AISE3350A: Cyber-Physical Systems Theory

Final Project

December 5, 2025

Sarthak Bali – 251393953

Alex Hazen – 251367477

Jon Das – 251372549

Jan Wodnicki - 251372702

Introduction

Cyber-physical systems (CPS) consist of three entities, sensing, computing, and actuating working together to enable real-time processing and response. This project integrates computer vision and corresponding game theory to develop a CPS suited to play Rock-Paper-Scissors-Minus-One (RPS-1).

The more commonly known variant Rock-Paper-Scissors (RPS) offers a simple pairwise competition between two players. The "Minus-One" (RPS-1) variant introduces complexity to this decision space by introducing one additional hand per player. Players must present both of their hands and then remove one, with the remaining pair being the decider. This addition makes the game a two-stage, simultaneous, zero-sum game.

The game of RPS-1 was chosen as it requires the project to apply all pillars of CPS. Sensing using Computer Vision (CV) and Machine Learning (ML) to detect the player's actions. Computation to identify gestures and compute the game theory principles and payoff matrices. Lastly, a decision recommending what the player should do to maximize their chances of winning.

Methods

The objective of the CPS is (1) to identify the hand shapes being presented by each player for stage one and (2) determine the optimal action to win the game through a game theoretical analysis. The workflow that was chosen to achieve this objective has three major steps, hand detection and classification, enumeration of all possible outcomes, and strategy selection. The system begins with detecting four hands and classifying them into rock, paper, or scissors. Then from the detected shapes, all possible hand-combinations are computed, and a payoff analysis is applied to recommend which hand should remain out for stage two. This method was chosen to ensure that the perception and decision-making components can operate independently and allow us to iteratively troubleshoot and improve them.

Strategic Framework and Loss Minimization Logic

First, the RPS-1 strategic framework was developed to formally represent decision making in the restricted RPS environment allowing players to have a subset of actions rather than a single move. This requires a structured representation of the outcomes of the game implemented using matrices. The first is a decision matrix that encodes the action pairs selected by each player, the second is an outcome matrix that captures the win-loss-draw payoff for player one according to the standard RPS rules. The two-matrix setup allows the program to easily map any pair of selected moves to a qualitative outcome and a quantitative score.

To account for the multi-choice sets that will be featured in the RPS-1 game the system computes cartesian products for both players and evaluates the resulting combinations. This enumerates all feasible interactions and creates statistics for expected wins, losses, and draws for player one. This mapping, although exhaustive, forms the basis for strategic analysis, enabling the overall risk profile to be quantified and associated with any declared set of moves.

To implement loss minimization the combinations of inputs from player one that yield negative outcomes are isolated. First the analysis identifies if any losing scenarios exist at all, if no such scenarios exist the model concludes the input combination to be a non-losing outcome. If a losing scenario exists, it is categorized based on which move by player one produced it. In the case of a single move producing the losing outcome, the system will recommend withdrawing that hand for stage two. When multiple loss producing moves are presented, the system applies a framework with a hierarchy of criteria to evaluate if a losing move can yield a draw scenario, if the losing moves are identical, and how many times that specific move has contributed to a loss. Using this ranked framework the system can rank-order the options player one has based on the risk contribution of the given moves.

The model was further refined with an extended valuation system based on strategy profiles. The profiles, "aggressive", "conservative", "balanced", and "unweighted" introduce a weighing factor that modifies the scoring applied on outcomes based on the likelihood of selections made by player two. Using this model on actions made by player two allows for adjustments of valuations and generation of a performance metric for each of the moves made by player 1. The derived score allows player 1 to find the move that is the most detrimental relative to the profile of player 2 and should therefore be removed to minimize losses.

Hand Detection and Gesture Classification

For the perception aspect of our CPS three different computer-vision and machine learning approaches were evaluated. The first approach used a MediaPipe landmark-based, rule driven classifier. In this approach the model uses 21 anatomically landmarked points to determine what move is being displayed by the players. This method is lightweight, easy to reproduce, and does not require any training, making it optimal for early development and real-time demonstration.

The second approach was a pretrained neural network designed by MediaPipe for general purpose gesture classification. This approach was abandoned due to its poor performance in real-world lighting and camera conditions. It displayed significant struggles with the scissors move due to the similarity to other open-hand gestures.

In the third approach a YOLOv11 object detection model was trained specifically for the three moves in the RPS decision space. This specific model was selected due to its fast, real-time multi object detection features which ensured that it can be fine-tuned on small, domain specific datasets. These capabilities are essential in the RPS-1 CPS as up to four hands need to be identified in a single frame.

To optimize this model for the RPS-1 use case various pre-processing tasks were applied, starting with building a reproducible dataset pipeline. Through imageannotation.py, YOLO-formatted labels are generated for raw images, and a pretrained YOLO model is applied to create boxes for each gesture. The program organizes and labels the images into their respective directories while creating a YAML file defining class indices. With this initial structure in place, using the configuration a YOLO11 backbone is fine-tuned for 80 epochs using standard augmentation.

Player Segmentation Logic

A player segmentation step is essential for the system to be able to correctly assign the detected hands to the correct player. This step operates differently depending on which of the detection pipelines are being used. The YOLO based system uses spatial partitioning of the camera frame itself, with hands in the left half of the frame being attributed to player one, and right half to player two. This method is simple but requires a fixed camera position and gameplay orientation but reduces variation which allows the model to pick up gestures more effectively.

System Integration

For the system to come together and act as a function CPS the outputs of the computer-vision subsystem needed to be connected to the decision engine in a matter consistent with CPS design principles. After detection through the YOLO classification, the generated symbolic labels are passed into evaluation functions defined in RPSLogic.py. All possible combinations of stage two results are put through the loss minimization algorithm, and the resulting recommendation is output to the video feed using OpenCV to provide real time feedback.

The subsystems for hand detection, hand classification, player segmentation, and strategic evaluation were implemented into two separate files to better organize the code. The background file is based on the logic behind the RPS-1 game and is organized in such a way that it can be called with player inputs and a strategy style, then return a recommended action, or state you have won by default. This function can be used outside of our current model and was separated so that it can be directly called by the user, via command line with the inputs as another option. The second file consisted of hand detection, hand classification, and player segmentation. This file splits the camera feed into quadrants based on the section of the game (four at the start, then two for the final) and then uses the trained YOLO model to detect hands and classify them within each of these quadrants. These classified hands are separated into being player one or player two based on the quadrants they were in, and then the logic function is called to compute what hand player 1 should drop, or the winner.

This separate design allows for easier adjustment and implementation of parts of our code for other purposes in the future, maintaining a good level of reproducibility and maintainability requirements for the course.

Results

Refer to Jupyter Notebook file.

Discussion

In our implementation, we have built upon YOLO11, as it proved the most robust at identifying and labelling four hands inside one frame. To make the YOLO11 model work for our application, we needed to train it using numerous hand gestures consisting of three classes: Rock, Paper, and Scissors. Due to the similarity in datasets found on Kaggle, three datasets were combined with the hope of reducing the chance of overfitting our model. To accommodate any differences in the image types across datasets, imageannotation.py handles image formats of

JPG, JPEG and PNG. Furthermore, imageannotation.py uses a pretrained YOLO11 model to identify the point of interest in each image (the hand) and draw a bounding box around it; the bounding boxes help focus on the custom model so that it knows the hand is the object of interest. After each image has been annotated with a bounding box, datasplit.py splits the data into 80% training and 20% validation sets; during the data split the images are randomly shuffled and then placed into new folders. The YOLO11 model requires a specific file type (.yaml) to initialize training, so once all the data has been split and reallocated datasplit.py creates a .yaml file to be used in rps_model.py.

Figure 1: Depicts the F1 curve which demonstrates the model's robustness in classification of rock, scissors or paper gestures. At a low confidence threshold, our model would struggle with identifying the true gesture; however, it quickly gained precision in the 0.95 range, from confidence interval of 0.2 to 0.8 which showed robust performance. From the graph above, it appears that our model runs optimally at around the 0.4 confidence interval range where it has an F1 score of approximately 0.98. The performance of the rock gesture is the lowest of the three gestures, however given its still relatively high performance it did not seem to affect our RPS-1 game.

Figure 2: Depicts the Precision-Confidence curve which demonstrates the RPS model's precision given a specified confidence level. At confidence level of 0.2, our model has an approximate precision of 0.93 which is great as it demonstrates that our model develops a very high precision of detecting the true gesture from rock, scissors and paper early on. From a confidence interval of 0.3 to 0.9, the model shows great robustness as it is the 0.95+ range for precision indicating that it rarely produces any false negatives. Overall, this curve indicates the robustness of our implemented model.

Figure 3: Depicts the Precision-Recall curve which demonstrates the RPS models high precision across the full range of recall values with the mean average precision (mAP) of 0.984 at 0.5 IoU. Overall paper and scissors gestures performed best with an average precision (AP) of 0.993 and 0.991 respectively, which are near perfect, showing great robustness. The rock gesture underperformed relative to paper and scissors gestures; however, it still had an AP of 0.967 which demonstrates strong detectability. The curve remained close to the top-right region, displaying excellent balance between precision and recall, indicating a few false positives and false negatives displaying great robustness.

Figure 4 and 5: Depict the raw and normalized Confusion Matrices. Both demonstrate how our YOLO11 model performs exceptionally well across the three gesture classes. The raw confusion matrix describes exactly how many errors were found inside the three gesture classes, showing relatively few errors for paper and scissors as they were between 10 and 15 respectively. The rock gesture once again showed the worst performance as it misidentified the rock gesture 40 times. The normalized matrix displays these errors with the relative error to the rest of the dataset, which identifies the performance of the paper is perfect with 1.0 and scissors being almost perfect with 0.99 scores. It also displays the rock score of 0.97 which shows its strong performance, though weaker than the prior gestures. Overall, the confusion matrices further enforced the robustness of our RPS model using the YOLO11 implementation.

Although this project appears at first to be a playful gesture-recognition game, its underlying technologies—computer vision, gesture recognition, real-time decision systems, and human–machine interaction—have far broader implications beyond the context of Rock–Paper–Scissors. The Rock-Paper-Scissors-1 game project exemplifies how gesture recognition technology can influence societal and economic norms, reshaping conventional thinking about human-computer interaction. At its core, this project highlights the promise of gesture-based interfaces, enabling users to engage with technology through simple hand movements, thereby increasing accessibility for individuals with limited mobility and enhancing hands-free operations in environments requiring sterility, such as surgical settings. Furthermore, integrating machine learning with game theory not only helps demonstrate AI decision-making processes but also promotes a deeper public understanding of artificial intelligence. By leveraging low-cost computer vision solutions, such as webcams and open-source frameworks, the RPS-1 project demonstrates that high-accuracy computer vision can be achieved without expensive hardware, fostering innovation and reducing barriers for startups across industries, including retail automation and robotics.

Economically, the automation of tasks traditionally reliant on physical input devices signals a shift in job dynamics, emphasizing the need for workforce adaptation toward roles in AI supervision and maintenance. The surge in market demand for touchless interactions has opened new avenues for growth across sectors such as healthcare, education, and retail, where gesture recognition can enhance the user experience. However, it is essential to acknowledge the ethical implications inherent in this technology, including privacy concerns related to data capture and the risk of algorithmic bias stemming from non-diverse gesture datasets. Addressing these challenges will be critical to ensuring a fair and responsible implementation that aligns with societal values and broader impacts for future projects that implement computer vision.

This project demonstrates the integration of computer vision, machine learning, and game-theoretic analysis into a unified, real-time decision-making system. The design of our system ensures it is maintainable, reproducible, and flexible for future enhancements. A refined YOLO11 model and a systematic loss-reduction strategy engine facilitate reliable gesture identification and strategy suggestions. Apart from this game, this project demonstrates the capability of CPS interfaces to improve accessibility, facilitate touch-free interaction, and assist intelligent decision-making systems. In the end, the influence of principled computational design and affordable sensing in resilient, real-time CPS solutions in significant applications extends well past entertainment.