Playing Card Detection Using OpenCV

Asher Gingerich, Stephen Boxerman

Introduction

The goal of this project overall is to create an artificial intelligence (AI) capable of playing the card game Whist with human players, using physical cards (manipulated by a human on behalf of the AI). Whist is a 4 player card game based on playing through a number of "hands" (rounds of deck distribution), winning "tricks" (playing the best card in a round), to gain points.

The computer vision component of this project is faced with the challenge of locating and identifying cards within an image, to communicate back to the decision making component of the central AI what the cards in its possession are. In solving this problem, we needed to try two approaches, due to shortcomings present within our first method of card identification.

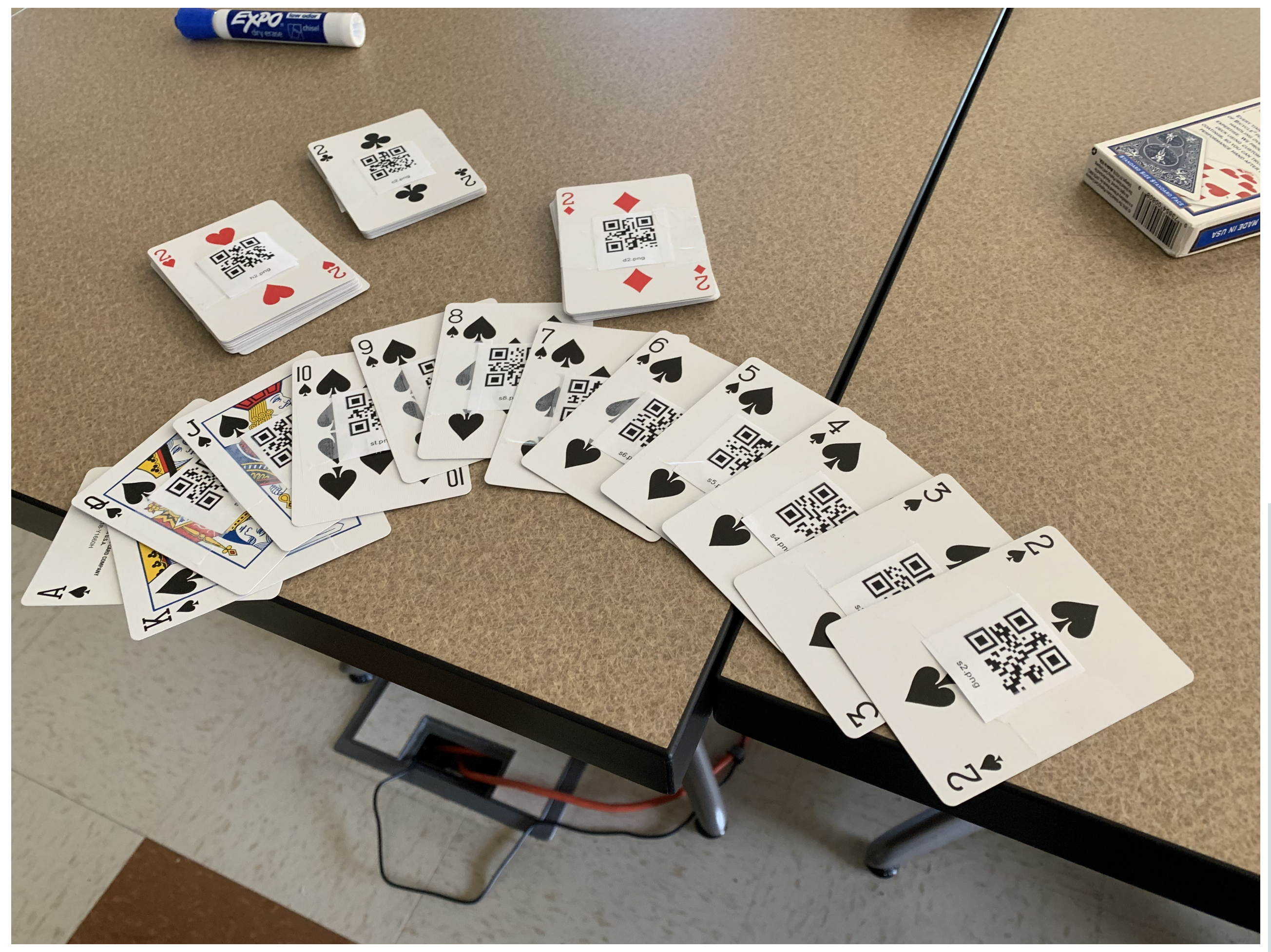
Figure 1: QR codes attached to the cards

Development Tools

This project was developed using OpenCV primarily, using Python as the development language. PyZBar and PyQRCode were used to develop the QR code solution, and ImageMagick was used to handle large scale image manipulation when printing the QR codes, or resizing the images taken with a camera to test the OpenCV card extraction method.

Method 1: QR Codes

The first milestone in this system was to create a simple method of identifying cards within an image. The use of QR technology simplified this task significantly. Through the use of the package PyQRCode, I generated a unique code for each card and, using ImageMagick, created a mosaic of all the QR images. Attaching these to the cards themselves, the system was able to identify, to varying degrees of success dependent on lighting and camera position, the cards present within an image, through the use of the package PyZBar.

This method of card detection allows for very easy flexibility in card rotation, and even allows for a level of card overlap, so long as the QR code itself is not covered. However, the code itself must be rather large to be easily readable by the camera, and the code needs to be present on the cards themselves. This solution, therefore, is not very flexible overall: it really only works for the one card deck that has the images present. The goal of card detection is to be able to identify cards in the realm of normal play, so they must be more resilient to overlapping AND be capable of detecting more than a single deck of cards.

There was another, larger problem with this solution. The card detection using QR is highly susceptible to blurring. It only took a slight amount of loss of focus in the camera and the card was no longer identifiable. When comparing the images in Figure 2, both images are clearly identifiable to the human eye. However, due to 2b's slight blurring and noise from the table, the QR code was not correctly located, thus the missing card 'd7', 7 of diamonds). This indicated that a more thorough solution would be required for use in a game environment.

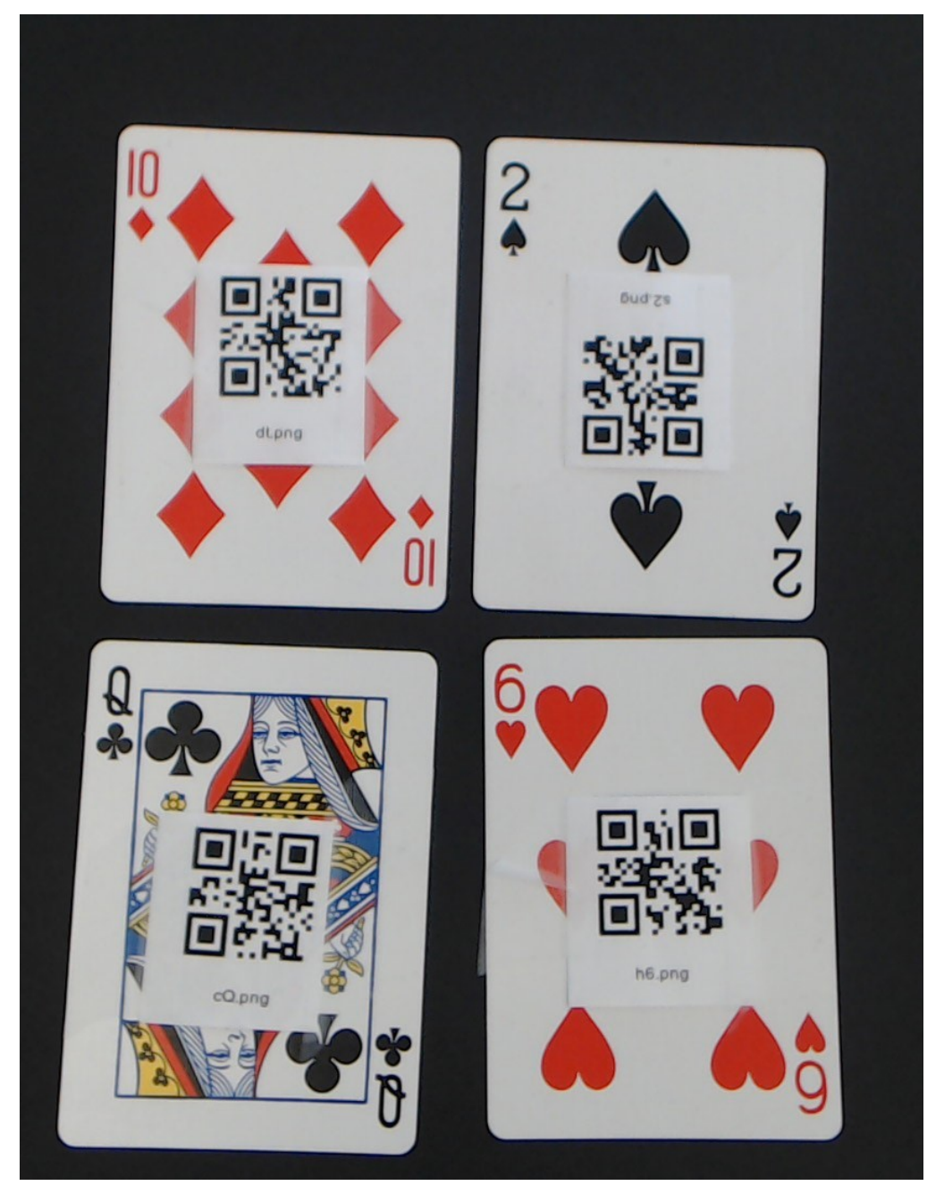
Figure 2a:
Results—['dt', 'h6', 's2', 'cQ']

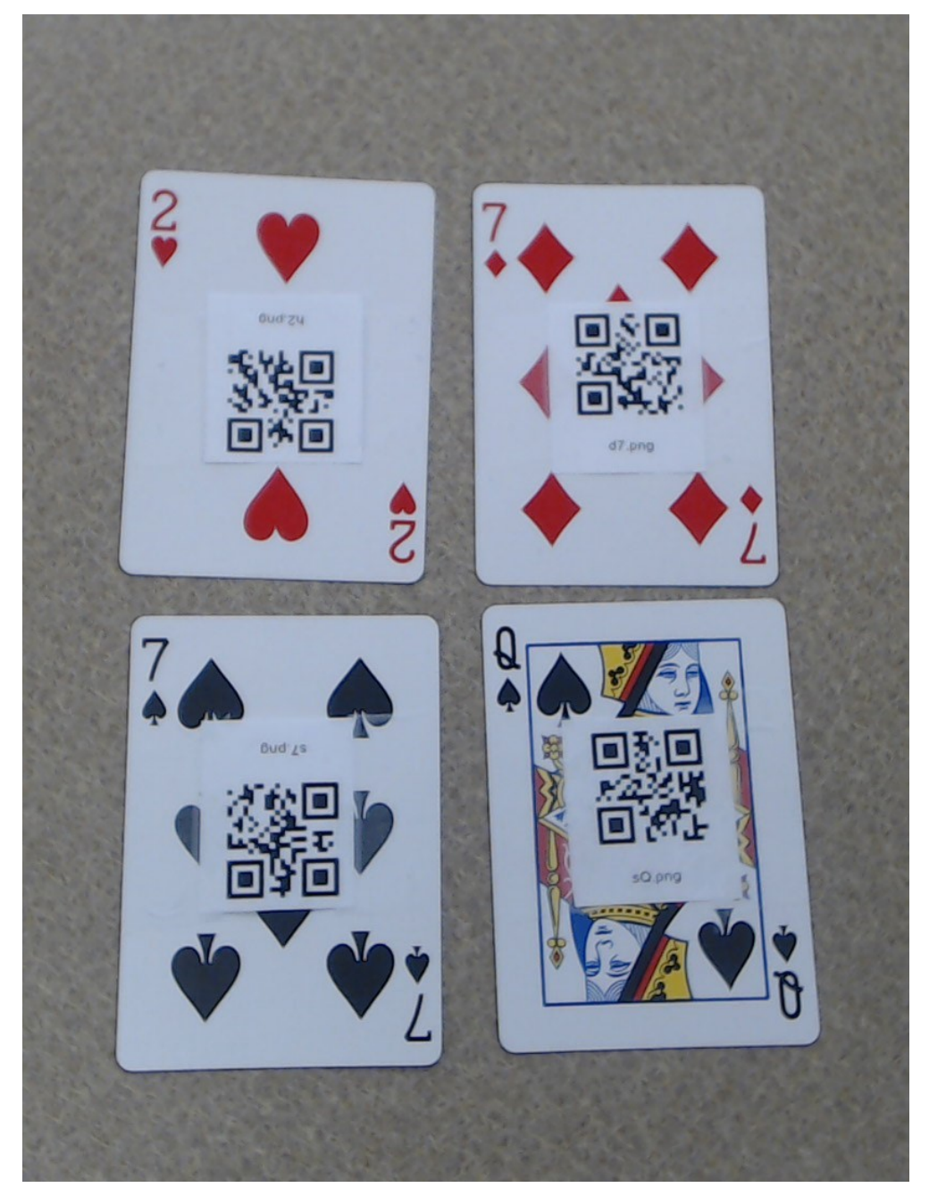
Figure 2b:
Results—['h2', 'sQ', 's7']

Card Extraction Process

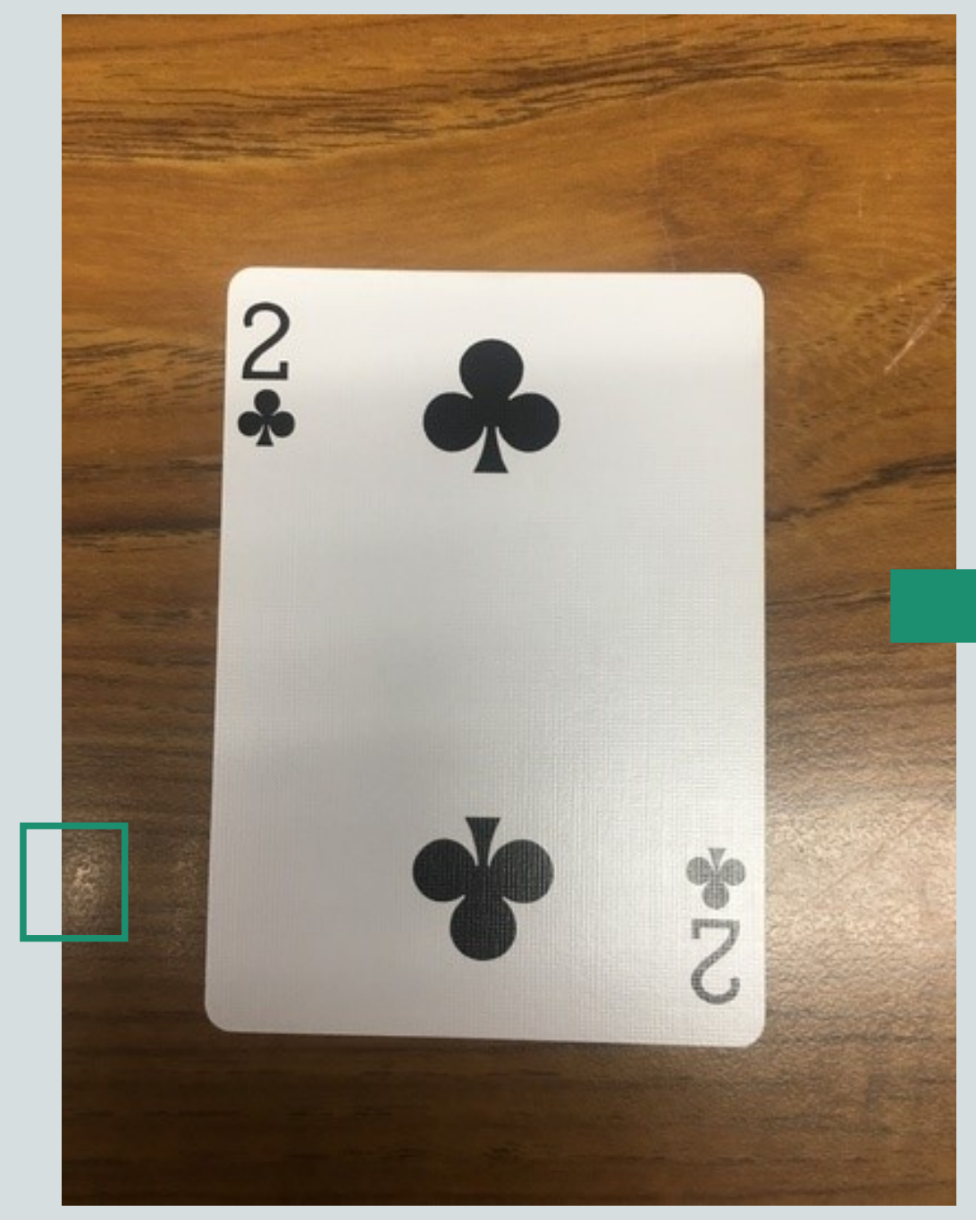
Figure 3a: Base Image

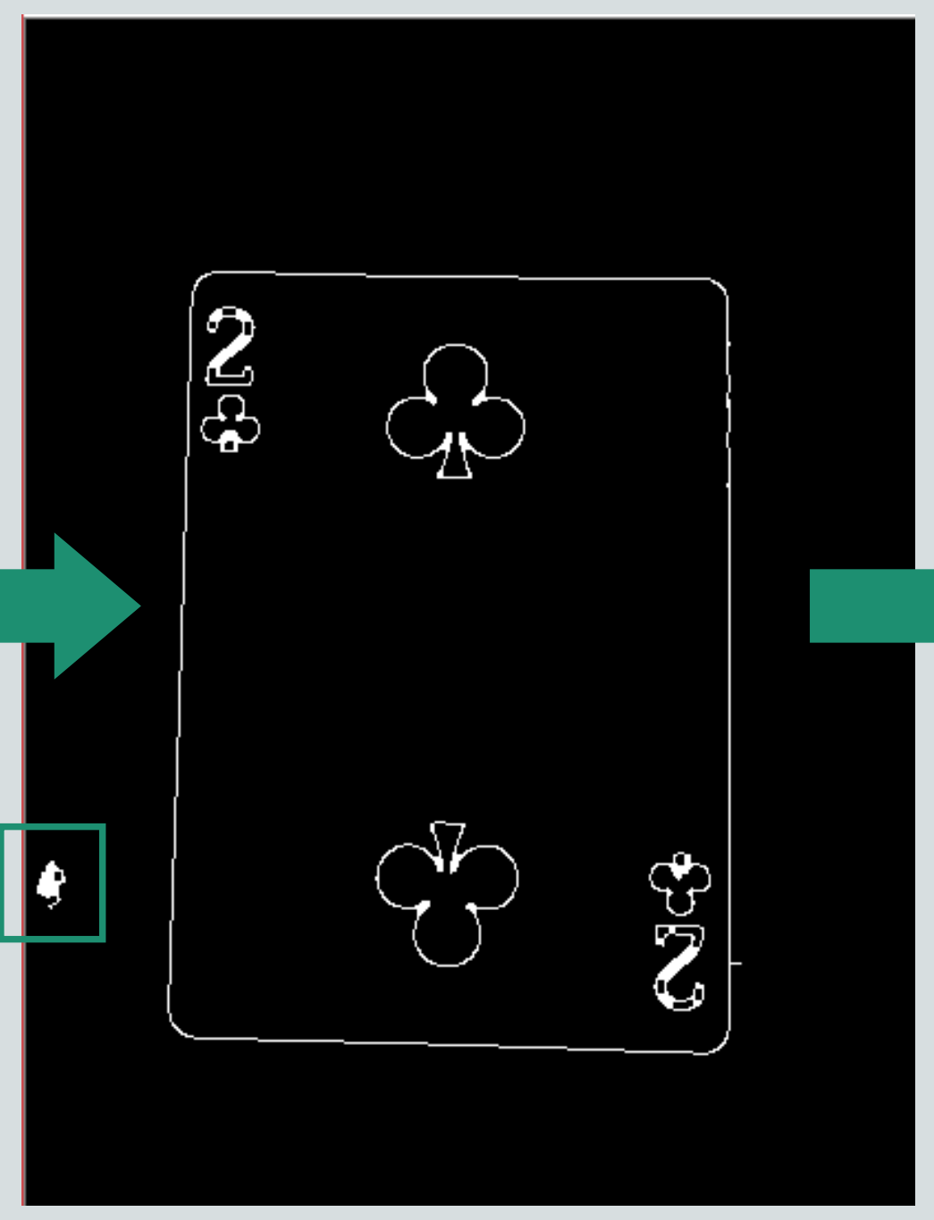
Figure 3b: Edge Detection

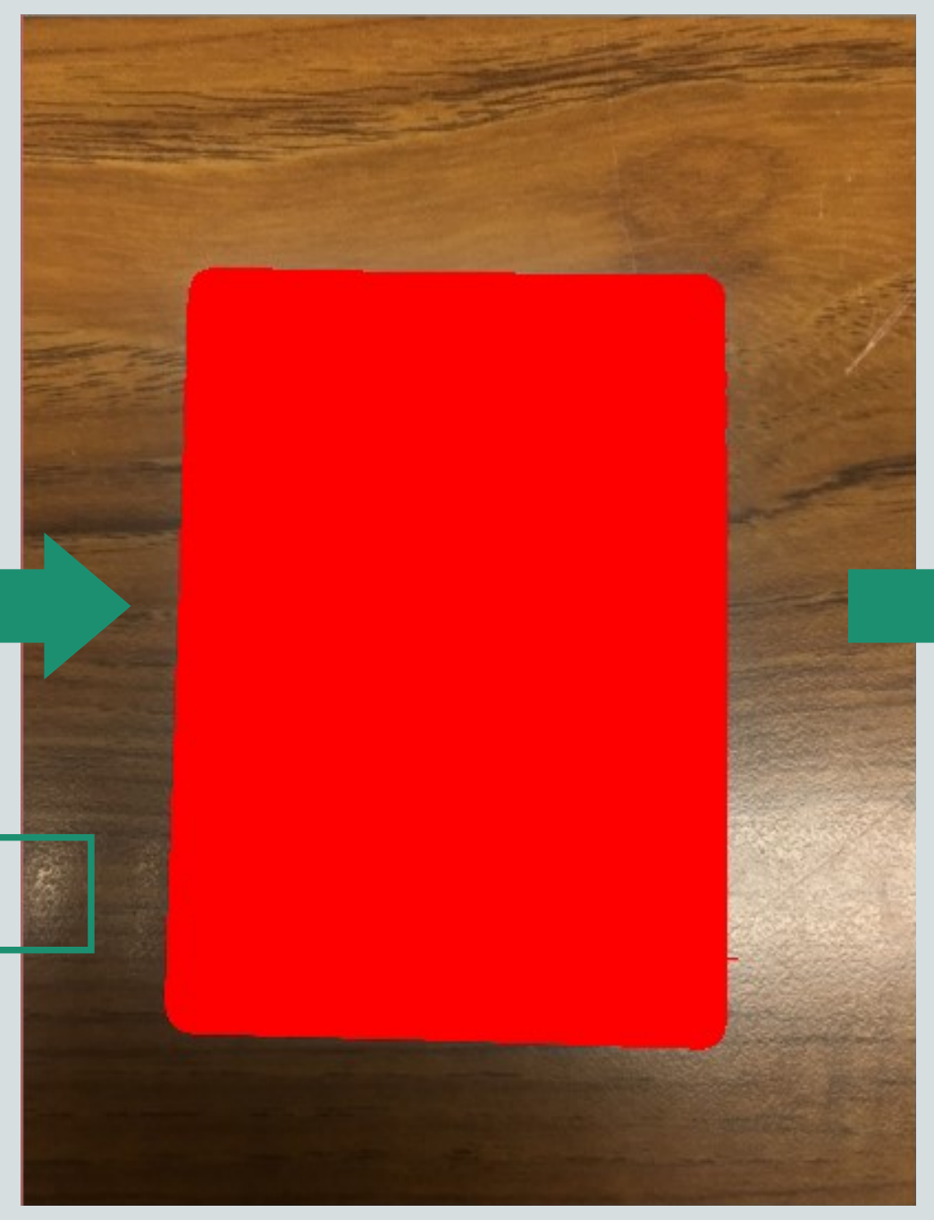
Figure 3c: Detected Card

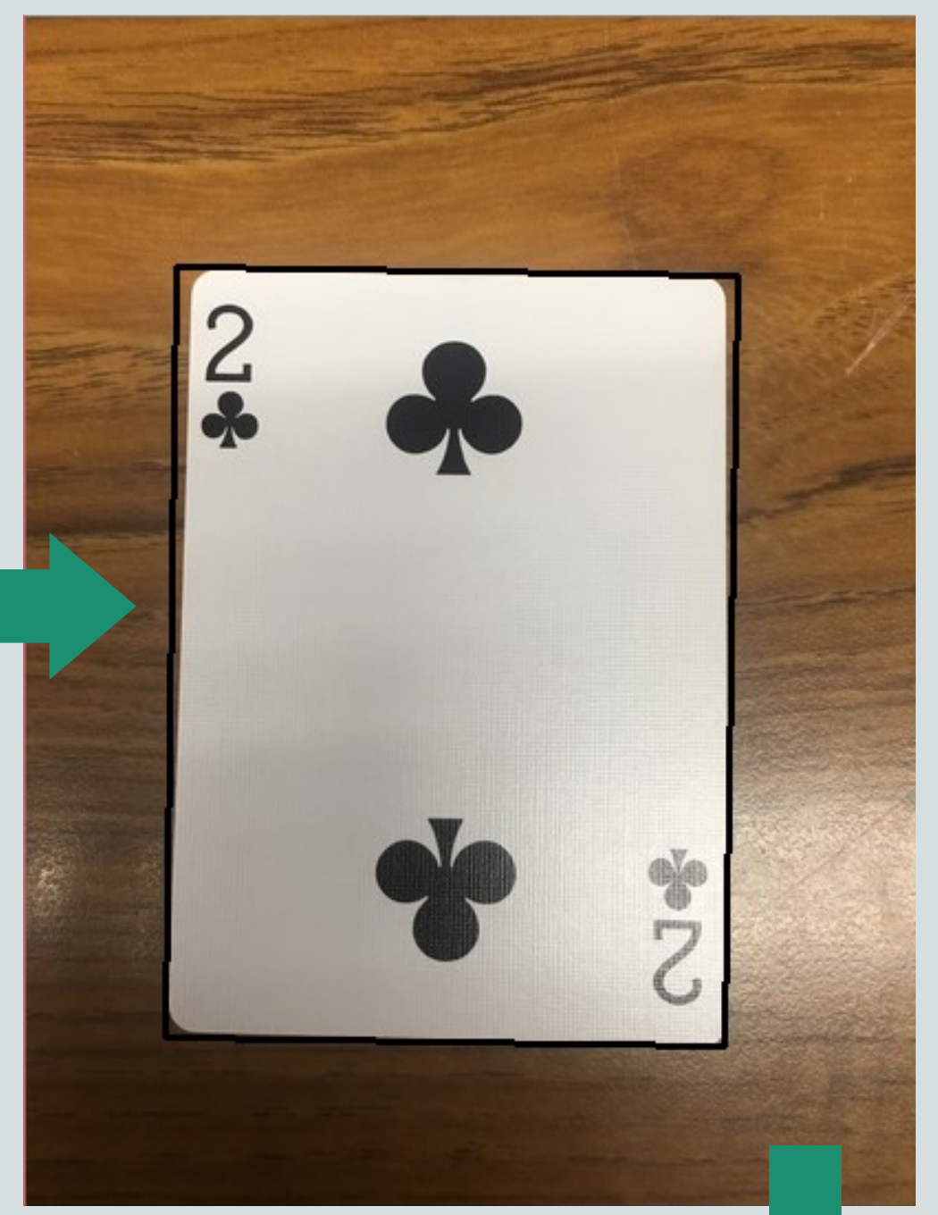
Figure 3d: Isolated Card

These are the steps the card extraction process follows. Given a base image, it detects the edges using Canny edge detection. Note the noise introduced in the lower left by the glare on the table. Then the card is isolated using contour detection, which eliminates the disconnected noise. Furth noise reduction is applied to smooth the edges of the card enough to draw a box as close to the edges of the card as possible.

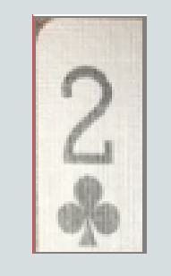
Figure 3e: Card Corner

Method 2: Card Extraction

To extend the capabilities of the card detection and identification, ideally no data outside of the card itself is necessary. OpenCV provides the tools necessary to isolate a card within a given scene, and thereby have a reliable data-capturing method to identify cards.

There is a multi-step process to accomplish this. First, the smoothed, grayscale image of the card is run through an edge-detection algorithm (Figure 3b). For this project, Canny edge-detection was utilized. A combination of noise reduction algorithms and OpenCV contour detection are utilized to isolate the card within the image (Figure 3c).

The outcome of this combination is an image mask that then has a minimum area rectangle created around it to create an easily digestible rectangular image (Figure 3d). The corner of that image is then extracted to obtain the card's corner, which contains the actual data consumed during card identification.

Future Work

For this system to integrate with the AI portion, first a neural network needs to be trained on the data this algorithm pulls from cards. A simpler solution is also possible, though it is not as accurate or flexible as a neural network. This solution would be utilized by comparing each corner extracted to a mask of that corner, and the corner mask it is most similar to would be the selected card.

Then that system needs to be able to retrieve information from a video of live gameplay, which is achievable through retrieving individual frames of the video, and using those images as the card detection. The framerate of that retrieval only needs to be high enough to accurately reflect the given state of the game, it does not need to be entirely in real time.