Automatic Goalie: Ping Pong Ball Trajectory Prediction with a Raspberry Pi

Gourishankar Mahadeo Bansode
School of Computing and Augmented Intelligence
Arizona State University
Tempe, Arizona, USA
gourishankar@asu.edu

***Abstract*—This report describes the *Automatic Goalie* project from CSE 598: Advances in Robot Learning. The goal was to build a small real-time system that sees a ping pong ball, predicts where it will land, and turns a servo-driven blocker toward that point. The system used a Raspberry Pi camera, a custom YOLOv8 detector, simple coordinate calibration, quadratic trajectory fitting, and MQTT-based motor control. The prototype ran quickly on lightweight hardware and gave strong predictions on several test throws, but it also showed clear limits when the data became noisy or the timing changed. This portfolio version uses simple language and includes a contribution statement, what was learned, and a short reference list.**

I. Introduction

Table-tennis robots are a good testbed for robot learning because they need fast perception, prediction, and control at the same time [1]–[3]. This project focused on a smaller version of that problem. Instead of building a full robot player, the team built a compact "automatic goalie" that watches a ping pong ball inside a small test box and turns a blocker toward the predicted landing point.

The main idea was simple: detect the ball in each frame, estimate where it is in the workspace, predict where it will go next, and move the servo before the ball arrives. Even though the idea sounds straightforward, the project became a strong systems exercise. The team had to combine computer vision, calibration, motion prediction, and hardware control into one pipeline that could run in real time on a Raspberry Pi. That is what made the project valuable for the portfolio: it was not only about one model, but about building a complete working robotics loop.

II. How the System Works

The setup used a Raspberry Pi 5, a camera running at 480p and 60 FPS, and a servo motor mounted at the front of the play area. The physical workspace was about 40 cm by 23 cm. The camera observed the ball from above, and the servo rotated a simple goalie arm to cover the expected landing region. Figure 1 shows the physical prototype used for testing.

To detect the ball, the team trained a custom YOLOv8 model using more than 1,000 labeled images collected from the project setup. The model was trained for 50 epochs and used at a confidence threshold of 0.5. In the original experiments, the detector reached a reported mAP of 0.92 and gave fast inference on resized frames. This gave the system a practical way to locate the ball in each video frame while still keeping the pipeline lightweight enough for real-time use.

Fig. 1. Physical prototype used for testing. The setup includes the top-mounted sensing hardware, the small play area, and the rotating goalie arm.

After detection, the system converted the ball position from image coordinates into approximate real-world coordinates. The center of the bounding box was used for the 2D location, and the ball's apparent size was used as a depth cue. This was not as accurate as a stereo camera setup, but it was enough for a first working prototype. Related work in robot table tennis shows that reliable ball tracking and trajectory estimation are central to good performance, even when the sensing setup is much more advanced than this one [1], [3], [4].

Once the system had a short history of ball positions, it smoothed the data and fit a quadratic curve to predict the flight path. The model then estimated where the ball would land and converted that point into a servo angle. Finally, the angle was sent over MQTT so the servo could rotate toward the target location. In simple terms, the full loop was: *see the ball, estimate the path, predict the landing point, and move the blocker*. That pipeline was the main technical contribution of the project.

To make the prediction more stable, the system kept a short motion buffer of recent in-flight points and applied simple smoothing before fitting the curve. This helped reduce frame-to-frame jitter from the camera without adding much delay. The team chose a quadratic model because it was lightweight,

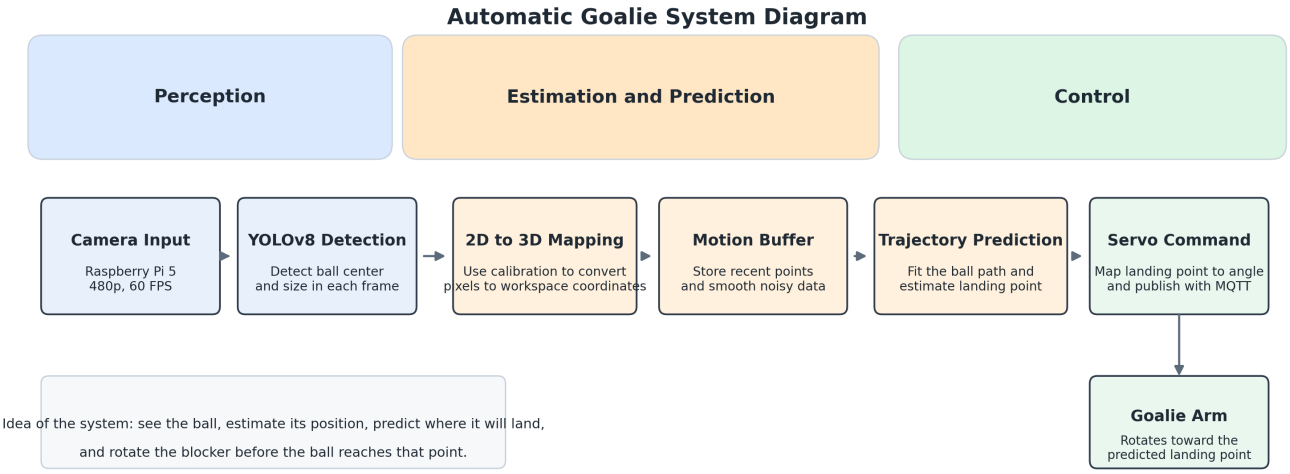

Fig. 2. System diagram for the Automatic Goalie. The pipeline moves from camera input to ball detection, position estimation, trajectory prediction, and servo control.

TABLE I
REPRESENTATIVE LANDING PREDICTIONS FROM TESTING

Target	Predicted	Comment
$(0, 20)$	$(0.0, 20.0)$	Exact
$(40, 10)$	$(40.0, 10.0)$	Exact
$(20, 18)$	$(20.5, 20.0)$	Close
$(35, 5)$	$(32.6, 16.1)$	Y error
$(30, 6)$	$(40.0, 20.0)$	X/Y error

easy to debug, and practical for a Raspberry Pi. A more complex prediction model may improve accuracy later, but this version emphasized a method that could run fast and remain understandable during testing.

After the landing point was predicted, the system mapped that location to a servo angle and sent the command through MQTT. This made it easier to separate the prediction code from the motor controller while still keeping the loop responsive. The team also logged throws and prediction outputs during testing. That record was useful for later analysis because it showed whether an error came from vision, calibration, timing, or servo motion.

III. RESULTS

The prototype produced several strong results. For example, a target at $(0, 20)$ cm was predicted as $(0.0, 20.0)$ cm, and a target at $(40, 10)$ cm was predicted as $(40.0, 10.0)$ cm. A middle target at $(20, 18)$ cm was predicted as $(20.5, 20.0)$ cm, which was still close enough to show that the model was capturing the general motion correctly. The detector was also fast, with inference around 2.7 ms per frame, and the servo usually moved to within about 5–10 degrees of a good blocking angle.

At the same time, the system showed clear failure cases. Low landing points on the y-axis were often overestimated. For example, one target at $(35, 5)$ cm was predicted as $(32.6, 16.1)$ cm. The system also showed x-axis errors when the extrapolation became too aggressive, such as a target at $(30, 6)$ cm being predicted as $(40.0, 20.0)$ cm. These mistakes usually happened when the ball path was noisy, when the bounce was hard to detect, or when the real frame timing differed from the fixed 60 FPS assumption. Overall, the project achieved its main goal: it proved that a small edge device could run a complete perception-to-control loop, but it also showed where the model still needs improvement.

The results also showed why full-system testing matters in robotics. A detector can perform well on individual frames and still fail to block the ball if timing, calibration, or servo motion is off. In this project, prediction accuracy depended on the whole pipeline working together: camera capture, coordinate conversion, motion fitting, and motor response. That made the project a useful example of real robot learning work, where performance depends on integration as much as on the model itself.

The tests also highlighted how difficult single-camera depth estimation can be. Because the system used the apparent size of the ball as a depth cue, small changes in detection could create larger errors in the final 3D estimate. That explains why some throws looked visually close to the true landing point but still produced noticeable coordinate errors. Even with that limitation, the system responded fast enough to demonstrate a real closed-loop robotic behavior, which was the most important success of the prototype.

During testing, the team compared behavior across different throws and landing zones inside the small workspace. The best cases happened when the ball stayed visible for enough frames and followed a clean arc. The harder cases appeared when the motion was noisy or the bounce changed the path quickly. Looking at these patterns helped the team understand not only whether the system worked, but also when it worked best and why it failed.

IV. Future Improvements

If the project continues, the first change would be better state estimation. Bounce detection would also be improved, and the system would be tested under more difficult lighting and background conditions. A second upgrade would be better depth estimation. Because this version used a single camera and the ball size as a depth cue, the 3D estimate was only approximate.

Several focused changes would likely improve the next version. A Kalman filter or another state estimator could smooth noisy measurements better than the current moving average. Dynamic frame-rate handling would also help because the current model assumes a fixed 60 FPS even though the real timing can change. Another useful upgrade would be a second camera or a more careful calibration procedure. That would make depth estimation stronger and could reduce the y-axis overshoot seen in testing.

V. Contribution and What Was Learned

This was a group project completed with Gourishankar Bansode and Rushi Ranpise. Gourishankar Bansode led the design of the full pipeline and hardware setup for the project. Rushi Ranpise supported data creation for ping pong ball trajectory detection by helping collect and label the dataset. Gourishankar Bansode designed the YOLOv8 training pipeline, calibrated the camera, wrote the trajectory prediction code, and later worked with the team to integrate MQTT with the servo and evaluate the final system performance during testing.

One part of this project that was especially useful was systems integration. It was not enough to have a detector that worked on saved images. The project required the camera, prediction code, and servo timing to work together in a live loop. That meant checking logs, studying bad predictions, and tracing whether an error came from detection, calibration, or timing. This process improved debugging skills and encouraged a more careful approach to evaluating a robotics system.

This project taught several useful skills. First, it showed how computer vision and hardware can be connected in one real-time robotics pipeline. Second, it provided practical experience with model training, camera calibration, and motion prediction.

The project also highlighted an important design tradeoff. A more complex learning-based predictor might improve accuracy, but the team chose a simpler method that was fast enough for the Raspberry Pi and easier to inspect during debugging. That choice showed that a practical robotics system is not always built around the most advanced model. In many cases, a solution that is fast, understandable, and reliable is more useful than a heavier model that is harder to deploy on real hardware.

This broader systems view was one of the most important lessons from the course. In class, robot learning can look like a model or algorithm problem, but this project showed that sensing, calibration, timing, communication, and control all affect the final result. That perspective will be useful in future projects that move from offline experiments to live autonomous systems.

The project also connected course concepts to engineering decisions. Instead of evaluating only model accuracy, the team had to think about the full path from sensing to action. That meant checking whether camera settings, coordinate mapping, and servo timing were all consistent with the prediction code. Learning to look at the full system, not just one algorithm, was one of the clearest outcomes of the project.

From a portfolio perspective, this project is useful because it includes the complete workflow of a robotics system. The work involved collecting and labeling data, training a detector, calibrating the camera, writing the prediction logic, connecting the controller, and testing the final behavior on hardware. That combination makes the project a strong example of applied robot learning rather than only an offline computer vision

VI. CONCLUSION

In this project, the team built a small automatic goalie that detects a ping pong ball, predicts where it will land, and turns a servo arm toward that position. The system combined vision, simple physics-based prediction, and hardware control in a compact prototype. The final results were promising enough to show that the idea works, and the failure cases gave the team a clear path for improvement. For the portfolio, this project shows both technical implementation and lessons learned from building a full real-time robotics system.

Another important outcome of the project was the team's clearer understanding of what it means to move from an offline model to a working robot. The detector, calibration method, trajectory fitting, and servo control all had to work together under real timing limits, and a weakness in any one of those parts affected the final block. That experience showed that robot learning is not only about training a better model; it is also about making sensing, prediction, and control reliable enough for real hardware. The team also learned to reason about latency, noisy measurements, and mechanical response as part of one connected system rather than as separate problems. This gives the project lasting value beyond the prototype itself. It provides a solid base for future work with better cameras, stronger state estimation, and more advanced learning methods. For the portfolio, it demonstrates design, integration, debugging, and evaluation in one end-to-end robotics system instead of only isolated model training.

REFERENCES

[1] H.-I. Lin, Z. Yu, and Y.-C. Huang, "Ball Tracking and Trajectory Prediction for Table-Tennis Robots," *Sensors*, vol. 20, no. 2, p. 333, 2020, doi: 10.3390/s20020333.

[2] Y. Huang, B. Scholkopf, and J. Peters, "Learning Optimal Striking Points for a Ping-Pong Playing Robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4587-4592, doi: 10.1109/IROS.2015.7354030.

[3] S. Gomez-Gonzalez, Y. Nemmour, B. Scholkopf, and J. Peters, "Reliable Real-Time Ball Tracking for Robot Table Tennis," *Robotics*, vol. 8, no. 4, p. 90, 2019, doi: 10.3390/robotics8040090.

[4] Y. Zhang, Y. Zhao, R. Xiong, Y. Wang, J. Wang, and J. Chu, "Spin Observation and Trajectory Prediction of a Ping-Pong Ball," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4108-4114, doi: 10.1109/ICRA.2014.6907456.