Comprehensive Light-Start Methods in Botball

James Gosling*, Matthias Rottensteiner, Alexander Müllner
Höhere Technische Bundeslehr- und Versuchsanstalt Wiener Neustadt
Higher Technical Federal Teaching and Research Institute Wiener Neustadt
Department of Computer Science
2700 Wiener Neustadt, Austria
*Corresponding author's email: gosling.james@student.htlwrn.ac.at

***Abstract*—This paper addresses a common issue in botball - the light-triggered start. In some cases, this problem can even result in the loss of an entire run. The paper provides tests, tips, and customised solutions to mitigate the negative effects that the sensor's specific environment can have on its functionality. However, it is important to note that not only every team should try to reduce noise and get the best possible signal, but that event organisers should know how to do this as well. New and experienced teams alike may not know how to deal with environments that change rapidly throughout the day and venues with many light sources. A single run that would have scored many points, but is ruined by a wrong light signal resulting in false starts, can ruin a team's morale forever.**

***Index Terms*—Light sensor, Botball, robotics, comparisons, filter techniques**

I. Introduction

For starters, a typical round will begin with mounting the lamp at the side of the table by the teams and according to their preferences, and then a hands-off period, which is the most critical section of any run. If the robot starts before the lamp gets turned on, the entire run amounts to zero points. A second chance is allowed, however. This paper is intended for both players and organisers. However, it is the players who will find it most useful. The paper compares several approaches to reducing environmental noise. It also addresses how to deal with a changing environment.

II. Experiment Setup

Tests will be carried out using the robot shown in figure 2. There will be a full explanation of each test when it is relevant, but generally the tests will be along the lines of this scheme: The robot will be placed in a specific location and its circumstances will be explained. Then it will move in a certain pattern of a certain range and its measured respective light values and the difference between its lowest and highest values will be compared. This ill usually be measured around the corners of a certain shape. This approach is efficient from the point of view of both time and explanation to the user. Although for a Botball light-start a robot typically remains stationary, different locations show the variability of the incoming light more consistently.

A. Equipment

The robot was designed with one goal in mind: To be as accurate as possible in the situations you can expect when

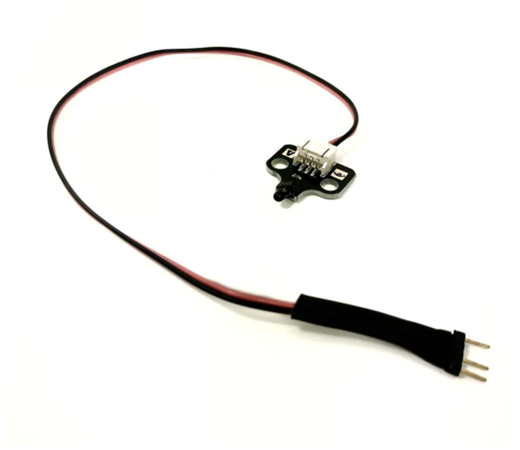


Figure 1: A picture of the light sensor used. [1]

mounting a sensor on a robot used in the Botball competition. The light sensor is placed at a height that is similar to what it would be like if it were placed on an iRobot Create, which is 92mm or 3.6 inches from the ground up. Because readings should be independent of the robot being used, it was not placed on a Create. This is done to ensure that the results and methods presented can be applied to as many different types of bots as possible. A front-facing light sensor and a rear-facing light sensor are present. These are analogue [2] and measure light in a radius of about 20 degrees [3]. With a spectral response of about 880nm, these works with variable resistors. This means that in order to detect something, there is a component in the sensor that changes its resistance depending on if a specific wavelength is shining on the sensor. [4] This method is usually used at a wavelength of around 610nm, which means that it is outside the infrared spectrum. These types of sensors, technically known as LDRs or light-dependent resistors, have a very high resistance when it is dark, but as the brightness increases the resistance decreases. [5] When the functions are called to read the measured light sensor response, these values are translated into numbers that can be read. Unless otherwise stated, primarily the front-facing sensor will be tested. A figure of this robot is shown here.

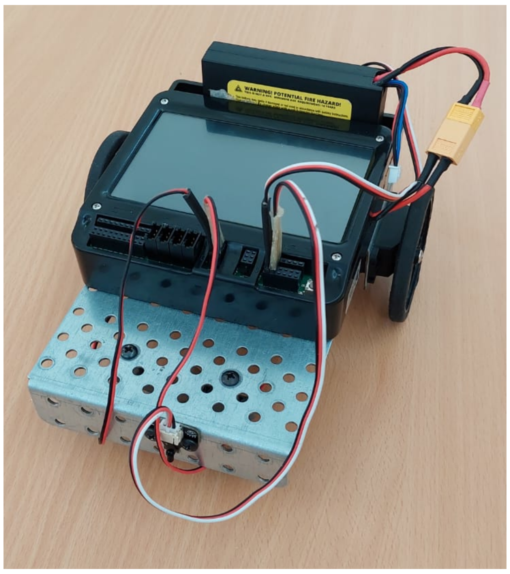


Figure 2: A picture of the robot used in the tests

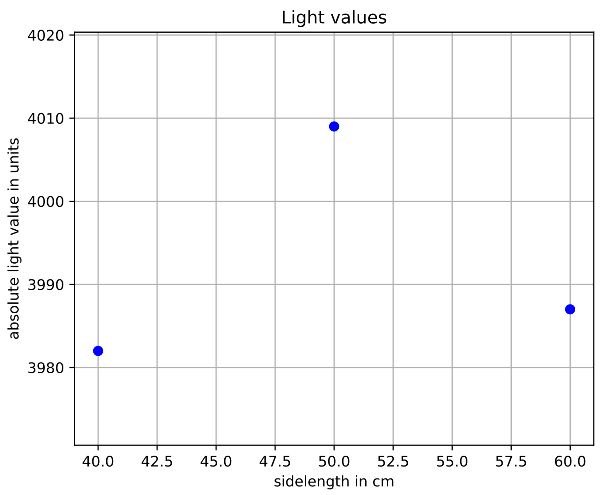


Figure 3: The value of the light at each measured point in the test

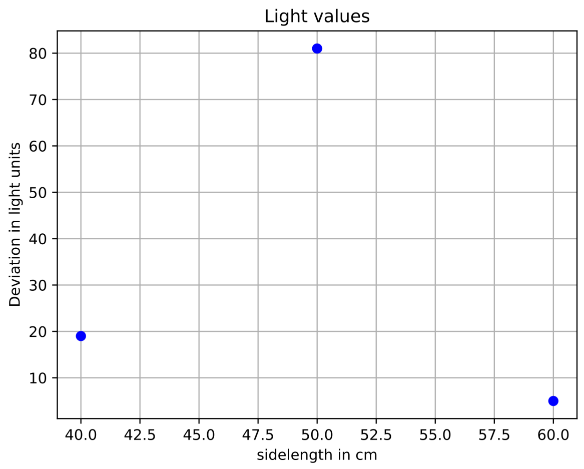


Figure 4: The maximum deviation between the light values at the measured points

B. Values and Units

The light sensor uses it's own metric, which ranges from the values 0 - 4096, with 0 denoting absolute brightness and darkness being represented with 4096. This is because the sensor is connected to the wombat with a pull-up resistor. Furthermore, the sensor seems to have a bias towards lower light levels, which might be explained by how it reads the infrared wavelengths. This means that, when not pointing towards a specific surface, it tends to select a value that is close to 4000. These given numbers will be simply called light (values) or units. Every graph will either be declared as one of these units or will be given in percent.

III. Implementations

For the first test, the robot will be driving along a square-shaped track and measure how different the detected values can be for different lengths of the side fragments. Every point shown on the graphs will be the average of all four points measured at that sidelength. The first set of tests will work in the following way: The test robot will drive in a square shape and the detector will be able to measure their respective luminance at certain distances. This will be done without anything obscuring or aiding the sensors. Anything that can or will help with the objective will be added later, as it is important to get an idea of the base values. Higher values indicate a darker subsurface.

The tests were carried out on a standard botball table [6], which means that the first tests with a side length of 40 cm are about the same size as the starting box. 60 cm is about 3/4 of the depth and almost 1/4 of the length of a typical botball table. The deviation is greatest in the middle test, meaning that at about half the depth of the table, the light changes rapidly and aggressively in our setup. It is at the square corners that all our values are recorded. The first thing to do is to check the position of the sensors, otherwise they could cast damaging shadows. In Botball you will probably place the sensor on the back of the robot. This will increase the darkness by up to 76%. This also means that a large amount of light is blocked, which helps to minimise environmental damage. As explained in the following section 'Suggested Solutions', this will become even more intense if the sensor is mounted downwards.

A. wait_for_light

The wait_for_light function is used for a start. It waits, and periodically generates a boolean value which, if true, breaks out of a loop and allows the rest of the program to commence. It continuously measures the light values, and if one converges from the last by a fixed unit difference of 100, the condition is met and the function ends. [7]

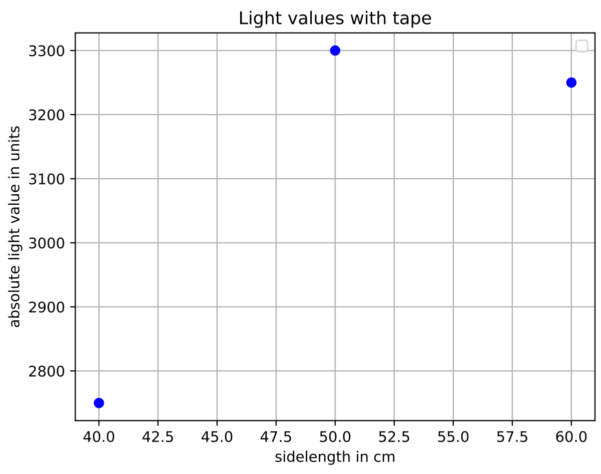


Figure 5: Light values at certain side lengths with tape on its sides

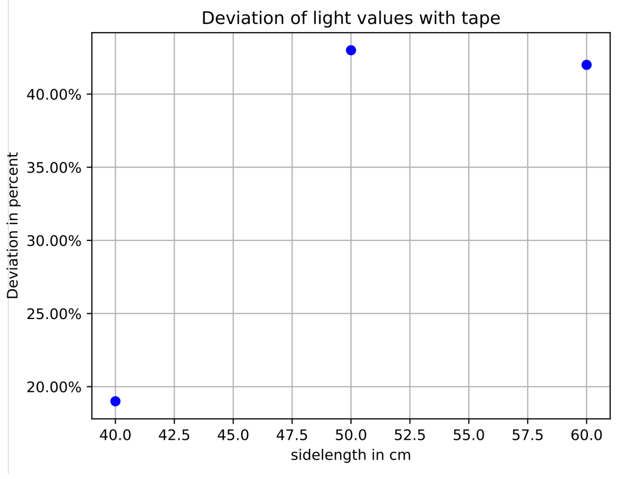


Figure 6: The maximum deviation between the squares points light values with tape on its sides

IV. Proposed Solutions

The results in this section might not be reproducible and therefore provide only anecdotal evidence. So, from now on the variation in % to the ratio without anything attached will be given. One thing we will not discuss is the so-called "straw solution", where you shield the sensor with a paper or plastic straw, because this approach is a beginner's solution.

A. Black tape

The simplest solution, and one used by many teams, is to place a piece of black tape or something similar around the light sensor, leaving a hole where the start signal will hit the sensor. The main advantage of this method of reducing the noise in the environment is that the light will not be picked up from any unwanted angles. Unfortunately, this approach has two major drawbacks: Firstly, the robot must be set up in exactly the same way every time, as a slight change in angle can have a significant distorting effect on the signal and, in the worst case, result in an early start or no start at all. A second disadvantage is that the tape can easily stick to the sensor and block the light. Removing it can also be tricky. Often a light sensor can be damaged. Leftover adhesive can also alter the sensor's input. All things considered, this solution is good, but risky and unstable. The results of our tests are shown in Figures 5 and 6, and are the subject of discussion below.

As demonstrated in figure 6, if you stick the tape around the sensor you get about 18 to 45 (!) % of the deviation you would get otherwise. Put differently, the values vary a lot less than without the tape, because excess light from unwanted angles is filtered out.

B. Rotation

The second experiment examines the influence of sensor rotation. As previously stated in the equipment section, two sensors are attached to the robot, but only one has been evaluated thus far. Now we are going to compare the two, with

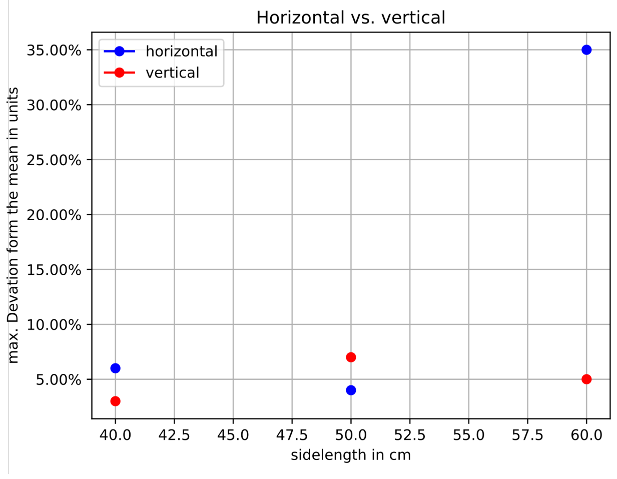
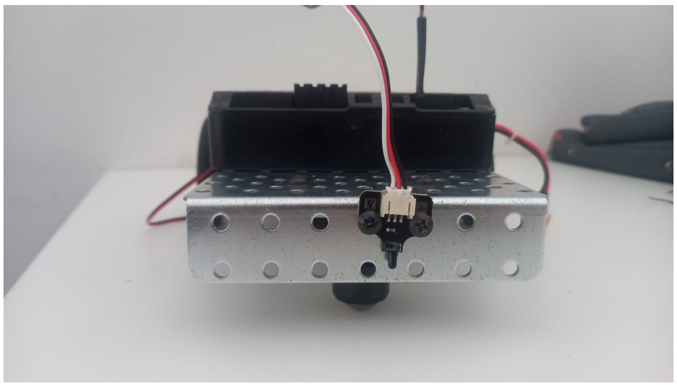
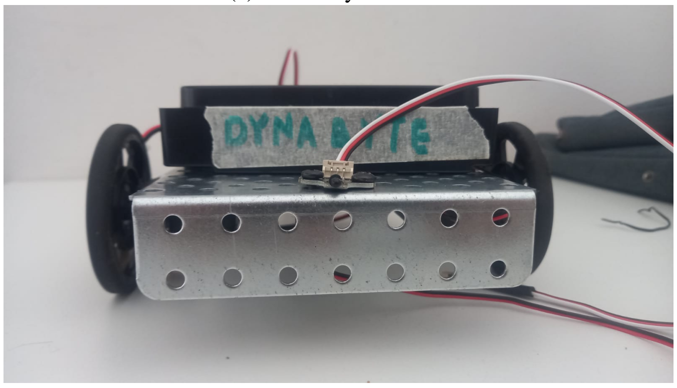


Figure 7: The difference in deviation / variation for horizontal and vertical sensors.

the main difference being that one is mounted horizontally and the other vertically, facing downwards. This is because if they were pointing upward, they would be facing the light from the room itself at all times. You can see both builds in figure 8 a & b respectively. The absolute readings will be skipped this time, as the light conditions will of course be the same for both sensors. The test was performed in a dimly lit room in order to keep the variation very small. It is worth noting that the underground is not a monochrome colour, but a representation of last year's botball table. As the vertical sensor is mounted downward, it can detect these shades and alter its value. This can sometimes result in seemingly illogical outcomes. However, it is important to note that only the deviation is being compared in this instance. Figure 7 displays the results.

As you can see, in spite of the underground, the downward-facing ones have the upper hand, as they usually do not


(a) Vertically mounted



(b) Horizontally mounted

Figure 8: Vertically and horizontally mounted sensors

come into contact with the bright light source coming in from diagonally above, which the horizontally positioned ones seem to catch. This means that you should position the lamp diagonally below your sensor at the start of botball rounds. You could also use it for other purposes, such as communicating game information between robots, as previously demonstrated in Botball. [8]

C. Kalman-Filter

A software-based approach could also implement the Kalman filter, which estimates the state of a system based on the data collected. The measurement variance is approximately 1700 units due to variations in light values when light is shone at it at a 20-degree angle in a normalised position. This is about 50% of the maximum value (4096) that the sensor can deliver! Therefore, the filtered approach would resemble the following pseudocode: [9]

```
class KalmanFilter:
    state_estimate = current state
    estimate_error = initial error estimate
    process_variance = how much the filter can
    change in a single step
    measurement_variance = how much the input can
    change in a single step

    method KalmanFilter(state_estimate):
        this.state_estimate = state_estimate

    method update(measurement):
        // Prediction step
        state_predict = state_estimate
        estimate_error_predict = estimate_error +
    process_variance

        // Update step
        kalman_gain = estimate_error_predict / (
    estimate_error_predict + measurement_variance)
        state_estimate = state_predict + kalman_gain
     * (measurement - state_predict)
        estimate_error = (1 - kalman_gain) *
    estimate_error_predict

        return state_estimate

function lightstart(filtered, noisy):
    if noisy < filtered * 0.8: //(1 - percent that
    the values have to change to start(for example:
    1 - 0.2 (20%) = 0.8))
        return true
    else:
        return false

function main():
    zen = Sensor()
    kalman_filter = KalmanFilter(zen.read())

    noisy_reading = 0.0
    filtered_light_level = 0.0

    while true:
        noisy_reading = zen.read()  // Replace with
    your actual sensor reading

        // Apply Kalman filter
        filtered_light_level = kalman_filter.update(
    noisy_reading)

        if lightstart(filtered_light_level,
    noisy_reading):
            all of your code here
```

In the main function, initialise two variables. Then, enter an endless while function. In this loop, continually reassign the variable value. Variable 1 gets the raw sensor readings, while variable 2 gets the value that the KalmanFilter method returns with variable 1 as its parameter. Then, call the lightstart function with variable 2 and variable 1 in an if-statement that, if true, breaks out of the aforementioned loop.

D. Sensor fusion

Sensor fusion can also be used in combination with any proposed solution. Sensor fusion involves the use of multiple sensors and the achievement of consistency between the values. This can be quite tricky, a simple but flawed example written in pseudocode can be found here;

```
lightstart_function((sensor1.read() +
sensor2.read())/2)
```

Other algorithms besides the simple arithmetic mean can be used, depending on the type of functions intended. There may not be enough space to place two sensors next to each other to aid each other in recognizing the signal sent out by the lamp. This section describes the potential uses of the subject matter but is too unstable for use in programming a robot's start time.

E. Moving Average

Another approach you could take (which can not be combined with the other shown here) would be the moving average filter. Although there are different variations such as the weighted or cumulative moving average, its simplest form would look like this [10]:

$$y_n = \frac{1}{2}(y_{n-1} + x_n)$$

where x_n is the incoming value and y_n, is the newest value, subsequently making y_{n-1} the last result of the function. This filter, much like the Kalman filter, takes previous steps and the new value into account, but unlike the Kalman filter, it only works with the absolute value of the last and current step, and doesn't need any temporary variables to update every step, hence it uses a step less than the Kalman filter. This may lead to a slight increase in performance, even if it can be considered negligible.

V. Conclusion

Using the Kalman filter in addition to mounting the light sensor vertically is probably the best way to go about solving your light-related problems. Sticking with the black tape, as KIPR suggests [2], can be done additionally and will probably help, but at the cost of an added risk of damaging your light sensor if the tape gets stuck, because if you try to remove it by force, it could tear the sensor. Recall that it also depends on the specific situation. Therefore, you should try a healthy mix of all the options shown in this paper, as well as some efficient ones from your imagination. There is no one-size-fits-all approach, you have to decide what works best for your setup.

Acknowledgement

The authors would like to thank Dr. Michael Stifter for supporting the paper in providing both writing and technical assistance. We would also like to thank Dipl.-Ing. Harald Haberstroh, who has so kindly offered his skills and years of knowledge to robo4you.

References

[1] "Light sensor." https://botball-swag.myshopify.com/products/light-sensor.

[2] Miller *et al.*, "Kiss institute for practical robotics botball kit documentation." https://www2.seas.gwu.edu/~ece001/6usefulLinks/HandyboardProgrammingManual.pdf.

[3] KIPR, "Sensor and motor manual." http://files.kipr.org/ebc/Source/resources/Sensor_and_Motor_Manual_BB2013-no_camera.pdf.

[4] V. Ryan, "Potentiometer/variable resistor." https://technologystudent.com/elec1/vary1.html.

[5] "Light sensors." https://www.electronics-tutorials.ws/io/io_4.html.

[6] PRIA, "2023 botball game table build." https://ecer.pria.at/wp-content/uploads/2023/02/2023-Game-Table-Full-Build-v1.2.pdf.

[7] KIPR, "kipr/libwallaby." https://github.com/kipr/libwallaby.

[8] I. Hönigmann, M. Eiwen, M. Guzmits, C. Kauhofer, P. Kain, and C. Schnabl, "Sensor based one-way communication in multiple mobile robot systems: an experiment." https://robo4you.at/publications/sensor_based_communication.pdf.

[9] Mathworks, "Understanding kalman filters." https://de.mathworks.com/videos/series/understanding-kalman-filters.html.

[10] D. Swoboda *et al.*, "A comprehensive study of simple digital filters for botball ir detection techniques." https://robo4you.at/static/88f989ab8bb2a7eab7ae4f6f15b1baaa/Study_of_Digital_Filter_Techniques.pdf.