Real-Time Detection and Recognition of Cards in the Game of Set

Hrvoje Ditrih[1], Sonja Grgić[1], Leona Turković[2]
[1] University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia
[2] University of Zagreb, Faculty of Croatian Studies, Borongajska cesta 83d, 10000 Zagreb, Croatia
hrvoje.ditrih@gmail.com

***Abstract*— This paper aims to present a procedure for real-time detection and recognition of the cards from the card game Set using computer vision techniques. Our approach to recognizing cards in images is done in three steps: segmenting cards from the image, extracting features from card images with horizontal and vertical lines, and card feature classification with support vector machine (SVM). Small number of features are extracted from relatively small subset of pixels to achieve real-time processing.**

Keywords—Computer Vision; Object Detection, Card Game Set; Card Segmenatation; Card Detection; Card Recognition; Neural Network; Card Classification

I. INTRODUCTION

Computer vision is a research field that aims to develop techniques that enable computers to see and understand the content of digital images in a similar way as people do [1]. The areas closely associated with computer vision are image analysis, image understanding, and image classification. At the current time, computer vision techniques are successfully applied in solving many problems in various areas such as medical imaging, optical character recognition, face recognition and biometrics, self-driving car, motion tracking and capture, etc. Object detection and recognition is a central technology in computer vision that deals with finding and identifying objects in digital images [2]. Object detection is a starting point for solving other complex problems in computer vision.

In this paper object detection methods will be used to detect playing cards in a video sequence and to recognize Set card features. The method presented in [3] uses OpenCV [4] for image upload, card identification, image segmentation, color recognition, and Keras [5] model to predict the class of the card. The model has 92% accuracy for categorical classification. The method presented in [6] used the following steps: extract individual cards from an image, classify each separate image using a deep convolutional network, and find valid set combinations. The results are very promising and for some parameter combinations the set will be found in each image where it exists.

In our approach detection and recognition of the cards is done in three stages on a live video feed using multiple machine learning classifiers. Features are extracted from a very limited number of pixels in the image to save processing time but still, classifiers have high accuracy on the test dataset: 97% – 100%.

978-1-6654-4437-8/21/$31.00 ©2021 IEEE

II. THE GAME OF SET

A. Cards of the Game

Set is a real-time card game whose deck consists of 81 unique cards that vary in four features. Each feature can vary across three possibilities as follows: color (red, purple, or green), shape (oval, squiggle, or diamond), number of shapes (one, two, or three), and shading (solid, striped, or outline). Each possible feature combination occurs just once in the deck. An example of cards from the deck can be seen in Fig. 1.

B. Rules of the Game

The object of the game is to find a set among cards that have been placed face-up on the table and arranged in a grid. A set consists of three cards in which each of the cards' features are all the same or are all different on each card. E.g., the number of shapes must be either same on all three cards or different on each of the three cards, and so on for other card features. Fig. 1 shows a combination of cards that form a set where all card features are different.

The game can be played by two or more players who are all trying to identify sets. The game starts with 12 cards dealt from the deck. When a player finds a combination of cards that forms a set, he must call it before collecting the cards from the table. Then, three new cards from the deck are dealt onto the table. Each time player calls an incorrect set, he must return three cards to the deck if he has any. If none of the players can find a set among dealt cards, three more cards are added to the table, and so on. The game ends when the deck is depleted, and players cannot find a set in the remaining cards. The player with the most cards collected is the winner [7].

C. The problem of Real-Time Detection and Recognition

Card detection and recognition is done on the live video feed from the camera. The developed procedure can detect and recognize an unlimited number of cards in the video frame. Since the aim is to produce real-time recognition, then card

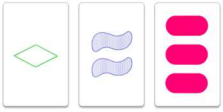


Figure 1. Example of cards in the game of Set

detection, feature extraction, and classification must be sufficiently fast.

For the most part of the game of Set, there would be 12 or more cards dealt. For this reason, feature extraction is done on a relatively limited number of pixels in the image which can reduce inference accuracy and robustness but it will boost the performance which is crucial for the real-time system.

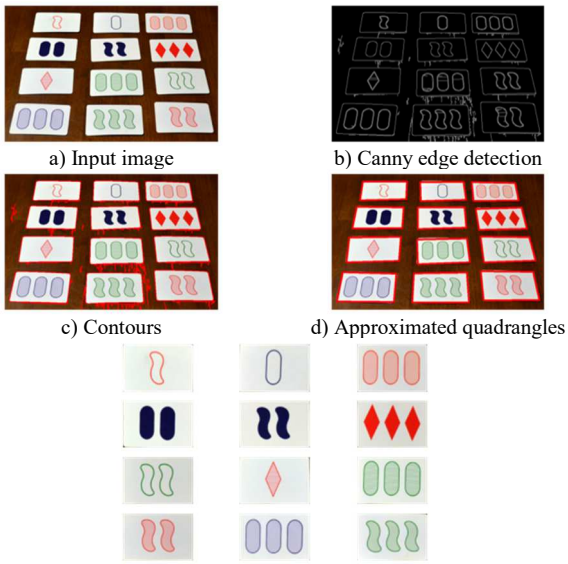
III. Card Detection and Recognition

Detection and recognition of the cards is done in three stages: segmenting the cards from the image, feature extraction from segmented parts, and card feature inference. For every of four card features, one classifier is trained. Also, one additional classifier is trained to determine if segmented part of the image represents a card from the game of Set.

A. Card Segmentation

The central technique used in card segmentation is finding contours with the OpenCV framework. Contours represent a curve joining all the continuous points along the boundary, having the same color or intensity. OpenCV contour finding works on grayscale images where objects that should be detected are white and the background is black [8]. Therefore, to detect objects in an image, a thresholding or edge detection operation like Canny is needed to get grayscale image.

Card segmentation is done in four stages: edge detection, finding contours, approximating contours with quadrangles, and perspective transformation. For edge detection (Fig. 2b), Canny method is used with hysteresis thresholding parameters 30 and 150. Contour finding (Fig. 2c) uses parameters `RETR_EXTERNAL` and `CHAIN_APPROX_SIMPLE`. The first parameter indicates that only external contours of objects are retrieved. This speeds up the process of finding contours since only the outer contours of the cards are needed. The second parameter is explained in [8] and simplifies contour analysis. To filter contours that do not represent cards, two heuristics are used: contour surface must be above 5000 px^2 to filter out very small objects or noise, and contour curve must approximately look like a quadrangle. Contours are approximated with closed polygons using OpenCV function `approxPolyDP` [9]. Accuracy is set to 0.07·*arLen*, where *arLen* represents contour perimeter length. Then, all contours that do not approximate quadrangles are filtered out. The result of heuristic filtering can be seen in Fig. 2d. The final step in card segmentation is perspective transform. Each quadrangle is transformed to an image of dimensions 450x300 px using OpenCV function `warpPerspective`.



a) Input image b) Canny edge detection

c) Contours d) Approximated quadrangles

e) Perspective transform

Figure 2. Card segmentation example

B. Cards Dataset

To train classifiers, a representative dataset is needed. Using the card segmentation procedure described in section A, a dataset consisting of cards from the game of Set was created. The dataset contains 935 images. Each card from the deck appears at least 10 times in the dataset. Images were captured in various lighting conditions, from different angles, and at different camera distances. Fig. 3 shows 30 randomly selected images from the dataset.

C. Scanlines

For each classifier, a uniform way of extracting features is used. Lines of pixels, called scanlines, are extracted from the segmented image. There are horizontal and vertical scanlines. Horizontal lines are placed on values 126, 150, and 174 of the y-axis. Vertical scanlines are placed on 157 and 255 values of the x-axis. All features are extracted from scanline pixels. Fig. 4 shows both vertical and horizontal scanlines and their order.

D. Segmented Image Classification

Card segmentation procedure and its filtering heuristic passes through all objects in the shape of quadrangles and cannot alone filter out objects that are not cards from the game of Set. For this reason, a two-layered fully connected neural network (FCNN) is used as the classifier. Input into FCNN are pixels from second horizontal scanline in HSV color space.

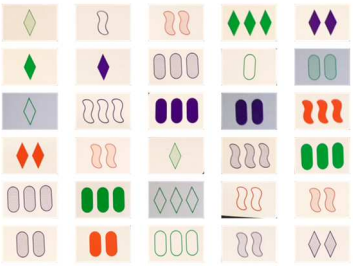


Figure 3. Cards dataset

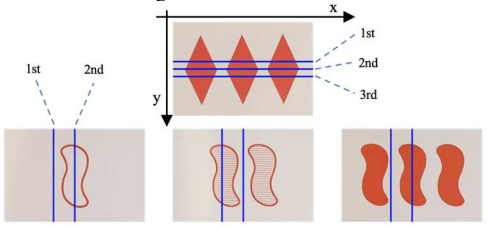


Figure 4. Horizontal and vertical scanlines

The reasoning for using HSV instead of RGB is that FCNN can directly, rather than implicitly, use H and V channels which determine pixels' hue and brightness. The training dataset consists of:

1. Images from the dataset described in section B annotated as true (935 images).
2. Again, images from the dataset described in section B, but rotated 90° and rescaled to dimensions 450x300 (935 images). The reason is to have images that look similar to cards but are annotated false.
3. 68 images that are not cards of the game of Set.
4. 100 generated images of uniform noise.
5. 100 generated images of uniform color.

There is also a test dataset consisting of images of cards from the game of Set downloaded from the open-source project[1] (81 images) and generated images of uniform noise and uniform color (100 + 100 images).

FCNN was trained with Adam [10]. The input layer consists of 3·450=1350 neurons. The first layer has 500, and the second 200 neurons. ReLu activation function was used. Output is one neuron with the sigmoid function representing the probability of the input image being a card of the game of Set. Mini-batch of size 20 was used with dataset shuffling at every epochs' end. Cross-entropy was used as a loss function.

E. *Number of Shapes Classifier*

Feature called hit position is used for the number of shapes classifier. Hit position is the normalized position of first non-white pixel on second horizontal scanline (Fig. 5). Hit position plot on the dataset can be seen in Fig. 6 alongside the result of training SVM classifier with linear kernel. To determine hit position, Gaussian adaptive thresholding was used with width 5 and threshold parameter 2. Adaptive thresholding was used rather than fixed because of non-uniform lighting conditions that can occur in the scene.

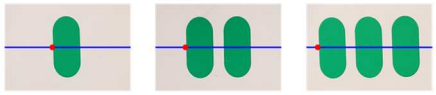


Figure 5. Hit position

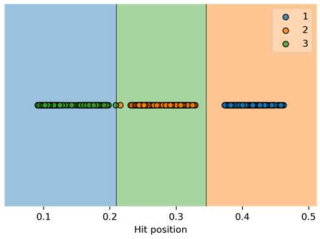


Figure 6. Number of shapes classfier

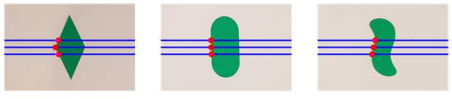


Figure 7. Hit positions for different shapes

F. *Shape Classifier*

For shape classification, distances between hit positions of different scanlines are used. Let hit positions of horizontal scanlines be *a*, *b,* and *c* (Fig. 7). Let *l1=a-b* and *l2=c-b*. Pair *(l1,l2)* is the feature used for shape classification. Fig. 8 shows *(l1,l2)* plot on the dataset alongside decision boundary for the SVM classifier with the polynomial kernel of degree 3.

G. *Color Classifier*

RGB pixel chosen with heuristic from the second horizontal scanline is used as a feature for the color classifier. The heuristic is choosing a pixel with the most representative color (Fig. 9). Firstly, pixels that are considered are the ones that pass adaptive threshold explained in section E. Then, pixel with the highest saturation is chosen based on HSV values. Often, dark pixels can have high saturation, and determining hue from dark pixels is more difficult than from bright ones. Therefore, only pixels that have the value of channel V higher than 29 are considered, except in cases where none of the pixels values are higher than 29. Fig. 10 shows the plot of the

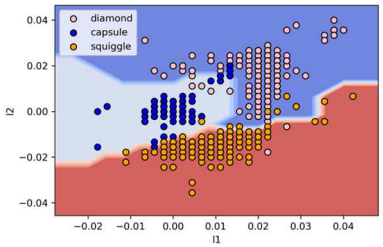


Figure 8. Shape classifier

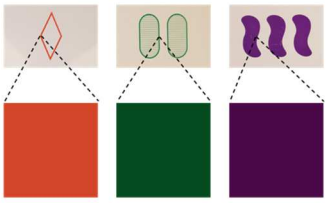


Figure 9. Most representative color of the second horizontal scanline

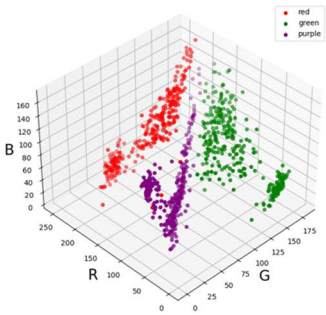


Figure 10. Most representative color plot on the dataset

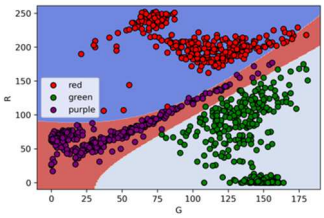


Figure 11. Color classifier

[1] https://github.com/nicolashahn/set-solver/tree/master/image-data/all-cards/labeled

RGB values of the most representative colors of cards in the dataset. SVM with the polynomial kernel of degree 5 was used as the classifier. The decision boundary for the fixed value *B=127* can be seen in Fig. 11.

H. Fill Classifier

Only the fill classifier uses vertical scanlines for feature extraction. Cards with two shapes use first vertical scanline, and cards with one or three shapes use second vertical scanline. Because of this, the number of shapes must be inferenced before continuing to fill classification.

First, scanline pixels are color corrected by transforming every pixel with a diagonal matrix whose values are $\frac{255}{R'}$, $\frac{255}{G'}$, $\frac{255}{B'}$, where R', G', and B' are RGB values of the brightest pixel in the scanline. Then, features are extracted from the 30 pixels around the center of the vertical scanlines. Considering pixels are in HSV color space, let $\bar{s}$ be the mean value of pixels' S channel, and let $\bar{v}$ be the mean value of the V channel. The pair $(\bar{s}, \bar{v})$ is the feature used for the fill classifier. Fig. 12 shows the plot of the feature on the dataset alongside the decision boundary of the SVM classifier with the RBF kernel.

IV. RESULTS

The accuracy of the classifier that determines if segmented part is the card from the game of Set is 99.67% on the training dataset and 100% on the test dataset. Even though the accuracy score is high on the train and test datasets for this classifier, its accuracy in the real world was not as high. The classifier has a bias towards classifying objects as non-cards, but it works well when cards are close enough to the camera and when the cards are positioned horizontally.

The performance of the card feature classifiers is dependent on the underlying algorithms of image processing and feature extraction. That is why the test dataset wasn't created to fine-tune the parameters of the machine learning classifiers. The accuracy scores of the classifiers are given in table 1.

All the card feature classifiers give high accuracy in a real-world application. The number of shapes classifier gives the best performance of all classifiers. The shape classifier is very susceptible to noise and slight errors during card segmentation. Color classifier best performs under good lighting conditions. In dark scenes, it has a bias towards classifying cards as purple because that is the darkest color out of the three. Fill classifier has problems distinguishing outline and stripes fill when cards are far away from the camera. The reason is low resolution of the segmented image making the striped fill look like outline fill. This can also be seen in Fig. 12 where outline and stripes fill form one cluster. All classifiers and feature extraction techniques are simple and fast enough for real-time usage.

TABLE I. CLASSIFIER ACCURACY ON THE DATASET

Classifier	Accuracy
Number of shapes	100%
Shape	97%
Color	99.79%
Fill	99.36%

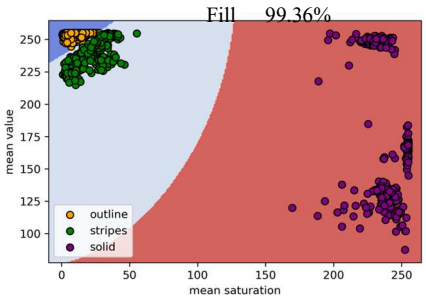


Figure 12. Fill classifier

V. CONCLUSION

This paper presents a procedure for detecting and recognizing the cards from the game of Set in real-time video feed using classic computer vision techniques – namely, image processing. Five classifiers are presented. Four for each of the cards' features and one for classifying segmented parts of the image. It was showed how using a very limited number of pixels from the image classifiers can produce impressive accuracy and speed.

Classic computer vision techniques for object detection and recognition lack in robustness compared to their deep learning counterparts. Therefore, it would be interesting to explore deep learning solutions for real-time card recognition. That said, classifiers presented in this paper can increase their robustness by extending the dataset and exploring more interesting image processing techniques. Namely, the neural network presented in this paper would greatly benefit from a larger and more varied dataset of Set cards and non-set cards.

REFERENCES

[1] V. Wiley and T. Lucas, "Computer Vision and Image Processing: A Paper Review", International Journal of Artificial Intelligence Research, vol. 2, no. 1, pp. 28-36, June 2018.

[2] K. Sharma and N. V. Thakur, "A review and an approach for object detection in images", International Journal of Computational Vision and Robotics, vol. 7, pp. 196–237, January 2017.

[3] A. Dotis, "Ready, SET, Image Recognition," *Medium*, Jun. 05, 2018. https://dganais.medium.com/ready-set-image-recognition-720be22d7051 (accessed June 3, 2021).

[4] J. Howse and J. Minichino, Learning OpenCV 4 Computer Vision with Python 3, Packt Publishing, 2020.

[5] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2019.

[6] S. Tetelepta, Detecting SET cards using transfer learning, https://towardsdatascience.com/detecting-set-cards-using-transfer-learning-b297dcf3a564 (accessed June 3, 2021)

[7] https://www.setgame.com/sites/default/files/instructions/SET%20INSTRUCTIONS%20-%20ENGLISH.pdf (accessed June 10, 2021)

[8] OpenCV: Contours : Getting Started'. https://docs.opencv.org/4.5.2/d4/d73/tutorial_py_contours_begin.html (accessed June 3, 2021)

[9] OpenCV: Contour Features, https://docs.opencv.org/4.5.2/dd/d49/tutorial_py_contour_features.html (accessed June 3, 2021)

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", arXiv:1412.6980 [cs], January 2017, http://arxiv.org/abs/1412.6980 (accessed June 3, 2021)