Distance Measurement using a Surface Micro-machined Accelerometer
Matthias Guzmits, Christoph Schnabl, Manuel Eiwen, Simon Marton-Lindenthal, Sebastian Rohrer, Alexander Lampalzer
Department for Computer Science, Secondary Technical College, HTL Wiener Neustadt

Distance Measurement using a Surface Micro-machined Accelerometer

***Abstract*—Orientation is one of the big challenges in autonomous mobile robotics. Outdoor orientation methods, such as GPS, aren't available in the Botball® competition. Therefore the possibilities for distance measurement in Botball® are nearly limited to the use of an accelerometer. In this paper, we conducted an experiment on whether the wallaby's IMU can be used to measure a driven distance. Therefore, we discuss the different types of accelerometers together with their use cases and the physical principles behind them. We discovered that the wallaby is nearly unusable for measuring distances with its IMU (inertial measurement unit). We provide an explanation, as well as the calculations which are used.**

I. Introduction

Knowing the exact distance driven is important in autonomous robot systems as there are many environmental factors (wheel friction, weight distribution, inaccurate motors) influencing the accuracy of driving. Reliable and accurate measuring results can be accomplished by using the Global Positioning System. Often, an update frequency higher than what GPS can provide is needed. Because of that, a common solution is to calculate the distance based on acceleration, while still using the GPS position as an external reference. Unfortunately, such Methods are only realizable in outdoor environments. Yet, accurate orientation techniques are often needed in the indoor settings the Botball® competitions get carried out. Even if the Botball® competition was carried out outdoor, navigation would not be possible using GPS, since the wallaby controller does not have a GPS receiver. As it is still essential to know how far the robot has moved, many different methods (e.g. using the wheel perimeter multiplied by the motor rotations as a reference or time-based methods) have established throughout the Botball® community. This paper outlines the aspects of using the accelerometer built into the KIPR Wallaby Controller to accomplish the task of distance measurement as an additional method of orientation for the Botball® competitions.

II. Accelerometer

A. Principle [1], [8]

An accelerometer measures acceleration force. Generally there are two types of accelerometers, the AC accelerometer and the DC accelerometer. The AC accelerometer has an alternating current output and is only able to measure dynamic forces. Static forces like gravity are not quantifiable for this kind of accelerometer. In comparison, the DC accelerometer has a direct current output and is capable of measuring static and dynamic forces. This sensor not only measures dynamic forces which appear during movement, but also static forces like gravity or pulling on the sensor. Accelerometers can use a bunch of techniques to measure the acting forces, but each differs in accuracy and areas of application. Every type is based on its own physical principles.

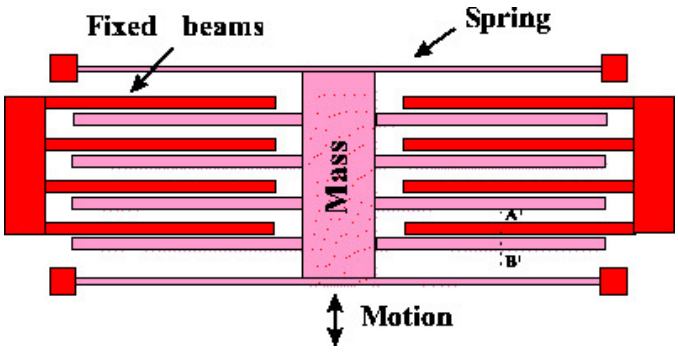

Fig. 1: Surface Micro-Machined Accelerometer [4]

1) Piezoelectric Effect [2], [3], [8]: One of these techniques is the piezoelectric effect. This type of accelerometer contains a microscopic crystal structure, which generates voltage if the crystal is getting stressed. In comparison to other accelerometer types these are more cost intensive, but they have a few advantages like better accuracy and better durability.

2) Surface Micro-Machined Accelerometer [2], [3]: Accelerometers of this kind consist of masses, springs and motion-sensing elements. The production process of these sensors contain standard IC (integrated circuit) processing techniques. The sensor is composed of the sensor's core, which is a 3D structure suspended above the substrate.This makes it possible to measure the axes. The sensor-core is a silicon structure or mass suspended which is held in place by "springs" above the substrate. They provide resistance to the movement of the acceleration forces. Both types have fixed plates that form a differential capacitor, which gets unbalanced by any movement. This results in a square wave output. The amplitude of this wave is proportional to the acceleration. On each axis is a demodulator that rectifies the signal. With the result of the demodulation the accelerometer determines the direction of the acceleration. The Wallaby's Inertial Measurement Unit (IMU) uses a Surface Micro-Machined Accelerometer.(see Fig. 1)

3) Other Types of Accelerometeres: In addition to the above, there are many other techniques to implement an accelerometer, such as with laser, optics or induction. Every method has its own advantages and disadvantage s. In order to choose the right one it is necessary to know what the sensor must be able to perform and in which environment it is used.

B. Common Use [5]

The accelerometer's typical use cases include protecting falling hard drives from a head crash by switching the hard drives off, developing airbags in cars and detecting plane crashes. Mobile phones and smartwatches containing an IMU. The IMU itself consists of an accelerometer and a gyroscope. But the accelerometer can also be a standalone sensor. IMU's are used in aircrafts, spaceships and as a supporting system in photography to stabilize the camera.

C. Accelerometer in Botball

The accelerometer built in the KIPR Wallaby is an analog sensor that can measure DC accelerations. This means, as mentioned above, it is also able to represent static forces. The sensor the Wallaby uses is capable of measuring forces between -2g and 2g and has an accuracy about 0.061mg per last significant bit. This sensor's problem is the extremely fast stream of data. When the accelerometer is pushed to one side with a movement not smooth enough, the values might invert themselves. For example, if the sensor is pushed forward in the x-axis and the movement abruptly stops, the positive values the accelerometer normally returns during movement turn negative. That could be the result of the law of inertia. The actual problem is, that the time interval between two measurements is not constant. The larger the difference between the amount of data in two intervals, the more inaccurate the calculated value per interval.

Characteristics(Linear acceleration)	**Specification**
Measurement range	$2g$
Sensitivity	$0{:}061mg/LSB$
Sensitivity change vs. temperature	$0{:}01\%/^{\circ}C$
Typical zero-g level offset accuracy	$60mg$
Zero-g level change vs. temperature	$0{:}5mg/^{\circ}C$
Noise density	$150ug/sqrt(Hz)$

TABLE I: Sensor characteristics [6]

The accleremoter is accessed by using the functions in KIPR's libwallaby. The library contains functions for each of the three axes. They return the measured acceleration as a signed short. 1G is represented by 2^{10} bytes.

- signed short accel_x()

  Returns the sensed x acceleration. The value is approximately 0 at rest and on a flat ground.

- signed short accel_y()

  Returns the sensed y acceleration. The value is approximately 0 at rest and on a flat ground.

- signed short accel_z()

  Returns the sensed z acceleration. The value is approximately -1024 at rest and on a flat ground. This deviation from the other two axis values exists because of the earths acting gravitational force of $9.81\frac{m}{s^2}$

Fig. 2: libwallaby's Accelerometer functions

III. Positioning

A. GPS

GPS is one of the best known methods to find the current position of a device. This method has many advantages like a really high precision. But on the other hand, there are some disadvantages. If the device does not receive any signals from satellites, it will not work. That is why GPS is only usable in outdoor environments. This technique is useless in the underground or in buildings which are shielded or built of reinforced concrete. In addition, some applications or use cases need faster updates. One of those are self-driving cars, which would be too unreliable without other methods, even for testing purposes.

B. Kalman-filter (GPS + Accelereometer)

This filter is a recursive filter which estimates the state of a dynamic system. This system consists of noisy measurements. The Kalman-filter uses different sources for the calculation of the position. The GPS and accelerometer are just two out of many possible sources. Basically the Kalman-filter uses the fact that every movement of an object is predictable. The GPS provides the basic position information. Since that information can fluctuate a lot, adding an accelerometer allows to calculate a smooth progress. Combined with the acceleromter it is possible to calculate a smooth progress. While the GPS waits for new data, the position is calculated using the current acceleration. Once the GPS receives new information, the process repeats itself. Because a GPS is used, this method is only applicable for outdoor usage.

C. Accelerometer Only

To use only an accelerometer is a solution to calculate the distance only by knowing the accelerometers sensed acceleration values is needed. This method is capable for indoor applications like Botball R .

$$v(t) = \frac{s}{t} \quad a(t) = \frac{v}{t}$$
$$a(t) = \frac{2s}{t^2} \quad s(t) = \frac{1}{2}at^2$$

For constant acceleration this is a simple task to achieve. Velocity is the distance s traveled in a certain interval of time t. Acceleration is defined as the velocity gained in a certain amount of time.

D. Further indoor navigation methods

There are other, more mathematically advanced methods. Most of them are premised on the uncertainty in mobile robot environments. Therefore, they use a probabilistic approach that leads to the research domain of probabilistic robotics.
The fundamental aspects of this area are Recursive State Estimation [13] techniques. Their idea is determining the next state with the current sensed data. State estimations entail noise corruption in their calculations. These methods can be split into two major categories: Gaussian Filters and Nonparametric Filters. One Gaussian-Filter technique is the already mentioned

Kalman- lter. Others are the Extended Kalman Filter or the Information Filter.

The aforementioned mathematical models used for localization can be combined with depth cameras, ultrasonic sensors or Lidars to achieve Simultaneous Localization and Mapping (SLAM) [12]. One of the major problems in Autonomous Mobile Robotics is not knowing your current location and environment at the same time. SLAM techniques calculate a map of the robot's environment and its position in the map concurrently.

IV. Accelerometer for distance measurement

$$v(t) = v(t_n) + \int_{t_n}^{t} a x dx$$

$$s(t) = s(t_n) + \int_{t_n}^{t} v x dx$$

Fig. 3: Distance using the Anti-Derivate of Velocity

$$s(t) = \frac{1}{2} a \quad t^2$$

Because the sensor sends data really fast, it is necessary to form the median of the received data. With it it is possible to get a snapshot of the acceleration if a speci c period of time. The period should be as small as possible to get the most accurate results. To get the driven distance, the acceleration is multiplied with the length of the period squared. To use this method in a game, all results must be added up.

A. Calibration

Because of Earth's gravitation calibrating the Wallaby's accelerometer is important when having to rely on the most accurate readouts possible. A calibration function exists in the libwallaby provided by KIPR, but as of the state of this paper, this function is not implemented yet. Therefore a custom calibration function is utilized. It should be used as often as possible when the device is not moving. A good approach is to re-calibrate the accelerometer when the bot is aligning itself on a pipe or when you know that the currently driven distance is de nitely correct.

```
struct accel {
    int ax;
    int ay;
    int az;
};

void calibrate( struct accel * bias) {
    static int data[3][100];

    for ( int i = 0; i < 100; i++) {
        data[0][i] = 1;
        data[1][i] = 1;
        data[2][i] = 1;
        msleep(10);     //wait 10 miliseconds
    }

    bias->ax = median(data[0]);
    bias->ay = median(data[1]);
    bias->az = median(data[2]);
}
```

Fig. 4: Accelerometer Calibration code snippet

The accel struct acts as a representation of an acceleration vector, where ax, ay, az relate to the corresponding axis. The idea behind the custom calibration function is to account for biases. That is achieved by passing an acceleration vector to the calibrate function, which lls the vector with the bias of each axis. They are calculated by taking the median value of 100 individual measurements with a time gap of 10 milliseconds. The entire calibration takes approximately 1.3 seconds. The calculated bias is later subtracted from the measured values during driving.

B. Digital Filter [7]

As the measured sensor values contain noise [11] a digital lter is applied. In this case, the moving median lter is used because outliers are completely ignored.

Fig. 5: Moving Median (darker color) vs Input Data (lighter color)

Fig. 5 describes the necessity of using a digital lter to manipulate the measured sensor values. On the x axis the elapsed time is displayed. The y axis corresponds to the values returned from the accelerometer. As visible above, the moving median lter is a little delayed, but smoothens the values for further processing.

```
int moving_median( int * data, int num_elements) {
    sort(&data);

    if (num_elements < 2) {
        return data[num_elements];
    } else if (num_elements % 2 != 0) {
        return data[(num_elements / 2)];
    } else {
        return (data[num_elements / 2] + data[(
            num_elements / 2) + 1]) / 2;
    }
}
```

Fig. 6: Moving Median Digital Filter

C. Numerical Integration

The returned values of the accelerometer are not integrable with an analytical integration method. Because of that, numerical integration is used to get the driven distance. It works by dividing the function in squares as small as possible, which are then added up. The result is the sum of all squares. There are some methods to do these integrations.

1) Rectangle Rule:

$$R_L(t) = \frac{b-a}{n} \sum_{i=0}^{n-1} f\left(a + \frac{b-a}{n} i\right)$$

Fig. 7: Rectangle Rule Formula

2) Trapezoidal Rule:

$$T_S(n) = \frac{b-a}{2n}\left(f(a) + f(b) + 2\sum_{i=1}^{n-1} f\left(a + \frac{b-a}{n} i\right)\right)$$

Fig. 8: Trapeziodal Rule Formula

```
int trapeziodal_integration( int * data) {
    int n = sizeof (data) / sizeof (data[0]);
    int sum = 0, a = 0, b = n - 1;

    for ( int i; i < n; ++i) {
        if ( i == 0 || i == n-1 ) {
            sum += data[i]/2;
        } else
            sum += data[i];
        }
    }

    return sum * ((b - a) / n);
}
```

Fig. 9: Trapeziodal Rule Implementation

D. Comparison

Fig. 10 shows the integrated values during a run. They correspond to the data displayed in Fig. 5. The values in Fig. 10 are numerically integrated using the rectangular rule. The integration gets inaccurate over time. To enhance the accuracy and minimize the appearing spikes in the calculation the moving median is necessary and has a huge impact on the result.

Fig. 10: Integrated moving median (darker color) vs Integrated input data (lighter color)

V. Experiment

The setup of the experiment is a simple bot with two wheels, a base-plate and a wallaby. The accelerometer in the wallaby is the most important element of the experiment. To verify that the calculations are correct, we measured the actual driven distance with a tape measure. The bot starts driving at the beginning of the measure tape and stops after a given amount of time. The program calculates the driven distance and displays it afterwards. After this step, it is necessary to compare the output from the program with the measured distance according to the tape measure. In the experiment the calculation is done while the bot is driving. The formula used is as follows:

$$s(t) = \frac{1}{2} a t^2$$

Fig. 11: Formula to calculate the driven way

The calculation is done with the raw data, the moving median and with attened values using a logarithmic calculation.

Driven Time	Base
2s	1.37
3s	1.65
4s	1.95
5s	2.22
6s	2.53
7s	2.9
8s	3.2
9s	3.8
10s	4.55
11s	3.62

TABLE II: Empiric determined base of the used logarithm

The uctuation of the values from the raw data is pretty huge. With the moving median the values are much better. The best result is achieved by attening of the raw data with an logarithm. The base of the logarithm comes from an empirically determined table, which is based on the time needed.

Table II shows the empiric determined base of the above mentioned logarithm. The values are an approximation to get closer to the real driven distance with the used calculation.