Calibrating a Spatial Augmented Reality Environment with Tangible User Interface Using an Embedded Optical Sensing System

Min Shan Luong* Technical University of Munich
Christian Eichhorn† Technical University of Munich
Maki Sugimoto‡ Keio University
Gudrun Klinker§ Technical University of Munich

ABSTRACT

In this report we present an approach to calibrate a projection-based augmented reality environment using a digital projector and a small cube robot with a phototransistor fixed on its topside. The robot is equipped with an internal tracking system to identify its 2D position and rotation on a specific play mat. We focus on determining the relative transformation between play mat and projector and investigate an approach for calibrating a projector as an alternative to well-established approaches, e.g., projector-camera calibration methods. More precisely, we calibrate a projector using measurements from a phototransistor. Finally, we demonstrate our calibration result in a simple game environment and evaluate the accuracy of the measurements and the alignment.

1 INTRODUCTION

Spatial Augmented Reality (SAR) as a subcategory of Augmented Reality (AR) combines the real and virtual world by displaying digital information via projectors and augmenting real objects. What distinguishes it from other categories in AR is that SAR detaches the display technology from the user to the environment [2]. As a result, a group of people, in contrast to a single person, can enjoy the SAR environment together and simultaneously as the visual augmentations are displayed within a shared space instead of requiring separate hardware for each user, such as head-mounted displays (HMDs). Moreover, using projections offers a larger field of view and higher user comfort.

However, a prior calibration of such systems is mandatory for a correct alignment of virtual and real world. This usually requires a calibrated camera in conjunction with the projector. In applications, in which the camera is not involved in any further process apart from the calibration, the camera becomes redundant. Therefore, we investigate an approach to calibrate a projection-based AR environment, which relies on pattern projections and photosensor information, as well as a small cube robot from the toio toy platform [20] whose 2D pose is known within a specific play mat. This robot can be controlled remotely and programmed to move within the play mat area. Alternatively, its position and rotation may be directly modified by physically moving it by hand. This offers possibilities for various applications, particularly for education and entertainment. Together with a calibrated projector setup, the toio system can be further expanded, for example, with visual effects and interactions between virtual and real objects. The robot environment and cube robot can be adjusted to fit any theme, while the robot itself acts as an interaction interface between virtual and real world, since it exists in both worlds.

In the following, we present previous work in the fields of pose tracking using an optical sensors system, and projection-based game or learning environments with augmented objects or robots. Then we describe our system environment and explain our calibration approach in detail. Finally, we evaluate our system quantitatively, as well as qualitatively by means of a simple game environment.

*e-mail: minshan.luong@tum.de
†e-mail: christian.eichhorn@tum.de
‡e-mail: sugimoto@ics.keio.ac.jp
§e-mail: klinker@in.tum.de

2 RELATED WORK

In contrast to other methods that use cameras for location tracking, [18] propose using light-emitting diodes (LEDs) with a passive binary mask as transmitter and photosensors as receiver of the emitted near infrared (IR) light. Multiple LEDs are arranged in an array, and the light of each LED is masked to create binary Gray Code images for spatial modulation, while light intensity sequencing provides a temporal modulation. Taking the epipolar geometry of the LED arrangement into consideration, it is possible to compute the 3D location of a photosensing marker. To determine its orientation, the marker measures the signal strength received from at least four LED beacons.

Augmented Coliseum presents a display-based game environment in which small robots are augmented with projections to fit the setting of toy tanks on a battlefield [12]. The authors use visual effects as for example explosions and laser beams as well as other interactions between real robots and virtual objects to enrich the gaming experience, e.g., force feedback on robots when hit by missiles, and obstacles that are impassable for robots making them bounce back when driven against. A physical simulation that is running in the background detects collisions using virtual models of real objects and purely virtual elements. Their display-based measurement system requires a projector for the augmentations. Additionally, five photosensors are installed at fixed locations above each robot. First, the position of each robot is determined by running spatial division binary search. After the initial absolute position has been identified, the projector projects a fiducial marker at this location so that the orientation of a robot can be calculated using the sensor information and the varying light intensity from the marker. The user can control the robots remotely while the robots are tracked by the measurement system. In addition to tracking a robot's pose, the same measurement system allows the user to control these robots by programming them to follow a fiducial marker.

A Table where Little People Live is an interactive digital art installation by teamLab [25]. Calibrated projectors project little animated people, animals, plants, and objects on a table for educational purposes. Beside other purposes, this artwork aims to encourage its users to nurture their creativity and power of expression, and to develop an understanding of physical laws, namely Newton's laws of motion. Visitors of teamLab's art museums can interact with this artwork using objects or their own hands. This is an example how our augmented toio system could be used to create a learning environment if we substitute the robots with the objects. [13] present another example. The authors propose a tabletop learning environment that helps children understand behaviors of light. The virtual content is displayed either on the surface of a large LCD or with a short-focus projector.

Nonetheless, the objects in the previously mentioned examples can only be influenced in one direction so that virtual objects are not able to affect real ones physically. As explained in [3], robots offer the possibility to appear as if they were affected by virtual objects. This can be achieved if the behavior of a robot, or any programmable object, is programmed accordingly. [3, 12, 14, 19] feature bi-directional interaction between real and virtual objects.

Robot Arena [3] is an infrastructure for building SAR games. Their system requires a projector-camera setup. Its calibration is

done automatically using computer vision or manually by the user.

In [19], the authors use a tele-operated small robot to interact with virtual objects displayed by a projector and a computer monitor. The projector is mounted to the ceiling so that the width of its projection area aligns with the width of the computer monitor. The authors place reflective dots on the robot which enables their motion capture system to localize and track the robot's movement.

In [14], the authors present a tabletop game called IncreTable. It uses a bi-directional projection display to create unique effects that are only achievable by using a two-projector setup, one for rear-projections, and another one for tabletop projections. An example for these kinds of effects are hidden projections, which are only visible by shading certain areas. In these areas, very bright projections from the tabletop projector hide projections from the rear projection screen. Their system supports digital pen input and remote control of robots. The authors use a similar approach to the display-based measurement system in [12] that can be used to both track or control a robot's pose. In [14], the robots follow fiducial markers that are used to control the robots. Alternatively, the user can control them via pen or game pad input or follow predefined paths. Additionally, custom-made tangible objects function as special interface to connect real and virtual world. The user places them manually at fixed locations in the virtual environment. The authors use a depth-sensing camera to track physical objects and capture the terrain of the table which affects the virtual environment.

3 System Overview and Environment

The purpose of our system is to calibrate a projection-based environment so that the projections from the virtual world are aligned with objects that exist in the real world, specifically a small cube robot and its operating space. To achieve this, we determine the projector's intrinsic parameters and estimate the relative transformation between the robot's operating space, namely play mat, and the projector. At the same time, we have a virtual space that mirrors the real world. By applying the previously estimated transformation on our virtual camera, we obtain a 6DoF (Degrees of Freedom) pose that is aligned with the projector. As a result, we obtain an aligned environment in which we can augment real objects. In the following, we present an overview of our calibration system and its environment.

3.1 Overview

Our system consists of the following main components:

- Toio system containing cube robot and play mat with a printed pattern [20].
- Photosensor (phototransistor).
- Microcontroller (MCU) with Wi-Fi module; we use the ESP-WROOM-02 development board (AE-ESP-WROOM-02-DEV) [7].
- Computer running a Unity project [26] with toio SDK for Unity [15]
- Calibrated projector mounted above the toio play mat with a resolution of 1920×1080 pixels

The robot's 2D position and rotation is known within the toio coordinate system as long as it is located on the play mat. This information allows us to create a virtual copy of this scene inside Unity. This virtual setup is described in Section 3.3. With the play mat's coordinate system aligned with Unity's world coordinate frame, we can obtain 3D positions of the robot. In Augmented Coliseum, it is necessary to restart the pose identification procedure of a robot when its photosensors lost their follow target, a fiducial marker [12], e.g., due to occlusion of one or multiple photosensors.

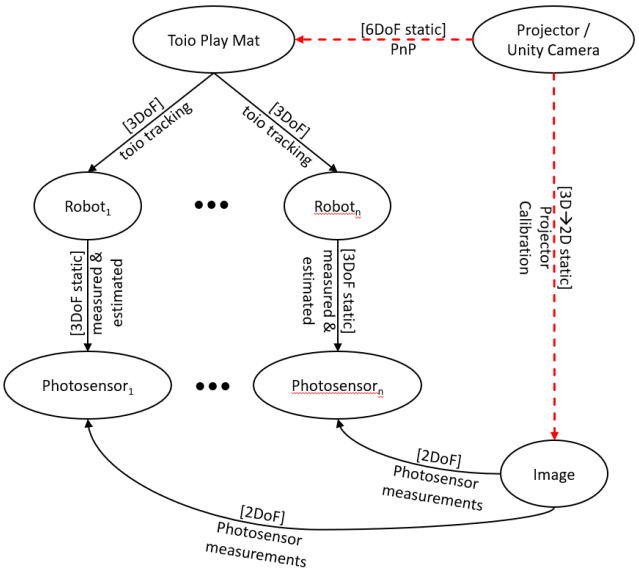

Figure 1: Spatial relationship graph for our system created according to [5]. One subproblem is determining the projector intrinsic parameters for projecting 3D points to 2D image points for which the 2D-3D Projective Calibration pattern [17] applies. Assuming the projector calibration to be known, the 2D-3D pose estimation pattern presented in [17] applies. Both are common problems. The latter can be solved, e.g., by solving the Perspective-n-Point problem.

In our scenario, the 2D position and rotation of the toio robot is known within the provided play mat via an internal optical tracking system. Therefore, it can recover its pose instantly.

In Prakash [18], specifically designed projectors are needed which cannot be used for augmenting our toio cube robot environment. Therefore, we use a standard projector instead of multiple LED arrangements. We divide the projected 2D image from the projector into subareas and temporally encode the projector image space with sequential binary Gray Code images as proposed in Prakash [18] (see Section 5). The projected pattern is received by a phototransistor, a type of photosensor for measuring brightness that is part of our embedded optical sensing system (see Section 3.2). This phototransistor is attached to the cube robot. Hence, we can identify the 2D position of a cube robot in this projector image space by decoding the information. This step in our calibration procedure is similar to the one of Augmented Coliseum [12] as Kojima et al. identify the position of a robot in projector space using photosensor measurements. However, instead of using multiple sensors, we propose a single sensor approach that uses multiple measurements of different 2D position to determine a robot's pose. To find a suitable approach to calibrate our system, we created a spatial relationship graph describing our setup (see Fig. 1). It shows that we can estimate the 6DoF transformation between a calibrated projector and the play mat, thus, also the robot itself, using the identified 2D positions together with their corresponding 3D points from the virtual world.

Our embedded optical sensing system plays an important role in collecting data to estimate the pose or the relative transformation between the play mat and the projector. We describe the development of this optical sensing system in Section 3.2. Moreover, this optical sensing system allows us to investigate an approach to determine the projector's intrinsic parameters that is based on photosensor measurements. We compare this calibration method to a well-established one that uses a camera in addition to the projector. Both calibration approaches are described in Section 6. A well calibrated projector is essential for our projector pose estimation algorithm as shown in

Fig. 1. Therefore, we calibrate the projector prior to estimating its 6DoF pose.

In addition to the play mat and a robot, we set up a camera in the virtual environment representing the projector from the real world. Using a sufficiently large set of point correspondences, we determine the 6DoF pose of the projector relative to the play mat (see Section 4). After the pose of the projector has been determined, we apply the same pose on the camera in the virtual setup. As a result, the virtual scene is aligned with the real world's setup when the projector displays images from the virtual camera. Moreover, we can extend interactions between objects in the virtual world to interactions between both virtual and real world objects. The idea is similar to [12] which has a physical simulation running in parallel. In Section 4.3, we shortly comment on the 3D projections within the toio environment.

To validate our calibration, we set up a game environment in which the robot can interact with virtual objects (see Section 7). Inspired by [3, 12, 14, 19], we added virtual obstacles that affect the driving behavior of the robot. We use Unity [26] to create our virtual environment. The program for the MCU is written in Arduino Sketch.

3.2 Embedded Optical Sensing System

In this section, we present the design of our embedded optical sensing system. It comprises an MCU with WiFi module, a photosensor which is in our case a phototransistor, and several resistors.

For designing the circuit, we built it on a breadboard and connected it to the computer. Using this connection, we can send data between the MCU and the computer via serial communication. The photosensor is connected to the MCU to read the phototransistor's sensor value. Between these two components, we inserted several resistors that are necessary for two reasons. On the one hand, we use them to control the photosensor's sensitivity level. To determine a suitable resistance value, we assessed different resistances by using an oscilloscope until we obtained meaningful sensor values from the phototransistor. In the end, we chose the resistance for which the photosensor located in an area with a black or white projection returns values that are far away from each other in these two cases. On the other hand, the pin on our MCU that is used for reading analog values (TOUT pin) is limited to detect input voltage values ranging from $0V$ to $1V$ [4]. Hence, we reduced the input voltage to a maximum of approximately $1V$ to avoid damages on the development board since the standard for power supplies is $3.3V$ or $5V$. One method to reduce voltage is by building a voltage divider that requires at least two resistors. The formula for calculating the output voltage from a given input voltage and two resistances is:

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2} \tag{1}$$

For our first trial, we used an LED as a photodiode light sensor as described in [1]. The advantage of LEDs over phototransistors is that it has a quicker response time, hence it can react faster to signal changes. However, it requires further adjustments to stabilize the sensor signals and obtain meaningful values even after adjusting the resistance. Therefore, we use a phototransistor in the current version. Our circuit design using a phototransistor is shown in Fig. 2.

For the final version, we created a smaller version of our circuit design for the cube robot to carry. To decrease the number of cables, we use a printed circuit board (PCB) that connects the electrical components. We store this circuit with its components in a customized 3D-printed plastic case that we attach to the topside of the robot. We designed the case using a computer-aided design (CAD) software such that the photosensor can be inserted above the robot model's local origin (see Fig. 3).

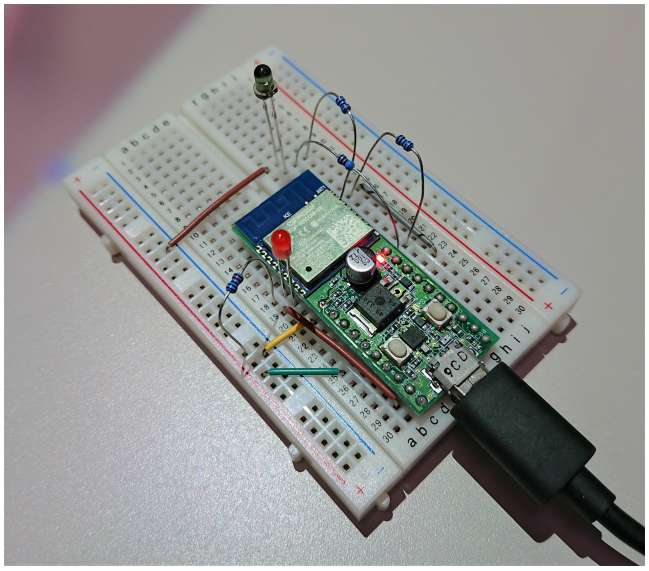

Figure 2: Circuit design on a breadboard. We use a $68K$ Ohm resistor and a voltage divider with $10K$ and $20K$ Ohm resistors because the TOUT pin can only take voltage from 0V to 1V. To validate that a WiFi packet has arrived, we added an LED and toggle between on and off on arrival of a WiFi packet. The LED is not part of the final circuit.

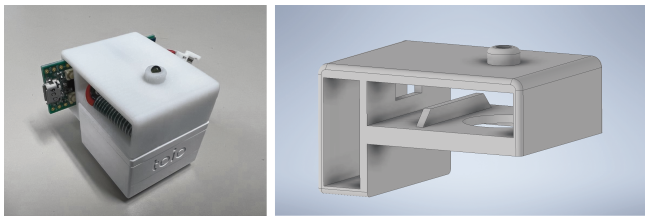

Figure 3: Right: final electrical circuit fit into a 3D-printed case. The phototransistor is placed in the whole in the topside of the case. Left: 3D model for the 3D-printed case.

3.3 Representation of Toio and Projector in the Virtual World

Our objective is to align the real world with the virtual world. Therefore, we require data from both worlds, which in our case are 2D-3D point correspondences. In this section, we describe the acquisition of the 3D points in the virtual space.

Sony and Morikatron provide a software development kit (SDK) for developing applications in Unity [15]. It comes with a textured 3D model of the cube robot and the play mat which we use to set up a virtual scene that mirrors the real world in Unity. Additionally, we add the 3D model of our case for storing the optical sensing electronics to create a virtual environment looking close to reality. Unity has a world coordinate space in which all 3D-objects are located. Each object in turn define their own object space. For simplification, we keep the play mat at the world origin so that its object space and world space are aligned. The toio cube robot is equipped with an identification sensor on the bottom of the cube. The pattern printed on the play mat holds unique information which the sensor can read to identify the position of the robot on the play mat [22]. Hence, we can assume the pose of the cube robot to be known relative to the play mat since the toio system tracks the robot internally. The toio SDK offers functions to convert information in toio coordinates to Unity coordinates. Therefore, we can mirror the real toio environment in the virtual scene in Unity. We synchronize

the pose in Unity by setting the position and rotation of the virtual cube robot according to the pose of real one.

The projector is represented in the virtual world by a camera that captures the virtual scene. Unity offers the option to use physical camera parameters for the virtual camera. Thus, we apply the intrinsic parameters obtained from the projector calibration (see Section 6) on the camera in Unity. Then, the projector projects the camera images into the real world to augment the toio environment. As the virtual camera and the projector are initially not aligned, we need to determine the relative transformation between the play mat and the projector. This procedure is described in Section 4.

4 Pose Alignment of the Projector and the Virtual Camera

Our goal is to identify the 6DoF pose of the projector relative to the play mat so that we can apply this pose on the virtual camera and accomplish an alignment of virtual camera and real projector. As shown in Fig. 1, we can achieve this goal by solving the Perspective-n-Point (PnP) problem with a set of 2D-3D point correspondences. The PnP problem describes the problem of estimating the 6-DoF pose of a calibrated camera with a set of n point correspondences. The 3D points are points in the world or object coordinate system, whereas the 2D points are given in image coordinates. Our setup allows us to collect 2D-3D point correspondences which is described in the following section. Furthermore, we describe how we use the estimated pose to achieve an alignment of the projections with the real objects and comment on the effect achieved using this calibration approach.

4.1 Collecting Point Correspondences

First, we collect sufficient point correspondences. Optimally, the points should be spatially distant so that small errors would not change the estimated pose significantly. Therefore, we chose points on a regular grid that covers the whole play mat with a reasonably large resolution to have enough point correspondences. We instruct the robot to drive to each of these points and project a pattern sequence to identify the photosensor location in the image space of the projector as described in Section 5. We record 3 measurements per axis of a point and calculate the median value of each axis respectively to counteract 2D measurement errors that might occur due to an unsynchronized timing of projections and sensor data requests. These points are our set of 2D points. Furthermore, to decrease the measurement errors, we suggest to adjust the threshold for detecting black or white areas, e.g., using normalization images and other enhancements described in [16], so that our method works under various ambient light conditions and with different projector brightnesses. Moreover, the range of the read sensor values differ dependent on the light's angle of incident on the sensor surface. This is a possible cause for wrongly identified 2d positions and should be taken into consideration.

The corresponding 3D points are collected in the virtual scene in Unity. Our virtual scene mirrors the real toio environment meaning that the robot's pose is known (see Section 3.3). In our setup, the play mat's coordinate system is aligned with Unity's world coordinate system. However, we need to consider the offset of the phototransistor to the robot. We estimate this offset with the knowledge we have about the position of the LED in the 3D-printed case for the MCU and phototransistor circuit (see Section 3.2), and the origin of the robot's virtual model [23]. To each 2D point, we save the respective 3D coordinates of the photosensor in Unity to create our set of 2D-3D point correspondences. Parts of the data collection process is shown in Fig. 4.

The 2D-3D point correspondences are saved in a JSON-file so that we can load them and thus skip the 2D-point acquisition step as long as the play mat and the projector pose remain unchanged. Furthermore, we can use this to expand our input data for the projector calibration using photosensor measurements (see Section 6.2).

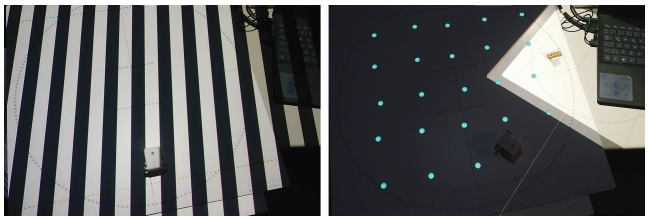

Figure 4: Our process of data collection (2D-3D point correspondences). Left: Structured light pattern sequence projected on the play mat and the robot. Right: point grid of the already saved 2D points. On the right, the virtual play mat is shown from the view of the virtual camera whose pose is not yet aligned with the projector's pose. The robot's pose relative to the play mat is already mirrored in the virtual environment before the pose alignment.

4.2 Estimating the Pose

Together with the intrinsic matrix of the projector that we obtained from the projector calibration (see Section 6) and the point correspondences, we estimate the pose by solving the PnP problem using OpenCV for Unity [6]. It assumes the pinhole camera model. In addition to solving the PnP problem, we use RANSAC to reject outlier points. Outliers may occur due to errors during the 2D point acquisition. The OpenCV library offers a PnP solver with RANSAC in the Calib3d module. Finally, we set the pose of the camera in Unity by applying matrix multiplication with the obtained extrinsic matrix from OpenCV. During all steps, we need to consider that Unity uses a left-handed coordinate system while OpenCV uses a right-handed one. Therefore, we convert both Unity 2D image coordinates and 3D world coordinates to their respective OpenCV spaces when necessary (e.g. before giving the point data sets to OpenCV for estimating the projector pose), and vice versa (e.g. when applying the pose to the virtual camera in Unity).

4.3 Three-Dimensional Projection

In the current implementation, the virtual scene is a perspective projection rendered from the view of the camera which is approximately in alignment with the projector. This allows us to project texture on all real objects that, at the same time, have a representation in the virtual world aligned in 3D. The visual result is comparable to the effect achieved using projection mapping. Thus, at those parts in the image, in which from the perspective of the projector there is a real object and we want to simply change the texture or anything on its surface, the current rendering method works well.

However, for virtual 3D objects that don't exist in the real world, it could be better to render an image from another view that is centered above the play mat. By rendering the scene from a centered top-down perspective, we would obtain projections that are less view-dependent so that projections of purely virtual 3D objects on the play mat look correct for multiple users regardless of their viewpoint. When rendering top-down, we want to keep the three-dimensionality of the virtual world. As a result, objects higher above the play mat appear larger in the image than the ones closer to the mat. Moreover, the shadow size is dependent on the distance between the shadow caster and shadow receiver. We consider these aspects in our current implementation that displays a perspective projection from the view of the virtual camera.

5 Data Communication and Temporal Encoding

In the following, we describe how data is communicated between the different components of our system. The computer is the central node of communication between the different components and is

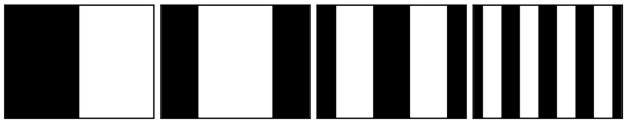

Figure 5: The first four pattern images of a Gray code sequence encoding the projector columns. From left to right: encoding of the most to least significant bit. Taken from [24].

responsible for their synchronization. Each of these components is necessary for our approach to temporally encode the projector's image space with a sequence of structured light patterns. This encoding is used to identify the position of the photosensor in the projector image space. Synchronizing the timing of each unit to send data is crucial to ensure that the code is received correctly.

The communication between the computer and toio robot is wireless via Bluetooth. The robot sends information about its own 2D pose on the play mat to the computer via this connection. In Section 3.3, the basic concept of how the toio system identifies the cube's pose is described. The reverse information flow from the computer to the robot can be used to send commands to the robot, for example playing sounds or specific driving behaviors. One option to control the robot from the computer is by indicating a target position with the mouse cursor on the virtual scene, or with a list of pre-defined target positions on the play mat. Another option the toio SDK offers is to instruct the robot to drive with separately assigned wheel speeds, each for an arbitrary amount of time.

Additionally, the computer is connected to a projector that projects a series of black and white patterns. These patterns divide the image space of the projector into smaller regions. More precisely, it is a temporal encoding of these regions as binary Gray Code with each image corresponding to one digit as it is done in [18].

Another name for Gray Codes is *reflected binary code*. The subsequent pattern image can be created from the previous pattern image by reflecting and appending this reflected image to the previous one which is exactly what we do (see Fig. 5). The advantage of Gray Codes over simple binary codes is that two neighbouring values (e.g. two neighboring areas in the encoded image space) differ by only one binary digit (bit) resulting in a Hamming distance of 1. In the case of projected structured light patterns, this means that this method is more robust to decoding errors compared to a simple binary encoding [24].

By displaying a Gray Code image pattern sequence along the horizontal and vertical axis, we create a regular rectangular grid with self-defined resolution ($2^m \times 2^n$; m and n equal to the number of binary digits per axis) for encoding 2D positions in the pattern coordinate system.

Ideally, the resolution should be close to the one of the projector image. Hence, we chose $m = 10$ and $n = 11$ resulting in a pattern resolution of 2048×1024 pixels. This resolution ensures that every pixel in the projector image, which has a resolution of 1920×1080 pixels, is covered by approximately one pixel in the pattern coordinate system.

The MCU comes with a Wi-Fi module that we use to establish a Wi-Fi network on the MCU. Then we connect the computer to the Wi-Fi network. As there might be issues with the security settings of the computer, we set it as private network and deactivate the network security system to allow incoming messages within this network. The computer sends a single char command as UDP packet through this network to request signal data from the phototransistor that is connected to the MCU (see Section 3.2), or other data depending on the command. On arrival, the MCU sends the current value from the photosensor back to the computer as UDP packet. This data has to be sent as byte array. We encoded with an initial char and an closing

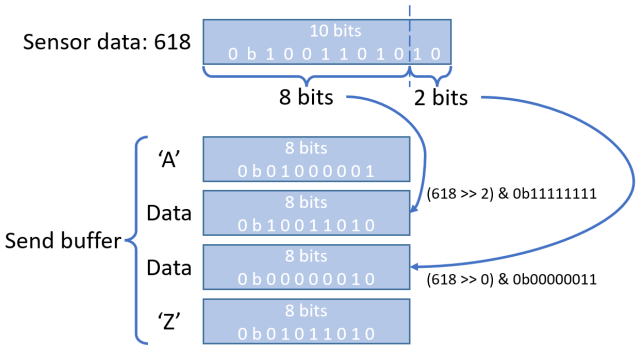

Figure 6: Send buffer containing 10 bits of sensor data with enclosing chars in binary representation. In this example, the sensor data we want to send is 618. To pack the data in a byte array, we need to split it in two $8bits$ chunks. To obtain the first chunk, we apply a binary shift by 2 followed by a bit mask on the sensor data value. Concerning the second chunk, we do not need a binary shift but we apply a bit mask to keep only the last two digits (the two least-significant bits).

char to ensure that a whole packet arrived on the computer. Since our data ranges from 0 to 1023, a bit shift is required to cover 1024 values or 10 bits (see Fig. 6).

On the computer, our program decodes the message and checks for transmission errors. In case the received message was corrupted, we send another request and repeat the procedure. Depending on a pre-defined threshold, our program interprets the signal from the photosensor as black or white, meaning that it is located within a black or white region of the projected pattern, respectively. We store this information in a binary representation. A white area corresponds to 1, and a black one corresponds to 0 in our Gray Code. As advised in [16], it is recommendable to display normalization images to determine an appropriate threshold depending on the current lighting condition and the brightness of the projector.

By appending data from the photosensor for each image in the pattern sequence at a fixed location, we obtain a binary Gray Code that we decode to identify a coordinate along one axis in our pattern coordinate system. Because the input lag of the projector is unknown, we wait for a predefined time before requesting measurements from the photosensor for the respective pattern image and binary code position. Repeating the same procedure along the other axis results in a 2D coordinate that differs from the projector's image coordinates. More specifically, it is a 2D coordinate in the previously defined Gray Code pattern coordinate system. To obtain the photosensor location in the projector's image space, we map the pattern coordinates to a range from the center of the minimum pixel to the center of the maximum pixel in the projector image space for each dimension (see Fig. 7). We decided that the Gray Code coordinate points are located at the center of a pixel in the Gray Code pattern coordinate system since we only know that our identified photosensor position lies within this pixel area, but its position within the pixel itself is unknown. Hence, assuming our object location to be in the center of a Gray Code pattern pixel could minimize the quantization error.

6 Projector Calibration

We intend to set up the virtual camera in a way that it holds the properties of the projector. Therefore, we need to determine the projector's intrinsic parameters. Applying these intrinsic parameters to the virtual camera is necessary for correct perspective projections from Unity's physical camera, and for estimating the 6DoF pose of the projector in our approach (see Fig. 1). For this purpose, we tested two different methods. This allows us to compare the results

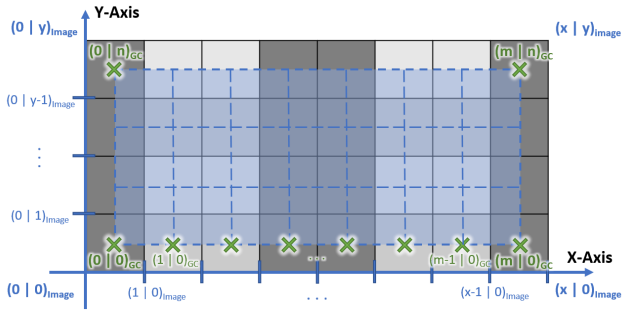

Figure 7: The image coordinate system of the projector (blue) marked with coordinate points in the pattern coordinate system (green). In this schematic visualization, the grid defined by the pattern sequence and the pixel coordinate system of the projector image are aligned for simplification. In the background, we show the third pattern image of a Gray code sequence encoding the projector columns for bit MSB 2 (third-most-significant bit). We map each pattern coordinate to an projector image coordinate within the area highlighted in blue.

from a well-established camera calibration method with our own method. In both cases, we assume that the projector's lens distortion is insignificantly small. Hence, we neglect it and fix the distortion parameters to zero.

6.1 Procam Calibration

In our first approach, we use the projector-camera (procam) calibration from [8]. It calibrates a projector and a camera given a set of checkerboard images taken by the camera. Two images are required for one pose. One shows the checkerboard pattern with color-coded spatial structured light pattern projected on it to determine initial projector intrinsic parameters, and another one with even lighting for calculating initial camera parameters using the method from [27]. After establishing initial intrinsic parameters for the projector and the camera, the authors use bundle adjustment to refine the results that may contain errors arising from an imperfect planar calibration board. Since the accuracy of the calibration result is highly dependent on the quality of the input images, we created multiple sets of images while varying the pose of the checkerboard pattern. For each set of images, we aim to cover the whole image space with checkerboard poses that highly differ from each other, and manually filter out images that are not suitable for achieving an accurate result as for example images, in which the pattern is wrongly detected, blurry images, or images with an unfavorable lighting (see Fig. 8). Furthermore, we use a calibrated camera, an Intel RealSense LiDAR Camera L515 [9], to have a ground truth for the camera matrix. We request its factory calibration for its color image stream according to [10] and compare this camera intrinsic matrix to the ones resulting from the procam calibration. Our candidate projector intrinsic matrices are the ones whose respective camera intrinsic matrix is similar to the one of the factory calibration. We confirm that the procam calibration outputs reliable intrinsic matrices depending on the quality of the input images and the visibility of the checkerboard and structured-light pattern.

6.2 Calibration Using Photosensor Measurements

We compared the results from the procam calibration to the ones of our own method that relies on measurements from the phototransistor. Our method uses the implementation for calibrating a camera in OpenCV for Unity [6]. It requires at least one set of 2D-3D point correspondences as input parameter. However, if we use only points from a single view, the estimated camera matrix would only re-project the 3D points from our input data correctly for this specific estimated camera pose. To avoid the case that errors

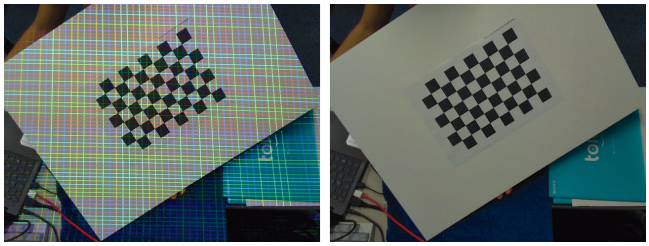

Figure 8: Images of a checkerboard pattern attached to a white board that were used as input for the procam calibration tool by [8]. The checkerboard pattern has a resolution of 7×10 with a square size of $25mm$ and is printed on a DIN A4 paper. Left: color grid. Right: even lighting

in the estimated intrinsic parameters compensate for other errors, e.g., in the extrinsic parameters, and vice versa, we need points from multiple views. Therefore, we vary the projector's pose to collect data sets from different perspectives. We alternate the pose mainly by rotating the projector since we are limited by the height of the ceiling and the size of the projection area which needs to be large enough to cover about 90% of the play mat. We manually filter out sets with obvious outlier whose identified 2D position in the projector image space deviates strongly from the real phototransistor position. We save all remaining data sets in a JSON-file to load and use them as input for OpenCV's camera calibration function.

We retrieve the 3D points from our virtual representation of the real setup in Unity (see Section 3.3). The procedure for obtaining 3D points per projector pose is described in Section 4.1. Moreover, per projector pose, these points have to be co-planar with the z-coordinate equal to 0. Hence, we transformed the points to comply with the requirements of OpenCV's implementation. To obtain the respective 2D-point positions, we spatially divide and encode the projector's image space as it is proposed in [18]. The projector displays the encoded information in form of an image pattern sequence along two axes, which is then received by the photosensor. Finally, we decode the data and convert the identified 2D position to a position in the projector's image space. The procedure for obtaining 2D points is explained in more detail in Section 5 and 4.1. After collecting data from 15 different projector poses, we estimate the intrinsic parameters of the projector using OpenCV for Unity.

Our current implementation of the photosensor measurement approach is not as reliable as standard calibration techniques using a camera and computer vision since the identified 2D positions might deviate from the actual positions due to inaccurate sensor measurements. It is necessary to integrate an outlier rejection algorithm to make our projector calibration method more robust. Moreover, our approach can be improved by making it less prone to measurement errors (see Section 4.1). However, under optimal conditions, our calibration approach yields similar results to the procam calibration. A comparison of the resulting intrinsic matrices from each of the projector calibration approaches is shown in the following (three matrices from the procam calibration, one matrix from the calibration using photosensor measurements):

$$projectorMat_{(1,procam)} = \begin{pmatrix} 2376.313 & 0 & 1009.074 \\ 0 & 2383.285 & 1005.604 \\ 0 & 0 & 1 \end{pmatrix}$$

$$projectorMat_{(2,procam)} = \begin{pmatrix} 2396.889 & 0 & 978.934 \\ 0 & 2394.495 & 998.202 \\ 0 & 0 & 1 \end{pmatrix}$$

$$projectorMat_{(3,procam)} = \begin{pmatrix} 2388.296 & 0 & 983.430 \\ 0 & 2385.895 & 1002.125 \\ 0 & 0 & 1 \end{pmatrix}$$

$$projectorMat_{sensor} = \begin{pmatrix} 2349.9054 & 0 & 984.4547 \\ 0 & 2351.2605 & 1077.1596 \\ 0 & 0 & 1 \end{pmatrix}$$

7 Results and Evaluation

The objective of our calibration approach is to have an interactive, projection-based environment augmented with synthetic information which are aligned with objects existing in the real world. To analyze our outcome, we demonstrate the results of our calibration in a simple game environment which is described in Section 7.1. For an accurate estimation of the projector's extrinsic and intrinsic parameters, the accuracy of the input data is essential. Therefore, we evaluate the accuracy of the identified 2D position of the photosensor. Furthermore, we point out the limitations and problems of our system in Section 8.1.

7.1 Qualitative Evaluation

We developed an object-catching game to implement a qualitative analysis of our calibration outcome. environment. The goal of this game it to catch as many collectibles as possible while they are falling from above. We developed this game in Unity [26] as a subsequent stage after the calibration process is complete. Our game environment features following elements:

- Player unit (cube robot with visual augmentations)
- Play mat as driving area for the robot and as projection surface
- Collectibles (green spheres) spawned by a thrower unit
- Puddles
- Walls

The cube robot can be controlled by the user via mouse input. It can be driven within the play mat. However, as soon as the robot leaves the play mat, the user needs to manually reset the cube to a position on the play mat for its pose to be tracked and updated in the virtual environment. The robot's pose can be instantly recovered due to toio's internal tracking system. This means, that the user has also the option to change the robot's pose on the play mat by directly manipulating it by hand. Since the robot is augmented with a texture, we advise the user to mainly use mouse input to control the robot so that the texture can be projected onto the actual projection target without distractions.

To avoid the case in which the robot accidentally drives off the play mat, we added virtual boundaries along each side of the mat. As these walls are only existing in the virtual environment, we detect collisions with the cube using its virtual 3D twin. Collision detection with other virtual elements works in the same way. When the system detects that the robot drove into a wall, the robot automatically starts driving backwards for a certain amount of time.

Beside the walls, we project collectible spheres on the play mat. These collectibles are spawned next to the virtual mat and fly in a high curve over the play mat until it reaches the ground. The flying curve can be estimated by the size and the speed of a sphere's shadow on the mat before the sphere itself enters the camera view frustum. The robot is augmented with a plane that indicates approximately the bounding box of this player unit. By driving the robot to the location of a collectible before it touches the ground, the player can collect the sphere. The number of collected spheres is then updated and displayed on the player unit plane. To make a successful catch more apparent, we added visual effects and sound played from the cube.

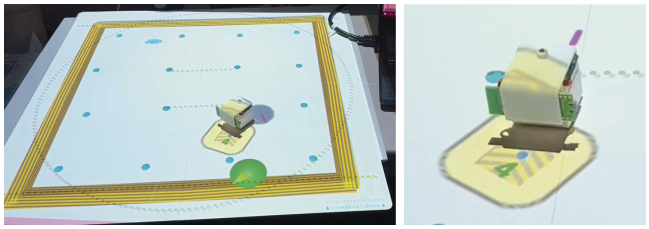

Figure 9: Game environment projected onto the play mat after the camera has been aligned with the projector. A lag between virtual and real world player unit is visible.

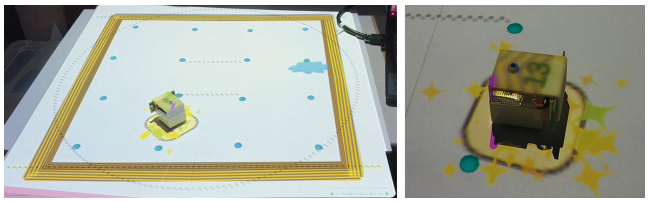

Figure 10: Game environment showing aligned virtual and real objects. As we project the virtual play mat on the real one, we can see the displacement of the texture.

Furthermore, puddles are spawned randomly on the virtual mat and decrease in size until they eventually disappear. When the robot drives over a puddle, it starts spinning for a certain amount of time and the user temporarily loses control. Programming the robot's behavior allows us to imitate certain events in the real world as for example slipping on puddles.

The object collision detection with the robot only works as fast as the pose of the robot is updated in the virtual environment. In some cases, when the pose is not updated frequently enough, the robot becomes stuck in the virtual wall. The update frequency is dependent on the notification interval of the robot's identification sensor. We set the minimum update interval to 0 milliseconds to obtain the most recent information about the robot's pose whenever possible. However, "the notification interval may vary depending on the status of the central unit" [21]. We measured the elapsed time between updates received on the computer which is ranging from approximately 30ms to 100ms. This can cause visual lags of the player unit as visible in Fig. 9. Other reasons contributing to this lag could be the communication speed between all involved components, specifically

- the latency for sending messages via Bluetooth between computer and robot, and
- the input lag for sending rendered images from the computer to the projector and displaying them.

Apart from the small amount of lag that is visible in Fig. 9, our system looks well calibrated for a game environment (see Fig. 10). However, we can observe misalignments. This becomes evident when we for example compare the distance of the components existing in the real world to their augmentations, i.e. the projected texture of the play mat and the texture of the cube robot with wire case.

7.2 Accuracy of Photosensor Measurements and Calibration

In most of the cases, the sensor value measured at the same position in the real world results in the same value. In 11 out of 11 measurements at the same position, the identified 2D coordinate in the projector image space remained unchanged. When we inspected

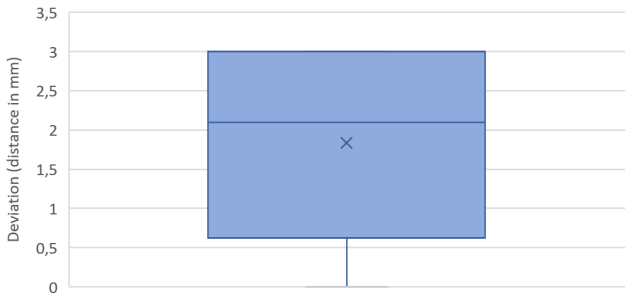

Figure 11: The distance from the identified 2D position of the photosensor in projector image space to its actual position measured in millimeters. In total, we recorded 8 measured distances for this diagram. The distances range from $0mm$ to $3mm$ The mean deviation is $2.1mm$.

one of our collected data sets with 3 measurements per point in each axis, all three values of each the x and the y coordinate per point were identical for 24 out of 25 points. The single outlier exhibits a deviation of less than 5 pixels along one axis in the projector image space while the value in the other axis stayed constant over all three measurements.

After we identified a 2D position in projector image space, we draw a gizmo, in our case a sphere, at this location which is visible in the projection (see Fig. 12). Then we compare it with the actual photosensor location and measure the distance between the photosensor and the projected sphere. We observe a deviation of up to $3mm$ for 8 measurements at varying positions (see Fig. 11).

We added a small sphere relative to the virtual robot at the location at which the photosensor is located relative to the robot in the real environment (see Fig. 9 and Fig. 10). We use this sphere as a visual indicator for the position of the photosensor in the camera image space after estimating and applying the projector pose on it. To evaluate our results of both the intrinsic matrix and the estimated extrinsic matrix, which are highly dependent on accurate photosensor measurements, we measure the distance between the real and the virtual photosensor position. Within 6 calibration trials, we observe a maximum deviation of $10mm$ at different positions (see Fig. 13). Since the result from the whole calibration process shows small misalignments, there might be errors in the measured data or the estimated intrinsic and extrinsic parameters. A possible issue could be errors in terms of the 3D positions of our point correspondences. The cause could be that we manually calculate and partially estimate the height of the phototransistor. Moreover, the toio tracking might be not accurate as the position and orientation values of the robot from its internal identification sensor oscillate even if the robot is not moving in the real environment. This would influence our 3D data points used for the projector calibration as well as the projector pose estimation. The maximum re-projection error (root-mean-square error) for all inlier points after estimating the pose by solving PnP with RANSAC is less than 1.3 pixels.

8 Conclusions and Future Work

We demonstrated in a game environment that we achieved a decent alignment of the virtual and real world by means of our SAR calibration approach. Furthermore, we have shown that a projector calibration without a camera is feasible using robots with a photosensor attached. Our approach yields results that are comparable with the ones from a procam calibration software. Compared to a procam calibration, it is slightly less accurate and more time consuming.

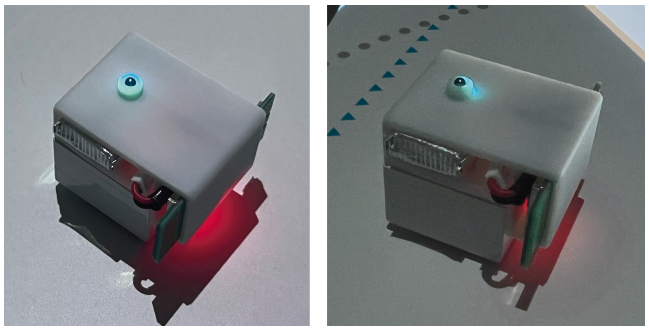

Figure 12: A projected sphere indicating the identified photosensor location in projector image space.

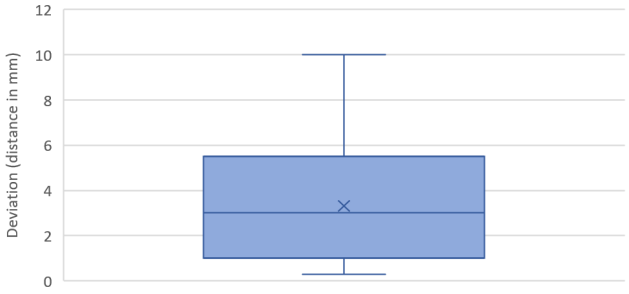

Figure 13: The distance between the projected photosensor position after the system has been calibrated and its actual position in millimeters. This diagram contains 13 measured distances ranging from approximately $0.3mm$ to $10mm$. The mean is at $3mm$.

8.1 Limitations and Problems

Our system once detects the projector pose and respectively transforms the virtual camera relative to the play mat. This means that if the play mat or the projector is moved, we need to repeat the calibration procedure. During the calibration, the projection of the structured light pattern sequence and requesting and reading photo sensor values need to be timed correctly. Because of the input lag of the projector, the sensor value could otherwise be read when the respective pattern image for the currently requested position in the binary digit array is not displayed. If we were determining the input lag of the projector in advance, we could speed up the process of obtaining photo sensor values after projecting the next structured light pattern in the pattern sequence. Furthermore, the photosensor measurement is currently not stable for different projection angles and is highly dependent on the angle of incident of the light rays coming from the projector. Moreover, the ambient lighting is not taken into consideration. Thus, we only receive meaningful sensor values with approximately the same ambient lighting. The used structured light encoding method becomes less accurate the more narrow the Gray Code stripes become. The projected stripes become increasingly blurry and the difference in intensity of black and white areas become less clear such that it becomes more challenging to differentiate between these two areas [11].

8.2 Future Work

In future works, we suggest to explore the option to replace the phototransistor with an LED for a faster response time when reading sensor values. Furthermore, we recommend to adjust the photosensor's threshold for differentiating between black and white areas in the future. The position of the phototransistor could also be registered properly to the robot coordinate system, as we are currently

only estimating its position based on the 3d-printed casing, of which we know the dimensions and mounting position.

The current setup uses only one projector. However, using multiple projectors could be a possible extension to show texture projected on the registered real objects from multiple perspectives.

The toio system supports the tracking of multiple cube robots. It could be beneficial to add a second toio robot equipped with our embedded optical sensing system to speed up the collection process of the point correspondences. With regard to the object-catching game, it would be possible to introduce a second player with the second cube. Using this second cube, the second player could throw collectibles from outside the virtual boundaries and change the position from which the collectibles are thrown by hand. For example, the second player could throw a collectible from the second cube's current position in its forward direction by hitting its topside as this kind of input can be recognized by the toio system. In this way, the area outside the virtual boundaries can be utilized as well. Moreover, the application becomes a multiplayer game which could add the fun of playing together.

Acknowledgments

I wish to thank Prof. Maki Sugimoto for advising and supporting me throughout this project both on-site and remotely. I appreciate his help, particularly in creating the small-sized circuit for the MCU and the photosensor, and in designing a 3D-printed case for these components. Many thanks to Christian Eichhorn who gave advice to me remotely during regular meetings with a focus on scientific work.

References

[1] Analog Devices Inc. Activity: Led as light sensor: Adalm2000.

[2] O. Bimber and R. Raskar. *Spatial augmented reality: Merging real and virtual worlds*. Peters, Wellesley, Mass., 2005.

[3] D. Calife, J. L. Bernardes, and R. Tori. Robot arena. *Computers in Entertainment*, 7(1):1–26, 2009. doi: 10.1145/1486508.1486519

[4] Device Plus Editorial Team. Using esp-wroom-02 wifi module as arduino mcu, 09.01.2018.

[5] F. Echtler, M. Huber, D. Pustka, P. Keitler, and G. Klinker. Splitting the scene graph - using spatial relationship graphs instead of scene graphs in augmented reality. In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications*, pp. 456–459. SciTePress - Science and and Technology Publications, 2008. doi: 10.5220/0001099804560459

[6] Enox Software. Opencv for unity.

[7] Espressif Inc.

[8] B. Huang, Y. Tang, S. Ozdemir, and H. Ling. A fast and flexible projector-camera calibration system. *IEEE Transactions on Automation Science and Engineering*, 18(3):1049–1063, 2021. doi: 10.1109/TASE.2020.2994223

[9] Intel Corporation. Intel realsense lidar camera l515.

[10] Intel Corporation. Intel realsense: rs-enumerate-devices tool.

[11] D. Kim, M. Ryu, and S. Lee. Antipodal gray codes for structured light. In *2008 IEEE International Conference on Robotics and Automation*, pp. 3016–3021. IEEE, 2008. doi: 10.1109/ROBOT.2008.4543668

[12] M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, M. Inami, and H. Nii. Augmented coliseum: An augmented game environment with small vehicles. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP '06)*, pp. 3–8. IEEE, 2006. doi: 10.1109/TABLETOP.2006.3

[13] M. Kuwahara and N. Umezu. Learning environment based on an interactive projection table for children. In *2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 109–113. IEEE, 2021. doi: 10.1109/AIVR52153.2021.00026

[14] J. Leitner, M. Haller, K. Yun, W. Woo, M. Sugimoto, M. Inami, A. D. Cheok, and H. D. Been-Lirn. Physical interfaces for tabletop games. *Computers in Entertainment*, 7(4):1–21, 2009. doi: 10.1145/1658866.1658880

[15] morikatron Inc. and Sony Interactive Entertainment Inc. toio sdk for unity.

[16] MVTec Software GmbH. gen_structured_light_pattern (operator).

[17] D. Pustka, M. Huber, M. Bauer, and G. Klinker. Spatial relationship patterns: elements of reusable tracking and calibration systems. In *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 88–97. IEEE, 2006. doi: 10.1109/ISMAR.2006.297799

[18] R. Raskar, H. Nii, B. deDecker, Y. Hashimoto, J. Summet, D. Moore, Y. Zhao, J. Westhues, P. Dietz, J. Barnwell, S. Nayar, M. Inami, P. Bekaert, M. Noland, V. Branzoi, and E. Bruns. Prakash. *ACM Transactions on Graphics*, 26(3):36, 2007. doi: 10.1145/1276377.1276422

[19] D. Robert, R. Wistorrt, J. Gray, and C. Breazeal. Exploring mixed reality robot gaming. In M. D. Gross, N. J. Nunes, E. Y.-L. Do, S. Brewster, and I. Oakley, eds., *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, pp. 125–128. ACM, New York, NY, USA, 2011. doi: 10.1145/1935701.1935726

[20] Sony Interactive Entertainment Inc. toio.

[21] Sony Interactive Entertainment Inc. toio core cube specifications: Configuration: Identification sensor id notification settings.

[22] Sony Interactive Entertainment Inc. toio core cube specifications: Identification sensor.

[23] Sony Interactive Entertainment Inc. toio core cube specifications: Shape and size.

[24] G. Taubin, D. Moreno, and D. Lanman. 3d scanning for personal 3d printing: Chapter 5 - structured lighting. In Unknown, ed., *ACM SIGGRAPH 2014 Studio on - SIGGRAPH '14*, p. 39 ff. ACM Press, New York, New York, USA, 2014. doi: 10.1145/2619195.2656314

[25] teamLab Inc. A table where little people live, 2013.

[26] Unity Technologies. Unity.

[27] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. doi: 10.1109/34.888718