INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Licenciatura em Engenharia de Electrónica e Telecomunicações e de Computadores

September 27, 2019

Autopilot for a remote-controlled Sailboat.

Bruno Pinto

Final report made in the scope of Project and Seminar, of the degree course in Electronics, Telecommunications and Computers Engineering

Summer Semester 2018/2019

Mentors: Professors Pedro Fazenda and Jorge Pais

Abstract

This work consists in the development of an autonomous navigation system for a radio-controlled sailboat. The objective is to be able to navigate following a set of GPS coordinates, planned from predetermined geographic points by a base station.

As the propulsion system of the sailboat relies on wind forces instead of motors, sailing techniques will be introduced and its mechanical structure will be presented.

The system-level description (software, hardware, sensors, actuators) as well as the control architecture will be discussed.

Simulated results will be presented and discussed to validate the proposed solution with the use of Robot Operating System (ROS) as the software platform.

The sensors used are: The Global Positioning System (GPS), to determine the coordinates of the vehicle and its trajectory and indirectly its speed; the anemometer to determine the speed of the wind in relation to the sailboat and its direction; an inertial measurement unit (IMU) for navigation and orientation relative to a inertial frame.

A wide range of technologies will be used. In particular, the use of the Robot Operating System (ROS), GPS, IMU, Linux, Raspberry Pi, etc.

Keywords: sailboat; auto-pilot; remote control; autonomous navigation; Robot Operating System.

Resumo

Este trabalho consiste no desenvolvimento de um sistema de navegação autónomo para um veleiro rádio telecomandado. O objetivo visa conseguir pilotar o veleiro seguindo um determinado conjunto de coordenadas, planeada, a partir de pontos geográficos pré-determinados por uma estação base.

Como o sistema de propulsão do barco à vela assenta nas forças do vento em vez de motores, as técnicas de velejo serão introduzidas e apresentada a sua estrutura mecânica.

A descrição a nível de sistema (software, hardware, sensores, atuadores, comunicações), bem como a arquitetura de controlo serão discutidas.

Resultados simulados serão apresentados e discutidos para validar a solução proposta, com o uso de da plataforma Robot Operating System.

Os sensores utilizados são o Global Positioning System (GPS), para determinar as coordenadas do veículo e a sua trajetória e indiretamente, a sua velocidade; o anemómetro para determinar a velocidade do vento em relação ao veleiro e a sua direção; uma unidade de medida por inércia para navegação e orientação relativa a um referencial inercial.

Utilizar-se-á um conjunto variado de tecnologias. Nomeadamente, a utilização do Robot Operating System (ROS), GPS, IMU, Linux, Raspberry Pi, etc.

Palavras-chave: veleiro; piloto automático; controlo remoto; navegação autónoma; Robot Operating System.

Contents

List of Figures

List of Tables

1 Introduction

In this section, the goals and motivation are presented and past work in the area is described to provide the reader a context.

Figure 1: Sailboat used in the project

1.1 Goals and Motivation

Sailing boats are sails-propelled vessels by the wind-driven lift between the sails, similar to a wing of an airplane. As autonomous vessels it's an emerging technology, capable of operating, without human intervention, for long periods of time with low maintenance and low consumption, although relatively slow, it manages to cover a large sea area, therefore good candidates for long term research missions. Control of a vessel propelled by a propeller is relatively easier than controlling a sailboat. To move a

sailboat along a path, one has to consider some conditions, such as: the direction and speed of the wind, the orientation of the boat, its physics, the operation of the sails and the rudder. A control architecture, in order to fulfill the defined trajectory, must use the above information, through the sensors on board, to manipulate the orientation of the sails and the rudder.

In the next sections, the operation and structure of a sailboat will be presented, the high-level description of the system (software, hardware), the proposal of the control architecture and the autopilot, its implementation, tested and validated in a simulator.

1.2 State of Art

Many institutions and universities have developed autonomous navigation systems for sailboats (of various sizes), of varied complexity used for many applications.

The ATLANTIS project, a wind-powered catamaran from Stanford University [3], United States of America, dates back to 1997, was one of the first robot ships with autonomous navigation systems forming the basis for further projects. The thesis contains several equations, which allow to estimate very accurately the orientation and position of the boat, and presented several mathematical models of the boat, although simple, are sufficient for the control problems.

The Aland Sailing Robots [2] (ASR) and Fast [1] utilizes classical control theory for the sailboat controllers. Both uses a PI controller on the rudder and the sail is based on the angle-of-attack with the error defined as the desired orientation minus current orientation.

The ASR's main goal of is to take part in the competition Microtransat challenge (cross the atlantic fully autonomely) requires a tough design to endure the harsh oceanic conditions. Developed, in Sweden, main goal is maritime surveillance and research. It has on board sensors for temperature, conductivity, pH for water, atmospheric pressure and humidity for monitoring atmospheric and water conditions. The autopilot software is runs on a Raspberry Pi, programed in C++, suported my MATLAB in analysis and synthesis of the control architecture.

Figure 2: Robot sailboat, University of Aland.

FASt, has the capacity to carry equipment up to a few kilograms, to carry out oceanographic research missions with voltage monitors, ambient light sensor, interior temperature and a set of water sensors distributed in various places inside the hull. The computing system is implemented in a small board computer based on a FPGA and has capacity for short and long range communications. The autopilot software runs as a set of processes communicating by UDP sockets. Each level receives and sends commands via UDP protocol, including reading sensors, controlling the actuators and run the control algorithms.

The Voilier Autonome Instrumente pour Mesures Oceanographiques de Surface (VAIMOS) boat [5] uses a simple line-following technique, using an LQR controller with focus on using less energy from the actuators, and for the sail control, it uses data observed in (or inspired by) human behavior (machine learning), making the system independent to any mathematical models of the sailboat. The high-level control of VAIMOS also has autonomous behaviors for obstacle avoidance.

Figure 3: FASt, autonomous sailboat developed by FEUP

A major project, comprising of many Brazilian universities, developed the N-Boat [4] robot sailboat, using an Arduino module as the low level controller of N-Boat, linking actuators straight to the sensors through control laws and a Raspberry Pi responsible for high-level navigation decisions, planning and external communication. A PI controls the rudder and fuzzy logic controls the sails.

The university of Southampton developed a navigation system [10] to compete in the "World Robotics Sailing Competition". The main computer of the Black Python is a Raspberry Pi.The software is written in Python and utilises ROS (Robot Operating System). This project utilizes a dynamic weighting approach to choose the best sailing maneuver whilst having minimal sensor knowledge of weather and boat state, something like human sailor would navigate based on observations and experience, making small adjustment as the manoeuvre proceeds. (arXiv:1903.06677 [cs.RO]) This is an example of adaptative control, that attempts to mimic somekind of a human control architecture.
Gion-Andri B [7] with the AVALON project, developed path planning algorithms and weather forecast systems to allow the sailboat to navigate across the Atlantic Ocean (Microtransat). Planned optimal navigation

course and efficiently control the boat on this course. The path planner uses an A* algorithm to generate the fastest path to a given destination. It is able to avoid both static and dynamic obstacles. For the sail, the controlled value is the angle of attack, a PID controller was designed to control the rudder. A sailboat can not sail at all angles to the wind. To do that, the controller is designed as a state machine that switches states depending on the angle to the wind. All software is developed in C++ and Python and runs on a Linux OS.

There is also been works about the implementation of non-linear controllers [8] using the backstepping method combined with more complex mathematical model of the sailboat including fluid dynamics.

Although there exist several projects, the various solutions follow the same set of concepts and ideas. The solution is to implement a control architecture for the rudder and another for the sail, which allows the sailboat to sail over a "straight" line to the surface of the water. The techniques used in the implementation can vary, from control theory, fuzzy controllers, automatic learning, among others. For the present work, classical control algorithms and a state machine will be used implemented on the software/hardware platform Robot Operating System (ROS)

2 Sailing Concepts

This section introduces to some of the concepts in the sailing world, such as: the structure of a sailboat, sailing techniques and points of sailing.

2.1 Sailboat structure

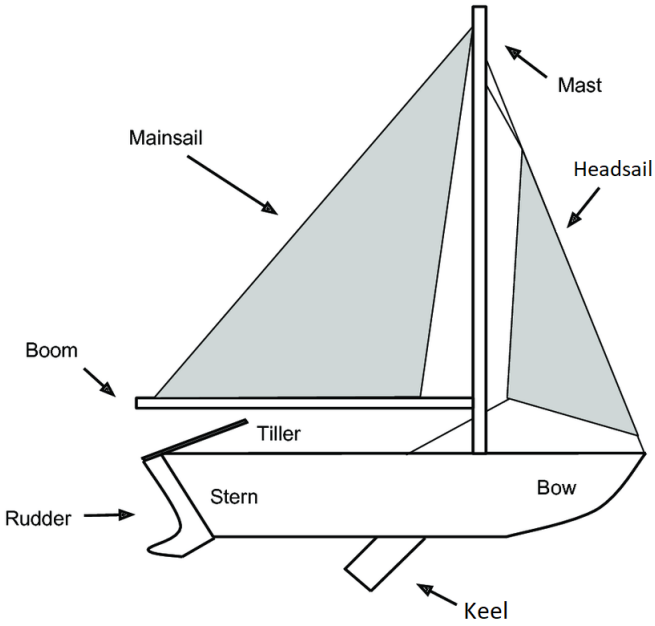


Figure 4: Sailboat structure.

Hull – Ship's main structure, that keeps the boat to stay afloat.

Rudder – It's a submersed structure responsible for steering the sailboat, which allows the autopilot to control the heading. If the angle of the rudder is big, it may cause drag and slow down the boat, something that's worth considering, if the autopilot is meant for racing.

Mainsail and headsail – Use the wind power as a propulsion force via lift and drag ("catching the wind") to the sailboat. The angle of the sail and how tight it is, depends of the angle of attack (angle respective to apparent wind), limited by the point of sail. Both sails are connected to the mast. The only difference between the mainsail and headsail, is that the mainsail is actuated, might be fully tighten or free to move around, whereas the headsail is not actuated, it's completely free.

Keel – It's a submersed structure that serves, mainly, 2 purposes: helps the sailboat, when is abeam, to move forward, as show in the fig. 5;

increases stability by preventing from being blown sideways by the wind, as shown in fig. 6.

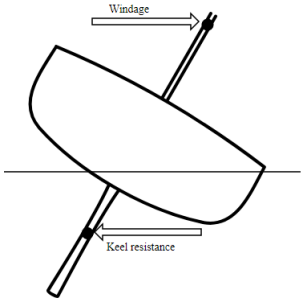


Figure 5: Keel Resistance against the wind.

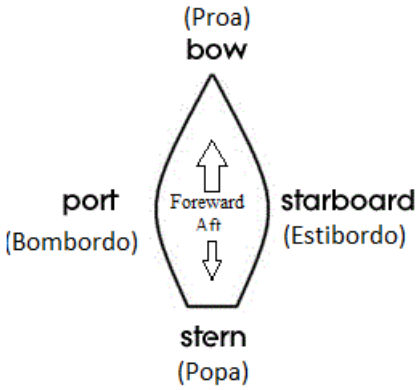


Figure 6: Nautical terms relative to a ship's position.

2.2 Points of Sailing

In order to design and implement an autopilot for a sailboat, first its necessary to know how a sailboat behaves (dynamics), the sailing techniques and points of sailing (boat's orientation relative to the true wind).

There are 2 basic sailing maneuvers: jibe and tack.

Jibe/Bearing Away (virada em roda) is a sailing maneuver, when the sailboat is Broad Reaching on starboard, maneuvering to Broad Reaching

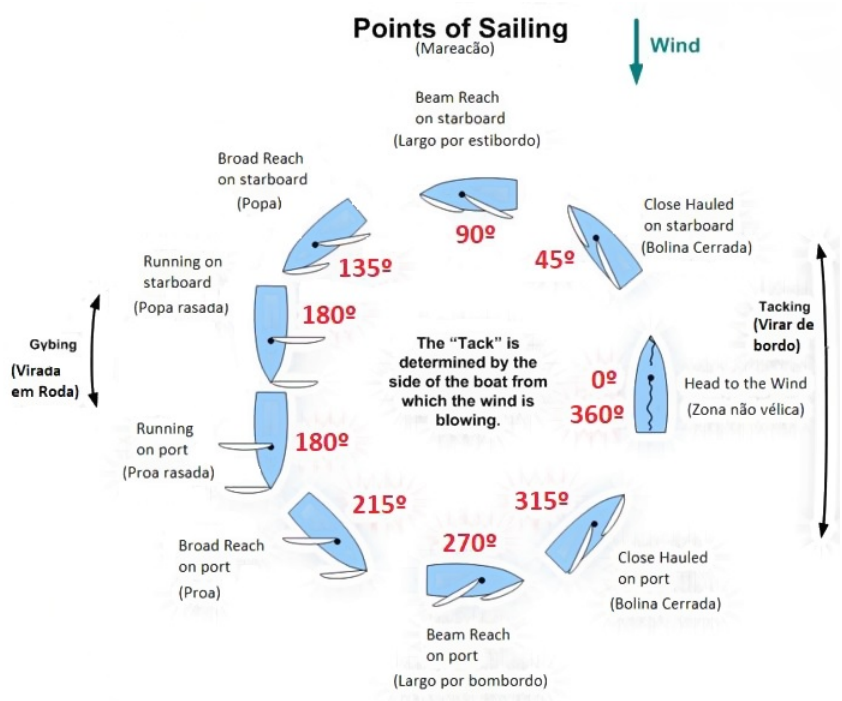


Figure 7: Points of Sailing.

on port (135º to 225º degrees, turning its stern through the wind, such that the wind direction changes from one side of the boat to the other (shown in figure 4). The sail sheets are eased, creating a larger sail area, larger resistance to the wind, thus generating more propulsion, making the boat sail faster. The mainsail and the headsail must be in opposite sides and fully eased to ensure maximum sail area, thus the sailboat is faster in broad reaching than running (straight downwind), but jibing is a dangerous technique, when the sailboat is switiching sides, can cause a very high stress on both sails, forcing the sails to move aggressively and swing across the deck at high and harm anyone on board or simply snap the mast.

Tack (virar de bordo) is a sailing maneuver, when the sailboat is close hauling on port maneuvering to close hauling on starboard. This technique is used when the desired direction is nearly directed into the wind (the boat is 45º degrees relative to the wind). The mainsail is tightened ("velas caçadas") and the rudder movement is kept to a minimum to avoid losing speed and far as stop the vessel completely. It's important to "catch" the

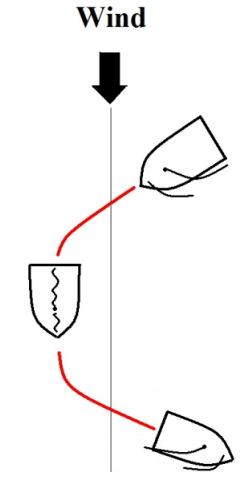


Figure 8: Jibe Manuever.

wind back in the sails as quickly as possible and regain speed.

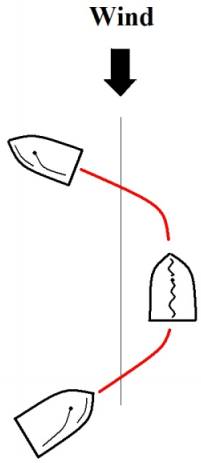


Figure 9: Tack Manuever.

From those 2 sailing maneuvers, there several techniques employed depending on certain circumstances: Beating to Windward ("bolinar") is a sailing technique to move towards the wind, in a "zig-zag" fashion, continuously tacking until reaching the target.

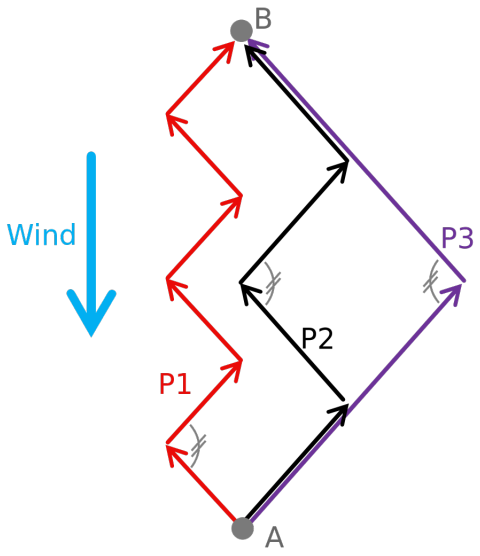


Figure 10: Sailing towards the wind in a “zig-zag” movement.

Stopping (Heaving to) a sailboat can be done by to turn it directly into the wind to make it start losing speed, releasing the sail sheets slowly and turning the rudder to force the vessel into the direction of the wind.

3 System-Level description

In this section, is presented the sensors and actuators used in the project, the hardware and software architectures.

3.1 On-Board Computer (Raspberry Pi)

The computer aboard the sailboat, collects the data from the sensors, processes them, and triggers the commands required by the actuators. The computer to use will be the Raspberry Pi, which contains some functionalities, desired in this work, such as: fast and efficient calculation processing; Internet connection and the ability to communicate with the outside via Wireless technology and the possibility of using an Operating

System, such as Linux, that will allow the installation of the software to user in this work, the Robot Operating System (ROS).

3.2 Robot Operating System (ROS)

"The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms" - ROS org
ROS, despite its name, is not an operating system, but rather an open source ecosystem that allows users, hardware abstraction, low-level hardware control (Device Drivers), implementation of communication protocols, an operating system, hence its name. Provides libraries and tools for writing and developing robotic applications. The ROS ecosystem builds around around nodes (executables), those nodes communicate by sending (Publish) and receiving (Subscribe) messages with a specific topic name.

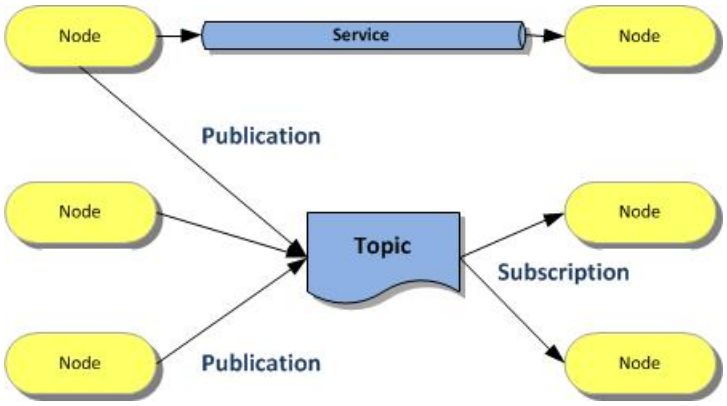


Figure 11: Robot Operating System - nodes and topics.

Some of the advantages of ROS are:

- Provides complete solutions to common problems (navigation, detection of obstacles, etc.), so there's no need to reinvent the wheel every time a project commences;

- Existence of a simulator (Gazebo);

- Efficient and simple management of multitasking programs, allowing parallel solving of various problems;

- Its easy to replace nodes with similar interfaces, allowing flexibility and modularity in your system;

- The various components (hardware and software modules) communicate with one another through a distributed messaging system based on the TCP/IP protocol, thus allowing abstraction and decreasing dependency. This way of communicating is similar to what FAST boat system does, but with UDP data transfer protocol.Its possible to have a arduino or MATLAB for example publishing messages and having a ROS node subscribing to them;

- Can be installed in a Raspberry Pi;

- A reasonably large list of sensors and actuators compatible with the platform.

Some of the disadvantages are:

- One factor posing a challenge to the industry, is the security issues in the design implementation of robot operating system;

- No real-time computation. ROS is mainly running on Ubuntu, which is not a hard real-time operating system;

- It requires a whole computer to work. Compared to microcontrollers, it uses lots of more power and space, and has no guarantee of real time control;

- The learning curve is steep, one needs to spend a considerable amount of time before working with this platform.

3.3 Hardware

In order to determine which sensors to use, there is need to define what states to know about the system in question and what goals to meet. In

the case of a sailboat, it is necessary to: determine its position and its speed (GPS); orientation and inclination (IMU)), wind speed and direction (anemometer).

Other sensors can be used, such as: a camera, infrared sensors, ultrasound or LIDARs for detection of obstacles, temperature, pressure and humidity sensors for the implementation of a meteorological system, among others, depending on the application of the system. The following is the description, the importance and the reasons of their use:

GPS - It is a satellite navigation mechanism, directly providing the position (longitude and latitude) and indirectly the speed of the receiver, although it causes errors. Used in this work to trace a trajectory of the sailboat.

Anemometer – It is an instrument that measures the speed and direction of the true wind in relation to the sailboat, these variables are extremely important for the handling of the boat, since the latter is propelled by wind. The sails and the rudder are manipulated to the right angles, so that the boat moves efficiently and, in a way, to make the most of the wind.

Compass – It is an instrument of navigation and guidance, providing the angular direction of the system at any instant of time, relative to the cardinal points. The compass is used in this work to turn the sailboat in the desired direction.

Gyroscope – It is a device that measures or maintains the angular velocity. Its operation is based on the principle of inertia, it uses the gravity of the earth to determine the orientation of the object on it is placed. It consists of a freely rotating disk mounted on a rotating shaft; it opposes any attempt to change its original direction. It is used in this work to know, directly, the angular velocity of the sailboat in the 2 axes of motion (axis of the x and y axis) and indirectly the Euler angles (pitch, yaw and roll), that shall be described in later sections, with the integral of the angular velocity, which accumulates errors in the long of the various calculations.

Accelerometer – It is a device that measures non-gravitational acceleration. When placed on an object, it senses vibrations associated with linear velocity changes (linear acceleration). It uses microscopic crystals, which when deformed, cause vibrations (analogous to a mass with a spring), an electric voltage is generated, and read by the sensor. It is used in this

work, to determine the linear velocity (with the calculation of the integral of the acceleration that, once again, accumulates errors and the Euler angles (pitch, yaw, roll).Those last 3 sensors, form an Inertial Measurement Unit (IMU). All sensors, together, provide the information needed to send the commands to the 2 available servo motors aboard the sailboat, one motor to control the sail angle, the other to control the rudder angle.

The servo motors are electromechanical devices with an associated control mesh, generally consisting of a DC motor, a control loop (PID + oscillator) and a sensor (potentiometer) that measures the angular position by the electric resistance. The control signal, which drives the DC motor, is generally affected via a PWM signal. Figure 12, shows the connection between the computer, sensors and actuators.

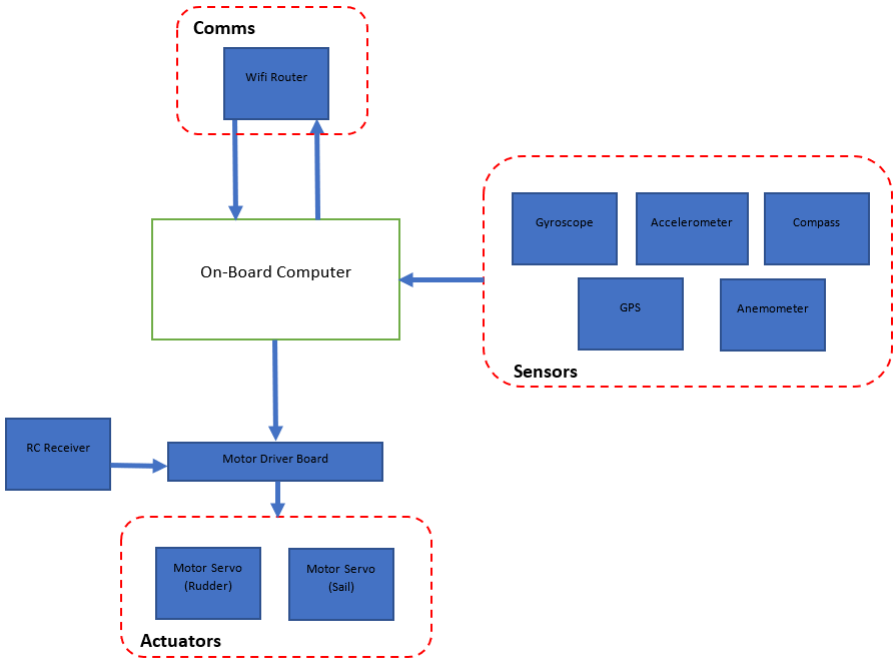


Figure 12: Hardware System Architecture.

3.4 Software

In this section, the proposed software architecture to implement the auto pilot is presented.
A simulator was developed to simulate all sensors and actuators described with a state space model (position, heading, linear velocity and angular velocity).

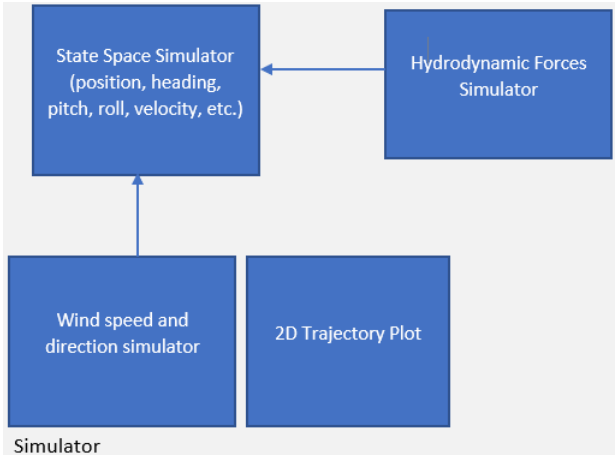


Figure 13: State Space simulator.

The Auto Pilot code is responsible for telling the sailboat what to do and how to do it. It takes the estimated state space of the system (position, velocity, acceleration, orientation, etc.) and information about the wind (direction, speed) by the State Estimator Module.

The Tasks Planner module takes waypoints (GPS coordinates) through the applications layer to decide what task to execute. The Tasks Runner module tells how to execute that task and send the appropriate commands to the Actuator Processing module, that converts the commands into correspondent PWM values, before sending it to the servo motors on-board the sailboat.

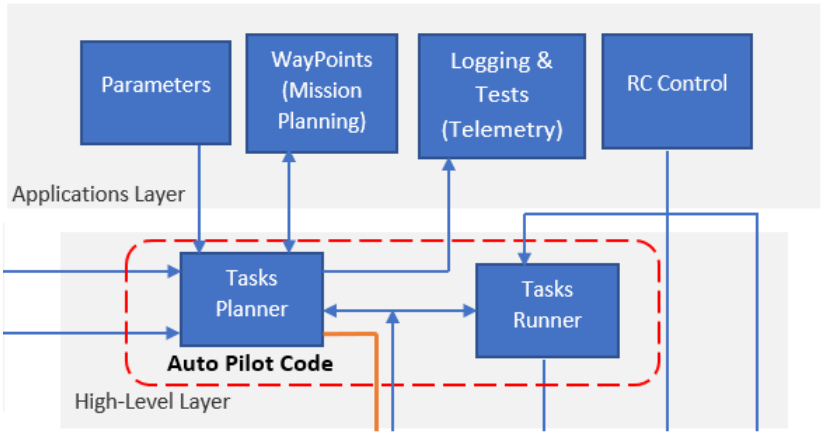


Figure 14: Tasks Planner and Tasks Runner modules in the High-Level layer.

The sensor processing module, takes raw data from the sensors, converts them to appropriate state space variables, further fused in the state estimator module.

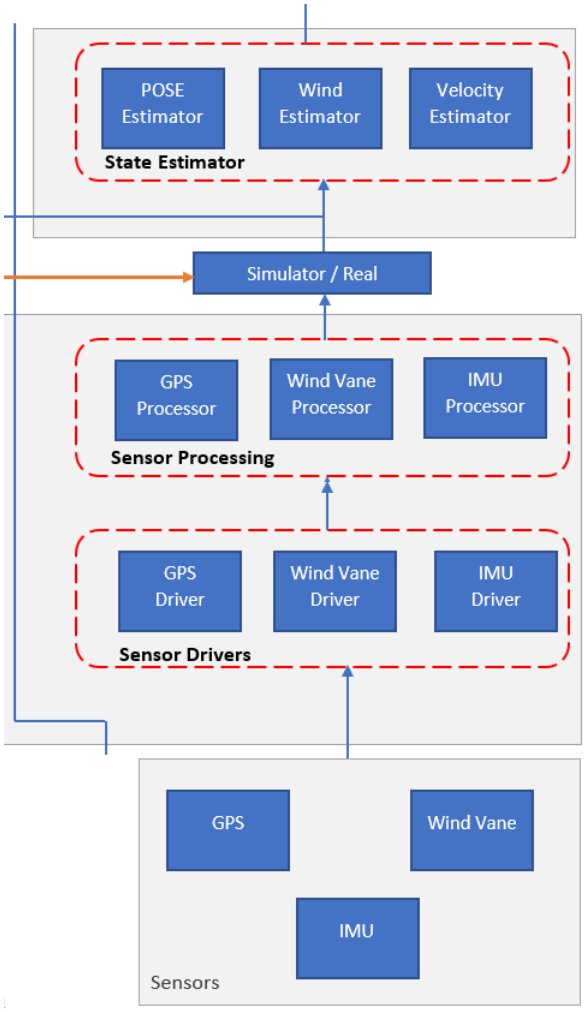


Figure 15: Sensor's and State Estimator's module.

The autopilot can work both in simulator or reality mode. In simulator mode, the state space variables and wind information are provided by the state space simulator; In reality mode, is provided by the sensor processing module. The simulator also adds hydrodynamic forces and wind information to the state space simulator.

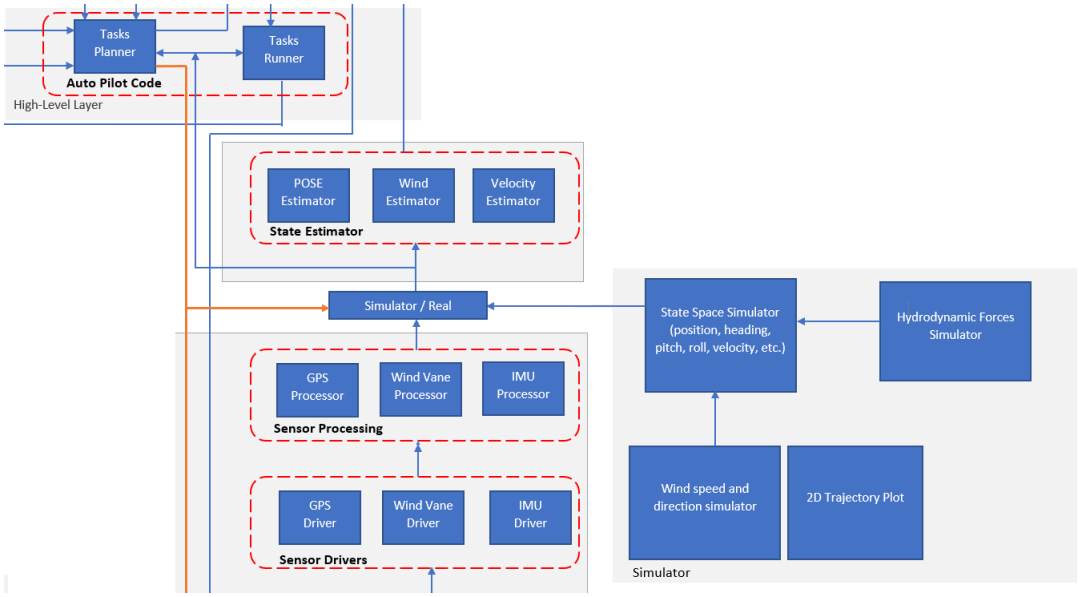


Figure 16: Switching between simulation and reality.

The 2D Plot module, allows to visualize the state of the sailboat in 2D, providing information about x and y coordinates, wind direction, boat's heading, speed and time. If the user wishes to manual control the boat, it can by switching automatic mode to manual mode within the applications layer. In manual mode, the motors only respond to the commands sent by the radio receiver on-board.

4 Mathematical Model

In this chapter, the dynamic model of the sailboat with three degrees of freedom (3 DOF) is described using Newton's second law of motion, also considering: the apparent wind and water current speed and direction and their respective aerodynamics.

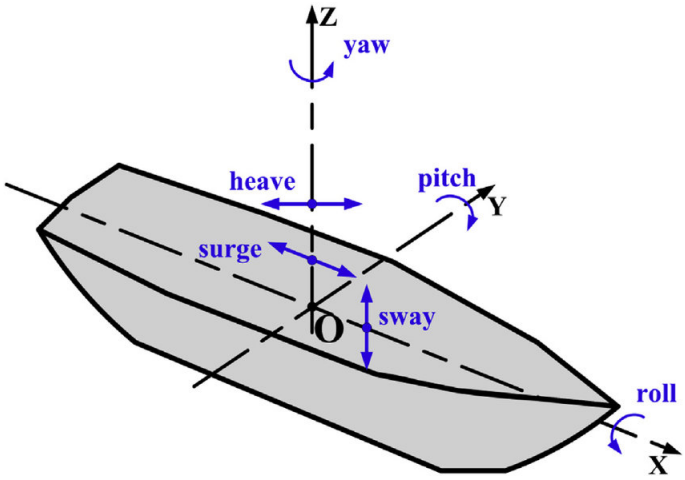


Figure 17: Sailboat degrees of freedom

4.1 Motivation

The main aim of this project is to design an autopilot to steer the sailboat from point A to point B, thus its necessay to understand, how the sailboat behaves within the environment and reacts to the actuator commands, based on the laws of physics and points of sailing.

Constructing a model makes possible to build a simulator for the sailboat to help to design a controller without doing several tests in the real system, thus improving logistics. An initial controller is implemented based on the simulator and tweaked as needed in the real world.

4.2 Background info: Ship's motions

A sailboat has 6 degrees of freedom: 3 rotational (pitch, roll, yaw) and 3 translational (sway,heave,surge). In this project, we will consider only 3 degrees of freedom (yaw,sway and surge), because the boat will navigate in calm waters, so pitch and heave motions are equal to zero and its motion is confined to the horizontal plane at sea level, thus roll is zero.

The vertical/Z axis (yaw axis) is the vertical line trough the origin. The yaw is the turning **rotational motion**. The sway is the side-to-side turning **linear motion** generated by the water and wind currents exerting

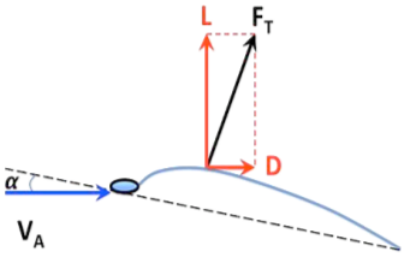


Figure 18: Lift and Drag

forces against the hull. Both motions determine the boats orientation relative to the true north.

The longitudinal/X axis (roll axis) is the horizontal line trough the origin. The roll is the side-to-side **rotational motion**. The surge is the forward/backward **linear motion**.

The lateral/Y axis (pitch axis) is the horizontal line trough the origin, perpendicular to the longitudinal axis. The pitch is an up and down **rotational motion** of the boat. The Heave is the **linear motion** counterpart of the pitch (up/down linear motion), both shall not be considered in the derivation of the system dynamics, because the sailboat will navigate in calm waters, up and down movements shall practically be non-existent.

4.3 Force on Sails

It's the interactions between the wind and the sails that generates propulsion. The forces on sails (lift, drag) depend on the true and apparent wind speed and direction. The apparent wind is the wind measured on-board (moving surface) that creates the aerodynamic force with force components: lift, drag. Depending on the point of sail, one of the two forces is predominant.

Drag (Air resistance) – It's a force opposite to the relative motion of the object and in the same direction as the apparent wind.

Lift – When the sails are trimmed correctly (cable tight or eased) the true wind is deflected downwards and by Newton's third law, a upwards force is

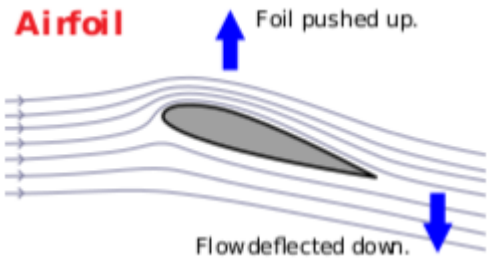


Figure 19: foil deflection and contracction

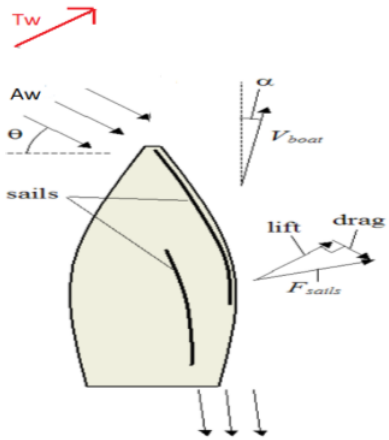


Figure 20: Force on Sails

exerted (lift) making normal (90°) angle with the apparent wind, generating propulsion.

Let's analyze a single point of sail (Broad Reach). The apparent wind blows at an angle ψ (relative to the boat's horizontal axis) flowing through the sails, generates 2 force components: lift (perpendicular to the wind), drag (parallel to the wind), which add to a resultant force, called Force on Sails.

The air flow trough sails are very similar to air flow on an aircraft wing (airfoil). The mainsail is trimmed to optimize the air flow to generate as much lift as possible and less drag. In this particular point of sail, the magnitude of the lift vector is much bigger than the magnitude of the drag vector, thus propelling the boat, with velocity vector (Vboat) making an

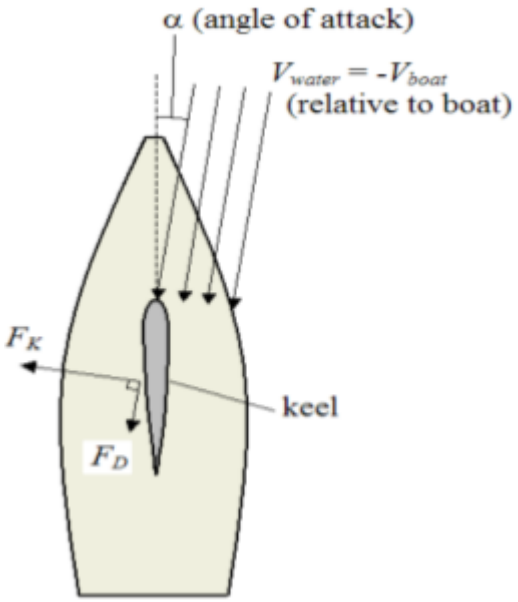


Figure 21: Angle of attack

angle α (boat's angle of attack). The boat's angle of attack is defined as the angle between the boat's course vector (Vboat) and the apparent wind vector (AW).

The force exerted on the sails by the apparent wind causes sideways motion of the boat but the keel exerts a force, minimizing that sideways motion, so the boat moves forward with the angle of attack as low as possible. F_k, is the force exerted by the water on the keel and is perpendicular to the water's flow direction (loosely equivalent to lift) and F_D, same as before, is parallel to the water's flow direction.

So F_sails = - (F_K + F_D). Bigger F_K and lesser F_D, resists more the force on sails, minimizing the boat's sideways motion.

4.4 Dynamics

A mathematical model must be constructed, one simple enough (to avoid unnecessary complex information), but rich enough to capture important

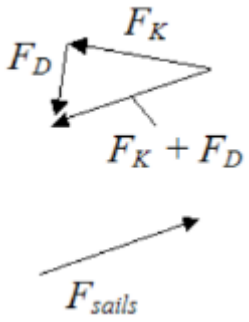


Figure 22: Forces exerted by the keel and due do drag.

informatin of the dynamics of a sailboat for control purposes. In this particular example, the boat's dynamic model, describes how the rudder, sail (treated as inputs), the wind (treated as a disturbance) affects the boat states (position, heading, etc.).

Complex mathematical models (6-DOF) can be found in the literature [9], but such models also require more complex control laws, possibly non-linear [8]. Xiao and Jouffroy [6] reduced the model to 4-DOF. The model described below, is based on the works of John Melin [2] and Le Bars, F., Jaulin, L. Ménage, O (VAIMOS) [5], simplified even further to 3 DOF, neglecting the roll motion, resulting in simple enough to capture essential dynamics and possible to design a linear controller.

This dynamic model of the sailboat was derived with the following assumptions:

• The sail and the rudder are modeled as rigid structures.

• The sailboat shall navigate in calm waters: effects from waves and current is neglected.

• The sailboat's motion is confined to the horizontal plane.

• The two sails are combined into one sail only.

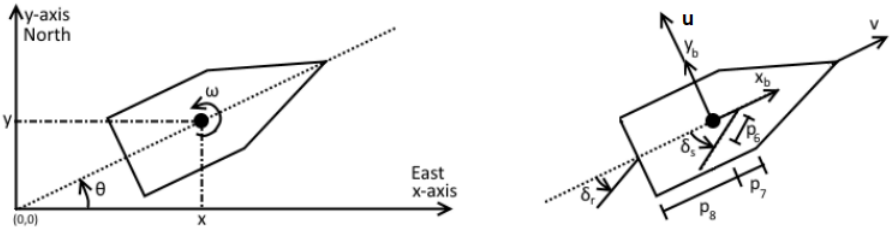


Figure 23: Body and NED coordinates frame

The model is given by non-linear differential equations in state space by:

$$\dot{x} \ = \text{f} \ (\ \mathbf{X},\ \mathbf{U},\ \mathbf{W}_{a,t}\ ) \quad \text{Where,}$$

• $\mathbf{X}$ is the 6x1 state vector, $\mathbf{X}$ = [x y ψ u v w] ;
• $\mathbf{U}$ is 2x1 control signals vector defined in the b-frame, $\mathbf{U}$ = [δ_r δ_s],
Where δ_r is the angle of the rudder and δ_s is the angle of the sail
• $\mathbf{W}_{a,t}$ is the true wind 2x1 vector defined in section 2.3;

There are 2 frames of reference to describe the state vector:

• North East Down coordinates as an inertial reference frame (n - frame) - determines the position
of the sailboat [x, y] and its orientation relative to the geographic north [ψ].

• Body fixed frame (b - frame) - determines the angular [w] and linear velocity [u, v] of the boat relative to the inertial reference frame, Its origin is assumed to
coincide with the sailboat's center of gravity COG and center of mass COM.

$\dot{x}$, $\dot{y}$, $\dot{\psi}$ equations results from the transformation of the body frame (b-frame) to the inertial frame (NED) around the yaw axis (z) through an angle ψ.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} u \\ v \\ \omega \end{bmatrix}$$

, resulting in

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} ucos(\psi) - vsin(\psi) \\ usin(\psi) + vcos(\psi) \\ \omega \end{bmatrix}$$

This equations assumes 0 degrees heading to be the standard definition (unit circle), however, in this project, heading is defined as the angle relative to the geographic north, so a -90 degrees shift must be considered in the model, yielding

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} usin(\psi) - vcos(\psi) \\ ucos(\psi) + vsin(\psi) \\ \omega \end{bmatrix}$$

4.5 True and apparent Wind

The true wind (direction and speed) can be measured, if the wind vane is placed in a static surface. If the wind vane is placed on a moving surface, such as a sailboat, then the readings obtained are the speed and direction of the apparent wind, so the **apparent wind** can be defined as the true wind experienced by an observer in motion. Apparent wind (speed and direction) can be measured by an anemometer, mounted on a moving sailboat. In sailing, knowing the true wind is crucial to determine the angle of the sails of a sailboat which can safely and effectively travel, achieving the fastest speed possible.

Apparent wind (AW) can also be defined as a vector sum of the true wind (TW) minus the Velocity of the boat (VB):

$$TW = \begin{bmatrix} TWS.cos(TWA) \\ TWS.sin(TWA) \end{bmatrix}$$

, is the true wind vector

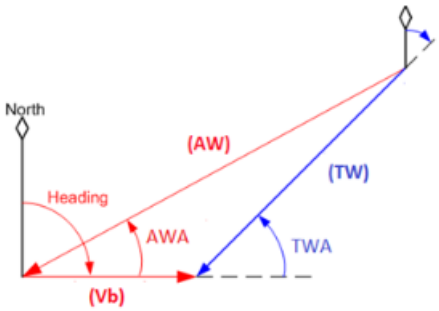


Figure 24: Relation between apparent and true wind e boat's velocity

$$AW = \begin{bmatrix} TWS.cos(TWA - heading) - Vx \\ TWS.sin(TWA - heading) - Vy \end{bmatrix}$$

, is the apparent wind vector

Where V_x is the x component of the velocity of the boat and V_y the y component.

The apparent wind speed can be calculated as follows:
AWS = $\sqrt{AWx^2 + AWy^2}$

The apparent wind direction can be calculated as follows:

$$\text{AWA} = \text{atan2(AWy,AWx)}$$

4.6 Forces and aerodynamics

A sailboat moves due to forces acting on the sail, function of both wind and the boat's speed and orientation. The sum of all forces acting on the sails is called total aerodynamic force,

$$\vec{F_T}$$

. We assumed that the sail and the rudder are rigid foils (deformation neglected), so the the total aerodynamic force can be simplified into a drag and a lift force.

$$L = 1/2\rho A(v_a)^2 C_L$$

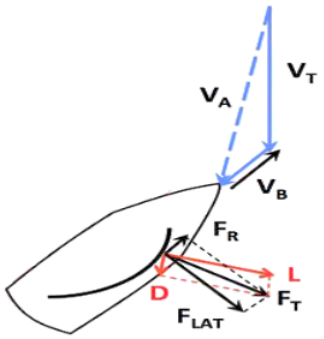


Figure 25: Lift and Drag

$$D = 1/2\rho A(v_a)^2 C_D$$

where ρ is the density of the water, A is the surface area of the rigid foil, Va is the fluid apparent velocity, CL and CD are lift and drag coefficients respectively. These coefficients depends on the mediums angle of attack, α and the foils shape. The drag and lift coefficients could be approximated by:

$$C_L = k1sin(2\alpha)$$

$$C_D = k1(1 - cos(2\alpha))$$

That net force is what propels the sailboat, nonetheless, are resisted by forces generated by the interactions between the water and the boat, given that, its possible to decompose the total aerodynamic force into a driving force and a lateral force:

$$\vec{F_T} = \vec{F_R} + \vec{F_{LAT}}$$

The driving force is resisted by the forward motion (contributes to the boat's velocity) and the lateral force by the resistance from the keel and lateral motion (contributes to the boat's orientation). Those 2 vectors relate with Drag and Lift in the following way:

$$F_R = Lsin(\alpha) - Dcos(\alpha)$$

$$F_{LAT} = Lcos(\alpha) + Dsin(\alpha)$$

The sail angle of attack is defined as $\delta_s - AWA$ and assuming the wind is the only fluid interacting the sail, then the fluid apparent velocity is equal

to the apparent wind vector magnitude (AWS). Therefore, the total aerodynamic force $\vec{F_T}$ applied on the center of the sail is equal to:

$$f_s = p_4 AWS sin(\delta_s - AWA)$$

$$Fs_x = f_s sin(\delta_s)$$

and

$$Fs_y = f_s cos(\delta_s)$$

, decomposed to the x and y direction, respectively.

The rudder angle of attack is just δ_r and by assuming that the only fluid interacting with the rudder is the water, then the apparent water and sailboat speeds are equal, therefore the water generates a hydrodynamic force acting on the rudder that is equal to:

$$f_r = p_5 u^2 sin(\delta_r)$$

$$Fr_x = f_r sin(\delta_r)$$

and

$$Fr_y = f_s cos(\delta_r)$$

, decomposed to the x and y direction, respectively.

The sailboat while moving is counter-acted by forces that resists motion. The net force can be broadly composed of the following types of forces:

- Inertial forces (produces a resistance force when the boat is moving, both in forward and lateral motions).

- Damping forces (forces act in opposition to the boat's motion reducing its velocity). The linear or tangential damping force is assumed to be equal to

$$pv^2$$

, where v is the boat's forward velocity. The angular damping force is assumed to be equal to

$$p3\omega v$$

, where w is the sailboat heading rate.

Given the descriptions of the forces and dynamics involded, the state-space equations is composed by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{u} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} usin(\psi) - vcos(\psi) \\ ucos(\psi) + vsin(\psi) \\ \omega \\ (f_s sin(\delta_s) - f_r sin(\delta_r) + v\omega p_9 - p_1 u^2)p9 \\ (f_s cos(\delta_s) - f_r cos(\delta_r) - u\omega p_9 - p_2 v)p9 \\ (((p_6 - p_7 cos(\delta_s))f_s - p_8 f_r cos(\delta_r) - p_3\omega)/p10 \end{bmatrix}$$

The table below, presents the parameters p1-p10 (those values were taken directly from John Melin thesis [2]):

number	value	parameter
p1	0.03	drift coefficient
p2	40 kg/s	tangential friction
p3	6000 kgm	angular friction
p4	200 kg/s	sail lift coefficient
p5	1500 kg/s	rudder lift coefficient
p6	0.5 m	distance to sail CoE
p7	0.5 m	distance to mast
p8	2 m	distance to rudder
p9	300 kg	mass of the boat
p10	400 kgm2	moment of inertia

Table 1: Sailboat parameters

This set of parameters does not define the current small remote-controlled boat. The estimation of the parameters would require the execution of experiments on sea with data logging capabilities to perform analysis on the data a priori and with the aid of system indentification techniques, estimate the boat's parameters. Thats arguably a project on its own and is proposed as future work.

5 Auto Pilot

In this section, the autopilot code (Tasks Planner, Tasks Runner, State Controllers) is presented and discussed.

5.1 Tasks Planner

This high-level state machine (Figure 14) tells the sailboat what to do at all times via Tasks. The system loads waypoints or tasks, then sets the next task to execute. The task is executed until the Task Completed Flag is set, always calculating the desired heading correcting the boat's orientation to target at all times. The sailboat remains in this sate until all waypoints were reached or all tasks completed.

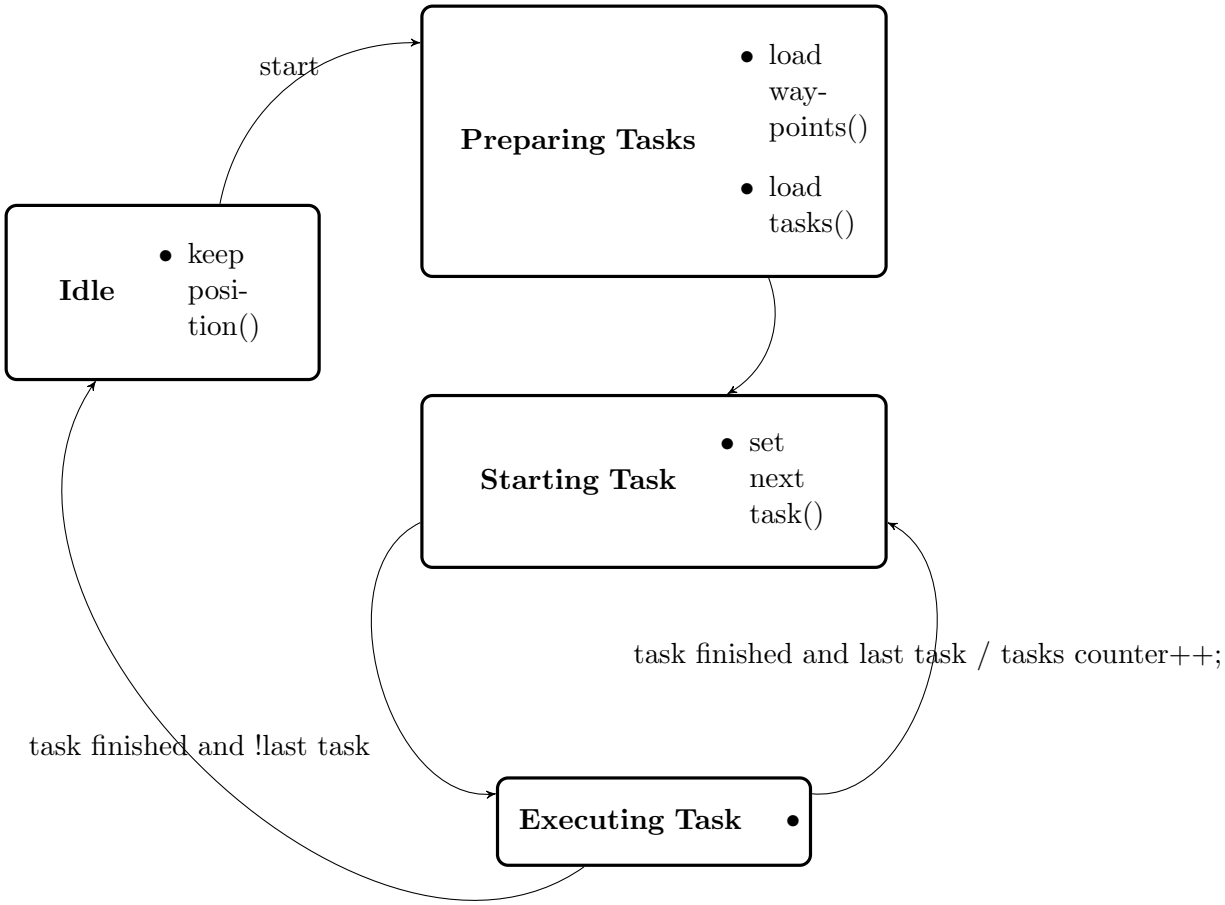


Figure 26: Tasks Planner state machine

5.2 Tasks Runner

There are multiples tasks the sailboat is able to perform, such as: Sailing to a waypoint, return to safety, station keeping and stop.

The most common task is sailing to a waypoint. The state machine to execute that task is shown in Figure 15.

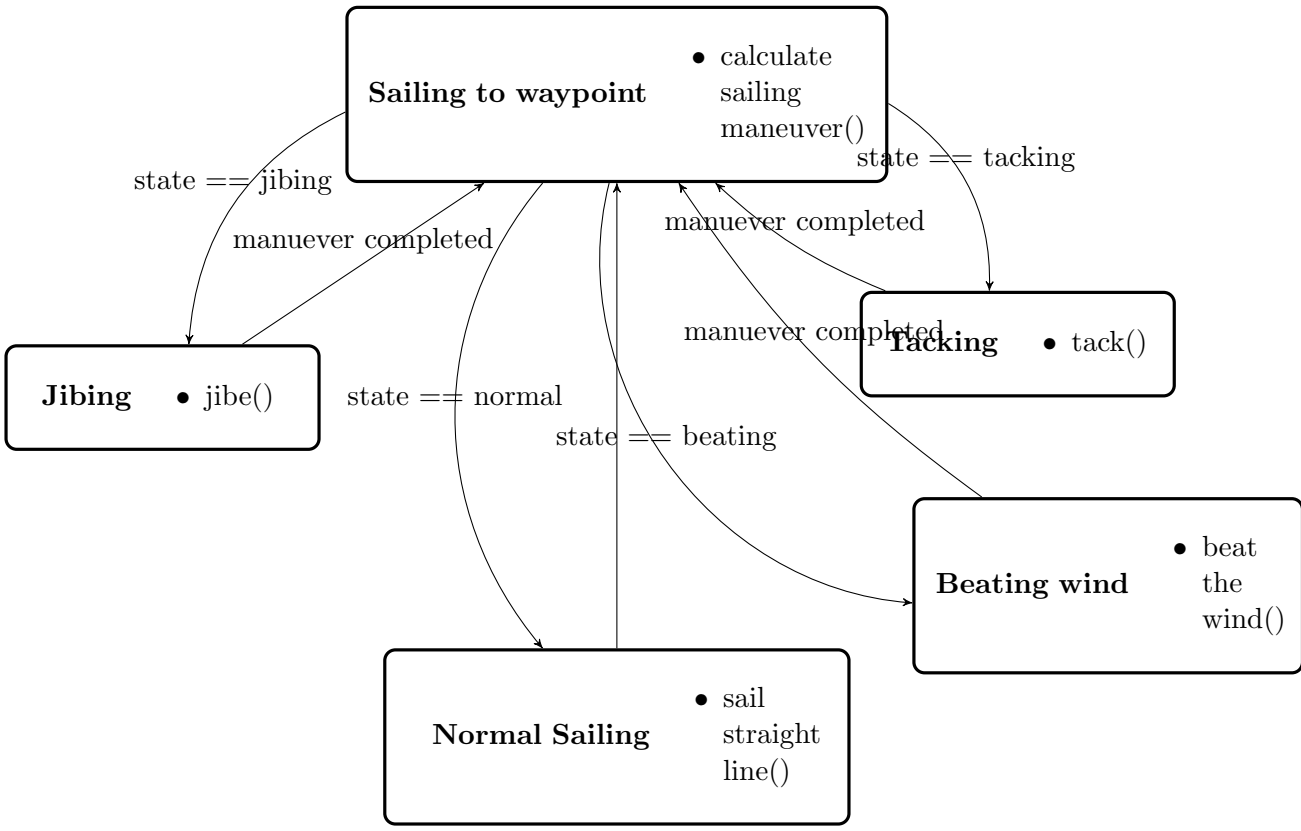


Figure 27: Tasks Runner state machine

Initially, target positions (waypoints) are provided, either by the autopilot or a base station and a Task is executed. The boat's current heading and the apparent wind direction are read to decide which sailing technique to perform (**calculate sailing maneuever**)
Depending on the apparent wind direction, the Tasks Runner may choose to:

•Sail directly to the waypoint (Normal Sailing);
•“Beat the wind”, if sailing directly towards the wind;
•Perform a ”tack”, it its currently against the wind, but the next waypoint is not;
•Perform a ”jibe”, it is currently running downwind, but the next waypoint is against the wind;

Notice, that switching from the states ’Tacking’, ’Jibing’ and ’Beating the wind’ to the ’Sailing to waypoint’ state occurs only when the current maneuever completes, see self-navigation is a dynamic task in a dynamic environment, thus a static state machine implementation is not feasible. Think, for example, the following situation: The boat is now heading towards the wind and the goal, so the tasks planner issues a beating the wind maneuever. The boat navigates 45 degrees (beating angle) plus its current orientation, as soon as the boats reaches this desired orientation, the tasks planner issues the boat to get back on the track, however the sailboat will sail against the wind once again, being issued one more time to steer far away from the wind, this occurs many times in short amount of time. See the picture 27 for the pictoral representation.

According to the maneuver, the rudder and sail angles is set and finally, the commands to the motors is sent and the task starts again.

6 Navigation Module

This modules exists as module itself, because its purpose is independently of the kind of robot system. It is responsible for loading the waypoints, calculate distances and heading between waypoints, converting apparent wind to the true wind and all calculations regarding manipulation of angles, such as determining if the boat’s heading is against or opposite to the true wind.

This module is directly tied with the points of sail, such it determines, based on the boat’s current heading and goal heading as well the true wind

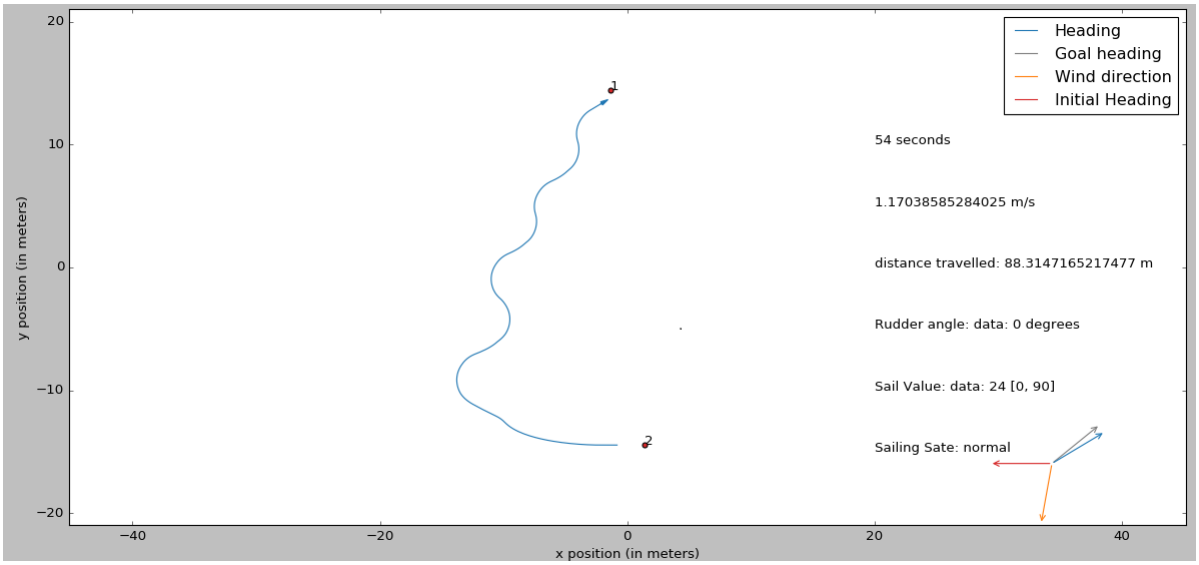


Figure 28: Many switches in small amount of time leads do oscillations and indecision around boundary conditions

angle, its positiion relative to the true wind.

angle	domain
heading	[0, 360] from the geographic north
heading to waypoint	[-180, 180]
apparent wind	[0, 360] from the geographic north
true wind	[-180, 180]

Table 2: Angles domain

When the boat is sailing under normal conditions, the rudder angle is determined by the PID controller. The error is the difference between the goal heading and the boat's current heading. Since the goal heading has angles between -180 and 180 and the boat's heading has angles between 0 and 360, a simple angle subraction yields a wrong result, therefore a transformation is necessary, given by the following code:

```
def angle_subtract(self, goal, current): # to keep the angle
    between 180 and -180

   angle = (goal-current) mod 360

        if angle > 180:
            angle = angle - 360

   if angle < -180:
       angle = angle + 360

        return angle
```

That way, the difference is always between -180 and 180 (shortest path). For example, if the goal heading is -30 degrees and the heading is 30 degrees, the difference is -60 degrees.

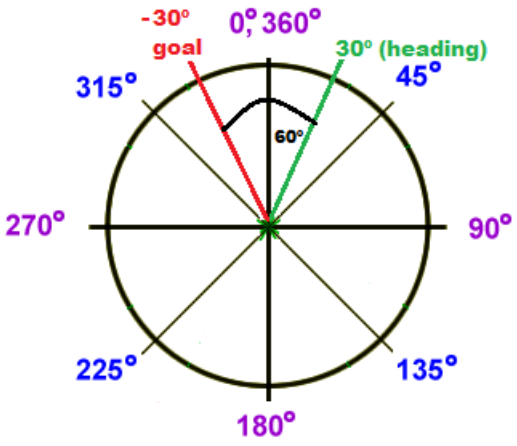


Figure 29: Sailboat used in the project

In the opposite, goal heading is 30 degrees and heading is -30 degrees (in fact 330 degrees), then the difference is now 60 degrees. In the case when the heading is 330 degrees, a simple subtraction would yield -300 degrees, which is wrong, the boat still reaches the goal heading, but it takes the longest path.

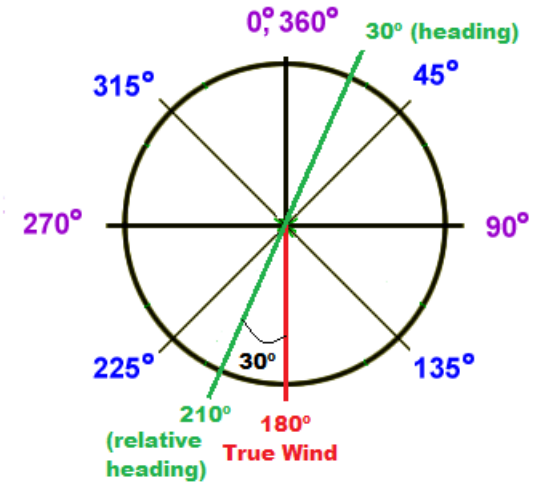


Figure 30: Sailboat used in the project

```
def heading_relative_to_wind(self, compass_angle,true_wind_angle):

   angle = compass_angle - 180
   return self.angle_subtract(angle, wind_angle)
```

6.1 State Controller

The purpose of this section, is to design and implement a low-level control architecture to get the sailboat from its initial position to the final position (given coordinates), autonomously. There are 2 state controllers: one for controlling the heading and other to sail against the wind and two low-level controllers: rudder and sail controllers. The state control implementation is described as follows:

- With the heading controller, the heading to target is provided as a set point to the rudder controller. The current heading is, continuously, estimated by the attitude estimator, feedback, then compared with the set point (error term) and serves as an input to the rudder controller;

- With the beating the wind controller, the setpoint is the beating wind angle (45° degrees) plus the boat's current orientation, compared with the apparent wind direction.

- The rudder controller feeds PWM signals to the rudder motor, forcing the sailboat current attitude to match the target attitude (set point).

- The Sail Controller, does not depend of the target orientation, it solely depends on the wind direction, measured by the anemometer. As depicted in previous sections, the angle of the mainsail, is determined by the sailboat point of sail (its orientation relative to the apparent wind direction).

- Performing a jibe or a tack maneuver, just sets the rudder (overrunning the rudder controller) to its maximum and minimum angle and the sail is trimmed according to the point of sail.

- The control loop runs until the distance to target is less than 10 meters. If there are more waypoints, the loop runs again, if not, a stop condition flag is set.

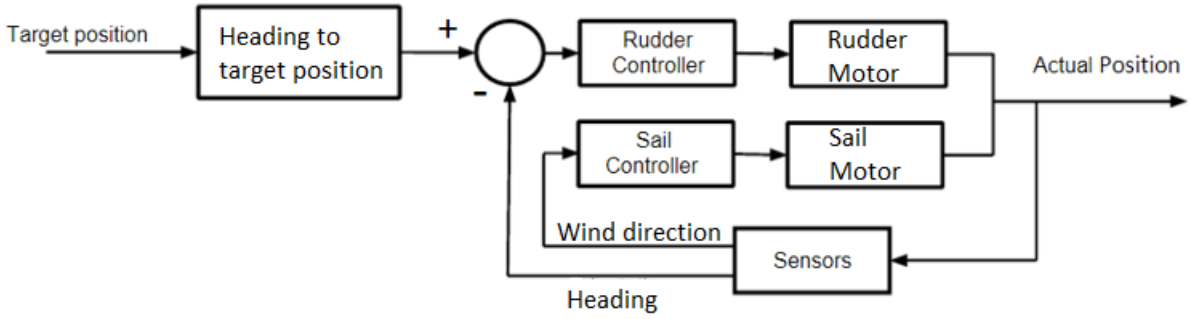


Figure 31: Heading Controller.

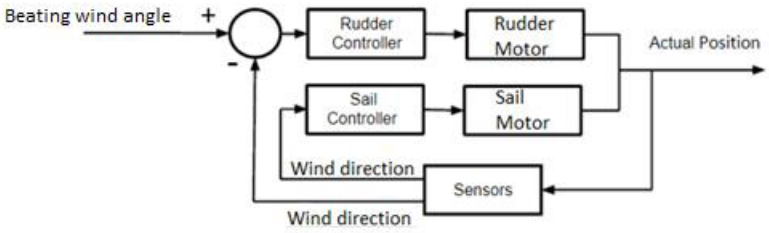


Figure 32: Beating the Wind Controller.

The angle of the mainsail, can be determined by the following formula

$\delta_s = \min\ \delta_s + (\max\ \delta_s - \min\ \delta_s) \times \frac{|TWA|}{180}$

Where min δ_s, is the angle of the sail when the rope is fully tight (5°), and max δ_s the angle when the rope is totally free (40°). Described in above sections, TWA is the direction of the true wind.

Rudder Controller

The rudder is responsible for controlling the heading of the sailboat, forcing it to match the target orientation, thus reducing the error term. The control signal, u_r, is determined by using the PI control loop on the error term ϵ. First, lets the define the error term:

$$\epsilon = \phi_g - \phi_c,$$

Where ϕ_g is the desired angle (heading goal) and ϕ_c is the boat's current orientation. The control signal, u_r, is the output signal of the rudder controller computed as follows:

$$u_r = K_p\epsilon + K_i \times \int_0^t \epsilon dt$$

But, the PI controller is implemented in a microcontroller, so we must compute a discretized version, such as follows:

$$u_r = P_k + I_k \text{ , where } P_k = K_p\ \epsilon_k$$
$$I_k = I_{k-1} + K_i \epsilon_k Ts$$

Where Ts is the sampling time, the rate which the control signal is evaluated. The sampling time was defined as 10 Hz.

The gains used in the PI rudder controller were:

Kp	Ki
1	0.1

Table 3: Rudder controller PI gains

7 State Estimator

In this chapter, the motivation to use a state estimator module is presented, the state estimation problem is formulated and a solution, based on the Linear/ Extended Kalman Filter (EKF), is shown and discussed.

7.1 Motivation

In order to the sailboat accomplish its tasks, it's necessary to provide the system precise information about its state, despite the presence of measurement noise and model uncertainty.

The sensors provide information about the state of the system, with some error, caused by the computation as well by the intrinsic inaccuracies of the sensors itself, such as a non-zero variance, the resolution, the frequency of the measurements, interferences, etc, as well model uncertainties, such as an unaccounted non-linearities, unknown parameters, imperfections in the model, etc.

We also know, given the dynamics of the system, $\dot{\vec{X}}$, how the sailboat moves, and how the inputs sent to the motors, affect the dynamics, but we don't know everything about its motion, the model is not perfect, for example, in this work, we are disregarding complex fluid dynamics and other types of forces. Our prediction tells us something about the state but with uncertainties. The state estimator proposes to reduce it by combining mesurements from the sensors and predictions from the model,

to estimate the next state of the system, capable of updating the information about the states without new measurements or sensor failure.

If both, process noise (disturbances) and measurement noise is assumed to be Gaussian distributed variables then the state estimator is reduced to a Linear/Extended Kalman Filter.

7.2 Problem Formulation

The boat's state changes due to external inputs (disturbances) such as the wind, current, waves motion and control inputs. The system's behavior is described by the given inputs (with unexpected and non-deterministic disturbances) and the measured outputs (with sensor noise and errors in the system model). Sensors on-board such as the GPS, a compass and the IMU are able to provide that information. (wind vane will not be discussed in application of the Kalman filter).

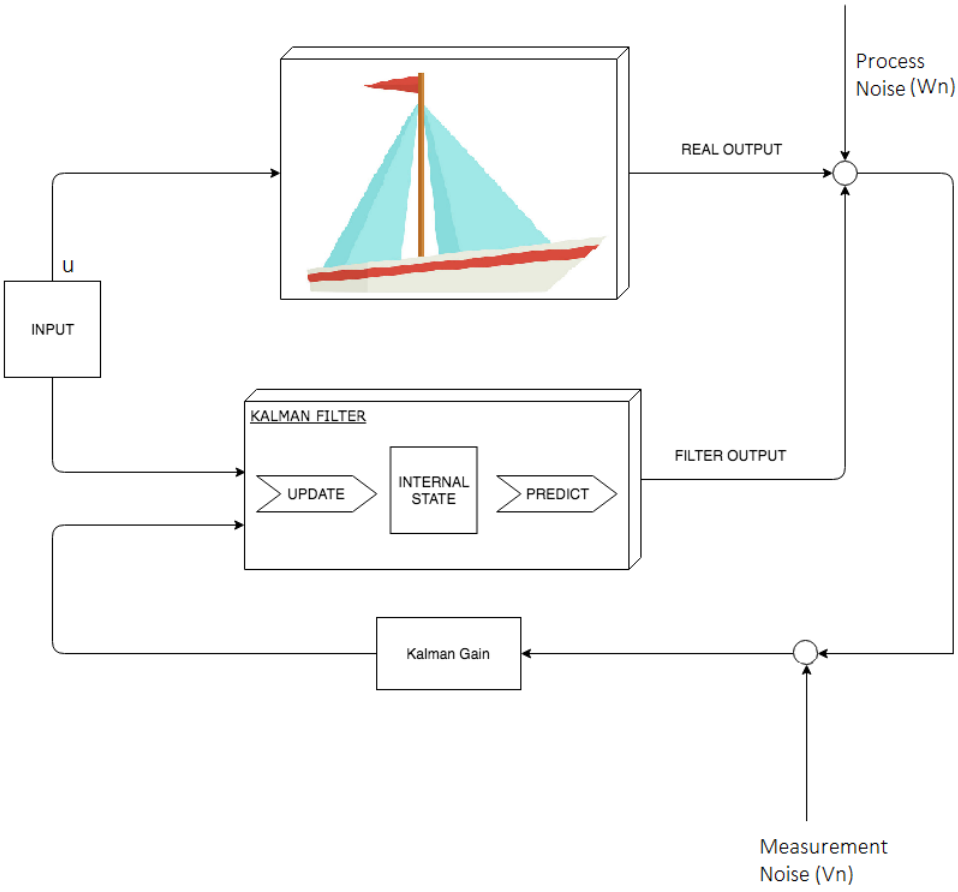


Figure 33: State estimation problem formulation.

The GPS sensor provides measurements: of the boat's position $\vec{p}$, accurate to 3 meters; of the boat's absolute speed, $\dot{p}$ via the Doppler Effect with accuracy to 0.05 m/s. But it has a low sample time (5 Hz), though not critical for systems with slow-varying velocity.

The accelerometer measures the 3 axis linear acceleration vector ($\vec{a}$) with high accuracy and with integration over time,

$$\int_0^t a \, dt$$

yields velocity and position by

$$\int_0^t v \, dt$$

But with every integration a drift error occurs, which makes, in long term, its use unpraticable. It is sensitive to temperature and its efficiency degrades over time.

Accelerometers, can measure indirectly roll and pitch angles, assuming the only force acting on the accelerometers was gravity. Vibration and other external forces will directly affect pitch and roll angles, making those signals very noise. Notice that the accelermeter can't measure the heading.

This technique estimate angles that aren't sensitive to vibration and other external forces, present in the accelerometers. but gyroscopes are noisy and imperfect, every time we add new gyro measurements, we add errors to our angle estimates.

The Magnetometer measures the strength of the magnetic field in the 3 axis, as well. Magnetic North is the direction of the horizontal component of the Earth's magnetic. The True North Pole is the axis of the earth's rotation, is the direction along Earth's surface towards the geographic North Pole North is equal to the True North, except in the poles. The angle between the direc-
tion of local Magnetic North and True North is called 'magnetic declination.

The magnetometer is also prone to errors, due to magnetic interference, local changes in magnetism, proximity to electric devices, being necessary to combine its measures a gyroscope and an accelerometer hence the Kalman Filter.

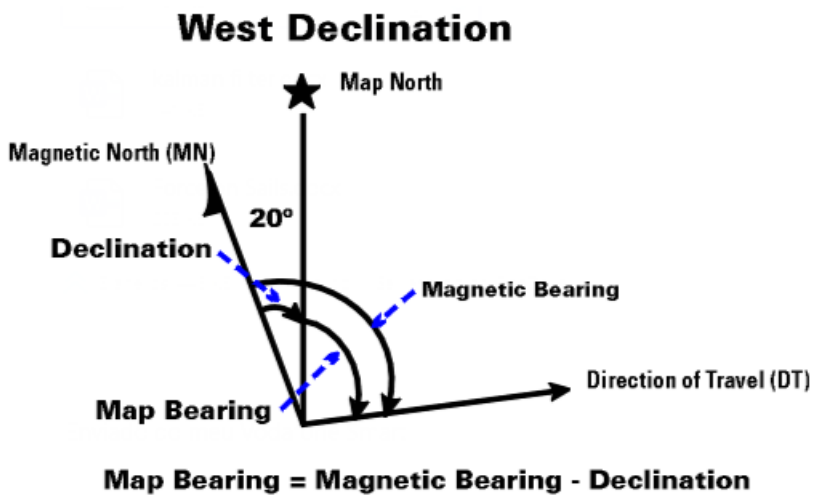


Figure 34: Difference between Magnetic North and Geographic North.

7.3 Solution

The Kalman Filter is an optimal estimation algorithm, combining noisy measurements from sensors and uncertainties about a dynamic system, to produce the optimal estimate of the current state of the system, but the boat's dynamics is a set of non-linear differential equations, consequently, the process and measurement noise are no longer Gaussian distributed variables, so a Kalman Filter is a not feasible technique. To solve this problem, at each time step, k, the system dynamics is linearized around the previous estimates, X_{KP} and control inputs u_k, resulting in the 'Extended' Kalman Filter, notice that this filter is not an optimal estimator like the Kalman Filter linear counterpart.

The system dynamics is given by

$$X_K = f(\ X_{K-1}, u_K) + w$$

Where:

- f is the non-linear discretized system equations and k is the current time step, $\vec{X} \in R^n$ is the state vector.

- $\vec{u} \in R^n$ is the control vector

- $\vec{w} \in R^n$ is the process noise, assumed Gaussian distributed random variable, with mean zero and covariance, Q.

At the time instant k, the state vector, is measured $\mathbf{Y}_K$, according to

$$\mathbf{Y}_K = \mathrm{h}(\ \mathbf{Y}_{KM}\ )$$

Where:

- h is the sensor model, which tells what states are measured and how.

- $\vec{v} \in R^n$ is the sensor noise, assumed Gaussian distributed random variable, with mean zero and covariance, R_K .

The Kalman filter assumes that the state variables are random and gaussian distributed with mean value, μ and covariance σ^2 . Its not possible to exactly know and measure the actual state but it's possible to predict the most likely state.

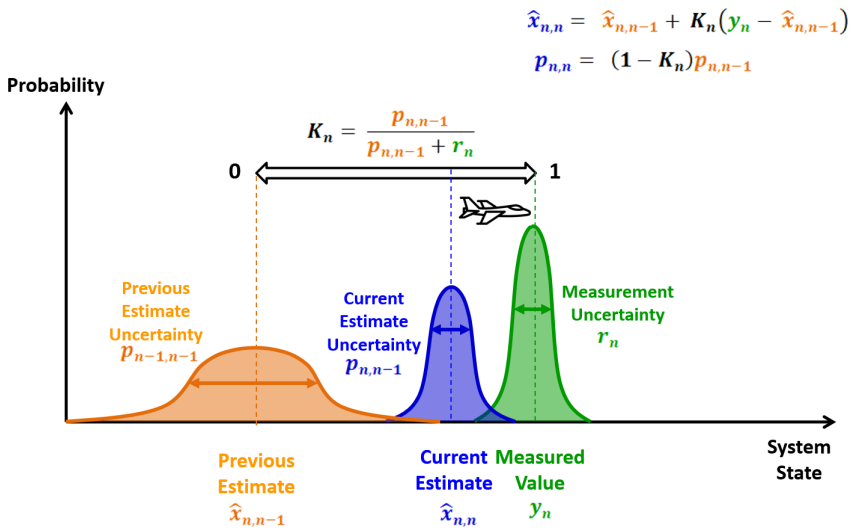


Figure 35: Pictorial representation of the Kalman Filter.

The state estimate vector, $\mathbf{X}_{KP}$, is the mean vector. The prediction matrix, $\mathbf{P}_{KP}$, is the covariance matrix , such X $\tilde{\mathrm{N}}(\mathbf{X}_{KP},\mathbf{P}_{KP})$, yields the following equations:

$$\begin{aligned} \mathbf{X}_{KP} &= \mathrm{f}(\ \mathbf{X}_{KP-1},\mathrm{u}_K) \\ \mathbf{P}_{KP} &= \mathrm{F}_K\ \mathrm{P}_{KP-1}\ \mathrm{F}_K + Q^T \end{aligned}$$

where $\mathrm{F}_k = \frac{\partial f}{\partial x}\ (\ \mathbf{X}_{KP}\mathbf{,u}_K\ )$ is the jacobian of the vector-valued function f at the points X_{Kp} and u_k

The state measured vector, $\mathbf{Y}_{kp}$, is the mean vector and $\mathbf{S}_k$ is the covariance matrix, such y $\mathrm{N}(\ \mathbf{Y}_{kp},\mathbf{S}_k)$, give the following equations:

$$\begin{aligned} \mathbf{Y}_{kp} &= \mathrm{h}(\ \mathbf{X}_{KP}\ ) \\ \mathbf{S}_k &= \mathrm{H}_k\mathrm{P}_{kp}\ \mathrm{H}_k{}^T + \mathbf{R} \end{aligned}$$

,where $\mathrm{H}_k = \frac{\partial h}{\partial x}\ (\ \mathbf{X}_{KP}\ )$ is the jacobian of the vector-valued function h at the points X_{Kp}

There are two Gaussian distributions: One for the state estimation (based on the model), one for the state observed (based on the measurements). Therefore, are combined to produce the optimal state vector estimate, as follows:

$\mathbf{X}_K = \mathbf{X}_{KP} + \mathbf{K}[\mathbf{z}_k$ - $\mathrm{H}_k\mathbf{X}_{KP}]$, this computation is the innovation step, where $\mathbf{z}_k$ contains the measurements in the time step k .

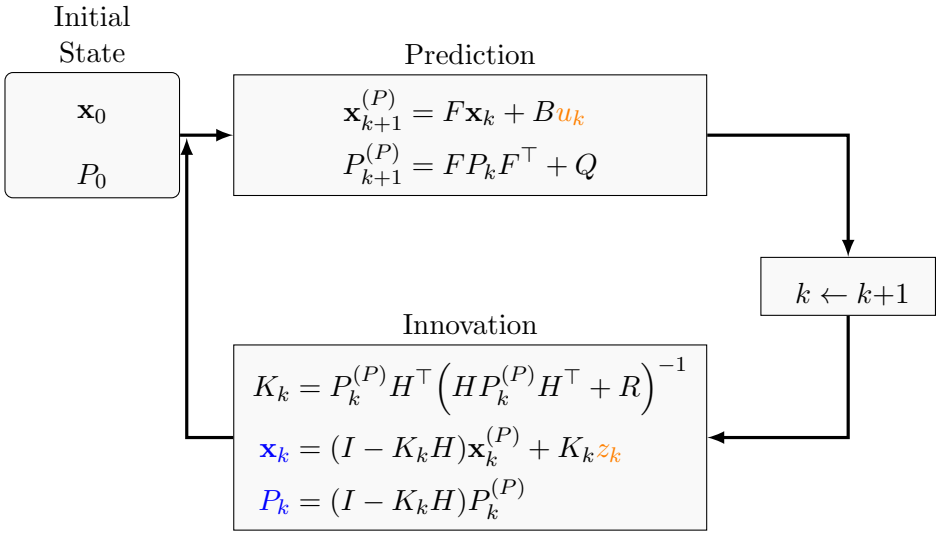
K is the Kalman Gain, computed as such:

$$\mathbf{K}_k = \mathbf{P}_{KP}\ \mathrm{H}^T\mathbf{S}_{k-1}$$

The Kalman Filter is an iterative algorithm, before running the loop again, the prediction must be updated:

$$P_K = (I - \mathbf{K}_k H_k) P_{kp}$$

The figure below summarizes the Kalman Filter equations.



8 IMU data fusion

8.1 Preliminaries

The motivation to implement a filter on the IMU and the magnetometer was already discussed, but before dwelving deep into the details of the implemented sensor fusion algorithm, its noteworthy to present some of the common approaches found in the literature. There are multiple solutions to estimate the Euler angles, some purely academic (read not used in industrial applications) [19] , like Adptative [20] [21], Error States Kalman Filters [22] with or without quaternions in the state vector, as well some other, industrially "proven", like:

- Complementary filter : a simple filter that combines roll and pitch angles calculated from the accelerometer readings with the roll and pitch angles by integrating the gyro angular rates over time (6 DOF). This method is very common and similar in performance to a Kalman Filter, in 6 DOF [REF], however, its not suitable for a 9 DOF (including a compass), because the main issue of the magnetometer readings is not noise, but rather unpredictable disturbances in the Earth's magnetic field, caused by

electrical motors, cables and natural anomalies.

- Mahony Filter: This filter [24] uses feedback theory, where the angles calculated from the acceleremoter are the setpoint and the rotation rates are disturbances.

- Madgwick Filter: Madgwick [23] uses a unit quaternion to represent the orientation of a rigid-body in 3-D space and uses a machine learning algorithm (like gradient descent) to minimize the error between the accelerometer and gyroscope readings.

- Linear/Extended Kalman Filter : The EKF is de facto standard in industry in the estimation of the Euler angles. Theres been effort in extending the fundamental theory of operation to another KF schemes, adpative, error states, neural networks to find its matrices. While their results are promising and their ideas intellectually challenging, due to limited time and the current scope of the project, those methods are not feasible. Understanding how filter works is one thing, how to design it properly is sure another! [18]

8.2 Sensors - Accelerometer

A 3-axis accelerometer measures linear accelerations, typically, modeled as

$$a_m = 1/m(\vec{F} - \vec{F_g})$$

, where $\vec{F}g$ is the force due to gravity and $\vec{F}$ is the net force acting on the body. Both forces are expressed in body frame. In order to calculate the Euler angles, the accelerometer should only measure the force due to gravity, so its assumes no acting external forces ($\vec{F} = 0$), linear acceleration in x and y axis are zero and equals to mg in the z axis.

$$\vec{F}g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}$$

then

$$\vec{a_m} = -1/m\vec{F_g}$$

, however its necessary to transform the force due to gravity (expressed in the body-frame) to the inertial frame, by a rotation matrix, as follows:

$$\vec{F}g = R_I^B \times \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} -mgsin(\psi) \\ mgcos(\psi)sin(\phi) \\ mgcos(\phi)cos(\psi) \end{bmatrix}$$

, dividing by -m yields,

$$\vec{a_m} = \begin{bmatrix} gsin(\psi) \\ -gcos(\psi)sin(\phi) \\ -gcos(\phi)cos(\psi) \end{bmatrix}$$

Solving for the roll ϕ and ψ angles, results in

$$\phi = arcsin(am_x/g)$$

$$\psi = arctan(am_y/am_z)$$

The derivation, assumed no external forces acting on the body, of course this is not true, accelerometers are sensitive to vibration and other non-gravity accelerations in the short-term. Those values should be combined with another sensor, the gyroscope.

8.3 Sensors - Gyroscope

A 3-axis gyroscope measures angular rates in the body-frame. A gyro is the opposite of an accelerometer, is unaffected by external forces (accelerations), unsensitive to vibrations and less noisy, however a gyro is a not perfect sensor, a static bias is the largest contributor of error, due to they way, how angles are estimated (by integration of the angular rates), errors accumulate over time causing the estimated angles from the gyroscope to drift. The solution to this problem will be presented and discussed in Sensor Fusion section.

A transformation from the body-frame to the inertial-frame must be carried out, however, the sailboat's rolling and pitching motions are neglected, the rotation matrix

$$R_I^B$$

is just a 3 by 3 indentity matrix, this approach avoids working with a non-linear equation, thereby avoiding the use of the Extended Kalman

Filter and the gimbal lock singualarities,but when desining the Kalman filter, this introduced error should be accounted for.

8.4 Sensors - Magnetometer

A 3-axis magnetometer measures the earth's magnetic field strength, expressed in the body frame. The approach used to estimate heading is to compute the orthogonal components (Bx/By) of the magnetic vector as follows:

$$\text{heading} = \text{Magnetic North - Declination angle}$$

$$\text{heading} = \arctan2(-B_y, B_x) \text{ - Declination angle}$$

Earth's magnetic field changes slowly overtime, so similar to the gravity vector, earth's magnetic north its used as a reference vector, in other words, those sensors are absolute, unlike the gyroscope which is relative, in a sensor fusion algorithm, the gyroscope's initial conditions, required for the integration computation are set by this two sensors. Altough, the earth's magnetic field does not change rapidly, the local magnetic field measured by the magnetometer is prone do distortion, mainly by two types of distortion:

- Hard iron : Its an permanent external magnetic field, adding at all times, to the unaffected local magnetic field, for example, the sailboat has 2 servo motors (located in the center of the boat) which by themselves creates a magnetic field, since those servo motors will not change location or direction troughout the navigation, those distortions are additive and permanent (bias), luckily those types of interferences can be easilly removed by subtracting the offset in the calibration procedure.

- Soft iron : This type of distortion occurs when a material like copper, used in electrical wires, is in the proximity of the sensor, generating a distortion ependent on the orientation of the material relative to the compass, since its not a permanent source of interference, one can't simply subtract an offset, correcting this type of distortion requires a more sophisticated, yet more computationally expensive, method, for example a 3x3 transformation matrix. In this project, will be used a simpler and less expensive algorithm, yet as effective as the matrix transformation,

remember a Kalman Filter is used to combine the data from all 3 sensors, its more than good enough, there is no need for a more expensive method.

8.5 Sensors - Calibration

Static bias values and noise properties are readily available in the IMU datasheet, those values were indentified across an entire range of temperatures, surfaces and motion conditions, nevertheless an experiment was conducted to estimate those parameters in order to obtain the best accuracy. The IMU was held stationary on a flat surface (close) for a minute and half, all datas was stored on a file. Each of one the values in the x,y and z axis were stored in an array making possible to do statistical analysis (i.e to find the mean and the standard covariance of each one of the sensors and their 3 axis values). Once the mean value is determined, its possible to simply subtracting this value to new measurements.

Below, a calibration procedure for each of one the sensors that composes an IMU will be discussed.

Note, those calibrations can and should execute before a new navigation commences. Those routines runs automatically when the autopilot initiates.

8.6 Magnetometer

As explained before, hard irons are permanent and additive, so the offset is subtracted from new measurements, but one can't simply calculate the compass stationary readings mean. Unlike accelerometers and gyroscopes that, in an ideal world would measure zero acceleration and 0 zero angular rate, the magnetometer dosen't measure zero magnetic field strength while at zeros degrees from the magnetic different, therefore holding the IMU stationary is not a good experiment to estimate the magnetometer bias. The sensor was moved in a big figure eight, rotating in multiple times around the X, Y and Z axes.

The offsets are calculated, as proposed below:

```
offset_x= (max( mx ) + min( mx ) ) / 2
offset_y = (max( my ) + min( my ) ) / 2
offset_z = (max( mz ) + min( mz ) ) / 2
```

The scales factor:

```
 scale_x = (max( mx ) + min( mx ) ) / 2
 scale_y = (max( my ) + min( my ) ) / 2
 scale_z = (max( mz ) + min( mz ) ) / 2
```

So, the readings are calibrated with subtracting the offset and dividing by the scale factor:

```
calibrated_mx = (mx - offset_x) / scale_x
calibrated_my = (my - offset_y) / scale_y
calibrated_mz = (mz - offset_z) / scale_z
```

Every time a new measurement is taken, its multiplied by its respective scale factor.

8.7 Accelerometer

The accelerometer calibration routine is done by calculating the mean value of each of the 3 axis, stored on a non-volatilve memory device (i.e SD Card) and every time, an acceleration value is read, subtracts the offset value.
accelerometer bias (in m/s2) x axis = 0.22894 ; y = 0.015338 ; z = -0.025839 (excluding the constant 9.81)

8.8 Gyroscope

No calibration (bias removal) was performed. The implemented Kalman filter will include 3 additional states (3-axis gyroscope bias in degrees/second).

8.9 Sensor Fusion Design

To design a KF, one must construct the matrices F,B, the states and inputs vector, x and u respectively (in other words build a right model for

the task at hand). There is two main approaches in designing a state space model:
- Build a state vector with a unit quaternion, the matrix F is based on quaternion algebra and calculus
- Design the KF state vector as such:

$$\vec{X} = \begin{bmatrix} \psi & \phi & \psi \end{bmatrix} \tag{1}$$

But there is a problem this state vector formulation. In the innovation step, because of angle overflows (transitions from -180 and 180 degress for example, gives a false impression. For example, the angle changes from 180 to -180 yielding an error of 360 degreees. A minimum angle difference calculation can be applied to give zero degrees, however this correction is done in the innovation step, the error in already there in the predicition step, so a new state vector is formulated, based on the difference of the angles between two time steps.

Another issue is the drift, caused by integrating the gyroscope angles with a bias value, eventually making the drift to be large enough that the Kalman Filter, no matter how well designed, won't estimate the angles properly, the state vector expands with the inclusion of the bias values of all 3 axis of the gyroscope and at each time step, the bias values are estimated and subtracted, dynamically from the corresponding gyro readings.
Gven that description, the error states formulation with bias estimation is given by:

$$\vec{X} = \begin{bmatrix} \psi & \phi & \psi & bx & by & bz \end{bmatrix} \tag{2}$$

The rest of the matrices are constructed as follows:

$$\vec{u} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix} \tag{3}$$

$$\vec{z_k} = \begin{bmatrix} \psi_a & \phi_a & \psi_{mag} \end{bmatrix} \tag{4}$$

yielding the state space model:

$$\begin{bmatrix} \delta\psi \\ \delta\phi \\ \delta\psi \\ bx \\ by \\ bz \end{bmatrix}_{k+1} = \begin{bmatrix} O_3 & -I_3 \\ O_3 & I_3 \end{bmatrix} \times \begin{bmatrix} \delta\psi \\ \delta\phi \\ \delta\psi \\ b_x \\ b_y \\ b_z \end{bmatrix}_k + \begin{bmatrix} I_3 \\ O_3 \end{bmatrix} \times \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$y_k = \begin{bmatrix} I_3 & O_3 \end{bmatrix} \times \begin{bmatrix} \psi \\ \phi \\ \psi \end{bmatrix}_{k+1}$$

The covariance matrix Q, represents how much we thrust the model (uncertainties) and covariance matrix R, tells how noisy are the measurements. The values for the matrix R were initially chosen based on the variance of each signal (calculated with while the IMU and the compass were held stationary) and tweaked until results were satisfactory. The matrix Q was initially the indentity matrix and tuned as necessary.

$$Q = \begin{bmatrix} I_3 * 0.001 & I_3 * 0.01 \end{bmatrix}$$

$$R = \begin{bmatrix} 12 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 12 \end{bmatrix}$$

8.10 Results and Conclusion

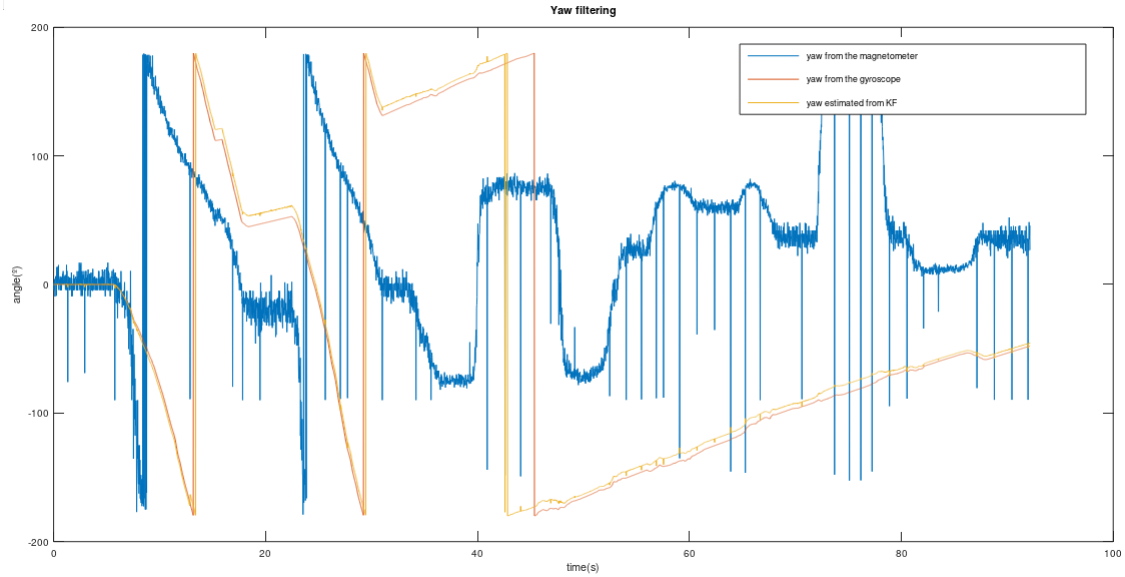


Figure 36: estimated yaw angle from the implemented Kalman Filter.

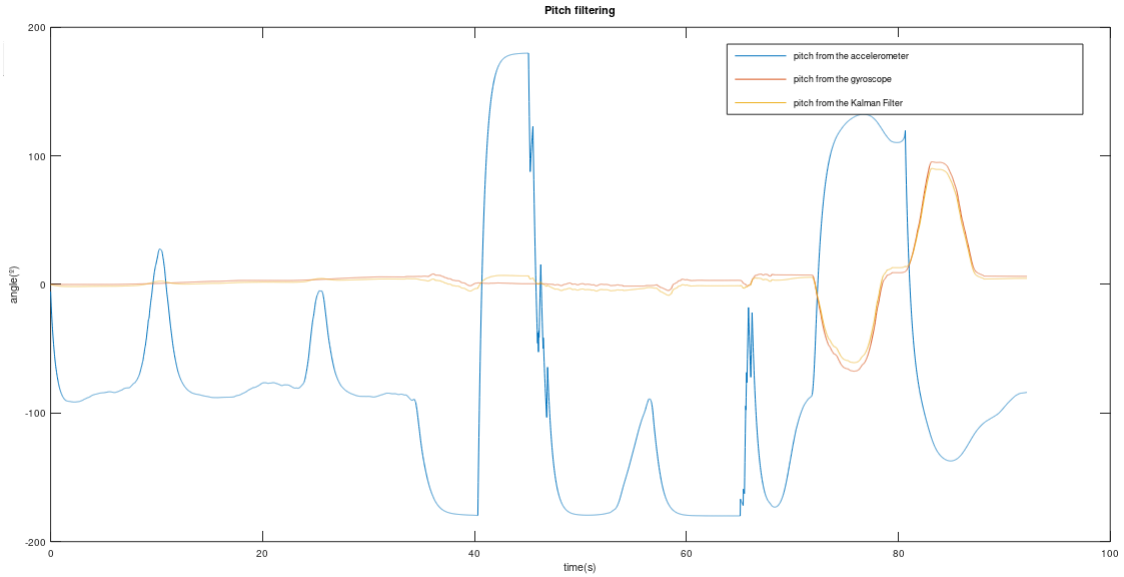


Figure 37: estimated pitch angle from the implemented Kalman Filter.

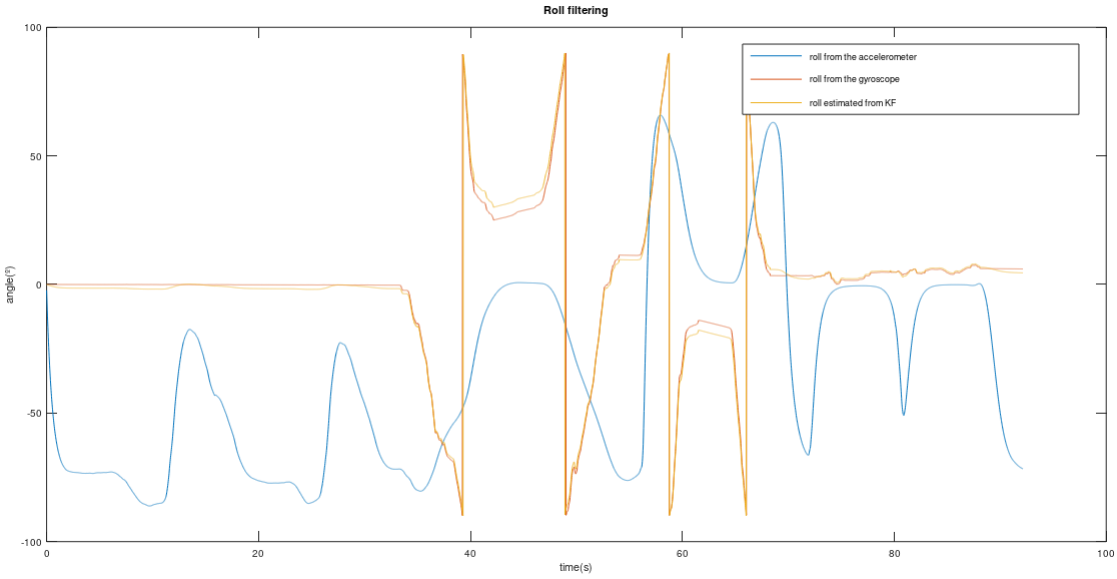


Figure 38: estimated roll angle from the implemented Kalman Filter.

Those values were measured, with the following sequence of actions:

- Starts at 0 degrees of pitch, yaw and roll;
- Rotates two times around a circle (makes 2 full 360 degrees rotations);
- Performs a roll motion, from 0 degrees to one side and to another;
- Finally executes a pitch motion, from 0 degrees to down position and to the down position.

As, shown in the plots, the KF measures all 3 angles, in all of the actions performed, despite the large noise found in the both accelerometer and the compass readings. Notice, especially, in the pitch plot, how much the angle caculated from the gyroscope drifts over time, altough the KF drifts as well, but less than the gyro readings. The results could be more satisfactory, the gains need more tweak, but its extremely time consuming by trial and error, a auto-tuning method should be investigated in order to find the best possible gains.

Side note 1 : There are some abrupt changes in the plots above, that happens when the angles transits between the limits, such as 180 degrees to -180 degrees or 90 degrees to -90 degrees.

Side note 2 : The yaw/heading value is constrained to -180 and 180 degrees, for estimation only, because this angle is calculated from computing a arctangent function, however, the heading is further converted to the domain utilized in the navigation module, which is 0 degrees to 360 degrees.

9 Wind filter

The wind vane used in this project is an analog sensor, where the sailboat's direction relative to the apparent wind maps to a voltage value ,with a sample rate of 10 Hz. Like the other sensors on board, its also senstitive to noise. The filter employed to smooth the wind direction data (suppressing the higher frequencies) was a simple moving average (Low Pass Finite Impulse Response Filter). Since wind changes slowly over time (slow dynamics) the time it takes the moving average filter to estimate a wind direction is not an issue.

The implementation is simple, five measurements from the wind vane are stored at all times in an fixed-size array and its average is computed, in the next time step, the fifth (last) measurement is removed from the array and replaced by the new one, meaning that each new estimation n can be computed with only two additions, instead of the 5 additions.

Note, its very important to calibrate the sensor, before the boat navigates. The wind vane should be mounted as high as possible away (i.e. top of the mast) from any part of the vehicle that may disturb the air flow around the vane.

10 Position and Orientation (POSE) and Velocity estimators

This estimator is implemented with the Extended Kalman Filter. The The vector-valued function f($\vec{X}$], u) is the model of the sailboat described in chapter 4. The goal of this estimator is to determine the position, orientation and the velocities of the sailboat (the linear and the angular velocities serves only for instrumentation and visualization purposes, its not necessary for control). One question immediately arises, "why there is

an addition orientation estimation?", because the GPS can also measure the orientation, moreover, the mathematical model can predict it, more predictions and more measurements, the merrier.

The EKF matrices were designed as follows:

$$\vec{X} = \begin{bmatrix} x & y & \psi & \dot{x} & \dot{y} & \dot{\psi} \end{bmatrix} \tag{5}$$

$$\vec{u} = \begin{bmatrix} \delta_s & \delta_r & TWA & TWS \end{bmatrix} \tag{6}$$

$$\vec{z_k} = \begin{bmatrix} x_{gps} & y_{gps} & \psi_{gps} & v_{gps} & \psi_{IMU} \end{bmatrix} \tag{7}$$

$$Q = \begin{bmatrix} I_6 * 0.05 \end{bmatrix}$$

$$R = \begin{bmatrix} I_3 * 0.3^2 \end{bmatrix}$$

The GPS, has a variance of 30 cm (value found in the datasheet), the other values were initially equal to 1.

The results will be in the 'results and conclusion' section, when its possible to see the sailboat navigate while correcting its course dynamically, with the estimated heading from the POSE and Velocity estimator as the feedback path.

11 Simulation and Results

In this section, the simulated results are shown and commented. The sailboat was tested in 2 different scenarios: running towards the wind with jibe or tack maneuver and sailing between multiple waypoints. The sailboat's current orientation is the heading estimated from the POSE and Velocity estimator module.

11.1 Sailboat running towards the wind with jibe maneuver

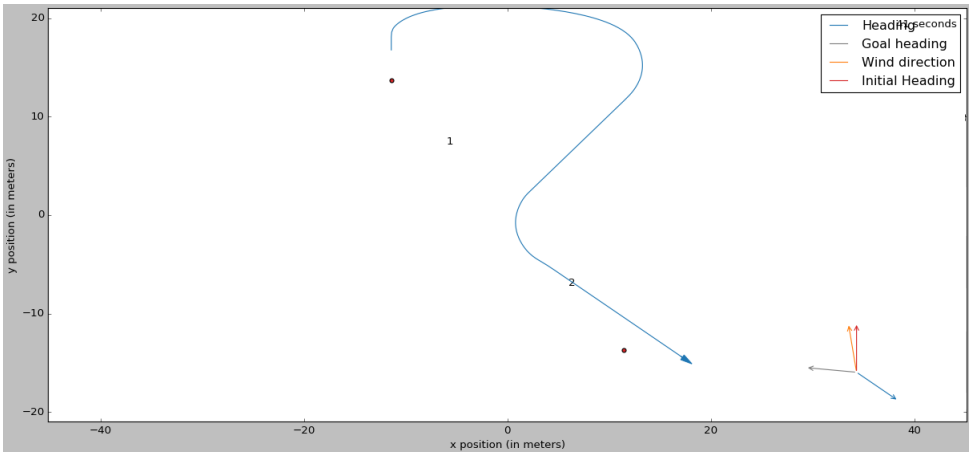


Figure 39: sailboat performing a jibe maneuever.

In this scenario, the sailboat starts with a initial heading opposite to the wind (running) and the goal heading is against the wind (zero degrees relative to north), so it performs a jibe maneuver on port to starboard in the beginning and since thus it can't sail directly to target, it must "beat the wind", executing a zigue-zague movement towards the next waypoint.

11.2 Sailboat running towards the wind with performing a jibe maneuever in the beggining

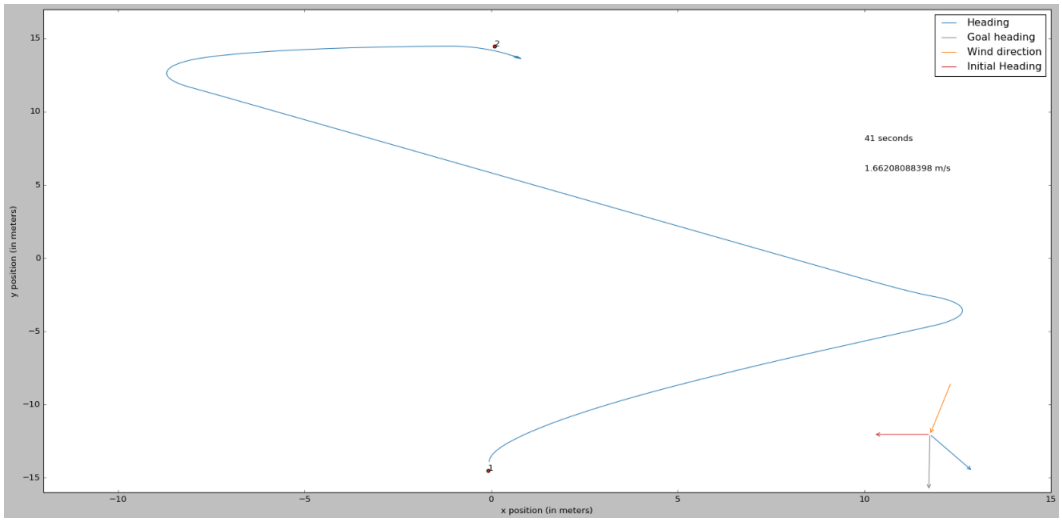


Figure 40: Beating the wind with a Tack maneuver with wind direction at zero degrees.

In this scenario, the wind is against the sailboat trajectory, thus it can't sail directly to target, it must "beat the wind", performing a tacking maneuver on port to starboard (zig-zag) until it reaches the target point.

11.3 Complete path

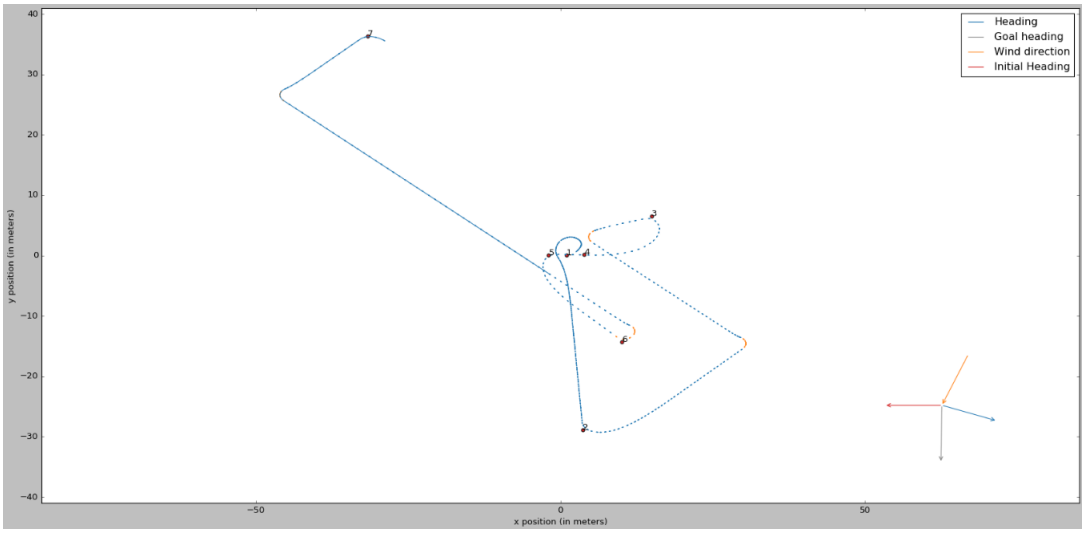


Figure 41: Sailing between multiple waypoints with both jibe and tacking with wind direction at zero degrees

Figure 42 shows the sailboat navigating throughout multiple waypoints, facing multiple wind directions throughout the course. From waypoint 1 to waypoint 2, the wind is blowing downwards, making the sailboat sail in a straight line. From point 2 to point 3, the sailboat travel's direction is now facing the wind, so it executes a tacking maneuver (switch starboard to port) approaching the waypoint with a jibe. From point 3 to 5 and from point 5 to point 6 it sails in a, relatively, straight line, because the wind is blowing in a favorable direction. From point 6 to 7, the wind is blowing upwards, so the boat must beat the wind, conducting a tacking maneuver, switching from starboard to port.

11.4 Conclusion

Introductory sailing concepts and navigation techniques was presented and shown in the simulator. The equations used in the simulator wasn't described, due to number of pages constraint, but will be in the final report. The results shown in chapter 5, shows that the sailboat can navigate throughout multiple waypoints, with many points of sailing (chapter 2), such as: - When the wind is against the boat, it executes jibe

and tack maneuvers to beat the wind; - When the wind is behind the boat, it sails in a straight line; - In other wind directions, the sailboat executes both jibe and tack, as described in chapter 2.

11.5 References

[1] FASt - An autonomous sailing platform for oceanographic missions, Jose C. Alves Nuno A. Cruz
[2] Aland Sailing Robots - Modeling, control and state-estimation for an autonomous sailboat, John Melin
[3] System Indentification for precision control of a gps-autonomous catamaran, Gabriel Hugh Elkaim
[4] Design and Implementation of a Control System for a Sailboat Robot, N-BOAT
[5] Le Bars, F.; Jaulin, L. An Experimental Validation of a Robust Controller with the VAIMOS Autonomous Sailboat. In Proceedings of the 5th International Robotic Sailing Conference, Cardiff, Wales, UK, 17–21 September 2012
[6] Modeling and Nonlinear Heading Controlof Sailing Yachts
[7] Design and Implementation of a Navigation Algorithm for an Autonomous Sailboat (AVALON, ETH Zurich),
[8] Backstepping Control of an Autonomous Catamaran Sailboat
[9] Modelling and Control Design of a Robotic Sailboat, Sorbonne Université
[10] Adaptive Probabilistic Tack Manoeuvre Decision for Sailing Vessels, Mar 2019

Index