Received: 14 April 2023 | Revised: 15 May 2023 | Accepted: 26 May 2023 | Published online: 14 June 2023

RESEARCH ARTICLE

Artificial Intelligence and Applications
2023, Vol. 00(00) 1–8
DOI: 10.47852/bonviewAIA3202962

BON VIEW PUBLISHING

Playing Blackjack Using Computer Vision

Anas Akkar[1], Samuel Cregan[1], Justin Cassens[1], Maame Araba Vander-Pallen[1] and Tauheed Khan Mohd[1,*]

[1]*Department of Math Computer Science, Augustana College, USA*

Abstract: The field of computer vision (CV) is rapidly evolving, with a focus on analyzing, manipulating, and understanding images at a sophisticated level. The primary objective of this discipline is to interpret the visual input from cameras and utilize this knowledge to manage computer or robotic systems or to generate more informative and visually appealing images. The potential applications of CV are wide-ranging and include video surveillance, biometrics, automotive, photography, movie production, web search, medicine, augmented reality gaming, novel user interfaces, and many others. This paper outlines how CV technology will be utilized to achieve a winning outcome in the game of Blackjack. The game of Blackjack has long captivated the attention of enthusiasts and players worldwide. One area of particular interest is the development of a winning strategy that maximizes the player's chances of success. With the advent of sophisticated computer algorithms and machine learning techniques, there is enormous potential for research in this area. This paper explores the game-winning strategies for Blackjack, with a particular focus on utilizing advanced analytical methods to identify optimal plays. By analyzing large data sets and leveraging the power of predictive modeling, we aim to create a robust and reliable framework for achieving consistent success in this popular casino game. We believe that this research avenue holds enormous promise for unlocking new insights into the game of Blackjack and developing a more comprehensive understanding of its intricacies.

Keywords: OpenCV, Blackjack, computer vision, strategy, gaming

1. Introduction

Wouldn't it be very convenient to use your phone while playing Blackjack online and instantly know whether to hit, stand, or double? Wouldn't it also be convenient to use your phone during a Chess game and instantly know what your best move is? You could use the internet or simulations to help yourself in these situations. However, what if we could use our smartphone camera and have it detect the best next move for either Chess or Blackjack. That would be a more optimal solution and it is possible using OpenCV. OpenCV (Open Source Computer Vision Library) is a computer vision (CV) and machine learning software library built to give a structure for CV applications (Culjak et al., 2012). We can use this library to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, and recognize scenery. OpenCV has provided many Application Programming Interface and was made compatible with mostly every device. It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS.

Accurately identifying objects opens the door for many possibilities for a computer program, and developers can use this technology to work on projects of interest to them. A shared interest of this group of programmers is an interest in card games. From poker classes to family game nights, there is a background in cards and an appreciation for the impact that a card game can have on strengthening relationships. For this reason, the general objective of this project is to develop a program that can assist in card game playing and learning (Goyal et al., 2017).

For any project of a significant size such as this, it is necessary to break it into smaller chunks of work to be done. A basic flowchart of our project's goals is shown below in Chart I. This outlines the order of the steps that must be completed in order to complete the project. Table I provides specific information about each objective. As with any software project, the development team must be agile and quickly adapt to changing demands and goals, so these objectives are structured as general guidelines that are subject to change. However, these objectives provide a good outline for what the project will entail (Kadir et al., 2014).

2. Related Work

OpenCV began as a research project at Intel in 1998 and has been publicly available since 2000 (Opencv, 2020). It provides programmers with tools to use CV to develop further programs. These tools are a "mix of low-level image-processing functions and high-level algorithms such as face detection, pedestrian detection, feature matching, and tracking" (Pulli et al., 2012). Using these basic tools, programmers can expand and develop their own code to complete tasks that are of specific interest to them.

Currently, there are a few different methods that OpenCV can use for object recognition. These include but are not limited to:

*Corresponding author: Tauheed Khan Mohd, Department of Math Computer Science, Augustana College, USA. Email: tauheedkhanmohd@augustana.edu

template matching, color-based matching, and shape-based recognition (Object Recognition: What is It and How Does It Work? 2021). Template matching is the most basic strategy and employs already known templates of objects and compares them with objects seen by OpenCV (Gollapudi, 2019). Color-based matching is also a fairly simple method; it uses colors on the RGB scale to allow a program to identify an object based on color criteria. Shape-based recognition goes hand in hand with template matching and uses known shapes for comparison to identify an object in question. Recognition is also broken down into the type of image a program is identifying. Active recognition refers to a program detecting an object in a live image, while passive recognition means that the program is assessing objects in a still image (Dutta, 2019).

One such project using OpenCV was a project conducted in 2013 on traffic sign recognition (Geronimo et al., 2013). A group of graduate students aimed to use CV to detect and analyze a road sign from a camera located in a moving vehicle. Software such as this has been implemented for speed limit signs in particular in high-end vehicles as early as 2009 (Geronimo et al., 2013) but had not been refined to account for all sorts of traffic signs, variable weather conditions, lighting, and other challenges. The strategy for recognition used for this project was most likely very similar to a card identifying project; there are a finite number of traffic signs that the software could "match" an identified traffic sign to, just as there are a finite number of cards that can be matched. This project likely had the additional challenge of tracking a road sign in a moving vehicle, which may impair the clarity of the picture that the camera is able to capture. This is just one example of a CV-based project that has been explored.

One group of developers used CV to identify playing cards on a table for a poker game. They were able to identify the cards and count chips on a table with up to 94% accuracy (Martins & Reis, 2011). The group outlined a series of steps in their work. First, they needed to "extract" the card from the playing surface. This was done by the contrast of the playing card to the table that it rested on. Additionally, they did work on identifying chips using a similar strategy of color-based recognition. Next, the group used "template matching" to identify the cards. Since a deck of playing cards has a finite number of possibilities, the program can use the known possibilities to identify the card that it sees. This project ended at this level however, leaving much room for future work to be done on the topic in terms of developing algorithms for specific card games (Soo, 2014).

Advancements in CV have led the way for further development of artificial intelligence (AI) and machine learning technology. CV bridges the gap between code and reality, allowing programs to have "human-like recognition ability" (Ward, 2021). Just as machine learning is the latest buzzword and next big thing in the software world, CV is the next big step for machine learning. In the previously cited article, researchers outlined how the medical field can implement CV to allow machine learning programs to help with surgery. In reference to the benefit that CV provided to the project, they state: "Without sight, AI was operating blindly, and its procedural understanding was inflexible and limited" (Ward, 2021). Effective CV can drastically improve machine learning programs and allow them to not only learn much quicker but to be used in greater applications as well. This sets the limits of what CV can do for AI extremely high; there are many possibilities that have yet to be discovered.

Guennouni et al. (2014) give an overview of the cascade object identification technique and how the cascade classifier uses Haar-like feature selection. The paper then presents the results of a performance comparison between a normal platform and an embedded device using an OpenCV-based solution for multiple object detection.

Jalled and Voronkov (2016) use this paper to create an OpenCV-Python program that uses the Haar cascade technique to detect objects and faces. Unmanned aerial vehicles (UAVs) are now utilized to detect and attack infiltrating ground targets. The fundamental disadvantage of this sort of UAV is that the item is not always accurately detected, resulting in the object colliding with the UAV. The project's goal is to prevent UAV collisions and damage. The Viola–Jones algorithm is used in UAV surveillance to recognize and track individuals. The train function is used to train the algorithm, which uses the cascade object detector function and vision. The key benefit of this code is that it takes less time to process. The Python code was tested using a video and image database, and the results were verified.

Druzhkov et al. (2011) consider the object detection issue, which is one of the most hotly debated subjects in CV right now. The issue can be summarized as follows: "where the instances of a specific object class are in the image?" The main goal of this paper was to create a high-performance, open-source implementation of two state-of-the-art object recognition and general machine learning algorithms: discriminatively trained component-based models and gradient boosting trees, respectively. Real-world applications were used to test the accuracy of the implemented algorithms. One of the most prominent open source CV libraries has incorporated the implementation.

Chandan et al. (2018) demonstrate how the single-shot detector (SSD) algorithm detects objects in real-time circumstances. They present a real-time study of the ecosystem that may help any business achieve exceptional results by enabling security, order, and utility. They then show how the model may be used in closed-circuit television, drones, and other surveillance equipment to identify attacks in settings where guns are prohibited, such as schools, government offices, and hospitals.

Saxena et al. (2019) address the use of a Haar cascade classifier. The case study of a face detection and item detection, such as watch detection and pen detection, is the major topic of this paper. The designed system's purpose is to make life easier and more convenient and enhance the mundane aspects of our life. It is an effort to build one's own Haar classifier using OpenCV.

Jakubović and Velagić (2018) use brute-force matchers to solve a challenge of feature matching and object recognition in two photos. For feature recognition and descriptor extraction, the suggested system included many concurrent algorithms, including ORB (Oriented FAST and Rotated BRIEF), BRISK (Binary Robust Invariant Scalable Keypoints), SIFT (Scale Invariant Feature Transform), and SURF (Scale Invariant Feature Transform) (Speeded-Up Robust Features).

Markuš et al. (2013) provide a method for detecting visual objects that is based on an ensemble of optimal decision trees grouped in a cascade of rejectors. The trees employ pixel intensity comparisons in their internal nodes, allowing them to analyze picture areas quickly. A face detection issue is used for experimental analysis. The findings obtained are encouraging and show that the procedure has practical relevance. They also examine its noise sensitivity and demonstrate how to accomplish quick rotation invariant object recognition.

Ditrih et al. (2021) present an approach for continuous finding and recognition of cards from the game set using CV technologies. Their approach of detecting cards in photographs is divided into three stages: fragmenting cards from the picture, separating highlights from card pictures using level and vertical lines, and card

inclusion order with the use of a vector machine. To provide consistent treatment, a small number of highlights are deleted from a small selection of pixels.

Mehmood et al. (2019) intend to direct execution exploration of deep learning-based calculation, for example, SSD, in IoT-based implanted gadgets for intelligent home apparatuses control (Le et al., 2022). We have created a clever home computerization framework based on object discovery calculation in light of model view regulator engineering sent on Cloud of Things such as Amazon Web Service cloud for customers to review their houses from a distance. For communication with connected IoT devices, the message lining telemetry transport (MQTT) protocol is used. We introduced the concept of a circulating intermediary to assist a large number of distributers and endorsers in load-adjusting.

Sharma et al. (2021) addressed topics ranging from how artificial awareness and AI computations aid in object recognition to how OpenCV is a very useful tool for newcomers who want to find out how continuous article differentiating proof and following should be achievable. It also demonstrates the flexibility of a global positioning framework to a moving camera, which is suitable for vehicle safety applications. Image distinguishing proof employs techniques such as article placement, acknowledgement, and division. The use of artificial consciousness and AI accelerates the processing of information while maintaining the standard of the outcome. As an example, by applying AI, we can surely do difficult tasks.

Culjak et al. (2012) show and quickly make a reader familiar with the guts and bolts of OpenCV (Open Source PC Vision) without going through the large instructional guides and books. OpenCV is an open source toolkit for image and video analysis that was first presented by Intel more than a decade ago. Since then, a number of developers have contributed to the most recent library enhancements.

3. Proposed Methodology

Today, CV is widely used everywhere, "both in academia and industry" (Minichino & Howse, 2015). It can reach consumers in many contexts via webcams, camera phones, or even gaming sensors. It also is one of many recent technological advances that have helped to pave the way forward for fully autonomous vehicles. In our project, we are trying to develop a program that can identify game cards and help playing and learning. OpenCV Python can help us explore solutions to these requirements in a high-level language. We are also looking forward to developing an environment that links Python, OpenCV, depth camera libraries (OpenNI, SensorKinect), and general-purpose scientific libraries (NumPy, SciPy). The main card game that we are trying to implement our program on is Blackjack.

Blackjack is played with a standard deck of 52 cards. Every card has a value equal to its number, with face cards worth 10 and Aces can be worth 1 or 11. The goal is to get the sum of your cards as close to 21 as possible without going over. Players are dealt two cards initially, and the dealer is dealt two cards. The player's cards are both visible. One of the dealer's cards is visible, and one is hidden. The player must decide to hit, stand, or double down. If the player hits, they are dealt another card. If the sum of their cards ever goes above 21, the player is bust and loses the game. If the player stands, it is then the dealer's turn. The dealer reveals their hidden card and will take hits until the dealer's hand is 17 or greater. The dealer always hits if their total is below 17, and always stays if their total is 17 and up. If the dealer goes above 21, the player wins. Also, whoever has the hand closest to 21

Figure 1
Blackjack basic strategy for hard totals (https://www.blackjackapprenticeship.com/blackjack-strategy-charts/).
Blackjack strategy card

HARD TOTALS										
DEALER UPCARD										
	2	3	4	5	6	7	8	9	10	A
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	D
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
8	H	H	H	H	H	H	H	H	H	H

KEY		
	H	Hit
	S	Stand
	D	Double if allowed, otherwise hit

without going over is the winner. The rule for doubling down is as follows: Players are allowed to double down only on their first turn and only if their first two cards are summing to 9, 10, or 11. If a player doubles down on their first turn, their wager is doubled, and they receive a hit card. Play then resumes as normal. In the short time we have, it would be extremely difficult to implement an algorithm that counts cards and is the most likely to win. Consequently, instead, we are going to implement a strategy called "basic strategy" that consists basically of playing like the dealer, with occasional changes and doubling down depending on the situation. The details of the basic strategy are shown in Figure 1 (Blackjack Strategy Charts – How to Play Perfect Blackjack, 2021).

The "basic strategy" is separated based on whether or not the player's hand is "hard" or "soft." A "soft" hand is a hand that uses the Ace as an 11. So, for example, an Ace + 6 is a "soft" 17 hand. The strategies of play become different if a player's hand is "hard" or "soft." For this project, we will not need to make this distinction; we consider all totals to be "hard" totals (Baldwin et al., 1956).

Now, in order to be able to implement this strategy, we need our computer to be able to identify the cards. An existing technique mentioned in the Literature Review that can help us identify which of the 52 cards are we looking at is "template matching" (Brunelli, 2009). Template matching is a method for identifying a template image in a larger image and OpenCV already has a function that serves this purpose.

The way OpenCV allows this function is by simply sliding the template image over the input image and comparing them. Several comparison methods are implemented in OpenCV (Bradski, 2000).

We need two primary components:

- Source image (I): The image in which we expect to find a match to the template image
- Template image (T): The image which will be compared to the source image.

Our goal is to detect the highest matching area. To identify the matching area, we have to compare the template image against the source image by sliding it. By sliding, we mean moving the patch one pixel at a time. At each location, a standard is calculated so it represents how "good" or "bad" the match at that location is. For each location of T over I, you store the standard value in a result

Figure 2
The matching methods available in OpenCV (https://medium.com/analytics-vidhya/opencv-object-detection-using-template-matching-methods-63ac15d74742). Template matching methods

a. **method=CV_TM_SQDIFF**

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

b. **method=CV_TM_SQDIFF_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

c. **method=CV_TM_CCORR**

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

d. **method=CV_TM_CCORR_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

e. **method=CV_TM_CCOEFF**

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

f. **method=CV_TM_CCOEFF_NORMED**

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

matrix R. We are to use the OpenCV function matchTemplate() to search for matches between an image and an input image. It implements template matching in the function matchTemplate(). The available methods are 6 shown in Figure 2 below (Prasad, 2020).

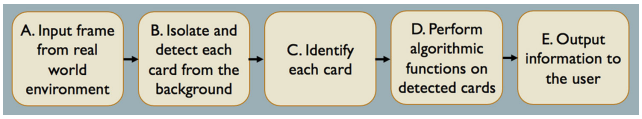
The procedure followed while coding will be as follows:

- Declare some global variables, such as the image, template and result matrices, as well as the match method.
- Load the source image and template
- Perform the template matching operation.
- Normalize the results
- Localize the minimum and maximum values in the result matrix R by using minMaxLoc() function.
- For the first two methods in Figure 2, the best matches are the lowest values. For all the others, higher values represent better matches.
- Display the source image and the result matrix.
- Determine whether there is a match between the two compared images.

4. Framework

At its core level, our program takes an input of a real-world environment, turns this into data, and outputs information to the user. Using the OpenCV library (Opencv, 2020), the program can view the real-world environment through a camera and can manipulate what it finds to compute data to be useful to the user. A visualization of the actions the program performs is shown in Figure 3.

Figure 3
Visual depiction of program flow. Program flow chart



4.1. Input frame from real-world environment

This is the raw input that the program takes. This will be a frame that comes from the video feed that the user will define once the desired cards are focused in the frame. The program will allow the user to move the camera and focus the desired cards, and wait for a user "button press" to evaluate the desired frame. At this point, the program only sees exactly what the camera sees.

4.2. Isolate and detect each card from the background

Once the user has sent the program the desired frame to evaluate, the program will begin to decipher what it is given in the image. The first step the program takes is to isolate each card from the background noise in the frame. The program does this by color detection on the hue, lightness, and saturation scale. It isolates the color white on the frame, assuming that the cards will

be white. Then, the program identifies the borders of the detected white space to create shapes of every border it detects. Using the information from these borders, then program then runs a series of tests to determine whether the border detected is a card or not. These checks are based around the shape, area, and length of the sides of the detected shape. The program then creates new images of only each individual card detected and stores them into an array.

4.3. Identify each card

The next step in the program is to identify and categorize each card that is detected. This is done using the strategy of template matching (citation here). Template matching is a CV strategy that identifies things that are seen by the computer by comparing them to already known images that are already given to the program. In this programs case, the program has every card already loaded in as a reference. For every card that is detected, the program will compare with the existing images in its database to determine the best match.

OpenCV template matching works by individually matching each pixel on one image to the other. The code will then provide a number of how well the imported image matches with each one of the cards in the programs database. Then, it takes the highest number as the best match.

4.4. Perform algorithmic functions of detected cards

The algorithm written in this program is specific to Blackjack. Once the user has scanned both the players cards and the dealers cards, this information is sent to the algorithm to determine what the users smartest move is. The inputs for the algorithm are the players two cards and the dealers one visible card. The output is the players action in string form. The algorithm is written based off of mathematically determined Blackjack odds, which are shown in Figure 1.

4.5. Output information to the user

After the algorithm determines what the user should do, the program will display this information to the user on the screen. For Blackjack, the two options that a player has are to "hit" or "stand." There is a special case where the user can double down when they are dealt two cards of the same value. In this case, the program will treat both of the players hands as if they were separate hands and perform all of the previously mentioned steps again.

In summary, the framework of Blackjack is also known as Twenty-One, which is a popular card game played in casinos and online gambling websites. The game is played with a standard deck of 52 cards, and the objective of the game is to beat the dealer by having a hand with a total value closer to 21 than the dealer's hand, without exceeding 21.

The game starts with the dealer shuffling the deck of cards and then dealing two cards to each player, including themselves. The dealer's first card is dealt face down, while the second card is dealt face up.

Each player then takes turns to make their move, starting from the player on the dealer's left. The player has several options to choose from, including:

1. Hit: Take another card from the dealer to try and get closer to 21.
2. Stand: Keep the current hand and end their turn.
3. Double: Double their original bet and take one more card from the dealer.
4. Split: If the player has two cards of the same rank, they can split them into two separate hands, with each hand having its own bet. The player can then play each hand separately.
5. Surrender: Some casinos allow players to surrender their hand and forfeit half of their bet.

The value of the cards in Blackjack is as follows:

- Cards 2–10 are worth their face value.
- Face cards (Jacks, Queens, and Kings) are worth 10 points each.
- An Ace can be worth 1 or 11 points, depending on which value would be more beneficial for the player's hand.

If the player's hand exceeds 21 points, they are said to "bust" and lose the game. If the player chooses to stand, the dealer then reveals their face-down card and continues to draw cards until they have a total of at least 17 points. If the dealer busts, all remaining players win. If the dealer does not bust, the hands of each remaining player are compared to the dealer's hand, and whoever has the hand closest to 21 without exceeding it wins.

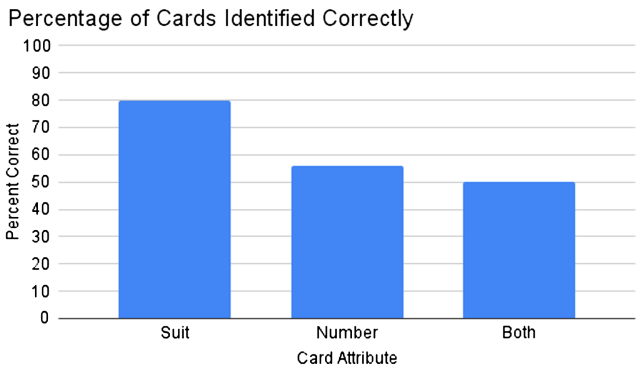
Overall, the game of Blackjack is a fun and exciting game that requires both luck and skill. With a basic understanding of the rules and strategies of the game, players can increase their chances of winning and have a great time at the table.

5. Methods

When it comes to taking data, we focused only on the accuracy of the identification of cards. Our algorithm for move feedback is independent of this identification and relies solely on logic, so we did not feel it needed to be included. With this in mind, our strategy for taking data was to go through every card and test the accuracy of our program. We had every card placed in approximately the same location with similar lighting to try and simulate more consistency. The particular variables we measured were the accuracy of the suit recognition as well as the card value recognition. The percentage of successes has been displayed in Figure 4.

There are a few main causes of failed identifications which we noticed while testing. The lesser of these issues was poor lighting. Our identification methods implement the RGB scale to isolate a range of colors representing white which then allows for recognition of a card object. However, if the lighting is too dark, then the shaded part of the card will not be recognized and cause issues. If the lighting is too bright, then the card will have glares

Figure 4
Results from card identification. Percentage of cards identified correctly



when trying to compare to the baseline images which leads to poor comparisons.

Additionally, cards with less detail struggled to identify correctly. If this was the only problem to occur in a trial, while the card as a whole did not get identified correctly a majority of the time, the suit still tended to be correctly identified. This is because our program takes in the cards from the camera feed, isolates them, and for each looks for comparisons to the baseline images. However, for cards with less detail like a two of hearts, the heart will be a major identifier for that card and will be found many more times in a card such as the ten of hearts. This in turn will lead to higher matches even though it is with the wrong card. This is a bug we are still not positive how to reduce the occurrence of.

Lastly, an issue we have had is with our camera gaining proper focus on the cards. When the focus is good, the cards identify correctly a majority of the time, even with the previous two issues occurring. This is because the poor focus leads to a blurred image and so the value portion of the card is more easily mistaken by our program. Also, if the focus is poor, then the suits also had a lower rate of success.

6. Results

The idea of object recognition is one which has been explored and expanded upon already in many ways as mentioned in the Literature Review section. Our project on object recognition, particularly the recognition of playing cards, is one which can be further developed into applications to help teach people card games while providing the numbers behind the logic. Applications similar to this already exist in a 2-dimensional world on a computer screen, but it has yet to make the jump to 3-dimensional space. This is the way that our idea differs from these current programs. We will expand them from a 2-dimensional computer application and implement them in a 3-dimensional scenario. This means that people in real-world situations could utilize the application to complete the same task as the 2-dimensional computer program allowing for immediate feedback, but through offline means. Some may then think that there are already physical objects which require no online service which can advise you throughout different card games such as a chart with all the possible blackjack combinations, telling you what the best play is in any given scenario. This is also true. However, why have an extra object to worry about keeping track of when you could just have a phone application which weighs nothing and is part of an object you would have with you anyways? Additionally, this program eventually could have the ability to implement many various algorithms to work for different games whereas a physical object tends to be for just one game and only works for games with limited possibilities. Games like Texas Hold'Em have too many hand possibilities to fit onto one card or physical object, especially when taking into account the community cards. In this way, a flexible application is more useful than any physical object or online program.

The possibilities and results of a game of Blackjack can vary depending on various factors, such as the specific rules of the game being played, the number of decks being used, the skill level of the players, and the strategies employed by the players. However, there are some general outcomes and possibilities that apply to most games of Blackjack:

1. Winning with a Blackjack: If a player's first two cards are an Ace and a 10-point card (10, Jack, Queen, or King), they have a "Blackjack" and win the game automatically, unless the dealer also has a Blackjack. In this case, the game is a push (tie), and the player's bet is returned.
2. Winning by having a higher hand than the dealer: If the player's hand is closer to 21 than the dealer's hand, without exceeding 21, the player wins the game. The payout for winning is usually 1:1, meaning the player wins an amount equal to their bet.
3. Losing by busting: If the player's hand exceeds 21 points, they "bust" and lose the game automatically, regardless of the dealer's hand.
4. Losing by having a lower hand than the dealer: If the player's hand is worth less than the dealer's hand, the player loses the game.
5. Push (tie): If the player's hand has the same value as the dealer's hand, the game is a push (tie), and the player's bet is returned.
6. Insurance: Some Blackjack games offer the player the option of taking insurance if the dealer's face-up card is an Ace. This is a side bet that pays 2:1 if the dealer has a Blackjack, but the player loses their main bet.

The possibilities of winning in Blackjack can be improved by using basic strategy, which involves making decisions based on the player's hand and the dealer's up card. However, even with basic strategy, there is always an element of luck involved in the game.

7. Future Work

The work done in this project really highlights the vast potential for CV and OpenCV. Specifically, it shows how CV could be used for various applications with card games. Once solid code is developed to identify a playing card, any game imaginable can be written into code to help aid or even teach card games. For example, this project shows that the card game algorithm written specifically for Blackjack is correct 100% of the time when the playing cards are identified correctly. This goes to show that any other game logic can be written into an algorithm that will also be correct 100% of the time. Therefore, the only work that needs to be done to improve an application of this kind is on the CV side to improve the accuracy of identifying cards. The research and data collected in this experiment show that even with low budget equipment under variable light conditions, OpenCV can be used to identify the suits of cards with up to 80% accuracy and identify the number of the card correctly over 50% of the time. While this may seem low, further checks can be implemented to improve the code to get it closer to the 100% accuracy mark (Younis et al., 2020).

Developing algorithms for specific card games only requires knowledge of the card game and research on the rules of the game. In this instance, we used well-known and mathematically determined Blackjack odds for determining what the user's correct move should be (see Figure 1). This information was obtained by research online. While luck obviously plays a part in the real-world playing of card games, every card game has similar information available, as its core card games are all based on mathematical odds (Othman et al., 2018).

CV has much room for improvement, and as the field progresses more and more methods will become available that can be used to more accurately identify cards (Khan et al., 2019).

8. Discussion

In addition to the ideas we hope to implement into our program with recognizing playing cards based on their rank and suit, there are a few more ideas which we will not be able to implement, but believe could be achieved with more time and through a similar strategy.

One of these additional ideas was that we thought about utilizing the object recognition from an aerial view in an outdoor parking lot to maintain count of available spots and even provide the location of these spots so someone entering the lot could be given accurate locations as to where they can park and directions to get there (if it is a large lot). This could be implemented in a few ways, either by identifying vehicles and keeping track of their locations relative to the parking spaces or by identifying the parking spaces and tracking whenever something is obstructing the spot. An issue with the latter would be if say an animal or piece of garbage were sensed and then an open spot would be marked as occupied. Thus, the first implementation would seem the better choice, but that would also bring different problems. There are many different vehicle types from SUVs to trucks to motorcycles which all utilize the same parking spaces. So this program would have to be able to recognize all of them which seems a strenuous task on its own.

Another example of potential use would be in poker. Typically in high level poker tournaments which are televised, the cards of each player are known to the commentators and viewers. This is done by having the players place their cards face down on the table which contains cameras inside of it and then the information is told to someone who inserts the data into the required system to have it show up on the TV screen. With our program, the cards could be automatically recognized and the correct information input directly onto the screen without having to worry about any middleman (Leveille, 2014).

To expand even further with the poker and touch on a point made in the Intellectual Merit section, this project could expand out to other card games as well. For poker, the algorithm would be difficult in terms of advice for future moves. But if it simply is used for calculating odds, then this idea would work. All one would need is to scan their own cards and then also the community cards as they are provided throughout each hand, and if the cards are being correctly identified, then odds of potential hands could be quickly predicted. However, this in itself may not be entirely useful as it is only giving you information on your own hand. If you also knew an opponent's cards, say in an all-in-and-call scenario where you want to know what your outs are, then this program would be capable of providing that live information for you.

Additionally, this type of program can expand from just playing cards to other types of cards, potentially containing numbers and letters. It could help children to learn how to recognize numbers and letters with the addition of verbal speaking to the program. If someone could grab a random card and scan it and immediately get feedback as to what is on the particular card, then this could help with learning at a young age.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

References

Baldwin, R. R., Cantey, W. E., Maisel, H., & McDermott, J. P. (1956). The optimum strategy in blackjack. *Journal of the American Statistical Association*, *51*(275), 429–439.

Blackjack Strategy Charts – How to Play Perfect Blackjack (2021). Retrieved from: https://www.blackjackapprenticeship.com/blackjack-strategy-charts/.

Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(11), 120–123.

Brunelli, R. (2009). *Template matching techniques in computer vision: Theory and practice*. John Wiley & Sons.

Chandan, G., Jain, A., Jain, H., & Mohana (2018). Real time object detection and tracking using deep learning and OpenCV. In *2018 International Conference on Inventive Research in Computing Applications*, IEEE, 1305–1308.

Culjak, I., Abram, D., Pribanic, T., Dzapo, H., & Cifrek, M. (2012). A brief introduction to OpenCV. In *Proceedings of the 35th International Convention MIPRO*, IEEE, 1725–1730.

Ditrih, H., Grgić, S., & Turković, L. (2021). Real-time detection and recognition of cards in the game of set. In *2021 International Symposium ELMAR*, IEEE, 161–164.

Druzhkov, P. N., Erukhimov, V. L., Zolotykh, N. Yu., Kozinov, E. A., Kustikova, V. D., . . . , & Polovinkin, A. N., (2011). New object detection features in the OpenCV library. *Pattern Recognition and Image Analysis*, *21*(3), 384–386.

Dutta, A. (2019). Object detection and facial features identification in python using OpenCV.

Geronimo, D., Serrat, J., Lopez, A. M., & Baldrich, R. (2013). Traffic sign recognition for computer vision project-based learning. *IEEE Transactions on Education*, *56*(3), 364–371. https://doi.org/10.1109/TE.2013.2239997.

Gollapudi, S. (2019). *Learn computer vision using OpenCV*. Springer.

Goyal, K., Agarwal, K., & Kumar, R. (2017). Face detection and tracking: Using OpenCV. In *2017 International Conference of Electronics, Communication and Aerospace Technology*, IEEE, *1*, 474–478.

Guennouni, S., Ahaitouf, A., & Mansouri, A. (2014). Multiple object detection using OpenCV on an embedded platform. In *2014 Third IEEE International Colloquium in Information Science and Technology*, IEEE, 374–377.

Jakubović, A., & Velagić, J. (2018). Image feature matching and object detection using brute-force matchers. In *2018 International Symposium ELMAR*, IEEE, 83–86.

Jalled, F., & Voronkov, I. (2016). Object detection using image processing. *arXiv preprint:1611.07791*.

Kadir, K., Kamaruddin, M. K., Nasir, H., Safie, S. I., & Bakti, Z. A. K. (2014). A comparative study between LBP and Haar-like features for face detection using OpenCV. In *2014 4th International Conference on Engineering Technology and Technopreneurship*, IEEE, 335–339.

Khan, M., Chakraborty, S., Astya, R., & Khepra, S. (2019). Face detection and recognition using OpenCV. In *2019 International Conference on Computing, Communication, and Intelligent Systems*, IEEE, 116–119.

Le, N., Rathour, V. S., Yamazaki, K., Luu, K., & Savvides, M. (2022). Deep reinforcement learning in computer vision: A comprehensive survey. *Artificial Intelligence Review*, *55*, 2733–2819.

Leveille, C. (2014). Facial Tic detection using computer vision.

Markuš, N., Frljak, M., Pandžić, I. S., Ahlberg, J., & Forchheimer, R. (2013). Object detection with pixel intensity comparisons organized in decision trees. *arXiv preprint:1305.4537*.

Martins, P., Reis, L. P., & Teófilo, L. (2011). Poker vision: Playing cards and chips identification based on image processing. In

Pattern Recognition and Image Analysis, 436–443. https://doi.org/10.1007/978-3-642-21257-4_54.

Mehmood, F., Ullah, I., Ahmad, S., & Kim, D. (2019). Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT. *Journal of Ambient Intelligence and Humanized Computing*, 1–17.

Minichino, J., & Howse, J. (2015). *Learning OpenCV 3 computer vision with python: Unleash the power of Computer Vision with python using OpenCV*. Packt Publishing.

Object Recognition: What is It and How Does It Work? (2021). Retrieved from: https://cprimestudios.com/blog/object-recognition-what-it-and-how-does-it-work.

OpenCV (2020). Retrieved from: https://opencv.org.

Othman, N. A., Salur, M. U., Karakose, M., & Aydin, I. (2018). An embedded real-time object detection and measurement of its size. In *2018 International Conference on Artificial Intelligence and Data Processing*, IEEE, 1–4.

Prasad, D. (2020). OpenCV object detection using template matching methods. Retrieved from: https://medium.com/analytics-vidhya/opencv-object-detection-using-template-matching-methods-63ac15d74742.

Pulli, K., Baksheev, A., Kornyakov, K., & Eruhimov, V. (2012). Real-time computer vision with OpenCV. *Communications of the ACM*, *55*(6), 61–69. https://doi.org/10.1145/2184319.2184337.

Saxena, M. R., Pathak, A., Singh, A. P., & Shukla, I. (2019). Real-time object detection using machine learning and OpenCV. *International Journal of Intelligent Systems and Applications*, *11*(1), 0974–225.

Sharma, A., Pathak, J., Prakash, M., & Singh, J. (2021). Object detection using OpenCV and python. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking*, IEEE, 501–505.

Soo, S. (2014). Object detection using Haar-cascade classifier. *Institute of Computer Science, University of Tartu*, *2*(3), 1–12.

Ward, T. M., Mascagni, P., Ban, Y., Rosman, G., Padoy, N., Meireles, O., & Hashimoto, D. A. (2021). Computer vision in surgery. *Surgery*, *169*(5), 1253–1256. https://doi.org/10.1016/j.surg.2020.10.039. Retrieved from: https://www.sciencedirect.com/science/article/pii/S0039606020307492.

Younis, A., Shixin, L., Jn, S., & Hai, Z. (2020). Real-time object detection using pre-trained deep learning models Mobilenet-SSD. In *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, 44–48.

How to Cite: Akkar, A., Cregan, S., Cassens, J., Vander-Pallen, M. A., & Mohd, T. K. (2023). Playing Blackjack Using Computer Vision. *Artificial Intelligence and Applications* https://doi.org/10.47852/bonviewAIA3202962