Computer Vision in Botball - Potential & Limits

Koenigsreiter Simon, Lampalzer Alexander, Maschek Linda, Himmer Fabian, Grill Matthias
MMag. Dr. Stifter Michael
Department for Computer Science
Secondary Technical College, HTL Wiener Neustadt
E-Mail: simon.koenigsreiter@gmail.com, sascha@localhost.systems

***Abstract*—The eyes are one of the most important parts for our perception, so its not far fetched to try and use cameras to enable robots to see as well. This paper, which was written during preparation, for the Botball competition outlines some of the difficulties as well as some of the strengths of using cameras for object detection and distance calculation. In order to achieve this goal several experiments about the most common problems were conducted. These showed, that the camera and the wallaby controller included in the 2018 Botball kit were well suited for detecting a number of objects at the same time, with and without movement. However, the differentiation of objects of the same color is still a big problem. Using a camera is most certainly an opportunity each team should look into.**

***Index Terms*—Optical Cameras, Wallaby, Robotics, Botball**

INTRODUCTION

Since the end of the 20th century there has been a massive increase in computation power of modern embedded processors [1], which enables more advanced applications to utilize near real-time image processing. Understanding and using this technology may be an important asset in the career development for every robotics student.

This years' Botball competition challenges students to use cameras and sort game objects based on their color by requiring the robot to place foam cubes of the same color in a zone marked by a smaller foam cube. In the process of evaluating the usability of the mentioned optical sensor in combination with libwallaby – a software library provided by KIPR – a variety of tests was run which documented the limitations of cameras beyond simple object and color detection.

This publication paper assesses the usability of the digital camera, which is included in the 2018 Botball kit, based on the proposed usage and specification in the Botball robotics program. It may also serve as a guide for future Botball teams which are assessing the use of an optical camera as a part of their robot. This paper is going to present a comprehensive overview of strengths and weaknesses of cameras as used as sensors for the Botball competitions.

I. OVERVIEW OF THE HARDWARE

A. The controller

The Wallaby controller was chosen as it is the only official controller for the Botball competition. It has a single core ARMv7 processor running at a minimum frequency of 275 MHz and a maximum of 720 MHz. Furthermore the controller includes 512 Mb of onboard RAM.

B. The camera

All experiments were conducted using the Logitech C170 camera which has a resolution of 640x480 pixel and a 1.3 megapixel sensor. The diagonal field of view measured 58°. [2]

Fig. 1. An image of the camera used throughout the experiments [2]

II. OVERVIEW OF THE SOFTWARE

A. Libwallaby

The core functionality of the experiments were provided by the libwallaby. It is a software library used to interface with the Wallaby firmware to read sensor values, move motors and other utility functions. It is based on the so called libkovan which are both licensed under the GPL-3.0. This library was published on Github [3]. The following tests use the latest release available since the 15th of January 2018. [4]

III. Experimental Evaluation

The following tests were executed in the context of a Botball competition, the hardware was taken from the official Botball kit. Furthermore the experiments were conducted inside a basement room with no windows and fluorescent tubes overhead, therefore always under the same light conditions.

A. Evaluating the camera tracking range

This experiment evaluated the tracking range of the Logitech C170 in combination with libwallaby while it was mounted on a table. The target is a 4.445 cm long, wide and high cube.

The tracking range was measured by slowly moving a green foam cube away from the camera until no object detection was possible anymore. The foam cube had an edge length of 4.445 cm. The result shows that this camera is good in detecting objects over relatively long distances which may prove to be vital for applications using camera images as their main source of visual information.

The expected range was at least the length of a game-table which measures 2.47 m. The actual result was 3.50 m which was 1 m above the expected value.

B. Distinguishing colored blocks

In this experiment the ability of libwallaby to detect multiple foam blocks based on their color was tested. For this experiment the blob tracking algorithms of libwallbay were used.

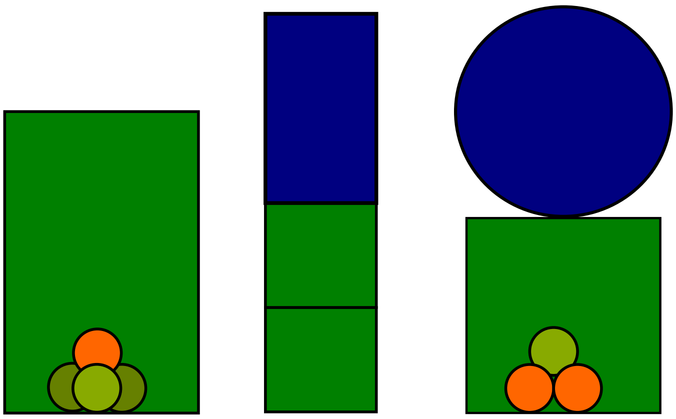

Fig. 2. Schematic setup of the experiment. (Frontal view)

It was expected of the library to easily detect a change in color and to observe the combined edges of each cube and the edges of same colored cubes.

The experiment showed that the library is capable of finding the edges of differently colored cubes but as soon as they are stacked, cubes of the same color appear as one big cube.

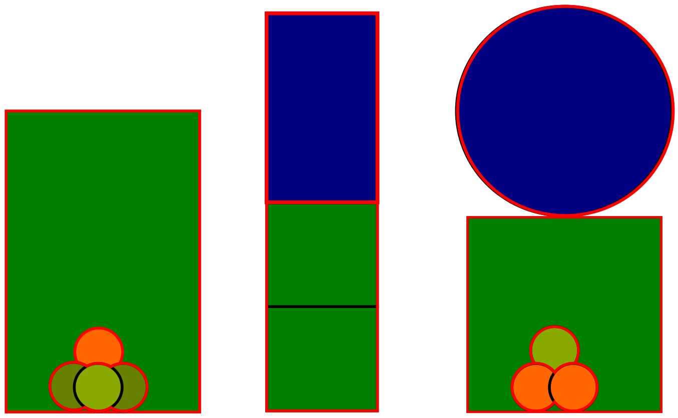

Fig. 3. The results of the experiment where the red edges represent the detected edges by the program

C. Differentiating colored poms

The purpose of the third experiment was to distinguish small colored balls which are referred to as poms.

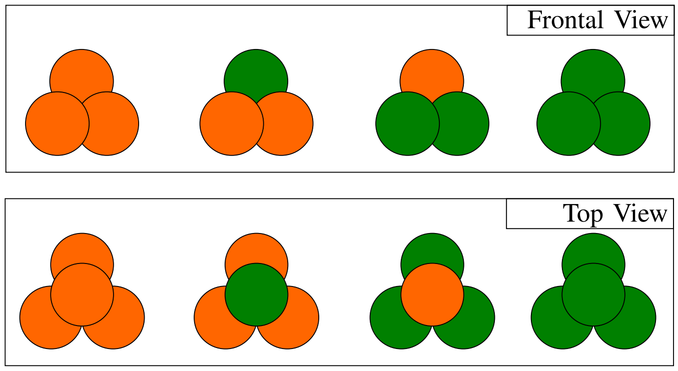

Fig. 4. Frontal and top view of the experiment

The expectations were that libwallaby is capable of detecting a change in color and detecting multiple poms.

As expected there were no problems identifying different colors, however it was impossible to differentiate poms of the same color, independent of the distance between the camera and the poms.

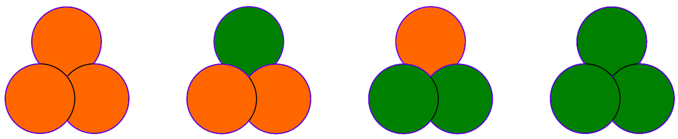

Fig. 5. The purple edges represent the edges as detected by the camera.

D. Calculating the distance based on variation of the frontal view dimensions

This experiment intended to determine how precise the camera is as a means to measure the distance to an object. Therefore a 101.6 by 101.6 millimetre cube was placed 20 cm away from the camera and an image was taken as a reference.

Afterwards the same foam cube was moved 10 cm further away from the optical sensor. After the second image was taken the difference between the area inside the bounding box of the first cube and the second cube was calculated. The result was divided by ten to calculate the change of area for one centimeter of distance change. For the next iteration a new cube with the same dimensions was placed 25 cm centimeters away from the camera.

As the resolution of the camera is on the lower end of the current technology standard it is a realistic assumption that the accuracy of the distance calculation is in the range of plus/minus one centimeter.

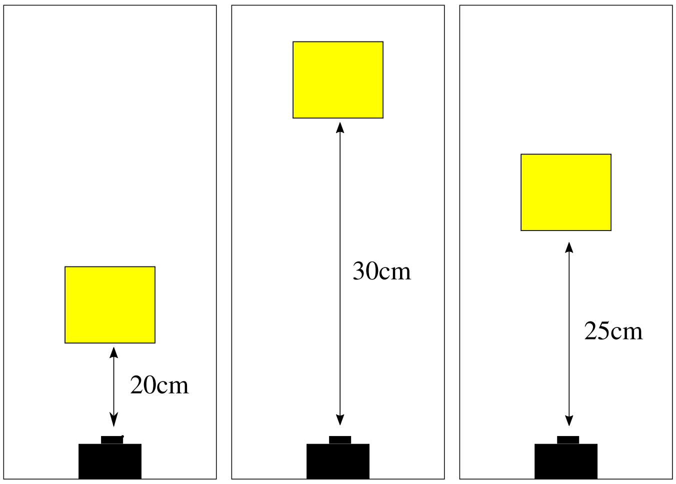

Fig. 6. Top view where the black boxes represent the camera and the yellow squares the 101.6 by 101.6 millimetre cube

To get a more representative value for the measurements, the test was repeated ten times for each distance. The recorded values for each indivdual distance are shown in table I.

The average as well as the median was considered in the following experiments, however they only differ marginally (50 $pixel^2$). Furthermore the median is more robust than the mean.

For the following formulas the median change in $pixel^2$ per centimeter is needed. It was calculated by subtracting the median value of 20 cm from the median of 30 cm. The difference was divided by ten to get the value.

$$\begin{aligned} &\frac{med_{30cm}\, pixel^2 - med_{20cm}\, pixel^2}{10\, cm} \\ =&\frac{9024\, pixel^2 - 4795.5\, pixel^2}{10\, cm} \\ =&422,85\, \frac{pixel^2}{cm} \end{aligned} \tag{1}$$

This value represents the median change per centimeter over the entire number of test case where:

- $med_{20cm} = 4795.5\, pixel^2$
- $med_{25cm} = 5814\, pixel^2$
- $med_{30cm} = 9024\, pixel^2$

If the computed value is now multiplied by five the following result will show up:

$$422.85\, \frac{pixel^2}{cm} * 5\, cm = 2114.25\, pixel^2 \tag{2}$$

To get the theoretical value for the distance of 25 centimeter the new value will be added to the median of 20 centimeter:

$$\begin{aligned} &med_{20cm}\, pixel^2 + 2114.25\, pixel^2 \\ =&4795.5\, pixel^2 + 2114.25\, pixel^2 \\ =&6909.75\, pixel^2 \end{aligned} \tag{3}$$

To get the median variance of the two values a subtraction will calculate the median delta:

$$\begin{aligned} \Delta &= theoretical\, pixel^2 - med_{25cm}\, pixel^2 \\ \Delta &= 6909.75\, pixel^2 - 5814\, pixel^2 \\ \Delta &= 1095.75\, pixel^2 \end{aligned} \tag{4}$$

And to calculate the deviation over ten runs the following formula is suitable:

$$\frac{1095.75\, pixel^2}{422.85\, \frac{pixel^2}{cm}} = 2.59\, cm \tag{5}$$

This shows that there is a significant change over the course of ten iterations of the same experiment. Unfortunately there is no other camera available to compare values.

E. Number of poms to be detected at the same time

This experiment should test the limits of the wallaby controller and the logitech camera in a scenario, where the target objects are moving aside the camera. To achieve this the controller and camera were mounted on a vehicle called iRobot Create 2 which is driving along a black line. A number of poms were laid out along the 240.03 cm long black line. The distance between the individual poms varied over the length of the black line. At each iteration of the experiment the speed of the vehicle was increased incrementally until the maximum speed of 500 millimeters per second was reached. [5]

The assumption is that the controller has enough computational power to detect all poms at a low speed whereas it was expected that not all poms are detected at a higher speed of the vehicle as the camera may not be able to capture pictures in a suitable quality to accurately detect the pom.

After ten iterations at various speeds – ranging from the iRobot Create 2 minimum speed to its maximum speed – the experiment was successful in every aspect. Each test run had a success rate of 100% of detecting every pom placed on the black line.

Fig. 7. Top view of the experiment where the black box represents the camera and the green circles the poms; The camera moves from the right to the left with a constant speed

F. Necessary resolution to detect a cube

The last experiment was designed to show at which resolution it is still possible to detect a cube. A simple pseudo-code example was designed and a possible solution can be found in Appendix (VI-A).

There is currently no functionality implemented in the libwallaby to scale down an image. After evaluating the complexity of a possible implementation in libawllaby it was decided that the effort for this functionality would by far exceed the scope of this paper.

IV. Conclusion

The results of the experiments showed that the camera is a versatile sensor in combination with the libwallaby. The object tracking algorithms have proven to be very good in tracking objects over a large distance, even for smaller targets. It is also possible to capture pictures with sufficient quality for object tracking at the maximum speed of the "iRobot Create 2". The color detection is reliable for Botball competition applications. It is also possible to use the Logitech C170 to calculate the distance to an object, although it is not advised, as it is not as accurate as an ET-sensor or a Tophat-Sensor for example.

V. Acknowledgement

First of all the team would like to thank the administration of the HTL Wiener Neustadt for all the financial and infrastructural support.

Furthermore the authors of this paper are grateful to Daniel Swoboda for proofreading this paper throughout the whole writing process.

Additionally the team extends warmest thanks to our professor Harald Haberstroh who has been an integral part of the team, inspiring us and supporting in finding solutions to a multitude of problems that arose during the Botball season.

Last but not least the authors thank Lord Michael Stifter for his continuous support throughout the whole season and for providing us with valuable feedback on the paper.

VI. Appendix

A. Pseudo Code for downscaling an image

```
camera = CameraDevice
resolutionX = 160
resolutionY = 120
applyResolution(camera,
    resolutionX, resolutionY)
while resolutionY > 1 &&
    detectCube(camera)?:
resolutionX = resolutionX - 4
resolutionY = resolutionY - 3
applyResolution(camera,
    resolutionX, resolutionY)
printf("The resolution where a
    cube can't be detected is:
    %dx%d", resolutionX,
    resolutionY)
```

Listing 1. Algorithm to decrement the resolution until the resolution is smaller than 0 or no cube can be detected

TABLE I
TEST RESULTS FOR THE DISTANCE CALCULATION

	20cm	***25cm***	***30cm***
Test run 1	4761 $pixel^2$	5776 $pixel^2$	9024 $pixel^2$
Test run 2	4830 $pixel^2$	6006 $pixel^2$	9024 $pixel^2$
Test run 3	4761 $pixel^2$	5776 $pixel^2$	8928 $pixel^2$
Test run 4	4830 $pixel^2$	6006 $pixel^2$	9024 $pixel^2$
Test run 5	4899 $pixel^2$	5928 $pixel^2$	8928 $pixel^2$
Test run 6	4761 $pixel^2$	5776 $pixel^2$	9024 $pixel^2$
Test run 7	5040 $pixel^2$	5776 $pixel^2$	9024 $pixel^2$
Test run 8	4761 $pixel^2$	6006 $pixel^2$	9024 $pixel^2$
Test run 9	4760 $pixel^2$	5852 $pixel^2$	9024 $pixel^2$
Test run 10	4968 $pixel^2$	5776 $pixel^2$	9024 $pixel^2$

Results over ten test iterations.

TABLE II
A SHORT SUMMARY OVER TEN ITERATIONS

	20cm	***25cm***	***30cm***
Minimum	4760 $pixel^2$	5776 $pixel^2$	8929 $pixel^2$
Maximum	5040 $pixel^2$	6006 $pixel^2$	9024 $pixel^2$
Median	4795.5 $pixel^2$	5814 $pixel^2$	9024 $pixel^2$
Average	4837.10 $pixel^2$	5867.80 $pixel^2$	9004.80 $pixel^2$

References

[1] S. B. Chien, Andrew A., "The Future of Microprocessors," May 2011. [Online]. Available: https://cacm.acm.org/magazines/2011/5/107702-the-future-of-microprocessors/fulltext

[2] "Webcam C170." [Online]. Available: http://support.logitech.com/en_us/product/webcam-c170/specs

[3] B. McDorman and N. Zaman, "libkovan: Standard library for the kovan robot controller," Feb. 2017, original-date: 2012-07-16T17:15:04Z. [Online]. Available: https://github.com/kipr/libkovan

[4] J. Southerland, "libwallaby: Library for interfacing with the "wallaby" firmware on the 2016 Botball Robot Controller," Nov. 2016, original-date: 2015-11-02T17:39:27Z. [Online]. Available: https://github.com/kipr/libwallaby

[5] (2018, apr) irobot create 2 open interface specification based on the irobot roomba 600. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/create_2_Open_Interface_Spec.pdf