Software Overview

Year: 2023 **Semester: Spring** **Team: 3** **Project: "Rigged" Card Shuffler**
Creation Date: Jan 25, 2023 **Last Modified:** Feb 26, 2023
Author: Utkarsh Priyam **Email: upriyam@purdue.edu**

Assignment Evaluation:

Item	Score (0-5)	Weight	Points	Notes
Assignment-Specific Items				
Software Overview		x2		
Description of Algorithms		x2		
Description of Data Structures		x2		
Program Flowcharts		x3		
State Machine Diagrams		x3		
Writing-Specific Items				
Spelling and Grammar		x2		
Formatting and Citations		x1		
Figures and Graphs		x2		
Technical Writing Style		x3		
Total Score				

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:
Relevant overall comments about the paper will be included here

1.0 Software Overview

Our product takes in a deck of cards, where the cards are stored in an arbitrary order. Given that deck and a set of configuring constraints on the output deck, such as card position specifications or specific shuffle algorithms, our shuffler scrambles the cards according to the user configurations and outputs the reordered deck. The software behind this functionality has three main components: the microcontroller's hardware management, the RaspberryPi's computer vision card identification, and the RasPi's card sort order determination algorithm.

The micro's hardware control refers to its communication and data exchange with the RasPi, which hosts the computationally intensive brains of the product, as well as the micro's control of hardware used to isolate, sort, and reorder the deck of cards. Primarily, the former entails exchanging byte-sized packets with the RasPi, while the latter involves stepping and spinning motors according to positioning data provided by the RasPi.

The computer vision software involves identifying the cards by capturing and processing images of the cards in the input order. The card identification process works by extracting contours, or edges, from the image. It achieves this using sobel convolutional filters [4] to highlight edges within the image. The contours are then used to compute various points of interest, or key points, by finding positions where contours intersect or turn sharply. Additionally, the contours are used to isolate the keypoints using bounding box [5] analysis on the various components visible on the cards. Finally, the different bounding boxes and key point components are matched against "ground truth" card images via a brute force matching search. The "ground truth" card that most closely matches the features present on the card image is then used to identify the rank and suit of the card image that was captured. For this card matching process, noise and timing considerations are absolutely critical. In particular, noise in the captured images, whether vis-a-vis sensor noise, ambient lighting inconsistencies, or other sources, can significantly affect the quality and efficacy of the identification and matching algorithms. To combat this, we implement a multi-phased denoising solution, using a combination of blurring and blending filters to smoothen noisy fluctuations, contrast-based filtering to define sharp edges over smooth gradient transitions, and monochromatic filtering to reduce the effects of multi-channel color on card identification. From a timing perspective, our current algorithm takes roughly 20 to 50 ms to identify each card, from the moment the image data is captured by the camera sensor to when the response is ready to be sent back to the microcontroller.

Finally, the card sort determination algorithm determines card sort order in one of two modes: card-agnostic and card-sensitive. In card-agnostic mode, the specific cards put into the input do not matter, so the sorting process can bypass scanning and identifying the cards. This case applies to when the user wants to use a standard shuffle algorithm or a randomized permutation; the shuffler simply provides a random or pseudo-random permutation list [7] as the output shuffle. On the other hand, in card-sensitive mode, the specific cards matter, because the user requested that certain cards appear in certain positions in the output deck, typically with the goal of stacking the deck for the user's advantage. In this mode, the sorting algorithm fixes the

positions of specified cards or card patterns, before then randomly filling in any unspecified positions with the remaining available cards.

2.0 Description of Algorithms

Our software system consists of three main components: the microcontroller's hardware management, the RaspberryPi's computer vision card identification, and the RasPi's card sort order determination algorithm.

With respect to the micro's hardware management software, there are no particularly significant algorithms in use. At most, it uses a multiplier and boolean logic to compute the optimal direction and number of steps to activate the card binning mechanism's stepper motor for, in order to store the current card in the bin location specified by the RasPi.

In the computer vision code, there are a few key algorithms which are fused together to provide the overall card identification functionality. Specifically, the software combines brute force matching with key point analysis from SIFT [1] and contour mapping from OCR [2]. The key point analysis refers to the extraction of key points, or salient points of interest, from the overall image. By identifying and isolating these key points, the algorithm can match images by seeing which pair of images share the most key points. Our algorithm determines key points using contour mapping, which refers to the identification and isolation of contours, or curves and edges, from the image. The algorithm extracts the edge data of the image, from which it uses the specific curves and corners to select key points. Lastly, the brute force matching phase, as the name suggests, does a comprehensive 1:1 comparative search over the images, both to find corresponding key point matches and to directly compare pixels from the original images themselves. Overall, the algorithm can be described as "Brute-Force Contour-Based Key Point Matching" for image identification. As mentioned prior, there is also a denoising preprocessing phase, which uses various techniques including smoothing filters and threshold contrast filtering to remove most noise from the provided images.

Finally, the card sort determination algorithm uses different algorithms depending on the mode of execution. When run in its card-agnostic predetermined sort mode, the algorithm uses a lookup table with predefined card permutation orders to implement standard shuffles such as the perfect faro shuffle [8]. On the other hand, when some or complete randomization is required, the algorithm uses the Fisher-Yates shuffling algorithm [3] to create a truly random card permutation. When run in its card-sensitive mode, however, the algorithm determines the target card sort order by fixing the positions of any specified cards as per user configuration, before filling in the remaining cards randomly with the methods used in card-agnostic shuffling.

3.0 Description of Data Structures

Our software systems only make use of very fundamental data structures, such as lists and maps. Namely, the computer vision model makes use of image data stored in numpy ndarray objects [6], which are just advanced multi-dimensional lists. The card sorting algorithm makes use of a card lookup map and a list to hold the optimal sort order. The microcontroller does not use any

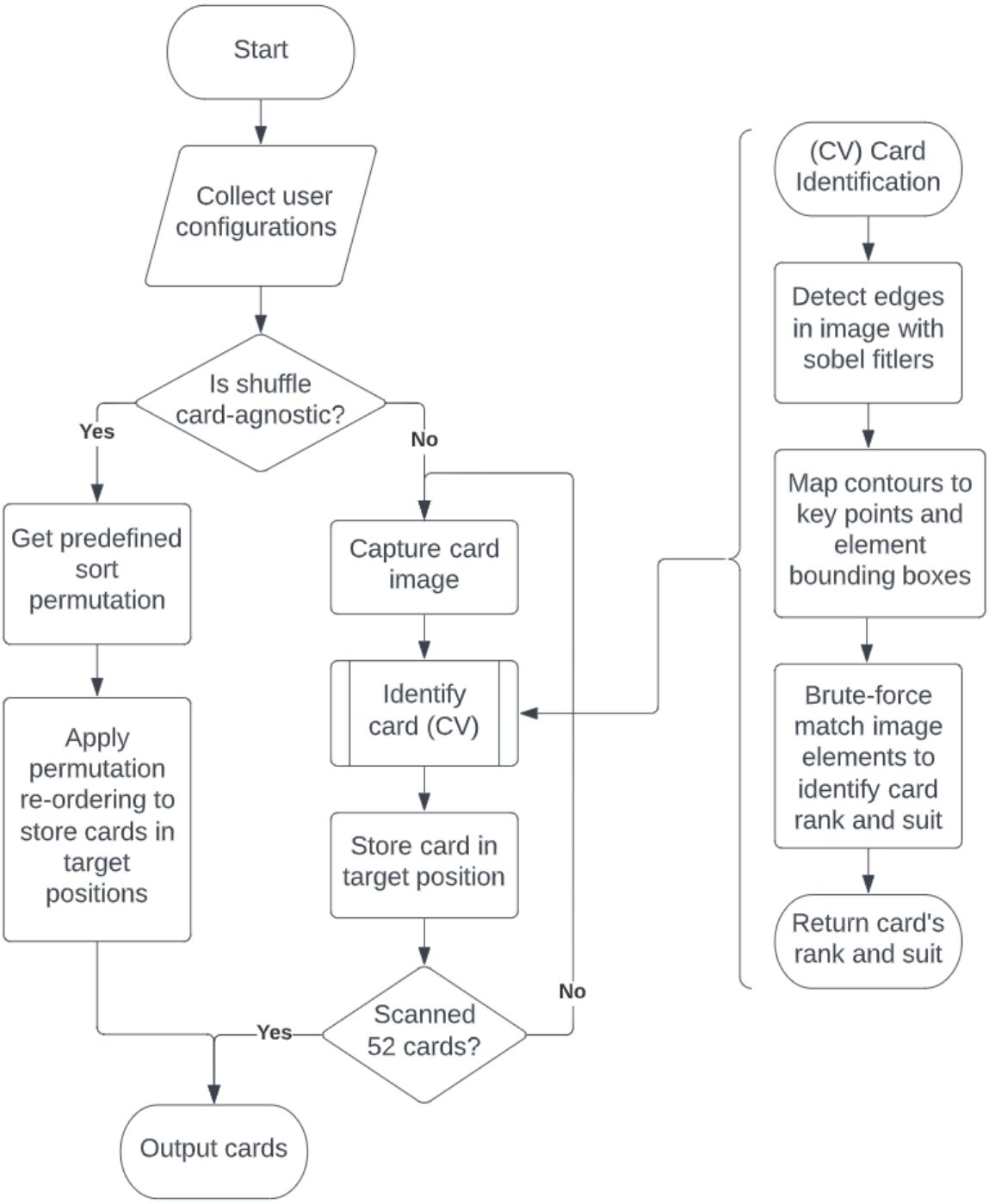
significant data structures, as card processing happens in an online setting, so no state saves are necessary. At most, it uses char arrays to hold strings with user configuration inputs.

In terms of communication packets, there are only transmissions between the microcontroller and the RaspberryPi, which typically consist of single byte messages and acknowledgements. At first, the microcontroller sends a string of comma-separated values holding the user configurations to the RasPi, thus allowing the Pi to compute target sort orders. During sorting operation, the micro sends control signals of a single bit, requesting the Pi to scan the current card whenever a new one is visible for sorting purposes. On the other hand, the Pi responds with a "location acknowledgment" byte that specifies one of 52 locations the current card should be stored in. No further communication occurs between the microcontroller and the RasPi.

4.0 Sources Cited:

[1] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," Computer Science Department, University of British Columbia, 1999. [Online]. Available: https://www.cs.ubc.ca/~lowe/papers/iccv99.pdf. [Accessed: 28-Jan-2023].
[2] "Text Detection and Extraction using OpenCV and OCR," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/text-detection-and-extraction-using-opencv-and-ocr/. [Accessed: 28-Jan-2023].
[3] "Fisher–Yates shuffle," Wikipedia: The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle. [Accessed: 28-Jan-2023].
[4] S. Sodha, "An Implementation of Sobel Edge Detection," Project Rhea. [Online]. Available: https://www.projectrhea.org/rhea/index.php/An_Implementation_of_Sobel_Edge_Detection. [Accessed: 28-Jan-2023].
[5] "Image Processing and Bounding Boxes for OCR," Nanonets. [Online]. Available: https://nanonets.com/blog/image-processing-and-bounding-boxes-for-ocr/. [Accessed: 28-Jan-2023].
[6] "numpy.ndarray," NumPy API Reference v1.24. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html. [Accessed: 28-Jan-2023].
[7] "Permutation," Wikipedia: The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Permutation#Definition. [Accessed: 28-Jan-2023].
[8] "Faro shuffle," Wikipedia: The Free Encyclopedia. [Online]. Available: https://en.wikipedia.org/wiki/Faro_shuffle. [Accessed: 28-Jan-2023].

Appendix 1: Program Flowcharts



Appendix 2: State Machine Diagrams