הנדסת תוכנה
שנקר
מהנדסים טאלנטים בתוכנה

Jackoscope: Smart Card Counting Detection System

Student Names: Dean Shalev, Omer Portnoy

Student IDs: 209707470, 207251018

PROJECT BOOK FOR THE COURSE:

Methods in Software Engineering

Abstract

Jackoscope represents a transformative approach to casino security, introducing a real-time card counting detection system that revolutionizes traditional surveillance methods. Designed as a virtual monitoring service, our system facilitates seamless analysis of blackjack players, akin to having an AI-powered observer at each table. Utilizing computer vision and machine learning algorithms, our solution captures real-time data on player behaviour, betting patterns, and card values, enabling casino staff to identify potential card counters promptly.

This system integrates advanced technology into casino operations, aiming to enhance fair play by offering data-driven insights rooted in both visual assessments and statistical analysis. Casinos will benefit from a user-friendly interface, real-time alerts for suspicious activity, and a comprehensive platform for monitoring player behaviour over time, ensuring the integrity of blackjack games.

The Jackoscope system addresses the long-standing challenge of detecting card counting in casinos, a practice that, while not illegal, can significantly impact casino profits. By automating the detection process, Jackoscope provides a consistent, objective, and scalable solution that surpasses traditional human observation methods in both accuracy and efficiency.

Table of Contents

Table of Figures

1. Introduction

1.1. Overview

Jackoscope is an innovative software-based system designed to revolutionize the way casinos detect card counting in blackjack games. By leveraging advanced computer vision techniques, machine learning algorithms, and real-time data analysis, Jackoscope provides casino management with a powerful tool to maintain fair play and protect their interests.

The system continuously monitors blackjack tables through video feeds, recognizing cards, tracking player actions, and analyzing betting patterns. When suspicious behaviour indicative of card counting is detected, Jackoscope generates real-time alerts, allowing casino staff to respond promptly and discreetly.

1.2. Problem Description

Card counting in blackjack has been a persistent challenge for casinos since the 1960s when mathematical strategies for gaining an advantage over the house were first publicized. While not illegal, card counting can significantly impact casino profits, especially when employed by skilled practitioners.

Traditional methods of detecting card counters rely heavily on human observation, which presents several limitations:

- Inconsistency: Human observers may vary in their ability to spot card counting behavior.
- Fatigue: Constant vigilance is difficult to maintain over long periods.
- Limited coverage: It's challenging to monitor multiple tables simultaneously.
- Subjectivity: Decisions based on human observation can be influenced by personal biases.
- Training requirements: Substantial training is needed for staff to effectively spot card counters.

These limitations create a need for a more reliable, consistent, and scalable solution to protect casino interests while ensuring fair play for all patrons.

1.3. Motivation

The primary motivation behind the development of Jackoscope is to address the shortcomings of traditional card counting detection methods and provide casinos with a cutting-edge technological solution. Key motivating factors include:

1. Enhancing fairness: By accurately detecting card counting, casinos can maintain a level playing field for all players.
2. Protecting profits: Minimizing losses due to advantage play helps casinos maintain financial stability.
3. Improving efficiency: Automating the detection process allows security staff to focus on other critical tasks.

4. Increasing coverage: The ability to monitor multiple tables simultaneously enhances overall security.
5. Providing objective data: Data-driven insights reduce reliance on subjective human judgement.
6. Staying ahead of sophisticated counters: As counting techniques evolve, so too must detection methods.

1.4. Project Goals

The Jackoscope project aims to achieve the following primary goals:

- Develop a highly accurate card recognition system capable of identifying card values and suits in real-time from video feeds.
- Implement a robust algorithm for detecting card counting behavior based on betting patterns and play decisions.
- Create a user-friendly interface for casino staff to monitor tables, view alerts, and analyze historical data.
- Ensure the system operates discreetly without disrupting the gaming experience for honest players.
- Achieve a detection accuracy rate of at least 95% for known card counting techniques.
- Design a scalable system architecture capable of monitoring multiple tables concurrently.
- Implement strong security measures to protect sensitive data and prevent system tampering.
- Develop comprehensive documentation and training materials for easy system adoption and use.

1.5. Approach

Jackoscope employs a multi-faceted approach to achieve its goals:

- Computer Vision: Utilizes advanced image processing techniques to recognize and track cards on the table.
- Machine Learning: Employs trained models to analyze player behavior and detect patterns indicative of card counting.
- Real-time Processing: Processes video feeds and generates alerts with minimal latency.
- Data Analytics: Provides in-depth analysis of historical data to identify long-term trends and patterns.
- User-Centric Design: Focuses on creating an intuitive interface that caters to the needs of casino staff.
- Modular Architecture: Designs the system with separate components for easy maintenance and future enhancements.
- Continuous Learning: Implements feedback mechanisms to improve detection accuracy over time.

1.6. End-Users Description and Scenarios

Jackoscope is designed to serve various roles within a casino's operations. The primary end-users of the system include:

1. Casino Managers:
   - Profile: Senior staff responsible for overall casino operations and decision-making.
   - Needs: High-level insights, trend analysis, and system performance reports.
2. Security Staff:
   - Profile: Trained personnel tasked with maintaining game integrity and responding to suspicious activities.
   - Needs: Real-time alerts, detailed player information, and tools for immediate response.
3. Surveillance Operators:
   - Profile: Technical staff monitoring video feeds and security systems.
   - Needs: Integration with existing surveillance systems, multiple table views, and video playback capabilities.
4. IT Administrators:
   - Profile: Technical experts responsible for system maintenance and troubleshooting.
   - Needs: System health monitoring, configuration tools, and update management.
5. Compliance Officers:
   - Profile: Staff ensuring adherence to gaming regulations and internal policies.
   - Needs: Audit logs, data export capabilities, and compliance reporting tools.

1.7. Scenarios

Scenario 1

Process Name	Detecting a Skilled Card Counter
Participating Actor	Alice – Casino Manager
Flow of Events	Alice receives an alert from Jackoscope indicating suspicious betting patterns at Table 7. The system has detected a correlation between the player's bet sizes and the true count of the deck, a classic sign of card counting. Alice opens the Jackoscope interface and reviews the provided data:

	• A graph showing bet sizes over time, overlaid with the estimated true count. • Video snippets of key moments in the player's session. • A summary of the player's win/loss record and duration of play. Based on this information, Alice decides to have a security team member observe the table discreetly. The security staff uses Jackoscope's mobile interface to view real-time updates on the player's activities while maintaining a presence on the casino floor. After confirming the suspicious behavior, Alice decides to approach the player, kindly informing them that they are no longer welcome to play blackjack at the casino, as per house rules regarding advantage play.

Scenario 2

Process Name	System Calibration and Testing
Participating Actor	Bob – IT Administrator
Flow of Events	Bob, an IT administrator, is tasked with calibrating Jackoscope for a newly opened blackjack table. He follows these steps: 1. Accesses the system's configuration interface and adds the new table to the monitoring list. 2. Adjusts the camera settings to ensure optimal card recognition for the table's lighting conditions. 3. Runs a series of test scenarios, including simulated card counting behavior, to verify system accuracy. 4. Fine-tunes detection thresholds based on the test results to minimize false positives while maintaining high sensitivity. 5. Generates a calibration report for management review and approval. Throughout the process, Bob uses Jackoscope's built-in testing tools to validate each component of the system, ensuring that card

	recognition, betting pattern analysis, and alert generation all function correctly for the new table.

1.8. Audience

The primary audience for the Jackoscope project encompasses various stakeholders within the casino industry:

1. Casino Owners and Executives:
   - Interest: Protecting profits, ensuring fair play, and leveraging technology for competitive advantage.
   - Expectations: A robust, reliable system that provides clear ROI and integrates smoothly with existing operations.
2. Casino IT Departments:
   - Interest: Implementing and maintaining advanced technical solutions.
   - Expectations: Well-documented, secure, and maintainable software with good vendor support.
3. Security and Surveillance Teams:
   - Interest: Enhancing their capability to detect and prevent advantage play.
   - Expectations: User-friendly tools that augment their expertise and streamline their workflows.
4. Regulatory Bodies:
   - Interest: Ensuing that technological solutions comply with gaming regulations.
   - Expectations: Transparent operations, audit trails, and fair treatment of players.
5. Software Developers and Engineers:
   - Interest: Understanding the technical implementation of AI and computer vision in a real-world application.
   - Expectations: Insights into system architecture, algorithm design, and integration challenges.
6. Academic Researchers:
   - Interest: Studying the application of machine learning and computer vision in gaming security.
   - Expectations: Methodologies, performance metrics, and potential areas for further research.

1.9. Glossary

- Card Counting: A strategy used in blackjack to determine when the player has a statistical advantage over the house.
- True Count: In card counting, the running count divided by the number of decks remaining to be dealt.

- Advantage Play: Any legal method used by players to gain a statistical advantage over the casino.
- Computer Vision: A field of artificial intelligence that trains computers to interpret and understand visual information.
- Machine Learning: The use of algorithms and statistical models that enable computer systems to improve their performance on a specific task through experience.
- False Positive: An error in which the system incorrectly identifies a player as a card counter.
- Latency: The delay between the input (e.g., a player's action) and the output (e.g., system alert) in a computing system.
- API (Application Programming Interface): A set of protocols and tools for building software applications.
- SQL (Structured Query Language): A standard language for managing and manipulating relational databases.
- Git: A distributed version control system for tracking changes in source code during software development.

2. Technological Survey

This section provides a comparative analysis of different technologies, such as programming languages, databases, and algorithms, to identify the best options that fit our project's constraints, requirements, and objectives.

2.1. Programming languages

The choice of programming languages for Jackoscope was driven by the specific requirements of each system component. After careful consideration, the following languages were selected:

- Python:
  - Primary language for backend development and data processing.
  - Reasons for selection:
    a. Rich ecosystem of libraries for computer vision (OpenCV) and machine learning (TensorFlow, PyTorch).
    b. Excellent performance for data processing tasks.
    c. Clear, readable syntax facilitating easier maintenance and collaboration.
    d. Strong community support and extensive documentation.

- JavaScript:
  - Used for frontend development to create an interactive and responsive user interface.
  - Reasons for selection:
    a. Enables dynamic, client-side interactions without page reloads.
    b. Wide array of frameworks and libraries (e.g., React) for building complex UIs.
    c. Asynchronous programming capabilities for smooth real-time updates.
    d. Universal browser support.

- SQL:
  - Employed for database management and querying.
  - Reasons for selection:
    a. Efficient for handling structured data and complex queries.
    b. ACID compliance ensures data integrity.
    c. Widely understood and used in the industry.

- HTML5/CSS3:
  - Used for structuring and styling the web-based user interface.
  - Reasons for selection:
    a. Industry standard for web content structure and presentation.
    b. Responsive design capabilities for various screen sizes.
    c. Broad browser support and rendering consistency.

2.2. **Frameworks and Libraries**

- OpenCV:
    - Purpose: Computer vision tasks, including card recognition and image processing.
    - Key features utilized:
      a. Image preprocessing (filtering, thresholding)
      b. Contour detection for identifying card shapes
      c. Template matching for recognizing card ranks and suits

- TensorFlow:
    - Purpose: Machine learning model for pattern recognition and behavior analysis.
    - Key features utilized:
      a. Neural network architecture for classifying betting patterns
      b. Transfer learning for adapting pre-trained models to our specific use case
      c. TensorFlow Serving for deploying models in production

- Flask:
    - Purpose: Web framework for the backend API.
    - Key features utilized:
      a. RESTful API design
      b. WebSocket support for real-time communication
      c. Integration with SQLAlchemy for database operations

- React:
    - Purpose: Frontend framework for building the user interface.
    - Key features utilized:
      a. Component-based architecture for reusable UI elements
      b. Virtual DOM for efficient rendering and updates
      c. React Hooks for state management

- SQLAlchemy:
    - Purpose: SQL toolkit and Object-Relational Mapping (ORM) library.
    - Key features utilized:
      a. Database abstraction layer for supporting multiple database backends
      b. Query optimization and caching
      c. Migration tools for managing database schema changes

- NumPy and Pandas:

- Purpose: Data manipulation and analysis.
- Key features utilized:
  a. Efficient array operations for image processing
  b. Data structuring and analysis for betting pattern recognition
  c. Statistical functions for probability calculations

- Matplotlib and Plotly:

  - Purpose: Data visualization for the user interface.
  - Key features utilized:
    a. Interactive charts and graphs for displaying betting trends
    b. Real-time plotting capabilities
    c. Customizable visualizations for different data types

2.3. **Hardware**

The hardware requirements for Jackoscope were carefully considered to ensure optimal performance and reliability:

- Cameras:
  - Specification: High-definition IP cameras with at least 1080p resolution and 30 fps.
  - Justification: Clear, high-quality video feeds are essential for accurate card recognition.
- Server:
  - Specification: High-performance server with multi-core CPU, minimum 32GB RAM, and SSD storage.
  - Justification: Processing multiple video streams and running machine learning models in real-time requires significant computational resources.
- Network Infrastructure:
  - Specification: Gigabit Ethernet or higher, with low-latency switches.
  - Justification: Ensures smooth transmission of video data from cameras to the central processing server.
- Workstations:
  - Specification: Modern PCs or laptops with multi-core processors and dedicated graphics cards.
  - Justification: Enables smooth operation of the user interface and real-time video playback for security staff.
- Mobile Devices:
  - Specification: Tablets or smartphones with high-resolution displays and modern web browsers.
  - Justification: Allows security personnel to access the system while moving around the casino floor.

2.4. **Frameworks and Libraries**

Jackoscope employs several key algorithms to achieve its functionality:

- Card Recognition Algorithm:

    - Purpose: Identify and classify playing cards from video frames.
    - Approach:
      a. Image preprocessing (grayscale conversion, thresholding)
      b. Contour detection to isolate card shapes
      c. Perspective transformation to obtain a top-down view of each card
      d. Template matching or CNN-based classification for rank and suit identification

- Betting Pattern Analysis:

    - Purpose: Detect patterns in betting behavior that may indicate card counting.
    - Approach:
      a. Time series analysis of bet sizes
      b. Correlation analysis between bet sizes and estimated true count c. Statistical tests for detecting non-random betting patterns

- Card Counting Detection:

    - Purpose: Estimate the count maintained by a potential card counter.
    - Approach:
      a. Implement common card counting systems (e.g., Hi-Lo, KO, Omega II)
      b. Maintain a running count based on observed cards
      c. Calculate true count by estimating the number of decks remaining

- Player Behavior Classification:

    - Purpose: Classify players as potential card counters or ordinary players.
    - Approach:
      a. Feature extraction from betting patterns, play decisions, and estimated count
      b. Machine learning classification (e.g., Random Forest, SVM, or Neural Network)
      c. Continuous model updating based on confirmed detections and false positives

- Alert Generation:

    - Purpose: Generate timely and relevant alerts for casino staff.
    - Approach:
      a. Define thresholds for suspicious behavior based on multiple factors
      b. Implement a scoring system that combines various indicators

c. Use time-based rules to avoid alert fatigue (e.g., cooldown periods between alerts for the same player)

These algorithms work in concert to provide a comprehensive card counting detection system, balancing accuracy with computational efficiency to enable real-time monitoring of multiple tables.

3. Requirements

3.1. Functional requirements

The Jackoscope system must fulfill the following functional requirements.

- **Card Recognition:**
  - FR1.1: The system shall accurately recognize and digitize playing cards from video feeds in real-time.
  - FR1.2: Card recognition accuracy must be at least 99% under normal casino lighting conditions.
  - FR1.3: The system shall handle partially obscured cards and fast dealer movements.
- **Player Tracking:**
  - FR2.1: The system shall associate recognized cards and bets with specific player positions at the table.
  - FR2.2: The system shall track players across multiple hands and sessions.
  - FR2.3: The system shall handle player changes at table positions without manual intervention.
- **Betting Pattern Analysis:**
  - FR3.1: The system shall record and analyze betting patterns for each tracked player.
  - FR3.2: The system shall calculate statistical measures of betting behavior, including bet spread and correlation with the true count.
  - FR3.3: The system shall detect sudden changes in betting behavior that may indicate team play or player substitution.
- **Card Counting Detection:**
  - FR4.1: The system shall implement multiple card counting systems (e.g., Hi-Lo, KO, Omega II) and track the count for each.
  - FR4.2: The system shall estimate the true count based on the cards played and the estimated number of decks remaining.
  - FR4.3: The system shall correlate player betting patterns with the estimated count to detect potential card counting.
- **Alert Generation:**
  - FR5.1: The system shall generate real-time alerts when suspected card counting behavior is detected.
  - FR5.2: Alerts shall include the player's position, a confidence score, and a summary of the suspicious behavior.
  - FR5.3: The system shall provide configurable alert thresholds to balance sensitivity with false positive rate.
- **User Interface:**
  - FR6.1: The system shall provide a web-based interface accessible from desktop and mobile devices.

- FR6.2: The interface shall display real-time video feeds from monitored tables with overlay information (e.g., detected cards, bet amounts).
- FR6.3: Users shall be able to view historical data, including betting patterns and alert history.

- **Reporting:**
  - FR7.1: The system shall generate daily, weekly, and monthly reports on detected suspicious activities.
  - FR7.2: Reports shall include statistics on system performance, including false positive rates and detection accuracy.
  - FR7.3: The system shall allow custom report generation based on user-specified parameters.
- **System Administration:**
  - FR8.1: The system shall provide an interface for adding, removing, and configuring monitored tables.
  - FR8.2: Administrators shall be able to update card recognition models and detection algorithms without system downtime.
  - FR8.3: The system shall maintain audit logs of all administrative actions and system events.
- **Integration:**
  - FR9.1: The system shall provide APIs for integration with existing casino management and surveillance systems.
  - FR9.2: The system shall support importing player information from casino loyalty programs for enhanced tracking.
- **Data Management:**
  - FR10.1: The system shall store historical data for a configurable period, with a minimum of 6 months.
  - FR10.2: The system shall provide data export functionality in common formats (e.g., CSV, JSON).
  - FR10.3: The system shall implement data retention policies in compliance with relevant regulations.

3.2. Non-Functional requirements

The Jackoscope system must also meet the following non-functional requirements:

- **Performance:**
  - NFR1.1: The system shall process video feeds with a maximum latency of 500ms.

- NFR1.2: The system shall support simultaneous monitoring of at least 50 tables without degradation in performance.
- NFR1.3: Database queries for historical data shall return results within 2 seconds for datasets up to 1 million records.

- **Reliability:**
  - NFR2.1: The system shall have an uptime of at least 99.9% (excluding scheduled maintenance).
  - NFR2.2: The system shall implement fault-tolerant mechanisms to handle hardware failures without data loss.
  - NFR2.3: Regular automated backups shall be performed with a recovery point objective (RPO) of 1 hour.

- **Security:**
  - NFR3.1: All data transmissions shall be encrypted using industry-standard protocols (e.g., TLS 1.3).
  - NFR3.2: The system shall implement role-based access control with fine-grained permissions.
  - NFR3.3: The system shall comply with relevant data protection regulations (e.g., GDPR, CCPA).

- **Scalability:**
  - NFR4.1: The system architecture shall allow horizontal scaling to accommodate growth in the number of monitored tables.
  - NFR4.2: The database design shall support efficient querying and indexing for large datasets.

- **Usability:**
  - NFR5.1: The user interface shall be intuitive and require minimal training for basic operations.
  - NFR5.2: The system shall provide context-sensitive help and tooltips for all features.
  - NFR5.3: The interface shall be responsive and compatible with major web browsers and mobile devices.

- **Maintainability:**
  - NFR6.1: The system shall be designed with modular architecture to facilitate easy updates and maintenance.
  - NFR6.2: Comprehensive documentation shall be provided for all system components and APIs.

- NFR6.3: The system shall include built-in diagnostics and logging for troubleshooting.

- **Compliance:**
  - NFR7.1: The system shall adhere to relevant gaming regulations and standards.
  - NFR7.2: All algorithms and detection methods shall be transparent and explainable to regulatory bodies if required.

- **Localization:**
  - NFR8.1: The user interface shall support multiple languages, with initial support for English, Chinese, and Spanish.
  - NFR8.2: The system shall handle different currencies and chip denominations used in various casinos.

These requirements form the foundation for the design and implementation of the Jackoscope system, ensuring it meets the needs of casino operators while maintaining high standards of performance, security, and usability.

4. Architecture.

4.1. The Jackoscope system employs a distributed, multi-tier architecture, to ensure scalability, maintainability, and performance. The architecture is divided into the following main components:

1. Data Acquisition Layer -
    - Comprises the network of cameras and sensors deployed at blackjack tables.
    - Responsible for capturing high-quality video feeds and transmitting them to the processing layer.
2. Processing Layer -
    - Consists of high-performance servers running the core Jackoscope software.
    - Handles video processing, card recognition, and real-time analysis.
3. Data Storage Layer -
    - Utilizes a distributed database system for storing historical data, user information, and system configurations.
    - Implements data portioning and replication for performance and reliability.
4. Application Layer -
    - Hosts the web server and application logic.
    - Manages the user authentication, access control, and session management.
    - Implements the RESTful API for client-server communication.
5. Presentation Layer -
    - Delivers the user interface to various client devices (desktops, tables, smartphones).
6. Integration Layer -
    - Provides interfaces for integrating with external system (e.g., casino management systems, surveillance platforms).
    - Implements message queues for asynchronous communication between system components.

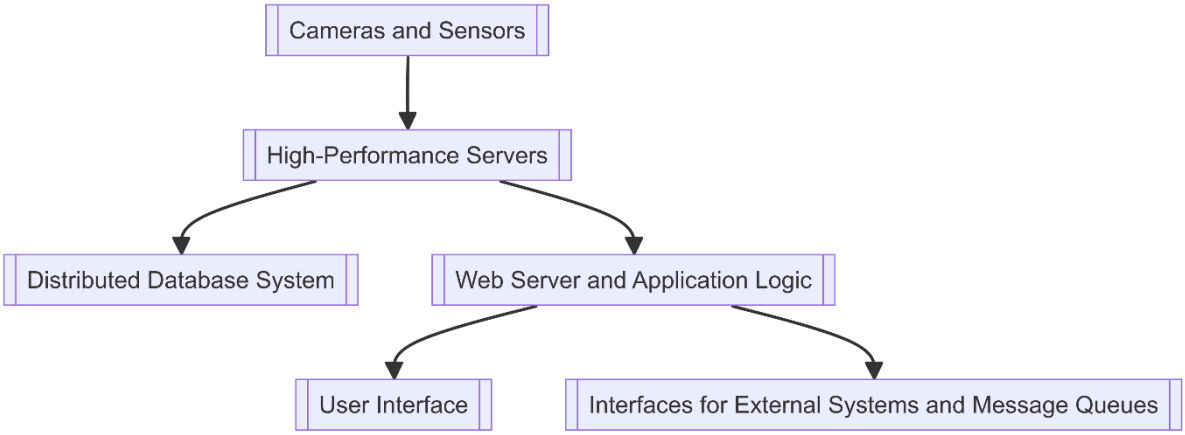

Figure 1: System Architecture

4.2. Data Flow

The data flow within the Jackoscope system follows these main paths:

1. Video Feed Processing:
    a. Cameras capture video of blackjack tables.
    b. Video streams are transmitted to the processing servers.
    c. Image processing algorithms extract relevant information (cards, bet amounts).
    d. Extracted data is temporarily stored in memory for real-time analysis.
2. Player Behavior Analysis:
    a. Betting patterns and play decisions are analyzed in real-time.
    b. Machine learning models assess the likelihood of card counting.
    c. Results are stored in the database and used to generate alerts if necessary.
3. Alert Generation and Notifications:
    a. When suspicious behavior is detected, an alert is generated.
    b. The alert is stored in the database and sent to the application server.
    c. Relevant users are notified through the user interface and/or mobile notifications.
4. Historical Data Analysis:
    a. User requests historical data through the UI.
    b. The application server queries the database.
    c. Retrieved data is processed and formatted.
    d. Results are sent back to the client for display.
5. System Configuration and Management:
    a. Administrators make changes through the management interface.
    b. Configuration updates are stored in the database.
    c. Relevant system components are notified of changes and update their behavior accordingly.

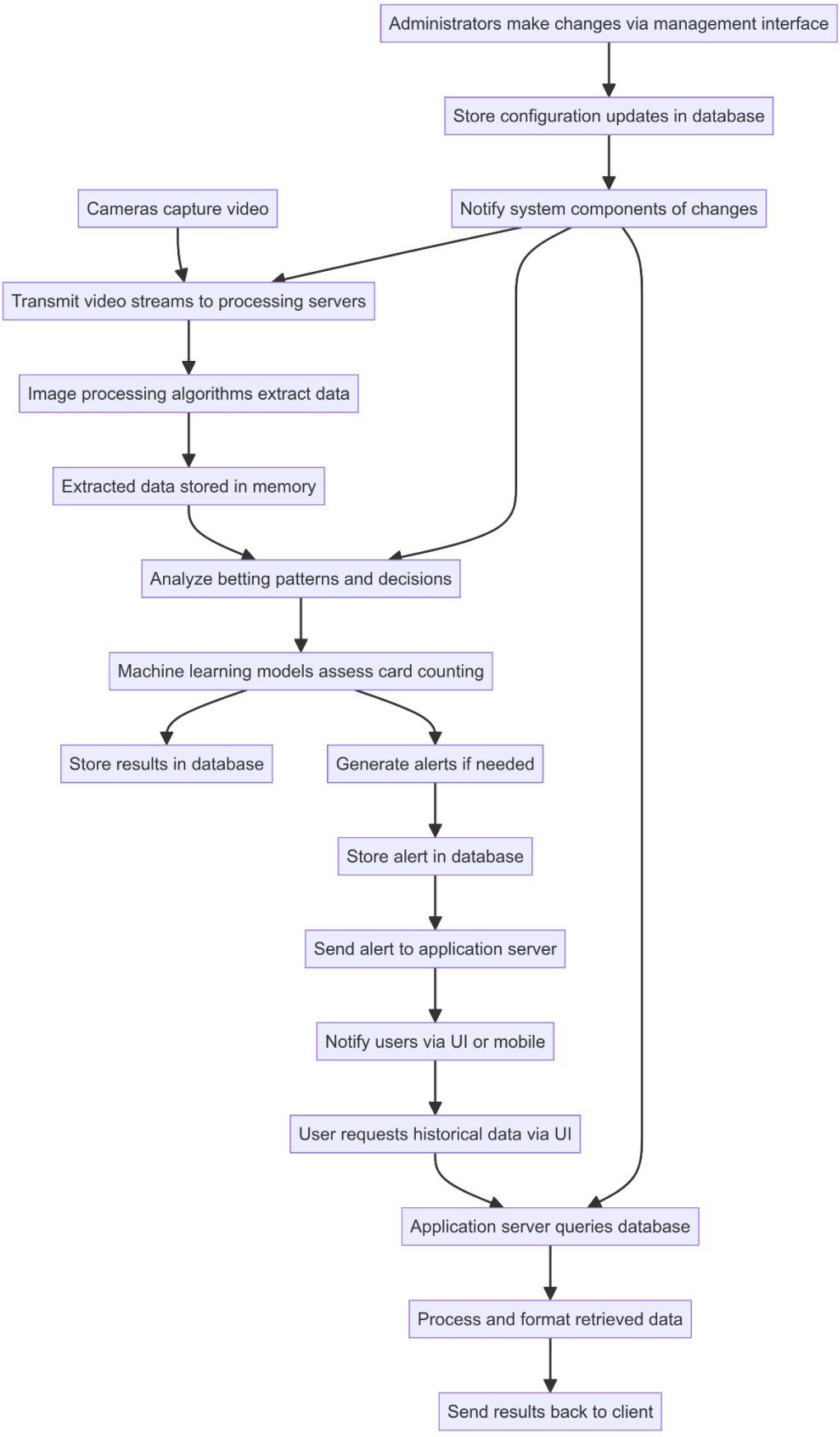

Figure 2: System Data Flow Diagram

5. System Design

5.1. **Data Design** - Database Description

Jackoscope utilizes a relational database management system (RDBMS) to store and manage data efficiently. The database schema is designed to support the system's requirements for real-time processing, historical analysis, and system management. Here are the key tables and their relationships.

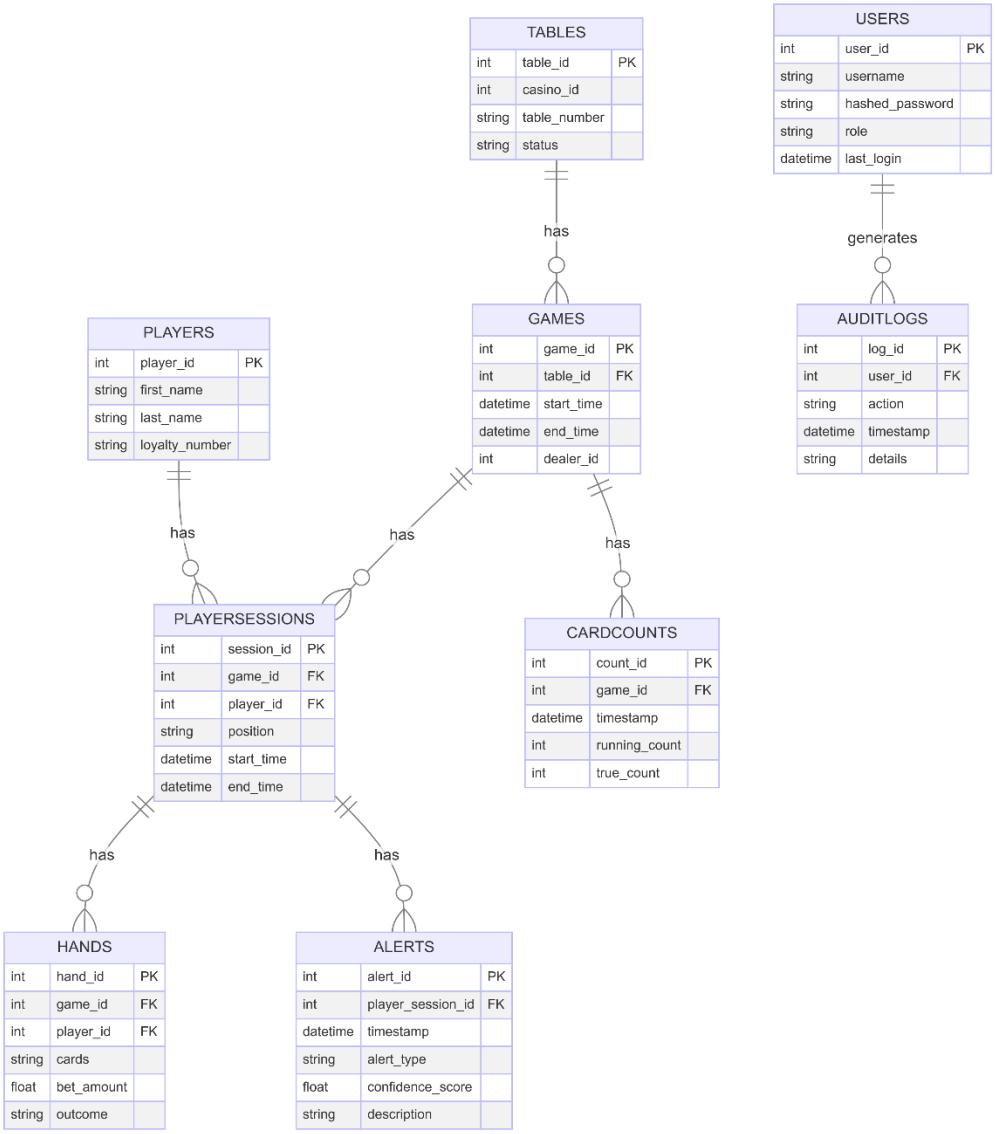

Figure 3: Database Diagram

5.2. **Structural Design - Class Diagram**

The Jackoscope system is implemented using an object-oriented approach.

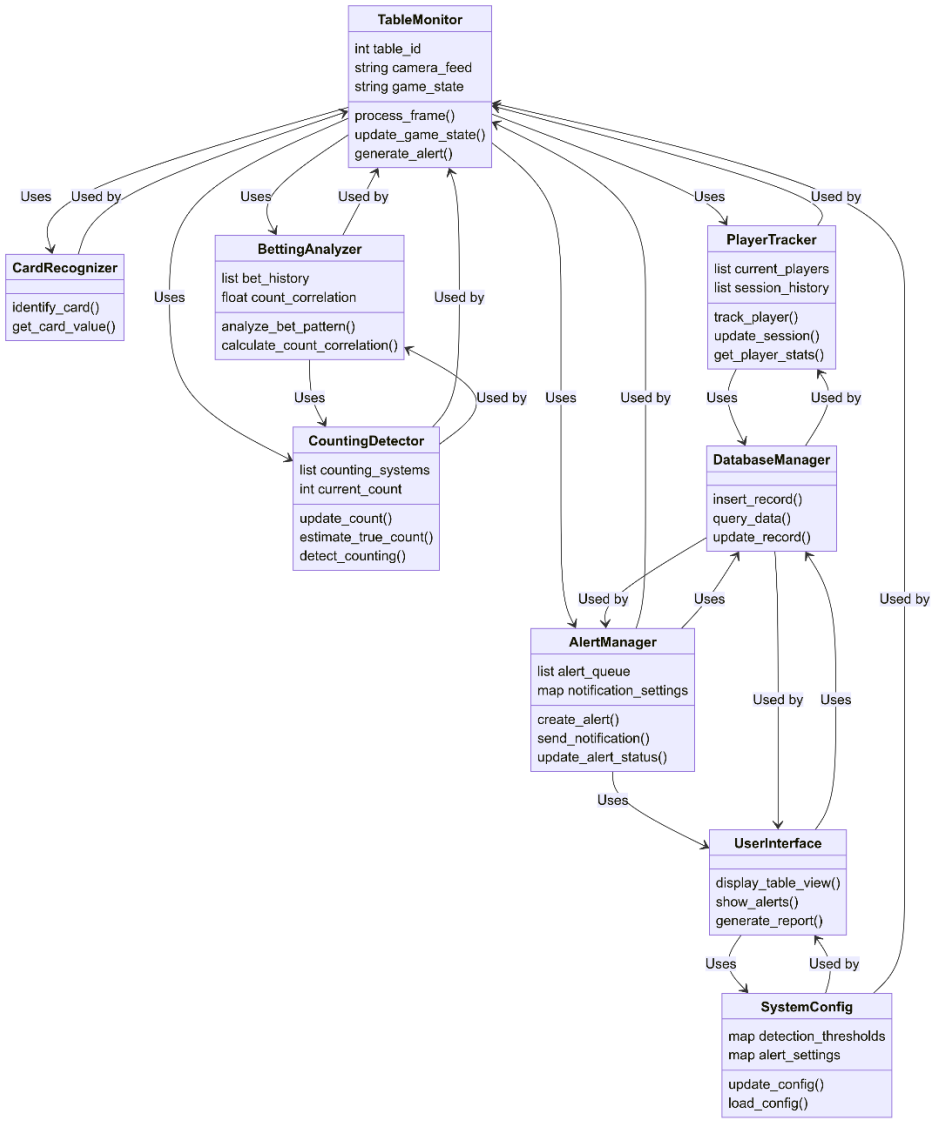

Figure 4: Class Diagram

5.3. Interactions Design

- **Use Cases**

The system's use cases encompass key interactions between users and the system:

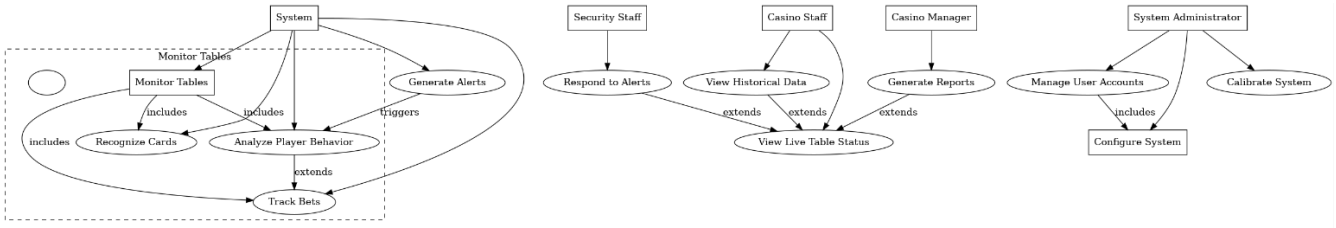

Figure 5: Use Case Diagram

- **Sequence Diagram**

To illustrate the dynamic behavior of the Jackoscope system, we'll focus on a key scenario: detecting and responding to suspected card counting. This sequence diagram will show the interactions between various system

components and actors.

This sequence diagram demonstrates the flow of information through the system, from initial card recognition to alert generation and staff response. It highlights the system's real-time processing capabilities and the interaction between automated components and human operators

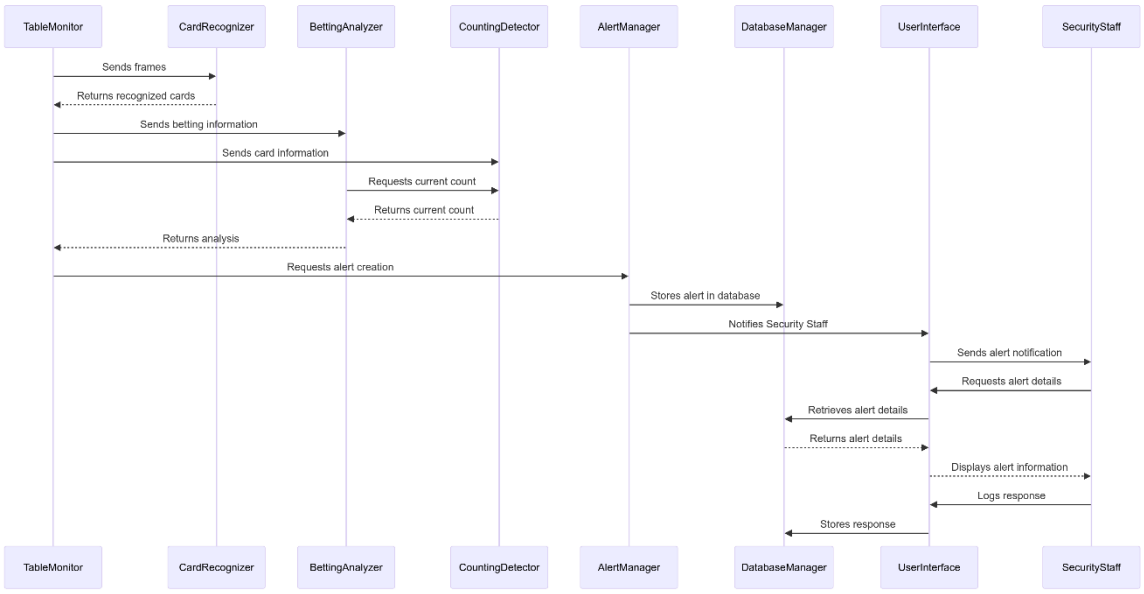

Figure 6: Sequence Diagram

5.4. **Description of Algorithmic Components**

Jackoscope relies on several key algorithms to perform its core functions. Here's a detailed look at the most crucial algorithmic components:

- **Card Recognition Algorithm**

  The card recognition algorithm is a critical component of Jackoscope, responsible for identifying cards from video frames. It employs a combination of computer vision techniques:

a) Image Processing:

- Convert frame to grayscale
- Apply gaussian blur to reduce noise
- Use adaptive thresholding to handle varying lighting conditions

b) Contour Detection:

- Find contours in the pre-processed image
- Filter contours based on area and aspect ratio to identify potential cards

c) Perspective Transform:

- For each potential card, perform a four-point perspective transform to obtain a top-down view

d) Card Classification:

- Extract the rank and suit regions from the transformed card image

- Use a pre-trained Convolutional Neural Network (CNN) to classify the rank and suit
- The CNN is trained on a large dataset of card images to ensure robust recognition under various conditions

- **Betting Analysis Algorithm**

  This algorithm analyses a player's betting behaviour to detect patterns that may indicate card counting:

a) Bet Sizing:
- Track the minimum and maximum bets for each player
- Calculate the bet spread (ratio of max to min bet)

b) Bet Correlation:
- Maintain a running count based on the Hi-Lo system
- Calculate the correlation coefficient between bet sizes and the running count

c) Trend Analysis:
- Use linear regression to identify trends in bet sizing over time
- Detect sudden changes in betting strategy

- **Card Counting Detection Algorithm**

  This algorithm estimates the count that a potential card counter might be using:

a) Multi-System Tracking:
- Implement multiple card counting systems (e.g., Hi-Lo, KO, Omega II)
- Update counts for each system based on cards played

b) True Count Estimation:
- Estimate the number of decks remaining in the shoe
- Calculate the true count for each system (running count/decks remaining)

c) Player Behaviour Correlation:
- Compare player's actions (betting, insurance, deviation from basic strategy) with optimal play for each counting system

These algorithms work in concert to provide a comprehensive card counting detection system. They balance accuracy with computational efficiency to enable real-time monitoring of multiple tables simultaneously.

6. Implementation

6.1. Development Environment

The development of Jackoscope utilized the following tools and technologies:

1. Version Control:
   - Git for version control
   - GitHub for repository hosting and collaboration
2. Backend Development:
   - Python 3.10
   - Flask web framework
3. Frontend Development:
   - React.js for building the user interface
   - Bootstrap for styling
4. Database:
   - SQLite for data storage
5. Computer Vision:
   - OpenCV for image processing and card detection
6. Hardware:
   - Raspberry Pi for video capture and processing
   - USB webcam for capturing video feed

6.2. Card Detection Algorithm Implementation

The implementation uses OpenCV for image processing and custom functions from the 'Cards' module for card detection and recognition. The algorithm preprocesses the image, finds card-shaped contours, and then attempts to recognize the rank and suit of each detected card.

```python
def preprocess_image(image, method='original'):
    """Returns a thresholded camera image."""
    if len(image.shape) == 3:
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    else:
        gray = image

    blur = cv2.GaussianBlur(gray, (5, 5), 0)

    if method == 'original':
        # Original method
        img_h, img_w = gray.shape[:2]
        bkg_level = gray[int(img_h/100)][int(img_w/2)]
        thresh_level = bkg_level + BKG_THRESH
        retval, thresh = cv2.threshold(blur, thresh_level, 255, cv2.THRESH_BINARY)
    elif method == 'adaptive':
        # Adaptive thresholding
        thresh = cv2.adaptiveThreshold(blur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
    elif method == 'otsu':
        # Otsu's thresholding
        retval, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    else:
        raise ValueError(f"Unknown thresholding method: {method}")
```

Figure 7: Pre-process Image Algorithm

6.3. User Interface

The Jackoscope user interface is implemented using React and Bootstrap, providing

a clean and functional layout for monitoring card games. The interface consists of several key components:

- Video Feed Display: The main feature of the interface is a large video feed display area. This shows the live feed from the selected camera when active.
- Statistics Card: This component displays real-time statistics, including:
  The current card count
  A card history section showing recently detected cards (rank and suit)
- Video Control: A simple control panel allows users to start and stop the video feed with a single button.
- Thresholding Method Selection: Users can choose between different thresholding methods for card detection:
  Original
  Adaptive
  Otsu – This allows for adjustment of the card detection algorithm based on different lighting conditions or table setups.
- Camera Selection: A dropdown menu allows users to switch between available cameras. If no cameras are available, it displays a message to indicate.

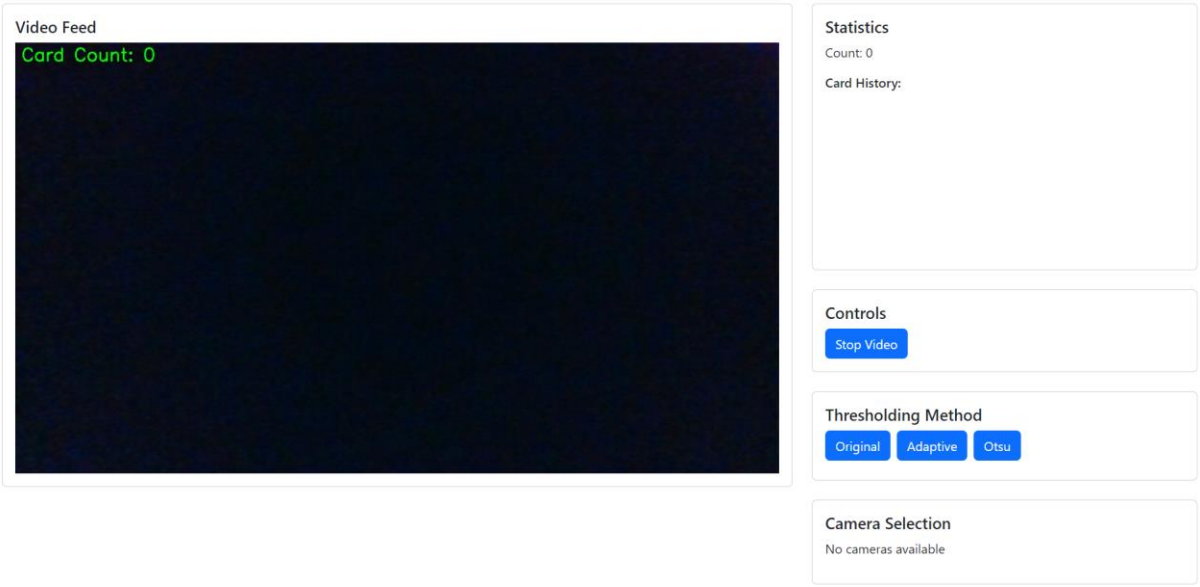

Figure 7: User Interface

The layout is responsive, with the video feed taking up most of the screen on larger displays, while the control panels and statistics are arranged in a column on the right side.
This design ensures that all critical information and controls are visible immediately.

The use of React allows for real-time updates of the statistics and card history without needing to refresh the page. The interface fetches updated statistics every second, ensuring that the displayed information is always current.

Overall, the user interface provides a straightforward and efficient way for casino staff to monitor games, adjust detection settings, and view real-time statistics on card counting activities.

6.4. Backend Implementation

The backend of Jackoscope is implemented using Flask, a lightweight WSGI web application framework in Python.

Here's an overview of the key components:

- **Main Application (app.py):**

  This script sets up the flask application, initializes logging, and creates a global CardDetector instance.

```python
app = Flask(__name__, static_folder='../card-recognition-frontend/build', static_url_path='')
logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger(__name__)

card_detector = None

def initialize_card_detector(camera_index=0):
    global card_detector
    try:
        card_detector = CardDetector(camera_index)
        logger.info(f"CardDetector initialized with camera index {camera_index}")
    except Exception as e:
        logger.error(f"Failed to initialize CardDetector: {str(e)}")
        card_detector = None
```

Figure 9: Main Backend Application

- **API Endpoints:**

  The Backend provides several API endpoints to support the frontend functionality.

  These endpoints handle video streaming, statistics retrieval, video control,

thresholding method selection, and camera switching.

```python
@app.route('/')
def serve():
    return send_from_directory(app.static_folder, 'index.html')

@app.route('/api/video_feed')
def video_feed():
    return Response(gen_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/api/get_stats')
def get_stats():
    return jsonify({
        'count': card_count,
        'history': [card for card in card_detector.card_history if card[0] != "Unknown"
                    and card[1] != "Unknown"] if card_detector else []
    })

@app.route('/api/start_video')
def start_video():
    global video_on
    video_on = True
    return jsonify({"status": "Video started"})

@app.route('/api/stop_video')
def stop_video():
    global video_on
    video_on = False
    return jsonify({"status": "Video stopped"})

@app.route('/video_status')
def video_status():
    global video_on
    return jsonify({"status": video_on})
```

Figure 10: Backend API Endpoints

6.5. Integration

The integration between the frontend and backend is achieved through API calls. The React frontend makes HTTP requests to the Flask backend to fetch data and control the system.

Here are some key integration points:

- Video Feed: The video feed is embedded in the React component using an img tag with the src attribute pointing to the '/api/video_feed' endpoint.
- Statistics Updates: The frontend polls the '/api/get_stats' endpoint every second to update the card count and history.
- Video Control: The start/stop video button triggers calls to 'api/start_video' or '/api/stop_video' endpoints.
- Thresholding Method: Clicking on a thresholding method button sends a request to '/api/set_thresh_method/<method>'.
- Camera Selection: Changing the camera in the dropdown menu sends a POST request to '/api/set_camera' with the new camera index.

6.6. Error Handling and Logging

The system implements basic error handling and logging:

Backend Logging: The Flask application uses Python's logging module to log

information and errors. Log messages are created for important events such as CardDetector initialization and camera switching.

Frontend Console Logging: The React application logs errors and warnings to the browser console, particularly for API request failures and when no cameras are available.

Error Responses: API endpoints return appropriate HTTP status codes and error messages when operations fail, which the frontend can then handle and display to the user if necessary.

7. Validation

7.1. **Testing Methodology**

The testing of Jackoscope was conducted in stages to ensure each component functioned correctly both in isolation and as part of the integrated system. The testing process included:

a) Unit Testing:

Individual functions in the CardDetector class were tested with known inputs to verify correct outputs.

React components were tested in isolation to ensure they render correctly and handle state changes appropriately.

b) Integration Testing:

The interaction between the Flask backend and React frontend was tested to ensure proper data flow and API functionality.

The card detection algorithm was tested with the video feed to verify real-time processing capabilities.

c) System Testing:

End-to-end tests were conducted to simulate real-world usage scenarios.

Performance testing was done to ensure the system could handle continuous video processing without significant lag.

7.2. **Validation Results**

- Card Detection Accuracy:
  a) The System was teste with a variety of playing cards under different lighting conditions – though the most important factor was that the cards lay on a stark black surface.
  b) Results showed that the card detection algorithm could accurately identify cards in most cases, with some challenges in low-light conditions or with partially obscured cards.
- Real-time Processing:
  a) The system demonstrated the ability to process video feeds in real-time, with minimal latency between card placement and detection.
- User Interface Responsiveness:
  a) The React-based user interface proved responsive, updating statistics and video feed controls without noticeable delay.
- Thresholding Methods:
  a) Testing of different thresholding methods (Original, Adaptive, Otsu) showed varying levels of effectiveness depending on the lighting conditions.
- Camera Switching:

a) The system successfully detected available cameras and allowed switching between them, although some inconsistencies were noted in camera enumeration across different systems.

7.3. **Challenges and Limitations**

- Lighting Sensitivity:
  a) The card detection algorithm showed sensitivity to lighting conditions, with accuracy decreasing in suboptimal lighting.
- Processing Power Requirements:
  a) Real-time video processing proved computationally intensive, potentially limiting the number of tables that can be monitored simultaneously on a single system.
- False Positives:
  a) In some cases, the system incorrectly identified objects as cards, highlighting the need for further refinement of the detection algorithm.
- Camera Compatibility:
  a) Not all webcams were equally compatible with the system, with some requiring additional configuration or driver updates.
- Machine Learning:
  a) The model created using a public dataset did not prove accurate enough to use in the system.
  b) To create an accurate enough model, a more extensive data set would be required as well as perhaps additional learning supervision.

8. Summary, Evaluation and Conclusion

8.1. Summary of Achievements

The Jackoscope project successfully implemented a basic card detection and monitoring system with the following key achievements:

- Real-time card detection from video feeds using computer vision techniques.
- A user-friendly web interface for monitoring card games and controlling the system.
- Implementation of multiple thresholding methods to adapt to different lighting conditions.
- Integration of camera selection capabilities for flexible deployment.
- Basic card counting functionality to assist in detecting potential advantage play.

8.2. Evaluation of System Performance

The system demonstrated promising capabilities in controlled environments:

- Card Detection: Showed good accuracy under optimal lighting conditions.
- User Interface: Provided a clean, intuitive interface for monitoring and control.
- Real-time Processing: Achieved near real-time performance on standard hardware.
- Flexibility: The ability to switch cameras and adjust thresholding methods added valuable adaptability.

However, the system also revealed areas for improvement:

- Robustness: Performance varied with lighting conditions and card orientations.
- Scalability: The current implementation may face challenges in monitoring multiple tables simultaneously.
- Accuracy: While basic card counting was implemented, more sophisticated detection of advantage play techniques could be developed.

8.3. Future Work and Improvements

Based on the current implementation and identified limitations, several areas for future work emerge:

- Enhanced Card Recognition:
  - Implement machine learning models for more robust card recognition across various conditions.
  - Develop techniques to handle partially obscured or overlapping cards.
- Advanced Player Behavior Analysis:

    - Implement more sophisticated algorithms for detecting various advantage play techniques beyond basic card counting.
    - Develop pattern recognition for player betting behaviors.
- System Optimization:
    - Optimize image processing algorithms for improved performance on less powerful hardware.
    - Explore distributed processing techniques for scalability to multiple tables.
- User Interface Enhancements:
    - Develop more detailed statistical analysis and reporting features.
    - Implement customizable alert thresholds and notification systems.
- Integration Capabilities:
    - Develop APIs for integration with existing casino management and security systems.

8.4. **Conclusion**

The Jackoscope project has laid a foundation for a computer vision-based card game monitoring system. While the current implementation successfully demonstrates core functionalities such as card detection and basic counting, it also highlights the complexities involved in creating a robust, casino-grade surveillance system.

Moving forward, the insights gained from this project provide a valuable roadmap for future development. With further refinement and expansion of its capabilities, Jackoscope has the potential to evolve into a powerful tool for ensuring fair play and security in casino card games.

The project not only serves its primary purpose of game monitoring but also provides a practical exploration of integrating various technologies - from image processing to web development. This interdisciplinary approach offers rich learning opportunities and paves the way for innovative applications in gaming security and beyond.

9. References

- OpenCV Documentation. https://docs.opencv.org/
  Used extensively for image processing and card detection algorithms.
- Flask Documentation. https://flask.palletsprojects.com/
  Reference for building the backend web server.
- React Documentation. https://reactjs.org/docs/
  Guide for developing the frontend user interface.
- Bootstrap Documentation. https://getbootstrap.com/docs/
  Used for styling and responsive design of the user interface.
- Python Documentation. https://docs.python.org/3/
  Reference for Python programming language used in backend development.
- JavaScript Documentation. https://developer.mozilla.org/en-US/docs/Web/JavaScript
  Reference for JavaScript used in frontend development.
- Git Documentation. https://git-scm.com/doc
  Used for version control throughout the project.
- SQLite Documentation. https://www.sqlite.org/docs.html
  Reference for the database used in the project.
- Raspberry Pi Documentation. https://www.raspberrypi.org/documentation/
  Used for setting up and configuring the Raspberry Pi for video capture.
- "Blackjack Card Counting & Advanced Strategy" by Dan Pronovost, 2019.
  Provided background information on card counting technique.
- OpenCV-Playing-Card-Detector https://github.com/EdjeElectronics/OpenCV-Playing-Card-Detector
  Open-source card detection project – provided a basis for Jackoscope.

10.Appendices

Appendix A: Code Repository

The Complete source code for the Jackoscope project is available in the following GitHub repository:

https://github.com/meandean17/Jackoscope-Ver2

Appendix B: Setup and Installation Guide

- Backend Setup:
  a) Install Python 3.10 or later
  b) Install requires Python packages:
     pip install flask opencv-python numpy
  c) Set up the webcam or USB camera (preferable)
- Frontend Setup:
  a) Install Node.js and npm
  b) Navigate to the frontend directory
  c) Install dependencies:
     npm install
  d) Built the frontend:
     npm run build
- Running the Application:
  a) Start the Flask server:
     python app.py
  b) Access the application through a web browser at http://localhost:5000