Blackjack Master: Blackjack Card Counting with Classical Computer Vision Techniques

Alexander Wang
Stanford
alexjw3@stanford.edu

Nicholas Reisner
Stanford
nreisner@stanford.edu

Parrish Pipestem
Stanford
pipestem@stanford.edu

Abstract

This project presents a computer vision-based system for detecting and analyzing blackjack hands from real-world images. Using a combination of classical computer vision techniques, the system identifies playing cards, determines the optimal move based on blackjack basic strategy, and maintains a running card count using the Hi-Lo method. The approach relies on edge detection, contour approximation, and template matching to recognize individual cards accurately. The system then overlays game information onto a visual heads-up display (HUD), providing real-time strategy recommendations and betting advice. Testing on various images demonstrated high accuracy in card recognition and strategic decision-making. The results suggest that this lightweight, non-learning-based method is an effective and efficient tool for blackjack strategy assistance, with potential applications in real-time gameplay analysis and training. Future improvements include enhancing robustness against occlusions, variations in lighting, and rotational distortions.

1. Introduction

1.1. Motivation

Blackjack has retained its allure in casinos worldwide by striking a balance between luck and skill. Basic strategy already brings players close to the house edge, but card counting nudges the odds further by enabling well-timed bets and situational decisions. This project sets out to automate both card recognition and count tracking using computer vision, blending a centuries-old game with modern technology.

1.2. Significance

While typical blackjack software often relies on manual inputs or simulated graphics, our approach handles actual card images captured at the table. By integrating computer vision to identify ranks, keep a running count, and offer betting or play suggestions, the system opens the door to advanced practice tools, real-time assistance, and streamlined data gathering.

1.3. Challenges

While blackjack is a logical application of computer vision, a number of challenges exist:

1. **Variability in Card Images** – Angles, shadows, and partial obstructions can cause unpredictable outputs, requiring robust detection methods.

2. **Accuracy vs. Performance** – Edge-based and template-matching steps must be tuned to run fast enough for real-world scenarios without sacrificing precision.

3. **Integration** – Bringing the detection pipeline together with count-based strategy calls for careful handling of data flows, ensuring each module works seamlessly with the rest.

2. Related Work

2.1. Playing Card Recognition

The task of recognizing playing cards has been explored primarily via short academic or personal projects and have employed a variety of computer vision techniques, ranging from traditional image processing methods to more advanced machine learning approaches.

Several projects have highlighted the efficacy of the classical computer vision techniques. A popular approach is the use of feature extraction methods such as SIFT (Scale-Invariant Feature Transform) [1] or ORB (Oriented FAST and Rotated BRIEF) [2]. Feature extraction methods are effective in a controlled settings, but their adaptability to real-world gameplay environments is limited.

More recent developments have incorporated deep learning techniques, particularly convolutional neural networks (CNNs), to improve card classification [3]. These models are capable of handling variations in lighting, angle, and occlusions more effectively than traditional methods. However, they require large datasets for training and can be computationally intensive, making them less ideal for real-time applications.

2.2. Blackjack Strategy

Various methods have been explored to enhance decision-making in blackjack. Many AI-driven approaches focus on reinforcement learning to develop optimal strategies, where an agent learns by playing thousands of simulated games. Some projects have implemented Monte Carlo simulations or Deep Q-Networks (DQN) [4] to predict the best move based on the current game state.

While these techniques perform well in simulations, they have drawbacks. First, reinforcement learning approaches, while powerful, may fail to follow the well-established optimal strategy [5]. Additionally, these techniques are designed to work with digitally provided inputs rather than extracting information directly from visual sources. This makes them impractical for real-world gameplay, where hands must be identified from physical cards before making strategic decisions.

2.3. Our Approach

Our system combines two core components—card detection and strategy recommendation—by leveraging classical computer vision techniques alongside established blackjack strategies. The key features of our system include:

1. Card detection through edge detection and contour approximation, allowing for efficient and reliable extraction of playing cards from images.

2. Rank recognition via template matching, removing the need for complex feature extraction or neural network-based classification.

3. Strategy recommendation based on the well-known Hi-Lo card counting system, enabling dynamic betting suggestions and decision making by estimating the player's advantage as the game progresses.

Compared to machine learning approaches, this method offers faster performance, greater interpretability, and significantly lower data requirements. It does not rely on large, labeled datasets or extensive training, making it easier to implement and deploy. The result is a lightweight and transparent system that runs in real time, making it well-suited for constrained environments such as mobile devices or embedded systems.

3. Methodology

This section outlines our approach to detecting playing cards, extracting relevant information, and providing real-time blackjack strategy recommendations. The system employs classical computer vision techniques, template matching, and blackjack strategy logic to ensure efficiency and reliability.

3.1. Card Detection Pipeline

3.1.1 Image Preprocessing

Before identifying individual cards, the input image is preprocessed. Each input image is converted to grayscale, reducing visual complexity and enabling other classical techniques.

We experimented extensively with more extensive preprocessing. For example, we implemented a Gaussian blur and color-based filtering that targeted the white cards. However, we ultimately observed that these methods either had no measurable benefit or even slightly degraded performance. Given the improved computational efficiency of less processing, we opted to retain only the grayscale conversion in the final pipeline.

3.1.2 Card Detection

To identify candidate card regions within the image, we first applied Canny edge detection to isolate areas of rapid intensity change. Figure 1 shows the original input alongside its corresponding edge map. From this edge map, we extracted all external contours, which were then smoothed to reduce noise.

Contours were identified as cards using a polygonal approximation. We fit a polygon to each contour and applied a set of geometric filters to identify card candidates.

1. Four edges to match the rectangular geometry of standard playing cards.

2. Area above a user-set minimum area to remove small objects.

3. Approximately parallel opposing sides to ensure the card is rectangular.

These geometric constraints helped eliminate background objects and noise that might otherwise interfere with detection. The final filtered polygons, as illustrated in Figure 2, are used to isolate the cards.

3.2. Rank Recognition

After identifying the locations of potential cards, each detected card region was individually processed to extract its rank (i.e., number or letter). This process focused on

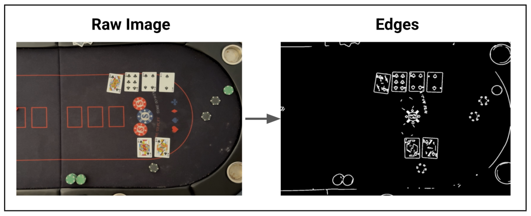

Figure 1. Canny edge detection. Left: the original input image. Right: the edge map produced by Canny edge detection, highlighting areas of rapid intensity change.

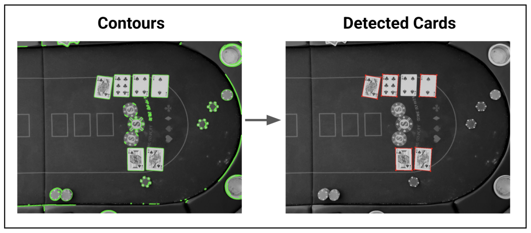

Figure 2. Card detection through contour approximation. Left: raw contours detected from the edge map. Right: resulting polygons filtered and identified as candidate card regions.

analyzing the top-left corner of each card, where the value is typically printed in a standardized position.

Each card's top-left corner was first cropped to isolate a small portion of the card containing the identifying mark. This region was then normalized in brightness and contrast and then put through a binary threshold to separate the colored features from the white background.

To extract just the rank character, we inverted the binary region and searched for clearly defined contours. Small shapes or those touching the edge of the image were discarded, and then the largest suitable contour—assumed to be the number or letter—was retained. Final checks ensured that the extracted region was of an appropriate size to avoid false positives (e.g., the patterned back of a card or a falsely identified card altogether). The full pipeline for rank extraction is illustrated in Figure 3.

The isolated character was then compared against a set of known templates using a similarity-based matching approach. We initially experimented with templates created from close-up photos of each card rank. These were processed using fixed thresholding, then shrunk and re-enlarged to simulate the effects of the pipeline on photos taken a reasonable distance from the card. However, these handcrafted templates did not reliably match the true pipeline. We ultimately achieved better results by generating templates from the same table-shot images used during evaluation. Specifically, we selected one clear example of each card from our dataset and extracted its rank using our full processing pipeline to use as our template. These in-context templates generalized well and led to more consistent and accurate matching performance.

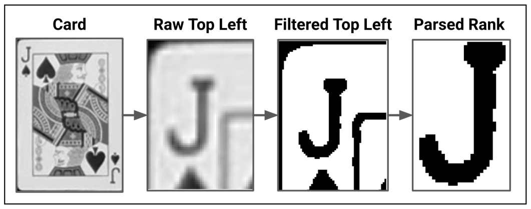

Figure 3. Rank extraction pipeline. Far left: the full detected card. Middle left: the cropped top-left region. Middle right: the binary thresholded region. Far right: the isolated rank character after contour filtering.

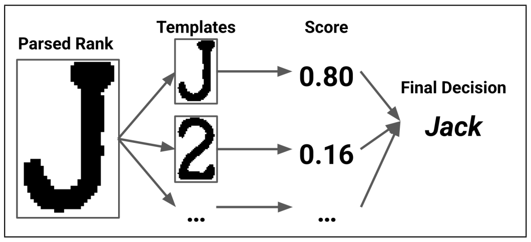

Figure 4. Template matching example. Far left: the parsed rank from a test image. Middle left: the full set of templates. Middle right: normalized correlation scores. Far right: the selected best match.

These templates were resized to match the dimensions of the detected character region, and the best match was selected based on a normalized correlation score. If no match was above a set threshold or the character region could not be isolated altogether, the card was excluded from further analysis. The matching process and score visualization are shown in Figure 4.

3.3. Strategy Recommendation

We implemented the Hi-Lo card counting strategy, which estimates the player's advantage by tracking the ratio of high to low cards remaining in the deck. This provides a basis for adjusting bet sizes and informs decision making based on the current game state using the established probabilities of blackjack.

Once each card in the image was identified and matched to a rank, we reconstructed the game state by assigning cards to either the dealer or the player based on the vertical position of cards in the image (cards appearing higher were the dealer cards and those lower down were assigned to the player).

If the hand was in progress, we determined an optimal move using a basic blackjack strategy lookup. At the end of each hand, we updated the running count by applying a standard card counting system, adjusting the count based on the values of all revealed cards. The true count was then calculated to generate a betting suggestion based on predefined

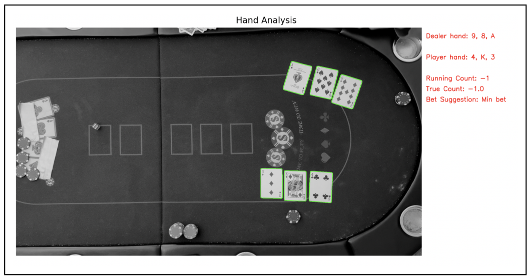

Figure 5. Final system output. The display includes the dealer's and player's hands, optimal strategy recommendation, running and true count, and bet suggestion.

thresholds.

Finally, the results were provided in a user-friendly visual display, as shown in Figure 5. The display noted the dealer's and player's hands, recommended strategy, running count, true count, and betting suggestion.

4. Results

4.1. Testing and Evaluation

Our system was evaluated on a dataset of blackjack hands captured on a real poker table (i.e. relatively busy background) from a bird's-eye view perspective and given a variety of card orientations. We examined the performance of our system up to each major sub-system. We utilized a development set of 10 images, containing 33 cards in total, and a testing set of 97 cards over 25 images.

4.1.1 Card Detection

We first evaluated how well the system detected individual cards in an image. We counted a card as accurately detected if the top-left region that is subsequently extracted has the rank character in it. The results are as shown in Table 1.

Task	**Development Set**		**Test Set**	
	Precision	**Recall**	**Precision**	**Recall**
Card Detection	1.000	1.000	0.979	0.948

Table 1. Card detection performance on the development set (33 cards over 10 images) and test set (97 cards over 25 images).

These values indicate that the system correctly identifies most playing cards while minimizing false positives. The high precision suggests that the system rarely misclassifies non-card regions as cards, while the slightly lower recall score reflects the system's tendency to miss cards over falsely identifying them.

4.1.2 Rank Recognition

Once a card was detected, the system assigned a rank using template matching. The accuracy of this process was measured based on how often the detected rank matched the ground truth. The results are found in Table 2.

Task	**Development Set**		**Test Set**	
	Precision	**Recall**	**Precision**	**Recall**
Rank Recognition	1.000	1.000	0.978	0.928

Table 2. Rank recognition performance on the development set (33 cards over 10 images) and test set (97 cards over 25 images).

The precision remains roughly the same between card detection and rank recognition, indicating that when the system identifies a card and assigns a rank, it is usually correct. However, the recall drops at the rank recognition stage because it accumulates the errors from the earlier card detection stage. If a card is missed entirely or its top-left region is incorrectly extracted, it cannot be successfully assigned a rank even if the recognizer itself would have worked correctly.

4.1.3 Strategy Recommendation

Lastly, we evaluated our system's ability to make the correct Hi-Low recommendation. In other words, we evaluated the performance of our system to correctly assess the game state—which can then be fed into the pre-established Hi-Low strategy. So in addition to the aforementioned, upstream tasks of card detection and rank recognition, we now also assess our system's ability to assign cards to the player versus the dealer. The results are shown in Table 3.

Metric	**Development Set**	**Test Set**
Accuracy	1.000	0.880

Table 3. Strategy recommendation performance on the development set (10 images) and test set (25 images).

The system consistently provided correct strategic guidance based on detected hands. In addition to the aforementioned upstreamed errors, this step faced unique challenges from game state edge cases. For example, scenarios where the player splits (i.e. makes two hands out of one) were not present in the development set.

4.2. Limitations

Despite strong performance, the system has a few limitations:

- **General Robustness** – We found strong performance even though our system was developed and tested on an authentic poker table in real-life, non-optimal lighting conditions. The system was able to accurately isolate cards in most cases, however some scenarios presented room to improve general robustness. For example, in instances when the card was placed on a busy section of table pattern, the pattern would be incorrectly merged with the edge of the card, leading to false negatives. Other issues occurred if the cards overlapped or glare obscured the card's features.

- **Point-of-View** – Our initial system was developed and tested on images captured from a bird's-eye view. To explore more realistic gameplay scenarios, we extended the system to handle images taken from the player's perspective. While cards could still be detected under this angled view, the resulting perspective transformation sometimes produced low-resolution or distorted top-left regions. This degradation impacted the accuracy of rank recognition, as template matching relies on clear, consistently aligned input.

5. Conclusion and Next Steps

This work presents an end-to-end system for blackjack hand recognition and strategic analysis using only classical computer vision techniques. By leveraging edge detection, contour approximation, and template matching, we developed a lightweight and interpretable system capable of detecting cards, recognizing ranks, and providing strategy recommendations in real time. Our system performed with high accuracy across a variety of real-world images, demonstrating its robustness even in challenging environments.

Looking forward, several promising directions remain. Expanding support for more complex game scenarios and improving performance under angled, low-resolution perspectives are key to real-world deployment. Also, while our system already functions well without the need for machine learning, incorporating learning-based components for specific tasks (e.g., rank recognition under occlusion) might improve performance without compromising interpretability or efficiency. Finally, adapting the system from still images to live video could enable a variety of real-time use cases.

6. Individual Contributions

- **Parrish Pipestem** – Developed the image preprocessing pipeline, including edge detection and card contour isolation. Integrated basic strategy and built the HUD.

- **Alex Wang** – Implemented card extraction and card counting. Responsible for data collection/cleaning and led the testing and evaluation phase.

- **Nicholas Reisner** – Built the advanced rank parsing and template matching functionality. Responsible for performance optimization and project presentation.

References

[1] D. Snyder, ”Playing Card Detection and Identification,” *Stanford University, EE368 Final Report*, 2019. 1

[2] Siromer, ”Detecting and Tracking Objects with ORB using OpenCV,” *Medium*, 2019. 1

[3] M. Castillo, B. Goeing, and J. Westell, ”Computer Vision for Card Games,” *Stanford University, CS229 Final Report*, 2017. 1

[4] J. Bramsen and A. Shankar, ”Card Recognition and Blackjack Strategy Using Deep Learning,” *Stanford University, CS230 Final Report*, 2018. 2

[5] J. R. Reiss, ”A Comparison between Cognitive and AI Models of Blackjack Strategy Learning,” *ResearchGate*, 2014. 2