ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Semester Project
Winter Semester 2012

Applying Kalman Filtering on a Quadruped Robot

Lucian Cucu
Microengineering

Supervisors:
Alexander Spröwitz
Alexandre Tuleu

Professor:
Auke Ijspeert

BIOROB
EPFL Biorobotics Laboratory

Contents

Chapter 1

Introduction

The quadruped robot Cheetah-Cub of the BioRob Lab is equipped with sensors that are not yet used for its locomotion. Current locomotion control is using a CPG-based open-loop control. One strategy to exploit the full potential of the robot would be to use sensor data in a closed-loop control to adapt the gait depending on the terrain or the wanted speed. Before doing so however, a guarantee is needed that the sensor outputs reliable information.

The goal of this project is to use Kalman filters to estimate the instantaneous roll and pitch angle, as well as instantaneous forward and upward speed of the center of mass of the robot in trot gait. A 9DOF (3-axis accelerometer, 3-axis gyroscope, 3-axis compass) Inertial Measurement Unit will be used and data processing, as well as filter implementation will be done in Matlab, on off line data.

A motion capture system will be used as reference to evaluate filter performance and to calibrate it.

Chapter 2 explains the basics of the Kalman filter and chapter 3 discusses different methods for rapidly calibrating the sensor. Chapter 4 deals with data fusion and comparison with the reference signal. In this chapter a first Kalman filter fusing accelerometer and gyro data is implemented for orientation (roll and pitch angle) estimation. In chapter 5 a second Kalman filter is designed for acceleration and speed estimation. It fuses g-free acceleration and a very simple locomotion model. Chapter 6 presents the results and a general discussion.

Chapter 2

The Kalman Filter

2.1 Theoretical concepts

Introduced by Swerling (1958) and Kalman (1960), the Kalman filter is a probabilistic tool based on mean squared error minimization for estimation of a state from indirect, stochastic data. It is mainly used when noisy data is available from multiple sources and must be combined to extract relevant information about a particular state of the system. The estimation will tend to be more accurate than any of the data taken individually. In other words, the variance of the results will be smaller than the variance of each data source

Its actual applications are mainly in guidance, navigation and control. It is particularly adequate for cases where multiple measurement sources are available and are correlated in some way.

We hereby propose a short overview of the theoretical concepts which lie beneath the Kalman filter. The key points from [1] are resumed.
The Kalman is an implementation of the Bayes filter, which in probabilistic robotics combines different *beliefs* in order to infer on the actual state of the system. A belief is represented through conditional probability distributions, which at time t has a mean of μ_t and a covariance matrix of Σ_t.

The general algorithm is composed of the following two steps, which are repeated in a loop:

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1} \tag{2.1}$$

$$bel(x_t) = \eta p(z_t \mid x_t)\, \overline{bel}(x_t) \tag{2.2}$$

where x_t is the state at time t, u_t the input at time t and z_t the measurement at time t and η a normalizer in the Bayes' rule.

It is the combination between two beliefs: a prior and a posterior one. In other words, one belief is inferred *before* and the other one *after* the measurement has been taken. The first (2.2) is called an estimation or a *prediction*. It makes an inference on the *probable* state *given* the input u_t and the previous belief $bel(x_{t-1})$. The second (2.2) is a *correction*. It incorporates the current measurement z_t *given* the current state x_t and the current belief $\overline{bel}(x_t$

As mathematical background, we shall just mention that derivation of the Bayes filter is based on the Markov assumption and on the Theorem of total probability (2.3). Described in a very simplistic way, the Markov assumption stipulates that the knowledge of any other state but the previous one

does not give any additional information. It is an approximation often used in robotics. As for the theorem of Total probability:

$$p(x) = \int p(x|y)p(y) \tag{2.3}$$

where $\eta = \frac{1}{p(y)}$ is the normalizer.
By taking a closer look one should notice the similarity between the latter and (2.2). A more detailed mathematical justification is however beyond the range and focus of this work.

Although the Bayes filter is able to combine multiple hypothesis and account for a dynamical system, where the robot perceives its environment through sensors and adapts its controls with respect to it, it still suffers from some major drawbacks. One of them is the difficulty of implement ion in real-time. As can be noticed in equation (2.2), the main problem is the integral over the previous states, which makes the algorithm computationally unadaptable.

The Kalman filter is a more practical algorithm, which implements the Bayes filter for *linear Gaussian systems*. By *linear* it is meant a system of the form:

$$\dot{x}_t = F_t x_t + B_t u_t + q_t$$

where q_t is white gaussian noise of covariance Q and zero mean
The algorithm for one iteration is:

$$\begin{aligned}
\bar{\mu}_t &= A_t \mu_{t-1} + B_t u_t && (2.4)\\
\bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + Q_t && (2.5)\\
K_t &= \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1} && (2.6)\\
\mu_t &= \bar{\mu}_t + K_t(z_t - \bar{z}_t) && (2.7)\\
\Sigma_t &= (I - K_t C_t)\bar{\Sigma}_t && (2.8)\\
& && (2.9)
\end{aligned}$$

return μ_t, Σ_t

with : $\bar{z}_t = C_t \bar{\mu}_t$

It can be decomposed in two major steps. (2.4) and (2.5) represent the *prediction step*, where a prediction of the system state is made based on the previous state estimation *and* the actual input while (2.7),(2.8) represent the *update step* or *innovation*, where the final state estimation is done after incorporating the measurement.

The inputs for this algorithm are μ_{t-1} and Σ_{t-1} ,which are the state vector and the covariance matrix at time $t-1$ respectively. $\bar{\mu}_t$ and $\bar{\Sigma}_t$ are the predicted (*a priori*) values of the state and its covariance matrix at time t, which together represent $\overline{bel}(x_t)$ from (2.2). After incorporation of the measurement at time t, z_t, the algorithm outputs μ_t and Σ_t, ,the corrected (*a posteriori*) state and covariance values for time t.

Q_t and R_t represent the noise affecting the input (or *process*) and the measurement (or *correction*). As this noise is supposed to be white, Gaussian, and independent, Q_t and R_t are diagonal covariance matrices. Their use and importance will be clarified in section 2.3

The key to understanding the algorithm lies within K_t, also known as Kalman gain. When the coefficients of R_t are low, meaning that the input suffers from low noise, the Kalman gain increases, according to (2.6), which means that the confidence in our filter increases. A higher importance would then be granted to any difference between the measured state z_t and the state measurement prediction and would directly affect the next estimation of the state, as seen in (2.7). Likewise, the covariance matrix of the state, Σ_t , would be smaller, which means the accuracy of the estimation is high. And vice versa.

The fact that each estimation μ_t is only based on the previous one μ_{t-1} and on the current measurement and input, makes the Kalman filter a particularly powerful tool for real-time systems.

2.2 Extension of the Kalman filter (EKF)

As mentioned in the precedent section, the Kalman filter is valid only on a *linear* Gaussian system. We however cannot guarantee that the process transition and measurement step could be expressed as a linear system. More generally:

$$x_t = g(u_t, x_{t-1})$$

In this case, a linearization is required, which as usual would be done using the Taylor expansion around the most likely value of the state, i.e its mean. Thus:

$$g(u_t, x_{t-1}) \approx g(u_t, x_t) + G_t(x_{t-1} - \mu_{t-1})$$

with G_t the Jacobian evaluated at u_t, μ_t.

The Extended Kalman filter algorithm yields thus:

$$\begin{aligned}
\bar{\mu}_t &= g(u_t, \mu_{t-1}) && (2.10)\\
\bar{\Sigma}_t &= G_t \Sigma_t G_t^T + Q_t && (2.11)\\
K_t &= \bar{\Sigma} H_t^T (H_t \bar{\Sigma} H_t + R_t)^{-1} && (2.12)\\
\mu_t &= \bar{\mu}_t + K_t(z_t - \bar{z}_t) && (2.13)\\
\Sigma_t &= (I - K_t H_t)\bar{\Sigma}_t && (2.14)\\
& && (2.15)
\end{aligned}$$

$$\text{return } \mu_t, \Sigma_t$$

where $\bar{z}_t = h(\bar{\mu}_t)$

EKF and another more complex extension, named UKF (Uscented Kalman Filter), are the versions of the Kalman filer which are the most used in practice. It has been shown [2] that in experiments similar to ours the EKF and UKF present roughly the same accuracy and that the relative simplicity

of the EKF makes it superior from a computational point of view. The choice of EKF is also done in this work.

2.3 Tuning the Kalman Filter

One of the most import part in designing a Kalman filter lies in its tuning. This constitutes its strength but also one of its major difficulties. Tuning is achieved by carefully selecting the Q_t and R_t covariance matrices stated in the equations above. These matrices represent the covariance matrices of the process (or input), respectively of the measurement noise. As the Kalman filter is a gaussian filter, by its definition (see [1]), we assume that the noise affecting it is unimodal,gaussian and independent. The above-mentioned matrices will therefore look like:

$$Q, R = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix}$$

where σ_i could be for example the measurement noise on each axis (in the case of R).

These coefficients should be high -which implies a bigger covariance- when the confidence in the measurement is low, and vice versa.

Unfortunately, the process or measurement noise are difficult to measure in practice. Measuring the noise from the sensors is often insufficient, as they tend to vary based on exterior factors and influences (e.g temperature, dynamics ,etc.). Moreover, the process and measurement noise might not be independent, nor perfectly gaussian. No canonical form has is yet available for finding these covariance matrices, still it is worth mentioning that one method could be the so called 'autocovariance least-squares ' proposed by [3].

Eventually, evolutionary algorithms like particle swarm optimization (PSO) can also be used [4].

In this work however, we will prefer a more intuitive, heuristical method to adjust these matrices, based on the specific model we are dealing with. Section 4.1 will cover an example of such an intuitive choice.

By proceeding so however, we can obtain satisfactory results even when the process that is estimated is not observable. This is the case in [5], where R is estimated from off-line data and Q is estimated with the help of a simple model. The authors obtain thus a Kalman filter that outruns any traditional one. Unfortunately, the authors only mention their model as being 'simple(poor)' but do not show it, nor the noise covariance matrices.

Chapter 3

Inertial Measurement Unit

The performance of the sensors plays a crucial role in the overall performance of the Kalman filter. It is therefore essential to have a precise idea of the sensor's capabilities and flaws.

As we also mention in the hardware description (section 6.1), the module is composed from two chips: one for the accelerometer and compass and the other for the gyroscope. The manufacturers of the accelerometer chip claim that the device is already factory-calibrated and ready to use [6]. The also mention some trimming values for the linear acceleration sensitivity and for zero-g level acceleration, which are stored in the non-volatile memory and are loaded at the beginning of each use . The trimming values would thus allow "the use of the device without further calibration", according to the data sheet (ibidem). As for the gyroscope chip, it is stated that: "factory-calibrated initial sensitivity reduces production-line calibration requirements [7].

Nevertheless caution is required when dealing with these statements, as the chips underwent also another assembly process, this time when mounted on the Roboard RM-G146 module, which we currently use and refer to as the IMU. Due to this additional step it is highly probable that most fabrication settings mentioned are no longer valid. Furthermore, even if they were still valid, characteristics may vary a lot from device to device. In the case of the gyroscope, the vendor's data mentions $\pm 25°/s$ of initial zero-rate output tolerance (bias) [7], which is unacceptable for our kind of application. These are the reasons why the device has to be recalibrated.

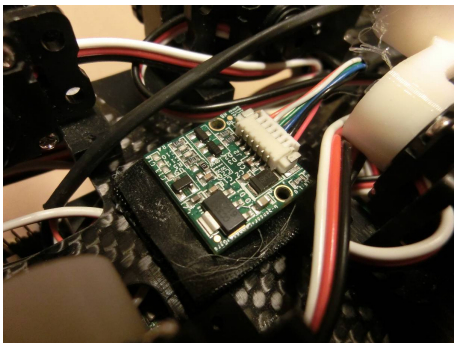


Figure 3.1: *RM-G146 9DOF IMU Module from RoBoard*

Yet a careful calibration of an IMU would require by itself another whole work of the scale of this one. This would include finding:

- biases - constant error factors
- scale factors - error factors depending on the magnitude of the current value
- misalignment - cross-axis error factor, due to misalignment between body and sensor axis, as well as between different sensor axis

as well as more subtle ones like temperature influence, dynamic behavior, non-orthogonality, non-linearity, hysteresis. However the last ones will be neglected, as they are not only very difficult to measure but their impact is considerably lower than from the first three ones [8]. Moreover, most of these parameters are intrinsic to the MEMS' nature. They cannot be corrected and also do not really change a lot from the values given in the data sheets.

For our purpose and desired precision, only some of these factors will be reconsidered.

3.1 Calibration

Before proceeding to using the IMU for other purposes we would like to demonstrates in this chapter the usefulness of spending some time with some easy "in-field" calibration methods. The goal is to stress the difference in results between using the IMU as it is provided by the manufacturer and using it after simple correction has been done. Furthermore, three different techniques will be compared, form an intuitive one to a more subtle but elegant one. We shall deal firstly with the accelerometer, and afterwards with the gyroscope. For different reasons discussed further on (section 4.1), the compass will not be used.

3.1.1 Method A

The first method consists in placing the accelerometer on a surface as horizontal as possible on its different faces and record the results under a given constant acceleration. Obviously, this will be gravity. In total, at least 6 measurements have to be taken, with each of the sensitive axis parallel and anti-parallel to the gravitational force ,as described in figure 3.2. Such a method has also been used in works like [9] and [8], where the bias is simply calculated from:

$$B = \frac{1}{2}(U_{a_+} + U_{a_-})$$

where

$$U_{a_+} = Ag + B$$

$$U_{a_-} = A(-g) + B$$

resulting for U_{a_+} and U_{a_-+} in a 3x3 matrix, where each line represents the output of the accelerometer when x,y and z are parallel respectively anti-parallel to g.

We have proceeded as follows: for each of these six positions 3 different measurements have been taken, each one with the IMU strapped with duck tape in a horizontal/vertical position to the table. The dif-

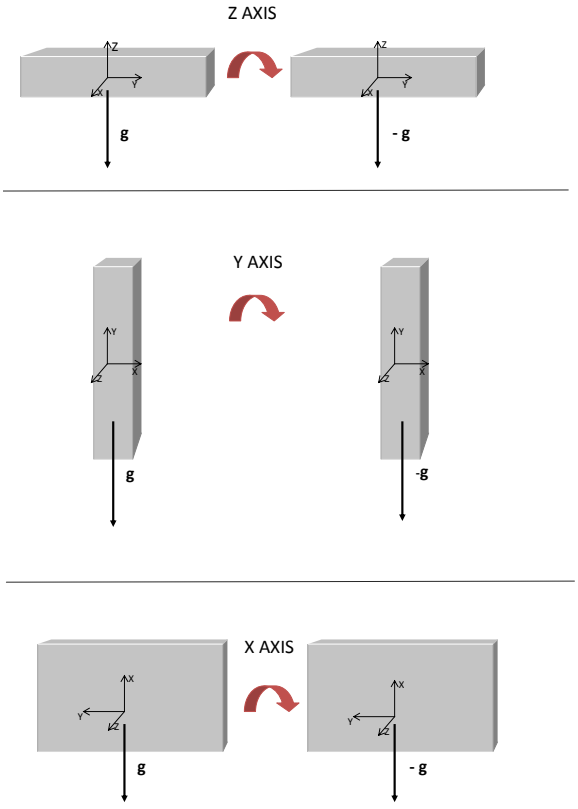


Figure 3.2: *basic principle of Method A: for bias extraction it is sufficient to place the IMU on a horizontal plane with the gravity vector parallel and anti-parallel to each axis*

ference within each set of 3 measurements is the yaw
angle, that changed to 0°, 90°, and 45° respectively.
Theoretically the measurement is supposed to produce the same result, but it is obviously not due must of all to experimental setup errors . An average for each set is done.

Although it would be also easily calculated by hand, the scaling factor is deliberately neglected here. This is done so because calculating it without considering the misalignment factor would be irrelevant (this will be done in the next method). It is also interesting to see what performance can be achieved with such a simplistic method and compare it with more elaborate ones.

3.1.2 Method B

The second method is a standard in-field calibration method, described also in [10]. It consists this time in not neglecting the scale and misalignment factor. This is equivalent to computing:

$$X = A_m \begin{pmatrix} 1/A_{Sx} & 0 & 0 \\ 0 & 1/A_{Sy} & 0 \\ 0 & 0 & 1/A_{Sz} \end{pmatrix} \begin{pmatrix} A_x - B_x \\ A_y - B_y \\ A_z - B_z \end{pmatrix} \tag{3.1}$$

$$= \begin{pmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{pmatrix} \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} + \begin{pmatrix} P_{10} \\ P_{20} \\ P_{30} \end{pmatrix} \tag{3.2}$$

where A_m is the 3x3 alignment matrix, A_{Si} the scaling factor for each axis, $Y = (A_x, A_y, A_z)$ the raw measurements from the accelerometer and $X$the true values of the acceleration, which we suppose of 1g or -1g

This time, all 18 measurements have been used and the least-square method is used for estimating the 12 factors:

$$P = [Y^T Y]^{-1} X$$

with X,Y having 18x3 dimensions and P 4x3

3.1.3 Method C

The third method is the most elegant one. It consists of slowly rotating the IMU on all directions, thus covering as many orientations as possible. It is usually used (as in [6]) for calibrating compasses, which suffer from local interferences of the magnetic field. This interferences are due to soft- and hard-iron effects. The first effect is generated by easily-magnetizable components of the device and result in time varying magnetic field, whereas the latter induces a time-invariant one, which is generated by permanent magnets or magnetized iron or steel in the vicinity.

If one maps all the measurements on a 3D plot, on should be able by least-mean squares to fit a regular geometric form between the points. Ideally, this would be a centered sphere with the radius equivalent to the earth's magnetic field's strength at the given location. However the present above-mentioned effects will tilt, respectively shift the sphere.

This whole concept can also be applied to an accelerometer, except that in this case, the offset would be the accelerometer's bias and the scale factor would be the internal stresses and flaws resulting from construction and assembly. We thus proceed to a slow and smooth rotation of the accelerometer in all directions. Other exterior acceleration other than gravity is assumed insignificant. The results after approx. 80 seconds sampling are shown in figure 3.3

Now, the general equation of a shifted and tilted ellipsoid is:

$$\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} + \frac{(z-z_0)^2}{c^2} + \frac{(x-x_0)(y-y_0)}{d^2} + \frac{(x-x_0)(z-z_0)}{e^2} + \frac{(y-y_0)(z-z_0)}{f^2} = R^2$$

where:

- x_0, y_0, z_0 are the biases
- x,y,z the raw data
- a,b,c semi-axes lengths
- d,e,f cross-axis effect, which tilts the ellipsoid
- the norm of R the expected value (in our case 1g)

These factors are than estimated through the least-mean square method. We then get the fitted ellipsoid from figure 3.4

The same concept, with however significant mathematical improvements, has been developed in [11]. The authors mention it can be used on any type of three-axis sensor using a sensor field.

One starts with the general equation of an ellipsoid,

$$(\vec{x} - \vec{b})^T A(\vec{x} - \vec{b}) = c$$

with A a symmetric positive-defined matrix and minimizes the error in the least-mean square sense. One of the subtleties lies in how A is defined with respect to the values gathered from the accelerometer and how D, the misalignment matrix is extracted afterwards. However, should the reader want to understand the technicals details of this method, we kindly direct him to the associated paper [11].

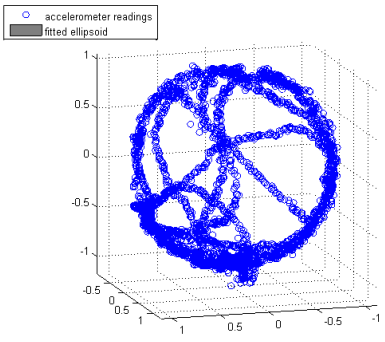


Figure 3.3: *trace of the gravity vector after a slow and smooth rotation in all directions*

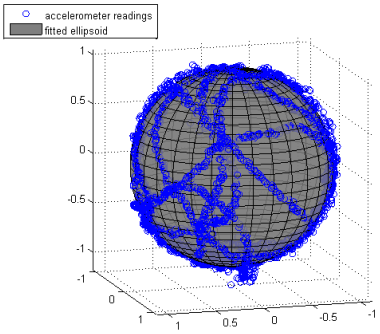


Figure 3.4: *fitted ellipsoid on the gathered data*

The Matlab function used to obtain the results is an implementation of [11] and has been provided by Adrien Briod. The only requirement for the function to work is an appropriate data set and the only precaution that has to be taken is to avoid as much as possible any additional acceleration and to cover all the possible directions, including possible singularity points. When applied, the result has is expected to resemble as much as possible to a sphere. Figures 3.3 and 3.4 show an application with the dataset found previously.

3.2 Comparing Results

The main goal of this project is to have an estimate of the robot's instantaneous speed and orientation. As the only sensor we have available is the IMU, this will inevitably have to be done by integrating the acceleration and the angular speed. In consequence it is crucial to have an idea of the respective drifts in a static case, as integrating a static error results in a very strong and rapid deviation.

In order to have an idea of the performance of each method we show in the following figures how the corrected data behaves in comparison with the non-corrected one. The same set of 18 measurements is used as for the previous methods, but this time the g vector is manually subtracted from the specific axis. For the last calibration one carefully-taken measure is used (the one illustrated in figure 3.4). In figure 3.5 the accelerometer's bias before and after each calibration method is shown. Table 3.1 quantifies the data shown in the figure 3.5.

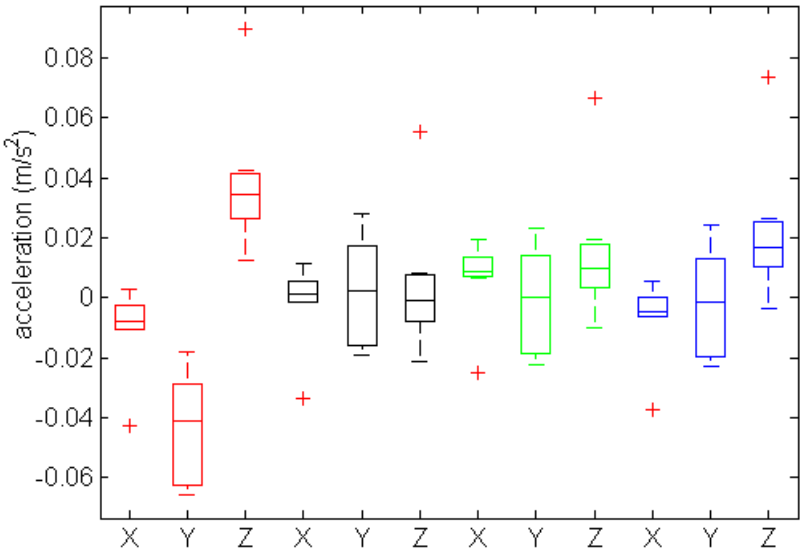


Figure 3.5: *remaining bias on each axis after calibration with the three different methods: raw data (red), method A (black), B (green) and C (blue)*

Despite relative difference between methods, a substantial improvement is observed in each case. Method A seems to perform even a bit better than method C, but application on real runs shows that both methods will differ very little in the final output (less than 4%). The last method will be privileged, because of its flexibility and very comfortable repeatability. Thus, the accelerometer could easily be calibrated before each set of experiments.

	No Method	Method A	Method B	Method C
X axis	-0.086 ± 0.113	0.0001 ± 0.111	0.077 ± 0.110	-0.053 ± 0.106
Y axis	-0.427 ± 0.178	0.020 ± 0.171	-0.009 ± 0.169	-0.018 ± 0.172
Z axis	0.359 ± 0.187	0.021 ± 0.189	0.129 ± 0.188	0.198 ± 0.189
		in m/s^2		

Table 3.1: *performance evaluation of each calibration method*

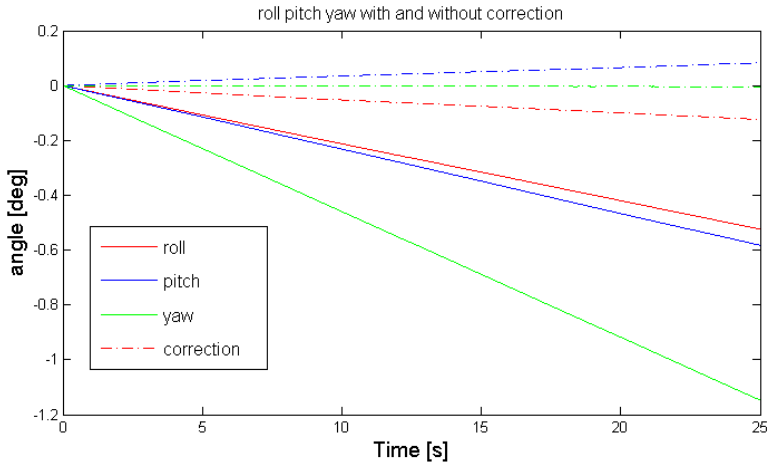


Figure 3.6: *qualitative results of gyroscope bias correction: integration over 25 seconds of the bias on each axis before (continuous line) and after (dashed line) correction*

The initial biases do not exceed $\pm 0.04 m/s^2$ which means $\pm 0.1\%$of the entire range, for a range of $\pm 1g$ that we use. According to [8] typical accelerometer biases vary between $\pm 0.02\%$ and ± 0.1 of the full scale. Our inital parameters are thus acceptable but still remain the worse that one could expect from such a device.

As for the gyroscope, only biases can be extracted. There is no easy way for computing the scale factors, as it would require being able to rotate the gyroscope on a horizontal plate with known and precise angle velocity. Yet, a very affordable method is presented in [12], using an accelerometer and a bike wheel, but we will leave this for a future work.

No particular caution is required when recording the gyroscope's bias, except that it has to stand still during the sampling. The offset from the origin shown in the data will be the bias. 11 experiments have been conducted with the IMU standing still. Each one lasts about 20 seconds and has been taken at intervals of about 1 minute. The mean value is chosen as reference bias for the rest of the experiments. Quantitative results are shown in figure 3.7. In figure 3.6 this mean bias is subtracted from the output on a measurement with the IMU standing still in a random position. In this case, the bias stays below $0.01°/s$.

Figure 3.7 also illustrates very well the bias drift.The recorded bias can variate for as much as the double without any change in the environment. In this case, the variation spreads as much as $0.6°/s$ from maximum to minimum. The full scale ($\pm 2000°/s$) is used here for measuring. Lowering the scale has a considerable impact on the bias drift: with a scale of $\pm 500°/s$ the drift is contained within a $0.15°$ range.

To conclude with, the IMU cannot be used without a minimum of calibration. Simple in-field methods are described and compared for the acceleretometer. The output is corrected, but only partially. A simplistic method is used for gyroscope bias correction. It is effective but still faulty:

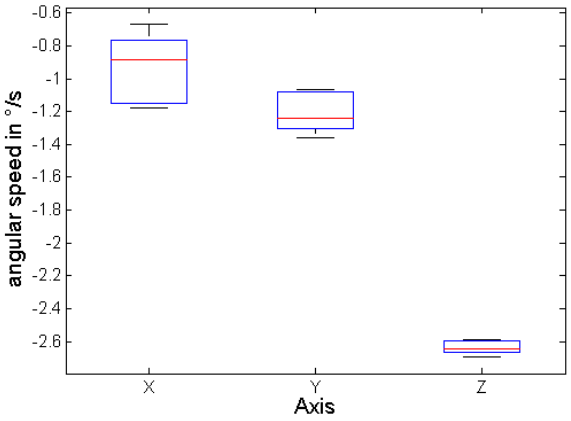


Figure 3.7: *bias variation for different samples with the IMU standing in the same position at constant temperature*

remaining bias and other error factors (such as scale factor, drift) still affect significantly the output. So after some time spent on calibration, we expect the IMU to be able to give acceptable results at least for low-precision applications.

Chapter 4

Ground truth and data comparison

We hereby remind again that the main goal of this project is to estimate the instantaneous forward speed, pitch and roll angle of the robot while trotting. In order to evaluate the performance of our future Kalman filter, it is mandatory that we have some kind of comparison or reference point. We have decided to use the motion capture system (MoCap) available in the lab as ground truth.

Our first concern will therefore be providing an adequate comparison setup between our two sources. Two major issues have to be handled: compensate gravity for the IMU and find the relative rotation between the MoCap frame and the IMU frame.

For the first one, the bearing of the device has to be estimated and for the second, the adequate rotation matrix has to be found. This will be handled in the next two following sections. Once we have this, it is expected that the recorded signals look alike.

4.1 Kalman filter for data fusion

As our IMU has 9 axis, it holds theoretically all the necessary information for an accurate bearing evaluation. After a simple vector decomposition, the accelerometer gives a first estimate of the pitch and roll angles, which is certified by the gyroscope when the device is moving. As for the compass, it can give the missing yaw angle, if the roll and pitch angle from the accelerometer are combined with the measured Earth's magnetic field vector.

The only thing we need is a method to fuse this data. The Kalman filter has been proved to be the most adequate tool for such a task, and is extensively covered in literature (for instance [13] , [14]). As this method has indisputable results, we shall use it also in our case.

Still, one exception is to be made. We decide not to use all the available data and leave aside the compass. The reasons are:

- some issues occur while using the magnetometer on our chip; sometimes, unexpected field disturbances provoke an overflow of the measurement data, thus making it unreliable
- calibration will be needed again

- relative or absolute yaw angle are irrelevant for our task, as we do not use a flying robot, nor do we steer it

- The additional complexity of a Kalman filter taking into account the yaw corrections is not in acceptable proportion with the possible additional precision

Despite this simplification and under certain conditions, the results can be satisfactory.

The current Kalman is quaternion-based and is similar to the one described in [13]. The Matlab function and design of the filter have been kindly provided by Adrien Briod. Minor changes have been made to adapt it to our device. We shall not go into details here but only comment the main points. A detailed explanation can be found in [13].

As expected, the Extended Kalman filter algorithm described in 2.2 is used. The state vector $\mathbf{x}$ represents the set of angles and biases. Quaternions are used instead of Euler angles, because of their mathematical elegance and their lack of singularities. This yields:

$$x^T = (q_0, q_1, q_2, q_3, \delta_x, \delta_y, \delta_z)$$

consisting of the normalized quaternion set $\vec{q} = (q_0, q_1, q_2, q_3)$ and $\delta = (\delta_x, \delta_y, \delta_z)$ the gyroscope biases on each axis found in section 3.2.

We briefly remind that $q_0 = cos(\frac{\theta}{2})$ and $(q_1, q_2, q_3)^T = sin(\frac{\theta}{2})(x, y, z)^T$ where θ is the angle angle and $(x, y, z)^T$ the axis of the wanted rotation. After rotation however, the quaternions do not correspond to an unitary vector anymore, so a normalization step is required at each iteration of the Kalman algorithm. A higher computational cost is the only major drawback of using quaternions instead of Euler angles.

Through differentiation we can link angular velocities to quaternion derivates:

$$\dot{q} = \frac{1}{2} Q \cdot q$$

where

$$Q = \begin{pmatrix} 0 & -(\omega_x + \delta_x) & -(\omega_y + \delta_y) & -(\omega_z + \delta_z) \\ \omega_x + \delta_x & 0 & \omega_z + \delta_z & -(\omega_z + \delta_z) \\ \omega_y + \delta_y & -(\omega_z + \delta_z) & 0 & \omega_x + \delta_x \\ \omega_z + \delta_z & \omega_y + \delta_y & -(\omega_x + \delta_x) & 0 \end{pmatrix}$$

The question now is, which sensor to use in the prediction step and which one in the correction step?
The gyroscope is reliable on the short term but tends to drift heavily with time. The accelerometer on the other hand, is less accurate to detect sudden changes because undesired accelerations mix up with gravity in an unpredictable way. Nevertheless it is very reliable on the long term, as it will tend to precisely indicate the g vector no matter how much time has passed (we assume that the parasite accelerations are never constant). We decide to use the gyroscope data for the prediction step and the accelerometer data for the correction step.

With the equation stated above, prediction step is:

$$\dot{\mathbf{x}} = g(\mathbf{x}) = \begin{pmatrix} \frac{1}{2} Q \cdot q \\ \mathbf{0} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} -q_1(\omega_x + \delta_x) - q_2(\omega_y + \delta_y) - q_3(\omega_z + \delta_z) \\ -q_0(\omega_x + \delta_x) - q_3(\omega_y + \delta_y) - q_2(\omega_z + \delta_z) \\ -q_3(\omega_x + \delta_x) - q_0(\omega_y + \delta_y) - q_1(\omega_z + \delta_z) \\ -q_2(\omega_x + \delta_x) - q_1(\omega_y + \delta_y) - q_0(\omega_z + \delta_z) \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

which in discretized and linearized form yields:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + g(\mathbf{x}_t)\Delta_t = \tilde{g}(\mathbf{x}_t) = \tilde{G}\mathbf{x}_t$$

with $\tilde{G}$ the Jacobian of $\tilde{g}$. There is no interest of explicitly showing it again here

As for the Update step:

$$\mathbf{z} = (a_x, a_y, a_z)^T$$

is the measurement from the accelerometer and

$$\bar{\mathbf{z}} = h(\bar{\mathbf{x}})$$

is the *predicted* measurement, as a function of the predicted state $\mathbf{x}$ with $h(\bar{\mathbf{x}})$ such that

$$h(\bar{\mathbf{x}}) = R_{transf}^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix}$$

where R_{transf} is the direct cosine matrix (DCM) expressed in quaternions, which expresses the global frame with respect to the local one.

We then derivate and find the Jacobian H, and continue with the other steps described in the Extended Kalman Filter.

The process covariance matrix Q will not be shown here as it has still not proved 100% correct. As for the measurement covariance matrix R, there is no theoretical model to describe it, so it is selected intuitively. The principle used is a simplification of the one presented in [9], where a compass is also used. It is reproduced here, on one hand to better understand the concept of intuitive Kalman filter tuning, and on the other hand to give credit to its author for his idea:

$$R = \begin{pmatrix} R_{var} & 0 & 0 \\ 0 & R_{var} & 0 \\ 0 & 0 & R_{var} \end{pmatrix}$$

with

$$\begin{aligned} R_{var} &= k_0 + k_1 R_{var,1} + k_2 R_{var,2} \\ R_{var,1} &= \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \\ R_{var,2} &= |g - g_{estimated}| \end{aligned}$$

Where $g_{estimated}$ is the norm of the acceleration sampled by the accelerometer. In other words, the covariance will grow as the rotation rate of the object grows, anticipating higher noise. Likewise, the bigger the difference between the g and $g_{estimated}$, the worse is the gravity compensation and the more we expect uncertainties due to additional accelerations. The whole is combined linearly with the k_i coefficients.

4.2 Relative Rotation Matrix

Now that we have two sets of Angles, we can find the rotation matrix mapping a vector from the MoCap frame to the IMU frame. The function which does this, takes two sets of Euler Angles as input and outputs the DCM (Direct Cosine Matrix). Or in other words, it gives the rotation matrix between MoCap and IMU frame. It was written by Adrien Briod.

The function implements the following main steps:

- transform the each Euler angle in a DCM
- project gravity vector on each DCM at each sample; two correlated signals are thus obtained
- use least-mean square method to find the transform that best characterize the mapping from one signal to the other; (ideally, the same acceleration signal should be seen from both sources, the only difference consisting in an offset)
- apply polar decomposition to select an orthogonal matrix (i.e rotation matrix) from the transform; this will provide the matrix we are looking for
- plot data as seen from the IMU and the one seen by the MoCap in the IMU frame

An illustration of the effect of this relative rotation matrix is shown later on in chapter 6, figure 6.6.

It is important to stress that this matrix does not indicate *the real* rotation between the two frames, but only the one which fits the best one signals to the other in a leat-mean square sense. So in order to make sense, the two signals have to be very similar, and have a minimum of offset. If this is not the case, the the procedure would be pointless and would output an irrelevant matrix.

Chapter 5

Kalman filter on Cheetah robot

In order to have a proper speed estimation it is first of all compulsory to be sure that the measured acceleration is as close as possible to the real one.

However, vibrations and high-frequency noise affect the acceleration measurement, so it is necessary to figure out a way to filter them as precisely as possible. The most instinctive method would be of course to apply a low-pass filter. Still, a perfect cut-off at the desired frequency is impossible (due to limited bandwidth, ripple effects, etc.). Moreover, any low-pass filter introduces a phase-shift depending on the frequency and filter order, which requires additional caution when handling with real-time estimations

Besides a classical low-pass filter, a median filter and a Kalman filter are also considered.

In [5] the authors demonstrate that a Kalman filter is more efficient than the traditional ones, if a process model is at hand. In this work, the authors filter data from an accelerometer with four filters respectively: Butterworth low-pass filter, median filter, discrete wavelet shrinkage and Kalman filter. The accelerometer is strapped on a human leg and samples data during a normal walk.

When it comes to tuning the Kalman filter, they state that " a relatively simple (poor) model can provide acceptable results via the selection of Q" [5] . They however do not mention this "simple" model, but we suppose that it has to be based on the human movement, its symmetry and some basic physical assumptions

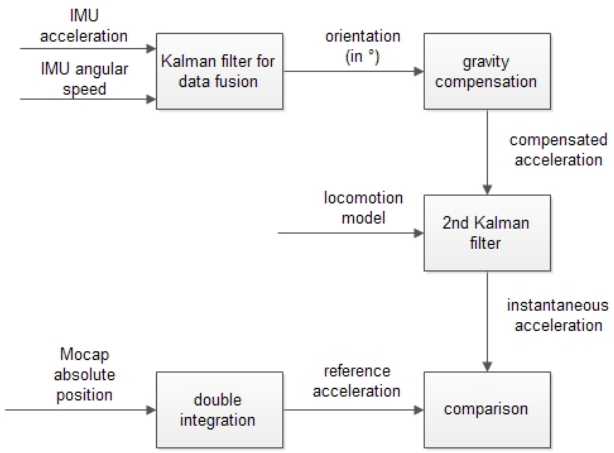


Figure 5.1: *general system setup with two stage Kalman filtering; 1rst Kalman filter estimates roll and pitch angle from IMU data, these are used for extracting gravity from the acceleration data and a 2nd Kalman uses the corrected acceleration and the locomotion model to estimate the instantaneous acceleration of the robot*

According to their results, the Kalman filter reaches better performance than the others, when the Q and R matrix are carefully tuned. We therefore assume that this will also be the case when estimating instantaneous acceleration on the Cheetah, if we are able to find a suitable physical model.

Figure 5.1 illustrates where such a locomotion model would fit in the overall process.

5.1 Locomotion model

Given our system, the most suitable physical model to describe it would of course be the locomotion model.

Extensive research on the topic has been conducted by Raibert et al. in [15], [16] and [17]. By studying symmetry in animal gaits he has showed that one can significantly simplify the locomotion model and obtain acceptable results when implementing on running robots. Particular attention has been put on gaits that require pairwise symmetry between legs, like trot, gallop and pace. In these cases, it can been considered that two legs could be replaced by only one virtual leg, placed at the center of mass as indicated on Figure 5.2.

In our case, the Cheetah is trotting, which means that the diagonal legs move identically, as shown in 5.3.

Raibert further assumes that in trotting gait the pitch angle of the robot (rotation around the axis perpendicular to the forward direction) is null, which means that the body is constantly parallel to the ground.

Although in reality, a ASLIP- advanced spring-loaded inverted pendulum - model is used for the legs [18], we will simply assume that our legs are simple guided springs. That will allow us to assume a sinusoidal trajectory similar to the one from figure 5.4, where the central point is characterized by zero speed on the vertical axis and zero acceleration.

The acceleration would hence also be similar to a sine wave.

As we know that the CPG moves each hip motor with a 2.5 Hz frequency, we shall assume that the acceleration pattern has a frequency of 5 Hz. The double frequency is due to the fact that we assumed two virtual legs at the center of mass, as Raibert's model suggests. The amplitude will be chosen in accordance with the data provided by the MoCap (as a simple average of the recorded peaks), and the phase shift adjusted by hand.

In conclusion, a simplistic sine wave is chosen as locomotion model and input for the Kalman filter. When compared with the MoCap data, the signal would look as illustrated in figure 5.5 . We will show that this crude and primitive model is nevertheless sufficient to outperform the classical filters.

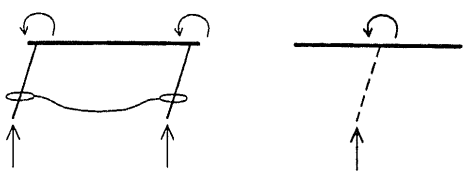


Figure 5.2: *equivalence between two legs moving identically and a* virtual *leg - picture from [15]*

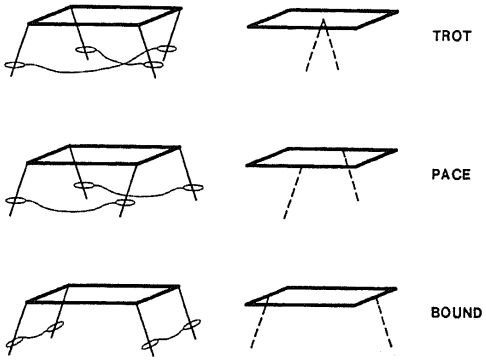


Figure 5.3: *illustration of pairwise leg symmetry - picture from [15]*

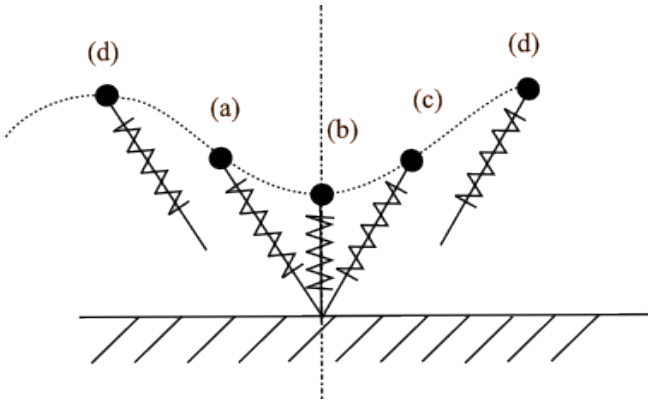


Figure 5.4: *the main phases of leg movement: (a) the foot has touched the ground and compresses the spring, the acceleration is backwards (b) the CoM is exactly above the foot, the spring is loaded , the vertical speed as well as the acceleration are null, (c) spring is released, the acceleration is forward, the leg takes off the ground and eventually we return in the phase (d)*

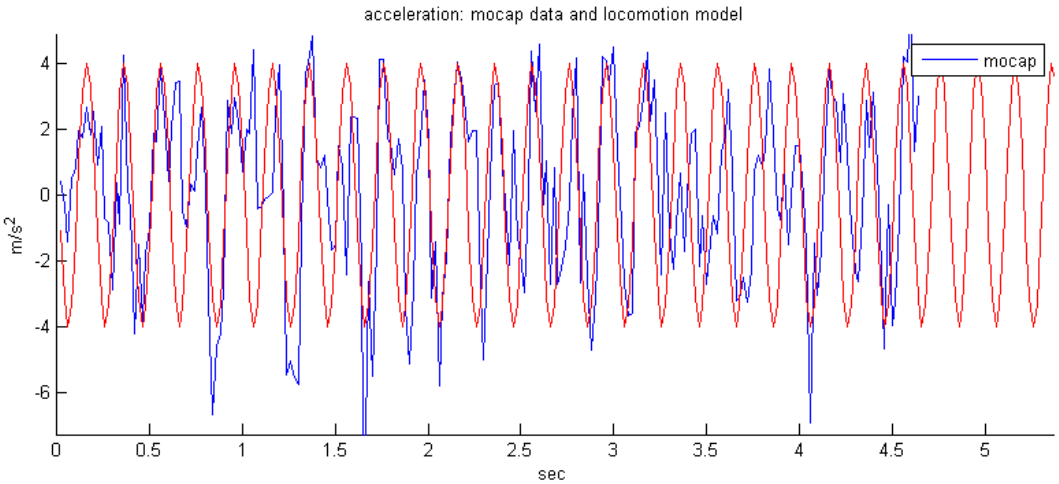


Figure 5.5: *locomotion model (red) superposed over the raw data (blue); with frequency double of the one of the gait and amplitude as average of the peaks; the evident similarities validate the choice of the model*

Chapter 6

Approach and Results

6.1 Hardware and Experimental setup

Cheetah-cub Currently the fastest robot for Froude number and body-length per frame for free trot on a flat surface [18], Cheetah-cub is a quadruped robot, equipped with pantograph legs. The robot uses RC servos motors for hip and knee actuation and with a CPG control is thus able to produce different gaits [18]. The robot is equipped with IR makers, allowing detection from the MoCap system (figure 6.1).

Motion Capture Motion data has been recorded by the motion capture system (MoCap) available in the lab. It uses infrared markers and computes their absolute position based on the time-of-flight principle. Based on the position it then computes the Euler angles in the zyx order. Before recording, a ground plane reference is set and the cameras are calibrated . The system (Optitrack s250e, Naturalpoint, Inc. 2011) is mounted as described in [18]. The data is sampled at 250 Hz and the precision is about 1mm. The data is than manually cleaned in Arena software (ibidem), exported into .bvh format and than low-pass filtered with a cut-off frequency of 18Hz in Matlab. Caution is required when extracting the MoCap data, as the MoCap computes the Euler angles based on a xyz sequence, whereas we need a zyx one (with z -yaw, y-pitch and x-roll).

Inertial Measurement Unit The IMU is a 9DOF Module (RM-G146,Roboard), which is assembled from a LSM303DLH (accelerometer and compass) and a MPU-3050 chip (gyroscope). A

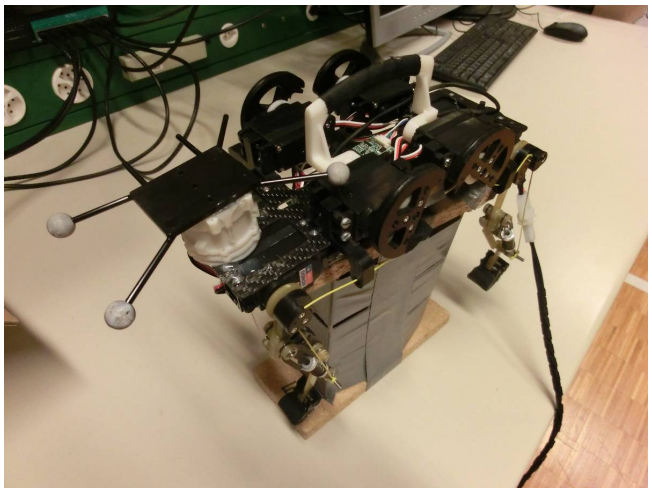


Figure 6.1: *Cheetah-cub robot with mounted IMU near the CoM and IR markers*

temperature sensor is also included. It measures 30x20mm and communicates via I2C. The driver for data extraction has been written by A. Tuleu. The data is sampled at 50Hz, output in a .txt file and then further processed in Matlab. The sensor is strapped as close as possible to the CoM, (on the back of the robot, at its center) with Velcro tape.

Calibration For the accelerometer calibration Method C (ellipsoid method) is used, for the reasons mentioned in section 3.1. IMU data sampling is started and the robot is than slowly rotated in all possible directions. The movements have to be as smooth and as continuous as possible, in order to avoid any other additional acceleration except gravity. 9 such calibration files have been made and the best (i.e. the one which best covered a sphere) was chosen.

For finding the relative rotation matrix between the MoCap and the IMU frame a similar procedure will be used as the one described above. This time the MoCap is recording and the sensor is rotated within smaller angles, thus avoiding gimbal lock regions (keep magnitude of pitch angle approx. below 80°).Hence, singularities in the Euler angles are avoided. Data from MoCap is manually offset to correspond with the one from the IMU. Again 9 rotations are done, and the one where both signals fit the best was picked.

Runs The robot has been run on a linear trajectory, whose starting and ending point are entirely covered by the MoCap system. Before beginning to gather accelerometer data form the IMU and MoCap, the first step of CPG controller is started, where the legs return in the home position and stay stiff. The robot is able to stand on the start position without additional support. The accelerometer, MoCap and run gait are then respectively started and a second person follows the robot and holds the supply cable, while avoiding as much as possible pulling on it. The same starting point and initial orientation are kept as much as possible for each run, and this is repeated 6 times. Trot gait at 2.5 Hz has been used. All the runs use the confC_speed_1.3d.cpg file.

In total, 6 straight runs with the Cheetah-Cub, 9 in-the-air runs for sensor calibration and 9 in-the-air runs for MoCap relative calibration have been conducted. Table 6.1 illustrates a summary of these conducted runs and their type.

type	description	# of runs	duration
calib_run	rotation of the device in all possible directions - used for accelerometer calibration	9	80 s
orient_run	rotation of the device within smaller angles (< 80° pitch) with MoCap ON - used for finding the relative rotation matrix	9	80 s
cheetah_run	runnning the chetah in a forward line with MoCap ON - trot at 2.5 Hz	6	6 s

Table 6.1: *summary of conducted runs*

6.2 Methodology

In the precedent chapters we have described the theoretical background of our approach. In this section, we talk about how we used these tools and give an overview of the different steps. All of them

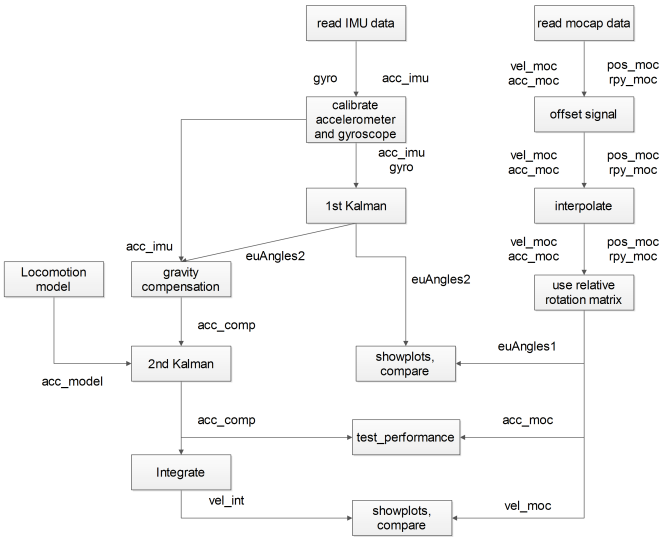


Figure 6.2: *overall process logic*

are implemented in Matlab.

First of all some sensor calibration runs (type calib_run) are conducted, as described in section 6.1. The best one is picked to calibrate the accelerometer using the ellipsoid method (section 3.1.3). Then some MoCap-IMU relative calibration runs (type orient_run) are performed to find the relative rotation matrix (section 4.2). The latter is used for the rest of our experiment.

The straight runs with the trotting Cheetah are then conducted.

For each run, the data from the MoCap is offset manually in order to correspond with the one from the IMU. As the sampling rates are different, it is necessary to interpolate the data to correspond to the sampling rate of the IMU (from 250 to 50HZ). Finally, we apply the relative orientation matrix and compare the results for roll and pitch. We plot those seen by the IMU alongside those seen by the MoCap in the IMU frame (example shown later on in figure 6.6).

Subsequently, the orientation found is used to subtract the gravity vector from the sampled acceleration. We estimate it with the second Kalman filter (designed in chapter 5) and the results are compared with the ones given by other filters.

The whole process is illustrated in figure 6.2 with the respective Matlab variables. This plot is not a dependency plot, but it only illustrates the logic in the Matlab structure as well as the overall process sequence.

6.3 Results and Comments

6.3.1 Pitch and Roll estimation

First of all, we have to question ourselves about the relevance and usefulness of a Kalman filter for the bearing estimation. Could not the integrated data from the gyroscope do the job?

In chapter 3 the imperfections of the device and some rapid ways to partially compensate them are discussed. Nevertheless, the remaining error is still an issue.

In the case orientation estimation, gyroscope bias plays an essential role. To illustrate how the it

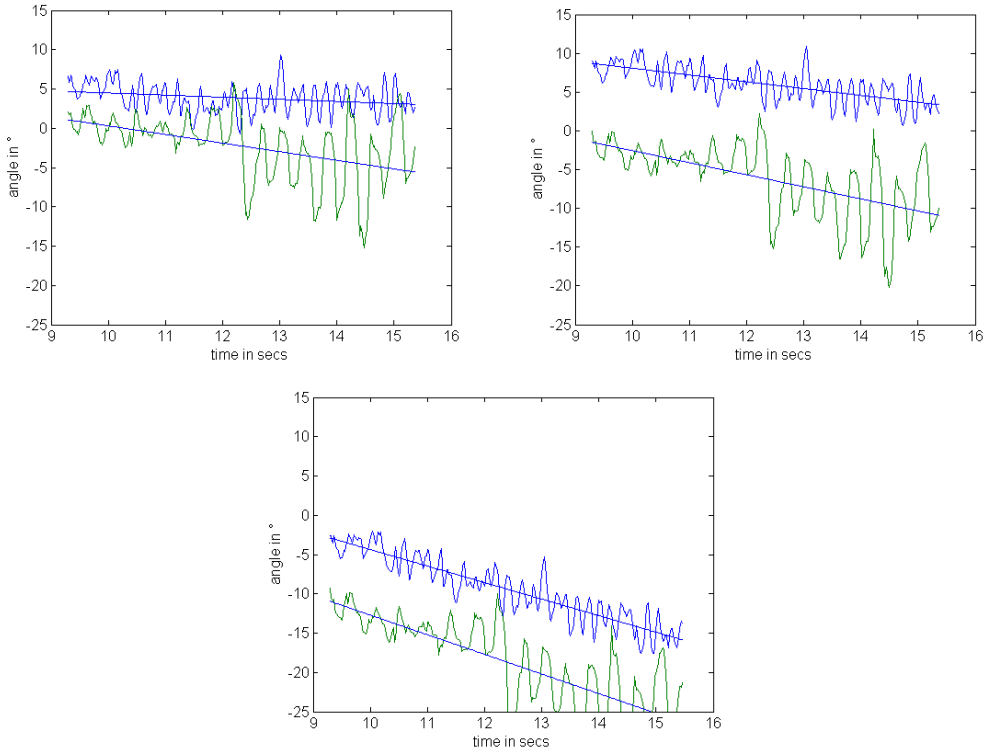


Figure 6.3: *benefits of Kalman and bias correction; Kalman filter with bias correction(left), only bias correction(right) and no bias correction at all (down); the tendency to drift is minimum when Kalman filter is applied, as the accelerometer also corrects the drift - illustration on a cheetah_run*

affects the measurement, we have drawn a least-square approximation line. The bigger the slope of the line, the more is the measurement affected by the bias. (The reciprocal is not necessarily true, as a perfectly horizontal line could also be the result of auto-compensating effects).In figure 6.3 three situations have been considered: in the first one the Kalman filter from section 3.1.1 is used with bias correction (identical to method A from section 3.1.1), in the second the angles are simply integrated from bias corrected IMU data and in the last one, the angle integration is done from raw IMU data, without any sort of correction

Numerical results are shown in table 6.2.

	method	slope on x axis [°/s]	slope on y axis [°/s]
cheetah_run1	Kalman with bias correction	-0.2416	-1.0176
	only bias correction	-0.8584	-1.4912
	no bias correction	-2.117	-2.5302
cheetah_run2	Kalman with bias correction	0.2802	-0.0563
	only bias correction	0.1834	-0.4348
	no bias correction	-1.1194	-1.3648
cheetah_run3	Kalman with bias correction	-0.252	-0.8612
	bias correction	-0.969	-2.0452
	no bias correction	-2.1727	-3.2272

Table 6.2: *tendency to drift in three different cases; the lower the slope, the lower the drift tendency - illustrated on 3 cheetah_runs*

We notice that besides one exception, the slope is always smaller in the case of Kalman filtering with bias correction, so it has a lesser tendency to drift. However the difference between applying bias

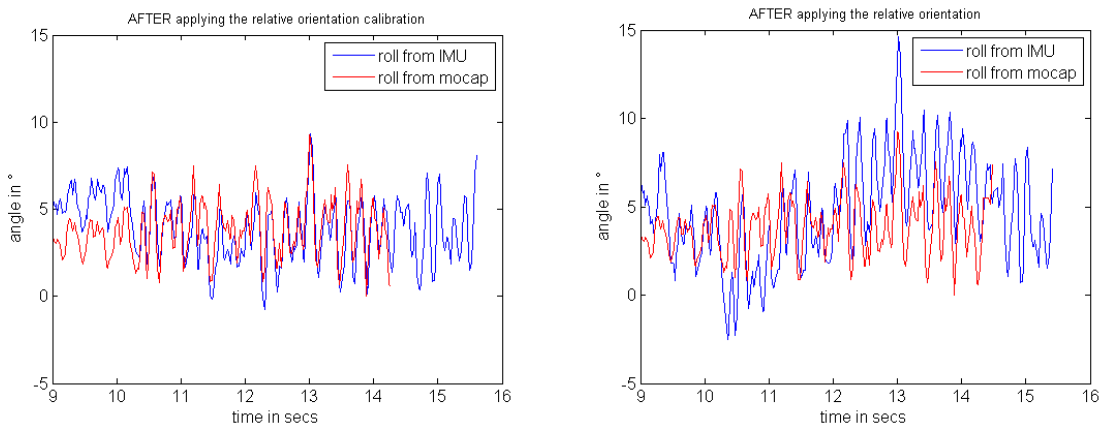


Figure 6.4: *Kalman results for a Q = 0.5 (left) and for Q=20 (right) compared with the MoCap data - illsutration on cheetah_run1*

correction or not is even more remarkable, especially given the very small effort that it requires. As for the Kalman, besides the time required to design and implement it, one has to also carefully tune it.

The Kalman filter can be tuned in a ways such that it gives more or less importance to our data sources. This is a remarkable advantage when dealing with uncertain hardware. Yet this is a quite tedious task and we have decided not to show here anything else from this process except the very minimum.Figure 6.4 shows the effects of tuning Q, the process covariance matrix. When Q is big, we express less confidence in the gyroscope. A similar approach is then conducted for R, which is has more coefficients.

We point out that in our case trusting the gyroscope yields better results. This not surprising, given the duration of the run. For this reason, we decide for further runs to keep low coefficients for Q. Of course, doing so increases also the importance we give to its drift. A similary approach is done also for R, but it is not shown here, as it is a bit more tedious (as explained in section 4.1, there are 3 coefficients to tune)

Secondly, before proceeding to the runs, the performance of the filter is evaluated in 'ideal' conditions. Finding the relative rotation matrix is a procedure where slow, smooth and large amplitude rotations are done. If the results are not satisfactory here, it would be pointless to go on.

As in [9], we will use the RMSE to compute the error between IMU and MoCap angles. We obtain 1.77°, 2.27° and 5.62° overall error for the roll, pitch and yaw angle respectively. After the dynamics occur, instantaneous error peaks can reach up to 11.6° and 15.3° for roll and pitch respectively (figure 6.5). The yaw angle put aside, the overall error is comparable to the one obtained in [9], where for a very similar experiment 1.2° and 1.13° rms Euler angle error is found. However, despite similar manoeuvres the maximal error peaks stay within 3.4°, which is far better than our results. This is not surprising, knowing that in [9] not only the compass is also used for data fusion, but the calibration procedure is more careful, taking into account misalignment and non orthogonality factors for both accelerometer and gyroscope, which is roughly 4 times more than with Kalman filtering.

In figure 6.5 the orientation estimation using only the gyroscope are also shown. The difference is evident, especially after the rotation movement, where the instantaneous error easily exceeds 20°. Overall RMS error between the MoCap and the gyroscope alone is about 6.4° and 7.5° for roll and

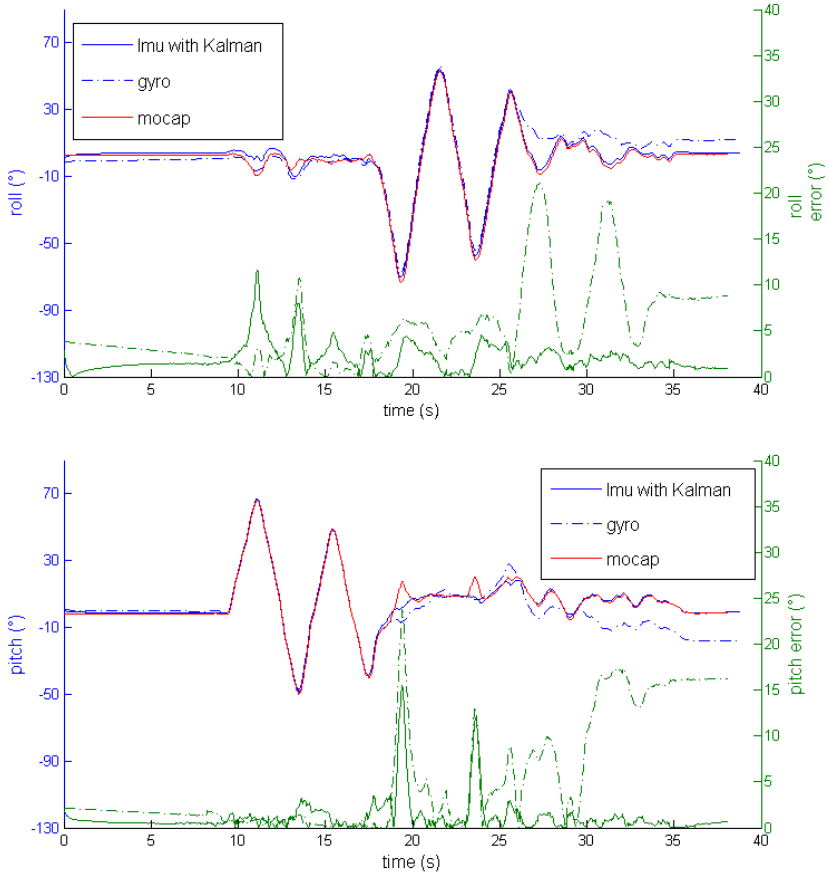


Figure 6.5: *roll angle (up) and pitch angle(down) comparison between IMU (blue line), MoCap (red line) and gyroscope alone (dashed line); instantaneous RMS error is indicated for both IMU and gyroscope with respect to the MoCap (green line); for wide movements, Kalman estimation is faithful to reality; overall RMS error for the gyroscope alone is 4 times greater than the one of the Kalman and continues on increasing significantly after the manoeuvres are done - illustrated on orient_run*

pitch respectively.

Thirdly, an qualitative evaluation of a cheetah_run, as well as the effect of using the relative orientation transform from section 4.2 is presented. For visibility reasons, pitch and roll angles are represented separately in 4.2. As expected, the transformation offsets the MoCap data in such a way that it best fits the imu data (6.6). Still, as these signals are not identical, they cannot be fit exactly.

The point of doing this operation is not to figure out how we could best fit one data with another, but to realise the difference between the reference signal and the signal from the IMU.

Despite the offset and the effect of a slight drift the peaks seem to oscillate more or less around the same value (i.e. with an amplitude of about $\pm 3°$ $\pm 5°$ for roll and pitch respectively).

6.3.2 Acceleration Estimation

We shall now focus on the results obtained from the 2nd Kalman filter, which combines acceleration input with locomotion model. Our evaluation method will be similar to the one used in [5], where the authors use SNR(signal-to-noise-ratio) and correlation to evaluate the different filters. The greater these values, the better the filter. In addition to that we shall also use the RMSE (root mean squared error), which naturally has to be minimized. We also use for comparison a Butterworth low-pass filter (10th order, 6Hz cut-off frequency) and a Median filter (5th order).

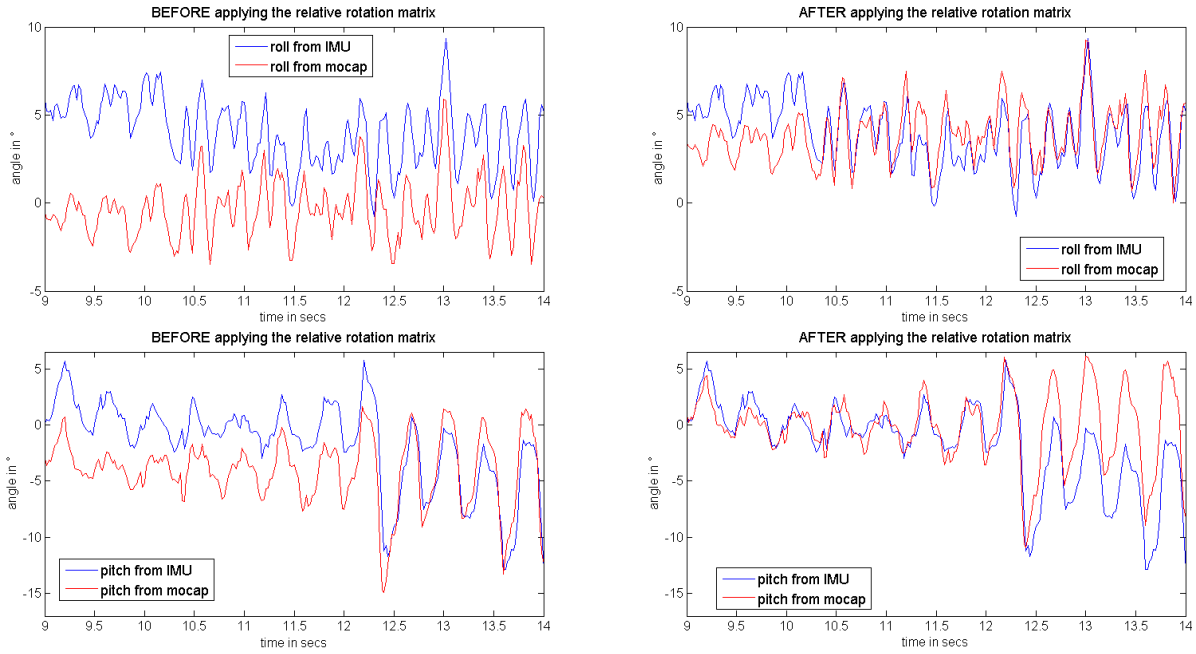


Figure 6.6: *BEFORE(left) and AFTER (right) applying the relative rotation matrix for roll (up) and pitch angle (bottom). After in-the-air run orient_run, the rotation frame from the MoCap to the IMU is found and applied to all cheetah_runs). The Mocap results are thus transposed in the IMU frame can can be more easily compared. The very small angle amplitudes characteristic of the trotting Cheetah, as well as imperfections of orientation estimation are also to be observed - illustrated on a cheetah_run*

Figure 6.7a shows the comparison between IMU and MoCap data for the forward direction(Y axis). Figures 6.7b ,6.7c and 6.7d depict the effect of the 3 different filters and Table 6.3 provides quantitative overall evaluation for all the runs.

We observe that although the performance of the filters is overall quite poor, the Kalman filter still outperforms the two others.

Evaluation method	Low-pass Filter	Median Filter	Kalman Filter
Correlation	0.28 ± 0.06	0.514 ± 0.04	$\mathbf{0.5909 \pm 0.04}$
SNR(dB)	-1.21 ± 0.4	0.71 ± 0.4	$\mathbf{1.5 \pm 0.5}$
RMSE	2.93 ± 0.2	2.35 ± 0.2	$\mathbf{2.13 \pm 0.02}$

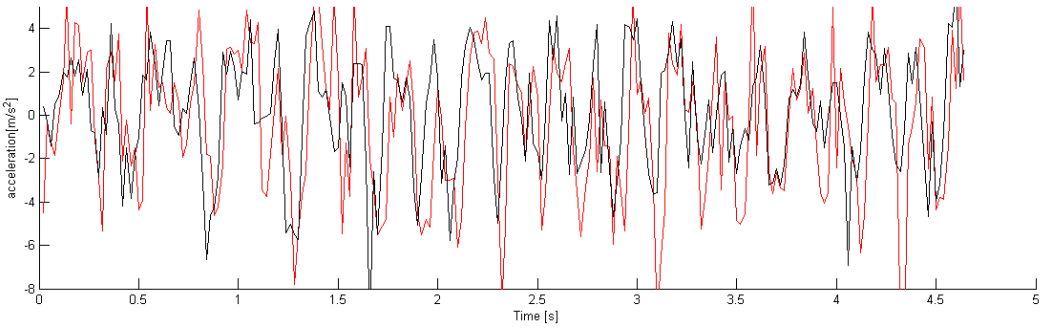
Table 6.3: *Performance comparison between three different filters for forward acceleration signals on all the cheetah_runs; best performances are marked in bold*

The same process should be done for the upward direction (Z axis). However, a brief glance at the raw signal reveals that the difference with the MoCap signal is so significant, that no filtering whatsoever could be achieved. Figure 6.8 shows the similarities between the reference and IMU data in the case of Y and Z axis for one run, and illustrates why the second set is unusable.

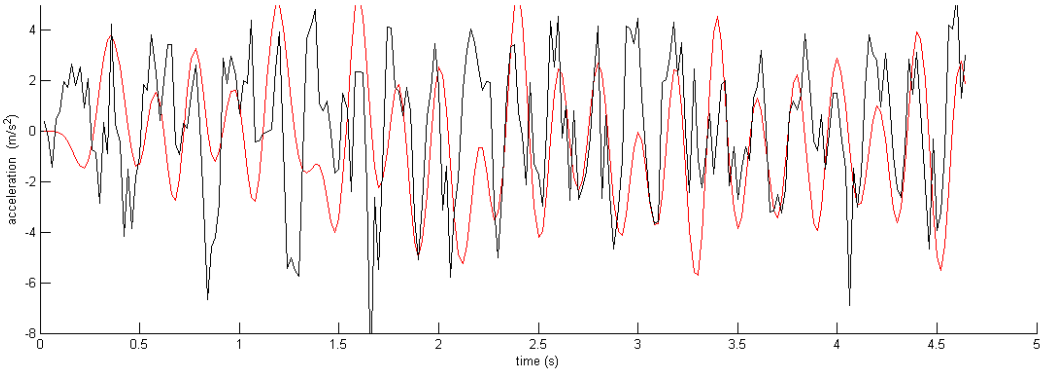
To further illustrate the previous affirmation, we apply the same performance evaluation criteria as before on the two signals. The results are listed in table 6.4. The difference in performance is unacceptable.

We thus consider the signal from Z axis not valid for filtering. As for the arguments that sustain this, they will be explained in the next section.

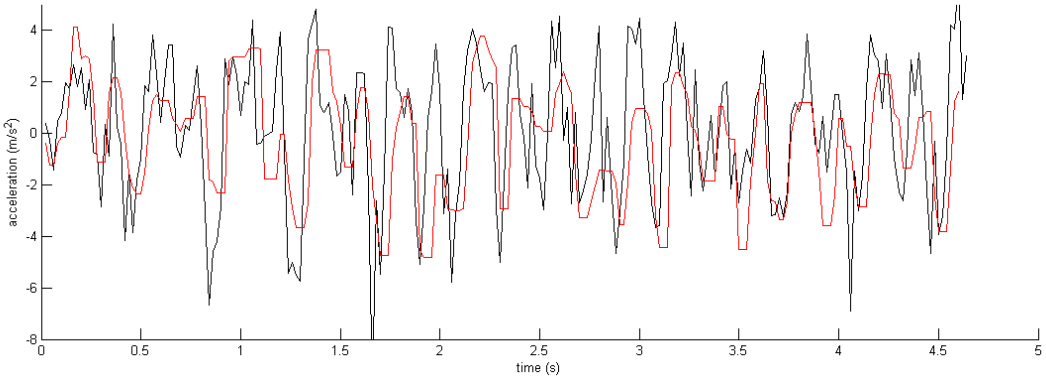
Based on the results obtained for the Y axis, we can now proceed to the speed estimation by



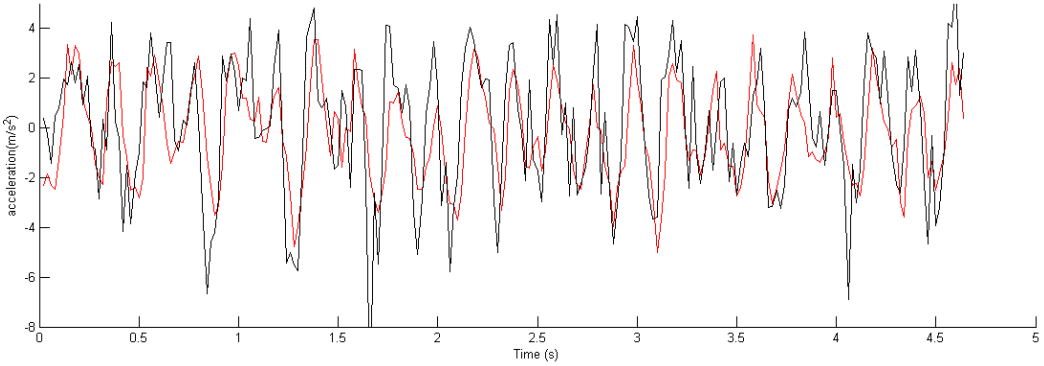
(a) no filter



(b) Butterworth low-pass filter

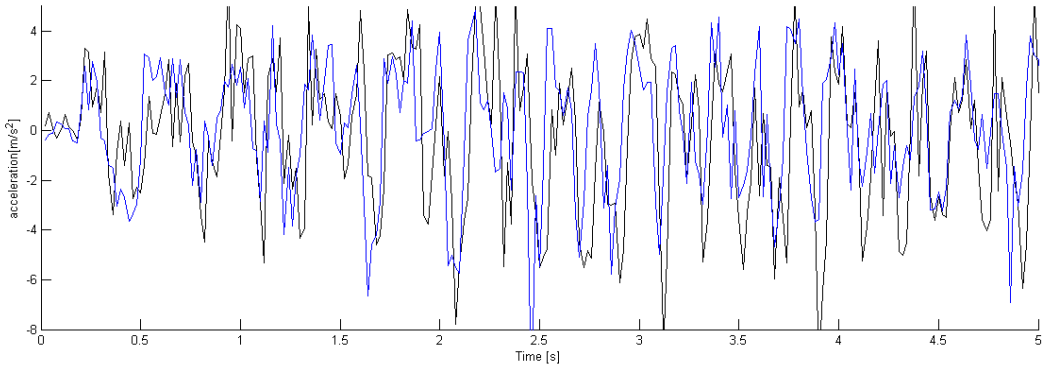


(c) Median filter

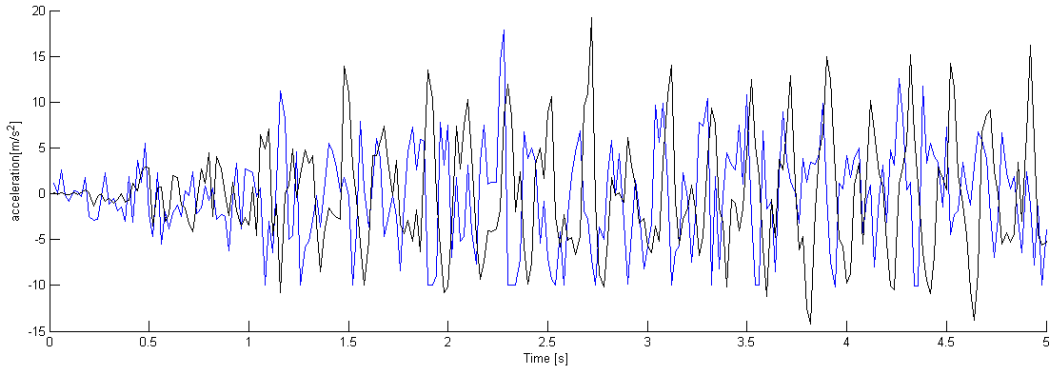


(d) Kalman filter

Figure 6.7: *acceleration on Y axis: MoCap (black) and IMU (red) with different filters*



(a) acceleration on Y axis



(b) acceleration on Z axis

Figure 6.8: *for each axis, the reference data (black) is represented against the unfiltered IMU data(red); whereas unfiltered IMU and MoCap data on Y axis present similarities, signals on Z axis differ significantly. Coriolis and remaining gravity add up and make the latter unusable for further filtering*

Evaluation method	Y-axis	Z-axis
Correlation	0.44	-0.24
SNR(dB)	-1.41	-3.29
RMSE	3.08	9.35

Table 6.4: *'Performance' comparison between unfiltered data; data on the Z axis is unacceptably bad*

integrating the filtered signal. Figure 6.9 shows the output.

The MoCap data shows the mean speed stabilizing itself at $\approx 0.4\frac{m}{s}$, which is similar to the value obtained in [18]. So the MoCap data behaves as expected, which is not the case for the one from the IMU.

The drift that appearing from the first seconds on is substantial. This not surprising, given the overall poor performance of our filtering and the error accumulation related to the integration process. Similarly, one could also notice the small wavelets occurring in the first 2 seconds, when the device is supposed to stay still. These wavelets are produced by the integration of noise and vibration taking place at the moment when the robot stands by for departure, and their amplitude is non negligible in comparison with the speed peaks around their mean value.

Of course, at this point one could consider designing *another* Kalman filter with a more elaborate locomotion model, this time meant to correct the integrated speed. We are nevertheless very skeptical about its ability to correct a drift of this scale.

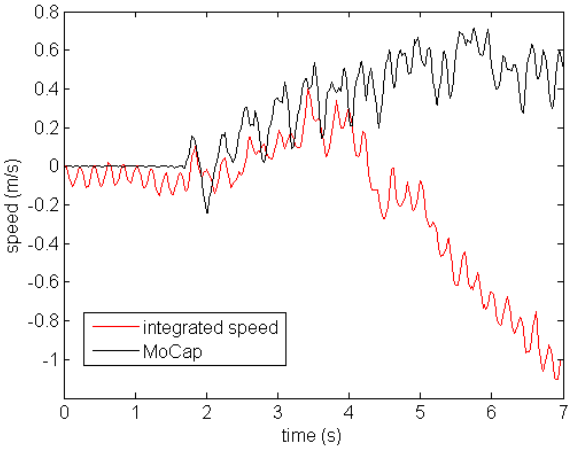


Figure 6.9: *integrated IMU speed (red) and MoCap speed (black)*

6.4 General Discussion

The results obtained above are acceptable, but are far from being outstanding.

In the case of roll and pitch angles, the reference signal coincides with the IMU signal for a short period of time, before starting to deviate. Given the size of the recordable area, it would have been anyway difficult to record more. Although it would have been interesting to be able to measure the spread of the drift on a longer run (about 30-40 seconds), short runs provide us with enough evidence that an instantaneous roll and pitch approximation is feasible, but is rapidly corrupt by bias when dealing with small range amplitutude (not bigger than $\pm 5°$).

Furthermore, the rms error observed in figure 6.5 though, is almost insignificant when dealing with big angle amplitudes ($\pm 60°$), but as it reaches about $1.77°$ and $2.27°$, it can still have a great impact on small angles measurements, like the ones we obtain when running the Cheetah-cub (roll about $\pm 3°$ and pitch about $\pm 5°$). At this scale, the impact of other error factors becomes also significant. It has been shown that the bias can drift for about $0.6°/s$ and that the calibration is faulty (section 3.2).

It is also worth reminding that the IMU samples at 50Hz and the MoCap at 250Hz. As the robot runs at 5HZ, the IMU will only be able to sample 10 times over one **full period**, which makes the output particularly vulnerable to parasitic accelerations. Outliers are much more difficult to detect and low passing produces only little effect. On the 10 samples over the period, 2 bad samples would be enough to significantly deform the signal. This is also likely to explain the local lack of similarity between MoCap and IMU signals that often occurs.

Adapting the scale of the IMU to the task also affects its performance. The accelerometer has been used at the lowest scale ($\pm 2g$), but by negligence, the gyroscope has been used at full-scale ($\pm 2000°/s$), whereas $\pm 500°/s$ would have been entirely sufficient. Lowering the scale reduces the bias drift by a factor of about 5: from $0.6°/s$ to $0.12°/s$. Due to lack of time, new runs on the cheetah have not been conducted.

Finally, the physical disposition of the IMU relatively to the IR markers plays also an important role: the MoCap will anyway record bigger angular speeds, as the IR markers are at the edge of the robot, whereas the IMU is near the CoM.

All these errors combined can sometimes lead to local discrepancies and overall mediocre results, as illustrated in 6.10

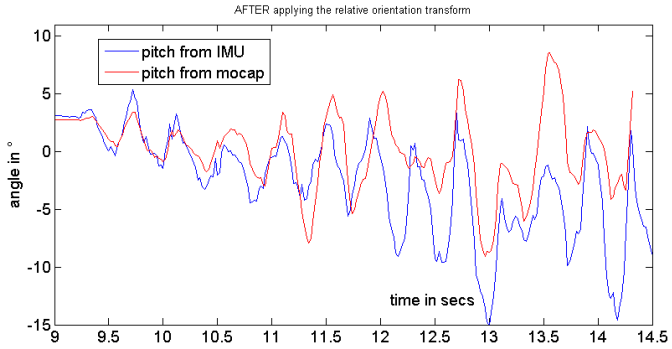


Figure 6.10: *combined error effects at pitch estimation: uncorrected bias, lack of scale factors and different frequency rate; most of all the very small angle amplitudes (about ±5°) significantly affect the overall quality - illustrated on cheetah_run3*

Still, despite these errors, we observe that the relative peaks (i.e. the relative amplitude yof each oscillation) are close to the MoCap values. This is purely a qualitative observation, but it implies that by finding a way to correct the above-mentioned features, including better calibration of the sensors *and* using a gait with wider angle amplitudes could lead to better results.

As for the second objective of this work, the relatively poor results obtained for the roll and pitch estimation indicated already that a speed estimation might be very difficult . We then tried to show what the Kalman filter *can* do, rather than what it cannot. We thus deviated a bit from the main goal and focused on the acceleration filtering, which has proved to be a more fertile terrain for expressing the potential of the Kalman filter and of the IMU.

Although we indeed demonstrate the relative superiority of the Kalman filter, the results are still not as good as the ones obtained in [5]. This is however explicable, as we are facing a different kind of problem. Whereas the authors from [5] deal with human walk, we deal with quadruped robot trot. We are facing a higher movement frequency, which lets the Kalman filter less time to react and with lower acceleration amplitudes, which means that it is more difficult the distinguish the real movement between noise and parasitic accelerations.

Moreover, Coriolis acceleration is deliberately neglected,

$$\vec{a}_c = -2\vec{\Omega} \times \vec{v}$$

because no reliable real-time speed estimation is available. The highest amplitude is obtained by the pitch angle (figure 6.6) and the highest speed is on the forward axis, which logically means that the Coriolis effect will be the most effective on the Z axis. A quick evaluation reveals that pitch angle has speed peaks at up to $4.0\frac{rad}{s}$, which in combination with the average speed on the Y axis of $0.4\frac{m}{s}$ yields a Coriolis acceleration of about $3.2\frac{m}{s^2}$. Given the range of our acceleration, this is substantial.

In addition, as roll and pitch angle estimation is faulty, gravity compensation is inevitably affected, which means that some residual gravity may also affect the acceleration signal. These two reasons fully justify why the discrepancy between the MoCap and IMU data for the Z axis acceleration is so big, that it could not be used.

Finally, the quality of the IMU plays also an important role. Despite the calibration conducted throughout chapter 3, the device is still inadequate for such precise tasks as roll and pitch angle estimation for a trotting robot.

The range of IMU sensors is quite large spanning from cheap home-assembled modules to reference models from which we hereby cite the MTi [19] from Xsens and OS3D from Inertial Labs [20]. Of course, the price difference between an MTi (1000 euros) and our RM-G146 (70 euros) is also substantial. Yet if we were to stay at the same price (80$), there are also amateur models with included 9DOF data-fusion algorithm such the one proposed by ArduIMU [21]. This would save the time allocated to calibration and orientation estimation.

This does not mean that the sensor is unusable, but that it ought to be used in applications where precision is less important than in speed estimation (e.g. gaming), or to be used *in combination with other sensors.* Throughout literature, Kalman filter is often used with additional sensors. In most of the cases where accurate information about the device's state is required, the GPS [14] data is also fused. In our case, a force sensor would be the most adequate, allowing constant correction when touching the ground. This would make possible the design of a more elaborate locomotion and Kalman model.

Chapter 7

Conclusion and Further Work

In this work we were able to estimate the instantaneous roll and pitch angle of a trotting robot using a first Kalman filter as well as the acceleration using a second Kalman filter. The first fuses acceleration and angular velocity from the IMU and the second gravity-free acceleration and a locomotion model.

The first step consisted in comparing different in-field calibration methods, and although a satisfactory method was found for the accelerometer (by ellipsoid fitting), the gyroscope still lacks of proper calibration.

The second step was creating a comparison setup, with the MoCap system acting as ground truth. This allowed us to evaluate and calibrate our filters.

The third step was the adaptation and design of the Kalman filters. The code for first filter was already provided but had to be adapted to our situation and the second was designed based on a simplistic locomotion model.

In a fourth step the results were compared and it is was noticed that their quality was not due to software or theoretical problems, but alone to hardware issues.

Roll and pitch estimation are usable for large movements (wide angle amplitude). At smaller range, as for runs on the ground, the errors become visible. In acceleration filtering on the Y axis the Kalman filter outperforms standard filters even if a very simple locomotion model is used. The quality of the estimation suffers though from reasons related to the nature of the robot's run and (sampling) limitations of the IMU. In the case of the speed however, the limitations and flaws of the Kalman design and IMU become evident, and a proper estimation is not feasible . The acceleration on the Z axis is unusable because of combined effect of Coriolis acceleration and imperfect gravity compensation.

We finally point out that the hardware plays a major role when pursuing accurate real-time state estimations, and indicate other hardware alternatives.

The overall idea of such a project was to be able to make relevant evaluations on the state of the robot and use this information in a closed-loop model. We have demonstrated that such a concept is feasible, yet some pieces of the puzzle still do not work properly.

If any future project is to continue on the lines sketched by this one, it should focus on finding adequate hardware, consider adding an additional sensor (e.g. force sensor) and extending the locomotion model. Taking the temperature and the compass into account in the Kalman model and most of all better calibrating the IMU might produce significantly better results. Alternatively, one could also research whether there is a way to properly calibrate the sensor with the help of the MoCap.

Eventually, if everything works out off-line, the final challenge would be to make it work in real time, on the robot.

Chapter 8

Acknowledgments

I would like also to thank Alexander Spröwitz and Alexandre Tuleu for supervising and supporting me during this project, as well as Prof. Auke Ijspeert for allowing me to take part in it. Special thanks goes to Adrien Briod for all the support and code he provided me with, as well as for being patient and kind enough to answer my exasperating questions. Finally, I thank everyone who bravely endured my complaints during this busy semester.

List of Figures

List of Tables

Bibliography

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[2] Jr. LaViola, J.J. A comparison of unscented and extended kalman filtering for estimating quaternion motion. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2435 – 2440 vol.3, june 2003.

[3] Brian J. Odelson, Murali R. Rajamani, and James B. Rawlings. A new autocovariance least-squares method for estimating noise covariances. *Automatica*, 42(2):303 – 308, 2006.

[4] N. Ramakoti, A. Vinay, and R.K. Jatoth. Particle swarm optimization aided kalman filter for object tracking. In *Advances in Computing, Control, Telecommunication Technologies, 2009. ACT '09. International Conference on*, pages 531 –533, dec. 2009.

[5] Wei zhong Wang, Yan wei Guo, Bang yu Huang, Guo ru Zhao, Bo qiang Liu, and Lei Wang. Analysis of filtering methods for 3d acceleration signals in body sensor network. pages 263 –266, nov. 2011.

[6] STMicroelectronics. Lsm303dlh. `http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00260288.pdf`. Accessed October, 2012.

[7] InvenSense. Mpu-3000/mpu-3050 product specification. `http://invensense.com/mems/gyro/documents/PS-MPU-3000A-00_v2.5.pdf`. Accessed October, 2012.

[8] Ugo Grimaldi Franco Ferraris and Marco Parvis. Procedure for effortless in-field calibration of three-axis rate gyros and accelerometers. *Sensor and Materials*, 7(5), dec. 1995.

[9] David Jurman, Marko Jankovec, Roman Kamnik, and Marko TopiÄ. Calibration and data fusion solution for the miniature attitude and heading reference system. *Sensors and Actuators A: Physical*, 138(2):411 – 420, 2007.

[10] STMicroelectronics. An3192, application note. `http://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf`. Accessed October, 2012.

[11] Timo Pylvänäinen. Automatic and adaptive calibration of 3d field sensors. *Applied Mathematical Modelling*, 32(4):575 – 587, 2008.

[12] A. Olivares, G. Olivares, J.M. Gorriz, and J. Ramirez. High-efficiency low-cost accelerometer-aided gyroscope calibration. In *Test and Measurement, 2009. ICTM '09. International Conference on*, volume 1, pages 354 –360, dec. 2009.

[13] Xiaoping Yun and E.R. Bachmann. Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking. *Robotics, IEEE Transactions on*, 22(6):1216 –1227, dec. 2006.

[14] Francois Caron, Emmanuel Duflos, Denis Pomorski, and Philippe Vanheeghe. Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects. *Information Fusion*, 7(2):221 – 230, 2006.

[15] M. Raibert, M. Chepponis, and H. Brown. Running on four legs as though they were one. *Robotics and Automation, IEEE Journal of*, 2(2):70–82, 1986.

[16] Marc H. Raibert. Running with symmetry. *The International Journal of Robotics Research*, 5(4):3–19, December 1986.

[17] Marc H. Raibert. Trotting, pacing and bounding by a quadruped robot. *Journal of Biomechanics*, 23(Supplement 1):79–81, 1990.

[18] Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion- design, control, and experiments with a compliant quadruped robot (under review). July 2012.

[19]

[20]

[21]