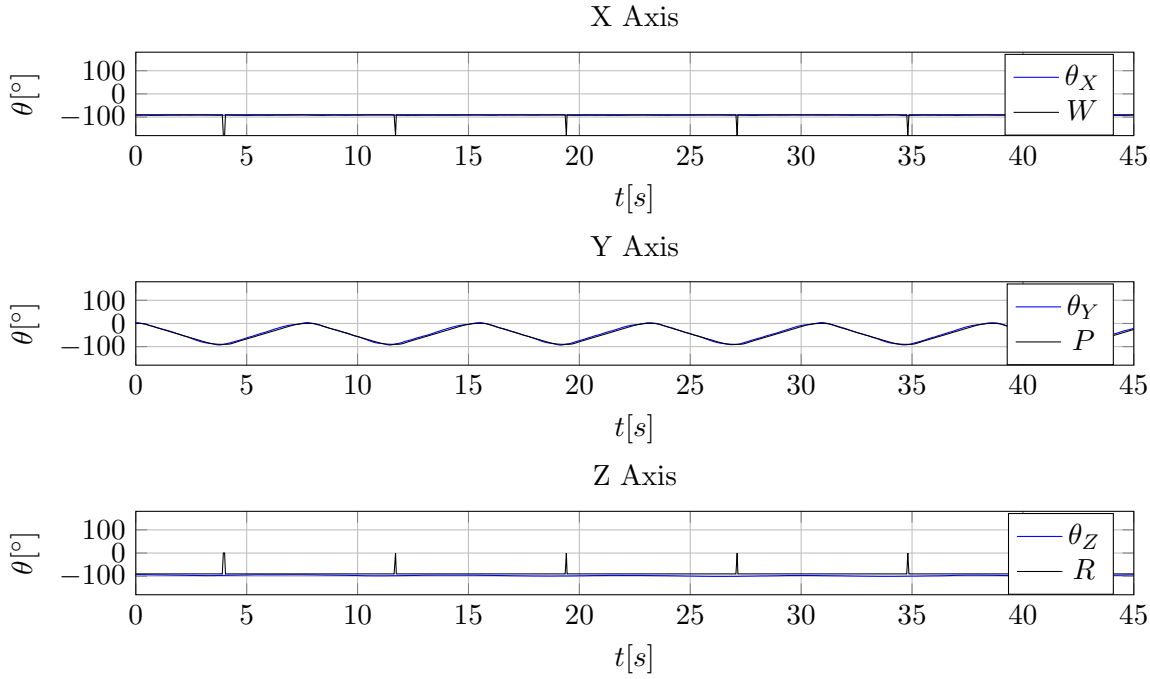


Figure 3.15: Angular Position VS Ground Truth (after adjustments).

is perpendicular to the ground, but no deductions can be made for the orientation of the x and y-axis, except that they are both parallel to the ground.

The gyroscopes are able to return the angular velocity and aren't as susceptible to noise as the accelerometers. It is possible to integrate the angular velocity in order to obtain the angular position. However, there seems to be a gain after integrating the data.

As an alternative, it is possible to use the *complementary_filter* node, which makes an average of the angular velocity values obtained in order to integrate them into angular position measurements. Thus, the angular velocity and position, are both outputs of this node. The node *complementary_filter* was tested against the ground truth obtained from the industrial manipulator, in which it was possible to see that the data is quite accurate and therefore, reliable.

Chapter 4

Experiments With Visual Features

This chapter analyzes how the visual data was treated in order to obtain angular position and velocity results. Although many of the methods used in this chapter are standard applications of algorithms designed by others, it is believed that a basic understanding of the used tools is important. In other words, tools like OpenCV and Matlab can be used to extract SURF features from an image and match the obtained results with a database, without knowledge of the method used by the function. However, having a general idea of what the function is doing may prove to be useful in more creative situations. Nevertheless, this chapter intends to give a simple, rather than a detailed, explanation of some concepts.

4.1 Blob Detection Method

This method was developed in order to detect regions in a frame, which differ from the surroundings, may it be color or brightness. Normally, within the same blob, the properties are approximately constant, meaning that a blob tends to be homogeneous. These blobs are large sets of pixels, which together form an object. Thus, they are commonly called *global features*.

4.1.1 Detecting the Blobs

The absolute easiest way to detect blobs is to threshold an image and look for aggregations of white pixels. A given set of white pixels, surrounded by black pixels is an object, a blob that was detected and has its own properties. These properties range from area, perimeter, centroid, bounding box, bounding elipse, Euler number, extrema among many others and are used to define, identify and distinguish or match blobs with same or similar properties.

This method was experimented in a dark room, pointing at a long rectangular window during the day as it can be seen in Figure 4.1a, which is very advantageous for blob extraction, since the difference in lighting will lead to a smooth threshold and therefore good blob detection (see Figure 4.1b). In this case, the long shaped rectangular windows present themselves as a big advantage, since they can be used to obtain the orientation of the windows and consequently the orientation of the camera (humanoid's head).

A standard way to obtain the orientation of a blob is calculating the angle between the x-axis and the major diameter of the ellipse that includes the blob. Thus, long rectangular shaped (or ellipse shaped) blobs are more robust to orientation calculation.

(a) Background of Visual Trial (frame zero). (b) Binary image of Background of Visual Trial.

Figure 4.1: Background and consequent threshold.

4.1.2 Blob Tracking

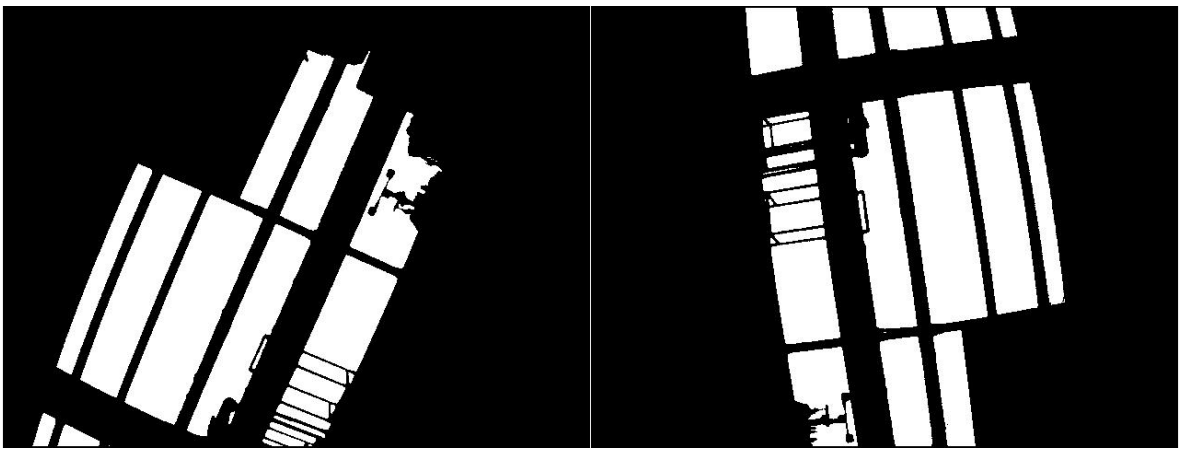
The first thing needed to do is to choose which blobs should be used at any instant in order to obtain the orientation. A way to do this is to keep track of the area of each blob. If the area of a blob changes more than a preset value, then it could be assumed that the blob is moving relative to the camera, causing it to be hidden by something or revealed by something, thus alternating the nature of the initial blob. These blobs should be discarded.

Looking at Figure 4.1b, it can be seen that the left side blobs are extremely smaller than the central ones. Assuming that those blobs do not represent a real (complete) object, their use should be discarded. In the same image, looking to the fourth central blob counting downwards, it can be seen that the blob is pretty slim, but in Figure 4.2a, its height seems to increase and in Figure 4.2b, the same blob gets divided (into four blobs), meaning that the blob can't always accurately describe its orientation. This blob should be ignored as well. As demonstrated, keeping track of each blob area is useful in order to make some initial assumptions.

As was said, the orientation of a blob is calculated using the x-axis and the ellipse greater diameter, meaning that if they're parallel, the angle is $\theta = 0°$. As seen, in this trial, the frame corresponding to the camera having an orientation of 0°, has the major diameter aligned with the the x-axis, meaning that the blobs and the camera "share" its orientation. Thus, at any point in time, Fit can be deduced that the window orientation will be the same as the camera orientation. This means that at any time the orientation of the camera can be calculated, without needing to rely on older data, which in turn reduces the amount of accumulated error. The results were as shown in Figure 4.3, where the mean difference (MD) is calculated using formula 4.1 and is equal to 1.56° when compared to the values obtained from FANUC robotic arm.

$$MD = \sum_{i=1}^{n} \frac{|data_{1_i} - data_{2_i}|}{n} \tag{4.1}$$

This method worked in retrieving orientation from the Visual Data. The value 1.56° may



(a) Camera at left extreme). (b) Camera at right extreme.

Figure 4.2: Examples of Hidden Blobs.

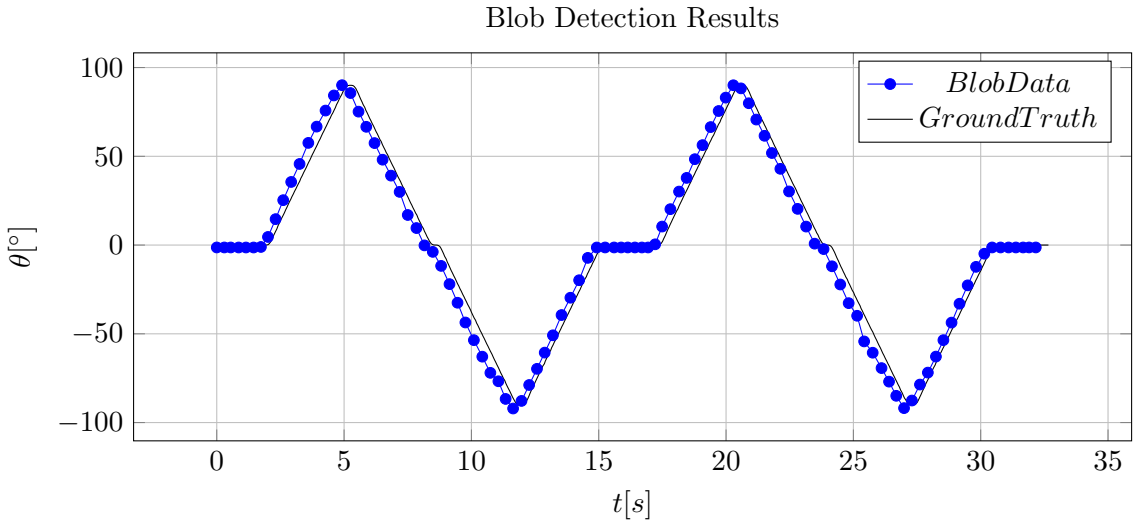


Figure 4.3: Blob Detection VS Ground Truth.

be compared to the obtained values in chapter 5.3. On a final note, if an experiment takes to long or the camera is moving far from its initial place, there is a high probability that the original blobs are lost. Therefore, new blobs must be found, if possible.

Although this method proved to be functional in this experiment, that will not always happen. Remember that in this case the background couldn't be more suited for blob detection, but that is not always the case. In order to test the robustness of this method, the same was applied to a second experiment (Figure 5.1b), in which there were no blobs detected at any time during the experiment, due the poor lighting conditions of the room. This proves that this method is not robust enough, since it won't always find blobs to obtain the orientation from.

For this fact, this method should be used as an auxiliary method to the feature detection, aiding the results whenever possible and discarded when not needed.

4.2 Local Feature Detection

The implementation of another method was found necessary. Thus, the local feature detection method was chosen. Local features are keypoints in a frame that can describe their surroundings and be easily distinguished from it. There are several methods to detect features, and some of the most known include the SIFT [41], SURF [42], and more recently the FAST [43] method, among others. In this work it was used the SURF method, because it is faster than SIFT, even though it isn't as accurate [44].

4.2.1 SIFT

As mentioned, SIFT is a method to detect and extract keypoints (pixels) from a frame. This method has the advantage of being scale-invariant (thus the name Scale-Invariant Feature Transform), meaning that the same keypoint will be detected in different images with different zooming of the same area. It is also rotation-invariant, meaning that rotated images will still detect the same local features.

SIFT features are obtained by applying a gaussian filter to an image several times with varying sigma. This will result in a succession of blurred images, getting blurrier the more times the gaussian filter is applied (the greater the sigma). In between blurred images, the frames are subtracted resulting in a difference of Gaussians and are grouped by octave (doubling the value of σ is an octave). As a feature enhancement algorithm, the difference of Gaussians can be utilized to suppress high-frequency spatial information, thus suppressing noisy data, increasing the visibility of edges and other details presented in a digital image. This means that homogeneous regions will likely be suppressed as well, leaving us with the outlines of an image. See Figure 4.4d.

Interest points in SIFT features, identify pixels that are local extrema of difference of Gaussian filters, at different scales. In other words, every feature detected using SIFT is a local maxima or minima in a scale of blurred images created by applying successive Gaussian filters into an image. Therefore, after obtaining the difference of Gaussians, each pixel will be analyzed across scales in order to check if it is a local maxima or minima, being compared to its 8 neighbors (in its frame), plus 9 correspondent neighbors in a neighboring cell. If a pixel is a local extrema, then it is selected as a possible keypoint candidate. After this it'll be analyzed further and discarded if need be. For example, local extrema that have low-contrast

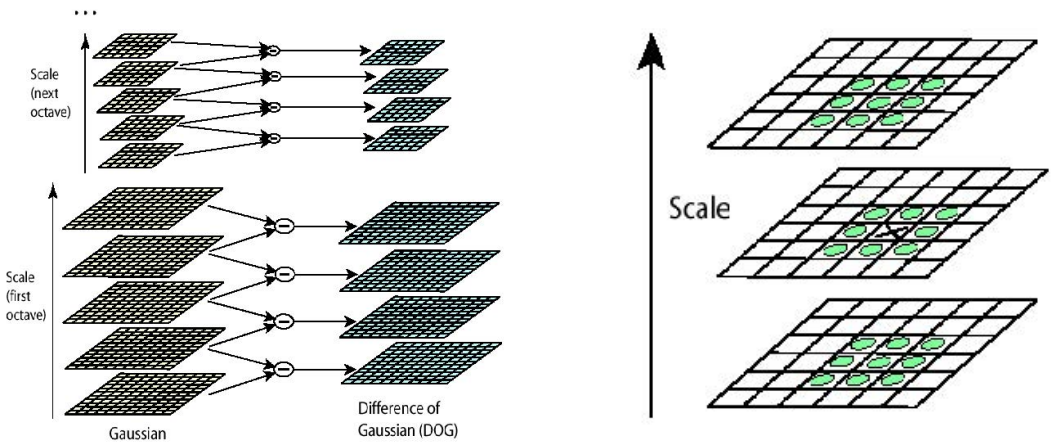
(a) Original Image.

(b) Gaussian Filter ($\sigma = 1$).

(c) Gaussian Filter ($\sigma = 2$).

(d) Thresholded result of Difference of Gaussians.

Figure 4.4: Example of Difference of Gaussians



(a) Difference of Gaussians, organized by octave..

(b) Checking for local extrema.

Figure 4.5: Difference of Gaussians.

or are located alongside an edge, are removed. Figure 4.5a and Figure 4.5b illustrate this process.

After obtaining a keypoint candidate, the need to give it an orientation arises, so that the feature can be analyzed in the same perspective, even if the image rotates (rotation-invariant). In order to accomplish this, a gradient orientation histogram [45] is computed in the neighborhood of the keypoint. The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with a σ that is 1.5 times the scale of the keypoint. Peaks in the histogram correspond to dominant orientations. The keypoint will redefine its orientation according to the histogram, aligning with the maximum, and all the properties of the keypoint are now measured relative to the keypoint orientation, providing invariance to rotation.

After obtaining a keypoint and its orientation, a feature descriptor will be created as a set of orientation histograms on 4×4 pixel neighborhoods. The orientation histograms aren't trying to orientate the feature (since that has been accomplished) but rather describe the pixels around it, so it can be compared in the future with other detected keypoints.

Histograms contain 8 bins each, and each descriptor contains an array of 4 histograms around the keypoint. This leads to a SIFT feature vector with 128 elements. This vector is normalized to enhance invariance to changes in illumination.

4.2.2 SURF

The SURF method is similar to SIFT and was developed by H. Bay, Ess, A., T. Tuytelaars and L. Van Gool [42].

Instead of using the difference of Gaussian, this detector is based on the Hessian matrix, which essentially describes the local curvature of a function. In SURF it is used the determinant of the Hessian matrix for location and scale selection.

This method makes the detector faster, since in DoG a gaussian filter is applied to the output of another gaussian filter, in order to reach the next level in the pyramid, whilst in Hessian, the scale space is analyzed by upscaling the filter size rather than iteratively reducing the image size.

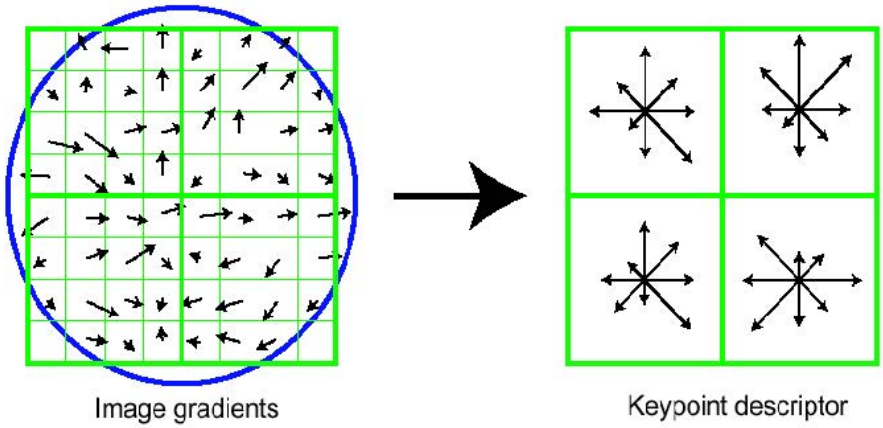


Figure 4.6: Local Orientation Histograms.

In order to obtain the orientation of the keypoint, a Haar-wavelet response is calculated, in x and y directions (rather than orientation histograms), in a circular shaped neighborhood of the interest point.

For the extraction of the descriptor, it is needed to construct a square region centered in the interest point and oriented accordingly to the orientation described. This region is separated in four sub-regions in order to keep spacial information, and each sub-region will be analyzed in order to obtain four values that describe the intensity structure, which will result in a 64 element descriptor for all 4×4 sub-regions.

4.2.3 Obtaining Overall Angular Position and Velocity

In order to obtain the orientation of the camera, it is needed to know how the features moved between frames. Specifically, the rotation that occurred between matched features in two consecutive frames. The first step is to do a matching between the features of each frame [46, 47]. The matching is, typically, a minimum euclidean distance comparison, where the descriptor from every interest point in frame i and frame $i+1$ will be compared.

After obtaining the matched points, a geometric transformation is estimated using the MSAC (M-estimator SAmple Consensus) algorithm [48,49], obtaining a transformation matrix of which the angular rotation that occurred is extracted. From this rotation, and knowing the previous angular position (θ_{i-1}), the current angular position is obtained (equation 4.2). Knowing the time of acquisition from each frame (which, using ROSTopics (see chapter 2.2) are known), calculating the angular velocity is possible (equation 4.3).

$$\theta_i = \theta_{i-1} + \Delta\theta[^\circ] \tag{4.2}$$

$$\dot{\theta}_i = \frac{\Delta\theta}{t_i - t_{i-1}}[^\circ.s^{-1}] \tag{4.3}$$

Where:

θ_i is the angular position at instant i $\dot{\theta}_i$ is the angular velocity at instant i t_i is the time at instant i $\Delta\theta$ is the angular position variation between the current frame and the last.

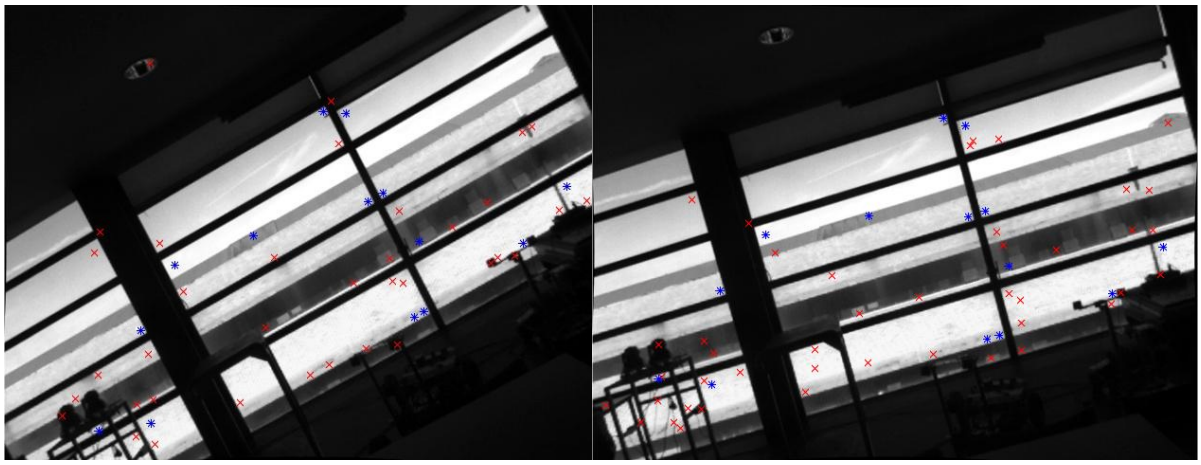


Figure 4.7: Consecutive frames with a portion of the features found (red cross) and features matched (blue asterisk). For visualization purposes, only a fraction (about 1/10) of the actual features is represented.

The first measurement of angular position and velocity in instant i was obtained using the equation 4.2 and equation 4.3. In Figure 4.7, it can be seen a portion of the features detected in a given frame $i-1$ and i and the matched ones between both frames. The first measurement is the rotation that the keypoints in $frame_{i-1}$ suffered in order to match the $frame_i$.

4.2.4 Obtaining Feature Angular Position and Velocity

After obtaining the first rotation measurement, it would be helpful to calculate the angular position of each feature and compare it to the matched feature in the previous frame. The purpose was to see how much the feature angular position had changed between frames and compare it with the estimated geometric transformation calculated previously.

The angular rotation seemed random from feature to feature. Three points are needed in order to calculate an angle. To calculate a feature angle, the points used were:

1. Feature to be calculated.
2. Center of the frame.
3. Point that, with point 2, describes a parallel line with the x-axis.

The problem was that, when trying to calculate the angle between the feature, the center of the image in different frames didn't coincide with the same point in real-life. To solve this problem, a solution was conceived that relies on calculating the centroid of a set of (matched) points, and use that point as the second point for angle calculations. This approach worked. Using two matched features can solve this problem as well.

The second problem was related to the location of the features in each quadrant. Sometimes happened for a given feature to change quadrant between frames. This leads to a huge rotation of the feature when calculating (for example 350° clockwise, when the feature had in fact moved only 10° anti-clockwise). The solution to this problem is quite simple: adding or subtracting 360° depending on the sign of the rotation calculated will solve this.

angle_tol	0.00	0.50	1.00	1.50	2.00	2.50
Visual MeasuresTaken	2	5	14	22	30	35
Inertial Data	1.52					
Blob Data	1.56					
Visual Data	1.62	2.05	2.64	2.34	3.43	3.50

Table 4.1: Comparison Between the Output of Inertial Data and Visual Data, using blob or feature detection.

After calculating the rotation of each feature, between each frame, the result was compared to the estimated geometric transformation and, if the difference between them was less than a given threshold (defined by the user), the calculated value would be deemed as acceptable. This means that the higher the threshold, the more measures will be accepted and therefore more measures will be fed into the Kalman Filter later on. However, with high thresholds, comes an increase in error. This threshold is called *angle_tol*. To better illustrate this method, a block diagram (Figure 4.8) is presented.

At this point, it is possible to analyze how the different methods differ when regarding their outputs. The formula 4.1 was used as comparison between the ground truth and the data obtained from the sensors. The results are as shown in table 4.1.

As mentioned, the higher the angle_tol values, the more visual rotation measurements will be admissible by the algorithm. That can be observed in the second line of table 4.1. However, it was expected that the accuracy of the output would become increasingly low, which is happening, but not in a linear fashion and with the exception of $angle\_tol = 1.00$. Let's take $angle\_tol = 2.50$ for instance. The algorithm obtained 35 different measurements of angular position during the experiment. In order to compare that output with the ground truth, an average for the 35 measurement was made and hereafter compared. For this fact, since so many measurement were obtained, some may compensate the error of others.

Notice that the values obtained after the comparisons are the mean difference between a given set of data and the ground truth, meaning that the lower the values, the more accurate the set of data is. In this experiment, it can be observed that the inertial data has the better output, followed by the blob detection method, followed by the feature detection method, in which the accuracy drops as the angle_tol increases. The results are compatible with what was expected.

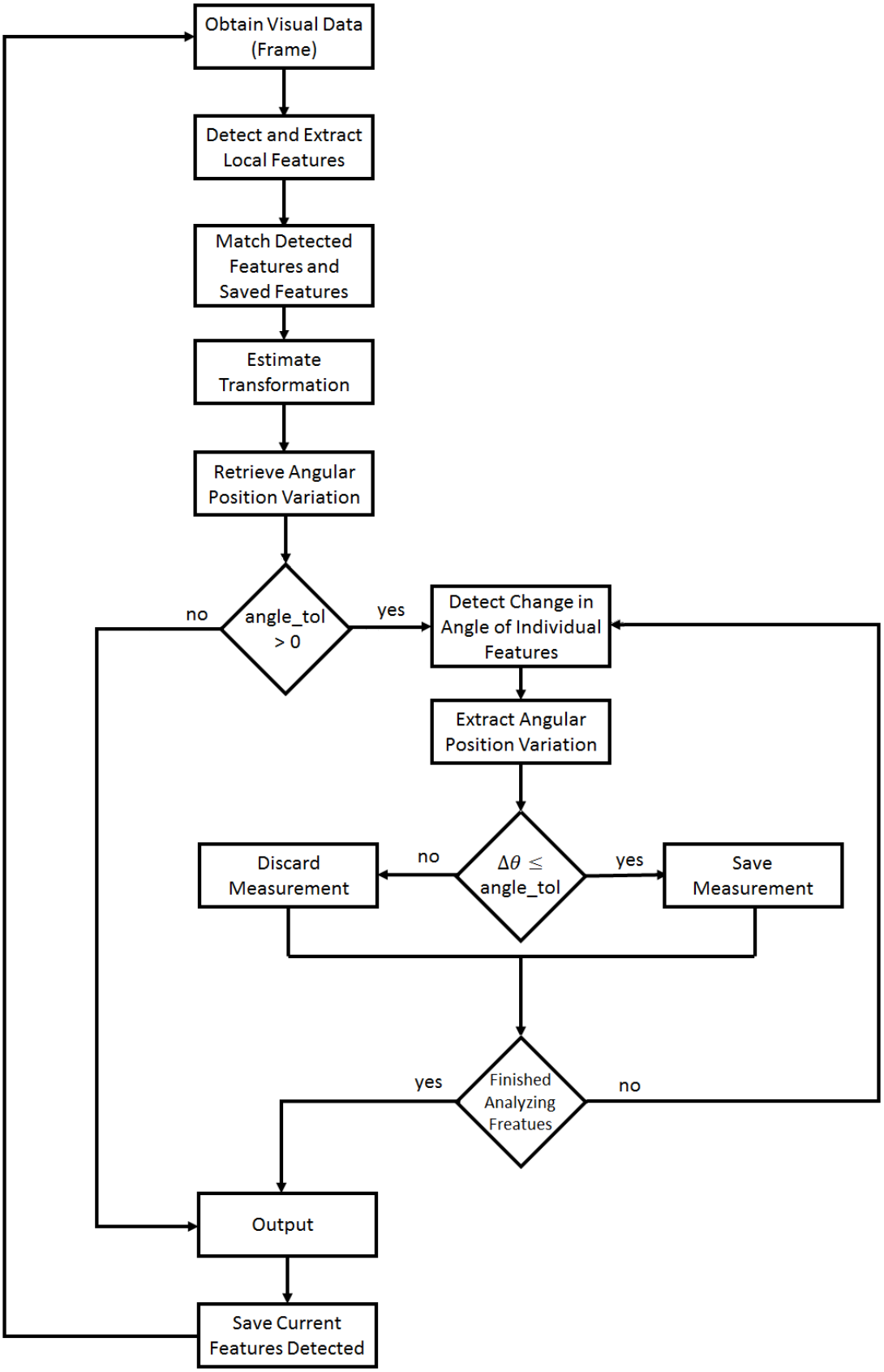


Figure 4.8: Block Diagram for local feature detection method.

Chapter 5

Merging Data Using Kalman Filter

In this chapter, it will be explained how the merging of visual and inertial data occurs. This merging is necessary in order to answer the question proposed by this dissertation, which is, will the merging of inertial and visual data improve the accuracy of the output when compared to the original individual data? Since the visual measurements can be obtained without further cost to the humanoid platform (the camera is already installed), it would be interesting to have an improvement in data accuracy. In order to achieve this merging, the Kalman Filter tool was chosen, since it allows the merging of the data based on the number of measurements that it receives, while eliminating noise.

5.1 Kalman Filter Tool

The Kalman Filter [50], also known as *linear quadratic estimation*, is a powerful statistic tool created by Rudolf Kalman based on linear algebra, which intends to estimate the value of a set of state variables. In order to do that, the filter does an estimation of the values that it is expecting, based on a model that is given, which describes the behavior of the system. The filter also includes the measurements taken by the system and compares them with what was expected, in order to know the legitimacy of the input. This tool is particularly powerful in eliminating noise in the measurements, especially if the system behavior is known and linear. However, as can be observed, most of the systems in nature are not linear. There are alternatives for models that do not behave in a linear fashion [51].

The Kalman Filter applied to the measured data will be presented in this chapter The objective is the determination of the angular position and velocity in x-axis, y-axis and z-axis. In order to better understand the Kalman Filter tool and obtain the most accurate output possible, several approaches were tried. After obtaining the inertial data and visual data, the measurements of each are compared with the ground truth, which will serve as the *control group*. Then, the data will be fed into two different Kalman Filter models, individually. In the first model, the angular acceleration will be considered zero at all times, since it can't be measured. In the second model, the angular acceleration will be estimated based on the measured values of angular velocity. Thus, obtaining four different outputs that will be compared with the ground truth individually. Finally, the original inertial and visual data will be fed into the two different Kalman Filter models, resulting in two more sets of data, that will then be compared with the ground truth. The purpose is to compare the accuracy of the outputs for each model and discover which one is the best. This trials were created

under two different background scenarios and room illumination, to test the robustness of the methods used.

5.2 Kalman Filter Model for Inertial Data

Regarding the model used for the inertial data, the state variable matrix (what will be tracked) and which model equations will better describe these state variables must be known. In this case, the most important variables are the angular position of each axis. Knowing the orientation in each axis, it is possible to know the overall orientation of the robot. It is also important to keep track of the angular velocity for two reasons. First, the more accurate the angular velocity is, the more accurate the orientation (angular position) will be. On the other hand, knowing the angular velocity will describe the movement of the robot in case of falling, i.e., if an angular velocity is positive in the x-axis, the robot is falling to the right, or the same angular velocity in the y-axis describes the robot as falling forward.

The node *"complementary_filter"* has for output the angular position, angular velocity and linear acceleration. The last is not needed, but the first two can be used as direct measures from the node. Knowing what to filter (state variables), and what to measure, it is time to write the equations that best describe the model.

Since the behavior of the humanoid is unknown, this is not a simple task. The humanoid may fall in different directions with different velocities or it can even begin to fall in a direction and then get back up. This kind of behavior is highly unpredictable. So an approximation has to be made. Imagine that the robot is in angular motion with constant acceleration. This kind of model is easily written, as was seen in equations A.1, A.2 and A.3. Therefore it is possible to write the same equations for the angular motion. Since the angular acceleration is not known, it can be assumed as process noise. This way, an overall model for any behavior of the robot is described. Although it may be uncommon, it is also possible to calculate the acceleration in each instant of time, since the last values of angular velocity are known, as well as the difference in time between them.

$$\theta_k = \theta_{k-1} + \dot{\theta}_{k-1}\Delta t + \frac{1}{2}\ddot{\theta}_{k-1}\Delta t^2[^\circ] \tag{5.1}$$

$$\dot{\theta}_k = \dot{\theta}_{k-1} + \ddot{\theta}_{k-1}\Delta t[^\circ.s^{-1}] \tag{5.2}$$

$$\ddot{\theta}_k = \ddot{\theta}_{k-1}[^\circ.s^{-2}] \tag{5.3}$$

Where:

θ_k it's the angular position in instant k
$\dot{\theta}_k$ it's the angular velocity in instant k
$\ddot{\theta}_k$ it's the angular acceleration in instant k
Δt it's a temporal iteration

As observed, these equations are very similar to the ones describing the linear motion (for better understanding of this equations, read Appendix A).

$$\begin{bmatrix}\theta_k \\ \dot{\theta}_k\end{bmatrix} = \begin{bmatrix}1 & \Delta t \\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta_{k-1} \\ \dot{\theta}_{k-1}\end{bmatrix} + \begin{bmatrix}\frac{\Delta t^2}{2} \\ \Delta t\end{bmatrix}\begin{bmatrix}\ddot{\theta}_{k-1}\end{bmatrix} \tag{5.4}$$

$$\begin{bmatrix}\theta_k \\ \dot{\theta}_k\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta_k \\ \dot{\theta}_k\end{bmatrix} \tag{5.5}$$

Having this, it is just a matter of defining the process (w_k) and measurement (v_k) noise, which differ from experiment to experiment, since they are consequences of the type of measurement devices used, or the model that was chosen. There is not a simple way to accurately guess their values, thus, simulating becomes a useful tool. Nevertheless, with some experience, it can be guessed how the magnitude of a value may affect the model after some simple trials.

Now, the Kalman Filter is projected for a 1D angular motion. Transforming it into a 3D one will better describe the behavior of the humanoid, and can be done easily, since the equations are all the same, changing only the size of the matrices. Furthermore, in this case, a given dimension (take x for example) will not affect the others, resulting in equations 5.6 and 5.7.

$$\begin{bmatrix}w_k \\ p_k \\ r_k \\ \dot{w}_k \\ \dot{p}_k \\ \dot{r}_k\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}w_{k-1} \\ p_{k-1} \\ r_{k-1} \\ \dot{w}_{k-1} \\ \dot{p}_{k-1} \\ \dot{r}_{k-1}\end{bmatrix} + \begin{bmatrix}\frac{\Delta t^2}{2} & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t\end{bmatrix}\begin{bmatrix}\frac{\dot{w}_{k-1} - \dot{w}_{k-2}}{\Delta t_{k-1}} \\ \frac{\dot{p}_{k-1} - \dot{p}_{k-2}}{\Delta t_{k-1}} \\ \frac{\dot{r}_{k-1} - \dot{r}_{k-2}}{\Delta t_{k-1}}\end{bmatrix} \tag{5.6}$$

$$\begin{bmatrix}w_k \\ p_k \\ r_k \\ \dot{w}_k \\ \dot{p}_k \\ \dot{r}_k\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}w_k \\ p_k \\ r_k \\ \dot{w}_k \\ \dot{p}_k \\ \dot{r}_k\end{bmatrix} \tag{5.7}$$

Where:

w is the rotation in x-axis
p is the rotation in y-axis
r is the rotation in z-axis

5.3 Visual Data and Data Merging

At this point, the model in Kalman Filter is created and the inertial data is integrated within it. In order to complete the data merging process, the measurements obtained by vision systems need to be fed into the Kalman Filter. For that it is needed to understand what data

is being received. An important detail to understand about the Kalman Filter method, is the meaning of the equation $y = C.x$. This equation essentially links the measurements obtained with the state variable matrix. What this means is that, the C matrix essentially describes the relation between a given variable present in the state variable matrix (x) and the measurements it produces in a given sensor (y). Therefore, adding the visual data obtained and then relate it to the state variable matrix is in order. In this case, the sensors output is the angular position and velocities for the three axis, and the camera output is variable. It is possible to obtain more than one measurement from the image, therefore the presented formula 5.8 has n measurements. In these experiments, the camera was aligned with the y-axis of the sensor. Since the robot obeys to the physical laws of motion, it is not necessary to adjust the model definition. The simplicity of this method results in an elegant solution to the merging data problem.

$$\begin{bmatrix} w_k \\ p_k \\ r_k \\ \dot{w}_k \\ \dot{p}_k \\ \dot{r}_k \\ C_{1_k} \\ \dot{C}_{1_k} \\ C_{2_k} \\ \dot{C}_{2_k} \\ ... \\ C_{n_k} \\ \dot{C}_{n_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ . & . & . & . & . & . \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_{x_k} \\ \theta_{y_k} \\ \theta_{z_k} \\ \dot{\theta}_{x_k} \\ \dot{\theta}_{y_k} \\ \dot{\theta}_{z_k} \end{bmatrix} \tag{5.8}$$

Regarding visual data, the only method used to obtain the angular orientation and velocities was the local feature detection method, in this case SURF (page 40). Despite the fact that blobs should be used to improve visual data whenever possible, such approach was not used. The reason is that, to compare the effectiveness of this method in different situations, where in some, blob extraction is not possible, it is advised to use the same method of data treating in order to understand the output of each experiment and how may the surrounding conditions affected the measured data. Thus, it was only used the feature extraction method in both the experiments. Notice that using the blob extraction is still a good approach whenever possible, but in this case, for comparing purposes alone, it was discarded.

As was said, there are some parameters that will allow us to detect more or less features during the feature extraction phase. This is due to the personalized approach on the matter. This approach was purely experimental and by no means is a standard or obligated protocol when treating visual data. Nevertheless it may be important to understand this approach. This parameters are mainly *angle_tol* and *angle_distance*, where the first one sets a maximum difference between the angle of two different features, when compared to the reference value previously calculated and the last is essentially the difference in euclidian distance that is accepted to change between two features in two consecutive frames, i.e., the distance between two features in one frame and in the next can't differ more than this parameter, in order for a given feature to be accepted. This two parameters help the algorithm to discard some false matches within the feature detection and matching phase.

(a) Base frame for first experiment. (b) Base frame for second experiment.

Figure 5.1: Background images from the experiments.

5.4 Experiments

In order to obtain more data, two experiments were realized and their original data treated for several values of *angle_tol*, with the purpose of trying to find some relation between the number of measures retrieved and the accuracy of the final result.

Formula 4.1 was used to compare data with the ground truth. Concerning the experiments background (source for visual data), as was said, two experiments with very different lighting conditions were produced. Figure 5.1a and Figure 5.1b correspond to the base frame (frame where the orientation of the camera is zero degrees) of the first and second experiment. As can be seen, the first experiment has better lighting conditions as well as features for blob extraction, though this method was discarded as explained before. Since the background has such different properties from experiment one to two, it is expected to be found a greater error in the experiment with the worse conditions (in this case the second). Due to mechanical problems of the piece developed for the experiments (see Appendix B), which has the purpose of joining all sources of data and connecting them to the FANUC's end-effector, the inertial data is slightly worse as well. The reason is that the experiments were done some months apart and between the experiments, the piece was slowly worn out, resulting in a tiny gap where the screws were to be attached to the end-effector. When a sudden change in direction (or acceleration to be more precise), the inertia of the system caused the piece to shake a bit. Since the purpose of this work is to merge data in order to improve noisy or inaccurate sensors, importance to this fact was not given. The results are presented in tables 5.1 to 5.4 and in Figure 5.2 to 5.5.

	angle_tol	0,00	0,50	1,00	1,50	2,00	2,50
	Visual Measures Taken	2	5	14	22	30	35
	Inertial Data	1,52					
	Visual Data	1,62	2,05	2,64	2,34	3,43	3,50
Kalman Filter	Inertial Data	1,41					
Acceleration	Visual Data	1,40	1,90	2,14	1,58	2,71	1,17
Not Calculated	Both Data	1,05	1,25	1,44	1,28	1,78	1,24
Kalman Filter	Inertial Data	1,41					
Acceleration	Visual Data	3,46	2,74	2,46	1,76	2,72	1,18
Calculated	Both Data	1,06	1,27	1,44	1,28	1,78	1,22

Table 5.1: Data from experiment 1 (No Error).

	angle_tol	0,00	0,50	1,00	1,50	2,00	2,50
	Visual Measures Taken	2	5	14	22	30	35
	Inertial Data	9,21					
	Visual Data	1,62	2,05	2,64	2,34	3,43	3,50
Kalman Filter	Inertial Data	5,63					
Acceleration	Visual Data	1,40	1,90	2,14	1,58	2,71	1,17
Not Calculated	Both Data	1,71	1,98	2,07	1,67	2,74	1,39
Kalman Filter	Inertial Data	5,27					
Acceleration	Visual Data	3,46	2,74	2,46	1,76	2,72	1,18
Calculated	Both Data	1,57	1,92	2,03	1,61	2,71	1,28

Table 5.2: Data from experiment 1 (With Error).

	angle_tol	0,00	0,50	1,00	1,50	2,00	2,50
	Visual Measures Taken	2	3	6	16	22	27
	Inertial Data	2,80					
	Visual Data	6,43	4,81	5,44	5,63	4,59	5,90
Kalman Filter	Inertial Data	2,61					
Acceleration	Visual Data	5,92	4,48	5,05	5,24	3,83	4,39
Not calculated	Both Data	3,08	2,81	3,26	3,30	2,77	3,17
Kalman Filter	Inertial Data	2,67					
Acceleration	Visual Data	7,26	5,65	5,64	5,75	4,47	4,49
Calculated	Both Data	3,14	2,87	3,30	3,35	2,82	3,18

Table 5.3: Data from experiment 2 (No Error).

	angle_tol	0,00	0,50	1,00	1,50	2,00	2,50
	Visual Measures Taken	2	3	6	16	22	27
	Inertial Data	9,18					
	Visual Data	6,43	4,81	5,44	5,63	4,59	5,90
Kalman Filter	Inertial Data	4,76					
Acceleration	Visual Data	5,92	4,48	5,05	5,24	3,83	4,39
Not calculated	Both Data	4,57	3,56	4,55	4,63	3,47	4,34
Kalman Filter	Inertial Data	4,07					
Acceleration	Visual Data	7,26	5,65	5,64	5,75	4,47	4,49
Calculated	Both Data	4,49	3,52	4,50	4,57	3,34	4,25

Table 5.4: Data from experiment 2 (With Error).

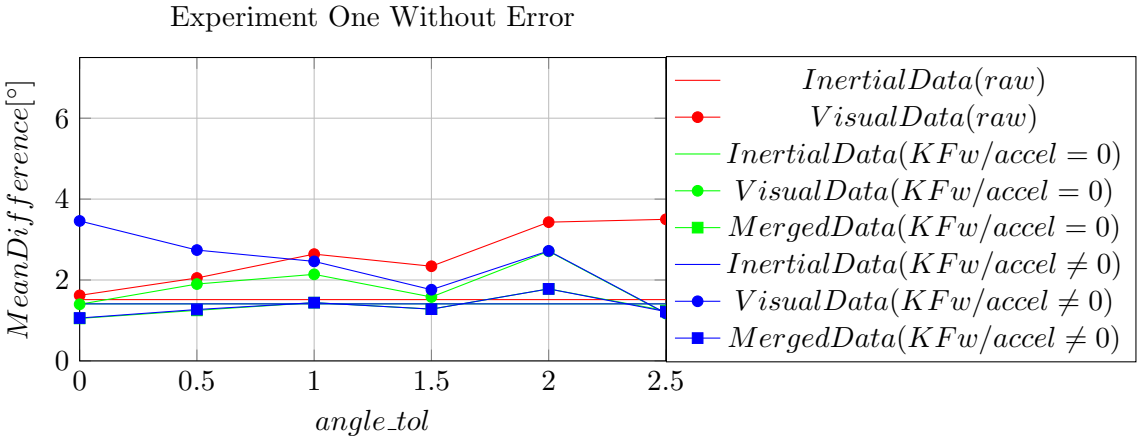


Figure 5.2: Accuracy of output when compared with input parameter angle_tol, in experiment 1 without error.

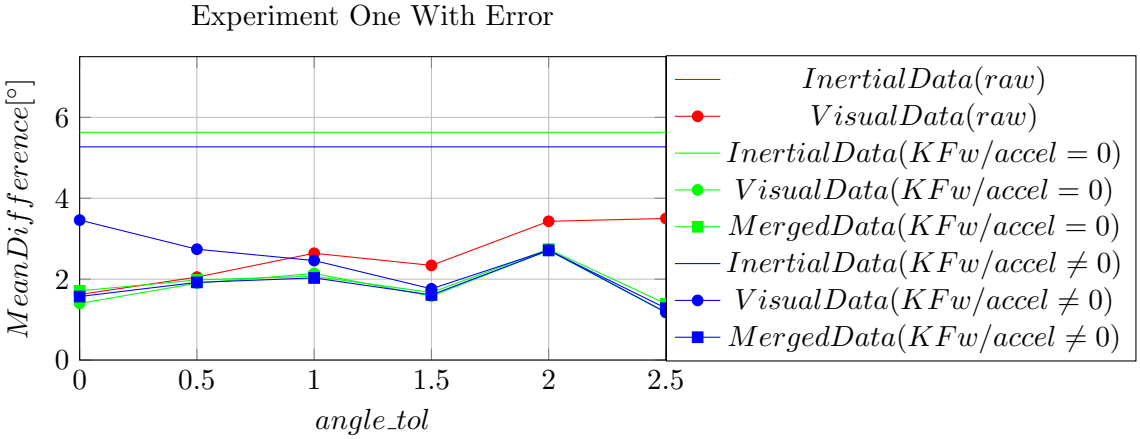


Figure 5.3: Accuracy of output when compared with input parameter angle_tol, in experiment 1 with error.

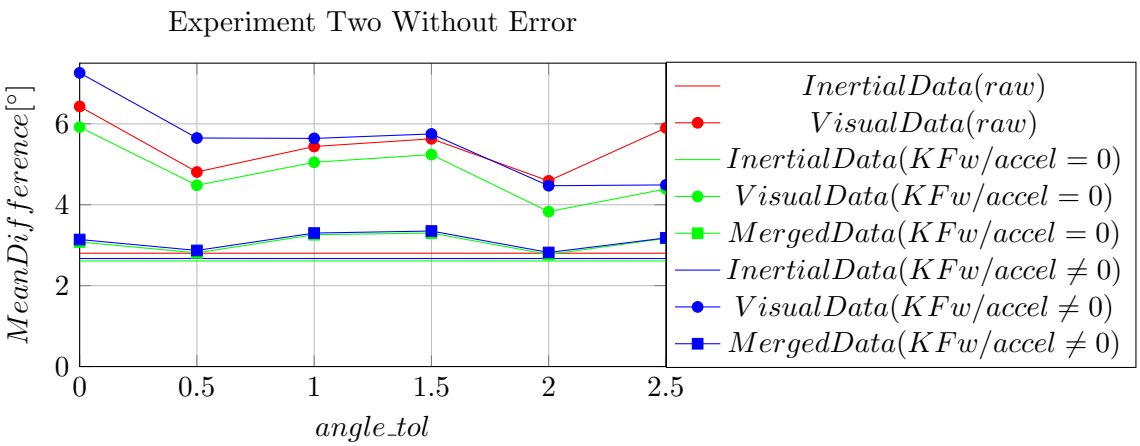


Figure 5.4: Accuracy of output when compared with input parameter angle_tol, in experiment 2 without error.

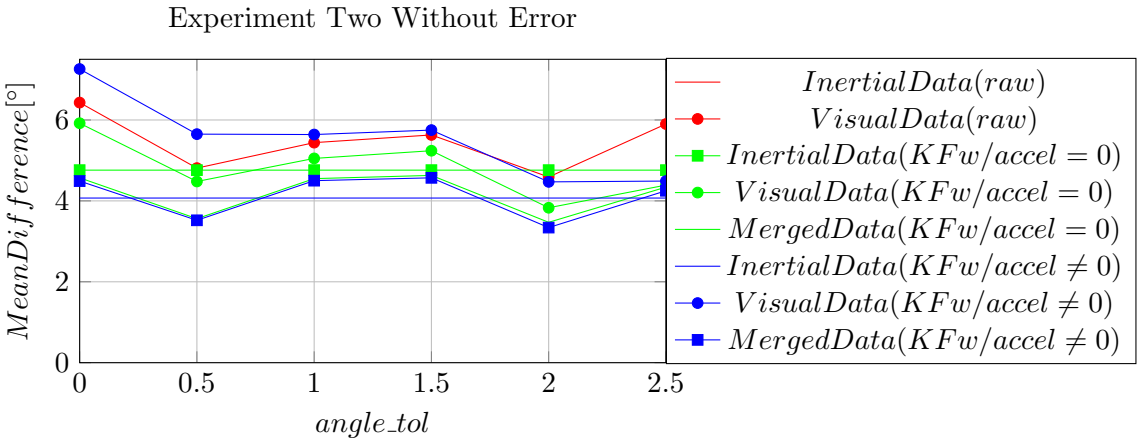


Figure 5.5: Accuracy of output when compared with input parameter angle_tol, in experiment 2 with error.

As can be observed in the tables [5.1-5.4] and Figures [5.2-5.5], the results may behave somewhat different from situation to situation. However, some properties remain the same.

5.5 Results

5.5.1 Experiment One

In the first experiment, with no noise added, it can be observed that, when acceleration is equal to zero in every iteration, the accuracy of the merged data is improved, though the improvement may be discarded (since the output obtained from calculating the acceleration is in the order of 1%). Since there is no difference in choosing to calculate the acceleration or not concerning the output, it is believed that it should not be calculated, since less calculations will lead to less processing power. The calculations made were timed, in order to understand how they affect the processing time needed, but the script is so fast (around $0.05s$) that these measurements are highly affected by the computer processing power, such that no consistent

results were obtained. However this can be easily explained by the fact that more tolerance in angle measures means an increase in overall error, although more measures can contribute for a more precise evaluation of the state variables. In other words, the greater the tolerance, the greater the inaccuracy, even though, more measures are obtained. When looking at the visual data (no treatment), it can be seen that the error increases with the tolerance. This value is calculated using the average of all results gathered, for example, $angle\_tol = 1.50$ has 22 measures for angular position and velocity, which the average will be compared to the ground truth. Regarding the Kalman Filter results, in Visual Data, the input is not an average. All the measurements obtained are used as inputs of the filter and it will calculate the angular position and velocity, based on those inputs. It can also be observed that the *angle_tol* value gives the best output when the merging data is 0.00 followed by 2.50, which is consistent with the explanation given. In the first case, a low number of measures is given, but they are as accurate as possible and in the last case, a less accurate input is given, but the high amount of measures make up for that fact. These observations are both valid when the acceleration is calculated or discarded at every iteration.

When the error is added to the inertial data, the results differ a bit. In this case, calculating the acceleration at each step will yield better results. The difference in results is a little bit higher, but they can't be seen as relevant. It can be seen that the average error difference between both cases is about 0.07°. The greatest difference is that the best outputs are related to *angle_tol* of 2.50, 0.00 and 1.50. In this case, the difference between the output in the first and second best results is 0.3° which may be seen as relevant (depending on the accuracy needed). Again, this high amount of inaccurate measures submitted to the Kalman Filter will create better output than a low amount of high accurate data. Notice that in this case, since the inertial data is highly inaccurate, the filter must rely on the Visual Data. Therefore, more measurements will yield more importance in the filter, i.e., since the filter will acquire more data at each instant from the visual sensor rather than from the inertial sensor, the filter will give more importance to the visual data. Finally, it can be observed that, in some cases, the visual data performs best applied directly to the Kalman Filter, rather than when merged with the inertial data. However, it is still an advantage to merge both data. The reason is that Visual Data takes longer to acquire than the inertial data, but the Kalman Filter is constructed in a way that will preserve all measurements, meaning that the output will have as many values in time has the data with more measurements, i.e., if the inertial data has 10 measurements per second, whilst the visual data has only 3 (which is approximately true in both cases), the Kalman Filter output will output 10 measurements per second, meaning that even if the visual data has less error when comparing to the merged data, the merged data has more detailed information over time. This also explains the increasing error, since when there isn't visual data, the filter only computes the inertial one.

It is important to remind that it is possible that the base value for angular position and velocity ($angle\_tol = 0$) may have an error, so using more measurements may dissipate that error.

5.5.2 Experiment Two

Concerning the second experiment, the first aspect to notice is that the visual input is much worse when compared to the first experiment. The lighting conditions explain this fact (see Figure 5.1 in pag. 49). Just as it happened in the first experiment, the filter performs best when acceleration is not calculated. It can also be observed that the data merging performs

worse than the inertial data (with exception for $angle\_tol = 2.00$) in the Kalman Filter when acceleration is not calculated. In this case the data merging is a waste of processing power, since the images have to be treated and fed into a Kalman Filter along with the inertial data (that is ready to use), obtaining an average accuracy of 0.03°. Nevertheless, it's important to remember that the inertial data is quite accurate, therefore trying to improve it with improper data will likely ruin it.

In this particular experiment, when error is added to the inertial data a situation of inadequate sensory data (both inertial and visual) emerges. Since the visual data is not accurate enough to describe the robot behavior, the inputs of the system can't be trusted individually. This experiment illustrates a situation where both sensors are unreliable by themselves. However, as can be seen in table 5.4, the output from the filter surpasses the output from the sensors (inertial and visual). As seen in the first experiment, when the error is added, the Kalman Filter which computes the acceleration at each instant, performs better.

In conclusion, merging data may or may not yield advantages. There are numerous variables that contribute to the amount of accuracy won or lost when using the filter. Each sensor accuracy, the number of measures and the construction of the filter are examples of variables that affect the output. However, it is proven that merging data may improve the original data. As a final note, it is important to stress that the model used in this dissertation was not describing the system motion but rather the laws of that motion, making it a general model, which in turn, affects the output. Nevertheless the results prove that merging different source of data is a solution to inaccuracy problems and therefore, should be used whenever possible.

Chapter 6

Conclusions

Due to the need of a balance algorithm for the PHUA, this thesis was elaborated in order to create a model which would merge different sets of data, to obtain a final set of state variables, that would describe the system with improved accuracy. To accomplish this goal, several trials were done in order to understand the meaning of the inertial and visual data and, later on, more experiments were created in order to implement a Kalman Filter model to improve the results of the original data.

The proposed experimental approach performed extremely well, regarding the creation of experiments and retrieving reliable ground truth from them. The use of the industrial manipulator proved to be an asset, since it was able to repeat the same experiment every time that it was needed. The obtained ground truth was essential for obtaining valid and objective conclusions. This tool also allowed the communication between the desktop computer and itself, which allowed to run the node *fanuc_control*, which retrieved the ground truth from the manipulator end-effector. Therefore, as a contribution to future work, it is advised to use the industrial manipulator as the ground platform for practical experiments, whenever possible.

The trials were successful, proving that it is possible to use different sources of measurements in order to merge and improve them into an overall set of state variables that describe the behavior of the system of study, as shown in chapter 5.3. As it was mentioned, the merging of different sets of data using the Kalman Filter, is possible and will improve the output accuracy, surpassing many times the accuracy of the original data. This means that, with the same cost (weight and/or monetary cost) the system is able to increase its perception abilities. The Kalman Filter model may need adjustments or several trials before it can be fully implemented in order to find out its best configuration (namely the model and measurement co-variance matrix definition), or if the filter should or should not calculate the accelerations at each iteration. Notice that *angle_tol* variable was a tool personally created for this project, and may not (probably should not) be used in order to obtain visual data. Nevertheless, another conclusion was reached while using this variable which is that, the Kalman Filter model is affected by the number of measurements fed into the filter per iteration.

The tools used served their purpose. The fire-wire camera used was able to retrieve visual information in experiments with poor lighting conditions, which indicates that the camera achieved its purpose within the system. As seen, the inertial sensors also performed well, giving us an accurate angular position (even without treatment), which improved after being merged with the visual data.

Even if the merging and improvement of data is possible using the tools described in this

dissertation, a study should be performed in the future, to compare the gain in accuracy versus the lost in computational power, with the purpose of analyzing the applicability of this approach in the humanoid specifically.

In summary, some conclusions may be taken from this dissertation, such as:

1. The industrial manipulator is a tool which should be used for creating experiments.
2. The tools and equipment used satisfied the requirements of this dissertation.
3. The Kalman Filter tool is extremely useful and allows to merge different sets of data.
4. The merged data will normally be more reliable than the original data provided by the sensors.
5. The higher the amount of measures per iteration, the more accurate the output of the filter becomes.
6. The improvement of accuracy tends to be greater in cases with worst sensory devices.

6.1 Future Work

In conclusion, this approach was validated by the results, and the next step is to implement this method in a real-life situation with real-time calculations. In order to accomplish this, a node that is based on this Kalman Filter model is needed. That node will have to overcome some difficulties. One of the greatest is the synchronization of the inertial and visual data when being processed by the Kalman Filter. As it was mentioned before, the data obtained from the camera, sensors and industrial manipulator was firstly acquired and then, using the ROSTopics time-stamps, synchronized. Only after this adjustment, were the calculations applied. In real-time applications this is not possible, since the data must be analyzed as it is read. Dealing with the delay that is inherent in visual processing, will surely be a challenging step to accomplish. The system may need some improvements and modification in order to solve this limitation. Maybe (this is speculation) there will be a need to create a Kalman Filter model that is able to "look back in time" to correct the inertial values every time that new visual data is received. In order to solve this problem, there may be a need of creating a node that analyzes the instant position and velocity and tries to guess what the movement will be in a given interval of time. This node would constantly try to guess the orientation obtained from the visual data in the future and merge it with the current inertial data (this could be the solution to the data delay). When the visual data arrived, the difference in what was currently measured and previously estimated could adjust the current state variable matrix somehow.

When it comes to the balance itself, there is still no response from the humanoid, even if it knows that it is off-balance. A node should be created that would have as output a certain movement of a robot for a given input (this input would be the output of the Kalman Filter model created in this dissertation project). In order to accomplish this, the "rigid" segments of the body should have its orientation known. Having the orientation of the segments and comparing the orientation of each segment with the rest of the segments would give a rough idea of how the robot is oriented in space. However, the inertial sensors alone may prove to be inaccurate in giving us this model. A direct consequence of this work is that this method can be applied to solve this problem. In this case, one could elaborate a Kalman Filter model

that would receive the inertial data and servomotor angular position, returning an improved estimate of the segment position in space, through Forward Kinematics (knowing the angular position of the servo-motors, the end-effector position and orientation is found.

Other consequence of this project may be the creation of an overall set of state variables that would describe the overall orientation of the humanoid, rather than just the head. In this case, since only the head motion is being measured, the state variable matrix has the size of [6x1]. Using all the measured segments in a single state variable matrix, a matrix with size [54x1] could be obtained, where 54 is the number of sensors (9) multiplied by 6 which describe the angular position and velocity in 3D space.

A final task must be accomplished, not in order to complete or aid this theme, but in order to facilitate future experiments using the industrial manipulator robot. During the elaboration of this thesis, a node, *fanuc_control*, was created but not finished. The node works but it is not yet robust and intuitive enough to work with. People who don't understand the mechanics of this node will have difficulties using it. Since there were some time limitations, finishing this node was not a priority. The node is working, but some functionalities must be added. For example, there are some missing functions in the node, and even though they are not as important as the ones currently in place, they should be added. There is also a need to read a list of commands from a *".txt"* file, or allow the node to create the said list of commands, using a terminal and/or using a GUI (Graphical User Interface) that allows an easy understanding and viewing of what is being done. There is also a need to be able to *kill* the FANUC's robot program if it is running on a loop, remotely. All of this tasks are possible, but they will need a massive amount of time in order to become completed. Currently, the best approach for using this node, is creating a program within the industrial manipulator that is called remotely, using the computer. This is a difficulty that must be overcome.

6.2 Final Discussion

Summarizing, the results of this work were in accordance to what was expected of it. The Kalman Filter model developed in this pages turned out to be a success, and best of all, there are endless applications for this type of tool. Some application may regard navigation control, especially for non tripulated vehicles, movement control, but not limited to this kind of robots, nor robots at all for that matter. The presence of the industrial manipulator also turned out to be an advantage to the project, without it, this work would lack a reliable ground truth to be based on. The design of the experiments were also aided by the manipulator, which helped to obtain inertial and visual data, that were later merged successfully, improving the accuracy beyond the individually sensors abilities.

As a final note, notice that the Kalman Filter is essentially a tool that adjusts the theoretic model of reality to the reality as measured, making it an extensible tool that can be used in different applications. Thinking of it in this way, means that there is no end for the applications of this tool. Therefore, creating a way to merge different types of data, is an improvement to the tool.

References

[1] Russon, M. (2014). Colin Angle, iRobot CEO: 'Sonny' humanoid robots too expensive to be a reality. *International Business Times UK*, Retrieved from http://www.ibtimes.co.uk.

[2] Santos, V., Moreira, R., & Silva, F. (2012). Mechatronic design of a new humanoid robot with hybrid parallel actuation. *Internetional Journal of Advanced Robotic Systems*, *9*.

[3] Estrelinha, M. (2013). Tele-operation of a humanoid robot using haptics and load sensors. *Unpublished master's thesis*, Universidade de Aveiro, Aveiro, Portugal.

[4] Barros, J. (2014). Cooperative haptics for humanoid robot teleoperation. *Unpublished master's thesis*, Universidade de Aveiro, Aveiro, Portugal.

[5] Rafeiro, T. (2013). Rede de sensores inerciais para equilíbrio de um robô humanóide *Unpublished master's thesis*, Universidade de Aveiro, Aveiro, Portugal.

[6] Rodrigues, M. (2008). Unidade de processamento e sistema de visão para um robô humanóide *Unpublished master's thesis*, Universidade de Aveiro, Aveiro, Portugal.

[7] Cruz, P. (2012). Haptic interface data acquisition system *Unpublished master's thesis*, Universidade de Aveiro, Aveiro, Portugal.

[8] Huang, C., Sue, P., Abbod, M., Jiang,B., & Shie, J. (2013). Measuring center of pressure signals to quantify human balance using multivariate multiscale entropy by designing a force platform. *Sensors*, *13*, 10151-10166.

[9] Maki, B., McIlroy, W., & Fernie, G. (2003). Change-in-support reactions for balance recovery. *IEEE Engineering in Medice and Biology Magazine*, *22*, 20-26.

[10] Horak, F., Henry, S., & Shumway-Cook, A. (1997). Postural perturbations: new insights for treatment of balance disorders. *Physical Therapy*, *77(5)*, 517-534.

[11] Kuo, A. (1995). An optimal control model for analyzing human postural balance. *IEEE Transactions on Biomedical Engineering*, *42(1)*, 81-101.

[12] Nashner, L., & McCollum, G. (1985). The organization of human postural movements: a formal basis and experimental synthesis. *Behavioral and Brain Sciences*, *8*, 132-172.

[13] Horak, F., & Nashner, L. (1986). Central programming of postural movements: adaptation to altered support surface configurations. *Journal of Neurophysiology*, *55*, 1369-1381.

[14] Dietz, V. (1992). Human neuronal control of automatic functional movements: interaction between central programs and afferent input. *Behavioral and Brain Sciences*, *72*, 33-69.

[15] Horak, F. (1987). Clinical measurement of postural control in adults. *Physical Therapy*, *67*, 1881-1885.

[16] McIlroy, W., & Maki, B. (1995). Adaptive changes to conpensatory stepping responses. *Gait and Posture*, *3*, 43-50.

[17] Allum, J., Keshner, E., Honegger, F., & Pfaltz, C. (1988). Organization of leg-trunk-head equilibrium movements in normals and patients with peripheral vestibular deficits. *Progress in Brain Research*, *76*, 277-290.

[18] McCollum, G., & Leen, T. (1989). Form and exploration of motor activity during postural control. *Journal of Motor Behavior*, *21*, 225-244.

[19] Vukobratovic, M., & Juricic, D. (1969). Contribution to the synthesis of biped gait. *IEEE Transaction on Biomedical Engineering*, *16(1)*, 1-6.

[20] Goswami, A. (1999). Postural stability of biped robots and the foot rotation indicator (fri) point. *International Journal of Robotics Research*, *18(6)*, 523-533.

[21] Hirai, K., Hirose, M., Haikawa, Y., & Takenaka, T. (1998). The development of honda humanoid robot. *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, *2*, 1321-1326.

[22] Vukobratovic, M., & Stepanenko, Y. (1972). On the stability of antropomorphic systems. *Mathematical Biosciences*, *15*, 1-37.

[23] Vukobratovic, M., & Borovac B. (2004). Zero-Moment Point - thirty-five years of its life. *International Journal of Humanoid Robots*, *1*, 157-173.

[24] Goswami, A., & Kallem, V. (2004). Rate of change of angular momentum and balance maintenance of biped robots. *IEEE International Conference on Robotics and Automation*, *4*, 3785-3790.

[25] Greenwood, T. (1965). Principles of dynamics. *New Jersey: Prentice-Hall, Inc.*

[26] Komura, T., Leung, H., Kudoh, S., & Kuffner, J. (2005). A feedback controller for biped humanoids taht can counteract large perturbations during gait. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 1989-1995.

[27] Jones, E., & Soatto, S. (2010). Visual-inertial navigation, mapping and localization: A scalable real-time causual approach. *International Journal of Robotics Research*, *30(4)*, 407-430.

[28] Konolige, K., & Agrawal, M. (2008). Frameslam: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, *24(5)*, 1066-1077.

[29] Peixoto, J., Santos, V., & Silva, F. (2016) Proprioceptive visual tracking of a humanoid robot head motion. *Lecture Notes in Computer Science*, *9730 LNCS*.

[30] Mourikis, A., & Roumeliotis, S. (2007). A multi-state constraint Kalman Filter for vision-aided inertial navigation. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 3565-3572

[31] Liu, C., & Prior, S. (2015). Computationally efficient visual-inertial sensor fusion for GPS-denied navigation on a small quadrotot. *2015 International Conference on Innovation, Communication and Engineering, Xiangtan*

[32] Furgale, P., Barfoot, T., & Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. *2012 IEEE International Conference on Robotics and Automation (ICRA)* 2088 - 2095

[33] Woodman, O. (2007). An introduction to inertial navigation. *Technical Report*, *696*, University of Cambridge, Cambridge, United Kingdom.

[34] Burg, A., Meruani, A., Sandheinrich, B., & Wickmann, M. (2004). MEMS gyroscopes and their applications. *Northwestern University.*

[35] 9 Degrees of Freedom - Razor IMU *Sparkfun*, Retrieved from https://www.sparkfun.com

[36] MinIMU-9 v2 Gyro, accelerometer, and compass (L3GD20 and LSM303DLHC Carrier) *Polulu*, Retrieved from https://www.polulu.com

[37] O'Kane, J. (2013). A gentle introduction to ROS. *Independently published.*

[38] LR Mate 200iB-200iB/3L *FANUC Robotics*, Retrieved from http://geek.nmt.edu/~bruder/final_project_files/Fanuc%20LR%20Mate_200ib_200ib_3l.pdf

[39] Cancela, R. (2007). Extensão e flexibilizaão da interface de controlo de um manipulador robótico FANUC *Unpublished master's thesis*, Universidade de Aveiro, Aveiro, Portugal.

[40] Using accelerometers to estimate position and velocity *Chrobotics*, Retrieved from http://www.chrobotics.com/library/accel-position-velocity

[41] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60(2)*, 91-110.

[42] Bay, H., Ess, A., Tuytelaars, T., & Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, *110(3)*, 346-359.

[43] Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. *Computer Vision European Conference on Computer Vision 2006*, *3951 LNCS*, 430-443.

[44] Juan, L., & Gwun, O. (2009). A comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing*, *3(4)*, 143-152.

[45] Hu, R., & Collomosse, J. (2013). A Performance evaluation of gradient field HOG descriptor for sketch based image retrieval. *Computer Vision and Image Understanding*, *117(7)*, 790806.

[46] Muja, M., & Lowe, G. (2012). Fast Matching of Binary Features. *CCRV '12 Proceedings of the 2012 Ninth Conference on Computer and Robot Vision*, 404-410.

[47] Muja, M., & Lowe, G. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *International Conference on Computer Vision Theory and Applications*, 331-340.

[48] Torr, P., & Zisserman, A. (2000). MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding 78(1)*, 138-156.

[49] Hartley, R., & Zisserman, A. (2003). Multiple View Geometry in Computer Vision. *Cambridge University Press*

[50] Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82(1)* 35-45.

[51] Einicke, G., White, L. (1999). Robust Extended Kalman Filtering. *IEEE Transactions and Signal Processing*, *47(9)*, 25962599.

Appendix A

Kalman Filter Description

Let's imagine that there is a vehicle that moves in a straight line with constant speed and there is the necessity of knowing its position and velocity at any given time. The functions that measure its movement in a single dimension are the following.

$$x_k = x_{k-1} + \dot{x}_{k-1}\Delta t + \frac{1}{2}\ddot{x}_{k-1}\Delta t^2[m] \quad (A.1)$$

$$\dot{x}_k = \dot{x}_{k-1} + \ddot{x}_{k-1}\Delta t[m.s^{-1}] \quad (A.2)$$

$$\ddot{x}_k = \ddot{x}_{k-1}[m.s^{-2}] \quad (A.3)$$

Where:

x_k it's the position in instant k
$\dot{x}_k$ it's the velocity in instant k
$\ddot{x}_k$ it's the acceleration in instant k
Δt it's a temporal iteration

In this case, the vehicle may be affected to a greater or lesser magnitude of friction, and at the same time, depending on its weight, it can have more or less inertial energy. For that reason, the formulas presented may not be 100% accurate. In order to solve this problem a GPS sensor was attached in the vehicle, which returns x_k and $\dot{x}_k$; those are the variables that can be measured. Now, the state variables matrixA.4 can be defined, as well as the measurement matrix A.5 as follows:

$$x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad (A.4)$$

$$y = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad (A.5)$$

.

Having the state matrix and measurement matrix define, the next step is to define, the behavior model equation of the system (equation A.6), as well as the measurement equation (equation A.7) using the following:

$$x_k = Ax_{k-1} + Bu \quad (A.6)$$

$$y_k = Cx_k \quad (A.7)$$

Where:

A e B are the state matrices
C relates the measurements with the state variables
u is the input of the system (in this case $u = \ddot{x}_k$)

Finally, since there is an error in the measurement, as well as in the process (friction, inertia, etc...), a co-variance matrix for the respective errors must be added to the formulas, resulting in equation A.6 and equation A.7. Imagine that the error in the process is calculated and discovered to be around $0.8m$ and $0.6m.s^{-1}$, and looking into the GPS datasheet, the error corresponding to position and velocity is respectively $0.5m$ and $0.6m.s^{-1}$, then they can be added into equations A.6 and A.7 , attaining:

$$x_k = Ax_{k-1} + Bu + w_k \tag{A.8}$$
$$y_k = Cx_k + v_k \tag{A.9}$$

Using the dynamic formulas A.1, A.2 and A.3, as well as the Kalman Filter prediction formulas A.8 and A.9, results in:

$$\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_{k-1} \end{bmatrix} + \begin{bmatrix} 0.8^2 \\ 0.6^2 \end{bmatrix} \tag{A.10}$$
$$\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} + \begin{bmatrix} 0.5^2 \\ 0.6^2 \end{bmatrix} \tag{A.11}$$

Having the model defined, it is needed to know the initial values for the state variables (x_0). The values may be known, or if that's an impossibility they can be assumed as the first values measured (this will result in a worst filter application, meaning that the filter will take longer to converge to reliable data). The filter will then compare the received (measured) data with the expected one, computing a gain (importance), which will determine the validity of the incoming data in the system, i.e, the veracity of the measurements with what was expected. For that, the following formulas are used:

$$G_k = P_k.C^T(C.P_k.C^T + R)^{-1} \tag{A.12}$$
$$x_k = x_k + G_k(y_k - C.x_k) \tag{A.13}$$
$$P_k = (I - G_k.C)P_k \tag{A.14}$$

Where:

P_k measures the certainty of the estimated values
G_k is the gain of the filter (importance of the measured data)
I is the identity matrix
R is the co-variance matrix respecting v_k
T indicates that a matrix is transposed

After analyzing these equations, it is possible to obtain some conclusions related to the Kalman Filter process. Looking at gain in equation A.13, for instance, it can be seen that if G_k is equal to zero, the state variables value is the same as the predicted for the filter, meaning that the measurements are not needed in order to know the values of the state variables. Physically, the behavior of the model is perfectly accurate with reality. On the other hand, if G_k is equal to one, then to the predicted value of the state variables, the difference of what was measured (y_k) and what we were expecting to measure (Cx_k) is added. Notice that if the measured values and the expected measured values are the same, then the prediction is accurate, as the difference of these values is equal to zero. Equation A.12 shows that G_k is dependent of P_k. If the measurements are not trustworthy, P_k is zero and the gain is zero, which leads to the above conclusions.

In short, the Kalman Filter is based in an infinite cycle of prediction and updating of variables, being that the last three equations belong to this last step (updating). The Kalman Filter prediction step is presented in equations A.15 and A.16. Notice that the symbol "ˆ " indicates that a given variable is an estimation made by the filter.

$$\hat{x}_k = A\hat{x}_{k-1} + Bu \tag{A.15}$$
$$P_k = A.P_{k-1}.A^T + Q \tag{A.16}$$

Where: Q is the co-variance of w_k.

It can be observed that the equation A.15 is already familiar (equation A.6). The objective of this equation is to estimate the state variables value in a given instant k. These values will later have to be updated. It can also be observed that equation A.16, has the goal of calculating the credibility in the measurements taken, given the model of behavior as well as the process error. It is important not to forget that the obtained value of P_k will affect the later value of G_k.

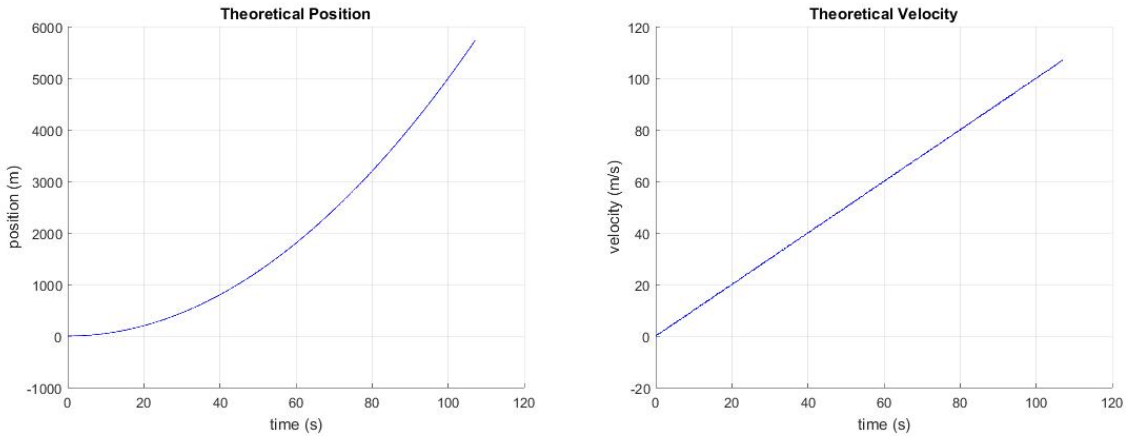
To conclude, the Kalman Filter application can be resumed in three steps:

i **Model Definition**
Necessary to make only once. It has the purpose of understanding the behavior of the system. In this step A, B, C, u, x_k, w_k and v_k are defined. It is not meant to calculate, but rather to characterize the system model.

$$x_k = Ax_{k-1} + Bu + w_k$$
$$y_k = Cx_k + v_k$$

ii **State Prediction**
Included in the infinite cycle, has the goal of estimating the state variable values as well as the "reliability" it can have in this specific instant in time. In this step, $\hat{x}_k$ and P_k are calculated.

$$\hat{x}_k = A\hat{x}_{k-1} + Bu$$
$$P_k = A.P_{k-1}.A^T + Q$$



(a) Theoretical Position of the Vehicle.

(b) Theoretical Velocity of the Vehicle.

Figure A.1: Kalman Filter Example - Theoretical Values.

iii **System Update**
Given the measured values and the expected values obtained from the model, the state variable values are updated as well as the reliability the system has. G_k, x_k and P_k are calculated in this step.

$$\begin{aligned} G_k &= P_k.C^T(C.P_k.C^T + R)^-1 \\ x_k &= x_k + G_k(y_k - Cx_k) \\ P_k &= (I - Gk.C)P_k \end{aligned}$$

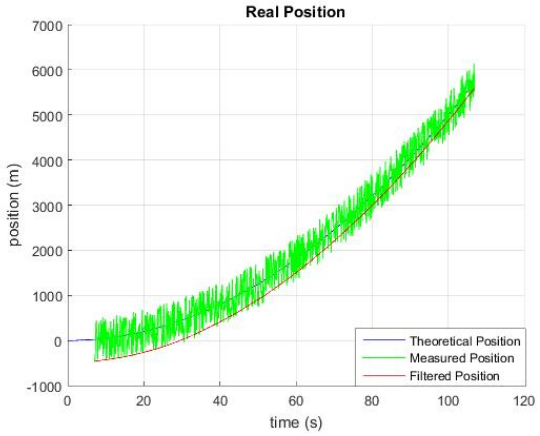
Practical Example Simulation

In order to exemplify how powerful Kalman Filter can be, a simple theoretic example was elaborated, with the example mentioned above (page 63), using however, exaggerated values to create noise. Thus, testing the ability to filter and improve the obtained data. Before presenting this example, it is important to notice some premises, that the example is based on:

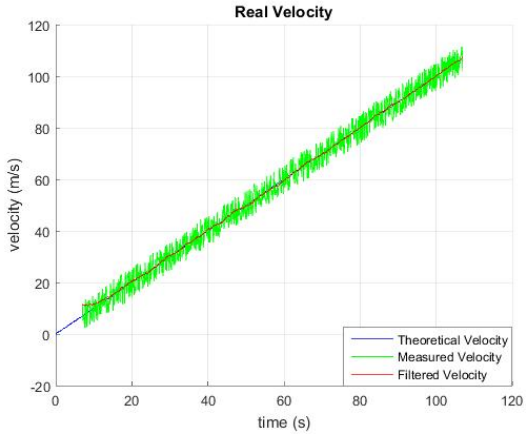
- The vehicle is moving when the reception of the values began. Thus, it is impossible to estimate any initial values for the position or velocity.
- Since the initial values aren't known, these will be assumed as being equal to the first measurements taken. This will create a slower response in the Kalman Filter process.

Figure A.1 shows the values of the position and velocity of the moving vehicle according to the generated model. In these values, the process noise is already represented. In Figure A.2 it can be seen, in green, the obtained raw values from the GPS as well as the treated (filtered) values, in red. For easier comparison, Figure A.3 presents the theoretical values (blue) and the filtered data (red).

As mentioned above, the error generated in the obtained values is amplified in a unrealistic manner, since there is no GPS which has an error of $484.19m$ in position or $4.52ms^{-1}$ in

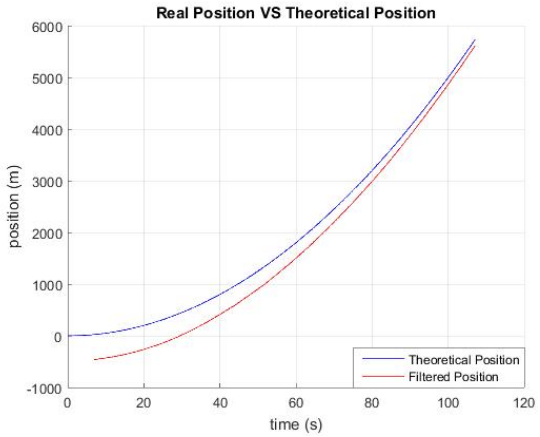


(a) Measured Position VS Filtered Position.

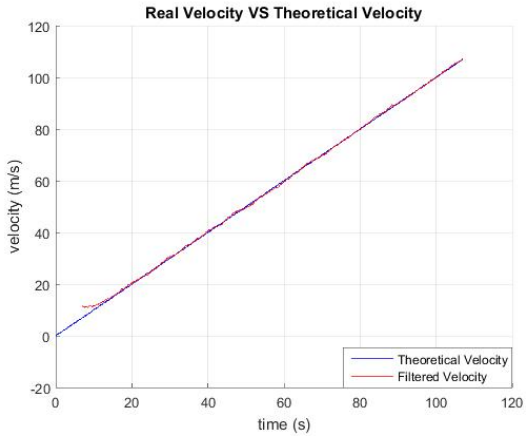


(b) Measured Velocity VS Filtered Velocity.

Figure A.2: Kalman Filter Example - Obtained and Filtered Values.



(a) Theoretical Position VS Filtered Position.



(b) Theoretical Velocity VS Filtered Velocity.

Figure A.3: Kalman Filter Example - Reality VS Theory.

velocity. However, this exaggeration allows us to verify the quality of the implemented filter, as well as some inner characteristics.

There are some conclusions that are immediately visible. For instance, Figure A.2a shows that the GPS measurements (green), start $7s$ after the beginning of the motion. The red line, which represents the filtered data has an error of $484.19m$ in position , early in the experiment, but is slowly diminishing through time, tending to approach the real value. At $107s$, the error is much smaller ($117.53m$ - which seems to be a huge error, but it is 4.12 times more accurate than the unfiltered data. Remember that there isn't any new source of data being used).

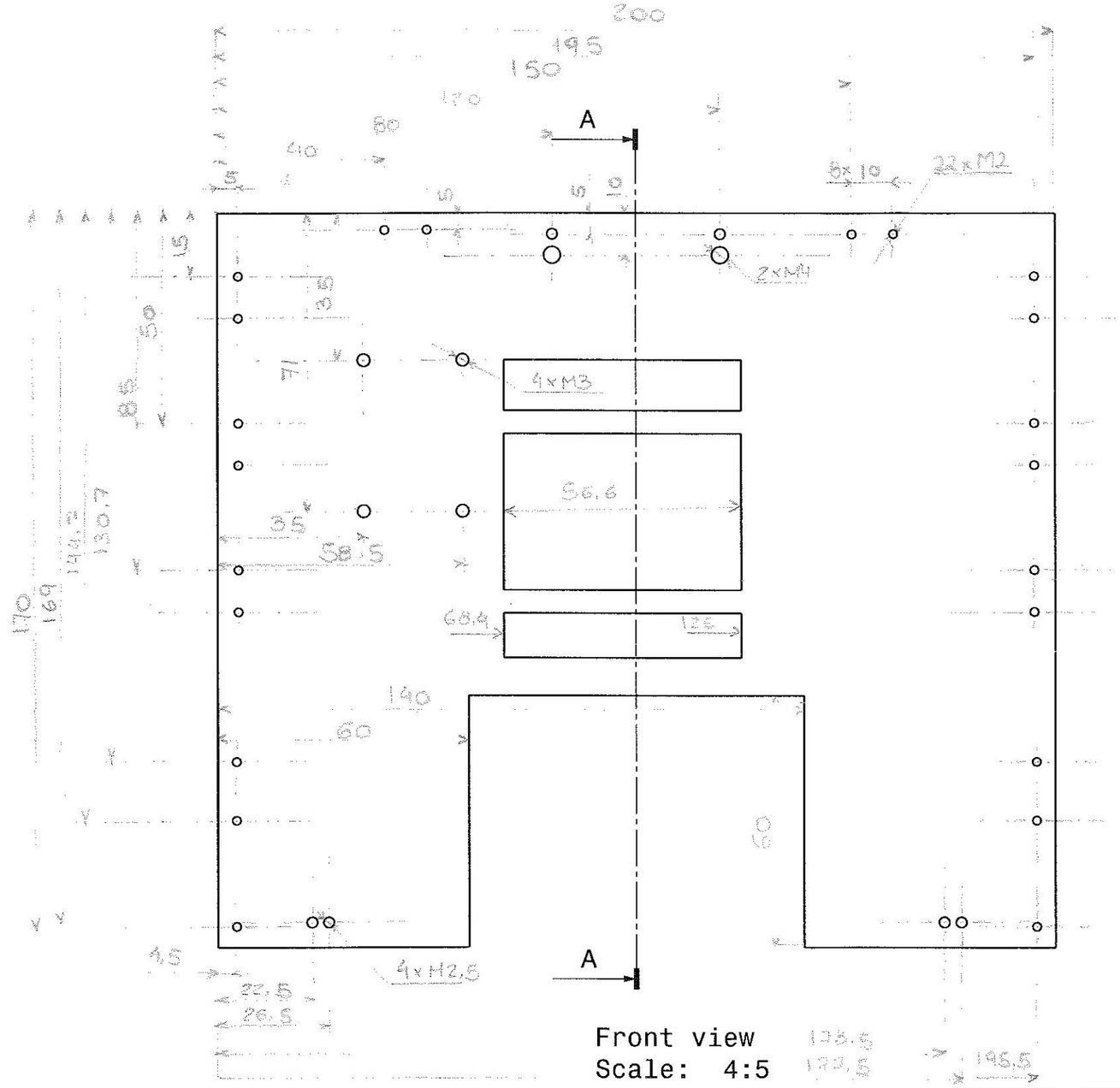
In Figure A.2b the behavior is similar, i.e, there is a trend for the filtered data to approach the theoretical values, even though the existing error is quite big.

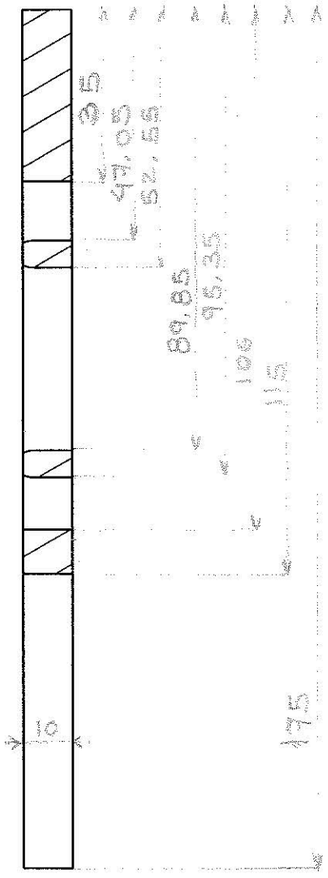
Notice that in the Linear Kalman Filter approach, this is probably one of the worst case scenarios, since only the model is known (without it, it would be impossible to implement the filter), not knowing, however, when the data started to be treated and without knowledge of the initial position or velocity of the vehicle. It is important to say that the theoretical data plays no part in the filtering of the data. There are a few ways to improve the behavior of the filter; for example, if the measurements start at the same time as the experiment, then an initial guess of the position and velocity could be reached (in this case they would both be zero).

Finally, it is essential to remember that the Kalman Filter is as good as its implementation. For the Kalman Filter to work properly, understanding and approximating the model behavior as deeply as possible is the key. This applies to the process and measurement error matrices as well. For that, it is advised to know the sensors (read the datasheet) and, if possible, analyze and quantify all the process noise variables. In case this last step is not possible, it is advised to run as many simulations as possible in order to understand which values will give the best outputs.

Appendix B

Drawing of the Designed Piece





DESIGNED BY: Joao Peixoto	PHUA
DATE: 30-03-2015	
CHECKED BY: XXX	
DATE: XXX	
SIZE A3	DASSAULT SYSTEMES
SCALE 4:5 / WEIGHT (kg) XXX / DRAWING NUMBER XXX / SHEET 1/1	

This drawing is our property; it can't be reproduced or communicated without our written agreement.

I	_
H	_
G	_
F	_
E	_
D	_
C	_
B	_
A	_