Black-Jack with Robotics

Voice-Controlled and Automated Card Dealing Using the Niryo-Robot

Hochschule Hof
Applied Robotics
winter semester 2024/25

1st Dominick Schrenk
University of Applied Sciences Hof
Hof, Germany
Email: dominick.schrenk@hof-university.de

2nd Peter Pham
University of Applied Sciences Hof
Hof, Germany
Email: peter.pham@hof-university.de

3rd Alexandra Sumera
University of Applied Sciences Hof
Hof, Germany
Email: alexandra.sumera@hof-university.de

4th Samuel Schweigert
University of Applied Sciences Hof
Hof, Germany
Email: samuel.schweigert@hof-university.de

***Abstract*—This paper presents the development of an automated Blackjack game system using the Niryo Ned 2 robot. The project demonstrates the integration of robotic precision and natural language interaction to create a seamless and interactive gaming experience.**

***Index Terms*—Blackjack, robotic automation, Niryo robot, voice control, human-robot interaction**

I. Introduction

The project presented in this paper focuses on the development of an automated Blackjack game system using the Niryo Ned 2 robot. The goal of the project is to simulate the card-dealing process in a Blackjack game, with the robot autonomously handling the physical distribution of the cards and chips to three players and a dealer. Additionally, the system incorporates voice-controlled commands, enabling players to issue commands such as “Hit” or “Stand”. By combining robotics with voice recognition and game mechanics, this project bridges the gap between physical automation and interactive control systems.

The relevance of this project lies in its practical application of robotic automation and its potential to demonstrate broader concepts in human-robot interaction. While the project is centered around a recreational activity, its significance goes beyond just playing a card game. The automation of game mechanics, such as dealing cards or responding to player commands, opens doors to enhanced casino experiences through interactive robotic dealers or hybrid gaming systems. Such systems could reduce operational costs, minimize errors, and offer players unique and engaging experiences.

Furthermore, automating physical tasks like dealing cards reflects concepts used in industries such as logistics and manufacturing, where robots handle repetitive or precise tasks efficiently [2]. The integration of voice control also mirrors advancements in everyday technology, like smart home devices, which aim to make human-technology interaction more natural and intuitive. Projects like this highlight the potential of robotics to enhance not just efficiency but also user experience.

II. Starting Point

In this section, the outline of the initial setup and key components used to develop this project will be outlined. This includes a description of the hardware and software employed to bring the project to life.

A. Available Hardware

The main hardware component for this project was the Niryo Ned 2 robot, a six-axis collaborative robotic arm (fig. 1). Known for its precision and adaptability, the Niryo Ned 2 was ideal for automating the physical tasks involved in a Blackjack game, such as picking up cards and chips and placing them accurately. The Niryo Ned 2 utilizes its robot software version 5.7.0.

The robot’s gripper was tested and calibrated to ensure it could handle the delicate nature of placing cards and chips without misplacing them. However, a significant challenge in the hardware setup was the robot’s ability to pick up the extremely thin game cards. To address this issue, elastic bands were wrapped around the gripper’s fingers, increasing friction and ensuring a firm grip on the cards without causing damage. This simple yet effective modification allowed the robot to reliably pick up and place cards with precision.

Fig. 1. Niryo Ned 2 Roboter. Handles the card and chips dealing autonomously.

Additionally, a USB video camera was placed in front of the Niryo robot to enable the reading and identification of individual playing cards (fig. 2). The webcam provided the necessary visual input to the system, capturing the cards as they were dealt or moved. The captured images were processed using a machine learning model (described in Section III), which analyzed the images to identify the cards accurately. This component was crucial for the automation process, ensuring that the robot could interact with the cards based on their specific values and suits, thereby enhancing the realism and flow of the game.

Fig. 2. USB Video Camera. Scans the used game cards.

The workspace was customized with a predefined table layout, including designated areas for three players and the dealer, ensuring consistent card and chip placement throughout the game simulation.

B. Software

For the development and testing phases, *NiryoStudio*, the official software for interacting with the Niryo Ned 2, was used to simulate and control the robot's actions. This software provided an intuitive interface for defining movements and testing various configurations.

The programming and customization of the robot's behavior were conducted in *PyCharm*, which served as the primary environment for coding the robot's actions. Python-based scripts were seamlessly integrated with the robot's software stack using the *PyNiryo* library, enabling precise control of the Niryo Ned 2.

Voice recognition was implemented using the *Vosk* library, which featured a lightweight model suitable for the project. The *Sounddevice* library facilitated microphone input to ensure smooth communication between players and the robot.

For card recognition, a pre-trained object detection model from the GitHub repository *Playing Cards Object Detection* was employed [1]. This model was further enhanced using the *Ultralytics YOLO* framework, known for its high-speed and accurate image detection [5]. The visual input from the camera was processed and analyzed with *OpenCV* (CV2) to refine card detection accuracy [4]. Additional data management and mathematical operations required for gameplay were handled using *NumPy* [3].

III. Concept and Contribution

This section outlines the key actions undertaken to realize this project, highlighting the modifications and enhancements that contributed to the automation and gameplay experience.

A. Game Logic and Rules

The game logic was developed based on standard Blackjack rules, with slight modifications to simplify the gameplay. Each participant begins with two cards, while one of the dealer's cards remains face down. The objective is to achieve a card total as close to 21 as possible without exceeding it. Players can either draw an additional card (*Hit*) or end their turn (*Stand*). Advanced actions such as doubling down or splitting were excluded to maintain simplicity.

If a player achieves a total of 21 with an ace and a card valued at 10, they automatically win the round. The dealer follows predefined rules, drawing cards until reaching a minimum total of 17. The inclusion of betting chips was later added, allowing the player to win double their bet when successful.

B. Initial Setup and Automation

The initial setup featured one player and one dealer. To replicate the casino experience, a camera with card detection capabilities was integrated. This ensured that only the dealer (or computer) could see the hidden card, preserving the game's strategic elements.

C. Card Detection Process

The card detection system underwent several iterations. Initially, a custom-trained model using a small dataset provided functional but suboptimal results. Training with a larger

dataset from Roboflow improved detection accuracy but required significant resources. To streamline development, a pretrained model meeting the required performance standards was adopted. The camera was positioned adjacent to the card dispenser, and cards were held in front of it by the robot. Cards with a detection confidence exceeding 0.8 were registered in the system (fig. 3).

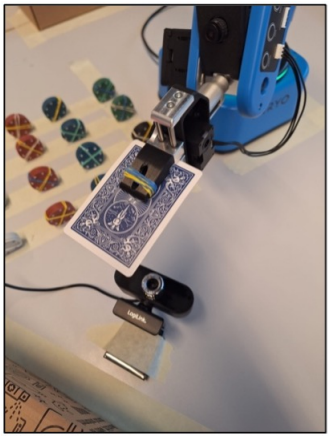

Fig. 3. Niryo robot holds a card over a camera in order for the camera to be scanned.

IV. Initial Setup and Configurations

For the initial setup, three components were required: the player, the dealer, and the card dispenser, where cards would be manually placed. Due to space constraints, both the player and the dealer can hold a maximum of four cards.

For the Niryo setup, several positions were required to ensure the game functioned properly. Initially, a default starting position was needed for stability. For the card dispenser, a position was necessary to allow it to pick up a card effectively. Each card position required two distinct configurations: one for placing the card and another slightly behind it to avoid disturbing adjacent cards (fig. 4). Finally, a position was designated for a celebratory gesture, with the arm pointing to the winner. However, this feature was removed in configurations involving multiple players.

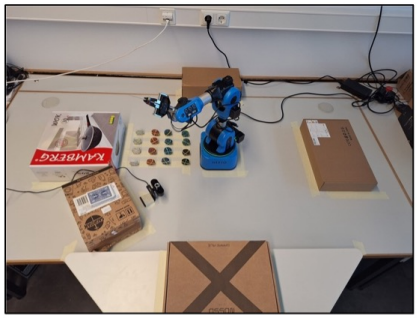

Fig. 4. Setup of the Niryo robot and its environment.

A. Extension - Voice Detection

As previously mentioned, our goal was to create an experience as realistic as being in a casino. To achieve this, we implemented a voice detection system, enabling players to say “hit” or “stand” instead of manually entering commands. Initially, we used the Google API, which provided acceptable results but had significant drawbacks. The detection process was slow, and the accuracy was insufficient for our needs.

We identified a better alternative: vosk, an offline speech recognition system. After testing both the larger model (vosk-model-en-us-0.22) and the smaller model (vosk-model-small-en-us-0.15), we concluded that the smaller model met our requirements effectively regarding accuracy and speed.

B. Extension - Multiple Players

To enhance the gameplay experience, we introduced the option to include multiple players at the table, simulating the environment of a real casino. At the start of the game, players can select the number of participants, with a minimum of one and a maximum of three players.

C. Extension - Betting Chips

Placing bets is a fundamental aspect of casino games, so we incorporated this feature into our game. Players can choose from the following chip values: 1, 2, 10, 25, and 100. If a player wins their hand, the dealer returns double the amount of the player’s original bet.

V. Challenges and Solutions

In this section, we will discuss the key challenges encountered during the development of this project and the solutions implemented to overcome them.

A. Challenge 1: Precise Arm Movements

One of the most persistent issues of this project was achieving the necessary precision in the arm movements to ensure that the cards remained securely on the card holder after being placed. The robot’s initial movement setup was not optimized for handling the delicate nature of the cards, resulting in frequent failures where the cards would slip from the gripper or fall off the holder due to improper placement. This was particularly problematic during the card distribution process, where the robot had to place the cards in a manner that prevented them from falling or being displaced during subsequent moves.

Moreover, the robot had to be able to position the cards precisely enough for them to be picked up again during later stages of the game, such as when a card needed to be flipped over for inspection or to be used in further gameplay. The arm movements had to be meticulously calibrated to ensure that the cards were placed at the right position, with enough space to avoid any overlap between the cards, since overlapping cards could create confusion for both the robot and the players,

making it difficult for them to distinguish between the cards and continue with the game flow.

To address this issue, a solution was developed that involved fine-tuning the robot's arm movements and adjusting the placement angles for greater precision. The robot's gripper was programmed to lower the cards gently and at specific angles to ensure they were placed on the holder without risk of falling. By breaking down the movements into smaller, more controlled steps, the robot could achieve a higher level of accuracy, minimizing errors in the card placement. This iterative approach significantly improved the stability of the card placement. On the downside, this approach resulted in an increase in complexity, requiring more lines of code to manage the multiple, finely tuned movements. This made the debugging process more difficult.

B. Challenge 2: Lighting Conditions and Card Recognition

Another key challenge was the robot's ability to correctly identify the playing cards through the machine learning-based recognition system. The card recognition system was highly sensitive to lighting conditions, and inconsistent or fluctuating light sources would often lead to misidentification of the card or to no identification at all. Shadows could distort the appearance of the cards, making it difficult for the system to distinguish between the cards and the background.

To address this issue, the lighting setup around the robot's workspace had to be carefully optimized. In addition, the model was trained with a variety of lighting conditions to make it more robust to changes in the environment. Through these measures, the robot achieved a higher accuracy in identifying the cards.

C. Challenge 3: Difficulty in Recognizing Multiple Cards Simultaneously

Another significant challenge arose in the card recognition process, particularly when multiple cards were placed close to each other. Initially, the plan was for the camera to evaluate the cards at the final stage of the game, after they had already been dealt to the players. However, the card recognition model struggled when trying to detect multiple cards in the same image, as it was overwhelmed by the placement of several cards at once. This difficulty in distinguishing and classifying the cards resulted in errors and increased, unacceptable latency, causing the robot to misidentify certain cards.

To overcome this challenge, the project team drew inspiration from real-world scenarios, such as online casinos, where cards are processed individually. Instead of attempting to analyze all the cards at once, the process was restructured so that the robot would scan and recognize one card at a time during the initial card dealing. By guiding each card to the camera separately, the system's accuracy was significantly improved, as each card was clearly isolated and identifiable.

D. Challenge 4: Calibration of the Robot's Gripper

Another challenge was ensuring that the Niryo Ned 2 robot could reliably pick up the thin cards used in the game. The robot's gripper was not able to fully close, which made it difficult to securely grasp the delicate cards without slipping them.

This problem was easily solved – elastic bands were wrapped around the gripper's "fingers" to increase friction and improve its ability to securely grip the thin cards (fig.5).

Fig. 5. Niryo robot with elastic bands around its gripper.

E. Challenge 5: Processing Delays and Real-Time Interaction

Another challenge was ensuring that the robot's actions, including card dealing, chip handling, and voice recognition, occurred in real-time, without significant delays that could disrupt the user experience. Latency in processing could cause awkward pauses or make the interaction feel unnatural, particularly in a fast-paced game like Blackjack.

To minimize processing delays, the system's architecture was optimized for real-time performance. The game logic was decoupled from the robot's control system to prevent slowdowns due to heavy computations.

F. Challenge 6: Voice Recognition Accuracy and Reliability

It should also be mentioned that challenges occurred during the development of the voice recognition system. Initially, the first voice recognition library used in the project – Google API – had difficulty accurately recognizing and interpreting the player's commands. This often resulted in unrecognizable or misinterpreted commands, which disrupted the flow of the game. For example, when a player used commands like "Hit" or "Stand," the system sometimes failed to recognize the input.

This issue was solved through the integration of a more accurate and suitable voice recognition library. This library provided better recognition of the voice commands needed.

G. Challenge 7: Handling Game Flow and Player Decisions

The `turn_order` function is crucial for managing the flow of the game, from placing bets to comparing points at the end. One of the biggest challenges was synchronizing the game flow and keeping track of each player's status in real time.

Another challenge was handling player interactions with the robot, particularly in inputting bets and decisions. Since the game progresses in rounds, with both players and the dealer acting based on the current game state, the robot had to respond accurately while following the game rules strictly (fig. 6).

One notable challenge with the `turn_order` function is making sure that the conditions and checks are applied at the right stages in the game. For example, it's crucial to ensure that players' points are checked after each round of drawing cards to see if they have exceeded the limit. Similarly, the dealer's status must be evaluated after they finish their turn. If these checks aren't placed at the correct points in the sequence, the flow of the game could be compromised, potentially allowing a player or dealer to continue with invalid points or affecting the overall game logic.

```
def turn_order(num_players):
    # Global variables: score_counter, bet_counter
    eliminated_players = [0, 0, 0]

    # 1. Place bets
    for i in range(num_players, 0, -1):
        bet_counter[i] = place_bet(i)

    # 2. Draw phase (2 rounds)
    for round in range(2):
        for i in range(num_players, 0, -1):
            new_cards = draw_cards(i)
            score_counter[i] += new_cards
            print(f"Player {i}'s points: {score_counter[i]}")

            # Check if the player is still in the game
            if score_counter[i] > 21:
                eliminated_players[i] = 1
                print(f"Player {i} is eliminated")

        # Dealer's turn
        dealer_cards = draw_cards('dealer')
        score_counter['dealer'] += dealer_cards
        print(f"Dealer's points: {score_counter['dealer']}")

        # Check if the dealer is still in the game
        if score_counter['dealer'] > 21:
            print("Dealer is eliminated")

    # 3. Player's decisions (Hold or Draw)
    for i in range(num_players, 0, -1):
        if eliminated_players[i] == 0:
            make_decision(i)

    # 4. Dealer finishes the game
    dealer_finish()

    # 5. Compare points to determine the winner
    for i in range(num_players, 0, -1):
        if eliminated_players[i] == 0:
            compare_points(i)

    print("Game over")
```

Fig. 6. Pseudo-code for handling the card drawing and bet placement phase in the game.

This example highlights the relevance of robotics in interactive environments, where robots must adapt to dynamic situations. Much like in logistics or manufacturing, robots in such contexts rely on real-time data to perform tasks. In this case, the robot must make decisions and follow set rules, reflecting the decision-making processes essential for real-world robotic systems. It underscores the significance of robots functioning autonomously while interacting seamlessly with humans — a core principle in numerous robotic applications.

VI. Evaluation

Overall, the project demonstrated the successful integration of robotics and software to create an interactive and automated card game simulation of Blackjack.

The Niryo Ned 2 robot excelled in automating tasks such as dealing cards and responding to player commands, thanks to iterative improvements in arm movements, gripper functionality, and system synchronization. The calibration of the gripper, enhanced by the addition of elastic bands, proved to be an effective solution for handling delicate cards, ensuring precise placements and pickups.

Despite its successes, the project revealed several limitations that offer opportunities for future enhancements. The iterative approach to refining arm movements increased the system's stability but also introduced greater complexity in the codebase, impacting maintainability. Additionally, the reliance on optimal lighting conditions for card recognition highlighted the system's vulnerability to environmental factors, requiring precise workspace setups. The sequential scanning method for card recognition improved accuracy but slowed down the gameplay process, which could be optimized in future iterations.

In conclusion, the project achieved its primary goals of automating a Blackjack game while identifying key challenges and practical solutions.

References

[1] T. Gop. *Playing Cards Object Detection*. Accessed: Dec. 19, 2024. 2024. URL: https://github.com/TeogopK/Playing-Cards-Object-Detection/tree/main?tab=readme-ov-file.

[2] Mecalux. *Robotics Applications in Logistics*. Published: 2024, Accessed: Nov. 18, 2024. 2024. URL: https://www.mecalux.com/blog/robotics-applications-logistics.

[3] NumPy. *NumPy Documentation*. Accessed: Dec. 5, 2024. 2024. URL: https://numpy.org/doc/.

[4] OpenCV. *OpenCV Python Tutorials*. Accessed: Nov. 27, 2024. 2024. URL: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.

[5] Ultralytics. *Ultralytics YOLO Documentation*. Accessed: Nov. 24, 2024. 2024. URL: https://docs.ultralytics.com/de.