DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Interactive Wall Tracking System for Digital Sports Games

Janosch Landvogt

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics: Games Engineering

Interactive Wall Tracking System for Digital Sports Games

Interaktives Wandtrackingsystem für digitale Sportspiele

Author:	Janosch Landvogt
Supervisor:	Prof. Gudrun Klinker
Advisor:	M.Sc. Sandro Weber
Submission Date:	16th of August 2021

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 16th of August 2021 Janosch Landvogt

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Gudrun Klinker and my advisor, M.Sc. Sandro Weber, for their continuous guidance, support and availability throughout this thesis.

I would also like to thank the indoor playground rabatzz! for their support during development and for providing space and resources to conduct the prototype testing.

Abstract

Location-based entertainment centres, such as indoor playgrounds, are well-known for the use of plastic balls in ball pits as part of their entertainment. Furthermore, these centres are often built in large halls, where unused space can be found on and near walls.

In this thesis, a system is analysed that makes use of such free space on walls together with plastic balls to enable digital sports games in this context.

This system overall consists of a projector that projects a game environment onto a wall. The plastic balls are then thrown onto the projection surface. The entire process is filmed by a camera that detects the balls, tracks them and calculates the corresponding impact points to enable correct interaction with the game.

For the detection and tracking of the balls, standard real-time computer vision techniques of the OpenCV library are applied to implement system.

The proposed system is then put into comparison with a laser grid based approach.

Zusammenfassung

Standortbezogene Unterhaltungszentren, wie z.B. Indoor- Hallenspielplätze, sind bekannt für die Verwendung von Plastikbällen in Bällebädern als Teil ihrer Unterhaltung. Zudem werden diese Zentren häufig in großen Hallen gebaut, wobei ungenutzter Platz an und nahe Wänden existieren kann.

In dieser Thesis wird ein System analysiert, welches den beschriebenen freien Platz an Wänden und Plastikbälle nutzt, um digitale Sportspiele in diesem Kontext anzubieten.

Dieses System besteht dabei aus einem Projektor, der ein Spiel auf eine Wand projiziert. Die Plastikbälle werden auf die Projektionsfläche, die das Spiel zeigt, geworfen. Dabei wird der gesamte Prozess von einer Kamera gefilmt, welche die Bälle erkennt und die entsprechenden Aufprallpunkte für die Interaktion mit dem Spiel berechnet.

Für die Erkennung und Verfolgung der Bälle werden gängige Echtzeit Computer Vision Techniken der OpenCV Bibliothek für die Implementierung des Systems genutzt.

Das vorgeschlagene System wird mit einem auf einem Lasergitter basierenden Ansatz verglichen, im Gegensatz zu seinem kamerabasierten Ansatz.

Contents

Acknowledgments **iii**

Abstract **iv**

Zusammenfassung **v**

1. Introduction **1**
1.1. Motivation . . . 1
1.2. The Camera Based System . . . 1
1.3. Thesis Structure . . . 1

2. Related Work **3**
2.1. Comparable Systems . . . 3
2.1.1. Laser Grid . . . 3
2.1.2. Laser Grid Drawbacks . . . 3
2.1.3. Laser Grid Advantages . . . 4
2.2. Camera Detection Systems in Ball Sports . . . 4
2.3. OpenCV . . . 4

3. Initial System Setup **6**
3.1. Projector Placement . . . 6
3.2. Camera Positioning . . . 7
3.3. Camera Calibration . . . 9
3.4. Checkerboard and Camera Image Calibration . . . 11

4. Detection and Tracking Algorithm **14**
4.1. Blurring . . . 14
4.2. Background Subtraction . . . 19
4.3. Color Filtering . . . 21
4.3.1. HSV Color Space . . . 21
4.3.2. Lighting and Value Calibration . . . 24
4.3.3. Rainbow Effect . . . 25
4.4. Choice of Ball Type . . . 29
4.5. Choice of Ball Color . . . 30
4.6. Resolution Optimization . . . 33
4.7. Contour Analysis . . . 36
4.7.1. Erosion and Dilation . . . 36

4.7.2. Contour Detection 36
4.7.3. Obstacle Detection 37
4.7.4. Single Ball and Multiple Ball Detection 37
4.8. Position Analysis 37
4.8.1. Viewing Angle Error 38

5. Conclusion **41**
5.1. Future Work 41
5.1.1. Depth Sensor and Camera 41
5.1.2. Resolution 42
5.1.3. Artificial Intelligence 42

A. General Addenda **43**
A.1. Figures 43

List of Figures **44**

Glossary **46**

Bibliography **47**

1. Introduction

1.1. Motivation

Digitization is a topic that constantly affects our lives. Digitization takes place everywhere and impacts all kind of areas, including the entertainment sector which is predominantly digital by now through the constant rise of video games. They have become by far the biggest form of entertainment when judging by revenue. [1] So it is obvious to expect a high popularity in this sector, especially when considering a younger audience.

This popularity can be leveraged to raise interest also in other areas of entertainment such as sports activities in location-based play centers. These centers are an established form of family entertainment throughout the world. Even by looking only at the VDH (Verband der Hallen- und Indoorspielplätze/Association of indoor and indoor playgrounds) there are already 79 member locations listed throughout Germany. [2] These members consist mainly of centers commonly known as indoor playgrounds.

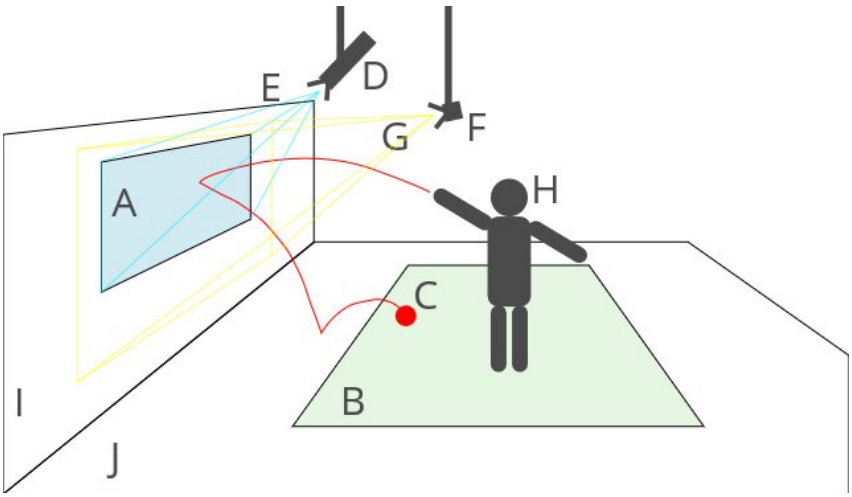
A constant feature in many of these indoor playgrounds is the use of ball pit plastic balls as part of their entertainment. Since indoor playgrounds are always looking for new playground equipment to fit in free spaces, the initial shared interest in ball games among their outlets can be used to create a system that enables visitors to play digital sports games. Physical activity with a fun approach, mental and physical health improvements or even learning new theoretical or physical skills during play are a potential goal for this system. There are comparable systems already existing where a laser grid based approach is used for the ball tracking. This approach has some drawbacks that could be overcome by a camera based approach.

1.2. The Camera Based System

The proposed system is based on the input mechanics of throwing plastic balls against the Wall Impact Area (WIA). (See figure 1.1) The WIA is defined by the projected image of a projector. In a similar position to the projector a camera is mounted that records the WIA and its surrounding space. The video stream of the camera is then used to detect the balls thrown at the wall and the impact points are calculated through image analysis.

1.3. Thesis Structure

The second chapter with related work introduces the comparison to the grid based system and similar techniques used in ball sports. The technical base represented by OpenCV is also



A: Projector image of D and area where the impacts of ball C are supposed to interact with the system (referred to as WIA); B: Movement area for the player H during game. Player may leave the area to get ball; C: Thrown ball with trajectory line (red); D: Projector projecting A; E: Projection volume of D; F: Camera facing A; G: View frustum of F; H: Player throwing ball; I: Wall; J: Floor;

Figure 1.1.: Visual System description.

introduced there.

The third chapter and initial system setup explains how the hardware is configured and introduces techniques and input values for the camera calibration.

The fourth chapter and main part of this thesis, evaluates the detection and tracking algorithm, and depicts the whole detection and tracking process of the system. Each subsection identifies a problem as it occurs in order of the algorithm, presents the necessary background knowledge and potential problem solving and concludes with the implementation of the solution identified for this use case.

The fifth and last chapter summarises the research findings of this thesis and projects possible future work.

2. Related Work

2.1. Comparable Systems

2.1.1. Laser Grid

There are particular systems that base their tracking on a laser grid approach. A range of small photoelectric sensors are placed inside a rectangular frame. The size of the frame determines the size of the WIA. The number of sensors placed inside the frame determine the grid size and the accuracy of the impact detection. An impact is detected whenever a ball hits the wall and the light beam is interrupted. A defined number of sensors is chosen, so that the ball always interrupts at least one of the horizontal and one of the vertical light beams. The impact point can then be calculated at the intersection of the interrupted light beams.

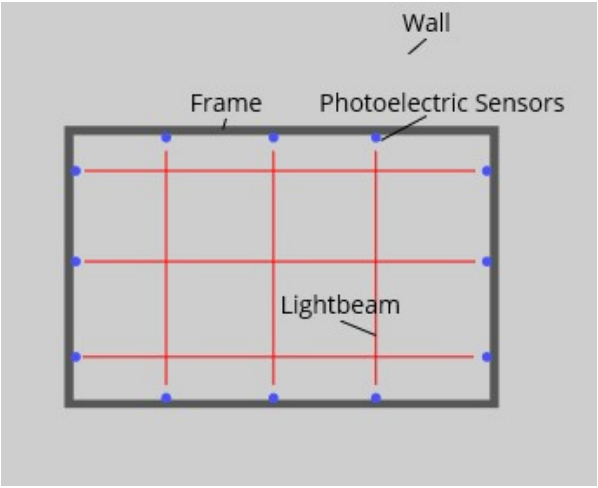


Figure 2.1.: Scheme of a laser grid based system

2.1.2. Laser Grid Drawbacks

There are some drawbacks to this approach that are not present in a camera based version. A camera based approach does not require any mounting on the wall, so there is no need for an actual frame. This has the advantage of being able to choose a WIA size that is then only restricted by the camera's field of view and the projector's image capabilities. In case of the system being to be moved to another wall, the camera can provide the option to change the WIA size very dynamically and without the need to buy a new frame. Not having to rely on

a frame also lowers maintenance demands, as especially the bottom part of the frame will constantly collect dust and needs to be cleaned regularly. Otherwise the dust may interrupt the light beam and trigger wrong system inputs.

Another cause with regards to wrong system inputs is the decalibration of light beams. If the light beam emitter is moved by a tiny fraction, for example by a bump on the frame, the light may no longer hit the sensor, causing the connection to be constantly interrupted. This leads to a complete failure of the system and requires a time consuming recalibration, whereas the camera image recalibrates itself at every system startup and is resistant to tiny movements.

A laser grid based approach cannot distinguish players by the color of the balls thrown without additional hardware. A multiplayer environment is only achievable when the WIA is split into one area per player. This significantly shrinks the interaction space per player and limits gameplay options, hence not representing an optimal setup.

2.1.3. Laser Grid Advantages

Due to the detection of objects through light beam interruption, almost anything can be used instantly and simultaneously as an input device. That may include player's hands, a large range of balls and other physical input objects.

As the grid does not have to rely on camera input, users cannot block the camera view and they are able to walk around freely and anywhere in front of the WIA without interrupting the detection and flow of the game.

2.2. Camera Detection Systems in Ball Sports

Detecting and tracking a ball during sports through video cameras is a well known problem that has been dealt with many times before. Besides Golf, Soccer, Football and many other ball based sports, tennis is also one of the sports where detection and tracking of a ball may be of interest. [3] During tennis matches, very high ball speeds can be achieved that are difficult to follow with the naked eye. To make umpiring easier and more reliable, technology such as the Hawk-eye system has been introduced to measure exact impact points of tennis balls. [4]

2.3. OpenCV

The OpenCV library will provide the programming backbone for this thesis focusing on computer vision related tasks. OpenCV is an open source computer vision library that provides a wide range of optimized algorithms that will be utilized to implement the detection and tracking of thrown balls, calibrate cameras, manipulate images and other computer vision related tasks. OpenCV is used by a large number of experts and companies and is well established in its field. The code is well documented, many tutorials and examples are offered and support for the Python programming language is given which makes OpenCV a strong

choice to use for this thesis. [5][1] Python will be the choice for implementing the system because it is equally well documented for the use with OpenCV, it requires few overheads and enables a fast prototyping.

[1] OpenCV - About

3. Initial System Setup

For the system to work as intended the setup process is essential. The following explains what to consider when placing the projector and the camera and how to calibrate the camera image.

In the WIA detection phase, the area that is supposed to be the interaction area is filled by the image of a calibration checkerboard that is shown by the projector. This is where the balls are thrown at during play. This checkerboard is used to calculate the camera calibration which will be discussed in section 3.3.

3.1. Projector Placement

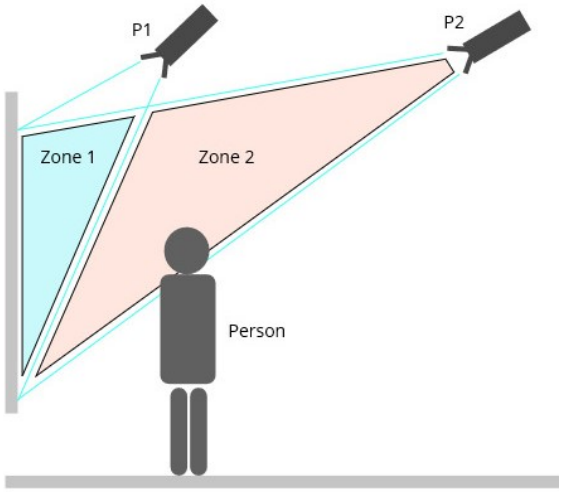


Figure 3.1.: Illustration of two different projector setups

As seen in figure 3.1, P1 is a short-distance projector with the technical capacity of projecting a clear image at a very steep angle close to the wall. The P1 projection volume is marked with Zone 1. P2 is a projector that needs a greater distance to the wall. Therefore the projection volume is larger than that of P1 and consists of Zone 1 and Zone 2 combined. In the figure, a person is shown blocking the P2 projection but P1 is not affected.

One of the advantages of a camera based approach, is the freedom and tolerance in positioning the camera and projector. Furthermore, the choice to position the projector itself,

is less limited than the one for the camera. In general, the positioning of the projector is mostly guided by safety, practicality and image optimization. If the projector is placed in such a way, that the image resolution is impaired, it will still only be a visual problem. If the projection is still aligned on the wall, the sub-optimal resolution will not affect the ball detection. The projector should be mounted outside the player's movement area to avoid contact during play. Therefore a position above and out of reach of the player is the most reasonable choice. Also, and accordingly to the capabilities of the device, it should be placed as centrally and close to the wall as possible. The proximity to the wall has the advantage that the players can move closer to the wall without casting a shadow on the WIA when blocking the projection image. In addition, the volume in which the balls thrown cast a shadow is reduced and the time in which the shadow is visible, is kept to a minimum. This is further illustrated in figure 3.1. This is essential because the background subtraction algorithm which is explained in the corresponding section, will recognise the shadows movements as rarely as possible. It is an advantage in this setup that the projector can compensate for horizontal displacement in the projection image, as this displacement correction provides more space for the camera to be placed with a central view of the WIA if necessary.

3.2. Camera Positioning

There are more things to consider when choosing the position of the camera than when choosing the position of the projector. The camera must be placed in the center of the WIA so that the image resolution and thus the precision of the ball detection are symmetrically distributed over the WIA. The resolution of the WIA can not be evenly distributed due to the nature of camera lenses. The fact that the sides of the WIA are further away from the camera than the center also results in a lower resolution towards the WIA borders. If the camera was not centered on the WIA, the ball detection would be more accurate on one side than the other. Consequently, the camera would be closer to one of the two edges of the WIA and then result in a better resolution in the camera image on that side. This would give an unfair advantage to one player, especially if two players were playing on two horizontal halves of the WIA. It would distort the playing experience and should be avoided.

The actual distance to the wall is also important when choosing the camera positioning. The camera must not be placed too far away from the wall, even if the field of view can be increased that way, since the resolution of the WIA will worsen the further away the camera is placed from the wall. Also, the camera must not be placed too close to the wall, as the height of the ball's impact on the WIA is used to calculate the impact position. If the camera is very close to the wall, it is very easy to see when the ball hits the wall by the changes in flight direction. But it is hard to determine at what height in the WIA the ball impacts. On the other hand, the viewing angle of the camera to the WIA is important for the already mentioned detection of changes in flight direction. As described, the camera will have a very good view on the change in flight direction when placed close to the wall facing straight down. The opposite would be a central placement of the camera in front of the WIA filling the field of view evenly. That way the camera would have a very good view on the height of the ball's

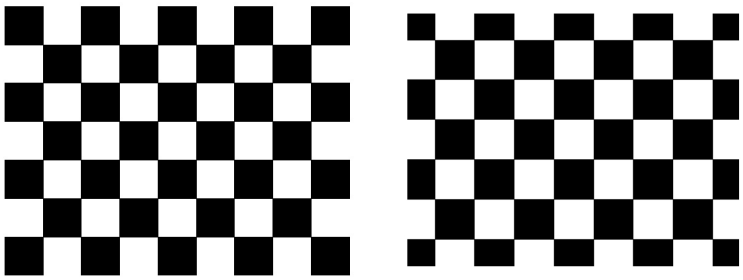
impact position but the change in flight direction after the impact would be difficult to see, especially with high throwing speeds. In addition, the camera would obstruct the view and flight path to the wall.

Therefore a position needs to be found, that optimizes both views on the WIA and the change in flight direction. During the development and testing of the system it was determined that the view on the WIA is more important to be optimized because the error viewing angle on the ball is more difficult to compensate than the detection of changes in flight direction. This is further described in section 4.8.1. Therefore, it is recommended to choose a viewing angle as close to orthogonal to the wall as possible.

For the positioning of the WIA in the camera view, a border around the sides and bottom of the WIA is of importance. The upper border of the WIA can be aligned with the upper border of the camera image because on average, there are no ball throws expected during play that leave the camera view frustum to the top. As speed and precision are the goals in a throwing motion, a straight, overhand throw style is expected with no high trajectory parabolas occurring. A border to the sides and to the bottom is necessary for the calculation of the impact point, explained in section 4.8. There need to be at least two detected frames of a ball recorded before the ball may impact inside the WIA. Therefore, the shortest distance from the border of the camera view frustum to the border of the WIA on the wall must be at least long enough to allow a ball, thrown from inside the movement area for the player, to be recorded by a minimum of two frames before impact.

The range of ball speeds to calculate the shortest distance is virtually impossible because a variety of different age groups can play with the system, especially in location based entertainment centers. These players all have very different throwing speeds depending on skill, height, physique, throw style and so on. That is why an optimal border size seems difficult to be calculated. But since the result of the camera setup can be tested and adjusted freely at any time, a few test throws will show whether those close to the edge are detected correctly. Plus, throws from the very edge of the movement area to the very edge of the WIA rarely happen, especially when compared to throws closer to the center and this problem does not present itself anymore.

3.3. Camera Calibration



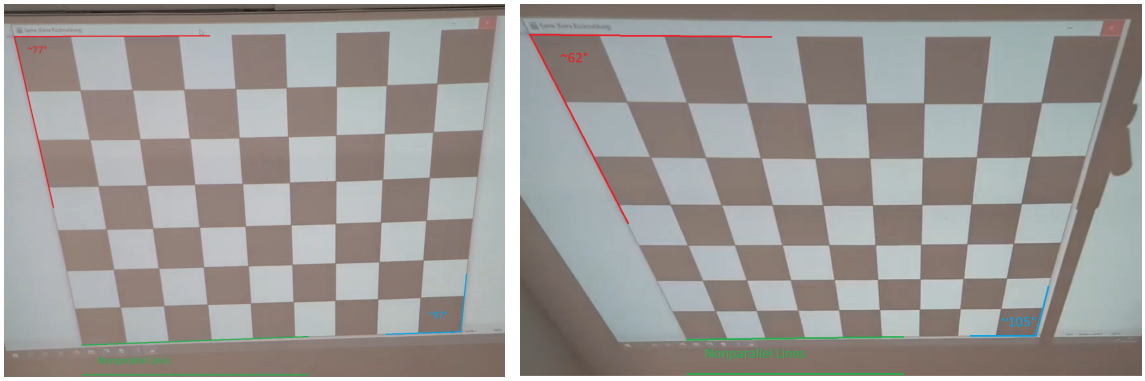
(a) Simple 9 by 7 checkerboard

(b) 9 by 7 checkerboard with inserted white border

Figure 3.2.: Example of checkerboard images to be shown by the projector

When the projector and camera are set up in their final position, the next step is to calibrate the camera image. Whilst the camera records the the WIA and its surroundings, and due to the specifics of camera optics, the recorded image appears distorted. This distortion can be visualised by projecting a checkerboard image onto the WIA. It can be observed that the straight lines of the projected image appear rounded in the camera image. In addition, the initial checkerboard image has a rectangular shape with 90 degree corners. (See figure 3.2) In the camera image the rectangular shape appears trapezoidal. This effect depends on on the intrinsic values of a camera such as focal length and optical centers and the positioning of the camera in relation to the checkerboard. Also, the overall orientation of the camera leads to the fact that the border lines of the checkerboard are not lined up with the borders of the camera image. [5][1] The following figures 3.3 and 3.4 illustrate the result of the corrected camera calibration.

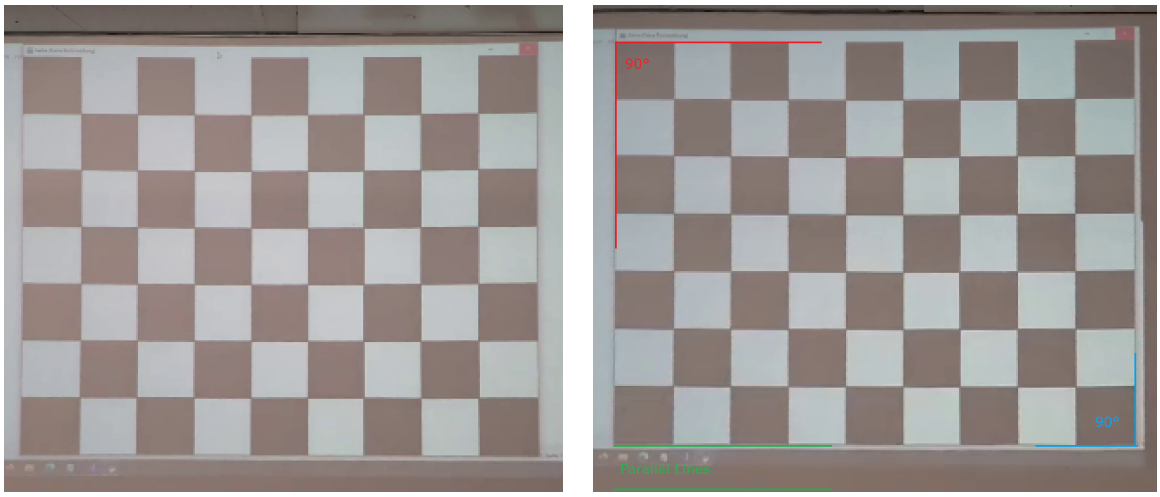
[1] OpenCV - Camera Calibration



(a) Camera image of the checkerboard with a *longer* distance to the wall and a *short* focal length depicting distortions. Red, top left: 77° corner angle; Blue, bottom right: 97° corner angle; Green, bottom left: Nonparallel Lines

(b) Camera image of the checkerboard with *shorter* distance to the wall and a *long* focal length depicting distortions. Red, top left: 62° corner angle; Blue, bottom right: 105° corner angle; Green, bottom left: Nonparallel Lines

Figure 3.3.: A comparison view between different camera positions and focal length settings



(a) Camera image as seen in 3.3a after the distortion introduced by the camera has been corrected. Now the checkerboard shape looks similar to the one shown by the projector 3.2 and is ready to be analysed further.

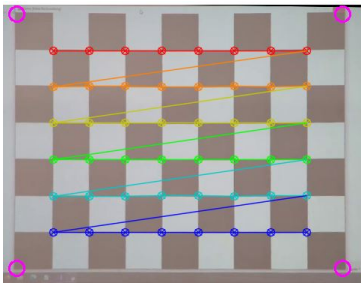
(b) Camera image as seen in 3.3b after the distortion introduced by the camera has been corrected. The correction is illustrated by the corresponding colored measurement features in reference to 3.3b and 3.3a. Red, top left: 90° corner angle; Blue, bottom right: 90° corner angle; Green, bottom left: Parallel Lines.

Figure 3.4.: A comparison view between different camera positions and focal length settings after being corrected through camera calibration transformation.

3.4. Checkerboard and Camera Image Calibration

The OpenCV camera calibration algorithm requires an input image of a reference checkerboard like figure 3.2 shows. The algorithm searches for the corner points of the checkerboards where the black and white squares touch each other's corners. This is achieved by several color and contrast manipulations to make them more visible for the algorithm detection. This begins not at the outer corner points where as few as only one square can be connected to a corner but at the inner corner points where always exactly four squares touch a corner point. This is further illustrated in figure 3.5. The detection algorithm requires a high contrast between the background in the camera image and the checkerboard itself.

During testing it was identified that the bright light of a projector makes the surrounding of the WIA appear very dark. When the checkerboard fills the entire WIA for calibration, the algorithm has trouble differentiating the black squares from the dark surrounding. Therefore, a white border was introduced to the checkerboard to increase the contrast of the black checkerboard squares to its surroundings and improve the checkerboard detection. Since the checkerboard is supposed to fill the entire WIA because it is used to determine the border of the WIA for later use, the white border was not placed around the existing checkerboard but inside the outer most squares of the checkerboard as shown in figure 3.2 (b). As the inner corner points are still fully visible, it introduced no detection problems when the outer squares where cut off by a white border. Due to the fact that all detected corner points have the same distance to each other, the actual outer border of the cut off squares can be reproduced by taking the distance between corner points as horizontal and vertical step distance for the square width. This makes the detection of the full WIA possible whilst also introducing a white border. [5][2]



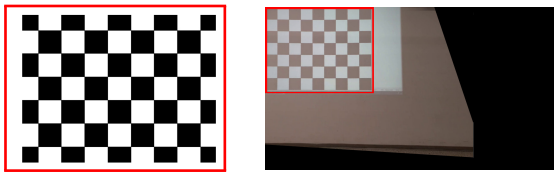
Simple 9 by 7 checkerboard

Figure 3.5.: Example of a detected checkerboard with highlighted inner corner points and calculated outer corner points

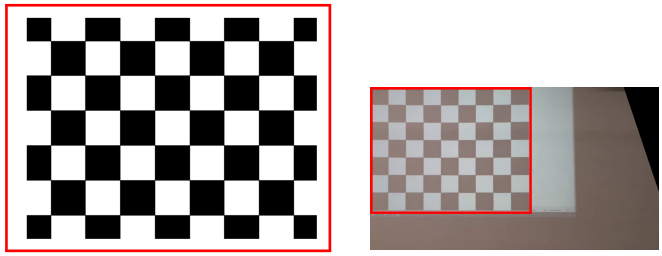
Following the requirement of the checkerboard, the next problem to consider is the adjustment of the checkerboard inside the checkerboard calibration image. The aforementioned corner points the detection algorithm searches for, need to be detected in both the checkerboard calibration image and the camera image. These corner points are then used to calculate a translation matrix that moves the corner points found in the camera image to the same

[2]OpenCV - Homography

orientation given by the checkerboard calibration image. If the resolution is the same for both images and the checkerboard fills the whole calibration image, the camera image will be translated so that the checkerboard of the camera image fills the whole camera image like the checkerboard image. The concept is further illustrated by figure 3.6. This process is very important for the correct detection and tracking of the WIA. This is also a procedure that can have very different inputs for each system setup and needs to be calibrated for each system individually.



Checkerboard: Lower resolution.
Camera Image: Checkerboard takes up less pixel space and left camera image side cut off.



Checkerboard: Higher resolution.
Camera Image: Checkerboard takes up more pixel space and more camera image border space cut off.

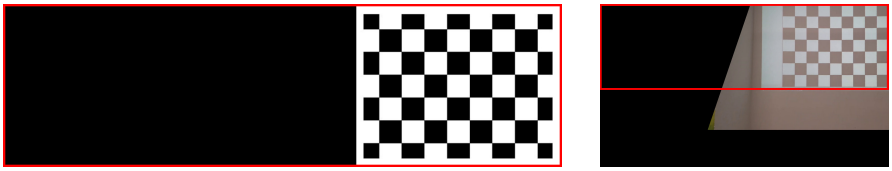
Figure 3.6.: A comparison view between different checkerboard calibration images and the resulting camera image transformations (Part 1)

An additional step to cut the camera image to size can also help to improve the algorithm quality. The aim is to cut out unnecessary space from the camera image due to the way the camera might be set up and record irrelevant areas.

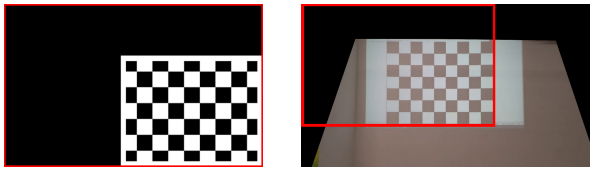
This calibration process for both the checkerboard calibration image and the camera image is done during the system initialisation. Therefore, values for the area of the camera image to be cut off are required. These values cut full lines of pixels from the outside for each side (top, bottom, left, right). So if an example value of 100 for the top and 50 for the right are given, the camera image will be cut 100 pixels from the top border and 50 pixels from the right border.

When the placement of the checkerboard needs to be determined, the desired resolution and border values for top and left need to be given. The resolution sates what resolution the image handled by the algorithm should be. It also implicates the pixel size of the checkerboard inside the camera image relative to the pixel size of the checkerboard calibration image.

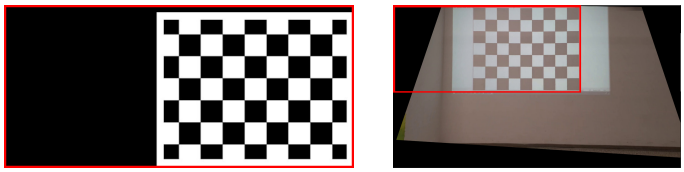
Further important factors for the choice of resolution size are explained in 4.6. To better understand the illustration of figure 3.7 it is important to understand that the resolution of the camera image the algorithm is working with, is not connected to the resolution of the checkerboard calibration image. The resolution of the checkerboard calibration image will affect the size of the checkerboard seen in the camera image. The border values for top and left determine the amount of pixels the checkerboard is offset from the top and left side of the checkerboard in the image which the algorithm works with.



Checkerboard: Large left side offset.
Camera Image: Checkerboard is moved to the right and left camera image side cut off.



Checkerboard: Left side and top side offset.
Camera Image: Checkerboard is moved to the center and too much empty space on top and camera image cut off at the bottom.



Checkerboard: Final checkerboard calibration image to use for the algorithm
Camera Image: Checkerboard is placed in such a way that the camera image fills the image centrally with all important features showing
The left side shows images of the input checkered calibration images and the right side shows the resulting camera image transformation for further use in the algorithm. The red outline depicts the border of each initial checkerboard calibration image (left side) laid over the camera image (right side).

Figure 3.7.: A comparison view between different checkerboard calibration images and the resulting camera image transformations (Part 2)

4. Detection and Tracking Algorithm

In this chapter, the procedure of the algorithm for detecting and tracking thrown balls inside a camera image is explained. Since the detection and tracking of balls is a well-known problem, the following sections explain the sub-algorithms that were chosen based on standard computer vision techniques and proven effectiveness in related work with computer vision that also have produced satisfactory results during the development of this system.

4.1. Blurring

When working in computer vision, blurring is often used to improve image detection. It seems counter-intuitive, to increase the robustness of computer vision algorithms by reducing the quality of an image by blurring it but when it comes to the detection of balls, reducing the image quality by blurring makes this process easier. When the not yet blurred image is filtered by a color filter that looks for certain colors and color patterns like the filter that will be discussed in section 4.3, several factors may worsen the result: The camera image itself has always a certain amount of noise in it. This noise is often introduced by a high ISO value, that might be necessary when using a relatively short exposure time. Blurring the image reduces that noise. There are also blur algorithms such as median blurring that handle ISO noise especially well. [5][1]

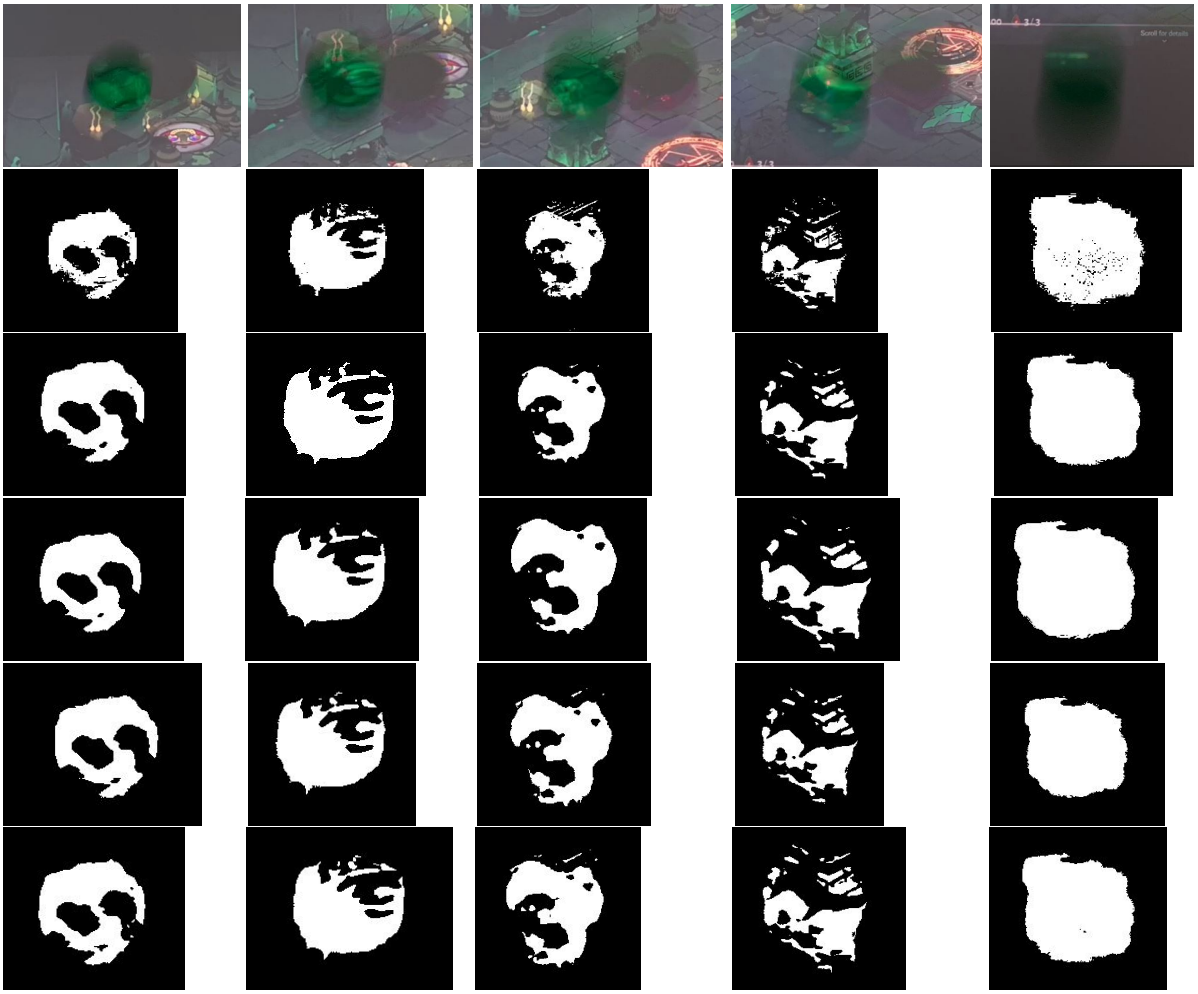
When a light source shines light on a smooth surface a specular highlight is seen on the surface. This is also the case for the balls used in the system because of their smooth plastic material. By blurring the image, the specular highlight gets averaged with the surrounding color turning an undetectable white spot into a detectable ball color.

During testing of all four blur algorithms proposed by OpenCV, averaging, gaussian blurring, median blurring and bilateral filtering, there was no significant improvement found for the tested settings. The test was done on the same hardware using the same test video recording from the same camera. To simulate the changing color of a game projected on the WIA in the background a gameplay video with lots of movement and colors, including various green values, was shown by the projector.[2] In the following figures 4.1 and 4.2 the impact of all blurring methods was compared. First, a HSV color filter with a lower V value was chosen to emphasize the effect on the blurring for different color values of the same hue. Then a higher V value was introduced to evaluate the importance of blurring and the color filter. The results are very obvious for this testing setup but might be different when other resolutions, background videos or further aspects are changed. These tests led to the

[1] OpenCV - Smoothing Images

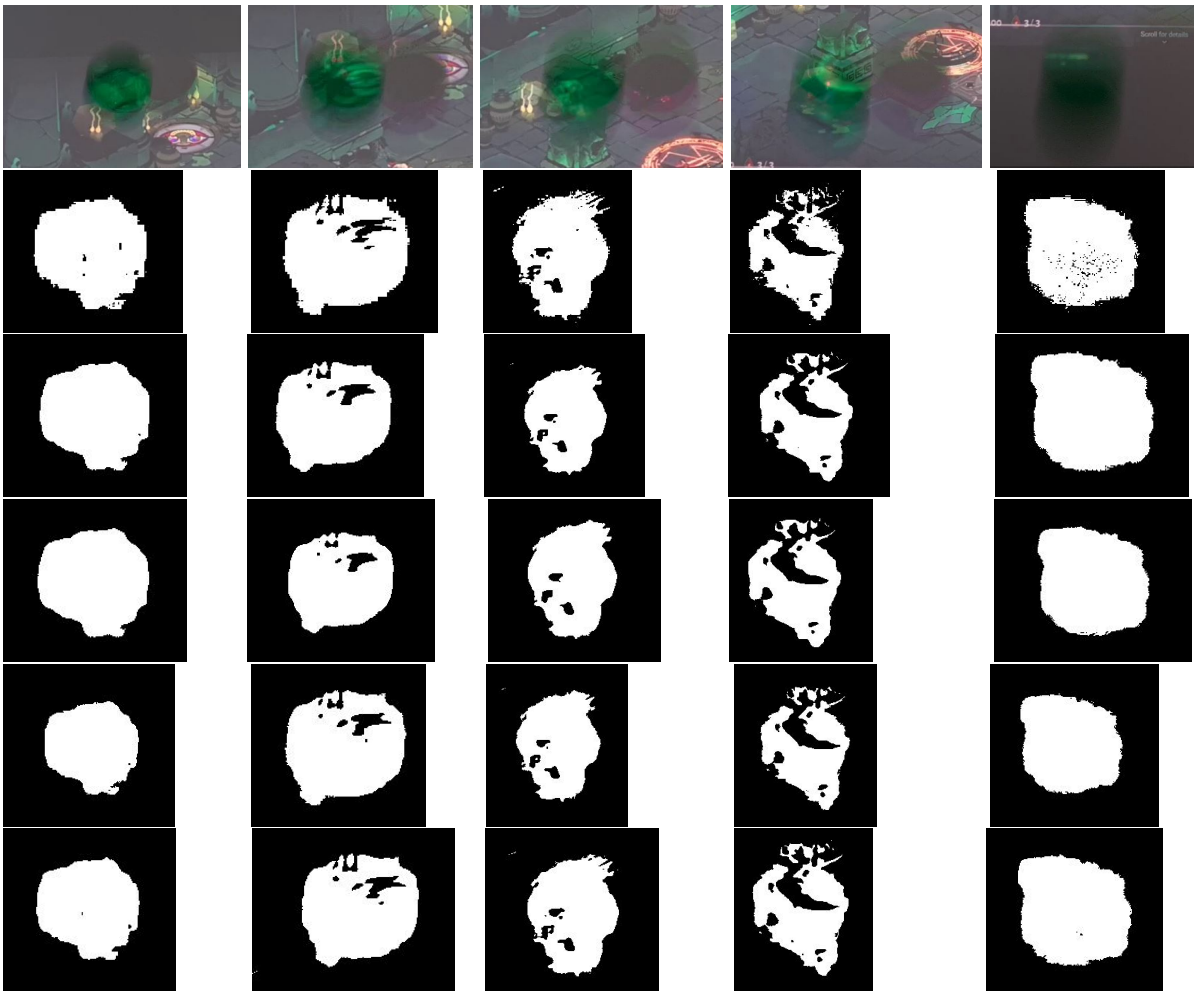
[2] Hades by Supergiant Games - https://www.supergiantgames.com/games/hades/ (10.08.2021)

result that blurring only has a small impact on the detection result of a green ball in a colored projector image. The choice for the values of the HSV color filter were far more important to successfully detect the green ball. Figure 4.1 shows very similar looking results for every method. The quality of each method was rated by how well it detected the whole area of the green ball and how few separated contours made up that area. The general use of a blurring algorithm seems useful to smooth out edges and reduce the number of separated singular pixels. This can be seen when comparing row 2 to all following rows. Figure 4.2 puts the increased V value into perspective and clearly shows that compared to 4.1 the correctly detected area is larger and the difference between the algorithms in 4.2 is again very minor. Therefore it is considered much more important to find fitting HSV values and a time efficient blur algorithm like gaussian blurring at the given values is sufficient.



All Rows: HSV color filter: lower-(35,80,30) upper-(90,255,**90**)
Row 1: Camera image of a green ball illuminated by the projectors colored game image
Row 2: No blurring
Row 3: Averaging; Code: cv.blur(camFrame,(5,5))
Row 4: Gaussian blurring; Code: cv.GaussianBlur(camFrame, (11, 11), 0)
Row 5: Median blurring; Code: cv.medianBlur(camFrame, 5)
Row 6: Bilateral filtering; Code: cv.bilateralFilter(camFrame,5,1500,1500)

Figure 4.1.: Comparison of all OpenCV blurring methods when using the same HSV color filter for thresholding. Method values were chosen to run at 30-40 Frames Per Second (FPS) for the testing setup



All Rows: HSV color filter: lower-(35,80,30) upper-(90,255,**115**); An upper V value of 115 was chosen for this filter to make the impact on mixing the same green "Hue" at the edge of different green "Values". Method values were chosen to run at 30-40 FPS for the testing setup
Row 1: Camera image of a green ball illuminated by the projectors colored game image
Row 2: No blurring
Row 3: Averaging; Code: cv.blur(camFrame,(5,5))
Row 4: Gaussian blurring; Code: cv.GaussianBlur(camFrame, (11, 11), 0)
Row 5: Median blurring; Code: cv.medianBlur(camFrame, 5)
Row 6: Bilateral filtering; Code: cv.bilateralFilter(camFrame,5,1500,1500)

Figure 4.2.: Comparison of all OpenCV blurring methods when using the same HSV color filter for thresholding.

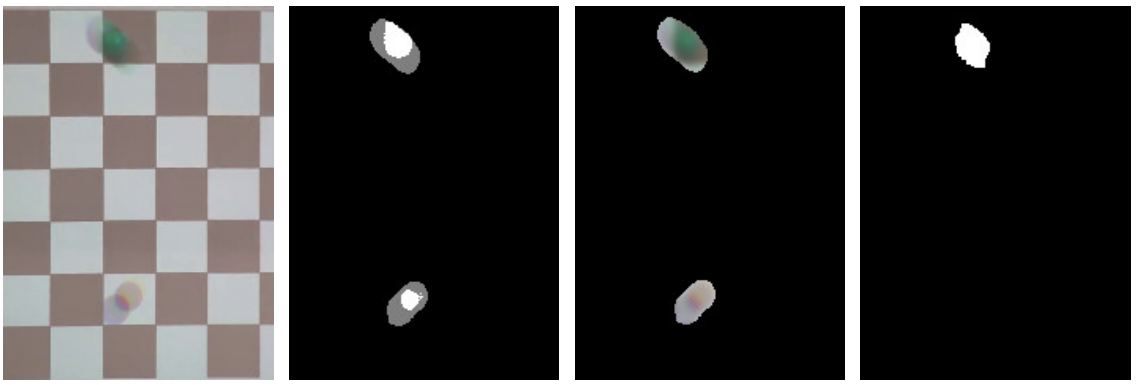
Eventually, the bilateral blurring resulted in no improvement, even though the theoretical functionality of the filter was expected to be very promising. Bilateral filtering is blurring images with respect to the edges between different colors. This would result in clear edges between for example large green and red colored areas but the various green or red colors inside these areas would be blurred together. This is exactly what was needed to detect the generally green colored ball area in the image in contrast to the differently colored background. As stated on the OpenCV documentation "If they (Function parameters for cv.bilateralFilter) are small (< 10), the filter will not have much effect, whereas if they are large (> 150), they will have a very strong effect, making the image look "cartoonish"." [5][3] The mentioned "cartoonish" look describes the desired effect well, but could not be achieved during the testing in this thesis even when using parameter values of 1500. Figure 4.3 shows a manually created comparison how the bilateral filtering was expected to work and enable an easy ball detection by color. A clear area separation by color with easily detectable colors was expected, such as the clear circular green area on the left representing the ball.

Figure 4.3.: Comparison an original camera image and the expected result of bilateral filtering created manually

[3]OpenCV - Image Filtering/bilateralFilter()

4.2. Background Subtraction

The major target regarding the camera image is to detect the thrown balls. Due to the nature of a throw, the ball will be constantly moving in the camera image. In general most objects that need to be detected for the system to work, are moving objects. That is why a filter for movement decreases the possible locations for balls in the camera image drastically and anything stationary can be filtered out and ignored. Since a fixed camera was used in the proposed system, the use of background subtraction algorithms seems useful here. These algorithms benefit from having a fixed camera.



(a) Camera image with moving ball and shadow

(b) KNN Background Subtraction Filter

(c) Camera image first blurred then masked out by background subtraction filter

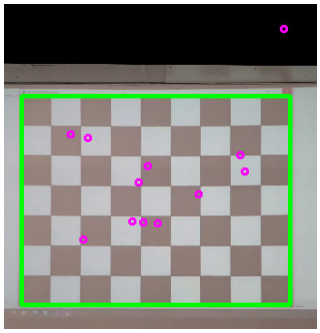
(d) HSV Color Filter for green ball

Figure 4.4.: Ball detection filter pipeline (Camera, KNN Filter, Color filter)

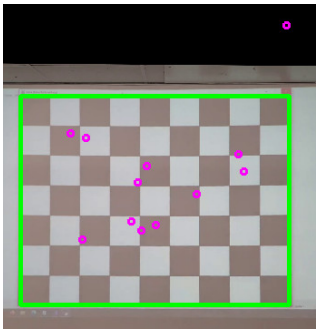
The goal of background subtraction is to differentiate between the background and also to differentiate between stationary and moving objects. This is achieved by initialising a background model which consist of a history of camera images. Then the latest camera image is compared with the background model and a threshold decides whether a pixel belongs to the background or foreground. Because the appearance of new stationary objects in our camera view is possible, especially when it comes to the projected image with moving game elements, the background model must be updated constantly. This is achieved by changing the camera images that make up the background model. It is preferable to use a high update value to sort stationary changes quickly into the background model. For example an appearing icon that does not move is seen by the camera. If the update rate is high, it will only take a short time until the icon is sorted into the background model. If the update rate is low, it will take longer for the icon to be sorted into the background model even if it does not move. It is better if stationary objects are sorted into the background as quick as possible since only moving objects are of interest. This concept is important to be considered for the user interaction design of the game.

Stationary objects like the room, the system it is located in, consisting of a wall, floor,

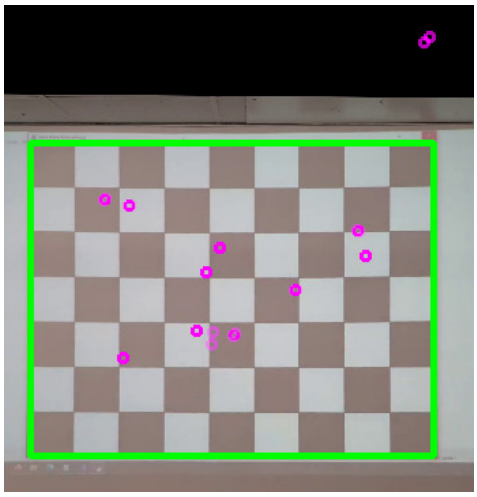
furniture etc., are initially sorted into the background and filtered out from the beginning. On the other hand, when moving objects travel through the camera image, they are sorted into the foreground and are detectable in isolation by the background subtraction filter. This is further illustrated in figure 4.4a which shows a cutout from the camera view with emphasis on a moving ball and its shadow in the projection image. Figure 4.4b shows how the background subtraction filters out everything except for movement.



(a) KNN



(b) MOG2



(c) Both results with 50 percent transparency

Figure 4.5.: Comparison between KNN or MOG2 background subtraction algorithm for impact detection

When using the OpenCV library for background subtraction algorithms, KNN and MOG2 are the two possible options to choose from. As the comparison in 4.5 shows, when using the same camera recording, both algorithms produce a very similar result. They also produce very similar wrong calculated impacts as seen in both algorithm related pictures in the top right corner. This led to the decision to choose KNN over MOG2 because KNN ran on average faster than MOG2 under same conditions. Both algorithms have the capability to differentiate between the actual moving object and its thrown shadows. This is working well for example

with detecting pedestrians on a road when the shadow is connected to the body like the example in the OpenCV Documentation demonstrates [5][4]. Because the ball is a flying object with a disconnected shadow projected onto the wall, the algorithm does not recognise the shadow as part of the ball and therefore splits the shadow and the ball into two separate moving objects. This can be observed in figure 4.4b where the white color depicts recognised moving objects and the gray color the shadow of the object. Because of this, the shadow separation feature of the background subtraction algorithms are not used. The movement filter is not sufficient enough to isolate only the green ball in the camera image, which is why a color filter is introduced in the next section.

4.3. Color Filtering

After the background subtraction filter recognised moving objects in the camera view, the result is used as a mask on the camera image to cut out only the moving parts. Therefore, the non-moving background areas that might have been recognised as ball-coloured are cut out since only moving areas can be thrown balls. This helps a lot to reduce false positives in the ball detection. To be able to set the filter variables more intuitively the image is transformed into HSV color space before the color filter is applied. When the color filter is applied to the cutout it filters for ball colored areas in the moving parts. The HSV color space is well-known in regards to computer vision and proves beneficial for cutting out object silhouettes [6]

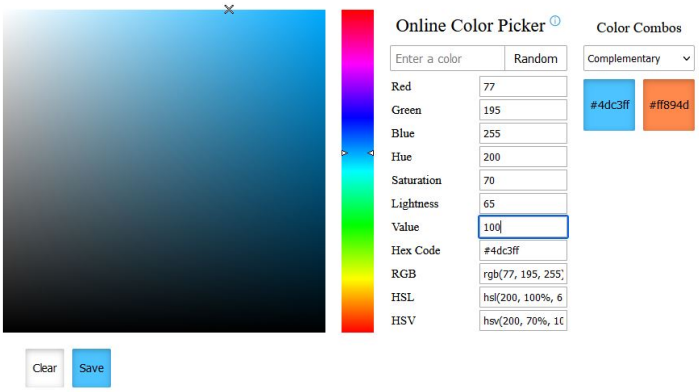
4.3.1. HSV Color Space

The camera image is first introduced by the camera to the algorithm in RGB Color space where RGB stands for the red, green and blue color channel. In OpenCV the RGB color space is actually BGR space meaning blue, green, red. But since this is just a different order of the color channels it can be ignored for further explanations. [5][5] The RGB channels are convenient for displaying an image on a computer screen, since the screen consists of many small pixels that have tiny red, green and blue Light Emitting Diodes (LEDs) to display the according RGB value of a pixel in the image. But this color space is unpractical for the use with computer vision and especially in the research application of this thesis.

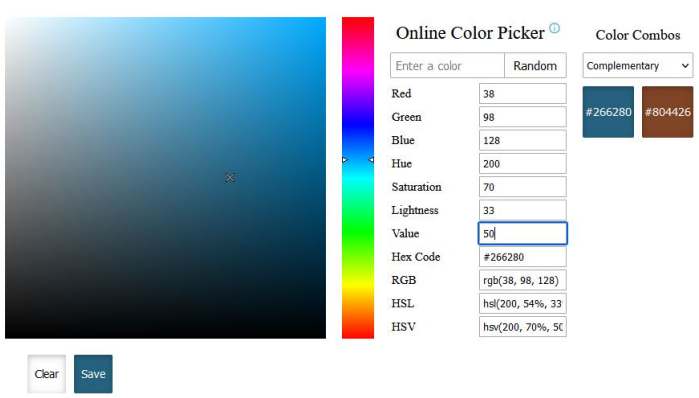
RGB is made up by three values that range in [0-255] intervals with one value for each color channel. These values describe how much of each color is contributing to the whole color shown by the LED's light. For example the color (255,0,0) would be displayed on a computer screen by only the red LED shining at full intensity giving a bright red color. The same principle holds for green with (0,255,0) and blue with (0,0,255). (127,0,0) would be the same color of red, but at a lower intensity, appearing as a dimmer red. (0,0,0) would be the color for black, where black is actually just the appearance of the monitor screen with no LEDs shining. So one could say that (0,0,0) does not represent black but rather the monitor color when it is turned off. White would be (255,255,255) where all three LEDs are shining at

[4]OpenCV - Background Subtraction
[5]OpenCV - Changing Color Spaces

full intensity. Together they create the appearance of a white color. This is where problems arise when trying to describe color how it is intuitively perceived by humans whilst using the RGB color space.



(a) Color Picker showing the color value for a perceptually light blue color. RGB:(77,195,255); HSV:(200,70,100)



(b) Color Picker showing the color value from (a) but with half the value. RGB:(28,98,128); HSV:(200,70,50)

Figure 4.6.: Color value comparison between the RGB and HSV color space [6]

Two example colors are shown in figure 4.6 where (a) shows a lighter blue color and (b) a darker blue color. To go from the color in (a) to the color in (b) using the RGB color space, one would have to decrease R by 39, decrease G by 97 and decrease B by 127.

Intuitively speaking, it is very hard for the average person to come up with these value changes on their own. But when looking at the values Hue, Saturation and Value, the same change from (a) to (b) can be achieved by decreasing only the Hue value by 50 and therefore intuitively halving its value. That is why the use of the more intuitive HSV color space is more practical when color needs to be adjusted manually which will be discussed again in section 4.3.2. [7] Therefore the RGB color space gets transformed into HSV color space using

[6] `https://www.qvcool.com/` `https://colorpicker.me/` (2021-08-03)

OpenCV library functions.

HSV stands for "Hue", "Saturation" and "Value". The goal of this color space is to represent color in a more intuitive way, similar to mixing paint. The concept of HSV is shown in figure 4.7. "Hue" defines the primary colors that make up the desired color. The hue value in the HSV color space ranges in an interval of [0-360] which describes the circular shape in 360 degrees of the cylinder shown in 4.7. In OpenCV this value is halved and ranges in an interval of [0-180] for reasons regarding how large numbers are stored in computers. [5][7]

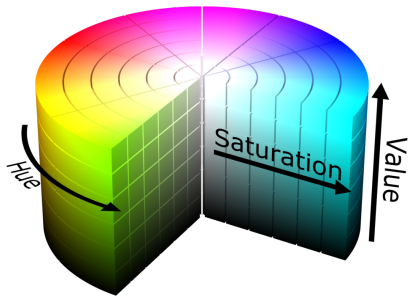


Figure 4.7.: Illustration of the HSV color space [8]

"Saturation" represents how much white is mixed into the color showing its closeness to white color or how washed out the color appears and ranges in an interval of [0-100]. A value of 0 has a lot of white mixed in the color intuitively meaning no saturation, a value of 100 has no white mixed in the color leaving the pure color from "Hue" at full saturation. All HSV colors in range of ([0-360],0,100) appear as white.

"Value" represents the darkness or lightness of a color and ranges also in an interval of [0-100]. The number represents how much black is mixed into the color showing its closeness to black color or dark the color appears. All HSV colors in range ([0-360],[0-100],0) appear as black. [8]

Especially when trying to describe the color appearance of a ball in different lighting situations, HSV has advantages over RGB. Considering the scenario of a green ball in a completely dark room, the ball would be seen as black because there is no light shining on the ball, but the actual color of the ball is still green. This information will be lost when using RGB. Since (0,0,0) depicts black in RGB there is no way of telling what color the ball is supposed to have under better lighting conditions from only looking at the RGB value of (0,0,0). In HSV values the green ball in a dark room can be described as (120,100,0) which intuitively describes the Hue of green, no white light and complete darkness. Both RGB and HSV would show the same black color, but in HSV you can imply that the ball is supposed to be green because of the H value. The S and V values would describe the lighting situation in this case. This explains why HSV is intuitively superior to RGB when trying to describe ball

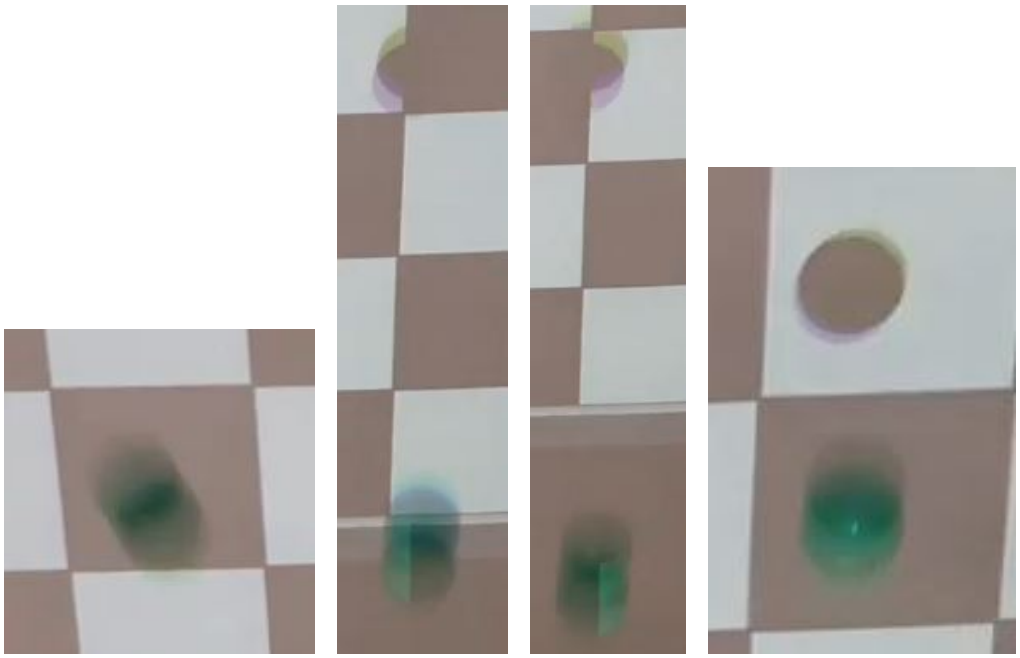
[7] OpenCV - Changing Color Spaces

[8] SharkD, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons, HSV Color Cylinder `https://upload.wikimedia.org/wikipedia/commons/4/4e/HSV_color_solid_cylinder.png` (03.08.2021)

colors in different lighting. [9]

4.3.2. Lighting and Value Calibration

For the algorithm and its filters to work properly, calibration values need to be adjusted during the setup of the system. Lighting is a huge factor when it comes to determine what colors the camera picks up. An extreme situation would be a completely dark room with no lighting and a black camera image or a very brightly lit room with overexposed colors that create a white camera image. Because the system is supposed to be set up inside buildings of location based entertainment centers, a rather constant indoor lighting with little variation from sunlight is expected. This is also emphasised by the projector lamp being the major light source next to the room lighting. When the thrown balls travel from outside the projection volume to the inside of it, a very significant change in lighting can be observed depending on the projected image. During the testing of the system the checkerboard from figure 3.2 was used as a projection image which provided very high lighting contrasts from the black and white squares.

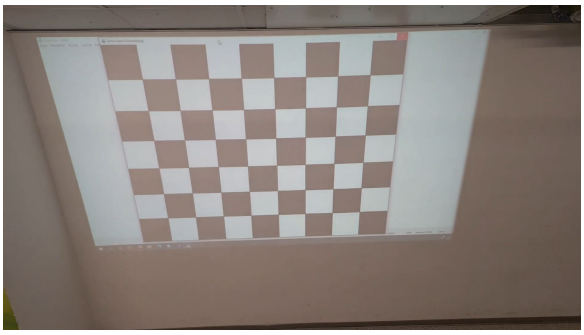


(a) Ball unlit on black square (b) Ball half lit on left (c) Ball half lit on right (d) Ball fully lit on white square

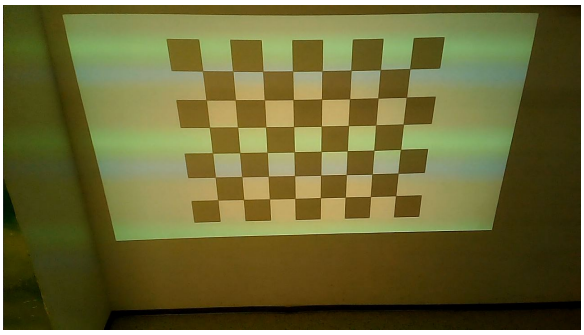
Figure 4.8.: Comparison of different lighting situations created by the projector image

Figure 4.8 illustrates the differently lit balls and the according changes of color lightness. This is where the previously mentioned advantages of the HSV color space apply. Setting up filter values for the color filter needs to be done for every system setup individually because lighting conditions may vary a lot between systems. Therefore, it is not possible to

generate filter values that work for every situation. Another aspect that affects the color of the camera image is the camera itself. The system can be used with any camera that fulfils the requirements for FPS, resolution and exposure time. Different cameras may produce a different colored image. For example, the Samsung Galaxy S20 smartphone camera and the Orbbec Astra Pro camera which were both used during development, create a very different image under the same conditions shown in figure 4.9. Besides the obvious differences in color, the images show the importance of exposure time in the system. This is further described in 4.3.3.



(a) Camera image of a Samsung Galaxy S20 Smartphone at full high definition resolution, 30 FPS.



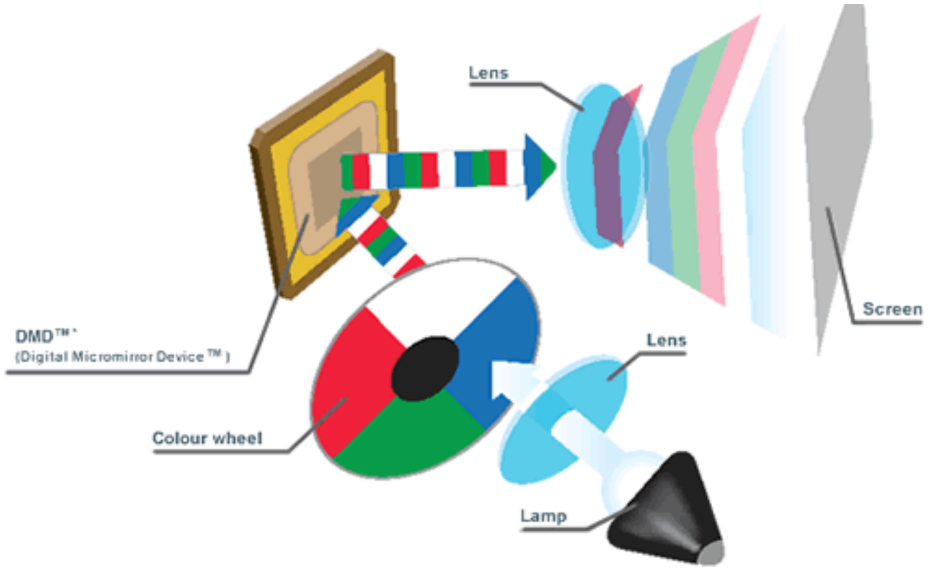
(b) Camera image of a Orbbec Astra Pro at high definition resolution and 30 FPS. The Rainbow Effect 4.3.3 can be observed.

Figure 4.9.: Comparison of different camera images under same conditions

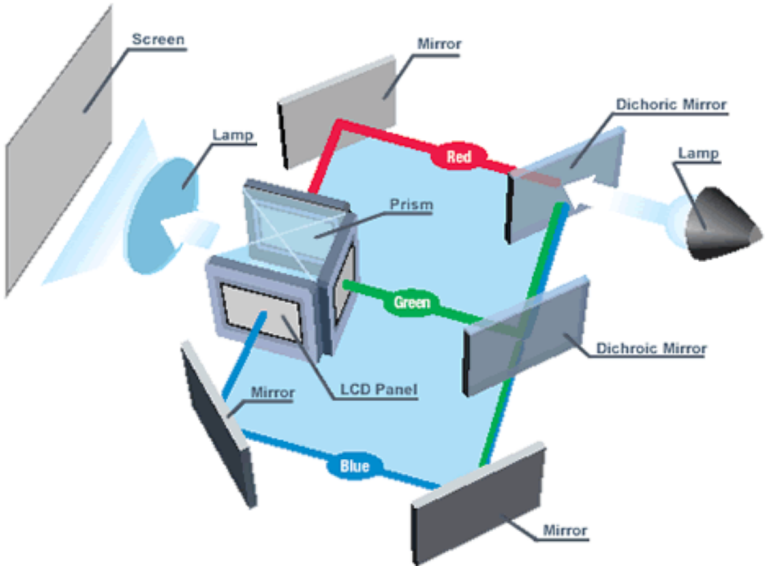
4.3.3. Rainbow Effect

The Rainbow Effect is the effect that can be observed in figure 4.9 where (a) has no Rainbow Effect and (b) has the Rainbow Effect shown in rainbow-like colored lines that are horizontally oriented in the camera image that travel from top to bottom. This effect occurs when a combination of a 1-chip DLP projector and a camera with an exposure time that is not synchronized with the projectors vertical hz value is used. The Samsung Galaxy S20 camera was used to the the image of 4.9 (a) and shows no rainbow effect because the exposure time was locked to a synchronized value for the projector. Image (b) was taken by the Orbbec Astra Pro with an automatic exposure time adjustment. This resulted in automated brightness adjustments for the lighting environment but also to changing exposure times that introduced the rainbow effect to the image. A 1-chip DLP projector is a certain type of projector technology shown in Figure 4.10 (a) where DLP stands for digital light processing.

This technology uses a lamp that emits white light. The white light then passes through a color wheel. This color wheel spins at high velocities and is, in its simplest form, made of three primary color filters of equal size. One third for red, green and blue. Each letting only the red, green or blue light of the white light pass through respectively. The fast spinning in combination with the color filters creates a stream of sequentially red, green and blue light. There are other color wheel designs with more and differently sized color filters but



(a) 1-chip system with sequential color



(b) 3-chip system with simultaneous color

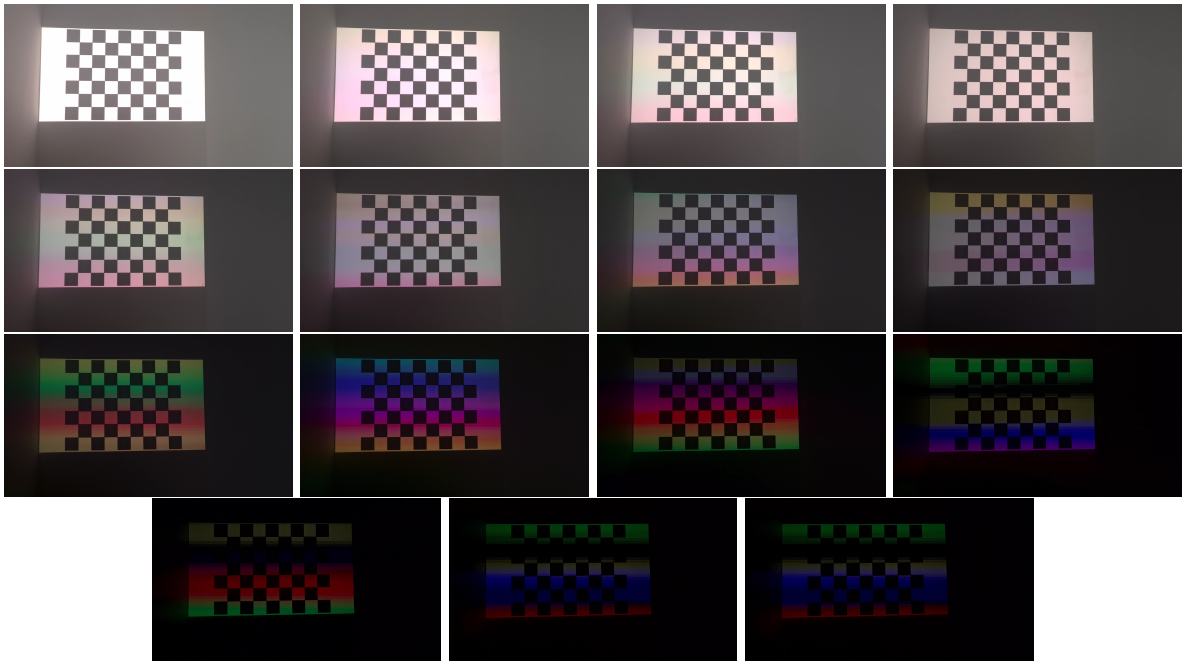
Figure 4.10.: Comparison between 1-chip DLP and 3-chip DLP projectors[9]

the concept stays the same. This stream of light hits onto a chip that gives the 1-chip DLP technology its name. This chip is made of very small mirrors that represent one pixel of the projected image for each mirror on the chip. These mirrors can either direct the light stream through the projector lens to create the projected image or direct it onto a black surface for absorption. This way a completely red image would be created by every mirror directing the

[9]Epson - `https://www.epson.com.au/news/technologynews/3lcd.html`

red light through the lens, when it is the correct time for the red light of the light stream to hit the chip and direct the green and blue light on the black surface for absorption. This process happens very fast and is dependant on the speed of the color wheel which is dependant of the hardware of the projector. Each projector has a so called vertical refresh rate. This describes how often in a second the image in its final color is presented per second. During the development phase of this thesis a projector with a vertical refresh rate of 60hz was used. So this projector would show a final image every $\frac{1}{60}$ second.

At this point, the exposure time of the camera used in the system becomes important. The camera can only record a correct image while using a 1-chip DLP when the exposure time and the vertical refresh rate of the projector are on the same frequency. In this case, the exposure time needs to be $\frac{1}{60}$ seconds while using a 60hz projector. When looking at the results of figure 4.11 then only the first image with $\frac{1}{30}$ seconds and the fourth image with $\frac{1}{60}$ seconds exposure time have no sign of a Rainbow Effect.



The pictures' exposure times while recording a 60hz projector with a Samsung Galaxy S20 are as follows from top right to bottom left: $\frac{1}{30}, \frac{1}{45}, \frac{1}{50}, \frac{1}{60}, \frac{1}{90}, \frac{1}{125}, \frac{1}{180}, \frac{1}{250}, \frac{1}{350}, \frac{1}{500}, \frac{1}{750}, \frac{1}{1000}, \frac{1}{1500}, \frac{1}{2000}$

Figure 4.11.: Comparison of different exposure times recording the Rainbow Effect

Though the fourth setting was chosen for testing because an exposure time of $\frac{1}{30}$ seconds is illuminated twice by two finished $\frac{1}{60}$ second interval images produced by the projector and therefore is getting too bright. Also, a shorter exposure time helps to reduce the motion blur effect that occurs when moving objects, such as a thrown ball, are exposed for a longer time. That leads to the balls being recorded as semi-transparent streaks instead of circular shaped balls. There are algorithms that make use of the streak recordings, but in this thesis a different approach was chosen that benefits from circular shapes. [10] Hence, an exposure

time of $\frac{1}{60}$ second gives a better image than $\frac{1}{30}$ seconds.

It is also important to note that the FPS of a camera define a minimum for the exposure time. Meaning that a 30 FPS camera can only record at an exposure time of $\frac{1}{30}$ seconds, a 60 FPS camera at $\frac{1}{60}$ seconds, a 120 FPS camera at $\frac{1}{120}$ seconds and so on. This thesis made use of a 30 FPS camera, since this is a common rate of FPS for cameras to operate at and therefore giving a lower FPS limit for a proof of concept for the system. More FPS come at a higher price point but would also increase the sampling rate of the thrown balls resulting in potentially better tracking. Regarding the rainbow effect a 120 FPS camera would not be able to record at an exposure time of $\frac{1}{60}$ seconds and the rainbow effect would occur. To fix this a 120hz projector would be necessary resulting again in additional costs.

The rainbow effect is solely bound to the use of 1-chip projectors. The use of 3-chip projectors that enable the red, green and blue light to be emitted not sequentially but in parallel, would remove the rainbow effect problematics. But these kind of projectors come at a higher price point as well, probably making the whole system too uneconomical. [11]

Further findings that were made during testing consisted of the orientation of the rainbow effect's colored lines. The orientation of the camera recording the rainbow effect determined the orientation of the rainbow lines. This is estimated to be caused by the technical aspects of a frame being built up by a monitor from top to bottom but was not further investigated during this thesis. Figure A.1 in the appendix shows an example depicting the observed behaviour.

Another but minor problem was the flicker effect that occurs when the exposure time of the camera is not synchronized with the voltage frequency of the lamps lighting the scene. The flicker was visible in the camera image as oscillating brightness levels over several frames. The difference in brightness was insignificant and also introduced no color differences. To prevent the light flickering an exposure time of $\frac{1}{50}$ seconds is necessary to synchronize with the 50hz voltage frequency. Since the light flickering only caused minor problems that could easily be overcome by the background subtraction, an exposure time of $\frac{1}{60}$ seconds was still chosen. Also, the rainbow effect and motion blur are much more important to prevent. [12]

4.4. Choice of Ball Type

For the balls to be used in this system two kind of balls have been analysed. Figure 4.12 shows both kind of considered ball types. The first feature that was considered was the throw behaviour of the ball. For the camera to be able to detect circular looking ball shapes at 30 FPS and $\frac{1}{60}$ seconds exposure time the ball has to have a relatively slow throw speed. The system could be generally designed to be used with any kind of ball but in this particular use case for location based entertainment centers soccer balls were excluded because kicking the ball was not intended due to the high forces of a kicked ball that potentially introduces more risk when kicked at a wall redirecting it, higher travel speeds and an expected lower precision than a thrown ball would have.

When considering a ball to be thrown, a size comparable to a baseball was expected to be a reasonable choice. Especially when these similar sized ball pit and sponge balls are already in use at location based entertainment centers and young children or teenagers are already familiar with them. Therefore the choice was cut down to only ball pit or sponge balls.

These balls are relatively light when compared to a baseball. Ball pit balls weigh about 10g and sponge balls about 15g. The diameter of the ball pit ball is a little bigger than the sponge ball resulting in a higher air resistance for the ball pit balls. Upon testing it was discovered that the ball pit balls absorb more energy upon impact than the sponge balls leading to a lower velocity after the impact. This further improves the detection.

(a) Ball Pit Balls (b) Sponge Balls

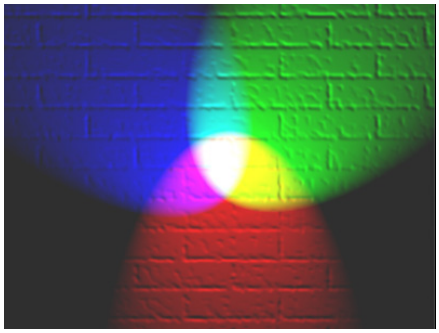
Figure 4.12.: Comparison between considered ball types.

The biggest advantage in using the ball pit balls is that they can be colored. More so, that they can have very distinctive colors that have little to no change throughout the whole surface of the ball. Due to the plastic material, the balls do not change their shape, size and surface. This is not the case with sponge balls. As shown in figure 4.12 the ball pit balls in (a) look very similar to their equally colored neighbours. In (b) all nine balls were initially of the same kind, but only the top row looks rather similar. The middle row shows greater discolorations that make it hard to define a color filter for this kind of balls. The bottom row shows the damage that sponge material can tolerate during its lifetime in a location based entertainment center. This makes it difficult to show up as a circular shape in the camera and

also creates a bad playing experience since it will not roll and bounce like a standard ball does. Furthermore, when a multiplayer is considered to work by using differently colored balls for each player ball pit balls have a higher variety and clearer distinction of colors. All these factors led to choosing the ball pit balls for the system.

4.5. Choice of Ball Color

There is a wide range of colors for ball pit balls, though the colors seen in 4.12 (a) can be considered as common for this type of balls. But when it comes to the most effective use in the system, blue, green, yellow and red are not equally useful. In light theory, red, green and blue are the key colors when considering additive primary colors. A mix of these three light colors make up all colors of light the human can perceive until they create white light where each primary color is equally mixed together. This is illustrated in figure 4.13.



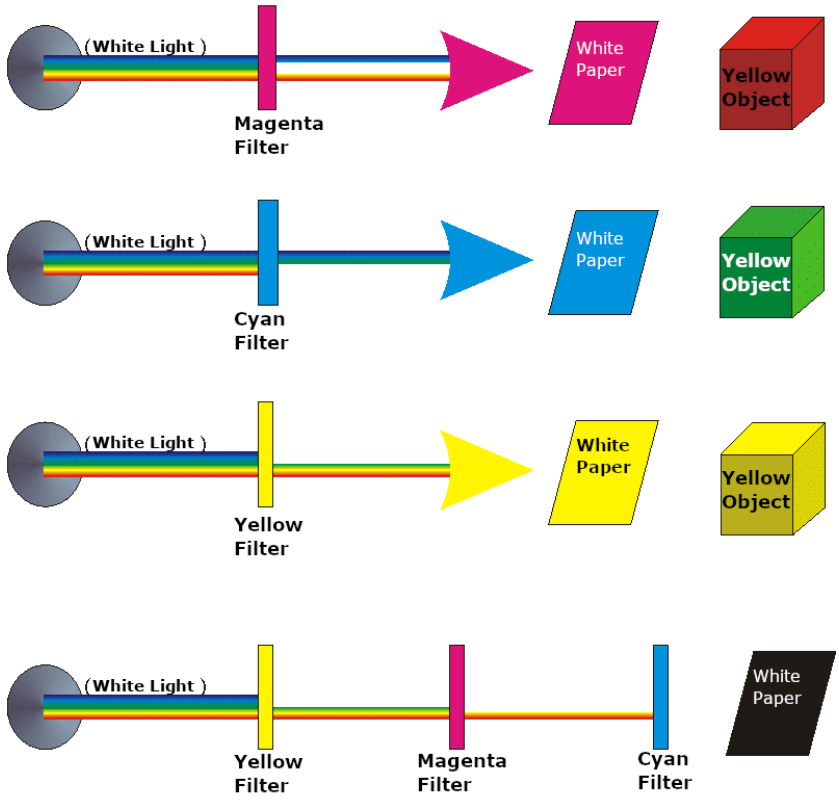
Red, green, and blue lights combining by reflecting from a white wall: adding red to green yields yellow; adding all three primary colors together yields white.

Figure 4.13.: Various Interactions of Colored Light[10]

The setting shown is very comparable with a white wall used to project the WIA onto and a projector emitting multi-colored light. It can also be observed that a mixture of red and green light creates the appearance of yellow light. This is essential for the choice of ball colors. The white wall in figure 4.13 is seen in different colors because white reflects all colors, absorbs no light accordingly and redirects the light wave into the camera. The black parts of the white wall are not illuminated by any light and therefore appear to be black. The red, green and blue parts are illuminated by the respective light and are both all reflected appearing as an accordingly colored wall. It should be mentioned that this explanation is general to grasp the idea behind the color choice and will not state accurate values, such as the fact that a wall never reflects 100% of light waves and ambient light is neglected.

[10] en:User:Bb3cxv, CC BY-SA 3.0 <http://creativecommons.org/licenses/by-sa/3.0/>, via Wikimedia Commons RGB Illumination, `https://upload.wikimedia.org/wikipedia/commons/2/28/RGB_illumination.jpg` (09.08.2012)

In case the wall is not white but painted in yellow, the the wall would reflect light very differently. The fact that a yellow painted wall appears as yellow in white light or daylight conditions is based on yellow colored surfaces only reflecting light waves that make up yellow and everything else is absorbed. As mentioned before, yellow light can be created by shining a red and a green light. So a yellow wall will reflect all wave light wave lengths from between red through yellow to green. On the one hand, this means that a yellow wall will appear as yellow when illuminated by a yellow light source or a red and green light source. But on the other hand, this also means that a yellow wall will appear as green when illuminated by a green light source or red when illuminated by a red light source. This is where the problem lies in the use of yellow balls or balls generally colored in non-primary colours.



Comparison of differently colored light interacting with white paper and a yellow object

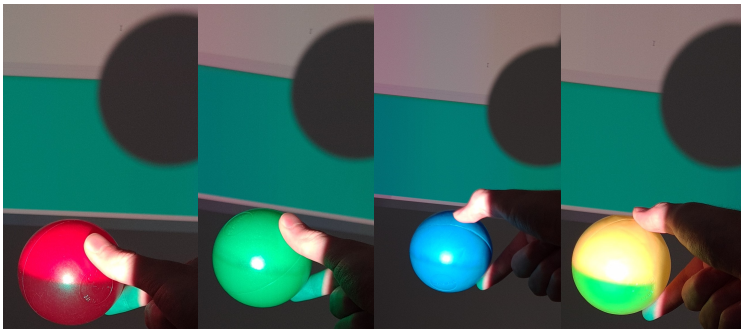
Figure 4.14.: Interaction of light with yellow objects [11]

A green ball will only be seen in different values of green depending on the lighting because green only reflects green light. The same principle holds for red and blue colored balls. They might have a different saturation and value but the hue will always be the according ball color. This is not the case with yellow balls. When looking at the example given in figure

[11] Herbert Bernstädt - `https://wiki.production-partner.de/licht/subtraktive-farbmischung/` (09.08.2021) (translated)

4.14 several different outcomes are illustrated when various light colors interact with a yellow object. When magenta light shines on a yellow object, the object will appear as red because yellow reflects the red part of magenta and absorbs the blue part. Analog to that, when cyan light shines on a yellow object, the object will appear as green because yellow reflects the green part of cyan and again absorbs the blue part. When yellow light shines on a yellow object, it will appear as yellow, because it reflects all green, yellow and red parts of yellow light. When all wave lengths of the white light are filtered out, everything will appear as black because no light will make it to the objects surface.

Figure 4.15 shows the testing result of the concept explained in figure 4.14 using a test environment with a projector. Here it is very important do understand that neither of the red, green and blue balls shown introduce problems because (a) only has a darker value of the same red light and (b) and (c) are still fully and correctly colored. Only the yellow ball in (d) is more problematic because it can be perceived as a green ball. Therefore the color of used balls is limited to the primary colors red, green and blue.



(a) (b) (c) (d)

a: Red ball, clearly visible difference between top and bottom half, top half reflects red part of white light shining bright red, bottom half reflecting no part of cyan light appearing unlit
b: Green ball, almost no visible difference between top and bottom half, top half reflects green part of white light shining bright green, bottom half reflects green part of cyan light shining bright green.
c: Blue ball, analog to b but with blue light.
d: Yellow ball, clearly visible difference between top and bottom half, top half reflects yellow part of white light shining bright yellow, bottom half reflecting green part of cyan light appearing green, index finger reflecting cyan light for difference comparison to green.

Figure 4.15.: Comparison of red, green, blue and yellow ball pit balls under white and cyan (blue and green) light

4.6. Resolution Optimization

The system has no upper limit regarding the resolution when different cameras with different resolutions are considered. Depending on the used camera, the image will have higher detail and the system most likely will cost more money. But a higher resolution is not always a better option. The resolution of a camera image has a great impact in the running time of the algorithm since a higher resolution results in more pixels that need to be analysed. Also, as mentioned in section 4.1, a higher detail in the image does not necessarily lead to better tracking. That is why there needs to be an optimal balance between resolution and running time of the algorithm: A resolution that is high enough to not lose precision in tracking and a running time fast enough to be used in real time. This needs to be tested with the hardware used in each system individually, since single or multiple components can be exchanged running at different speeds. The figure 4.16 shows latest result for the hardware setup used during development.

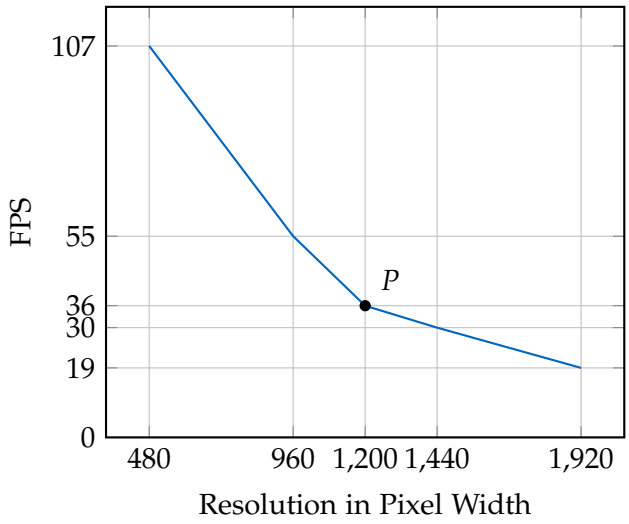
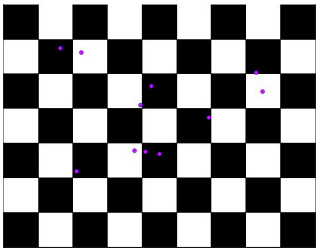


Figure 4.16.: Comparison of algorithm FPS in relation to camera image resolution

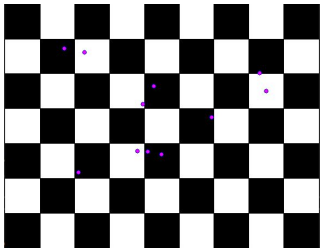
The x-axis represents the resolution of the camera image in pixel width. This means the x-axis has different scalar values from the full high definition resolution of 1920 times 1080 pixels. The y-axis represents the FPS the algorithm had for analysing a prerecorded video. The video was recorded in 30 FPS so when the algorithm is faster than 30 FPS it processes the images faster than a 30 FPS camera can deliver it to the algorithm. Thus resulting in a speed useful for real time. When the algorithm is slower than 30 FPS it processes the video frames slower than the camera can deliver them, resulting in a time delay that creates an unsatisfactory play experience. The plotted blue line represents the FPS speed results for each resolution tested. This was achieved by downscaling the pre-recorded video with OpenCV. The lower the resolution the lesser pixels to analyze and the faster the algorithm - but also, the lower the precision of the tracking. Point P represents a value for a possible optimum with a high resolution for precise tracking but still constantly above 30 FPS to ensure real

time tracking. Figure 4.17 shows the calculated impact ball points of a pre-recorded test video for different resolutions.

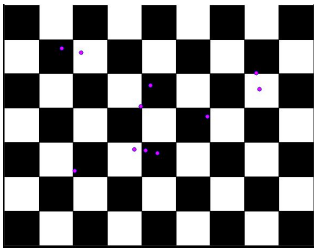
Each individual result, (a)-(e), as well as an overview of all results together (f) are given. The green points represent the highest resolution of the original video with 1920 pixel width and gradually shift their color towards red with decreasing resolution. The yellow points at 1200 pixel width represent the optimum value P from figure 4.16. It can be observed, that the lowest resolution (e) with red points is missing a detection point in the square going six squares to the right and three down starting from the top left corner. This missing point is tracked in (d) so greater tracking errors are expected for a resolution lower than 960 pixel width. The yellow points are relatively very close to the green points showing high precision at FPS for real time usability. Therefore, a resolution adjustment for the tested setup to a pixel width of 1200x675 is advised.



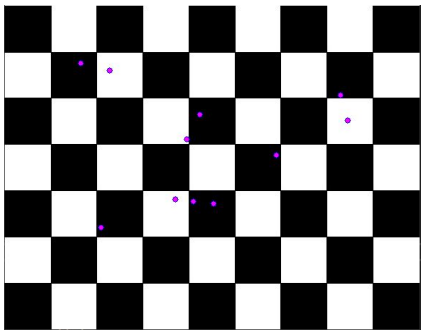
(a) Detection result 1920x1080



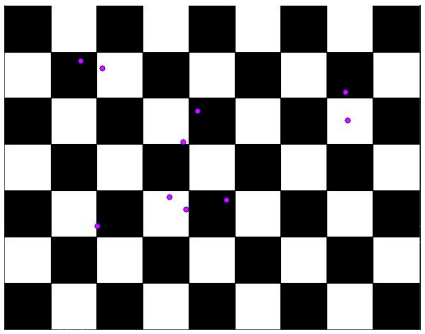
(b) Detection result 1440x810



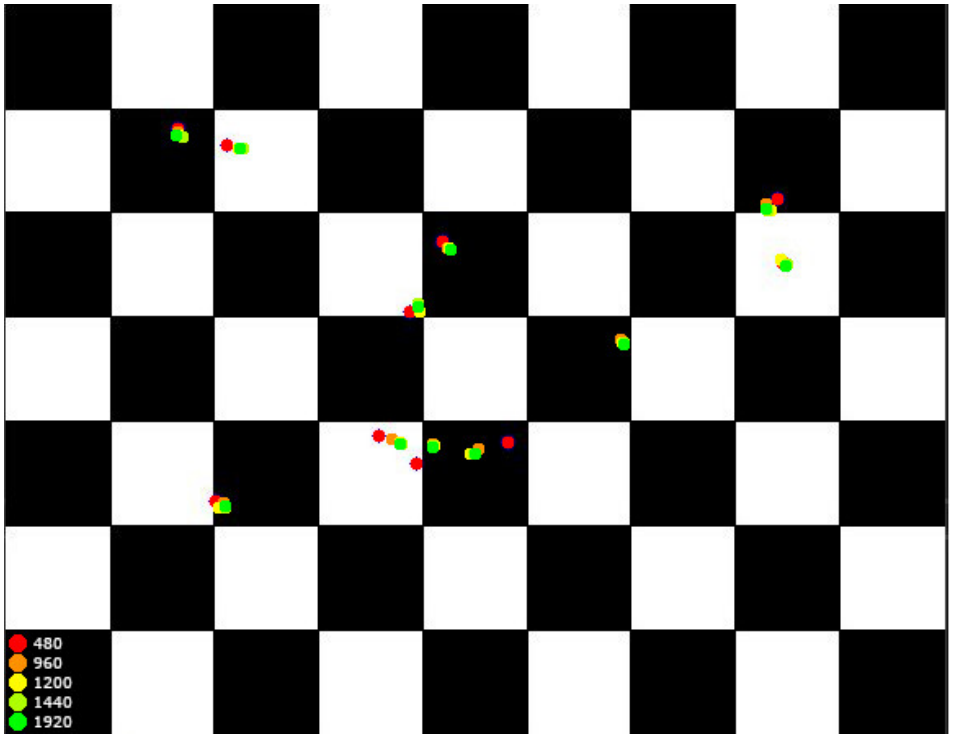
(c) Detection result 1200x675



(d) Detection result 960x540



(e) Detection result 480x270



(f) Comparison of all results together

Figure 4.17.: Comparison of the tracking precision for different resolutions

4.7. Contour Analysis

At this point the algorithm has reached a state where possible balls are marked by white areas in the camera image, called contours. To further increase the quality of these contours a common method of erosion and dilation is applied to them.

4.7.1. Erosion and Dilation

Erosion in this case is comparable to how sand or stones normally erode. Shapes get smoother because the area of white contours is decreased from where the edge to black is. Squares will get rounded off, thick lines will become thinner and smaller spots eventually disappear by becoming black. This method is useful to get rid of noise that was detected as very small areas compared to larger, correctly detected areas. Also correctly detected streaks of motion blurred balls are shaped more into a circular shape.

Because erosion decreases the overall white area of contours the dilation method is applied. It is essentially the opposite of erosion and increases the area of white at the edges to black. But because smaller white areas get erased during the erosion method they will stay erased because there is no white left to dilate. Thus noise is being removed. Dilation also smoothes shapes and supports a circular contour result. [5][12]

4.7.2. Contour Detection

Once the contours are in their final shape they get detected by the OpenCV find contours method. This process detects each individual white area. When white areas are detected, meaning that a ball is present in the camera image, contours are measured by their area size. A size filter is then applied that filters out all the contours that are too small or too big. Because the camera setup in the system and the ball size are fixed, a minimum contour size for a ball can be estimated by measuring the size of the contour of the ball when it is hitting the WIA at the greatest distance to the camera. This usually would be one of the bottom corners of the WIA. A maximum can also be estimated by averaging the ball contour sizes when the ball enters the camera image. The minimum is clearer defined than the maximum because a ball right in front of the camera lens would fill the whole image as one very large contour and still be a correctly detected ball, but this is not expected to happen during play. The goal is to define a maximum large enough to detect balls thrown at a closer starting point to the camera but small enough to filter out things like human body parts or larger moving objects in the camera view.

Out of this filter the largest contour is expected to be the best estimate for a ball because it has the clearest response to all filters due to its size. [5][13]

[12]OpenCV - Eroding and Dilating
[13]OpenCV - Contours

4.7.3. Obstacle Detection

Because contours that are filtered out as too large to be a ball are still detected contours, they can be used to determine if something unexpected is obstructing the play area. If for example a player is moving too close to the WIA and appearing in the camera image as a large moving contour, the information can be used accordingly. The player could be grabbing a ball that lies close to the wall, or could try to get closer to the wall to make targeting easier and eventually cheat since throwing from a distance is part of the gameplay or another person that is not playing could walk in front of the wall. All these scenarios can be handled by displaying a warning or a hint to step back from the wall to continue play. These obstructions may happen but could hinder the correct detection of the ball, so it is good game design to informs the player on how to move to ensure most optimal detection conditions.

4.7.4. Single Ball and Multiple Ball Detection

At this point an algorithm for the tracking of multiple same colored balls would come into place if multiple balls per player are used. If only one ball per player is allowed, it is due to the color filter that the balls can be easily assigned to each player. The contour detection process is executed for each color separately so that the ball position for each player can be determined. When one player can throw two or more balls at the same time, using the biggest filtered contour is no longer sufficient because multiple similar sized contours can be separate balls. The implementation of multiple ball detection exceeds the scope of this thesis research and is possible in future work.

4.8. Position Analysis

When the best estimation contour of the ball is determined, the ball's position is located by calculating the center of a minimal enclosing circle around the contour. OpenCV also provides methods to do that. These positions get collected for each ball color as long as the individual colored ball is visible inside the camera image. The positions then represent the basis for the tracking of the ball. To execute the final task, the calculation of the impact position of the ball, the last four positions of the ball, which are recorded with each camera frame, are analyzed. Two vectors are defined by the four latest positions. The last and second last define the first vector and the third last and fourth last define a vector. In each frame, the angle between both vectors is calculated. If this angle is close to 0 degrees the vectors are expected to be almost parallel. This means the ball is flying in its flight trajectory without any interference. When the ball hits the WIA, it is redirected by the wall and thus the flight trajectory is abruptly changing its direction. This change in direction can be detected by the angle between the two vectors. The size of the angle depends on the position of the camera in the system but also on the way the ball is thrown. Balls thrown orthogonally onto the wall will have a huge change in direction and the angle between the vectors will be relatively large, close to 180 degrees. When a ball is thrown almost parallel onto a wall, there will be little change in direction and the angle between the vectors will stay relatively small and close to parallel.

When a flight trajectory with significantly enough change in direction occurs, the two vectors that define the last two frames before and after the balls impact are then intersected. The intersection point is then taken as the ball's impact point. Because the impact of a ball is only of importance when it happened inside the WIA, the estimated impacts outside of the WIA can be ignored. This further reduces falsely detected impacts. After the ball's impact position is calculated, it still has an error depending on the viewing angle of the camera. This error is referred to as viewing angle error and is explained in the following.

4.8.1. Viewing Angle Error

The viewing angle error marks the difference between the impact point of the ball determined by the camera image and the actual impact point of the ball. Following camera calibration, the camera image itself should be in a linear space without any space morphing by the camera lens. This enables to read out the ball position by looking at the linear pixel coordinates. Due to the nature of a camera image, the correct position of the ball would only be possible the be read out in pixel coordinates when the camera faces the wall at a 90 degree angle and the ball would be in the center of the camera image. Since the camera will never be at a 90 degree angle to the wall because it would obstruct the player's view, the throw trajectory and the ball will have impacts all over the camera's image, the viewing angle error has to be considered. Figure 4.18 shows an illustration for the calculation of the viewing angle error.

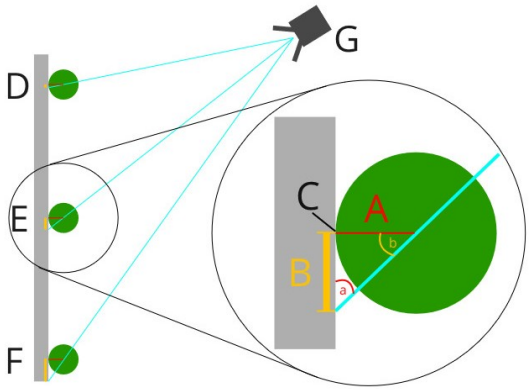


Illustration of the Viewing Angle Error. Different ball impact positions in relation to the camera are shown on the left. A close-up view of the central impact point is shown on the right. A: Radius of the ball; a: Angle between wall and camera view (α) B: Distance between the camera image impact point and the actual impact point of the ball(Viewing Angle Error); b: Angle between vector orthogonal to wall and viewing angle (β) C: Actual impact point of the ball; D:Ball impact at wall top with small viewing angle error; E:Ball impact at wall top with medium viewing angle error; F:Ball impact at wall top with large viewing angle error; G: Camera

Figure 4.18.: Viewing Angle Error

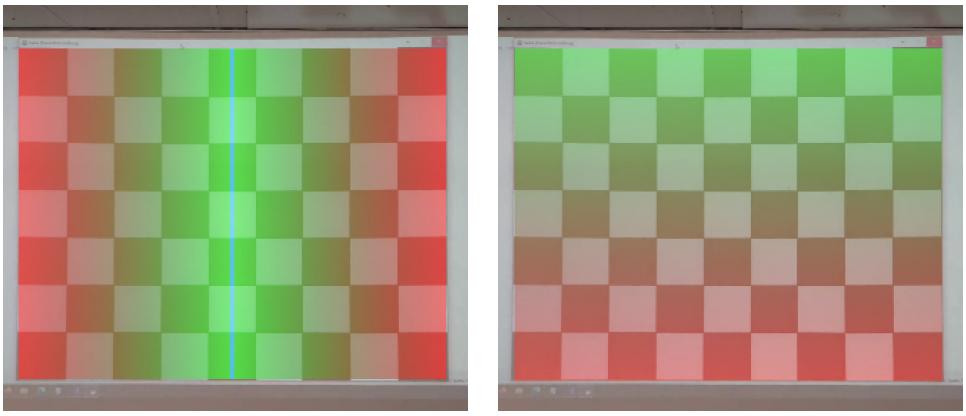
The blue lines in the figure represent the estimated impact point of the ball from the camera image by shooting a straight line from the camera through the balls center. This estimated impact point has the viewing angle error (shown as yellow lines). To calculate the correction, the angle α and the ball's radius at the current position of the ball are needed. This is done by measuring the angle to the camera at the very top (maxα) and the very bottom (minα) of the WIA. This measurement can be done in many ways but whether the angle and radius are measured by hand or by the system through the checkerboard calibration will not be part of this thesis. Once both of these values have been measured, they stay the same as long as the size of the WIA and its orientation to the camera does not change. To determine the correct α for the current ball position, the pixel coordinates of the ball in the camera image are linearly interpolated from maxαspace to minα in regard to the WIA's pixel coordinates origin. So when a ball impacts at the very top of the WIA it has the maxα and when a ball impacts at the very bottom of the WIA it has the minα. This needs to be done the same way with the radius of the ball. Now the angle α and the ball radius A are given for the error correction calculation and they can be inserted into the following formulas to be solved for the corrected impact point C (The following variables correspond to Figure 4.18):

$$180 - 90 - \alpha = \beta \qquad \text{(Calculation of } \beta\text{)}$$

$$tan(\beta) = \frac{B}{A} \iff A * tan(\beta) = B \qquad \text{(Calculation of B)}$$

$$EstimatedImpactPointY + B = C \qquad \text{(Calculation of C)}$$

The error correction calculation according to Figure 4.18 works only for the error correction along the vertical Y-axis. The same principle must be applied with values in respect of the X-axis to correct the viewing angle error in horizontal direction.



(a) X-axis error estimation. Red: Larger error; Green: Smaller error; Blue: Viewing angle allows correct position read without error.

(b) Y-axis error estimation. Red: Larger error; Green: Smaller error

Figure 4.19.: View Angle Error estimation on X- and Y-axis

Figure 4.19 illustrates the error on both axes. Though it needs to be mentioned that despite the fact that this error needs to be corrected for a better impact point calculation, it is not always necessary to do so. In camera setups, where the ball has a relatively large radius in the camera image, the error should be corrected, since it has a greater impact. But as the distance between the camera and the wall increases, the ball radius gets smaller and so does the error. Then, even though the error is still occurring and not corrected, an accurate impact estimation feeling would persist, since the error is too insignificant for the player to notice it. But especially those balls that hit the lower border of the WIA can then be recognised as outside of the WIA which leads to false negative ball impacts even though the user might clearly see that the ball was inside the WIA.

5. Conclusion

A prototype of the proposed system was successfully developed as part of this work, leading to a potential implementation of an interactive wall tracking system for digital sports games. The analysed algorithms proved to be sufficient to complete the task of detection and tracking colored plastic balls as an input device. Even though the chosen algorithms were of simpler nature, they were able to achieve a solid detection and tracking experience. Still, there are many aspects in the proposed system and used algorithms that can be improved by various different solutions.

But when compared to the initially introduced laser grid based approach, the camera based approach was successful to overcome the drawbacks identified in the laser based system solution. However, the camera system also presented a range of more challenging tasks and shortcomings that the laser grid system did not have or solved easier. In conclusion, I would say that no system outperforms the other. The camera system is technically more complex and demanding, this has not necessarily to be assessed as a handicap if the system works as intended. It does require a much larger overhead when it comes to initial setup and calibration, but this work still only needs to be done once for the system to then work in a fixed setup. Future research into alternative technical approaches could lower such complexity and demands and improve the overall system quality and eventually outperform the grid base system.

5.1. Future Work

5.1.1. Depth Sensor and Camera

One of the biggest challenges during development was how to detect the ball in a multi-colored image. This problem could be simplified significantly through the introduction of a depth sensor. This sensor would then be able to recognise the thrown balls in their original shape without the interference of any color or projector lighting. The combination of depth sensor and color image camera can achieve the best of both worlds as the color image enables the differentiation of balls for multiplayer. The use of sensors in which depth sensor and camera are combined in one and already synchronized would solve many difficulties.

As of 2020 OpenCV developed a low cost sensor with depth sensors, camera image and artificial intelligence capabilities that seem very promising for potential use in this context.[1]

[1] OpenCV Store - https://store.opencv.ai/products/oak-d (11.08.2021)

5.1.2. Resolution

To further optimise the running time of the algorithm and achieve better precision, the lower resolution video analysis could be used as an approach to find the general ball position in the image frame. Since the frame has a relatively low resolution, analysing the whole frame will take less time than analysing the original resolution. But when the general position is found, a region of interest is created around that location. This region of interest is then taken out of the frame with the original resolution with the aim to analyse only a small area with high precision. This could further improve run time and precision simultaneously.

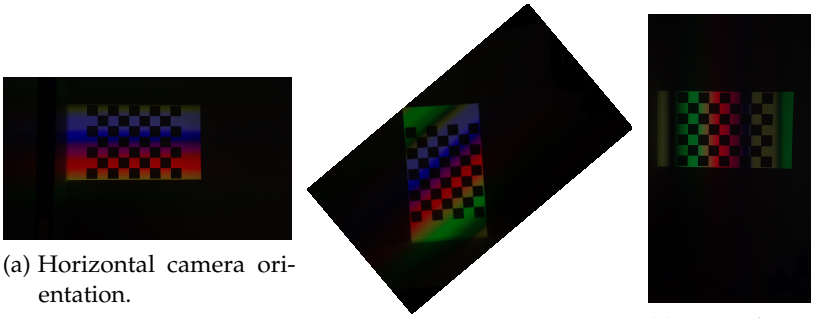
5.1.3. Artificial Intelligence

The algorithm proposed in this thesis is currently based on a more simplified approach to achieve initial important results in the detection and tracking of one or multiple balls. Building on this with further research, the described problematics could be tackled by the use of current artificial intelligence techniques, machine learning, neural networks and related applications.

A. General Addenda

The digital resources of this thesis are kept in this git repository. Additional resources, videos, code, tools, recordings, test results can also be found there. [13]

A.1. Figures



(a) Horizontal camera orientation.

(b) 40-degree angle camera orientation.

(c) Vertical camera orientation.

Figure A.1.: Rainbow lines oriented always parallel to the cameras longer screen side

List of Figures

1.1. Visual System description. 2

2.1. Scheme of a laser grid based system 3

3.1. Illustration of two different projector setups 6
3.2. Example of checkerboard images to be shown by the projector 9
3.3. A comparison view between different camera positions and focal length settings 10
3.4. A comparison view between different camera positions and focal length settings after being corrected through camera calibration transformation. 10
3.5. Example of a detected checkerboard with highlighted inner corner points and calculated outer corner points 11
3.6. A comparison view between different checkerboard calibration images and the resulting camera image transformations (Part 1) 12
3.7. A comparison view between different checkerboard calibration images and the resulting camera image transformations (Part 2) 13

4.1. Comparison of all OpenCV blurring methods when using the same HSV color filter for thresholding. Method values were chosen to run at 30-40 FPS for the testing setup 16
4.2. Comparison of all OpenCV blurring methods when using the same HSV color filter for thresholding. 17
4.3. Comparison an original camera image and the expected result of bilateral filtering created manually 18
4.4. Ball detection filter pipeline (Camera, KNN Filter, Color filter) 19
4.5. Comparison between KNN or MOG2 background subtraction algorithm for impact detection 20
4.8. Comparison of different lighting situations created by the projector image . . . 24
4.9. Comparison of different camera images under same conditions 25
4.11. Comparison of different exposure times recording the Rainbow Effect 27
4.12. Comparison between considered ball types. 29
4.15. Comparison of red, green, blue and yellow ball pit balls under white and cyan (blue and green) light 32
4.16. FPS Comparison 33
4.17. Comparison of the tracking precision for different resolutions 35
4.18. Viewing Angle Error 38
4.19. View Angle Error estimation on X- and Y-axis 39

A.1. Rainbow lines oriented always parallel to the cameras longer screen side . . . 43

Glossary

FPS Frames Per Second. 16, 17, 25, 28, 29, 33, 34, 44

LED Light Emitting Diode. 21

WIA Wall Impact Area. 1–4, 6–9, 11, 12, 14, 30, 36–40

Bibliography

[1] F. R. Statista. *Gaming: The Most Lucrative Entertainment Industry By Far*. URL: https://www.statista.com/chart/22392/global-revenue-of-selected-entertainment-industry-sectors/ (visited on 08/06/2021).

[2] VDH. *Indoor playground members in Germany*. URL: https://www.myvdh.de/der-vdh/hallen-und-indoorspielpl%C3%A4tze.html (visited on 07/22/2021).

[3] S. Yongduek, C. Sunghoon, K. Hyunwoo, and H. Ki-Sang. *Where are the ball and players? Soccer game analysis with color-based tracking and image mosaick*. URL: https://link.springer.com/content/pdf/10.1007%2F3-540-63508-4_123.pdf (visited on 07/22/2021).

[4] N. Owens, C. Harris, and C. Stennett. "Hawk-eye tennis system". In: *2003 International Conference on Visual Information Engineering VIE 2003* 322.10 (2003), pp. 182–185. DOI: 10.1049/cp:20030517.

[5] OpenCV. *OpenCV Homepage*. URL: https://opencv.org/ (visited on 08/05/2021).

[6] M. F. A. Ajmal C. Hollitt and H. Al-Sahaf. "A Comparison of RGB and HSV Colour Spaces for Visual Attention Models". In: *International Conference on Image and Vision Computing New Zealand (IVCNZ)* (2018). DOI: 10.1109/ivcnz.2018.8634752.

[7] G. H. Joblove and D. Greenberg. "Color spaces for computer graphics". In: *ACM SIGGRAPH Computer Graphics, 12(3)* (1978), pp. 20–25. DOI: 10.1145/965139.807362.

[8] A. R. Smith and E. R. Lyons. "HWB—A More Intuitive Hue-Based Color Model". In: *Journal of Graphics Tools* 1 (1996), pp. 3–17. DOI: 10.1080/10867651.1996.10487451.

[9] H. Palus. "Representations of colour images in different colour spaces". In: *Sangwine S.J., Horne R.E.N. (eds) The Colour Image Processing Handbook. Springer, Boston, MA.* (1998). DOI: 10.1007/978-1-4615-5779-1_4.

[10] D. Rozumnyi, J. Kotera, F. Sroubek, L. Novotny, and J. Matas. "The World of Fast Moving Objects". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5203–5211.

[11] E. Powell. *The Great Technology War: LCD vs. DLP*. URL: https://www.projectorcentral.com/lcd_dlp_update7.htm?page=Rainbow-Artifacts (visited on 08/08/2021).

[12] K. Chmielowiec. "Flicker effect of different types of light sources". In: *11th International Conference on Electrical Power Quality and Utilisation* (2011), pp. 1–6. DOI: 10.1109/EPQU.2011.6128852.

[13] J. Landvogt. *Interactive Wall Tracking System for Digital Sports Games*. URL: https://gitlab.lrz.de/IN-FAR/Thesis-Projects/ba-janosch_landvogt-wall_sports.git (visited on 08/16/2021).