Identification and Classification of Gambling Dice using Image Feature Detection and a Convolutional Neural Network

Jesse Patrick Sheehan
Department of Electrical and Computer Engineering
University of Canterbury
Christchurch, New Zealand
jps111@uclive.ac.nz

Richard Green
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
richard.green@canterbury.ac.nz

***Abstract*—This paper proposes a method to identify dice values using image processing and machine learning techniques. Prior research uses the pips of a die to determine its value, whereas the proposed method detects the outline of the die. Canny's edge detection and Suzuki's contour tracing algorithms were used to extract die faces from an image. A convolutional neural network was used to classify these die faces and to provide a probable number value. The proposed method shows a 90% success rate in correctly detecting and classifying dice values from a webcam.**

***Index Terms*—dice, gambling, computer vision, machine learning**

I. Introduction

Dice (figure 1) are small cube shaped objects often used in board games, card games, or casino games to produce a random number. They typically have small circles (pips) painted or embossed on each side to denote their value. The modern style dice have been dated to as early as 600 BC from China [1]. Modern dice are usually made of plastic and are white with black pips.

Figure 1. An example of five standard dice.

Dice value detection is a valuable tool for the gaming and gambling industries. Image recognition has often been used in conjunction with other techniques to protect casinos against cheating [2][3]. Common image processing and feature detection methods can be used to identify the position of dice. Machine learning algorithms can be used to classify the value of the dice. Consumer-grade hardware devices are now more capable of handling the processing requirements of such methods.

This paper proposes a method of detecting and classifying dice using these techniques.

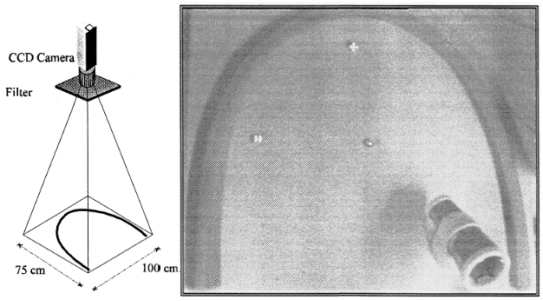

Figure 2. The apparatus setup including gaming table and camera.

II. Background

Prior research in this field have taken advantage of numerous methods.

A. Prior Research

The "SORTE" system was commissioned by the Portuguese Gaming Inspection Authorities for use in casinos [3]. It identified the locations of the pips on all dice and uses spatial locality to assign values to each. However, this system required specific dice, a birds-eye view of the surface (figure 2), and a careful ambient lighting setup.

The system designed by Lapanja, et al., detects dice values for a mechanical gambling machine [4]. It uses color difference to identify the pips and template matching to classify the value of each die. This method requires a birds-eye view, a high-contrast background surface, and relies on hard-coded masks for classification.

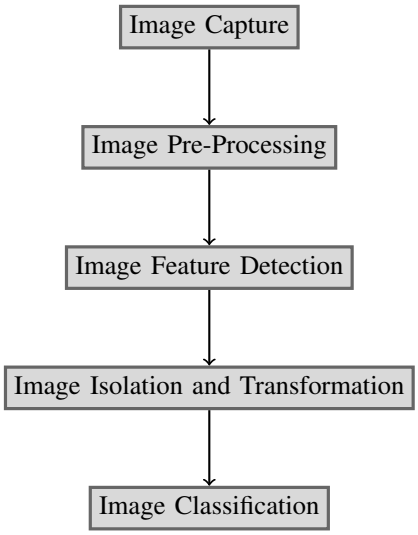

Figure 3. The image processing pipeline.

The system devised by Huang [5] uses a modified unsupervised gray clustering algorithm. However, it requires a birds-eye view, a high-contrast background, and the number of dice to be known in advance.

Finally, the method proposed by Chung [6] uses the least distance criterion to classify die values. This method detects the pips and then groups the pips into valid configurations based on the relative distance between each pip. This method requires a birds-eye view and has only been tested on up to four dice.

Dice detection does not live entirely in the realm of industry and academia. Amatuers and hobbyists have had success in creating programs that solve this problem [7][8].

All of these methods rely on detecting the pips and have been constrained by either the camera angle, the number of dice, or the background surface. The proposed method focuses on detecting the die outline and then classifying the value using a convolutional neural network (CNN).

III. Method

The proposed method is arranged in several stages. Figure 3 illustrates the process of classifying the dice values from an image.

A. Image Pre-processing

The image (figure 1) is obtained from a source such as a file or video stream. Pre-processing is performed to remove extraneous information from the image.

The image is converted to a grayscale color-space. Equation 1 is used to convert an RGB image into grayscale. It is based on the luminance formula from "Recommendation BT.601" [9]. Performing subsequent operations on a grayscale image increases speed at the cost of losing colour information.

$$\text{gray}(R, G, B) = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad (1)$$

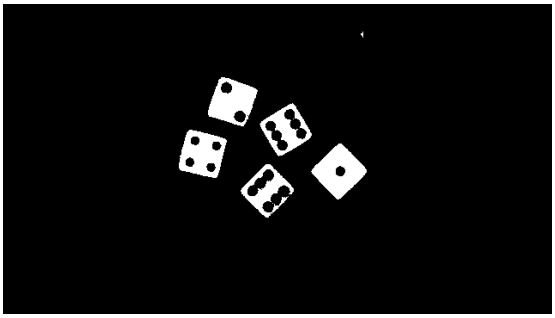

Figure 4. The image after thresholding.

Figure 5. The image after pre-processing.

A binary threshold is performed (equation 2). This reduces the amount of noise present in the image and draws attention to sharp edges (figure 4). This step is required for the edge detection step.

$$\text{threshold}(x) = \begin{cases} \text{black} & x \leq 160 \\ \text{white} & \text{otherwise} \end{cases} \qquad (2)$$

Smoothing is applied to the image to reduce the effect of any remaining noise. This is typically acheived with a 2-dimensional convolutional filter. A gaussian kernel of size 5 was chosen for the convolution (equation 3). The image (figure 5) is now sufficiently clean of noise and feature detection can take place.

$$K = \{0.0625, 0.25, 0.375, 0.25, 0.0625\}^T \qquad (3)$$

B. Die Face Detection

The dice detector stage is broken up into discrete steps. First the edges are detected using Canny's edge detection algorithm. Then the edges are joined into contours by Suzuki's contour tracing algorithm. Finally, the contours are filtered based on their size and shape. This leaves only the contours that are likely to be die outlines.

1) Edge Detection: The Canny edge detector [10] is used to find the edges of the dice. It builds upon the Sobel filter [11] which finds extreme vertical and horizontal gradients, G_x and G_y, in an image. The edge gradient, G, and angle, θ, for each pixel is then given by equations 4 and 5 respectively. The

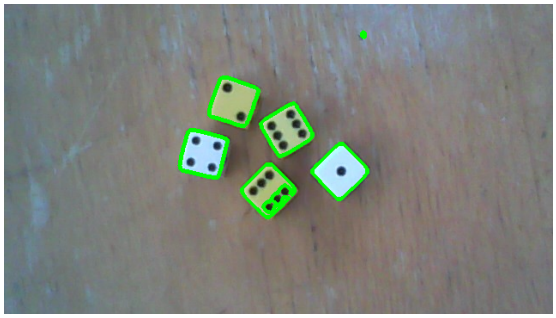

Figure 6. The original image with the edges highlighted.

Figure 7. The original image with the remaining contours highlighted.

edge gradient is always perpendicular to the edge, therefore it is rounded to either horizontal, vertical, or a diagonal.

$$G = \sqrt{G_x^2 + G_y^2} \tag{4}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{5}$$

Every pixel is then checked to ensure it is the local maximum in the direction of the gradient at that point. This has the effect of making the edges one pixel thin. Finally, a double threshold is applied to ensure that only the relevant edges are kept. Any pixels that have a gradient intensity above 320 are considered to be edges. Any pixels that have a gradient intensity above 110 and are connected to an edge are also considered edges. A small number of edges that are not related to the dice remain due to variations in lighting and background surface texture (figure 6).

2) Contour Tracing: The Suzuki contour tracing algorithm [12] joins related continuous edges. It walks over the image and categorises each edge. In this way it can create a hierarchy of edge features such as the outer border and hole border.

3) Contour Filtering: The contours are then kept if they have the following properties:

- They have four points (i.e. are quadrilateral).
- They have an area greater than 200 pixels square. This removes any small contours that are artifacts of the lighting.
- They are square-ish. The shape rejection filter (listing 1) will reject contours that are not square enough. The proposed implementation uses an acceptable error of 20%.

Listing 1. The shape rejection filter in Python.

```
def get_edge_length(p1, p2):
    return np.sqrt(
        abs(p2[1] - p1[1]) ** 2 +
        abs(p2[0] - p1[0]) ** 2)

def is_square(points, error):
    [p1, p2, p3, p4] = points
    e1 = get_edge_length(p1, p2)
    e2 = get_edge_length(p2, p3)
    diff = abs(e1 - e2) / max(e1, e2)
    return diff <= error
```

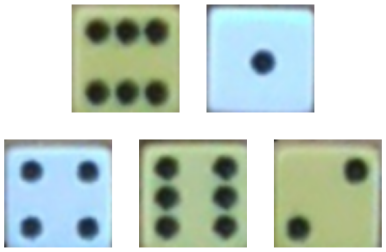

Figure 8. The die faces after transformation.

After contour filtering is performed only the dice are left (figure 7).

C. Die Face Transformation

The location and angle of the die face is now known within the original image (figure 6). Each die face must now be transformed before the CNN can be applied.

Each square region must be cropped from the original image and rotated so that it lies flat upon the cartesian axis. Rotation is performed with an affine rotation transformation. Each image must be scaled to 64x64 pixels for classification. The die faces have been isolated (figure 8) and can now be passed to the CNN for classification.

D. Machine Learning

Machine learning algorithms are often used to solve problems that would be difficult to solve using traditional computing methods [13]. A CNN is a class of deep neural networks that has many applications in computer vision, speech recognition, and many other fields [14][15][16][17]. A deep neural network is designed with many layers where the information will pass through. A model will then be trained on a large dataset with known parameters. This model can then be used to classify images that it hasn't seen before.

The proposed CNN has 4 layers and uses categorical cross-entropy as its loss function. This allows the model to produce probabilities for all six values while being aware that only one is correct. The model was trained on 1200 images of white and yellow dice that were manually captured and classified. Each image is grayscale and 64x64 pixels in size.

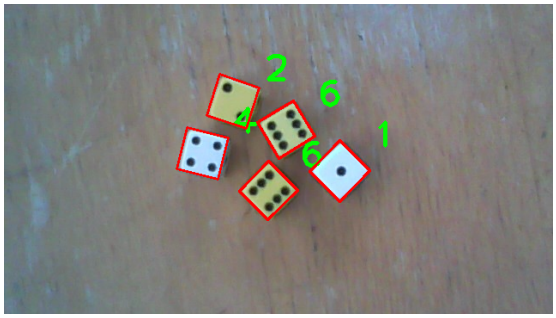

Figure 9. The final image with annotations.

E. Die Classification

The CNN determines the probability that the isolated die face has a specific number value. The largest of these values is selected and the original image is annotated with the most likely number for each dice (figure 9).

IV. Results

A laptop with the following hardware specifications was used for testing:

- 2.4 GHz Intel Core i5 processor,
- 8 GB of memory.

A proof-of-concept program was written. The following software was used to run the program:

- Python 3.8.1,
- OpenCV 4.2.0,
- Tensorflow 2.2.0,
- Windows 10 (64-bit).

A Logitech C170 webcam with a resolution of 640x360 pixels was used to capture images and videos. A wooden desk was used as hard surface where the dice could be rolled. A desk lamp was used to improve the ambient lighting on the desk.

The webcam was setup approximately 27 cm above the surface looking directly down (figure 10). Two white dice (black pips) and two yellow dice (black pips) were rolled 50 times. The webcam was paused while the results were recorded in a spreadsheet.

Over the 200 rolls, 95.0 % of the dice were detected, and 95.5 % of these were correctly classified. This yields an overall accuracy of 90.7 %. The distribution of the dice rolls was relatively equal between the different values (figure 11).

The proposed method is sensitive to the ambient lighting conditions and care was taken to reduce the reflections from the background surface and the dice themselves. This method also uses a birds-eye view camera, it does not work well when at an angle of more than 15° away from the die's surface normal.

V. Conclusion

The proposed method yields an accuracy of 90.7 % under the test conditions. All prior research showed a 100 % accuracy under their respective test conditions. Although the proposed

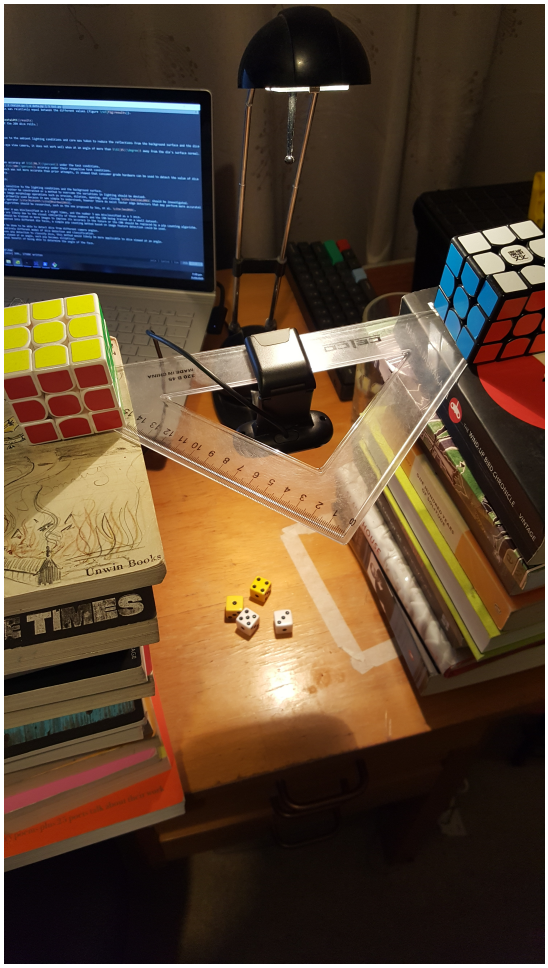

Figure 10. The test apparatus.

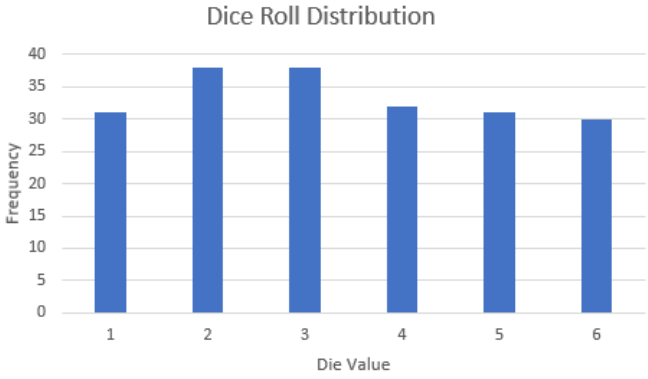

Figure 11. The distribution of the 200 dice rolls.

approach was not more accurate than prior attempts, it showed that consumer grade hardware can be used to detect the value of dice with high levels of accuracy.

A. Future Research

The dice detector is very sensitive to the lighting conditions and the background surface. In the future, this should either be constrained or a method to overcome the variations in lighting should be devised. More experimentation into image morphology operations such as erosion, dilation, opening, and closing [18] should be investigated. Canny edge detection was primarily used because it was simple to understand, however there do exist faster edge detectors that may perform more accurately, such as the declivity operator [19][20]. Faster contour tracing algorithms should be researched, such as the one proposed by Seo, et al. [21].

During testing, the number 4 was misclassified as a 2 eight times, and the number 5 was misclassified as a 3 once. The misclassifications are likely due to the visual similarity of these numbers and the CNN being trained on a small dataset. Either the CNN model should be trained on more images to improve its accuracy in the future or the CNN should be replaced by a pip counting algorithm. Once the image is segmented into different die faces, a simple pip counting method based on image feature detection could be used.

Further research must be done to be able to detect dice from different camera angles. This may involve an entirely different model of dice detection and classification. All prior research uses pip detection to classify dice, this method would likely be more applicable to dice viewed at an angle. This is because when viewed at an angle, each pip becomes elliptical. This has the additional benefit of being able to determine the angle of the face.

References

[1] P. Romanowski. (May 31, 2020). "Dice." Encyclopedia.com, Ed., [Online]. Available: https://www.encyclopedia.com/manufacturing/news-wires-white-papers-and-books/dice.

[2] N. Krahnstoever, J. Rittscher, P. Tu, K. Chean, and T. Tomlinson, "Activity recognition using visual tracking and rfid," in *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, vol. 1, 2005, pp. 494–500.

[3] B. A. B. Correia, J. A. Silva, F. D. Carvalho, R. Guilherme, F. C. Rodrigues, and A. M. de Silva Ferreira, "Automated detection and classification of dice," in *Machine Vision Applications in Industrial Inspection III*, F. Y. Wu and S. S. Wilson, Eds., SPIE, Mar. 1995. DOI: 10.1117/12.205506.

[4] I. Lapanja, M. Mraz, and N. Zimic, "Computer vision based reliability control for electromechanical dice gambling machine," in *Proceedings of IEEE International Conference on Industrial Technology 2000 (IEEE Cat. No.00TH8482)*, Jaico Publishing House. DOI: 10.1109/icit.2000.854173.

[5] K.-Y. Huang, "An auto-recognizing system for dice games using a modified unsupervised grey clustering algorithm," *Sensors*, vol. 8, no. 2, pp. 1212–1221, Feb. 2008. DOI: 10.3390/s8021212.

[6] C. H. Chung, W. Y. Chen, and B. L. Lin, "Image identification scheme for dice game," in *International Conference on Advanced Information Technologies*, Ta Hwa Institute of Technology, 2009.

[7] G. Vos. (Sep. 17, 2018). "Dice detection using opencv," [Online]. Available: https://gideonvos.wordpress.com/2018/09/17/dice-detection-using-opencv/.

[8] D. Pesce. (Sep. 6, 2019). "Reading dice with opencv," [Online]. Available: https://www.davidepesce.com/2019/09/06/dice-reader-part-1/.

[9] ITU-R. (Mar. 1, 2011). "Recommendation itu-r bt.601-7, Studio encoding parameters of digital television for standard 4:3and wide-screen 16:9 aspect ratios," [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf.

[10] J. Canny, "A computational approach to edge detection," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 1986.

[11] I. Sobel, "An isotropic 3x3 image gradient operator," *Presentation at Stanford A.I. Project 1968*, Feb. 2014.

[12] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, Apr. 1985. DOI: 10.1016/0734-189x(85)90016-7.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[14] J. Peña, P. Gutiérrez, C. Hervàs-Martínez, J. Six, R. Plant, and F. López-Granados, "Object-based image classification of summer crops with machine learning methods," *Remote Sensing*, vol. 6, no. 6, pp. 5019–5041, May 2014. DOI: 10.3390/rs6065019.

[15] A. Madabhushi and G. Lee, "Image analysis and machine learning in digital pathology: Challenges and opportunities," *Medical Image Analysis*, vol. 33, pp. 170–175, Oct. 2016. DOI: 10.1016/j.media.2016.06.037.

[16] S. Lawrence, C. Giles, A. C. Tsoi, and A. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997. DOI: 10.1109/72.554195.

[17] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," Apr. 8, 2014. arXiv: 1404.2188v1 [cs.CL].

[18] R. Szeliski, *Computer Vision*. Springer London, 2011. DOI: 10.1007/978-1-84882-935-0.

[19] P. Miché and R. Debrie, "Fast and self-adaptive image segmentation using extended declivity," *Annales Des Télécommunications*, vol. 50, pp. 401–410, 1995.

[20] S. Bhardwaj and A. Mittal, "A survey on various edge detector techniques," *Procedia Technology*, vol. 4, pp. 220–226, 2012. DOI: 10.1016/j.protcy.2012.05.033.

[21] J. Seo, S. Chae, J. Shim, D. Kim, C. Cheong, and T.-D. Han, "Fast contour-tracing algorithm based on a pixel-following method for image sensors," *Sensors*, vol. 16, no. 3, p. 353, Mar. 2016. DOI: 10.3390/s16030353.