Real-Time Game State Detection on a semi-automatic Foosball Table

David Hagens

Supervisors: Prof. Dr. Elke Hergenröther, Prof. Dr. Andreas Weinmann
Darmstadt University of Applied Sciences

Introduction

The automation of the game of Foosball provides a playground for deep reinforcement learning by training an AI agent to control one team of the Foosball table. De Blasi et al. [1] aimed to train an AI agent for the automation inside a simulation and reported a sim-to-real gap. This gap can include the absence of physical constraints in the simulation but also the absence of data in the real world. While the motors, which control the automated team, report their position and rotation, the human-played team does not provide this information. In contrast, this data is available in the simulation. Therefore, the AI agent can learn strategies based on the opponent's movement which is not applicable in the real world.

Figure 1. The semi-automated Foosball table.

The game state of a Foosball table is generally defined as (1) the position and rotation of each individual figures and (2) the position of the ball. While the velocity and direction of the ball is another important part, it can be calculated through multiple consecutive ball positions. The ball is already detected using traditional CV techniques by De Blasi et al. [1]. In contrast, the state of the figures is not detected. In this work, a concept for a game state detection system is presented which is able to detect the position and rotation of both teams of the Foosball table.

Concept

In a previous work by Horst et al. [2], a game state detection system for the white, human-played figures was presented. The system was able to detect all position by using the object detection network YOLOX. Additionally, the rotation detection was implemented for the white midfield rod using a modified ResNet18 image regression model. While this approach produced good results on the white figures, an adaptation to the black, automated figures was not possible. Therefore, a new end-to-end regression model to predict the position and rotation of each individual rod.

First, a training dataset is created containing the position and rotation of the rods. The black rods provide this information without further processing. In contrast, the data of the white rods need to be measured. The rotation is measured using Accelerometer sensors, namely the MPU6050 accelerometer by TDK InvenSense on a GY-521 breakout board. The position is calculated based on image data by using *a priori* knowledge of the rod positions and thresholding on a 1 px wide column. The algorithm was verified by calculating and comparing the positions of the black rods. Since the motors report the final positions even if the movement is not completed, the calculated positions are more accurate than the reported ones. Using this dataset, an end-to-end image regression model is trained on each rod. Five different feature extractor backbones were implemented and evaluated. The models were based on ResNet18, ResNet50, EfficientNetV2, MobileNetV3 and a custom architecture.

At the end, the predicted game state is distributed using a ZeroMQ publish and subscribe system which enables the usage of the data by further systems with minimal effort. As ZeroMQ is based on TCP, the whole detection system could also be outsourced to a permanently installed server to further improve the ease-of-use.

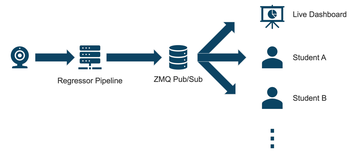

Figure 2. Overview of the game state detection process.

Results

The different feature extractor backbones were evaluated considering (1) the accuracy of the prediction and (2) the real-time objective which is defined as 60 fps. The desirable rotation error must be inside ±42 degrees since a figure would be able to block the ball inside this range. In contrast, the position needs to be more accurate with ±11 mm as the figure would be able to make a straight shot inside this range.

	ResNet18	ResNet50	MobileNetV3	EfficientNetV2	Custom
Mean Position Error (mm)	3.88	3.91	7.38	6.6	7.17
Mean Rotation Error (degrees)	5.93	4.1	10.18	11.79	14.71

Table 1. Mean position and rotation error of the different models.

As seen in Table 1, all models achieved the desired ranges for the position and rotation prediction. It is noted that the prediction of the rotation of the white rods is clearly worse than the prediction on the black rods. While the rotation is accurately measured by the motors on the black rods, the measurement of the white rods using the accelerometers introduces an error in the training data of around ±5 degrees on average. Additionally, the white figures can rotate freely around the full 360 degrees while the black rods are capped between 120 and 240 degrees. Overall, the ResNet-based models were the most accurate.

	ResNet18	ResNet50	MobileNetV3	EfficientNetV2	Custom
Mean overall Inference Time (ms)	88.88	120.83	117.00	249.31	71.58
Mean Inference Time per Rod (ms)	11.11	15.10	14.62	31.16	8.94
Mean overall FPS	11.25	8.28	8.55	4.01	14.01

Table 2. Mean inference time of the different models.

For the real-time objective, the inference times of the different models were measured on an AMD Ryzen 9 5900X, NVIDIA RTX 3080 with CUDA acceleration and 64 GB RAM. To achieve the desired 60 fps, the maximum inference time would be around 16.6 ms. The results are presented in Table 2. On average, the inference time of the ResNet18-based model was 88.88 ms or 11.25 fps. Since each rod is predicted sequentially, the inference time per rod is on average 11.11 ms which would result in > 60 fps. The slowest network was the EfficientNetV2-based model with an average inference time of 249.31 ms overall or 31.16 ms per rod resulting in 4.01 fps. The own architecture had the fastest inference times with 14 fps on average. Overall, the real-time requirement could not be met by any of the evaluated feature extractor backbones.

Considering the accuracy and the inference time, the ResNet18-based model generated the most promising results. The slightly better prediction quality of the ResNet50-based model especially in the rotation prediction is not enough to justify the higher inference time of an additional 33.14 ms.

Limitations

The presented system is subject to the following limitations:

- The concept only works on a Foosball table with an identical setup, i.e. the figures must be black and white and include rubber stoppers at the end of the rods;
- The quality of the detection is heavily dependent to the image quality of the camera;
- The rotation of the black rods is limited to the range between 120° and 240° due to the hardware limitation on the physical table; and
- The system does not include the ball into the game state.

If the system should be used on a Foosball table with different colored figures, a new dataset needs to be created and the models need to be retrained with the new dataset. The color of the figures should not directly influence the calculation of the position as long as rubber stoppers are installed at the end of the rods and the rods itself are not colored or metallic respectively.

The image quality of the camera and especially the shutter speed shows a strong influence on the detection quality. With a long shutter speed, the images will contain motion blur when an image is captured while the figures are in motion. Additionally, the used webcam does not allow manual focus or a lock of the autofocus resulting in overall blurred images. In the future, the webcam should probably be replaced by a professional camera with manual focus and shutter speed. This would result in better image quality while probably introducing the need for external lighting.

The limitation of the black rods rotation can be avoided by creating a training dataset on a manual Foosball table where the rotation in not limited by hardware. In the dataset creation, the rotation can be also measured by using accelerometers.

Conclusion and Future Work

In this work, a system was designed to capture the game state of a semi-automated Foosball game. The white figures of the Foosball table are played manually by humans and the black figures are automated using industrial motors. A training dataset was created by utilizing accelerometers to measure the rotation of the white rods and traditional CV techniques for the calculation of the position. The black figures report these data through the controlling motors. Using this dataset, an end-to-end regression model based on multiple architectures was trained and evaluated. The system outputs the predicted game state using a ZeroMQ publish and subscribe server which enables multiple clients on the same game state detector pipeline.

In the future, the presented limitations should be addressed. Especially the inference time of the models needs to be improved as the system is currently not able to achieve a real-time detection. Additionally, an improvement of the camera hardware would result in a better prediction as blurring occurs occasionally. Furthermore, the installation of permanent sensor hardware for the game state detection could be researched.

Overall, the detected game state achieves good results which contribute to DRL approaches by reducing the sim-to-real gap between a simulated learning environment and the real world. Additionally, the system can be used for imitation learning by capturing a manual Foosball game between human opponents. A DRL system could use the detected game state to learn human strategies in a Foosball game.

References

[1] Stefano De Blasi, Sebastian Klöser, Arne Müller, Robin Reuben, Fabian Sturm, and Timo Zerrer. Kicker: An industrial drive and control foosball system automated with deep reinforcement learning. *Journal of Intelligent & Robotic Systems*, 102(1):20, 2021.

[2] Ronny Horst, David Hagens, Elke Hergenröther, and Andreas Weinmann. Real time state detection of a foosball game using cnn-based computer vision. *SAI Computing Conference, London (accepted)*, 2024.