applied sciences

MDPI

Article

State Estimation of Over-Sensored Systems Applied to a Low-Cost Robotic Manipulator

João Moreira [1,*], Vítor H. Pinto [1,2], José Gonçalves [2,3] and Paulo Costa [1,2]

1 FEUP—Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal; vitorpinto@fe.up.pt (V.H.P.); paco@fe.up.pt (P.C.)
2 INESC TEC—Institute for Systems and Computer Engineering, Technology and Science, Centre for Robotics in Industry and Intelligent Systems (CRIIS), Campus of FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal; goncalves@ipb.pt
3 Research Centre in Digitalization and Intelligent Robotics (CeDRI), School of Technology and Management, Polytechnic Institute of Bragança, Santa Apolónia Campus, 5300-253 Bragança, Portugal
* Correspondence: up201506628@fe.up.pt

Abstract: There is an increasing demand for robotic manipulators to perform more complex and versatile tasks. In order to fulfill this need, expeditious calibration and estimation techniques are required as a first step for the correct usage of the manipulator. This article aims at finding a subset of these algorithms that could be used in a generic manipulator and should allow for its prompt use. Two models for the representation of the pose of the manipulator are described and used in the state estimation problem. The results of the implementation are tested, and some performance metrics are obtained.

Keywords: state estimation; robotic manipulator; over-sensored; automatic process; mechatronics

Citation: Moreira, J.; Pinto, V.H.; Gonçalves, J.; Costa, P. State Estimation of Over-Sensored Systems Applied to a Low-Cost Robotic Manipulator. *Appl. Sci.* **2021**, *11*, 2519. https://doi.org/10.3390/app11062519

Academic Editor: Manuel Armada

Received: 29 January 2021
Accepted: 8 March 2021
Published: 11 March 2021

1. Introduction

Robotics has been an integral part of society for several decades. It is present in industry, the medical field, and the military, among others. Wherever physical labor is required, robotics serves as a more economical, stronger, more precise, and overall more effective alternative to human labor. In addition, robots have been proven to be capable of performing tasks that are otherwise dangerous, if not impossible, for people to do, such as search and rescue missions in dangerous environments.

Within this very broad field, one of the most explored areas is the control of rigid structure robotic manipulators. There are still numerous problems to be solved within this scope, like the manipulator's configuration (number and type of joints, link's shape and size), that must be chosen according to its objective; its motors must be precisely controlled, and information such as the pose of the manipulator must be known to serve as an input to the controller. This last problem is commonly called in the literature state estimation. Being a common problem, many algorithms have been developed to help solve it. These algorithms read information from imperfect sensors and merge it to find what is believed to be the best estimate of the system's state.

In the context of robotic manipulators, there already are numerous examples of the usage of estimators to determine their pose. In most cases, the pose is simply defined as the angles of each joint, and it is estimated using non-linear variants of the Kalman filter. For example, in [1], several instances of the extended Kalman filter were chained to estimate the angles and angular velocities of each joint. In this example, the models of the manipulator are defined according to the orientation of the axes of rotation of the joints and the placement of the inertial measurement units, and the base is assumed fixed. In [2], the authors proposed to estimate the same variables using the extended Kalman filter by modeling them as a function of the inertial sensors' measurements. Despite being a more

generic approach, it requires each link of the manipulator to be equipped with one triad of gyroscopes and two triads of accelerometers. The study in [3] focused on the description of a variant of the Kalman filter named the adaptive square-root unscented Kalman filter and applied it to this context. It compares the results of this filter with the extended Kalman filter, the unscented Kalman filter, and its adaptive counterparts. It demonstrates that these variants are mostly suitable for the state estimation problem, each with its advantages depending on the nature of the measurements' noise. However, there is not much focus on the models used for estimation.

Examples of interesting sensors capable of obtaining important information that can be embedded are accelerometers, gyroscopes, magnetometers, and encoders, allowing the manipulator to estimate its state without the support of other external hardware. State estimation using only these kinds of sensors is considered automatic.

Estimators designed for a specific application are often better at estimating the variables of interest for the context for which they were built; however, this means that they are not portable from one system to another. That is, if another manipulator is to be built with a different purpose, another estimator must be developed for this new context. Considering this problem, the estimators applied to the system are designed to be as modular and reusable as possible. To serve as a test-bed, an over-sensored manipulator is used, and the main goal of this redundancy is to correct the sensor's errors. Beyond that, the use of a system like the one mentioned also means that the developed models can be used in other manipulators.

Throughout this article, the development process behind the presented estimators is described, as well as the developed test-bed manipulator. Section 2 presents the different models for manipulators and sensors. The experimental setup for the used manipulator is described in Section 3, while the different aspects of the implementation of the estimation algorithms are discussed in Section 4. Finally, some conclusions and remarks about future objectives are made in Section 5.

2. Models

The design process behind the implementation of a state estimator starts with knowing the system, as well as its components. More specifically, mathematical models capable of describing any of the subsystems and sensors of the robotic manipulator must be developed. A study of the used subsystems is performed throughout this section.

2.1. Inertial Measurement Unit

An inertial measurement unit (IMU) is, in its simplest form, a device composed of accelerometers, capable of measuring specific force (acceleration subtracted by the acceleration of gravity), and gyroscopes, capable of measuring angular velocity. With these measurements, position and orientation can be determined at any moment in time given an initial position, velocity, and orientation.

Most IMU sensors are built from 3 accelerometers and 3 gyroscopes, although some include 3 magnetometers as well. These trios of sensors are unaligned to provide measurements in all 3 spatial dimensions. However, these are not perfectly orthogonal to one another. This means that a measurement in one sensor might correspond to a value in more than one axis. Even if the axes were orthogonal, they still might not be aligned with the chosen referential. Another issue in modeling an IMU is the existence of the parameters of its model that depend on the specific conditions under which it is working, such as temperature and magnetic interference.

Regarding the IMU model, the parameters on which each of the IMU's sensors depend are the gain (g) and bias (b). These relate the electric signal measured by the sensor (m) and the value of the quantity being measured (v) by Equation (1).

$$v = g \times (m - b) \tag{1}$$

where v is usually expressed in m/s^2 (when measuring acceleration), rad/s (when measuring angular velocity), or μT (when measuring the magnetic field) and m is usually expressed in mV.

An especially troublesome parameter in this model is the bias. If it is not accounted for or if it is not correctly determined, it causes a systematic error in the measurements of the inertial sensors. This error accumulates when determining position and orientation since these are calculated by integrating the original measurements. As a consequence, a drift is accumulated in the position and orientation state variables.

This is the affine relation that models single axis sensors. However, as previously stated, the IMU is not made up of just one sensor. Moreover, each sensor can influence more than one axis of the chosen referential. This implies that each measurement (m_i, $i = \{x, y, z\}$) is influenced by each value (v_i, $i = \{x, y, z\}$).

That is:

$$v_i = g_{ix}(m_x - b_x) + g_{iy}(m_y - b_y) + g_{iz}(m_z - b_z) \quad (2)$$

$$i = \{x, y, z\} \quad (3)$$

Assuming $V = [v_x \quad v_y \quad v_z]^T$ and $M = [m_x \quad m_y \quad m_z]^T$, the new model equation is:

$$V = G \times (M - B_*) \quad (4)$$

where G is a 3×3 matrix with entry (i, j) equal to g_{ij} for $i = \{x, y, z\}$ and $B_* = [b_x \quad b_y \quad b_z]^T$.

Alternatively, the previous equation is often written as:

$$V = G \times M + B \quad (5)$$

with $B = -G \times B_*$. With the non-diagonal elements in the gain matrix (G), scaling factors and misalignments between the axes and the reference frame are all accounted for.

The calibration process for the IMU is finding the parameters G and B. This is usually done by reading several measurements in a multitude of scenarios and attempting to find the parameters that match the real values with the measurements. Even though this problem seems to be solvable by simply using algebra to solve the equations for the parameters, two problems stand in the way of using this method. First, all measurements are noisy, which means that if there are more measurements than parameters, the system of equations probably will not be solvable. Second, the use of these equations assumes the real values being measured are known.

Since acquiring the real values, often referred to as the ground truth, is theoretically impossible, instead of trying to achieve perfect real values, high precision instruments are used to obtain practically acceptable ones. However, with the creation of low-cost microelectromechanical system (MEMS) IMUs, the devices have become more popular among hobbyists. Those without special equipment needed to find a way to calibrate their sensors without a ground truth. This necessity brings up several new calibration techniques that rely on the properties of the measured quantities.

For example, when stationary, the accelerometers should measure a vector that has a constant norm. This property gives rise to Equation (6), which can be used to find the parameters:

$$\|G \times M + B\|_2 = g \quad (6)$$

This equation is not linear, which means least squares cannot be used directly in this case. Alternatives include using other optimization methods or redefining the variables to make the problem linear [4]. In the absence of magnetic field disturbances, this method can also be used to calibrate the magnetometers.

Alternatively, after calibrating one of the sensors, the others could be calibrated by assuming the dot product of their reference vectors (magnetic field and symmetry of

the acceleration of gravity) is constant in the working region [5]. This method has the added bonus of correcting misalignments between the frames of the accelerometers and the magnetometers.

The gyroscope does not have a reference vector to measure while it is standing still. The rotation of the Earth could be used; however, it is negligible when compared to the magnitude of the noise of low-cost MEMS gyroscopes. This absence of a reference makes it difficult to independently calibrate a gyroscope based on its static measurements. However, once the other sensors are calibrated, they can be used as an acceptable ground truth. When reading a reference vector from another sensor before and after a rotation, the rotation matrix obtained by integrating the gyroscope measurements should match those vectors [6]. This method, much like the dot product one, corrects for misalignments between sensor frames since it is using another sensor as the ground truth.

2.2. DC Motor

A DC motor is usually modeled as a resistor (R), a coil (L), and a voltage source ($v_b(t)$) in series [7,8], as presented in Equation (7).

$$v_s(t) = R \times i(t) + L \times \frac{di(t)}{dt} + v_b(t) \tag{7}$$

where $v_s(t)$ is the voltage applied to the motor and $i(t)$ is the current that goes through it.

This voltage source is a counter-electromotive force produced by the rotor, and therefore, it is proportional to the angular velocity with which it spins. Given that the power delivered at $v_b(t)$ is useful, then it is equal to the mechanical power delivered at the rotor. Since the useful electric power is equal to $v_b(t)i(t)$ and the mechanical power is given by the product of the torque $T(t)$ and the angular velocity of the rotor $\omega(t)$, the torque must be proportional to the current that goes through the stator.

$$v_b(t) = k \times \omega(t) \tag{8}$$

$$T(t) = k \times i(t) \tag{9}$$

where k is a constant with the same value for both equations. This information can be combined with the mechanical equation of the motor to define a continuous time model of the motor.

The mechanical equation relates the angular acceleration with all the torques applied to it.

$$J \times \frac{d\omega(t)}{dt} = T(t) - T_e(t) - B \times \omega(t) \tag{10}$$

In this equation, J is the mass moment of inertia, $T(t)$ is the torque produced by the motor, $T_e(t)$ is the set of opposing torques produced outside of the motor, and B is the drag coefficient (making $B \times \omega(t)$ the drag induced torque).

Combining the electric Equation (7) with the mechanical one (10), the following can be written.

$$L \times \frac{di(t)}{dt} = v_s(t) - R \times i(t) - k \times \omega(t) \tag{11}$$

$$J \times \frac{d\omega(t)}{dt} = k \times i(t) - T_e(t) - B \times \omega(t) \tag{12}$$

With these equations, the resulting continuous time model of the DC motor is:

$$\dot{x}(t) = A \times x(t) + B \times u(t) \tag{13}$$

$$y(t) = C \times x(t) \tag{14}$$

These can be presented in matrix form, as presented below:

$$x(t) = \begin{bmatrix} i(t) \\ \omega(t) \end{bmatrix} \qquad u(t) = \begin{bmatrix} v_s(t) \\ T_e(t) \end{bmatrix} \qquad y(t) = \omega(t)$$

$$A = \begin{bmatrix} -R/L & -k/L \\ k/J & -B/J \end{bmatrix} \qquad B = \begin{bmatrix} 1/L & 0 \\ 0 & -1/J \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

2.3. Robot Kinematics

Kinematics is the field of study of the motion of objects. In the case of robotic arms, this field is divided into two: forward and inverse kinematics. Forward kinematics is a tool for determining the position and orientation of any point of an arm in terms of the angles of the robot's joints. Inverse kinematics does the exact opposite: it finds out which angles should be in each joint in order to place a link in a certain position with a certain orientation.

An important aspect of robotic arms is the number of degrees of freedom (DOF). This number defines how many ways a manipulator can position and orient itself. The number of degrees of freedom a robotic arm has is equal to its number of joints. Considering there are only 3 spatial dimensions, a 6-DOF manipulator is already capable of positioning and orienting itself in (almost) any way.

The equations defining the forward kinematics of a manipulator are usually translated into matrices. These matrices correspond to linear transformations between two Cartesian referentials. Each of the matrices represents a rotation and a translation.

$$_{i-1}^{i}T = \begin{bmatrix} _{i-1}^{i}R & ^{i-1}t_i \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{15}$$

$_{i-1}^{i}R$ is a 3×3 matrix representing the rotation, and $^{i-1}t_i$ is a 3×1 vector representing the translation.

If one of these transformations is used to find the position of a link with respect to the previous one, several of these transformations can be chained to find the position of any link with respect to any other. For example, the transformation that relates link n with the base of the arm is:

$$_{0}^{n}T = {}_{0}^{1}T\,{}_{1}^{2}T \ldots {}_{n-1}^{n}T \tag{16}$$

where $_{i-1}^{i}T$ is the transformation between referentials $i-1$ and i.

Out of the many different ways of determining the individual transformations ($_{i-1}^{i}T$), one of the most used is the Denavit–Hartenberg method [9]. This systematic method of finding the entries of a transformation matrix takes only 4 parameters:

- a—distance between the previous z axis and the current one (translation along the x axis);
- α—angle between the previous z axis and the current one (rotation along the x axis);
- d—distance between the previous x axis and the current one (translation along the z axis);
- θ—angle between the previous x axis and the current one (rotation along the z axis).

These parameters should be applied in the previously described order, as is presented in Figure 1. However, it is indifferent about whether the translations and the rotations along the same axis are applied in reverse order.

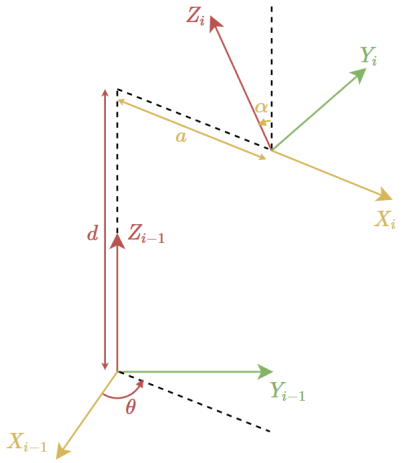


Figure 1. Example of a coordinate transformation with Denavit–Hartenberg (DH) parameters.

Once the parameters are determined, the transformation matrix is built from them.

$$ {}^{i}_{i-1}T = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & a\times\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & a\times\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17} $$

2.4. Kalman Filter

Once a system is modeled, its state can be approximated according to measurements taken from sensors. The state is a vector composed of all variables of interest of the system, and the way it is approximated is with sensor fusion techniques.

The techniques described below assume that the state of the system is a first-order Markov process. This means that the next state will depend on the current state, the system inputs, and some noise. Similarly to the next state, the measurements obtained from sensors are dependent on the state, the inputs, and some noise.

Generically, a digital system can be mathematically described by:

$$ x(k+1) = f(x(k), u(k), w(k), k) \tag{18} $$

$$ y(k) = g(x(k), u(k), v(k), k) \tag{19} $$

where x is the state, u the set of inputs, y the set of outputs, w the process noise, and v the measurement noise. It should be noted that the system may be dependent on time k, which means it is not necessarily time invariant.

To estimate the state of a system, the Kalman Filter (KF) makes some assumptions. The first is the linearity of the system, and the second is related to the process and measurement noises, both assumed to be zero-mean white noise stochastic variables. Under this assumptions, Equations (18) and (19) can be rewritten.

$$ x(k+1) = A(k)x(k) + B(k)u(k) + w(k) \qquad w(k) \sim N(0, Q(k)) \tag{20} $$

$$ y(k) = C(k)x(k) + D(k)u(k) + v(k) \qquad v(k) \sim N(0, R(k)) \tag{21} $$

where $N(\mu, \Sigma)$ is a Gaussian distribution of mean μ and covariance matrix Σ. Once again, it should be noted that the system matrices (A, B, C, and D) are not time invariant. In the case of the robotic manipulator, the system must be considered time variant, because different joint angles produce different transformations between the referentials of each link.

If these assumptions are valid, the KF is the minimum-variance unbiased estimator [10] that gives the most certain estimate out of the ones that do not produce a systematic error. The Kalman filter's algorithm is usually implemented in 2 stages: prediction and filtering.

The prediction stage uses the model of the system and the current state (or its estimate) to guess what the next state might be. Besides that, it determines the level of certainty of that guess in the form of a covariance matrix.

In the filtering stage, measurements are used to correct the guess of the prediction stage. The influence the measurements have in the final estimate depend on the previously calculated covariance matrix and the certainty granted to the measurements. A covariance matrix of the final estimate is also calculated as it is needed in the prediction stage. Assuming the process noise and the measurement noise are uncorrelated (which is the most common situation), the estimator is implemented with the following steps [11]:

Prediction:

1. $\hat{x}(k+1|k) = A(k)\hat{x}(k|k) + B(k)u(k)$ (next state estimate)
2. $P(k+1|k) = A(k)P(k|k)A(k)^T + Q(k)$ (next state estimate's covariance)

Filtering:

1. $e(k) = y(k) - C(k)\hat{x}(k|k-1) - D(k)u(k)$ (innovation)
2. $S(k) = R(k) + C(k)P(k|k-1)C(k)^T$ (innovation's covariance)
3. $K(k) = P(k|k-1)C(k)^T S(k)^{-1}$ (Kalman gain)
4. $\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)e(k)$ (current state estimate)
5. $P(k|k) = (I - K(k)C(k))P(k|k-1)$ (current state estimate's covariance)

In this algorithm, $\hat{x}$ is a state estimate, and the notation $(\alpha|\beta)$ refers to the instant at which the estimate corresponds (α) given measurements up to time β.

The variables calculated in Steps 1, 2, and 3 of the filtering stage are merely auxiliary. The innovation $e(k)$ is the error in the measurements, i.e., the difference between the expected measurements and the ones actually obtained. The innovation's covariance $S(k)$ is used to calculate the Kalman gain $K(k)$: a matrix that weighs the trust in the measurements against the trust in the model to find out which of these should have more influence in the filtered estimate.

These two stages are then applied iteratively to estimate the state at every time step. After the prediction stage, the current time instant is updated ($k \leftarrow k+1$), and the filtering stage can now be applied, since $\hat{x}(k|k-1) \leftarrow \hat{x}(k+1|k)$ and $P(k|k-1) \leftarrow P(k+1|k)$. Then, the prediction stage is executed again, making this process cyclic. Note the need for an initial estimate ($\hat{x}(0|-1)$) and its covariance ($P(0|-1)$) in order to begin the algorithm.

Recall that the Kalman filter assumes the system whose state is being estimated is linear. However, the robotic manipulator is not a linear system, some entries of the transformation matrices being trigonometric functions of the joints' angles. However, because the performance of the KF is such an appealing factor, many variants, capable of estimating the state of non-linear systems, have been developed. Some of these include the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), the Quadrature Kalman Filter (QKF), and the Cubature Kalman Filter (CKF), all of which have been proven reliable solutions to the state estimation problem in the context of robotic manipulators [1–3,12–14].

2.4.1. Extended Kalman Filter

As mentioned, one of the variants of the KF is the extended Kalman filter. This variant starts by finding the required Jacobian matrices (matrices composed of partial derivatives) and uses them to describe an approximation of the system, then it solves the estimation problem for the approximate system with a conventional KF.

$$A(k) = \frac{\partial f}{\partial x}(k) \qquad B(k) = \frac{\partial f}{\partial u}(k) \qquad C(k) = \frac{\partial g}{\partial x}(k) \qquad D(k) = \frac{\partial g}{\partial u}(k) \tag{22}$$

This simple workaround removes the assurance that this is the minimum-variance unbiased estimator. However, it produces good enough results to be used in many practical applications.

Even though this filter is usable in the context at hand, it presents some issues when used to estimate orientations. The main problem lies in the way that orientation is rep-

resented. For example, when the orientation of a body is described by a rotation matrix, then the correction of this orientation with a measurement, which is performed in Step 4 of the filtering stage, is the sum of two rotation matrices. This sum most likely is not a valid rotation matrix (i.e., is not orthogonal). This means that representing an orientation by the sum of its estimate with its error is not a valid model.

$$R_\theta = R_{\hat{\theta}} + R_\epsilon \tag{23}$$

This makes it difficult to apply the filtering step of the Kalman filter. One way to work around this problem is to assume that Equation (23) is valid and to alter the resulting matrix to become orthogonal again. If the orientation was being described by unit quaternions, this process would be even simpler, as the only change needed to turn the quaternion into a valid orientation would be to divide it by its magnitude. However, letting the filter assume that rotation matrices can simply be summed produces an incorrect model of the system, which in turn causes the filter to have a wrong sense of certainty about the state of the system [15].

A different way to model the dynamics of an orientation is through matrix multiplication.

$$R_\theta = R_{\hat{\theta}} R_\eta \tag{24}$$

This equation always produces at its output a valid rotation matrix (assuming there are no rounding errors), as long as the error matrix (R_η) is a rotation matrix as well. R_η can be made into a valid rotation matrix if it is built by an orientation representation that is not redundant (i.e., has no constraints), for example a rotation vector η. The conversion between rotation vector and rotation matrix can be performed using matrix exponentiation [11,15]. This approach to orientation estimation is called the Multiplicative Extended Kalman Filter (MEKF). This extra step in the MEKF is called the reset step, and it is what allows for the orientation to be stored in the rotation matrix while the filter operations are applied to the rotation vector. This filter is often used in spacecraft orientation estimation [16], which proves this filter's worth. Its intricacies will be further explored when the robotic manipulator model is explained.

2.4.2. Unscented Kalman Filter

Another variant of the Kalman filter is the unscented Kalman filter. This estimator does not assume linearity. Instead, it uses sigma points to describe the state's distribution, the sigma points being the vectors of possible states.

These points are calculated as a function of the previous estimate, the covariance matrices (of the state and noises), and some parameters. Then, they are updated according to the non-linear model of the system. Each of these points is then used, together with measurements, to calculate the Kalman gain. From this gain, the error between the expected measurements and the real ones, the predicted state, the filtered estimate, and its covariance matrix can be determined.

The UKF algorithm, in the case of additive noises, is described as follows [11,17]:

Sigma points:

1. $\mathcal{X}_0(k|k) = \hat{x}(k|k)$
2. $\mathcal{X}_i(k|k) = \hat{x}(k|k) + \left(\sqrt{\frac{n}{1-\mathcal{W}_0} P(k|k)}\right)_i$ $\quad i = \{1, 2, ..., n\}$
3. $\mathcal{X}_i(k|k) = \hat{x}(k|k) - \left(\sqrt{\frac{n}{1-\mathcal{W}_0} P(k|k)}\right)_{i-n}$ $\quad i = \{n+1, n+2, ..., 2n\}$

Prediction:

1. $\mathcal{X}_i(k+1|k) = f(\mathcal{X}_i(k|k), u(k), k)$ (next sigma points)
2. $\hat{x}(k+1|k) = \sum_{i=0}^{2n} [\mathcal{W}_i \mathcal{X}_i(k+1|k)]$
3. $P(k+1|k) = \sum_{i=0}^{2n} \left[\mathcal{W}_i (\mathcal{X}_i(k+1|k) - \hat{x}(k+1|k))(\mathcal{X}_i(k+1|k) - \hat{x}(k+1|k))^T\right] + Q(k)$
4. $\mathcal{Y}_i = g(\mathcal{X}_i(k+1|k), u(k), k)$ (expected "sigma" measurements)
5. $\hat{y}(k+1|k) = \sum_{i=0}^{2n} [\mathcal{W}_i \mathcal{Y}_i]$ (expected measurements)

Filtering:

1. $e(k) = y(k) - \hat{y}(k|k-1)$
2. $P_{y,y} = \sum_{i=0}^{2n} \left[\mathcal{W}_i (\mathcal{Y}_i - \hat{y}(k|k-1))(\mathcal{Y}_i - \hat{y}(k|k-1))^T \right] + R(k)$
3. $P_{x,y} = \sum_{i=0}^{2n} \left[\mathcal{W}_i (\mathcal{X}_i(k+1|k) - \hat{x}(k|k-1))(\mathcal{Y}_i - \hat{y}(k|k-1))^T \right]$
4. $K(k) = P_{x,y} P_{y,y}^{-1}$
5. $\hat{x}(k|k) = \hat{x}(k|k-1) + K(k)e(k)$
6. $P(k|k) = P(k|k-1) - K(k) P_{y,y} K(k)^T$

In the algorithm:

- n is the length of the state vector;
- $\mathcal{W}_i$ is the weight of the i-th sigma point, i.e., its contribution to the calculation of the estimates and covariances;
- $-1 < \mathcal{W}_0 < 1$;
- $\sum_{i=0}^{2n} \mathcal{W}_i = 1$, that is $\mathcal{W}_i = \frac{1-\mathcal{W}_0}{2n}$ for all $i = \{1, 2, ..., 2n\}$
- $\left(\sqrt{\frac{n}{1-\mathcal{W}_0} P(k|k)} \right)_i$ is the i-th column of the matrix $\sqrt{\frac{n}{1-\mathcal{W}_0} P(k|k)}$, and the square root matrix is obtained from the Cholesky decomposition.

3. Test Setup

Throughout this section, the robotic manipulator that is used as a test-bed is introduced and detailed.

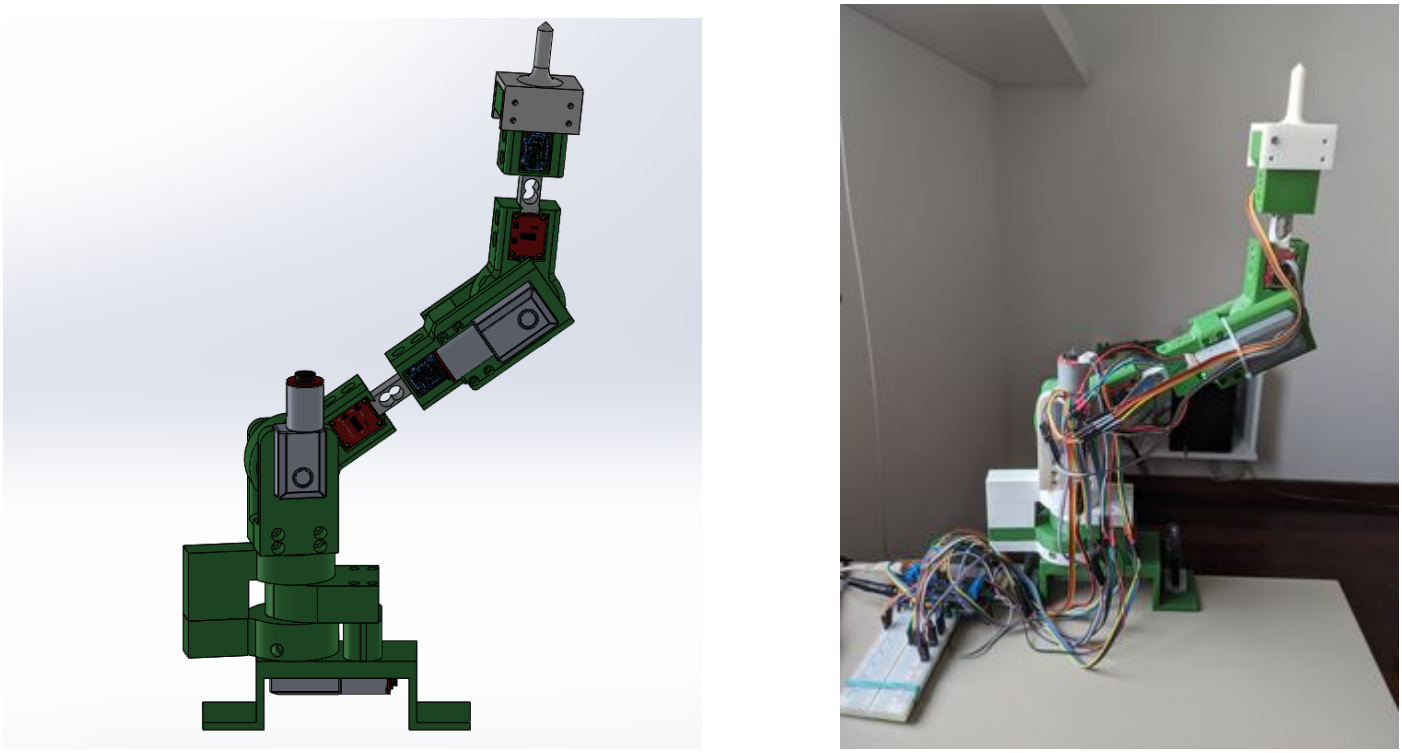
3.1. Hardware

A low-cost robotic manipulator was developed to validate the study of over-sensored systems within an academic context. With this purpose in mind, the manipulator was designed based on three criteria:

- Its structural components musts be 3D printable;
- The design and sensors have to be low-cost;
- It should be as modular as possible.

Following these objectives, the manipulator presented in Figure 2a was designed and then physically built, as shown in Figure 2b.

It is equipped with 3 JGY-371 motors with a worm gearbox and incremental encoder. The worm gearbox gives the motor the unique capability of self-locking, being able to save energy while the motor is stopped. There are 2 TAL220 load cells, mounted on both connection links, with the corresponding signal amplifiers (HX711). These make the system capable of measuring the torque applied in each link. Links 2 and 3 are also equipped with inertial measurement units. There is one MPU-9250 IMU in Link 2 and one Adafruit 3463 Precision 9-DOF IMU in Link 3. An Arduino Uno is used to process the data from the sensors and control the motors through an Adafruit Motor Shield v2. Table 1 contains a bill of materials with the price range of each component.



(**a**) 3D model. (**b**) Implementation.

Figure 2. Designed robotic manipulator.

Table 1. Bill of materials of the robotic manipulator.

Reference	Description	Price Range
TAL 220	10 kg Load Cell	10–12 €
A000066	ARDUINO UNO REV3	20–25 €
HX711	Load Cell Amplifier	12–15 €
ADA-3463	Adafruit Precision NXP 9-DOF Breakout Board	15–18 €
MPU-9250	GY-91 10DOF MPU-9250 and BMP280 Multi-Sensor Module	10–15 €
JGY-371	12V DC Motor with worm gearbox and incremental encoder	25–30 €
ADA-1438	Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit—v2.3	20–25 €
TOTAL		184–227 €

3.2. Software

The control loop of the robotic manipulator is separated into two parts. The Arduino is used for low level tasks such as reading measurements from the sensors and controlling the motors. Serving as a central processing unit, a PC application is used, connected directly to the Arduino Uno by a USB cable. This application is coded in the Free Pascal programming language, specifically using the Lazarus Integrated Development Environment (IDE). It is responsible for high level tasks such as state estimation and user interaction.

More specifically, the control loop starts in the Arduino, where raw measurements from all sensors are collected. The measurements are then transmitted to the PC via serial communication. The application receives these data and displays them in a Graphical User Interface (GUI). The GUI can also display graphically the evolution of any variable of interest and gives the user the ability to interact with the system. For example, the user can move sliders to set the position or velocity of any motor and can set the PID parameters for the control of the motors. Besides interacting with the user through the GUI,

the application pre-processes measurements according to its calibration parameters and then uses them in state estimation algorithms. The algorithms for calibration are also run in this application. The references for the motors are then sent to the Arduino, again using serial communication, which controls the motors using a PID controller. Figure 3 shows the GUI with which the user can interact.

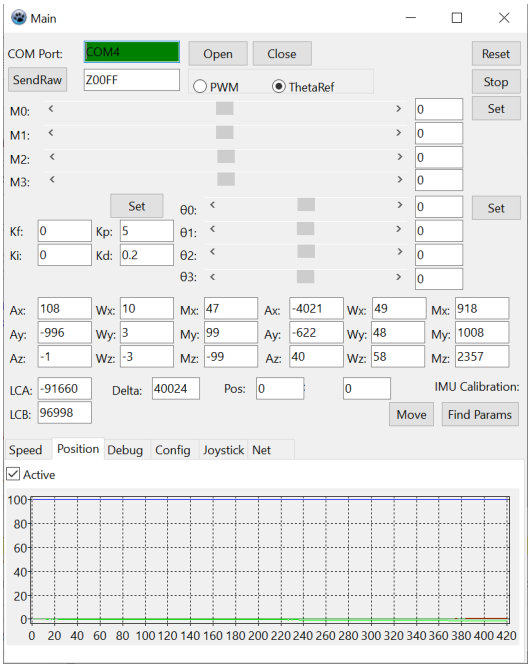


Figure 3. Graphical user interface of the PC application.

3.3. Models

Initially, the equations that define the behavior of the manipulator and its sensors are written in the most general form possible. This is done with the intent of keeping the models reusable for different manipulator configurations. However, the assumption that all joints are revolute is made because it keeps the model simpler, and it is a constraint that the implemented manipulator follows. Before describing the models, the used nomenclature is presented. Regarding the coordinate frames, two are used per joint. For joint number i (counting from the base), the two frames are i and $i-$. Both have their origins in the rotation axis of the joint and have the z axis aligned with it. However, frame i is fixed to the link directly after the joint, whereas frame $i-$ is fixed to the link directly before the joint. The frames in which the sensors take measurements can be rotated to align with the frames of the adjacent joints [1].

Some consequences of this nomenclature are that the transformation matrices ${}^{i-}_{i-1}T$ are constant and there is no translation in the transformations ${}^{i}_{i-}T$. This implies that all the necessary information required to determine the pose of the manipulator is described by the rotation matrices ${}^{i}_{i-}R$ given by:

$$ {}^{i}_{i-}R = R_{\theta_i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{25} $$

The argument can be extended to the velocity and acceleration of each joint. All the information needed to do so is the rotation matrix, angular velocity, and angular acceleration between the two joints. In the two following sections, two models are built based on this nomenclature. The sensors are also modeled according to each of the state choices.

3.3.1. Link Model

As mentioned, in this model, the state is composed of the rotation matrices ${}^{i}_{n}R$ and angular velocities ω_i of each link. Each of these represents the orientation of link i (with 0 being the base) with respect to an inertial frame n. The inertial frame is denoted by n because it refers to the navigation frame. This is a frame situated at a specific point on Earth's surface. As the planet rotates, so does the frame. Even though this frame is not inertial, it can be considered approximately inertial. As demonstrated in [15], the Coriolis and centrifugal accelerations suffered by this frame are negligible.

The choice of representing orientations with respect to an inertial frame means that any error in estimating the orientation of any link does not propagate to the next one. It also means that inertial measurements from the IMU will be easier to model as they are independent of the previous link. On the other hand, this will make sensors like encoders, which depend on the orientation of two links, have a more complex model.

The rotation matrix for each link evolves according to the following equation:

$$ {}^{i}_{n}R(k+1) = {}^{i}_{n}R(k)R_{\delta_i}(k) \tag{26} $$

where $R_{\delta_i}(k)$ is the rotation matrix corresponding to the rotation vector $\delta_i(k)$. $\delta_i(k)$ is the vector around which frame i rotates from instant k to instant $k+1$. The way the rotation matrix can be built from the rotation vector is through matrix exponentiation. The value of $\delta_i(k)$ is given by:

$$ \Delta_i(k) = \Delta t(\omega_i(k) + \epsilon_{\omega_i}(k)) = \Delta t\omega_i(k) + \epsilon_{\delta_i}(k) \tag{27} $$

with Δt the time interval between instants k and $k+1$, $\omega_i = {}^{i}\omega_{ni}$ the angular velocity of link i with respect to frame n represented in frame i, and ϵ the error of the model.

One filter designed to find the orientation of a rigid body using an IMU is the MEKF. As mentioned in Section 2.4.1, Model (26) must be linearized in order to apply the MEKF. With that intent, the rotation matrix ${}^{i}_{n}R$ is rewritten such that the state becomes a vector around an operating point.

$$ {}^{i}_{n}R(k) = {}^{i}_{n}\hat{R}(k)R_{\eta_i}(k) \tag{28} $$

In this equation, ${}^{i}_{n}\hat{R}$ is the rotation matrix representing the operating point, and R_{η_i} is the rotation around the operating point defined by the rotation vector η_i. R_{η_i} can be approximated by:

$$ R_{\eta_i}(k) \approx I_3 + [\eta_i(k)\times] \tag{29} $$

where $[\ \cdot\ \times]$ is an operator that gives a 3×3 matrix from a vector. This operator can be used to solve a cross product:

$$ a \times b = [a\times]b = -[b\times]a \tag{30} $$

and therefore is defined as

$$ [a\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{31} $$

with $a = [a_1 \quad a_2 \quad a_3]^T$. Given Equation (29), the linear model can be deduced:

$$ \eta_i(k+1) = \hat{R}^T_{\delta_i}(k)\eta_i(k) + \Delta t\epsilon_{\omega_i}(k) \tag{32} $$

Note that this model is not required for the update stage of the MEKF because η_i and ϵ_{ω_i} are zero-mean variables. This model is only needed to find the matrices required for the filtering stage.

Finally, the angular velocity ω_i is modeled as a random walk that is corrected by measurements:

$$ \omega_i(k+1) = \omega_i(k) + \epsilon_{\omega_i}(k) \tag{33} $$

Gyroscope Model

Once properly calibrated, each gyroscope should give a direct measure of one of the state variables (per link). Its measurement model is defined by the following equation:

$$\omega_{g_i}(k) = \omega_i(k) + \epsilon_{\omega_{g_i}}(k) \tag{34}$$

which is already linear.

Magnetometer Model

When considering a sensor like the magnetometer, all one needs to take into consideration is the orientation of the vector being measured, i.e., the magnetism vector. Because this vector exists within a field that is approximately constant for the neighborhood of the manipulator, the direction of the vector in the magnetometer can be considered the same as in the joint; that is $m_{m_i} = {}^i m$, where m_{m_i} is the magnetism vector in magnetometer i and ${}^i m$ is the magnetism vector in joint i. Given the magnetism vector in frame n (${}^n m$), which should be constant in time and only dependent on the manipulator's location on the Earth, the relation between the orientation of the link and the measurement is:

$$m_{m_i}(k) = {}^i_n R^T(k)\, {}^n m + \epsilon_{m_i}(k) \tag{35}$$

where ϵ_{m_i} is the magnetometer measurement noise.

The previous equation can be linearized:

$$m_{m_i}(k) = {}^i_n \hat{R}^T(k)\, {}^n m + [{}^i_n \hat{R}^T(k)\, {}^n m \times] \eta_i(k) + \epsilon_{m_i}(k) \tag{36}$$

Accelerometer Model

Considering a negligible acceleration when compared to the acceleration due to gravity, the accelerometer model is very similar to the magnetometer one. In fact, what is being measured is a rotated constant vector ($-\,{}^n g$). The measured specific force is given by:

$$f_{a_i}(k) = -\,{}^i_n R^T(k)\, {}^n g + \epsilon_{a_i}(k) \tag{37}$$

with ϵ_{a_i} the accelerometer noise.

Clearly, the condition that the IMU acceleration is negligible is false whenever the manipulator is moving. However, the variance of the accelerometers is larger when compared to the gyroscopes. This will have the effect, when applying the MEKF, of being mostly ignored when there is motion. This assumption was also followed in [1], where good joint angle estimates were still obtained despite it.

The similarity with the magnetometer model makes it simple to find the linear model of the accelerometer:

$$f_{a_i}(k) = -\,{}^i_n \hat{R}^T(k)\, {}^n g - [{}^i_n \hat{R}^T(k)\, {}^n g \times] \eta_i(k) + \epsilon_{a_i}(k) \tag{38}$$

Encoder Models

Incremental encoders measure the number of pulses produced in a time interval, which, given the resolution of the encoder, can be translated into the angular velocity of a joint. Therefore, their model is:

$$\omega_{ie_i}(k) = \left(\omega_i(k) - {}^i_n R^T(k)\, {}^{i-1}_n R(k) \omega_{i-1}(k)\right)_z + \epsilon_{ie_i}(k) \tag{39}$$

However, it can be expanded to describe the lack of angular velocity in the other axes (imposed by the joint):

$$\begin{bmatrix} 0 \\ 0 \\ \omega_{ie_i}(k) \end{bmatrix} = \omega_i(k) - {}^i_n R^T(k)\, {}^{i-1}_n R(k) \omega_{i-1}(k) + \begin{bmatrix} \epsilon_{\omega s_i, x}(k) \\ \epsilon_{\omega s_i, y}(k) \\ \epsilon_{ie_i}(k) \end{bmatrix} \tag{40}$$

where $\epsilon_{\omega s_i}$ is the error in angular velocity due to slack between the joints that allows them to slightly rotate along other directions.

3.3.2. Joint Model

The joint model describes the pose of the manipulator by the angles $\theta_i(k)$, angular velocities $\dot{\theta}_i(k)$, and angular accelerations $\ddot{\theta}_i(k)$ of each joint. These variables define the orientation of each joint relative to its previous joint instead of the n frame.

Given that the angular acceleration $\ddot{\theta}_i(k)$ is the derivative of the angular velocity $\dot{\theta}_i(k)$ and that the angular velocity $\dot{\theta}_i(k)$ is the derivative of the angle $\theta_i(k)$, the model can be immediately written.

$$\theta_i(k+1) = \theta_i(k) + \Delta t\dot{\theta}_i(k) + \frac{\Delta t^2}{2}\ddot{\theta}_i(k) + \epsilon_{\theta_i}(k) \tag{41}$$

$$\dot{\theta}_i(k+1) = \dot{\theta}_i(k) + \Delta t\ddot{\theta}_i(k) + \epsilon_{\dot{\theta}_i}(k) \tag{42}$$

$$\ddot{\theta}_i(k+1) = \ddot{\theta}_i(k) + \epsilon_{\ddot{\theta}_i}(k) \tag{43}$$

In this model, Δt is once again the time interval between the instants k and $k+1$.

One advantage of this model over the link model is that its state vector is much smaller, with only three variables per joint. Furthermore, it is a linear model. However, because the position of a specific point in the manipulator depends on the orientation of all the preceding joints, the error of each joint builds up when estimating that position. Regarding the sensor models, the advantages of this model are the disadvantages of the previous one, and vice versa. Encoders will have a much simpler model as they give direct measurements of the state, whereas inertial measurements will depend on the states of all preceding joints. Lastly, this model cannot be used to know the pose of the manipulator with respect to an inertial frame if the relation between the base frame (0) and frame n is not known. Throughout the rest of this section, the base is considered fixed relative to frame n to simplify the model.

The linearization of the models presented in this section is not calculated. This is due to the recursive nature of the models of the inertial sensors. This aspect produces complicated formulas that are configuration dependent and difficult to implement and debug. The lack of partial derivatives for those models prohibits the use of the EKF. Instead, the UKF should be used to estimate the state for this model.

Encoder Models

Given the nature of the chosen state, the encoder model is very simple:

$$\omega_{ie_i}(k) = \dot{\theta}_i(k) + \epsilon_{ie_i}(k) \tag{44}$$

with $\epsilon_{ie_i}(k)$ the incremental encoder measurement noise.

Magnetometer Model

Under the same assumptions as the link model, a magnetometer's measurements can be considered to be constant in the neighborhood of the manipulator. This means that, once again, the measurements can be considered to be obtained in the joints' frames. Given this assumption, the following can be written.

$${}^{i}m(k) = R^T_{\theta_i}(k)\,{}^{i-}_{i-1}R^{T}\,{}^{i-1}m(k) \tag{45}$$

Knowing that the measurements are noisy:

$$m_{m_i}(k) = {}^{i}m(k) + \epsilon_{m_i}(k) \tag{46}$$

This equation, along with the previous one, defines a recursive model that can be applied as long as the magnetism vector is known in any specific link. For example, if the

magnetism vector in the base (Link 0) is known, the measurement from the magnetometer in Link 2 is given by:

$$\begin{aligned} m_{m_2}(k) &= R_{\theta_2}^T(k)\,{}^{2}_{1}R^T\,{}^{1}m(k) + \epsilon_{m_2}(k) \\ &= R_{\theta_2}^T(k)\,{}^{2}_{1}R^T R_{\theta_1}^T(k)\,{}^{1}_{0}R^T\,{}^{0}m + \epsilon_{m_2}(k) \end{aligned} \tag{47}$$

Gyroscope Model

The angular velocity measured by a gyroscope is always done with respect to an inertial frame. However, given the relation:

$${}^{i}\omega_{ni}(k) = R_{\theta_i}^T(k)\,{}^{i}_{i-1}R^T\,{}^{i-1}\omega_{n(i-1)}(k) + {}^{i}\omega_{(i-1)i}(k) \tag{48}$$

the gyroscope measurements can be written as a function of the angular velocity of the joint (${}^{i}\omega_{(i-1)i}$), the angle of the joint (which affects $R_{\theta_i}^T$), and the angular velocity of the previous link with respect to the inertial frame (${}^{i-1}\omega_{n(i-1)}$). With this relation, the gyroscope model can be deduced.

$$\omega_{g_i}(k) = {}^{i}\omega_{ni}(k) + \epsilon_{g_i}(k) \tag{49}$$

with $\epsilon_{g_i}(k)$ the gyroscope measurement noise and ${}^{i}\omega_{(i-1)i}(k) = \begin{bmatrix} 0 & 0 & \dot{\theta}_i(k) \end{bmatrix}^T$.

Accelerometer Model

The acceleration of a joint can be written as a function of the acceleration of the previous joint:

$${}^{n}a_i(k) = {}^{i-1}_{n}R(k)\,{}^{i-1}\Omega_{n(i-1)}(k)\,{}^{i-1}t_i + {}^{n}a_{i-1}(k) \tag{50}$$

with ${}^{i-1}t_i$ the translation vector from joint $i-1$ to joint i in frame $i-1$ and:

$${}^{i-1}\Omega_{n(i-1)}(k) = [{}^{i}\alpha_{ni}(k)\times] + [{}^{i}\omega_{ni}(k)\times]^2 \tag{51}$$

From this relation, the specific force in a joint can be determined:

$${}^{n}f_i(k) = {}^{i-1}_{n}R(k)\,{}^{i-1}\Omega_{n(i-1)}(k)\,{}^{i-1}t_i + {}^{n}f_{i-1}(k) \tag{52}$$

Rotating it to the joint's frame:

$${}^{i}f_i(k) = R_{\theta_i}^T(k)\,{}^{i}_{i-1}R^T\left({}^{i-1}\Omega_{n(i-1)}(k)\,{}^{i-1}t_i + {}^{i-1}f_{i-1}(k)\right) \tag{53}$$

The matrix ${}^{i-1}\Omega_{n(i-1)}(k)$ is a function of the angular velocity and angular acceleration of frame $i-1$ with respect to frame n. However, the state is composed of variables that describe the angular velocity and angular acceleration of the joints. In order to write the accelerometer measurement as a function of the state variables, the angular velocity and angular acceleration are rewritten:

$${}^{i}\alpha_{ni}(k) = R_{\theta_i}^T(k)\,{}^{i}_{i-1}R^T\,{}^{i-1}\alpha_{n(i-1)}(k) + {}^{i}\alpha_{(i-1)i}(k) \tag{54}$$

$${}^{i}\omega_{ni}(k) = R_{\theta_i}^T(k)\,{}^{i}_{i-1}R^T\,{}^{i-1}\omega_{n(i-1)}(k) + {}^{i}\omega_{(i-1)i}(k) \tag{55}$$

with ${}^{i}\omega_{(i-1)i}(k) = \begin{bmatrix} 0 & 0 & \dot{\theta}_i(k) \end{bmatrix}^T$ and ${}^{i}\alpha_{(i-1)i}(k) = \begin{bmatrix} 0 & 0 & \ddot{\theta}_i(k) \end{bmatrix}^T$. Note that the influence of the angular acceleration in this model is what created the necessity for it to be included in the state vector.

With these three recursive expressions, the accelerometer can now be modeled.

$$f_{a_i}(k) = {}^{i}\Omega_{ni}(k)\,{}^{i}p_{a_i} + {}^{i}f_i(k) + \epsilon_{a_i}(k) \tag{56}$$

with ${}^{i}p_{a_i}$ the position of the accelerometer in the link and $\epsilon_{a_i}(k)$ the accelerometer measurement noise.

3.3.3. Application of the Models to the Manipulator

The models presented thus far are all symbolic and serve for all manipulators that follow the constraints initially mentioned. To make practical use of these models, the symbolic variables must be replaced by the appropriate constants for a specific manipulator.

The manipulator is represented in Figure 4 along with the frames of each joint. From this representation, the constant rotation matrices can be found. Frames 0 and $1-$ are aligned, as are are Frames 2 and $3-$. This means the rotation matrices between these frames are the identity.

$$ {}^{1-}_{0}R = {}^{3-}_{2}R = I_3 \tag{57} $$

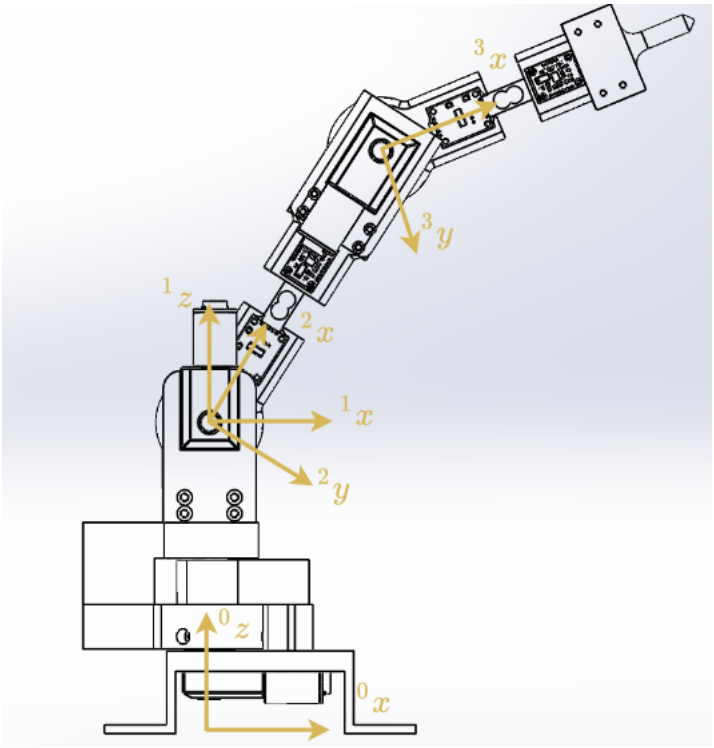


Figure 4. Manipulator sketch with coordinate frames.

In fact, if all joints have their angle at zero, the only referentials that are misaligned are 1 and $2-$. Using the Denavit–Hartenberg (DH) method, the rotation matrix ${}^{2-}_{1}R$ is the result of a rotation of $-90°$ around z followed by a rotation of $-90°$ around x.

$$ {}^{2-}_{1}R = R_{-90°x} R_{-90°z} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{58} $$

The translation vectors between the frames were measured with a ruler as precisely as possible. The following measurements are all in millimeters.

$$ {}^{0}t_1 = \begin{bmatrix} 0 \\ 0 \\ 174.5 \end{bmatrix} \qquad {}^{1}t_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad {}^{2}t_3 = \begin{bmatrix} 180.5 \\ 0 \\ 0 \end{bmatrix} \qquad {}^{3}t_{tip} = \begin{bmatrix} 190.5 \\ 0 \\ 0 \end{bmatrix} \tag{59} $$

Note that the transformation between Frames 1 and 2 is purely a rotation, which implies that vector ${}^{1}t_2$ is a null vector.

The positions of the accelerometers with respect to their corresponding joints are some of the constants in the accelerometer models. These were found by measuring the position of the IMUs with a ruler and adding the offset of the MEMS device present in the datasheet.

$$^{2}p_{a_2} = {}^{3}p_{a_3} = \begin{bmatrix} 94 \\ -1 \\ 11 \end{bmatrix} \tag{60}$$

Finally, the last constants needed are the time interval of the filter (Δt), which is kept undefined for now, the gravity vector in the navigation frame n, and the magnetism vector in the same frame. Assuming the base frame (0) is the same as the navigation frame (n) and that this frame follows the north-west-up convention, the gravity vector is $^{n}g = [0 \quad 0 \quad -9.81]^{T}$. The magnetic field vector in the navigation frame was later determined using the onboard magnetometers. From this information, the models developed earlier can be put to use.

4. State Estimation

The two different descriptions of pose, the link model and the joint model, are used with different filters. The link model has its state estimated using the MEKF as described in Section 2.4.1, whereas the UKF, which is described in Section 2.4.2, is used to estimate the state of the joint model. The covariance matrices in the algorithms were diagonal since the cross variance between different variables was negligible. The empirical values obtained for the variances of the state variables were:

- 4×10^{-5} for $\epsilon_{\eta_{ij}}$, with $i = \{1, 2, 3\}$ and $j = \{x, y, z\}$ (link model);
- 1×10^{-3} for $\epsilon_{\omega_{ij}}$, with $i = \{1, 2, 3\}$ and $j = \{x, y, z\}$ (link model);
- 1×10^{-7} for ϵ_{θ_i}, ϵ_{ω_i} and ϵ_{α_i}, with $i = \{1, 2, 3\}$ (joint model).

The sensors' variances were obtained experimentally and are presented in Table 2 in their respective units ($(\mathrm{rad/s})^2$ for gyroscopes and encoders and $(\mathrm{m/s^2})^2$ for accelerometers).

Table 2. Variance of the errors of the sensors.

Gyroscopes	Variable	$\epsilon_{\omega_{g_A x}}$	$\epsilon_{\omega_{g_A y}}$	$\epsilon_{\omega_{g_A z}}$	$\epsilon_{\omega_{g_B x}}$	$\epsilon_{\omega_{g_B y}}$	$\epsilon_{\omega_{g_B z}}$
	Variance	3.4×10^{-6}	6.5×10^{-6}	1.1×10^{-5}	9.7×10^{-6}	8.4×10^{-6}	1.3×10^{-5}
Accelerometers	Variable	$\epsilon_{a_{Ax}}$	$\epsilon_{a_{Ay}}$	$\epsilon_{a_{Az}}$	$\epsilon_{a_{Bx}}$	$\epsilon_{a_{By}}$	$\epsilon_{a_{Bz}}$
	Variance	6.7×10^{-6}	3.1×10^{-5}	2.3×10^{-5}	8.8×10^{-5}	1.9×10^{-4}	7.0×10^{-5}
Encoders	Variable	ϵ_{ie_1}	$\epsilon_{\omega s_1}$	ϵ_{ie_2}	$\epsilon_{\omega s_2}$	ϵ_{ie_3}	$\epsilon_{\omega s_3}$
	Variance	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}

After implementing the filters, each of the sensor models were tested individually (along with the manipulator model) to verify that they were working correctly. Most of these tests were performed with the manipulator stationary or while following this set of movements:

- $\theta_1 = 0°$ $\theta_2 = 0°$ $\theta_3 = 0°$ @0s
- $\theta_1 = 0°$ $\theta_2 = 90°$ $\theta_3 = 0°$ @5s
- $\theta_1 = 0°$ $\theta_2 = 90°$ $\theta_3 = -90°$ @10s
- $\theta_1 = 90°$ $\theta_2 = 90°$ $\theta_3 = -90°$ @15s
- $\theta_1 = 90°$ $\theta_2 = 0°$ $\theta_3 = 0°$ @20s
- $\theta_1 = -90°$ $\theta_2 = 0°$ $\theta_3 = 0°$ @25s
- $\theta_1 = -90°$ $\theta_2 = 90°$ $\theta_3 = 0°$ @30s
- $\theta_1 = -90°$ $\theta_2 = 90°$ $\theta_3 = -90°$ @35s
- $\theta_1 = 0°$ $\theta_2 = 90°$ $\theta_3 = -90°$ @40s
- $\theta_1 = 0°$ $\theta_2 = 0°$ $\theta_3 = 0°$ @45s

These movements were controlled using a PD controller with a proportional gain of five and a differential gain of 0.2. The error that served as the input to this controller was the difference between the references described above and the dead-reckoning of the encoders' measurements. On their own, the encoders do not provide an accurate ground truth. However, their measurements are reliable enough to verify the correctness of the models. Once this process was complete, all the sensors were used to test the accuracy of the estimators at finding the position of the tip of the manipulator. This accuracy test was performed with the aid of a structure with points in specific locations. This structure is displayed in Figure 5.

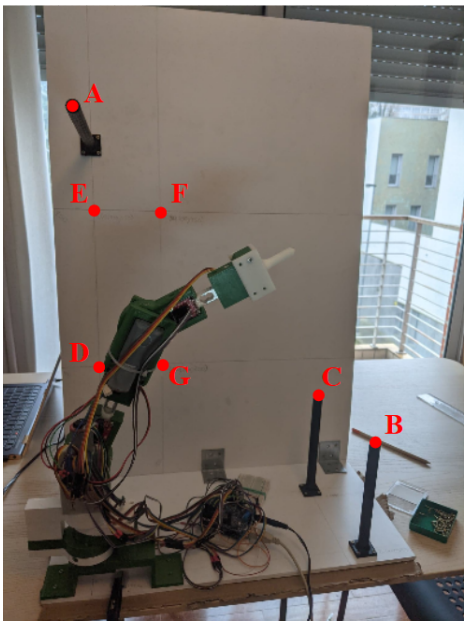


Figure 5. Structure used for accuracy tests with its interest points marked.

By manually driving the tip of the manipulator to each of the structure's points of interest and calculating the position from the estimated state, the error of the methods can be found for the different points. The tip was guided to the points in alphabetical order, starting at Point A, going through all the others, and finishing at A again. The coordinates of the presented points, in millimeters, are:

- A: $(0; 0; 545.5)$
- B: $(371; 0; 174.5)$
- C: $(340.5; 141.5; 174.5)$
- D: $(0; 224; 200)$
- E: $(0; 224; 450)$
- F: $(100; 224; 450)$
- G: $(100; 224; 200)$

This analysis is performed for the estimator of the link model in Section 4.1 and for the estimator of the joint model in Section 4.2.

4.1. Link Model Estimation

To verify the correctness of the sensor models, the estimated state of the link model, which is composed of rotation matrices, had to be converted into the angles of the joints. These angles could then be compared to the baseline created by dead-reckoning of the encoder measurements. The matrix that should be converted to a vector is ${}_{i-}^{i}R(k)$ for joint i. This can be obtained from the state matrices as:

$$ {}_{i-}^{i}R(k) = {}_{i-1}^{i-}R^{T}\,{}_{n}^{i-1}R^{T}(k)\,{}_{n}^{i}R(k) \tag{61} $$

Since the rotation of joint i is around the z axis of frame i for all joints, the expected rotation vectors should have their x and y components (θ_{ix} and θ_{iy}) close to zero, and the z component (θ_{iz}) should match the baseline angle (θ_{ib}).

4.1.1. Gyroscope Model

Using just the gyroscopes, the orientation of Link 1 remained static throughout the execution of the test. This is a consequence of the lack of an IMU in that link. Under these circumstances, the orientations of each link are independent. Therefore, there is nothing preventing adjacent links from being rotated by an axis different than the one allowed by the joints. This is evident in the graph of Figure 6.

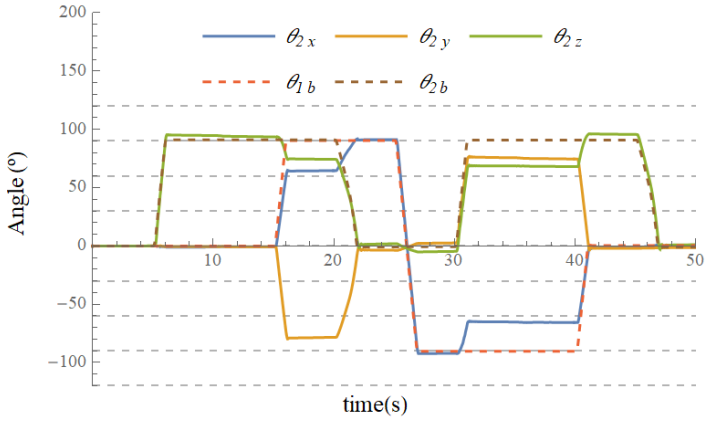


Figure 6. Evolution of Rotation Vector 2 when testing the link model using only gyroscope measurements.

This graph shows the evolution of the rotation vector θ_2 and the baseline for the first two angles (θ_{1b} and θ_{2b}). Note that θ_{1b} was included here since there would be no relation between this angle and the one estimated by the link model as the estimated orientation of Link 1 is static.

However, there should be a relation between θ_{1b} and θ_{2x}, because when $\theta_2 = 0$, a rotation of θ_1 is equivalent to a rotation of Link 2 around its x axis. This association is evident in the time intervals when $\theta_1 \neq 0$, that is $[15; 40]$ s. Note this relation is even stronger in the interval $[20; 30]$ s, which is when $\theta_2 = 0$.

The relation between θ_{2b} and θ_{2z} is also evident in the intervals $[5; 20]$ s and $[30; 45]$ s. Again, because the rotation vector is not always pointing along the z axis of Frame 2, the association is mostly present when $\theta_1 = 0$, which is in intervals $[5; 15]$ s and $[40; 45]$ s. However, even in these intervals, the value of θ_{2z} seems to be slightly higher than that of θ_{2b} by about 5°. This slight overshoot is actually visible when performing the test, which means the gyroscope is more accurate than the encoders. The reason the encoder measures a smaller value than the real one is that the movement performed at instants 5 s and 30 s is so fast that the encoder misses some pulses. This kind of error is correctable by the other sensors.

In the moments when $\theta_1 \neq 0$ and $\theta_2 \neq 0$, the rotation vector points along a direction that also has a y component, which is visible in the intervals $[15; 20]$ s and $[30; 40]$ s. These also correspond to the intervals when there is not a direct correlation between the baseline variables and the corresponding rotation vector components.

Figure 7 shows the graph with the components of Rotation Vector 3 (θ_{3x}, θ_{3y}, and θ_{3z}) and the baseline for the angle of the third joint (θ_{3b}). Assuming the orientation of Link 2 is being correctly estimated, this rotation vector should only have a z component. As expected, θ_{3z} roughly matches θ_{3b}.

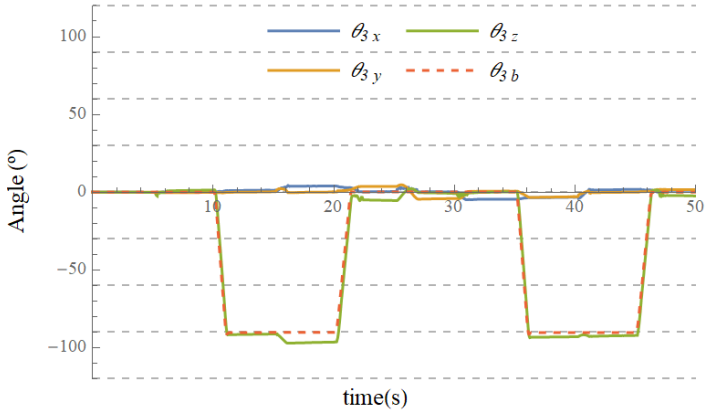


Figure 7. Evolution of Rotation Vector 3 when testing the link model using only gyroscope measurements.

However, the x and y components of the rotation vector are still significant during some periods. This discrepancy starts to be noticeable at 15 s, which is exactly when the first joint starts rotating out of the 0° angle. This error is justifiable by a slight tilt of the manipulator in one direction that should not be allowed by the manipulator. Nevertheless, it is relevant to have the model notice these imprecisions, as they influence the estimate of the position of the manipulator's tip. The reason the error only appears at that moment is related to the joint that moves. Once θ_1 varies, inertia in the manipulator can cause it to tilt to a different side from the original. Nevertheless, the x and y components are small when compared to the length of the rotation vector, which is approximately equal to θ_{3z}.

4.1.2. Accelerometer Model

Similarly to the gyroscope test, the orientation of Link 1 remained static throughout the execution of all movements for the test using only accelerometers. This is once again due to the lack of an IMU in Link 1. However, because the gravity acceleration vector was approximately parallel to the axis of rotation of Joint 1, the angle of this joint has no relation with any component of the available rotation vectors. Much like the previous test, the system is not fully observable. Nevertheless, the remaining angles can be estimated using the information from the accelerometers.

Figure 8 shows the evolution of the components of Rotation Vector 2 along with the baseline for the angles of Joints 1 and 2. The baseline for Joint 1 (θ_{1b}) is only included in the graph to show that it is not related to the rotation vector.

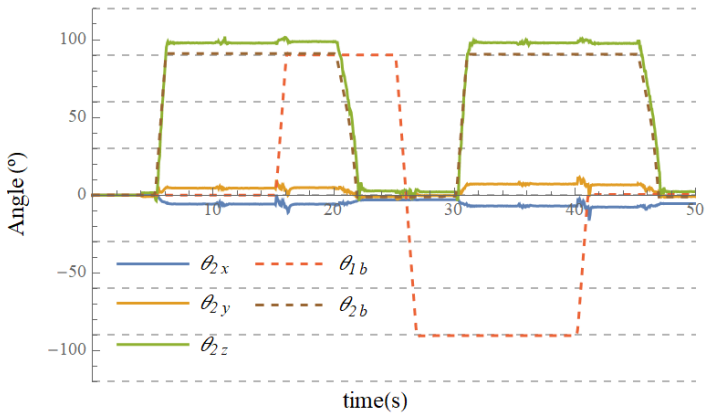


Figure 8. Evolution of Rotation Vector 2 when testing the link model using only accelerometer measurements.

As expected, the angle of Joint 2 is coincident with θ_{2z}. Furthermore, there are peaks in acceleration during the transitions, which influence the rotation vector in a visible way.

The accelerometer measurements even justify the encoder errors that were found using the gyroscope test. Once again, there is an overshoot of θ_{2z}.

An unexpected detail is the small value for the θ_{2x} and θ_{2y} variables that starts being noticed at around 5 s. This error is probably associated with imperfections in the initial angle of Joint 3. The initial gravity acceleration vector, and therefore the one expected in the inertial frame, is calculated as the average of the first 100 measurements of the accelerometers of IMU B. If θ_3 is slightly different from zero, then the initial gravity vector for Link 2 will be inconsistent with the one from Link 3. Therefore, the orientation of Link 2 changes slightly to match the initial acceleration vector.

The graph in Figure 9 displays the coherence between θ_{3z} and θ_{3b}. It also demonstrates that the initial error of θ_3 does not affect the orientation of Link 3, since the slight orientation change of Link 2 produces errors in θ_{2x} and θ_{2y} that are exact opposites of θ_{3x} and θ_{3y}. The peaks of acceleration during the transitions are also more noticeable in this link as a consequence of it being further from the base.

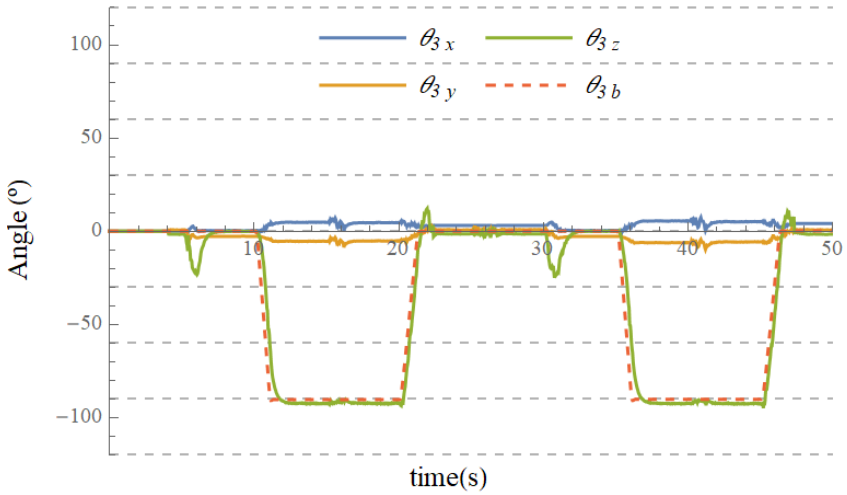


Figure 9. Evolution of Rotation Vector 3 when testing the link model using only accelerometer measurements.

4.1.3. Encoder Model

If the encoder model is correctly implemented, the z component of the rotation vectors obtained from the orientation estimates of each link should perfectly match the baseline angles since the baseline is obtained using the encoders. This is mostly verified in Figures 10–12.

There are slight changes noticeable when a different encoder detects movement. For example, at 10 s, which is when θ_3 changes, θ_2 is slightly affected by that movement. This is a natural consequence of the way ω_i was modeled. Since it follows a random walk model, there should be some inertia when altering the orientation of a link. For example, if θ_3 changes, then the expectations of the model for Links 2 and 3 to stand still force θ_2 to also change. However, the lack of a change in that joint detected by its encoder quickly reverts this small effect.

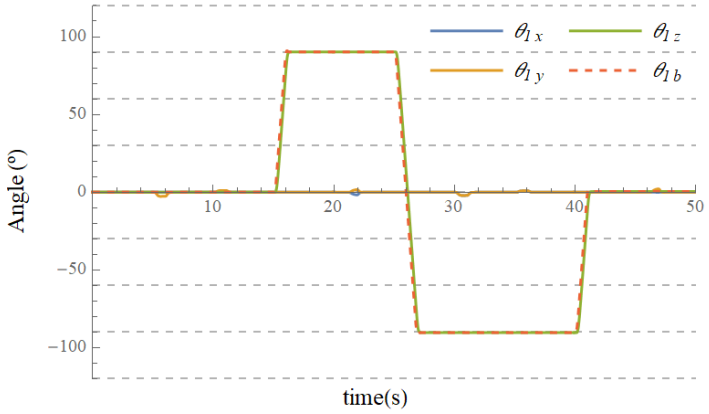


Figure 10. Evolution of Rotation Vector 1 when testing the link model using only encoder measurements.

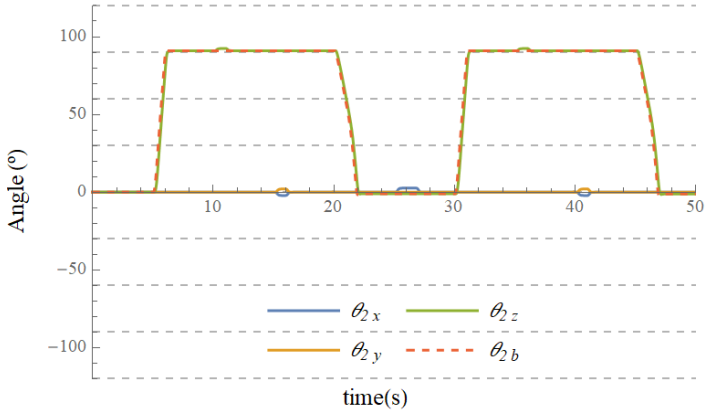


Figure 11. Evolution of Rotation Vector 2 when testing the link model using only encoder measurements.

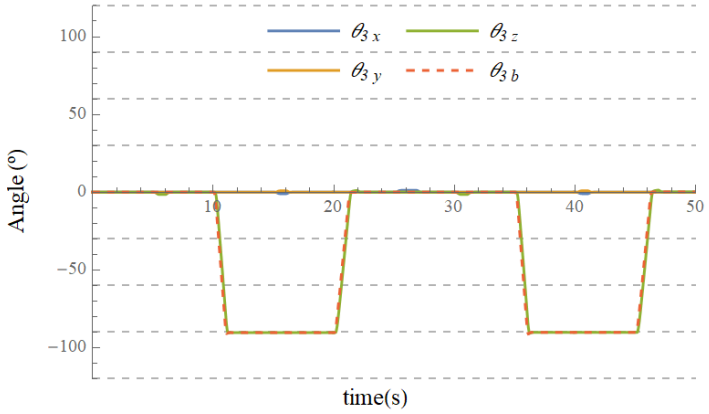


Figure 12. Evolution of Rotation Vector 3 when testing the link model using only encoder measurements.

4.1.4. Full Model

Combining the measurements of all available sensors produces the estimates from Figures 13–15.

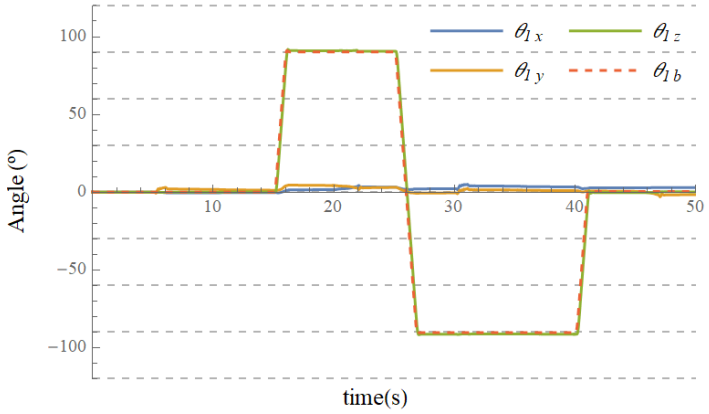


Figure 13. Evolution of Rotation Vector 1 when testing the link model using all measurements.

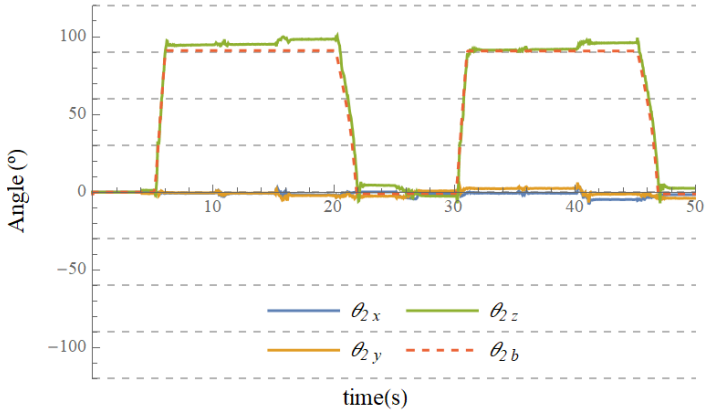


Figure 14. Evolution of Rotation Vector 2 when testing the link model using all measurements.

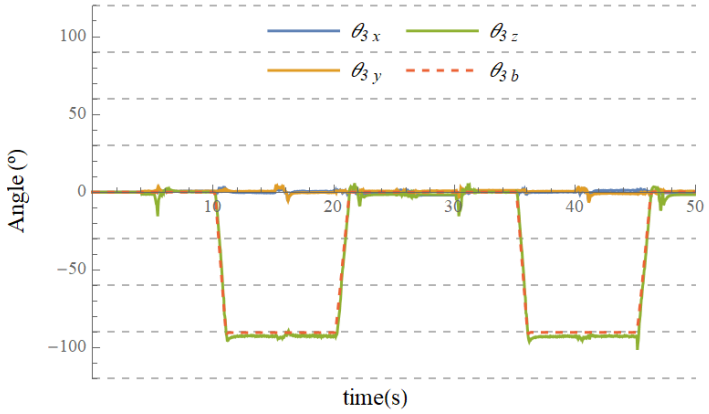


Figure 15. Evolution of Rotation Vector 3 when testing the link model using all measurements.

Clearly, the full model detects the overshoot of θ_2, as well as the smallvalues of the x and y components of the rotation vectors that the encoders cannot measure. Furthermore, this model is capable of estimating the value of θ_1 since it limits the relative orientations of adjacent links and attributes changes along the x axis of Links 2 and 3 to the first encoder's measurements. This gives the model a level of awareness that it lacks when using only the gyroscopes and the accelerometers, making the system fully observable.

Table 3 displays a summary of the previously discussed errors of the link model when using individual sensors.

Table 3. Summary of the errors of the link model using individual sensors.

Sensors Used	Problem	Cause
Gyroscopes	Static θ_1 rotation vector	Missing IMU in Link 1
	No direct relation of the components of θ_2 with θ_{1b} and θ_{2b}	Static orientation of Link 1 causes the orientation of Link 2 to have 2 DOF
Accelerometers	Static θ_1 rotation vector	Missing IMU in Link 1
	Significant values for the components θ_{2x} and θ_{2y}	Initial misalignment of Link 2 relative to Link 3
	Significant noise on θ_{3z} during transitions	Large distance from the base of the manipulator
Encoders	Low noise on θ_{2b}	"Inertia" of the model causes the orientations of the links to fall behind
	Overshoot not detected	Fast angular velocity of a joint causes the encoder to miss pulses

4.1.5. Accuracy Test

The accuracy test was performed using the link model under the previously specified conditions.

The norm of the error vectors was collected for each point and is displayed in Table 4. All the values from the table are presented in millimeters. Even though the tip of the manipulator was initially at Point A, its error is not displayed. This is because, at the starting point, the error is always zero since the initial conditions provided to the estimation methods correspond to the expectations of the position of the tip.

Table 4. Norm of the errors of the accuracy test performed on the link model.

Point	B	C	D	E	F	G	A
Norm of the Error (mm)	9.50	12.69	21.40	20.25	43.79	56.64	40.75
Norm of the Error projected on the xy-plane (mm)	9.49	10.60	19.32	16.19	42.48	54.98	40.04
Norm of the Error projected on the z axis (mm)	0.28	6.98	9.20	12.16	10.64	13.62	7.57

The maximum error has a norm of 5.7 cm, which is quite large in the scale of the manipulator. Note that most of the error is located in the horizontal projection. Furthermore, this error has a tendency to increase, except when reaching Points E and A. This large error is caused by a drift of θ_1. The accelerometers are the only sensors capable of correcting for sources of drift. Given that the acceleration of gravity is almost parallel to the rotation axis of θ_1, drift on this angle cannot be compensated. The reason this error is decreased when reaching Points E and A is that the error of the tip's position becomes less dependent on the value of θ_1 as the manipulator approaches a vertical pose.

A confirmation of the cause of the problem can be observed in Table 5. From it, it becomes even more evident that the error in θ_1 increases and that it is the most responsible joint for the accumulation of error.

Table 5. Error of the angles of the joints during the accuracy test for the link model.

Point	B	C	D	E	F	G	A
Error in θ_1 (°)	1.85	5.49	13.74	18.06	19.28	23.27	23.74
Error in θ_2 (°)	4.48	1.85	6.54	10.66	16.63	17.20	9.17
Error in θ_3 (°)	1.35	2.80	4.61	4.54	3.40	26.81	3.59

4.2. Joint Model Estimation

Unlike with the previous model, no conversion is needed in order to compare the results with the baseline given by the encoders. The state of the joint model already includes the best estimates of each angle.

4.2.1. Gyroscope Model

Figure 16 shows the estimates of the joints' angles obtained with the joint model using only gyroscope measurements. There is clearly an aberrant behavior that manifests at the first sign of movement. When θ_2 changes at 5 s, the estimate of θ_2 (θ_{2est}) falls short of the baseline (θ_{2b}), and the estimate of θ_3 (θ_{3est}) also changes. This error in θ_{3est} persists after rotation of that joint at 10 s. Later, when θ_1 changes at 15 s, the error in both estimates (θ_{2est} and θ_{3est}) is corrected.

These errors are the first sign of a flaw in the joint model. This model assumes that the pose of the manipulator is fully describable by the angles of each joint. However, as the results of the link model show, there can be slight misalignments between the links that result in a slightly different pose. These misalignments are not accounted for by the joint model. In this case, the consequence is that the small difference of the rotation axes of Joints 2 and 3 results in a different amount of rotation measured in the axes of the inclined joints.

The errors are corrected when θ_1 changes because the different IMUs measure that rotation in different axes. Assuming $\theta_2 \approx 90°$ and $\theta_3 \approx -90°$, the rotation of Joint 1 should be measured mostly in the y axis of IMU A and on the x axis of IMU B. Therefore, the only way the model can justify this change is by quickly altering its estimates to the actual values of θ_2 and θ_3. The full model avoids this flaw using the accelerometers to correct the steady-state value of these estimates.

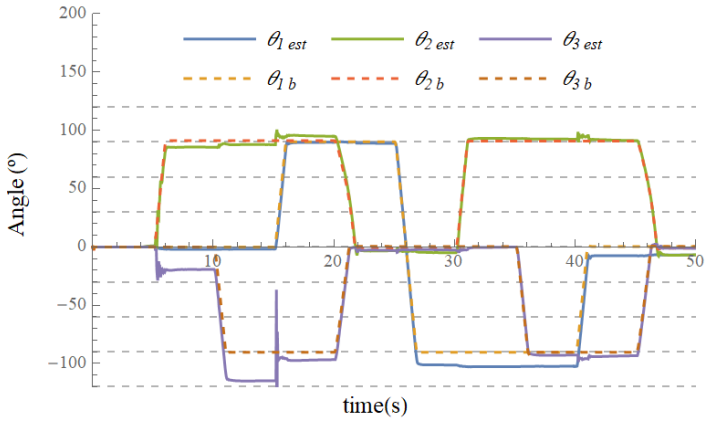


Figure 16. Evolution of estimates of the joints' angles when testing the joint model using gyroscope measurements.

4.2.2. Accelerometer Model

The accelerometer test results present even stronger evidence of the flaw in the joint model.

The graph in Figure 17 was obtained from the first two seconds of the execution of a static test. Despite the absence of movements, the results imply that the manipulator is spinning its first joint at high speed.

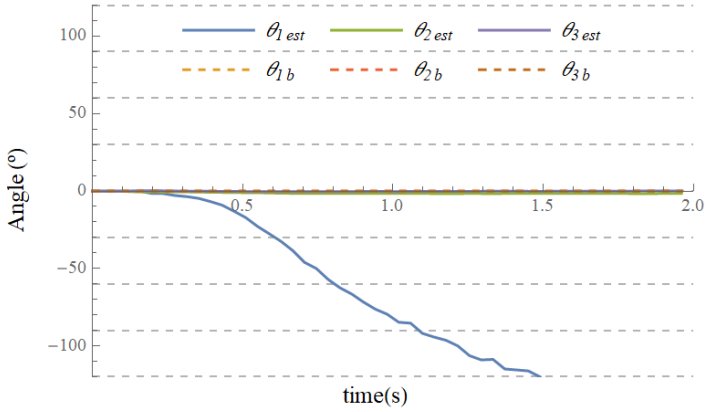


Figure 17. Evolution of the estimates of the joints' angles during a static test of the joint model using accelerometer measurements.

Once again, the culprit behind this problem is the tilt, which causes a misalignment of the links from their expected frames. This tilt causes a part of the gravity acceleration vector to be projected into the yz-plane of the IMUs. However, if the manipulator was perfectly vertical, the same vector should only be noticed in the x axis of the IMUs. Since there is no acceleration expected in that plane from that vector, the acceleration in that plane is justified as centrifugal acceleration. Considering the distance of the IMUs to the axis of rotation, the angular velocity must be high in order to justify the small acceleration measured in the horizontal plane. This angular velocity translates to a quickly changing value of θ_1.

The solution adopted for this problem was to deny the model the ability to justify the acceleration in the horizontal plane as centrifugal acceleration. In other words, the accelerometer model was simplified to assume the only acceleration measured was due to the acceleration of gravity:

$$^{i}g(k) = R^{T}_{\theta_i}(k)\,{}^{i-}_{i-1}R^{T\,i-1}g(k) \tag{62}$$

$$f_{a_i}(k) = -\,{}^{i}g(k) + \epsilon_{a_i}(k) \tag{63}$$

This is the same assumption as the one used in the link model. This new model is similar to the model of the magnetometer since it is equivalent to measuring a rotated vector that is static in inertial space. The results of this corrected model to the test with the usual movements are presented in Figure 18. Note that this model is still incapable of finding the value of θ_1.

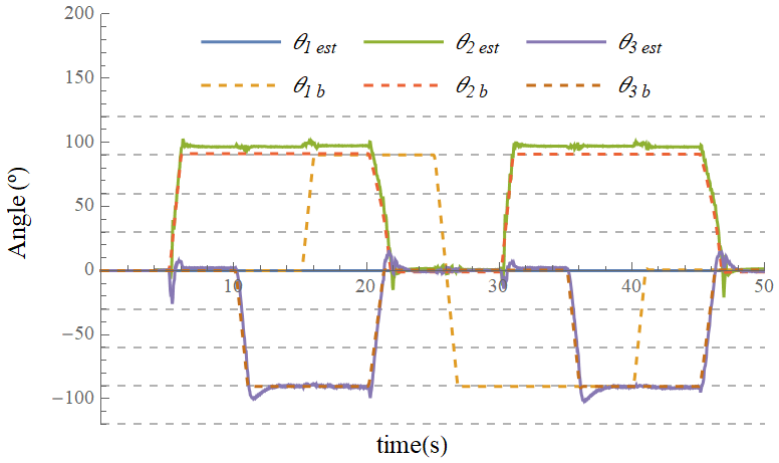


Figure 18. Evolution of estimates of the joints' angles when testing the corrected joint model using accelerometer measurements.

4.2.3. Encoder Model

The graph from Figure 19 displays what was expected: the estimates of the joint model using only encoders follow the baseline obtained from the same encoders. This is an obvious outcome, which has its limitations. The measurements from the encoders do not constitute a reliable ground truth and therefore should not be trusted as an accurate estimate. For example, this model is incapable of detecting overshoots like the ones observed in Figures 16 and 18 of the results of the gyroscope and accelerometer models.

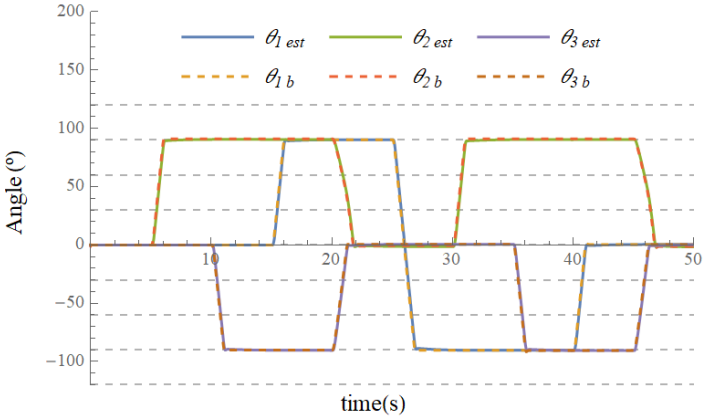


Figure 19. Evolution of estimates of the joints' angles when testing the joint model using encoder measurements.

4.2.4. Full Model

Finally, the results of the full model for the same test are presented in Figure 20. The gyroscope problem of the initially incorrect steady-state estimates of θ_2 and θ_3 is removed, and the overshoots are detected.

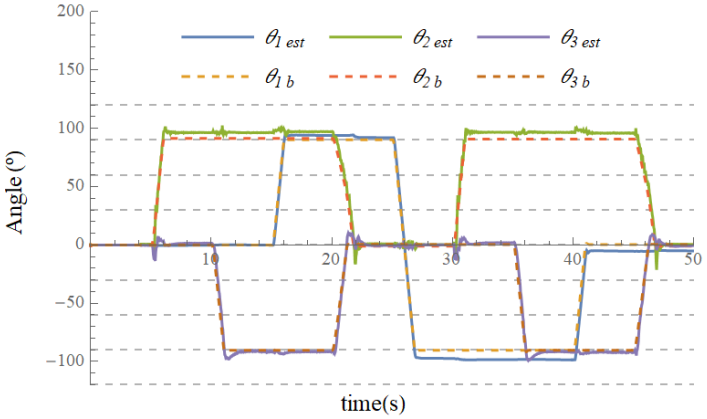


Figure 20. Evolution of estimates of the joints' angles when testing the joint model using all measurements.

A summary of the errors observed when using individual sensors is displayed in Table 6.

Table 6. Summary of the errors of the joint model using individual sensors.

Sensors Used	Problem	Cause
Gyroscopes	Incorrect amount of rotation estimated for θ_2 and θ_3	Misalignment of the rotation axes caused by a misalignment of the links' orientations
Accelerometers	Fast change of θ_1	A small acceleration projected into the horizontal plane is obtained from the tilt in the links and justified as centrifugal acceleration
	(After correction) inability to estimate θ_1	Gravity acceleration vector parallel to rotation axis of joint 1
	(After correction) significant noise on θ_{3est} during transitions	Large distance from the base of the manipulator
Encoders	Overshoot not detected	Fast angular velocity of a joint causes the encoder to miss pulses

4.2.5. Accuracy Test

The errors observed during the accuracy test can be found in Table 7. Much like the link model, the drift problem is maintained since the sensors used are the same.

Table 7. Norm of the errors of the accuracy test performed on the joint model.

Point	B	C	D	E	F	G	A
Norm of the Error (mm)	1.50	29.83	7.59	4.45	24.30	35.60	2.12
Norm of the Error projected on the xy-plane (mm)	1.06	29.80	6.44	3.67	23.94	35.55	2.12
Norm of the Error projected on the z axis (mm)	1.07	1.37	4.02	2.52	4.19	1.90	0.01

A notable difference is the overall performance of this filter relative to the other. The maximum error was around 5.7 cm for the link model, whereas the error for the joint model always stays below 3.6 cm. This improvement is not drastic. However, if the effects of drift are set aside, the vertical projection of the error goes from a maximum of 13.6 mm in the link model to 4.2 mm in the joint model. Apparently, the freedom given to the link model of tilting the links independently allows it to tilt to far off the real orientations. These extra degrees of freedom cause the drift to project to the other axes, allowing the maximum error to be higher.

For the sake of completeness, Table 8 presents the errors of the joints' angles. Overall, the error of this model, when not considering drift, is always below 1 cm, which, given the limited precision with which the structure was assembled, means this model might have even better performance when tested with a different ground truth.

Table 8. Error of the angles of the joints during the accuracy test for the joint model.

Point	B	C	D	E	F	G	A
Error in θ_1 (°)	0.16	4.63	0.93	0.85	5.55	8.33	10.01
Error in θ_2 (°)	0.83	0.61	0.31	0.08	1.69	0.01	0.02
Error in θ_3 (°)	1.94	1.60	2.22	1.03	4.56	0.75	0.68

5. Conclusions

In this paper, a robotic manipulator has its state estimated using two different filters, each of which applies to a different model. After a description of the different components of the manipulator and of some of the possible solutions to the state estimation problem, both models are depicted in a modular and reusable way.

The filters are then implemented and tested in a real platform. During the implementation of the filters, the correctness of the two used models is verified with extensive tests on the individual sensors. These reveals flaws in each individual sensor. However, when combined, these flaws are corrected. The only one that is uncorrectable by sensor fusion is the one from the joint model, which originated from the accelerometers measuring a small horizontal acceleration. This acceleration is caused by a slight tilt of the manipulator; however, the model justifies it as centrifugal acceleration, which causes the model to portray the movement of the manipulator as rapidly spinning along one axis. This problem is corrected by simplifying the accelerometers' model, by removing its ability to consider centrifugal acceleration.

Despite this flaw of the joint model, it still outperforms the link model in an accuracy test. This test is performed resorting to an external structure with specified points marked in multiple strategic predefined locations. The estimate of the position of the manipulator's tip is compared to the coordinates of the points of interest when the tip is driven to those points. Both models suffer from drift around an axis. Despite that fact, during the execution of the accuracy test, the joint model is able to keep its maximum vertical error under 5 mm. This is an error that is not affected by drift and therefore represents the accuracy the filter would have achieved if the drift problem were not present.

A possible solution to this issue would be to find a way to correctly calibrate the magnetometers. These would provide measurements that would compensate for drift in the direction in which it existed. Besides an improved calibration technique, the proposed models can be expanded to work with other sensors.

Author Contributions: Conceptualization, J.M., V.H.P., and P.C.; methodology, J.M. and P.C.; software, P.C., V.H.P., and J.M.; validation, J.M., V.H.P., and P.C.; investigation, J.M.; writing, original draft preparation, J.M.; writing, review and editing, J.M., V.H.P., J.G., and P.C. All authors read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cantelli, L.; Muscato, G.; Nunnari, M.; Spina, D. A Joint-Angle Estimation Method for Industrial Manipulators Using Inertial Sensors. *IEEE/ASME Trans. Mechatron.* **2015**, *20*, 2486–2495. [CrossRef]
2. Rotella, N.; Mason, S.; Schaal, S.; Righetti, L. Inertial sensor-based humanoid joint state estimation. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 1825–1831. [CrossRef]
3. Asl, R.M.; Hagh, Y.S.; Simani, S.; Handroos, H. Adaptive square-root unscented Kalman filter: An experimental study of hydraulic actuator state estimation. *Mech. Syst. Signal Process.* **2019**, *132*, 670–691.
4. Springmann, J.C.; Cutler, J.W. Attitude-independent magnetometer calibration with time-varying bias. *J. Guid. Control. Dyn.* **2012**, *35*, 1080–1088. [CrossRef]
5. Li, X.; Li, Z. A new calibration method for tri-axial field sensors in strap-down navigation systems. *Meas. Sci. Technol.* **2012**, *23*, 105105. [CrossRef]
6. Fong, W.; Ong, S.; Nee, A. Methods for in-field user calibration of an inertial measurement unit without external equipment. *Meas. Sci. Technol.* **2008**, *19*, 085202. [CrossRef]

7. Garcia-Rodriguez, V.H.; Silva-Ortigoza, R.; Hernandez-Marquez, E.; Garcia-Sanchez, J.R.; Ponce-Silva, M.; Saldana-Gonzalez, G. A DC motor driven by a DC/DC Boost converter-inverter: Modeling and simulation. In *Proceedings—2016 International Conference on Mechatronics, Electronics, and Automotive Engineering (ICMEAE 2016)*; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2016; pp. 78–83. [CrossRef]
8. Pinto, V.H.; Gonçalves, J.; Costa, P. Modeling and Control of a DC Motor Coupled to a Non-Rigid Joint. *Appl. Syst. Innov.* **2020**, *3*, 24. [CrossRef]
9. Kucuk, S.; Bingul, Z. Robot Kinematics: Forward and Inverse Kinematics. In *Industrial Robotics: Theory, Modelling and Control*; Cubero, S., Ed.; INTECH Open Access Publisher: Burghausen, Germany, 2006.
10. Verhaegen, M.; Verdult, V. *Filtering and System Identification, A Least Squares Approach*; Cambridge University Press: Cambridge, UK, 2007.
11. Barfoot, T.D. *State Estimation for Robotics*; Cambridge University Press: Cambridge, UK, 2017.
12. Maged, S.A.; Abouelsoud, A.A.; Bab, A.M.R.F.E.; Namerikawa, T. Stewart Platform Manipulator: State Estimation Using Inertia Sensors and Unscented Kalman Filter. In Proceedings of the 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), Beijing, China, 8–10 July 2016; pp. 1136–1140. [CrossRef]
13. Maged, S.A.; Abouelsoud, A.A.; Fath El Bab, A.M.R.; Namerikawa, T. A comparative study of Extended Kalman Filter and H filtering for state estimation of stewart platform manipulator. In Proceedings of the 2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Tsukuba, Japan, 20–23 September 2016; pp. 1727–1732. [CrossRef]
14. Al-Shabi, M.; Cataford, A.; Gadsden, S.A. Quadrature Kalman filters with applications to robotic manipulators. In Proceedings of the 2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), Ottawa, ON, Canada, 5–7 October 2017; pp. 117–124. [CrossRef]
15. Kok, M.; Hol, J.D.; Schön, T.B. *Using Inertial Sensors for Position and Orientation Estimation*; Now Publishers Inc.: Delft, The Netherlands, 2017; Volume 11, pp. 1–153. [CrossRef]
16. Crassidis, J.L.; Landis Markley, F.; Cheng, Y. Survey of nonlinear attitude estimation methods. *J. Guid. Control. Dyn.* **2007**. [CrossRef]
17. Terejanu, G.A. *Unscented Kalman Filter Tutorial*; University at Buffalo: Buffalo, NY, USA, 2011.