UNO Cards Detection Using Pattern Matching
Team: UNO Reverse

Amey Karan
2022111026
IIIT Hyderabad
amey.karan@research.iiit.ac.in

Devansh Kantesaria
2022112003
IIIT Hyderabad
devansh.kantesaria@research.iiit.ac.in

Abstract

This project presents a system for the automatic detection and recognition of UNO cards using pattern matching techniques. The UNO deck consists of 108 cards, each associated with a unique color and number. The proposed system is designed to take an image of a UNO card as input and accurately identify its color and type (number). To facilitate this, a custom dataset of approximately 5,000 images of UNO cards in various orientations was created.

1. Introduction

UNO is a card game that is played with a deck of 108 cards, each of which has a unique color and number. Out of these 108 cards, there are 76 Number cards, 24 Action cards, and 8 Wild cards. There are four color *suits* in the deck, which are red, yellow, blue, and green. Each suit has number cards, each having numbers from 0 to 9, and action cards (Skip, Reverse, Draw Two). Additionally, there are Wild cards and Wild Draw Four cards. We hereby call all 13 cards in a suit as *number cards*.

In this project, we propose a system that can automatically detect and recognize UNO cards using pattern matching. The system should be able to take an image of a UNO card as input and output the color and number (or action) of the card. Additionally, we have also created a dataset that contains about 5,000 images of UNO cards in various orientations.

2. Challenges

- **Feature Extraction** - Traditional feature detection algorithms like SIFT fail when applied to highly symmetric objects. SIFT is a robust algorithm for detecting and describing local features in images, offering invariance to scale, rotation, and affine transformations. However, the nature of UNO cards presents significant challenges for SIFT. These cards often exhibit repetitive patterns, symmetry in design, and consistent shapes, all of which can lead to ambiguous or incorrect feature matching. Specifically, SIFT detected multiple keypoints on similar or identical patterns across different regions of a card, resulting in incorrect correspondences during the feature-matching process. This can ultimately lead to inaccurate transformations and unreliable detections. [1][2][3]
- **Perspective Distortion** - Perspective distortion can significantly impact the performance of template matching algorithms. When cards are viewed from an angle, the perspective distortion can cause the card's shape to appear skewed or distorted. This distortion can affect the accuracy of template matching, as the templates may not align correctly with the distorted card image. To address this challenge, additional preprocessing steps or more sophisticated methods may be required to correct for perspective distortion and improve the accuracy of template matching.

3. Dataset

As an appropriate data source could not be identified online, we decided to create our own dataset. This process involved capturing photographs of various cards from different angles and orientations to ensure diversity. We worked with only the 4 suits (red, yellow, green and blue) in this project. Action cards like *wild* and *draw four* were excluded while creating the dataset. Specifically, 35 images of each card were taken, varying the angles and orientation to enhance generalisation. Furthermore, to broaden the dataset's variance, three distinct UNO card decks were utilized. In total, the resulting dataset comprises of 5720 (52 * 110) images, each corresponding to a unique card.

4. Methodology

4.1. Read the image

To read a random card from the dataset, the `get\_card()` function is employed. This function selects a card by randomising its number, suit color, and image index. A file path is then generated to retrieve the corresponding random image. If the image is not already in the correct orientation, it is adjusted by applying a 90-degree clockwise rotation using `cv2.rotate`. Gaussian Blur is then applied to smooth out rapid intensity changes, which helps minimize noise and prepares the image for edge detection. Since the background may contain numerous edges, blurring significantly improves clarity. Finally, edge detection is performed using the `Canny edge detection` algorithm, with gradient intensity thresholds set to 50 (lower) and 150 (upper). Pixels with gradients below 50 are discarded, those above 150 are identified as edges, and gradients between these thresholds are evaluated based on their connectivity to established edges. The resulting processed image clearly highlights the detected edges.

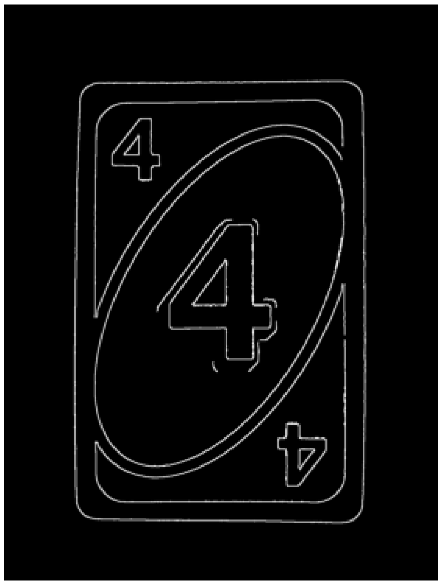

Figure 1. Reading an image

4.2. Segmenting the card

We now segment the card from the given image and remove unwanted artifacts like the background, to this end, we utilise morphological operations, employing a 5×5 square kernel to perform two sequential steps: dilation and erosion. These operations are repeated 15 times each, ensuring that the edges are continuous, well-filled, and clearly defined.

4.3. Corner Detection

Next, we proceed to detect the edges along each card in the image and identify its corners by analysing the intersections of these edges. Once the edges are detected, we extract straight lines representing the card's edges. These lines are then analysed to find their intersection points, which

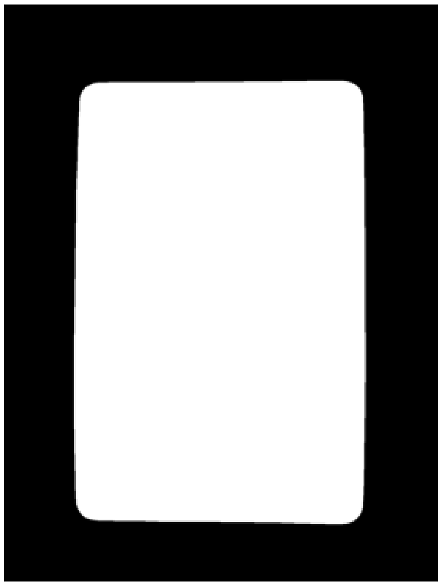

Figure 2. Segmenting the card

correspond to the corners of the card. This process involves determining where two or more detected lines converge, indicating a vertex. Careful consideration is given to ensure that only valid intersections, those that form a quadrilateral outline consistent with the shape of a card, are retained.

Output

```
Point 0:  [968 220]
Point 1:  [217 239]
Point 2:  [ 222 1421]
Point 3:  [ 985 1413]
Line 1:  1182x + -5y + -255299 = 0
Line 2:  -1201x + -746y + 1326688 = 0
Line 3:  1193x + -17y + -1151084 = 0
```

4.4. Cropping the image

We now crop the image to only have the card in focus. This can be done since we now have the corner points of the card. We use perspective transformation to extract and warp the region containing the card from an input image based on four provided corner points. The method generates two rectified views of the region by applying two distinct sets of target coordinates. Out of these two rectified images, only one is correct and useful, while the other is transformed wrongly. This is due to the fact that we have diverse input images, varying across rotations and orientations. Further downstream, only one of these images will be used.

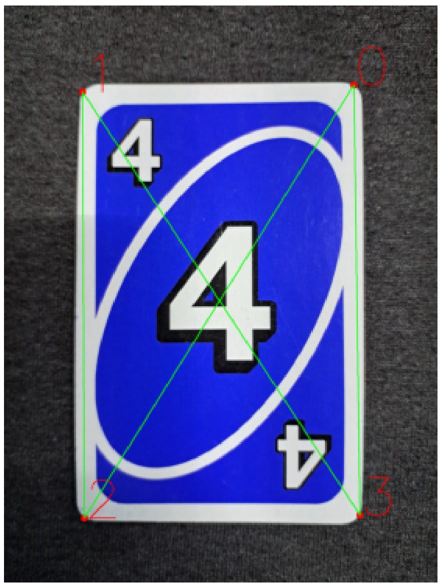

Figure 3. Corner Detection

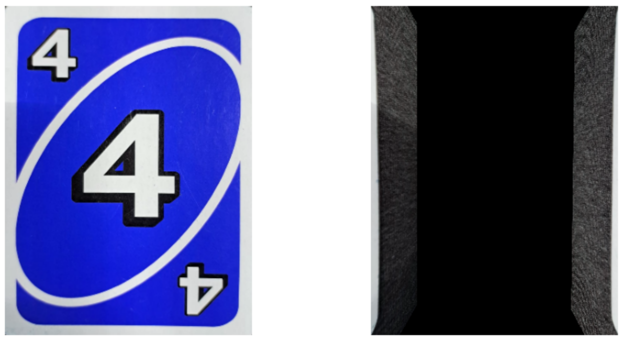

Figure 4. Cropping the image

4.5. Skeletonization

We now perform a series of steps to process and skeletonize the two images. Firstly, we zoom into the centre region of the input image to focus on the number. Then, we apply edge detection using the Canny algorithm. This will now enable us to segment the main focus of the image, which is the number we are trying to detect.

In this very step, one of the two images is discarded as it doesn't yield any meaningful contour. This is done by checking the percentage of white pixels in the contour's mask.

4.6. Matching against Templates

We are now ready to match the given image with the templates that we have for each digit. Each matching results in a match score. We choose the number with the highest match score as the label for the input image.

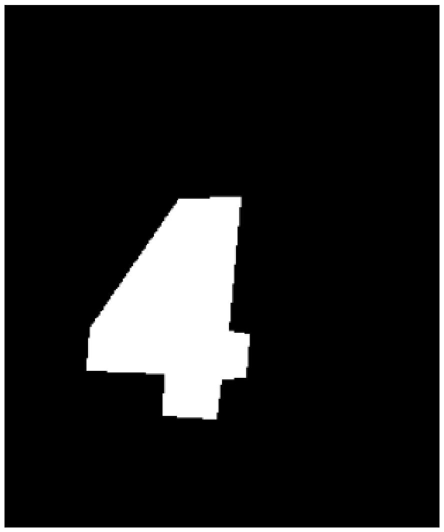

Figure 5. Skeletonization

As a single label classification was not giving good results, we decided to assign the top 3 matches as our prediction. This increased the accuracy of the system, providing greater confidence.

Output

```
Detected number:  4
Detected number:  9
Detected number:  0
```

4.7. Color Detection

To identify the suit of the card, we employed the method of maximum area coverage. We identified the amount of area covered in each of the color belonging to s suit. We assigned the suit with maximum area coverage as the label. This approach worked quite well and gave us 100% accuracy on the dataset.

5. Results

We used our approach to detect the suit and number from 3000 randomly selected images from our dataset 3. Our approach performed pretty decent on the dataset. We obtained an overall accuracy of 77%. We could correctly identify the suits of all cards. Additionally, cards with numbers 0 and 8 gave a 100% accuracy while cards with numbers 4 and $r(reverse)$ also performed good.

Cards with number 3 were often confused with 8 and 1 was misclassified as 7 and vice-versa.

The detailed result, along with how good each suit performed, are presented in Figure 6

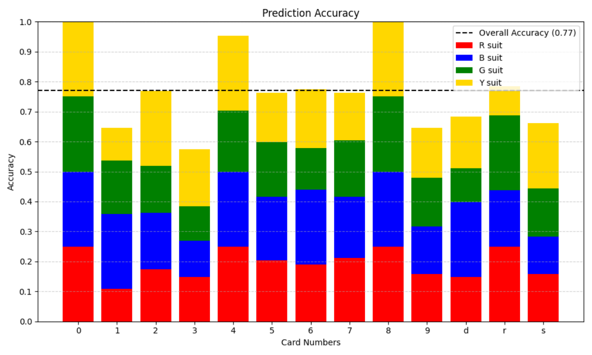

Figure 6. Result

6. Threats to Validity

- At the heart of the approach is template matching. We therefore require templates that accurately represent the number, factoring in all possible variances in orientations. It is extremely difficult to obtain such a close representative of the number. In our project, we used multiple templates (1-5) for a single digit. This could account for the variety of designs of the numbers found across different decks.
- Different manufacturers use different designs of the UNO cards. This leads to a lot of variance within a particular card of a suit. While we have included three different decks in our dataset, it might not contain various designs in other decks.

7. Conclusion

We have developed a system for automatic UNO card detection using pattern matching. Our methodology leveraged edge detection, image segmentation, corner detection, and template matching to accurately recognise the number and color (suit) of each card. Through the creation of a diverse dataset and employing robust image processing techniques, we achieved a promising overall accuracy of 77%, with some cards performing exceptionally well. Challenges, such as handling perspective distortion and the variation in card designs across different decks, were addressed through preprocessing and template refinement. Despite the success, further work could improve accuracy, especially for ambiguous cards, and expand the system's robustness to various card designs and real-world distortions.

References

[1] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1

[2] J. Pimentel and A. Bernardino. A comparison of methods for detection and recognition of playing cards. Technical report, Instituto de Sistemas e Robótica, Instituto Superior Técnico, n.d. 1

[3] David Snyder. Playing card detection and identification, 2019. Stanford CS 232 Winter 2019. 1