Nicolas VEYLON
Ecole Nationale Supérieur des Techniques Avancées Bretagne
Robotics

Arduino Powered Autonomous Rover (Report)

Proposed By:
Jian WAN
Lecturer in Control
Systems Engineering
University of Plymouth

Overseen By:
Luc JAULIN
Professor in Robotics
ENSTA Bretagne

June-August, 2017
University of Plymouth

ENSTA
Bretagne

Abstract

The purpose of this project is to develop an autonomous rover powered by Arduino as part of a feasibility study. As this project begins from scratch, a preliminary study of the different sensors is mandatory. In addition, a simulation has been developed in Python. It was used to test different navigation systems using Kalman filtering in a controlled environment. It results that the model used in the simulation does not fit with the real system. Nevertheless, the simulated rover manages to reach a target while avoiding obstacles despite the bad accuracy of the sensors employed.

Résumé

Le but de ce projet est de développer un rover autonome sous Arduino dans le cadre d'une étude de faisabilité. Comme ce projet part d'aucune base, une étude préliminaire des différents capteurs est nécessaire. De plus, une simulation a été développé sous Python. Elle a été utilisée pour tester différent systèmes de navigation utilisant du filtrage de Kalman dans un environnement controlé. Il résulte que le modèle utilisé dans la simulation ne correspond pas au système réel. Néanmoins, le rover simulé peut atteindre une cible tout en évitant des obstacles malgré la mauvaise précision des capteurs employés.

Contents

Introduction

Autonomous cars are increasingly present on the roads and the University of Plymouth is seeking for research in this field. In this context, Dr. Jian WAN proposed a small-scaled feasibility study that fits into a three months internship. The purpose of this project is to build an autonomous rover powered by Arduino. This project covers the study of the different sensors, the data processing, the hardware system integration, the navigation system using Kalman Filtering, but also a simulation based on research papers.

It is important to note that this project was shared with Sophie TUTON, another student from ENSTA Bretagne. Several parts of the project such as the Arduino sketches implemented on the rover or some studies on sensors do not appear in the report. Indeed, this report is only about what I have been working on during this internship.

Chapter 1

Build a Rover

Global Overview

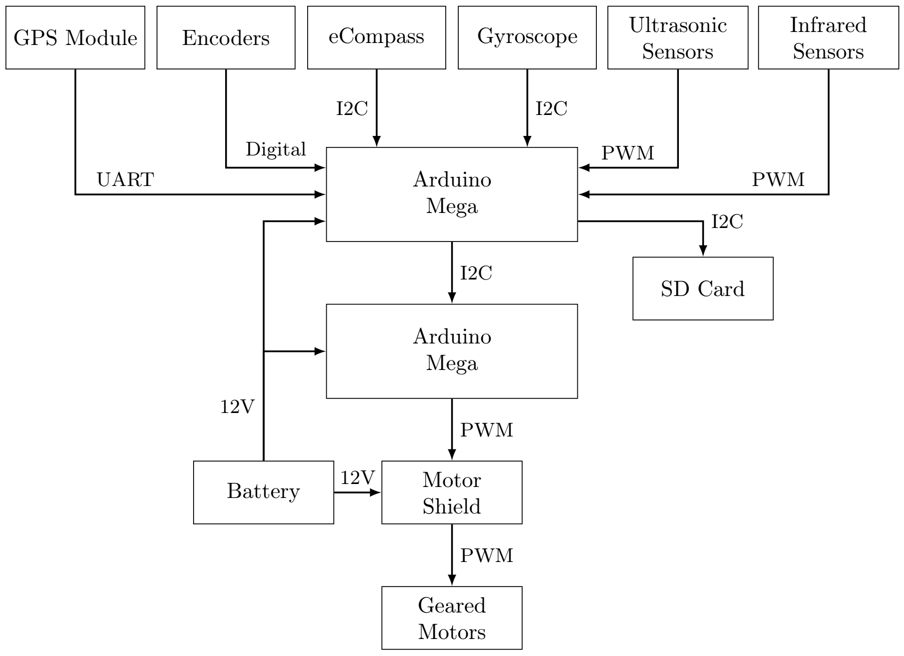


Figure 1.1 – Physical architecture of the rover

1.1 Sensors

1.1.1 GPS Module

Presentation

The GPS module provided is a NEO-6M GPS module. It is not very accurate (the accuracy of the horizontal position measurement is about 15 meters) but is enough for a feasibility study.

The module receipts – with its antenna – signals from the GPS constellation and transmits data messages via its serial pins. These messages are sentences that follow the NMEA communication standard, which is a well-known standard in the field of marine navigation. As the robot will only use the GPS module to know its horizontal position, we will only study the sentence that contain this piece of information: the GPGGA sentence. Here is an example of a GPGGA sentence:

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

Description

123519	The UTC date of the message: 12h 35 19
4807.038,N	The latitude of the receiver: 48°07.038' North
01131.000,E	The longitude of the receiver: 11°31.000' East
1	The "fix quality". 0: no fix, 1: GPS fix (GPS satellites found)
08	Number of satellites found
0.9	"Horizontal dilution of position". It gives information on the horizontal position accuracy
545.4,M	Altitude above sea level of the module, in meters.
46.9,M	The geoidal separation, which corresponds to the difference if height between the geoid surface and the ellipsoidal surface.
*47	The checksum, a hexadecimal number which corresponds to the XOR of every characters between $ and *. The checksum provide a way to check if the frame contains errors.

The blank before the checksum is filled with the time in seconds since the last DGPS and the DGPS station ID number when the fix quality is equal to 2, i.e. DGPS fix.

Code

The Arduino library used to work with the GPS module is TinyGPS++. This library makes the frame parsing transparent: It directly gives the latitude and the longitude of the module from the NMEA data flow.

1.1.2 Compass

Presentation

The robot's heading is a crucial piece of information for its navigation. The heading, coupled with the GPS module's data gives the course error that the robot has to correct in order to reach the target in straight line. Coupled in addition with the ultrasonic sensors' data, the robot can permanently localize obstacles.

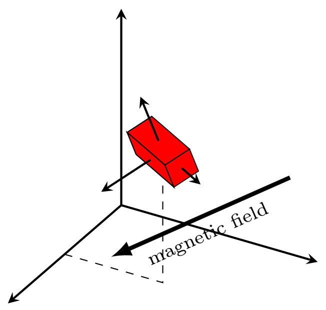


Figure 1.2 – Tilted sensor in the magnetic field

To get its heading, the robot has a eCompass, which is a tilt-compensated electronic compass. It is composed of a magnetometer and an accelerometer. The first one measures the magnitude of the local magnetic field, while the second one measures the magnitude of the resulting acceleration on him along three axes. To compute the heading, the magnetometer gives the direction of the Earth's magnetic field in its own frame of reference R_1. This is enough to compute the heading only if the sensor is level, i.e. if its XY plane is parallel to the ground. To compute a tilt-compensated heading, the accelerometers measures the direction of the gravitational field in R_1. As it must be straight down in the Earth's frame of reference R_0, the accelerometer gives access to the tilt, which the sensor can now compensate.

Magnetometer Calibration

During the first test of the magnetometer, the returned heading was not evenly spaced in 360 degrees. It is due to the fact that the magnetometer is subject to distortions that are mainly produced by the hard iron effect and the soft iron effect.

Hard iron distortions are produced by magnetized pieces of iron, which can be found in speakers for example. If this pieces are attached to the frame of the robot, the hard iron effect will beget offsets in the sensors output. This is the case for the ultrasonic modules. Soft iron distortions distort the magnetic field in such a way that it can be modeled by a 3x3 matrix.

Following [6], a model of a disturbed magnetometer is:

$$\begin{bmatrix}\hat{m}_x\\ \hat{m}_y\\ \hat{m}_z\end{bmatrix} = \begin{pmatrix}a_{xx} & a_{xy} & a_{xz}\\ a_{yx} & a_{yy} & a_{yz}\\ a_{zx} & a_{zy} & a_{zz}\end{pmatrix}\begin{bmatrix}m_x\\ m_y\\ m_z\end{bmatrix} + \begin{bmatrix}b_x\\ b_y\\ b_z\end{bmatrix} \Rightarrow \hat{M} = A \cdot M + B \qquad (1.1)$$

Where M is the true data, $\hat{M}$ is the output data, A the soft iron distortion and B the hard iron distortion.

The computation of A and M is done by a piece of software called *MagCal* fed with raw data from the three axes of the magnetometer. To obtain this data sample, an Arduino sketch sends the formatted outputs of the magnetometer on the serial link, while a Python script collects them via serial communication.

The raw data sample must be acquired when rotating the magnetometer so that the different orientations cover a whole sphere.

According to the model, if the set $\left\{[m_x, m_y, m_z]_n^T, n \in \mathbb{N}\right\}$ covers a sphere, then the disturbed set $\left\{[\hat{m}_x, \hat{m}_y, \hat{m}_z]_n^T, n \in \mathbb{N}\right\}$ must cover an ellipsoid.

The piece of software compute the twelve calibration values whose ellipsoid best fits the disturbed set.

Finally, $A^{-1}[\hat{M} - B]$ gives the undisturbed magnetometer outputs.

After this calibration, the returned heading was now equally spaced in 360 degrees.

Accelerometer Calibration

A one-axis accelerometer can be described as follows:

$$\hat{a} = Gain \cdot a + Offset \tag{1.2}$$

With a the real acceleration and $\hat{a}$ the sensor's output.

The calibration is about finding $Gain$ and $Offset$ for each axis. To achieve that, we can measure the outputs when the sensor is immobile and with one axis toward the centre of the Earth. Indeed, the gravity is the only acceleration the sensor is subjected to, so the real output along this axis is equal to $1g$:

$$\begin{cases} \hat{a}_{1g} = Gain \cdot a_{1g} + Offset = Gain \cdot 1g + Offset \\ \hat{a}_{-1g} = Gain \cdot a_{-1g} + Offset = Gain \cdot (-1g) + Offset \end{cases} \tag{1.3}$$

Therefore,

$$\begin{cases} Gain = \frac{\hat{a}_{1g} - \hat{a}_{-1g}}{2g} \\ Offset = \frac{\hat{a}_{1g} + \hat{a}_{-1g}}{2} \end{cases} \tag{1.4}$$

Finally, $a = \frac{\hat{a} - Offset}{Gain}$ gives the real acceleration magnitude along each axis.

1.1.3 Gyroscope

A gyroscope gives the rover's yaw rate with high precision and low noise. On one hand, an integration of this rate gives a robust relative heading for the rover on a short term, but not on a long term the integration error makes this heading drifts. On the other hand, the compass is noisy and sensible to magnetic disturbances. Both sensors has different defects, therefore a fusion can give a robust absolute heading. This will be discussed later in this paper.

Calibration

The calibration of the gyroscope is similar to the accelerometer one.

$$\hat{\omega} = \omega + Offset \tag{1.5}$$

$Offset$ corresponds to the gyroscope's output value when it is level and still.

1.1.4 Encoders

Using encoders is the best way to get the rover's velocity. Those used on the rover are cheap ones and their resolutions are 40 ticks per revolution. This low number makes the velocity difficult to compute with great accuracy.

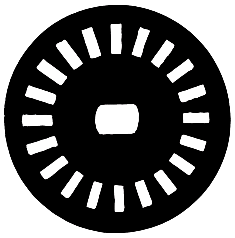


Figure 1.3
encoder wheel

Indeed, with a sampling period of 50 ms the computed velocity is accurate to the nearest 5 cm/s, which is an unsatisfactory accuracy. The latter is equal to 0.5 cm/s for a sampling period of 500 ms, but this computed velocity is an average of the true velocity on the last 500 ms. This means that the encoders give measurements on the true velocity with a precision of half a centimeter per second when the latter one is constant, but give bad measurements when the rover is accelerating. It must be noted in advance that despite this problem, a simple PID controller is still able to make the wheels reach the desired velocity in a couple of seconds.

1.1.5 Ultrasonic Range Finders

The rover uses three range finders to detect obstacles in order to avoid them.

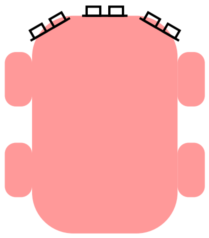


Figure 1.4 – Sensors positioning

Figure 1.5 – HC-SR04 Range Finder

1.2 Data Processing

1.2.1 Switching to a Local Map

The rover must reach a target whose GPS position is known. It is interesting to work not with latitudes and longitudes, but in an X-Y axis linked to a local map (arbitrary centred on the first position of the robot).

In the Cartesian coordinate system $R_0(\boldsymbol{O}, x, y, z)$ the position of $\boldsymbol{m}$ is desbrided as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \rho \cos l_y \cos l_x \\ \rho \cos l_y \sin l_x \\ \rho \sin l_y \end{bmatrix} \tag{1.6}$$

By differentiating this relation, we get:

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \underbrace{\begin{pmatrix} -\rho \cos l_y \sin l_x & -\rho \sin l_y \cos l_x & \cos l_y \cos l_x \\ \rho \cos l_y \cos l_x & -\rho \sin l_y \sin l_x & \cos l_y \sin l_x \\ 0 & \rho \cos l_y & \sin l_y \end{pmatrix}}_{J} \begin{bmatrix} dl_x \\ dl_y \\ d\rho \end{bmatrix} \tag{1.7}$$

The matrix J can be interpreted as a transformation matrix between R_0 and $R_1(\boldsymbol{m}, \overrightarrow{east_{\boldsymbol{m}}}, \overrightarrow{north_{\boldsymbol{m}}}, \overrightarrow{altitude_{\boldsymbol{m}}})$, the coordinate system centred on $\boldsymbol{m}$, and whose axes correspond to East, North and sky respectively. As each column of J represents the coordinate of the vector of the new base expressed in the former base, we can even rewrite J as follows:

$$J = \begin{pmatrix} -\rho \cos l_y \sin l_x & -\rho \sin l_y \cos l_x & \cos l_y \cos l_x \\ \rho \cos l_y \cos l_x & -\rho \sin l_y \sin l_x & \cos l_y \sin l_x \\ 0 & \rho \cos l_y & \sin l_y \end{pmatrix}_{R_0} = \begin{pmatrix} \overrightarrow{east_{\boldsymbol{m}}} & \overrightarrow{north_{\boldsymbol{m}}} & \overrightarrow{altitude_{\boldsymbol{m}}} \end{pmatrix}_{R_0} \tag{1.8}$$

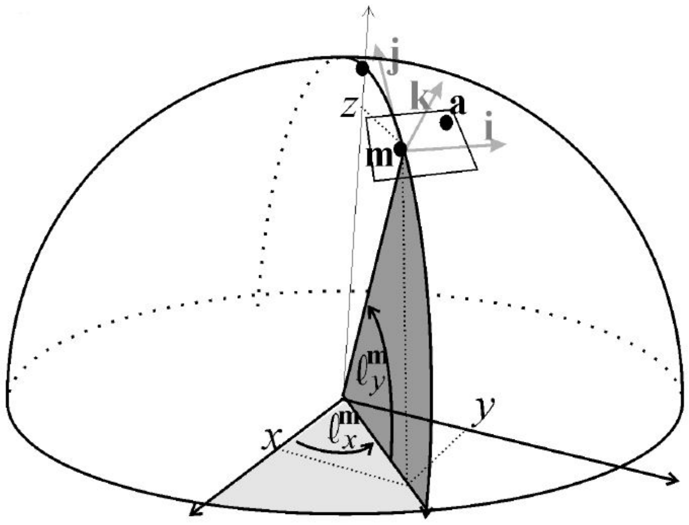


Figure 1.6 – local map centred on point the robot $\boldsymbol{m}$. $\boldsymbol{a}$ is the target to reach. Illustration from [2]

By normalizing each column of J, we obtain R, the rotation matrix between R_0 and R_1:

$$R = \begin{pmatrix} -\sin l_x & -\sin l_y \cos l_x & \cos l_y \cos l_x \\ \cos l_x & -\sin l_y \sin l_x & \cos l_y \sin l_x \\ 0 & \cos l_y & \sin l_y \end{pmatrix} \tag{1.9}$$

We will use this rotation matrix to obtain the coordinate of $\boldsymbol{m}$ in the local map centred on its initial position $\boldsymbol{m_0}$. Indeed:

$$\begin{aligned} \overrightarrow{\boldsymbol{Om}}_{R_1} &= R^T \cdot \overrightarrow{\boldsymbol{Om}}_{R_0} \\ \text{with } R^T &= \begin{pmatrix} -\sin l_{x_0} & \cos l_{x_0} & 0 \\ -\sin l_{y_0} \cos l_{x_0} & -\sin l_{y_0} \sin l_{x_0} & \cos l_{y_0} \\ \cos l_{y_0} \cos l_{x_0} & \cos l_{y_0} \sin l_{x_0} & \sin l_{y_0} \end{pmatrix} \end{aligned} \tag{1.10}$$

As $\boldsymbol{m}$ and $\boldsymbol{m_0}$ are close enough, we can simplify this expression using a first-order approximation:

$$\overrightarrow{\boldsymbol{Om}}_{R_1} \backsimeq \begin{bmatrix} \rho \cos l_y \cdot (l_x - l_{x_0}) \\ \rho (l_y - l_{y_0}) \\ \rho \end{bmatrix} \tag{1.11}$$

Therefore,

$$\overrightarrow{\boldsymbol{m_0 m}}_{R_1} \backsimeq \begin{bmatrix} \rho \cos l_y \cdot (l_x - l_{x_0}) \\ \rho (l_y - l_{y_0}) \\ \rho - \rho_0 \end{bmatrix} \tag{1.12}$$

It is possible to find the heading toward a target $\boldsymbol{a}$:

$$heading = \text{angle}(\overrightarrow{\boldsymbol{m_0 a}}_{R_1} - \overrightarrow{\boldsymbol{m_0 m}}_{R_1}) \tag{1.13}$$

1.2.2 Fill the Local Map with Obstacles

As noted previously, the obstacles are detected using ultrasonic range finders. To avoid them, the choice that has been made is to fill the local map with the different obstacles seen. This technique implies an estimation of the rover's position and heading.

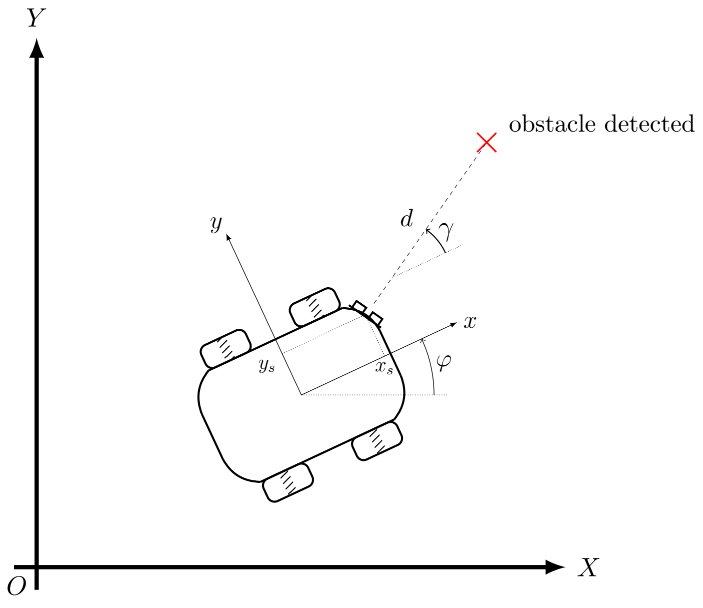


Figure 1.7 – Obstacle position in the local map

$$\begin{bmatrix} x_{obs} \\ y_{obs} \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} + R_\varphi \left(\begin{bmatrix} x_s \\ y_s \end{bmatrix} + d \begin{bmatrix} \cos\gamma \\ \sin\gamma \end{bmatrix} \right) \tag{1.14}$$

Chapter 2

Modelisation

Introduction

In order to design different systems such as localization, navigation or control for the rover, it is best to build first a model for the rover. It should be noted that the model must be very accurate in order to fit perfectly with the real system. In this condition, any designed system successfully integrated to the simulated rover can itself be successfully integrated to the real one. Therefore, reliable kinematic a dynamic models of the rover must be used.

As the real environment is mainly indoor and level, the model will only be two-dimensional.

2.1 Global Overview

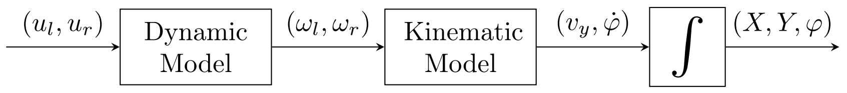


Figure 2.1 – simulated system

Notation

L	rover length	ICR	rover's instantaneous center of rotation
B	rover width	R	radius of curvature
b	wheel width	α	yaw rate correction parameter
r	wheel radius	l, c	patch related distances
X–Y	global frame	m	rover's mass
x–y	frame centred on the rover	M	mass matrix
x_l–y_l	frame centred on the rover's left wheels	C	resistance term
x_r–y_r	frame centred on the rover's right wheels	τ	traction term
φ	rover's heading in the frame of reference	I	rover's moment of inertia
v_x, v_y	rover speed components on its local frame	F	wheel/floor interaction related force
ω_l, ω_r	$\dot{q}$: wheels angular velocities	f	internal resistance force
u_l, ur	motor input $\in [0, 255]$	j	shear deformation
$\bar{\cdot}$	desired physical parameter	p	normal pressure on the contact patch
		μ	coefficient of friction
		K	shear deformation modulus

2.2 Kinematic Model

2.2.1 Vehicle geometry

The rover the project is based on is a four-wheeled skid steering vehicle. This type of vehicle turns by rotating the wheels on both sides at different speeds.

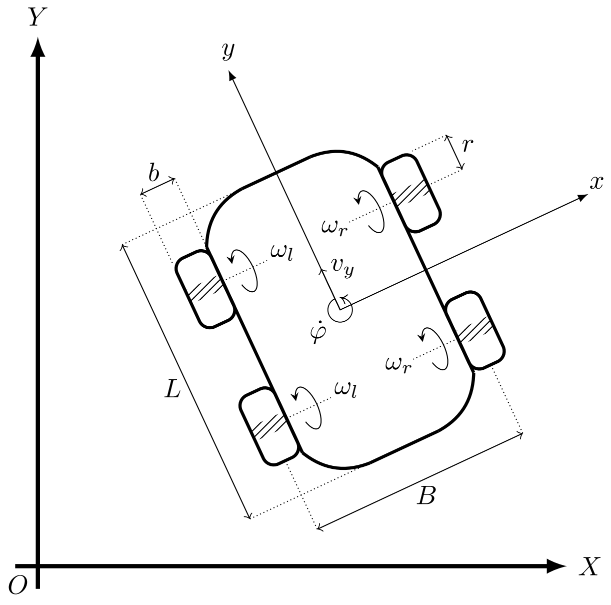


Figure 2.2 – Skid-steering vehicle

Figure 2.2 displays the geometry of the rover this report refers to.

2.2.2 Kinematic Equations

From [4], we know that the kinematic law of such a vehicle is given by:

$$\begin{bmatrix} v_x \\ v_y \\ \dot{\varphi} \end{bmatrix} = r \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{\alpha B} & \frac{1}{\alpha B} \end{pmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{2.1}$$

Where α is a terrain-dependent correction parameter [4, 9]. This parameter will be adjusted in order to fit with the experimental results. It should be noted that the lateral velocity v_x is zero. This is true only if one assumes that a skid-steering vehicle is equivalent to a differential drive vehicle [1][4, 7].

1. A differential drive vehicle is similar to a skid-steering vehicle, but with two wheels.

From these equations, we see that the longitudinal velocity is the average of the left and right wheels' velocities.

2.3 Dynamic Model

The kinematic model take the rotational velocities ω_l and ω_r as inputs. These rotational velocities are computed by the dynamic model. Indeed, ω_l and ω_r depend on the torque provided by the motors, the friction caused by the interaction between the wheels and the ground, but also internal frictions and the inertia of the rover.

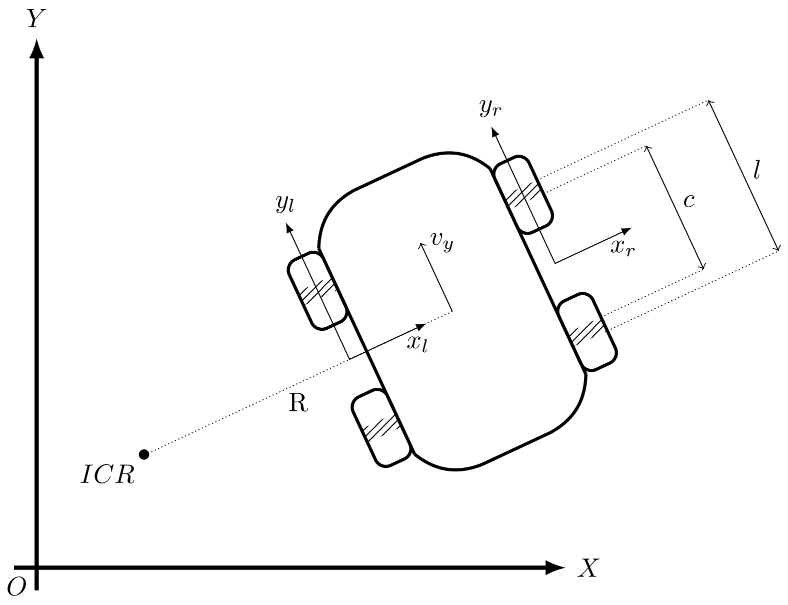


Figure 2.3 – Rover turning left

Let $\dot{q} = (\omega_l, \omega_r)^T$. Following [9], the dynamic model that rules this system is given by:

$$M\ddot{q} + C(\dot{q}) = \tau \tag{2.2}$$

Where M is the mass matrix, C is the resistance term, and $\tau = (\tau_l, \tau_r)^T$ is the torque of the left and right motors. This model is expressed in the local frame x-y. As stated above, this matrix equation brings together a traction term (τ), a inertia term ($M\ddot{q}$) and a resistance term ($C(\dot{q})$).

2.3.1 Inertia Term

Following [9], the mass matrix expressed in the local frame x-y is given by:

$$M = \begin{pmatrix} \dfrac{mr^2}{4} + \dfrac{r^2 I}{\alpha B^2} & \dfrac{mr^2}{4} - \dfrac{r^2 I}{\alpha B^2} \\ \dfrac{mr^2}{4} - \dfrac{r^2 I}{\alpha B^2} & \dfrac{mr^2}{4} + \dfrac{r^2 I}{\alpha B^2} \end{pmatrix} \tag{2.3}$$

Where m is the mass of the rover, B its width, r the radius of the wheels, and I the moment of inertia of the rover.

By approximating the rover as a rectangle with a equally distributed mass, I can be calculated using:

$$I = \frac{m}{12}(L^2 + (B-b)^2) \tag{2.4}$$

2.3.2 Resistance Term

Theory

As noted previously, the resistance term C encapsulates the effort created by the interaction between the wheels and the floor, and internal resistances

Let dF be the force caused by the wheel-floor interaction on an arbitrary point (x, y) on the contact patch. This force can be expressed as follows [9, 8]:

$$dF = \tau_{ss}(x,y)\ dS = \tau_{ss}(x,y)\ dxdy \tag{2.5}$$

Where τ_{ss} is the shear stress of the tread. τ_{ss} is function of the shear displacement:

$$\tau_{ss} = p\mu(1 - e^{-j/K}) \tag{2.6}$$

Where p is the normal pressure, that is $\frac{mg/4}{S_{patch}} = \frac{mg/4}{b(l-c)/2}$, μ is the coefficient of friction, K is the shear deformation modulus, and j is the shear deformation.

An excellent development of the shear deformation and of the angle of the sliding velocity had been made by Yu, following Wong's work. The formulas expressed in the global frame X-Y, given point (x_l, y_l) (resp. (x_r, y_r)) of the local frame $\mathrm{x_l}$-$\mathrm{y_l}$ (resp. $\mathrm{x_r}$-$\mathrm{y_r}$), and given a left turn, are:

$$j_{fr,X} = \left(R + \frac{B}{2} + x_r\right)\left[\cos\left(\frac{(l/2 - y_r)\dot{\varphi}}{r\omega_r}\right) - 1\right] - y_r \sin\left(\frac{(l/2 - y_r)\dot{\varphi}}{r\omega_r}\right) \tag{2.7a}$$

$$j_{fr,Y} = \left(R + \frac{B}{2} + x_r\right)\sin\left(\frac{(l/2 - y_r)\dot{\varphi}}{r\omega_r}\right) - \frac{l}{2} + y_r \cos\left(\frac{(l/2 - y_r)\dot{\varphi}}{r\omega_r}\right) \tag{2.7b}$$

$$j_{rr,X} = \left(R + \frac{B}{2} + x_r\right)\left[\cos\left(\frac{(-c/2 - y_r)\dot{\varphi}}{r\omega_r}\right) - 1\right] - y_r \sin\left(\frac{(-c/2 - y_r)\dot{\varphi}}{r\omega_r}\right) \tag{2.7c}$$

$$j_{rr,Y} = \left(R + \frac{B}{2} + x_r\right)\sin\left(\frac{(-c/2 - y_r)\dot{\varphi}}{r\omega_r}\right) + \frac{c}{2} + y_r \cos\left(\frac{(-c/2 - y_r)\dot{\varphi}}{r\omega_r}\right) \tag{2.7d}$$

$$j_{fl,X} = \left(R - \frac{B}{2} + x_l\right)\left[\cos\left(\frac{(l/2 - y_l)\dot{\varphi}}{r\omega_r}\right) - 1\right] - y_l \sin\left(\frac{(l/2 - y_l)\dot{\varphi}}{r\omega_r}\right) \tag{2.7e}$$

$$j_{fl,Y} = \left(R - \frac{B}{2} + x_l\right)\sin\left(\frac{(l/2 - y_l)\dot{\varphi}}{r\omega_r}\right) - \frac{l}{2} + y_l \cos\left(\frac{(l/2 - y_l)\dot{\varphi}}{r\omega_r}\right) \tag{2.7f}$$

$$j_{rl,X} = \left(R - \frac{B}{2} + x_l\right)\left[\cos\left(\frac{(-c/2 - y_l)\dot{\varphi}}{r\omega_r}\right) - 1\right] - y_l \sin\left(\frac{(-c/2 - y_l)\dot{\varphi}}{r\omega_r}\right) \tag{2.7g}$$

$$j_{rl,Y} = \left(R - \frac{B}{2} + x_l\right)\sin\left(\frac{(-c/2 - y_l)\dot{\varphi}}{r\omega_r}\right) + \frac{c}{2} + y_l \cos\left(\frac{(-c/2 - y_l)\dot{\varphi}}{r\omega_r}\right) \tag{2.7h}$$

and

$$\gamma_{fr} = \gamma_{rr} = \arctan\left(\frac{(R + B/2 + x_r)\dot{\varphi} - r\omega_r}{-y_r\dot{\varphi}}\right) \tag{2.8a}$$

$$\gamma_{fl} = \gamma_{rl} = \arctan\left(\frac{(R - B/2 + x_l)\dot{\varphi} - r\omega_l}{-y_l\dot{\varphi}}\right) \tag{2.8b}$$

where fr, rr, fl and rl refer to front-right, rear-right, front-left and rear-left respectively.

Therefore,

$$j = \sqrt{j_X^2 + j_Y^2} \tag{2.9}$$

Finally, the forces caused by the interaction between the wheels and the ground on each side is given by:

$$\begin{aligned} F_l = &\int_{c/2}^{l/2}\int_{-b/2}^{b/2} p\mu(1 - e^{-j_{fl}/K})\ \sin(\pi + \gamma_{fl})dxdy \\ &+ \int_{-l/2}^{-c/2}\int_{-b/2}^{b/2} p\mu(1 - e^{-j_{rl}/K})\ \sin(\pi + \gamma_{rl})dxdy \end{aligned} \tag{2.10a}$$

$$\begin{aligned} F_r = &\int_{c/2}^{l/2}\int_{-b/2}^{b/2} p\mu(1 - e^{-j_{fr}/K})\ \sin(\pi + \gamma_{fr})dxdy \\ &+ \int_{-l/2}^{-c/2}\int_{-b/2}^{b/2} p\mu(1 - e^{-j_{rr}/K})\ \sin(\pi + \gamma_{rr})dxdy \end{aligned} \tag{2.10b}$$

where the term $\sin(\pi + \gamma)$ implies that the force is in the opposite direction of the sliding velocity, and that only the longitudinal component is kept.

These forces must be added to the internal resistance force f to generate the final resistance torque term:

$$\tau = \begin{bmatrix} r(F_l + f) \\ r(F_r + f) \end{bmatrix} \tag{2.11}$$

Symetry

The previous section implies $\dot{\varphi} \geq 0$, i.e. a left turn. Symetry is used to compute F when the rover is turning to the right.

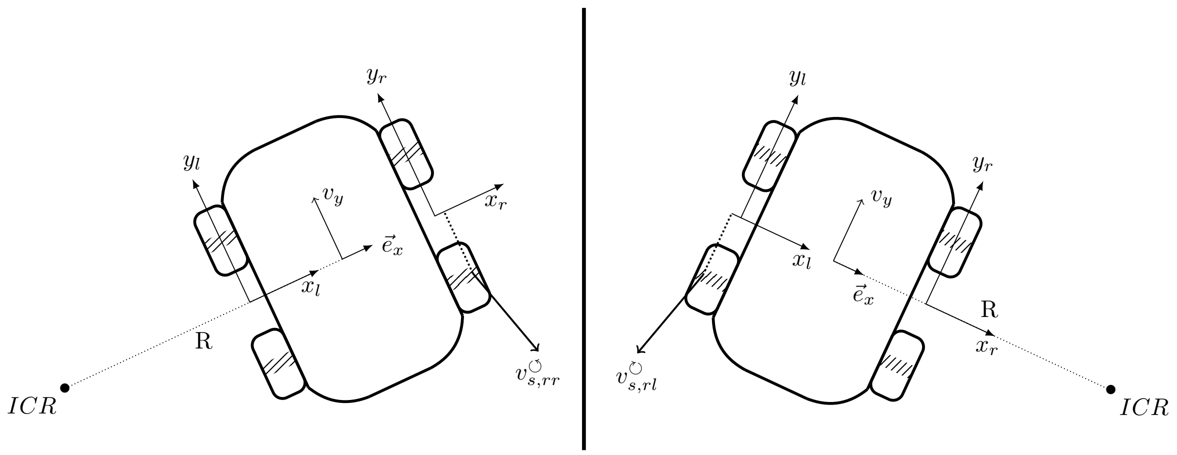


Figure 2.4 – Symetry principle

Figure 2.4 displays a relation between sliding velocities when turning left and right, noted $v_s^{\circlearrowleft}$ and $v_s^{\circlearrowright}$ respectively. In the example of the figure:

$$v_{s,rr}^{\circlearrowleft}(x, y) \cdot \vec{e}_x = -v_{s,rl}^{\circlearrowright}(-x, y) \cdot \vec{e}_x \tag{2.12}$$

Generally,

$$v_{s,r}^{\circlearrowleft}(x, y) \cdot \vec{e}_x = -v_{s,l}^{\circlearrowright}(-x, y) \cdot \vec{e}_x \tag{2.13}$$

This relation links $F_l^{\circlearrowright}$ with $F_l^{\circlearrowleft}$ and$F_r^{\circlearrowright}$ with $F_r^{\circlearrowleft}$, which are known from the **Theory** section.

Approximation

Due to the double integration, the time to compute each force F is significant. Moreover, F depends on the instantaneous radius of curvature R, which can reach very high values when the rover is moving forward. Indeed, $R = v_y/\dot{\varphi}$ with $\dot{\varphi} \simeq 0$. This makes the integration really hard to compute, and even make the integration fail.

In order to accelerate the simulation significantly, a good idea is to find a good approximation for this force. As $F = F(v_y, \dot{\varphi})$, the latter can be approximated by a 2D-regression using a least-square method.

The first step is to generate the data set, i.e. $\{(x_n, y_n, z_n) \mid z_n = F(x_n, y_n)\}$. Then an equation for the 2D surface must be chosen. After different attempts, the following equation has been accepted:

$$\begin{aligned} z = {} & c_1\ x^3 + c_2\ y^3 + c_3\ x^2 y + c_4\ xy^2 + c_5\ x^2 \\ & + c_6\ y^2 + c_7\ xy + c_8\ x + c_9\ y + c_{10} \end{aligned} \tag{2.14}$$

Finally, a least-square method is run to determine every parameter c_i.

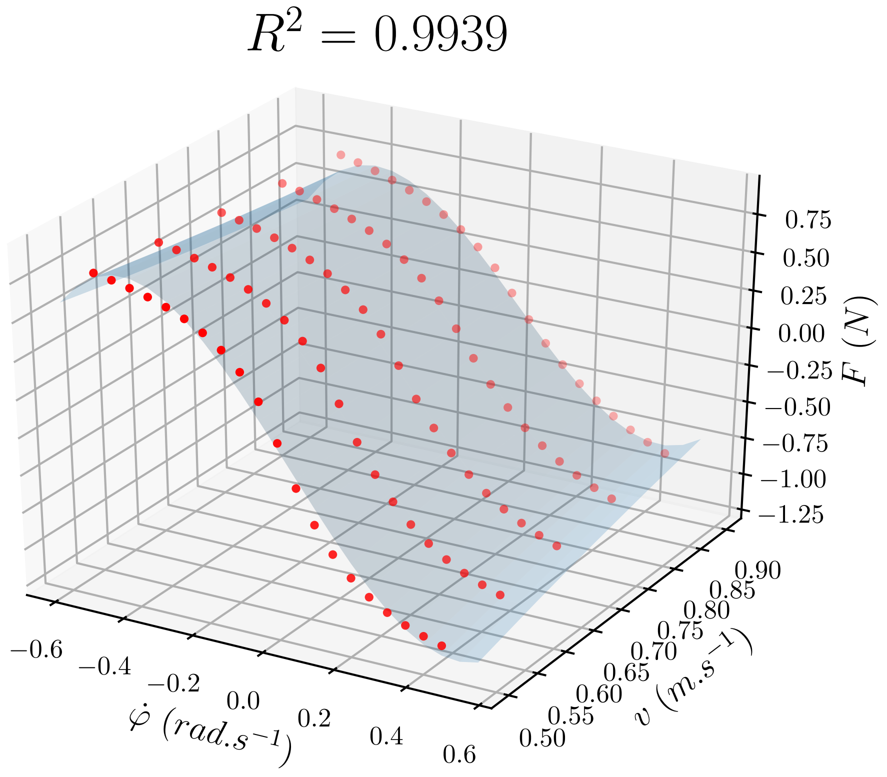


Figure 2.5

2.3.3 Traction Term

The equation that models the motor is the following [5]:

$$\frac{\tau_m}{\tau_s} + \frac{\omega_m}{\omega_n} = 1 \tag{2.15}$$

with τ_m is the motor torque, ω_m the motor speed, τ_s the stall-torque and ω_n the no-load speed. These two latter are given for a given voltage, following [5]:

$$\tau_s = \frac{V_m K_t}{R_a} \tag{2.16}$$

$$\omega_n = \frac{V_m}{K_B} \tag{2.17}$$

These equations shows that τ_s and ω_n are proportional to the voltage V_m. As τ_s and ω_n are given in the motor data sheet for a given voltage, the constants K_t/R_a and $1/K_B$ can be easily deduced.

Finally, we obtain from 2.15, 2.16 and 2.17 the equation for the motor torque:

$$\tau_m = \frac{V_m K_t}{R_a}\left(1 - \frac{V_m}{K_B}\omega_m\right) \tag{2.18}$$

As the rover has two wheels on each side, the traction term τ is:

$$\tau = \begin{bmatrix}\tau_l \\ \tau_r\end{bmatrix} = \frac{2V_m K_t}{R_a}\left(\begin{bmatrix}1 \\ 1\end{bmatrix} - \frac{V_m}{K_B}\begin{bmatrix}\omega_l \\ \omega_r\end{bmatrix}\right) \tag{2.19}$$

2.3.4 Results

To confront the model with the real rover, a good measurement of the rover velocity is mandatory. As the encoders give a bad estimation of the rover when the latter is accelerating – coupled with a low measurement rate – another method of acquisition must be used.

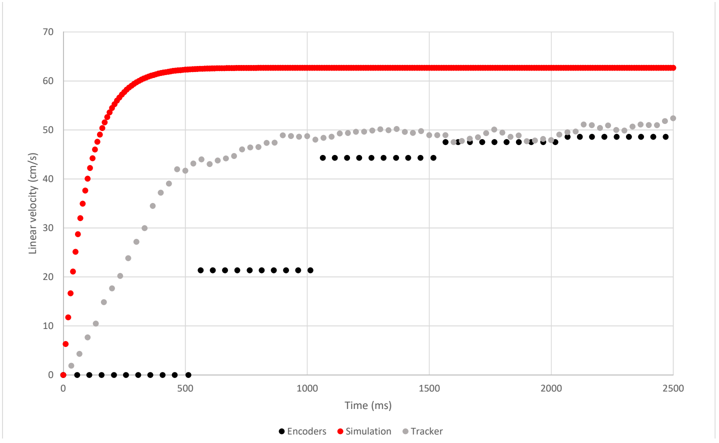


Figure 2.6 – model confronted to experiment for a constant PWM input equal to 80.

A physics oriented video processing piece of software named *Tracker* has been used to get a reliable estimation of the rover's velocity. It can track a target on a video to give a X-Y position in a given frame of reference. The velocity is the derivative of this position. It must be noted that the speed computed from the encoders is given because encoders measurements are very accurate at constant speed [2].

The graph 2.6 refers to a situation where a constant PWM input equal to 80 is sent to the motor controller on both sides of the rover. The idea is to confront the behaviours of the simulated rover with the real one. The difference of speed is striking. Despite the many weeks of tuning, the model has never matched with the real system in all situations (different inputs in the motor controller).

Even if the model does not fit with the real system, it is still a good model for a typical skid-steering rover. Moreover, it is stable and does not differ much with the real rover. Finally, it was too late to abandon this model and restart from scratch. Therefore, the navigation system elaborated in the next chapter has been firstly developed and tested on simulation using this flawed model.

2. At constant speed, the accuracy of the encoders is about 5 mm/s, see 1.1.4.

Chapter 3

Make the Rover Autonomous

3.1 Choice of State Observer

In order to apply a guidance algorithm to make the rover autonomous, it must be provided with an estimation of its state. Different state observers have been tested.

3.1.1 EKF with a Dynamic Model

The first state observer that have been implemented in the simulation was an extended Kalman filter (EKF) using the dynamic model highlighted in section 2.3. As said previously, the dynamic model does not fit with the real rover. Therefore, such a state observer cannot be implemented in the real rover. But it is not without point to study it with the simulation, as the latter uses the same model and therefore fits perfectly. Thus the performance of this state observer can be studied after all.

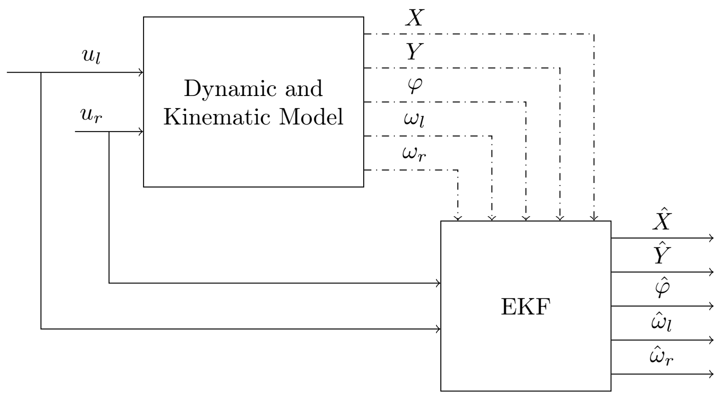


Figure 3.1 – EKF using the dynamic model

Unfortunately, this observer is sensible to the rover's main issue which is the encoders' sampling periods. As explained in subsection 1.1.4, in order to have a good accuracy on the wheels' speeds, the sampling period is set to 500 milliseconds. This begets the dynamic model to diverge.

A solution is to run the filter with a high rate, and use measurements of the speeds only every 500 ms in the correction step. But as this observer is only theoretical as it cannot be implemented on the real rover, this work has not been done because it was not a priority.

3.1.2 EKF with Kinematic

The first state observer that could be implemented on the rover is a extended Kalman filter that uses a simple kinematic model (see figure 3.2).

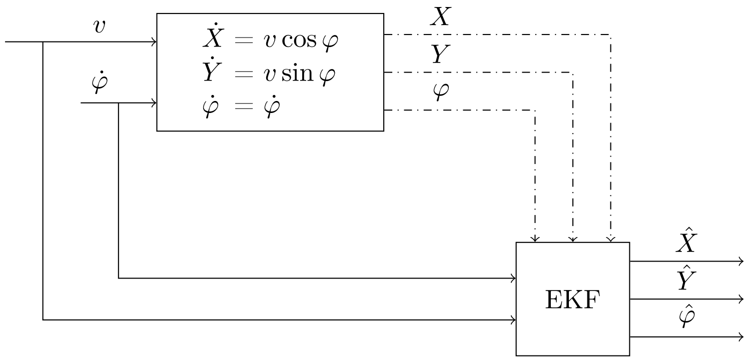


Figure 3.2 – EKF using the kinematic model

The inputs v and $\dot{\varphi}$ for the prediction step are given by the encoders and the gyroscope respectively. The measurements X, Y and φ for the correction step are given by the GPS module and the compass respectively.

3.1.3 Double Kalman Filter

It can be noted that the state observer of the last section uses a non-linear model due to trigonometric functions. However, the use of a extended Kalman filter is not mandatory. Indeed, two Kalman filters in series can remove this non-linearity. The use of this double Kalman filter is displayed in figure 3.3.

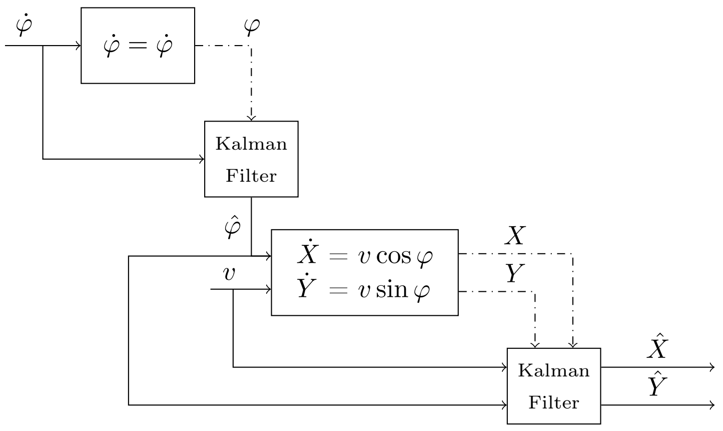


Figure 3.3 – double Kalman filter using the kinematic model

3.1.4 Back-Mean-Forth

Another state observer that has been tested is a pure creation called "back-mean-forth" (BMF). The idea is to combine dead reckoning with GPS data samples over time.

When the rover receives a GPS measurement $\vec{g}_n$ for its current position $\vec{p}_n$, the BMF subtracts it by the estimated displacement of the rover $\vec{d_n}$. This estimation is given by the dead reckoning. If the noise of the GPS position is centred, this operation gives a pseudo measurement of the initial position of the rover. Taking the average of all these pseudo measurements gives a better estimation $\hat{p}_0$ for the initial position of the rover. Finally, add it to the displacement $\vec{d_n}$ gives a better estimation $\hat{p}_n$ for the current position $\vec{p}_n$ of the rover.

This principle is summed up in figure 3.4. It also displays that the dead reckoning path estimation drifts over time. Therefore, the BMF method cannot be used for a long period of time, as any system using dead reckoning.

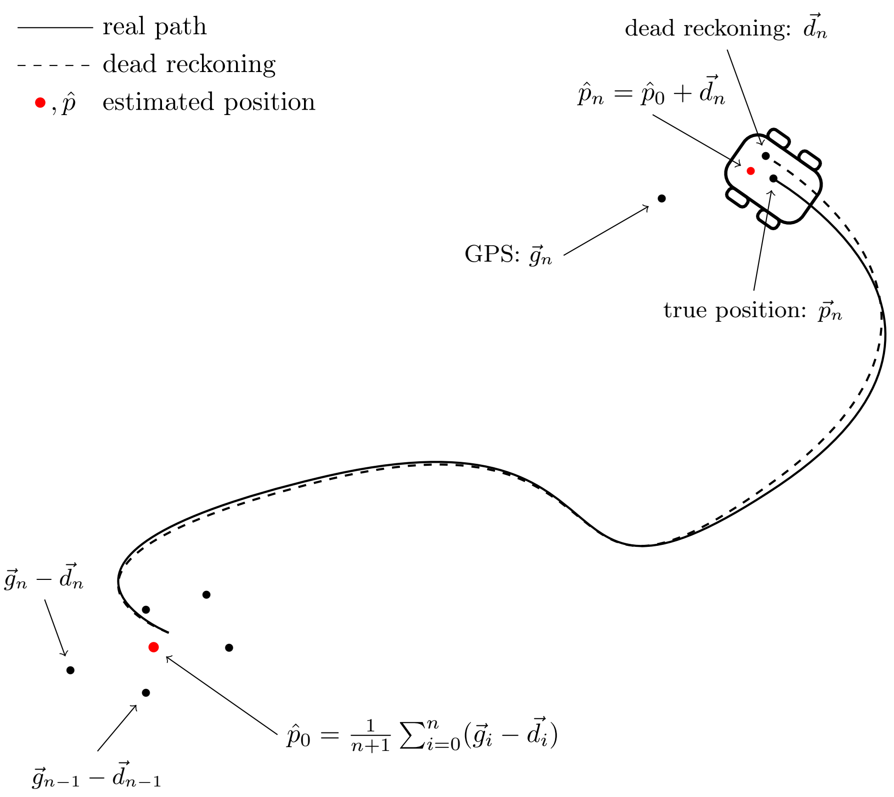

real path
dead reckoning
●, $\hat{p}$ estimated position
dead reckoning: $\vec{d}_n$
$\hat{p}_n = \hat{p}_0 + \vec{d}_n$
GPS: $\vec{g}_n$
true position: $\vec{p}_n$
$\vec{g}_n - \vec{d}_n$
$\hat{p}_0 = \frac{1}{n+1}\sum_{i=0}^{n}(\vec{g}_i - \vec{d}_i)$
$\vec{g}_{n-1} - \vec{d}_{n-1}$

Figure 3.4 – BMF principle

It must be noted that this method implies a centred noise for the GPS measurements, independent from the receiver true position. This is true only if there is no building around, i.e. no bounces [3, 1].

3.1.5 Confrontation

To confront the different state observers, the best is to confront them directly and all together on a simulation where the rover moves following predefined paths.

Easy path

Figure 3.5 displays the paths of the rover and of the different state observers. Here, the rover performs a large circular turn from position $(0, 0)$. All the state observers converge on the position of the rover. The double Kalman filter and the EKF estimations are virtually identical.

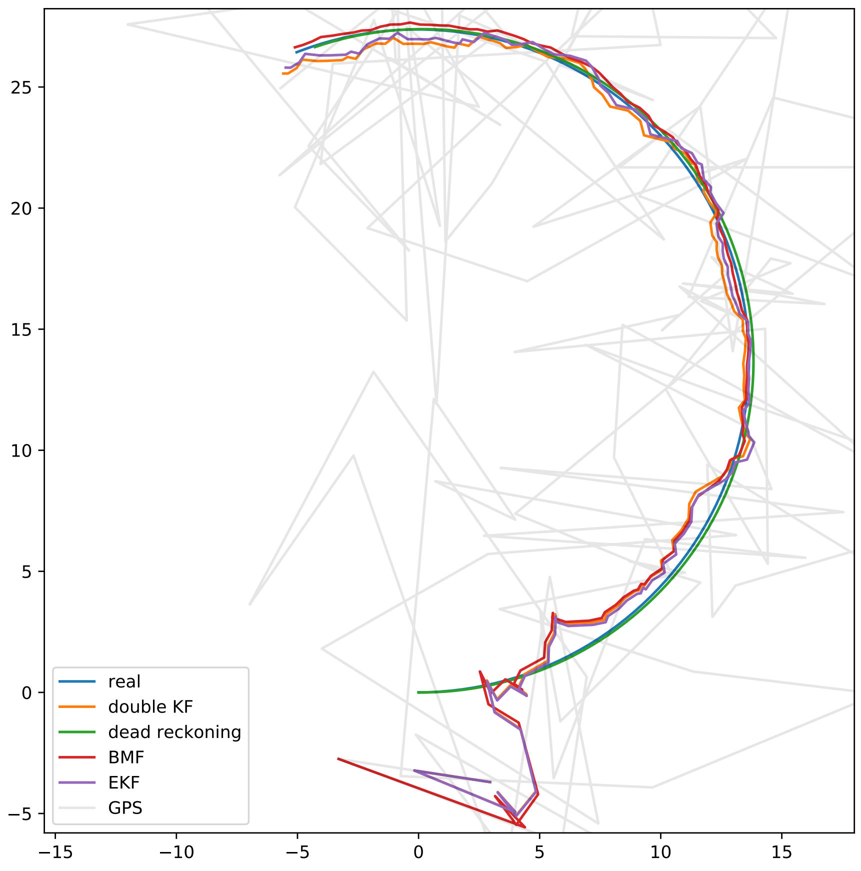


Figure 3.5 – confrontation of the different state observers for a large turn.

On this example, the BMF provides a better estimation than the Kalman filters. The reason is that the BMF relies more on the model than the Kalman filters. Indeed, the BMF uses the model through a dead reckoning, and the Kalman filters through their prediction steps. The difference is that the BMF considers the dead reckoning as the true path of the rover, unlike Kalman filters. But here, the dead reckoning is efficient because the path is simple. Therefore the BMF gives better results.

Complex path

Figure 3.6 displayed a situation where the path of the rover is complex.

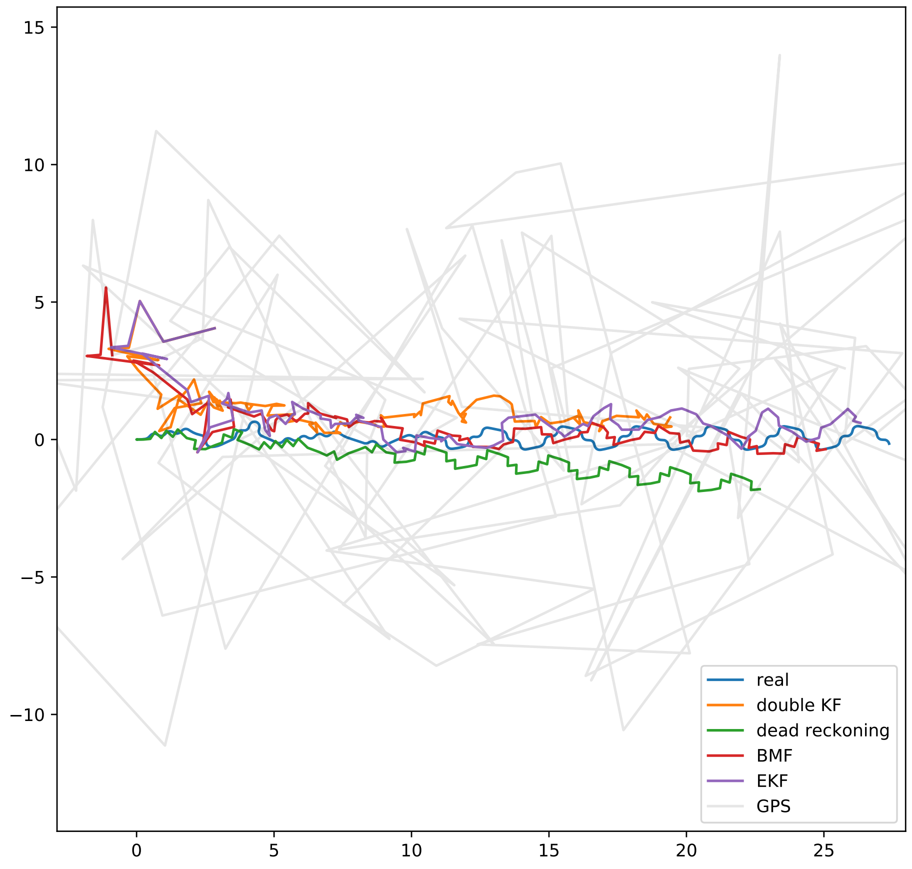


Figure 3.6 – confrontation of the different state observers for a complex path.

Here, the dead reckoning is lost and begets failures in the estimation of the position for every state observer. This is caused by the bad accuracy of the encoders. They cannot sense fast [1] accelerations of the wheels. This is the main problem of this rover.

3.2 Controller

As noted previously, the rover turns when the wheels on each side of the rover turn at different speeds. To make the rover go toward a desired heading $\bar{\varphi}$ with a desired speed $\bar{v}$, the controller must translate $\bar{\varphi}$ and $\bar{v}$ into desired rotary velocities $\bar{\omega_l}$ and $\bar{\omega_r}$. This is the role of the inverse of the kinematic model, seen in equation 2.1. To do that, the inversed kinematic model needs a desired yaw rate, computed by a PID controller given φ and $\bar{\varphi}$. Finally, two PID controllers compute the PWM inputs to the motor controller given ω_l, ω_r, $\bar{\omega_l}$ and $\bar{\omega_r}$.

The inner working of the controller is summed up in figure 3.7. The measurements that feed the various PID controllers are represented with dash-dotted arrows.

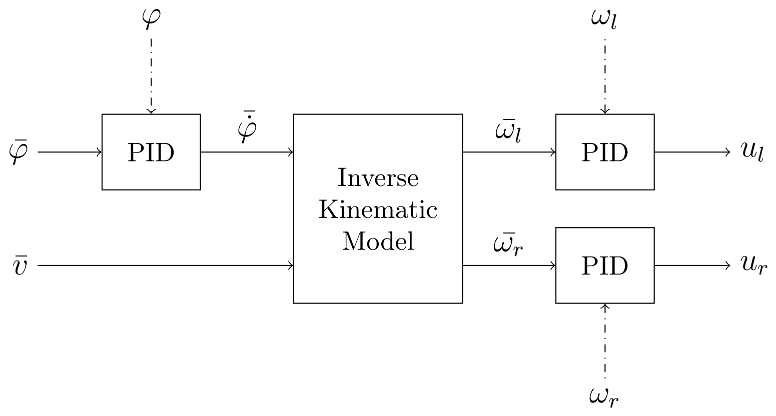


Figure 3.7 – inner working of the controller

1. Fast is relative. As noted in section 1.1.4, the encoders provide the average of the speed on a period of 500 milliseconds. Therefore, they smooths every acceleration of the rover.

3.3 Guidance

To make the rover go to a target while avoiding obstacles, the latter is provided with an artificial potentials based guidance system. The idea is to compute an artificial force applied to the rover that makes it move. In other words, the magnitude of the force is the desired velocity, and the angle is the desired heading. This force is the resultant of every force generated by an obstacle (repulsive force) or a target (attractive force).

3.3.1 Obstacles

Let $\vec{d} = [X_{obs} - X_{rover},\ Y_{obs} - Y_{rover}]^T$ the distance vector between the obstacle and the rover. The artificial repulsive force applied on the rover is given by:

$$\vec{f} = \frac{c}{\|\vec{d}\|^3}\,\vec{d} \tag{3.1}$$

Where c is a tuning constant, used to set a proper "no-go" zone for the rover around the obstacle.

This force is applied each time a range finder detects an obstacle. The position of the obstacle $[X_{obs},\ Y_{obs}]^T$ is computed using the formula 1.14.

3.3.2 Target

Again, let $\vec{d} = [X_{target} - X_{rover},\ Y_{target} - Y_{rover}]^T$ the distance vector between the target and the rover. The artificial attrative force applied on the rover is given by:

$$\vec{f} = \frac{c}{\|\vec{d}\|}\,\vec{d} = c\,\vec{u} \tag{3.2}$$

Where $\vec{u}$ is the unit vector from the rover toward the target. It can be noticed that c corresponds to the velocity that the rover must satisfy when no obstacle is nearby.

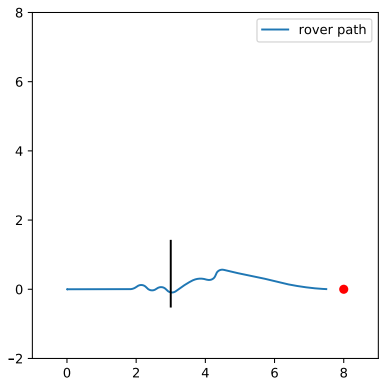
In order to make the rover stops on the target, the expression of the force changes near the obstacle (1m for example):

$$\vec{f} = c\,\vec{d} \tag{3.3}$$

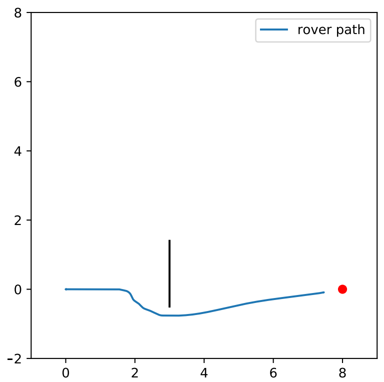
The norm of this force reduces when the rover approaches the target and reaches 0 on it. Another solution is to stop the guidance near the target. But this solution cannot be applied to a situation where the target is moving, because the rover would stop forever when it reaches the target.

3.3.3 Results

Figure 3.8a displays the trajectory of the rover that must reach a target (in red) while avoiding the obstacle (in black). The rover detects the obstacle but doesn't manage to avoid it. This is because the guidance system provides a new consign for the heading and the speed of the rover each 500 milliseconds as it requires an up-to-date estimation of the position of the rover. The latter is given by the state observer which is limited by the measurement rate of the encoders.



(a) measurements each 500 ms.



(b) measurements each 100 ms.

Figure 3.8
trajectory of the rover guided to reach a target by avoiding a wall with different measurement rate.

It is relevant to do the same experiment with the assumption that the encoders can have the same accuracy on the velocity, but with a measurement rate equal to 100 milliseconds. The results are displayed in figure 3.8b. In this experience, the rover does has time to correct its trajectory when approaching an obstacle too closely in order to avoid it.

To conclude, the guidance system works in a situation where the measurement rate is not too low. As noted previously, a period of 500 milliseconds between two measurements is not compatible with a reactive autonomous system.

Chapter 4

Simulation

The simulation has been developed in Python language and is object-oriented. The idea here was to create a simulation that is decomposed as the real system (frame, sensors, but also software blocs such as the controllers or the guidance system) to make it more intelligible. Moreover, the inner workings of the sub-systems of the rover are relatively independent and can be easily encapsulated within classes.

This chapter will be short as it is not about explaining each classes, firstly because a part of the simulation is just transcriptions of formulas seen in chapter 2 Modelisation, but also because it would be redundant with the comments supplied in the code.

The functioning of the simulation is largely summed up through the sequence diagram in figure 4.1 on the next page. It displays the most important objects working together during a step of the simulation. Furthermore, it is a representation of `main.py`.

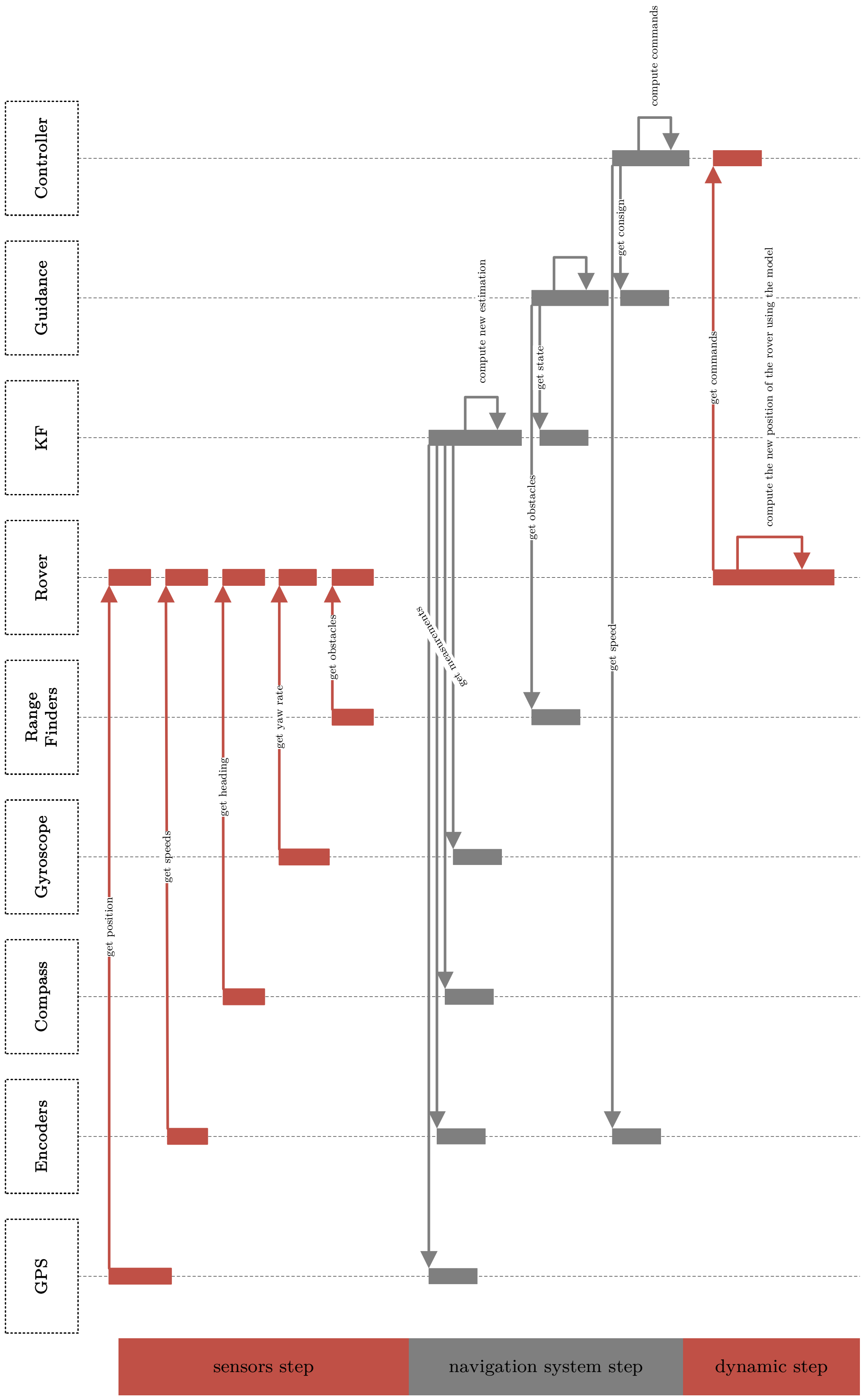


Figure 4.1 – sequence diagram of the simulation

Chapter 5

Conclusion and Remarks

This internship has been the best opportunity to connect many fields taught at ENSTA Bretagne in a challenging project. Sensors study, modelling, simulation, localization, guidance, control or even system integration. At the end of the internship, the rover simulated was autonomous but with minor flaws, and the real rover had serious flaws.

It is important to note that the main difficulties of this project was induced by the low-cost hardware components and Arduino. For instance, many weeks have been lost in solving bugs related to the i2c communication between the Arduino board and the gyroscope. The board gets stuck after few seconds or minutes in an infinite loop when reading the gyroscope. This was partially fixed by adding a flawed timeout within the official i2c Arduino library. It is important to note that Arduino may not be the best choice for a such a project. In fact, Kalman filtering requires double-precision floats while mainstream Arduino board provide only single-precision. Moreover, an Arduino board doesn't run on a microprocessor but a microcontroller that doesn't provide threading. Threading is virtually mandatory for a robot, as it must move, sense, and process data while adapting to its environment. The solution that has been developed are just workarounds. Furthermore, the frame was too low-cost and compels the use of very bad encoders. This induced many challenging problems because the state observers get lost by the poor precision of the measured velocities amongst others.

This internship was a rich experience and I hope that my tutor Jian Wan will keep taking students from ENSTA Bretagne in internship.

This conclusion was also about emphasizing on problems that can be avoided next year if other students like me from ENSTA Bretagne have the great opportunity to do their internships at the University of Plymouth with Jian Wan, who I particularly thank.

Bibliography

[1] Robin Heß and Klaus Schilling. *GPS/Galileo Testbed Using a High Precision Optical Positioning System.* Ed. by Noriaki Ando et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 87–96. ISBN: 978-3-642-17319-6. DOI: `10.1007/978-3-642-17319-6_11`. URL: `https://doi.org/10.1007/978-3-642-17319-6_11`.

[2] Luc Jaulin. *Mobile robotics.* Elsevier, 2015.

[3] E.D. Kaplan and C. Hegarty. *Understanding GPS/GNSS: Principles and Applications, Third Edition:* 2017. ISBN: 9781630814427. URL: `https://books.google.fr/books?id=y4Q0DwAAQBAJ`.

[4] Anthony Mandow et al. "Experimental kinematics for wheeled skid-steer mobile robots". In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on.* IEEE. 2007, pp. 1222–1227.

[5] Giorgio Rizzoni. *Principles and Applications of Electrical Engineering.* McGraw-Hill Science/Engineering/Math, 2005. ISBN: 0073220337.

[6] VectorNav. *VectorNav Magnetometer.* `http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/`. URL: `http://www.vectornav.com/support/library/magnetometer`.

[7] Tianmiao Wang et al. "Analysis and Experimental Kinematics of a Skid-Steering Wheeled Robot Based on a Laser Scanner Sensor". In: *Sensors* 15.5 (2015), pp. 9681–9702. ISSN: 1424-8220. DOI: `10.3390/s150509681`. URL: `http://www.mdpi.com/1424-8220/15/5/9681`.

[8] JY Wong and CF Chiang. "A general theory for skid steering of tracked vehicles on firm ground". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 215.3 (2001), pp. 343–355.

[9] Wei Yu et al. "Analysis and experimental verification for dynamic modeling of a skid-steered wheeled vehicle". In: *IEEE transactions on robotics* 26.2 (2010), pp. 340–353.