”Roll the dice” - Detection of Dice and Pips using Computer Vision Techniques

Jeeveswaran Kishaan
Masters in Autonomous Systems
Hochschule Bonn-Rhein-Sieg
Bonn, Germany
kishaan96@gmail.com

Abstract

This paper proposes a unique solution to the classical dice detection problem. It discusses a solution to extract the right frame from a video of throwing dice, from which the algorithm could proceed to detect the dice, its position, and calculate the number of pips on the top face automatically. The solution involves many functionalities from the open source computer vision library ”OpenCV”, and Scipy library's hierarchical clustering. The experimental results are discussed along with the challenges faced, assumptions made, advantages of this solution and its limitations. The paper concludes with avenues for future work in order to improve the accuracy in this task.

Index Terms

OpenCV, blob detection, region of interest, hierarchical clustering, dice, pips, frame

I. Introduction

Games involving dice is very common these days. Increasing number of casino games and electronic games which use dice to define the outcome or luck of the player is commonplace and gives a lot of thrill. The uniform distribution of the outcomes of a die makes it near impossible to predict the outcome of throwing a die before it comes standstill. Dice are produced in such a way so that the sides are all symmetrically shaped and the center of gravity of the die lies in the center of the cube which is made of a homogeneous material. Thus, the dice are produced with perfection and their uniformity in mind, which appeals to these games which depend on its outcome.

Though the outcomes of dice can be monitored using humans, it is both expensive and error prone. Tracking the outcome of multilple dice continuously is a very tiresome task and this makes it near impossible for a human to be employed to monitor the game for a long time. In addition, human monitoring is biased and it could not be trusted all the time. Machine vision is the hot topic of the 21^{st} century and its applications are proliferating. Starting from fault detection in factories to autonomous driving, computer vision has been thoroughly studied for its deployment in every stage of our daily life activities. A very clever and robust machine algorithm could prove cost and resource efficient to any task. The unbiasedness of such systems is obvious and this makes turning to computer vision a very logical step for the task of detecting and counting the score of dice in a game.

Though machine vision is appealing to this task, it is imperative that such a system is accurate and fast for it to be completely deployed to make a auto-recognition system. Delays in recognizing the score of the dice could extend the total duration of the game and such a system would fail to impress the fast-moving world. Any fallacy in recognizing the score would have serious impacts on both the system's credibility and cost of the organization.

This paper discusses a novel method to fast and easy detection of the location of dice and the scores on their front face. The remaining part of this paper is structured as follows. Section II discusses the work done by other researches on applying machine vision to this task. Section III describes the methodology taken to tackle this problem along with assumptions made, and the challenges faced during the realization

of this approach. Section IV elaborates on the experiments conducted using this approach on a set of 20 videos of dice being thrown and a detailed discussion of the results on every step of this approach along with illustrations of the outcomes. Section V concludes the paper by making conclusions on the approach, contributions made, and avenues to be explored in future that have the potential of making such a system better.

II. Related Work

Score detection algorithms became so appealing to many users as these systems are fraudulent-free and accurate. Lot of work have been put in this task of applying computer vision techniques for the task of calculating the score of the dice in gambling games machines. Approach taken by [4] automatically recognizes the position and scores of the dice thrown on casino tables. A monochrome CCD camera is employed to capture the image of the final position of the dice and the images are analyzed online for making out the pips on the dice. They employed many image processing algorithms for the task of instant throwing detection and grouping of the dice. Considerable features of this approach by Correia et al., are that the diameter of every pip and the distance spans between the pips on the die should be jotted down before the recognition algorithm, classification of the dice and the pips on the front face (side facing upward) is done through a template / pattern matching technique, and the electronic gambling device was developed using ”contactless electronic ID keys and a scanner” for the task of predicting the location of the position of every dice.

The work done by [1] proposes a unique clustering algorithm called ”Modified Unsupervised Grey Clustering Algorithm” for the task of automatically detecting the position of each die and extracting the pip score of it. The clustering algorithm is based on ”grey theory” whose aim is to ”provide relational analysis, predication, decision, control, and clustering data for the grey system” [1]. The grey clustering approach has also seen successful deployment in tasks such as predicting the fluctuations of stock market [5], and to monitor the rate at which the vegetation is recovered in places of earthquake based on satellite images [6]. Thus proposing such a clustering algorithm for this task is very novel and it resulted in a very fast and accurate dice detection system. Though the paper claims a 100% accuracy, the experiments were done on with dice on a white background. The results on different background could have proved the efficiency of such a system and the robustness of it. Thus, the efficiency of this approach on real-world problems is still a question.

The work [7] elucidates the importance of having 100% accuracy for the task of detecting the score of dice in real world examples and the pips are recognized ”using an array of contactless read-out points implemented as electronic ID key scanners”. The authors claim such a system could be very cost effective and robust to noise. Lapanja et al [8] leverage the color difference and quick and instant chroma-key applications. A solution to predict the location of the dice is discussed and and the number of pips is detected using a template matching mechanism is presented in this work. Glenn [3] extensively harnesses the open-source computer vision library Open-CV’s [9] functionalities to efficiently detect the dice and extract the number of pips from them. After extracting the image of the final position of the dice, Open-CV’s techniques are used to bound the dice, extract them, and recognize the pips on them. Though this approach is very accurate, the assumptions made in this work question the suitability of such a system in real-life scenarios. White dice on black tray comes with a huge advantage of a stark difference in contrast between the dice and the background which facilitates easy bounding of the dice. Consecutively, this paves the way for easy detection of black pips on white dice.

Another approach for this task was done by [2] using features of images of dice and least distance criterion (LDC) method. An automatic system to detect dice was implemented based on many image preprocessing methods to disregard noise and other circular and bigger objects. Post pre-processing the dice score calculation algorithm is implemented with the help of the LDC algorithm, dice score calculation and score exception processing methodologies. The novelty of this method lies in the fact that it tries to address the noise and foreign objects during the detection of pips. The works discussed in this section

propose many methodologies to pre-process, and extract features from the image for the task of dice score detection. However, a system which could address noise, blurred video, deceitful actions while throwing the dice, foreign objects which resemble the dice with pips in the same frame along with the dice, colored backgrounds, and transparent dice simultaneously at once is still a open research problem. This work tries to consider all these challenges while reading the scores in a detailed manner.

III. Methodology

The approach taken in this work to tackle the problem of detecting the number of dice and the number of pips on the top face of each of the dice and attributing it to the right die is discussed in this section. Many of the Open-CV's classical computer vision techniques and Scipy's [10] algorithms are utilized in this task. The main challenges which are unique in this problem are as follows;

- Videos of throwing dice are not uniform as the length of each video and frame rates are different in every video.
- The videos having set up at a higher level to include the surroundings of the tray as well results in detecting circular shaped objects in them.
- Different positioning of the camera means that the side faces of the dice are also ran for pip detection, and fish-eye lens necessitates image obtained through such a lens to be pre-processed before running the actual algorithm.
- Deceitful actions of the hand while throwing the dice and throwing the dice at different instances in the same video (throwing some after throwing the initial dice) makes it hard to extract the right frame to run the pip detection algorithm.
- Blurred, and noisy videos with different illuminations adds to the complexity.
- Foreign objects which resemble the shape and color of the dice with fake spots on the tray leads to false positives.
- The background / the tray and the dice in different colors, and transparent dice are hard to detect and classify.

A novel method which can address all these challenges while tackling the task of dice detection will be discussed below. A logical breakdown of the steps taken in this task is as follows;

1) Loading the video, going through every frame, and detecting the still frame
2) Finding the region of interest where pip detection should run.
3) Preprocessing the extracted still frame so that the pips could be detected easily.
4) Extracting the pips on every die and attributing them to the correct die along with the position of the die.

Fig.1 shows the sequence of these logical breakdown of steps. The videos were in the ".avi" format with different frame rates and lengths which are read using openCV's VideoCapture function.

A. Frame Aquisition

As considering each and every frame would cost a lot of memory and time, this method adapts a mechanism where only two frames per image is chosen for every second of the video. One of the assumptions made in this step is that the dice will be stationary for at least one second in the video. Thus, all frames are read and each and every frame's number is saved in a variable. The frame is considered for the extraction of the right frame only if the frame number is a multiple of half of the frame rate of the video. In addition the first few frames and the last few frames are ignored to make the frame extraction step concentrate only on the middle part of every video. This approach is taken hence it is seen that every video starts with a hand about to throw the dice on the tray and ends with a hand collecting all the dice back from the tray.

The frames acquired in such a manner are stored in a list after converting them into grayscale using openCV's cvtColor method. The images are converted to grayscale as this approach doesn't depend on

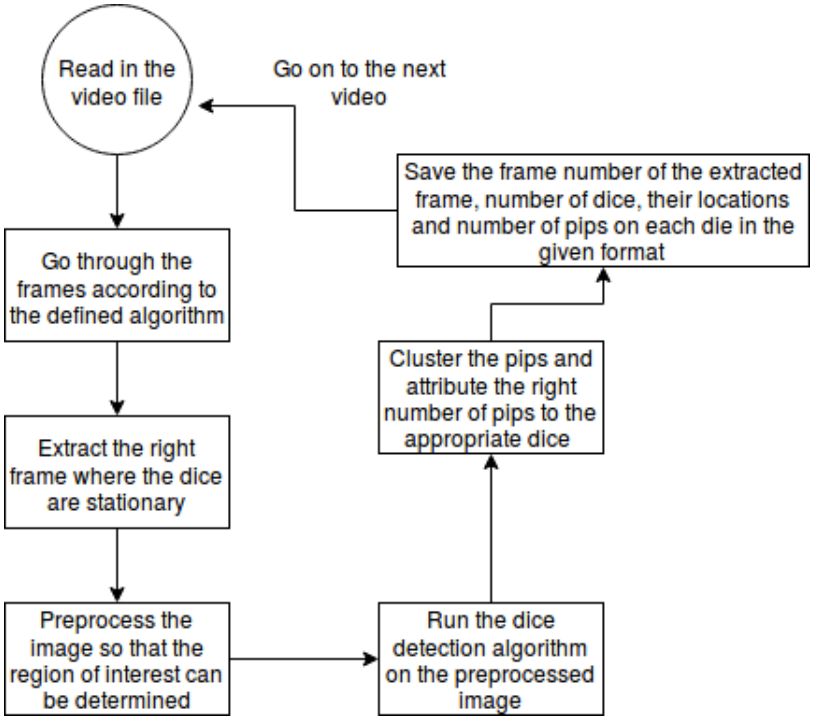


Fig. 1: Flow chart of the steps taken in the methodology.

any color dependent features to extract the number of pips or to detect the dice. The frames in the list are considered in pairs in a sequence from the beginning to end. The absolute difference between every pair of frames are acquired as an image using openCV's absdiff method. Following this, the image is processed using Gaussian blur of 3 by 3 kernel and binary thresholded to remove slight movement of any background elements (for ex. even a slight movement of the dress of the person who throws the dice could contribute much to the absolute difference).

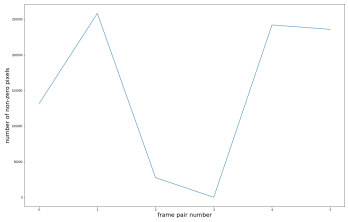
After all these steps, the number of non-zero pixels in the final image obtained is stored in a separate list for every pair of images. The list of number of non-zero pixels for every pair of frames represents the amount of movement occurred between the two frames in every pair. Thus, the second frame of the pair which had minimum movement between them would represent the frame which had no movements and thus the frame of the still dice after settling down. Fig.2 shows the plot of frame difference between every pair of frames for one of the video named "2018-10-08@16-03-10.avi".

As evident from the plot, the frame corresponding to the 4^{th} pair in the first example and 2^{nd} pair represent the frames with minimum movement. There were only three pairs in the Fig.2c as the video was very short and had a high framerate.

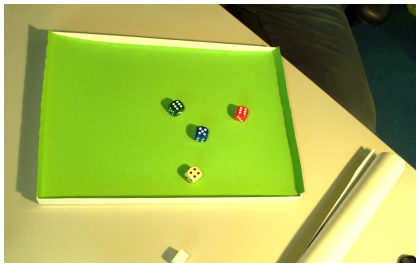
B. Defining the Region of Interest

After the acquisition of the right frame, the image with the still dice is passed through a bilateral filter to reduce the effects of any noise obtained from noisy cameras. Bilateral filter is chosen for this task as it produces better images after blurring rather than any other blurring algorithm which would blindly consider all the pixels in a pixel's neighborhood. Bilateral filter preserves the important features such as edges and corners as it takes the weighted average of neighboring pixels, thus giving more weightage to distinct features. This filter should be handled carefully and tuned accordingly as a bad application of this filter could result in false edges.

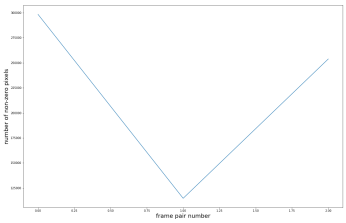
The blurred image is fit a fixed frame of black color to avoid detecting circular shaped objects that don't lie on the tray (thus not belong to any die). Pixels in the 6% of the top, 9% of the bottom, and 5% of each right and left sides of the image are set to black which results in an effect of putting a black



(a) Frame difference between the pair of frames read from the video.



(b) Extracted frame of stil dice from the video.



(c) Frame difference between the pair of frames read from the video.

(d) Extracted frame of stil dice from the video.

Fig. 2: Extracting the right frame.

frame over the image. The initial image and the image after fixing the frame is shown in Fig.3. As can be seen in Fig.3a there are circular designs on the dress of the person standing next to the table and a small circular object on the stand at the bottom of the image which would result in false positives when the pip detection algorithm runs on the image. Thus, using this frame, such misfortunes could be avoided. The assumption made in this step that all trays on which the dice are rolled are almost in the center of the area captured by the camera.

(a) Image fitted with a fixed-sized black frame.

(b) Initial frame with background.

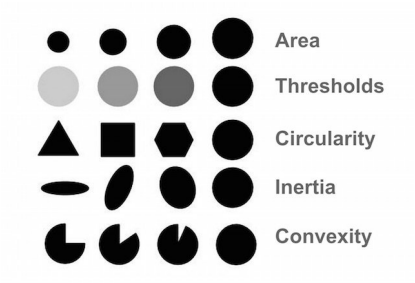
Fig. 3: Defining the region of interest by fixing a frame of black color around the image.

C. Detection of Pips

The image with the region of interest set would be used to detect the pips using openCV's blobdetection algorithm. The algorithm runs as follows [11];

1) Converting the initial image into many binary images by using the thresholding (setting any pixel with value higher than the thereshold to a color and the less ones to another color) function with different threshold values starting from minimum threshold to maximum threshold.

2) Grouping the white pixels that are closer to each other into single entity so that the blobs could be easily identified in each of the binary images.
3) The centers of each of the blob in each binary image is considered and the other blobs in the neighborhood are merged with the initial blob if the blob is within a minimum threshold of distance. This step avoids considering a big blob as two separate blobs.
4) Finally the position of the center of the blobs thus recognized and the radius of the corresponding blobs are returned.



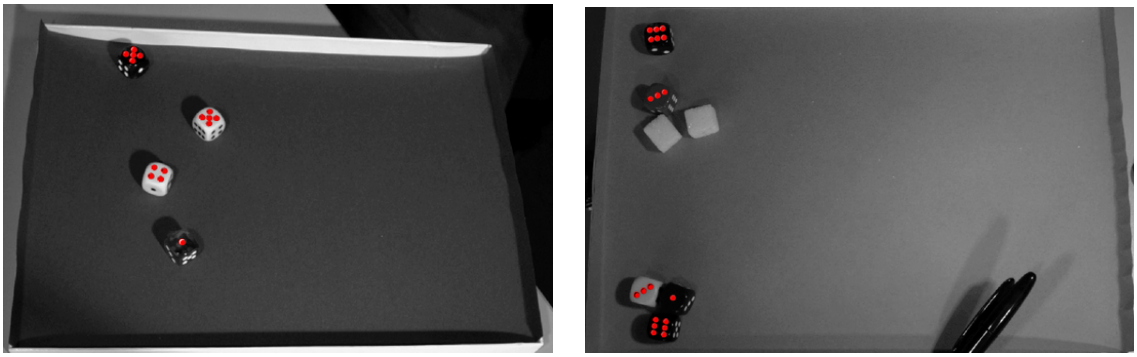
(a) Illustration of different parameters used in blob detection, Source: [11].

Parameter	Value
min_threshold	50
max_threshold	200
min_area	100
max_area	250
min_circularity	0.4
min_inertia_ratio	0.4

(b) Values of parameters used in this approach.

Fig. 4: Definition of parameters used in blob detection and values used.

In addition, openCV's blob detection offers the option of filtering the blobs thus detected using color, size, and shape (which is defined by how close to the circle the blob is, how convex the blob is, and how flat the circle is). Fig.4 shows the illustrative description of each of the parameters mentioned here and the values of those parameters used in this approach. It is assumed that the dice thrown in the videos are not smaller than a particular size (defined by the *min_area*), and not bigger than a particular size (defined by the *max_area*). Another subtle assumption is that the videos are not recorded in a very slanted position (defined by *min_inertia_ratio*). Given the video is recorded with a camera in a very slanted position, the pips on the top face would not be in a proper circular shape and eventually would have a very low inertia ratio, thus failing to get detected in the algorithms.



(a) (b)

Fig. 5: Result of the blob detection algorithm.

Fig.5 shows the outcome of such a blob detection algorithm on the extracted frames. As can be seen on the Fig.5b, such an algorithm easily avoids the sugar cubes which are almost the same size as the dice as this approach looks for pips rather than detecting the dice using other edge detection algorithms.

D. Attributing the Location of the Dice and Number of Pips

After the detection of all the pips, they need to be separated with respect to distinct dice and the right pips should be attributed to the right die. Scipy library's hierarchical clustering algorithm is used for this

task. The hierarchical clustering algorithm considers every pip as a separate cluster and combines the pips which are closer to each other into a single cluster as the threshold increases. It constructs a dendrogram as shown in Fig.6 where each letter represents a pip detected in the previous step. A horizontal line could be drawn at any level across this dendrogram and the resulting groups under the line is considered as clusters.

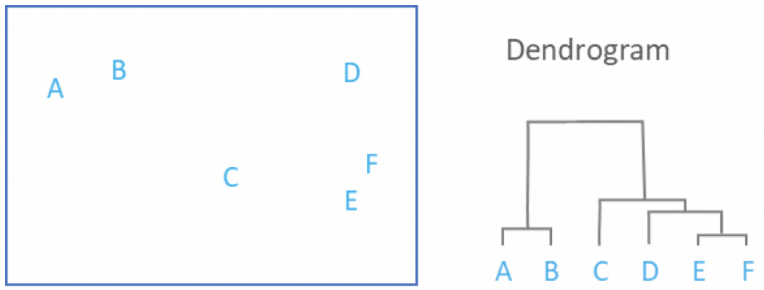
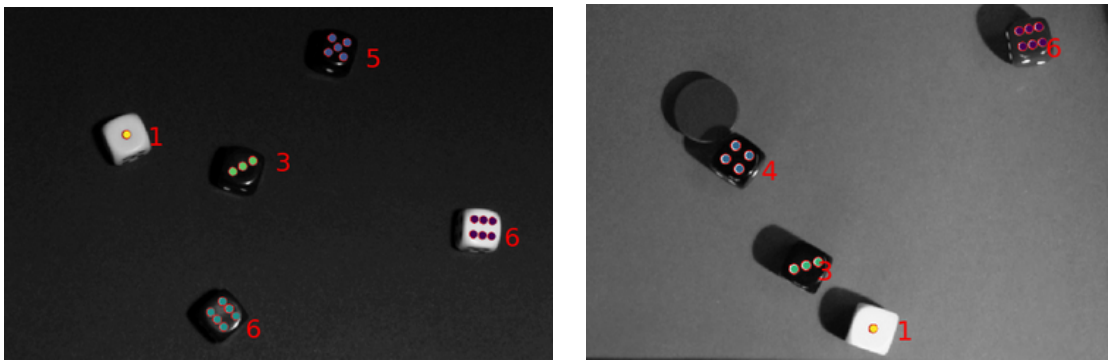


Fig. 6: Dendrogram created by a hierarchical clustering algorithm. Source: [12]

A threshold value of 39 is used in this application to separate the pips that are far away and group the pips that are very close to each other. The position of the pips obtained from openCV's keypoint detection algorithm is used to find the position of each die by taking the mean of all the pips' locations as the pips on a die would always be scattered around the center of the top face. Thus, every cluster is considered as a die and the position of the die is taken from the position of the pips in that cluster. The result of such a clustering algorithm is shown in Fig.7.



(a) 2018-10-08@13-54-00.avi (b) 2018-10-08@14-45-58.avi

Fig. 7: Result of the clustering algorithm.

As can be seen from the Fig.7, the right pips and number of pips are attributed to the right die using this clustering algorithm.

IV. Experimental Evaluation and Results

The method discussed in the methodology is used on a collection of 20 videos with different length, dices and backgrounds. The results have been tabulated in Table.I. The scores are evaluated based on the following rules;

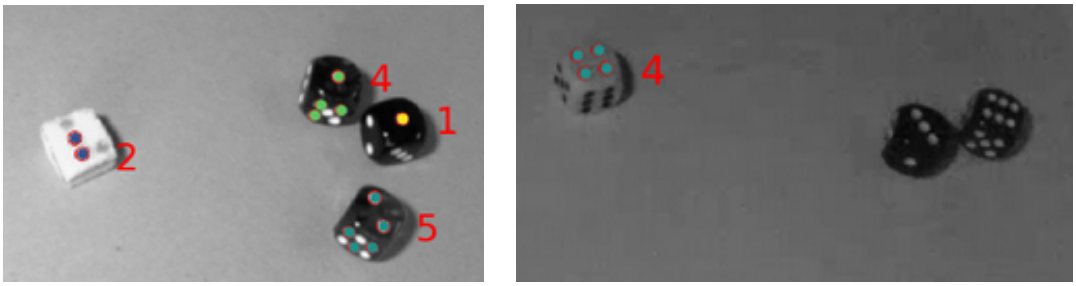
- +1 points for correct detection of the position of a die
- +1 points for correct detection of number of pips on the top face
- -1 points for detecting anything other than the dice on the tray

Based on the experimental results it is evident that the algorithm fails to achieve the full points due to detecting false positives and not detecting certain dice. It is also observed that the algorithm succeeded in

TABLE I: Experimental Results

Filename	number of dice	number of dice detected	number of false positives	total points available	points scored
2018-10-08@16-03-10.avi	4	4	0	8	7
2018-10-08@15-55-36.avi	4	4	1	8	5
2018-10-08@16-25-11.avi	4	4	0	8	6
2018-10-08@15-47-40.avi	6	2	0	12	3
2018-10-08@15-41-24.avi	5	4	0	10	7
2018-10-08@16-28-46.avi	4	4	0	8	6
2018-10-08@16-17-02.avi	4	4	0	8	8
2018-10-08@13-38-29.avi	6	6	0	12	12
2018-10-08@13-54-00.avi	5	5	0	10	10
2018-10-08@15-00-41.avi	5	5	0	10	10
2018-10-08@16-13-24.avi	5	6	1	10	6
2018-10-08@16-09-35.avi	3	3	0	6	6
2018-10-08@15-34-45.avi	5	3	0	10	6
2018-10-08@15-26-37.avi	6	7	2	12	8
2018-10-08@15-20-53.avi	5	2	0	10	4
2018-10-08@14-55-15.avi	6	6	0	12	12
2018-10-08@16-20-32.avi	3	3	0	6	5
2018-10-08@15-14-43.avi	5	5	0	10	9
2018-10-08@14-45-58.avi	4	4	0	8	8
2018-10-08@15-51-36.avi	6	5	0	12	9

extracting the right frame of still dice from all the videos. The algorithm runs in 3.194 seconds per image which is quite efficient. In order to analyze the failures, the outcomes were analyzed and the examples where the algorithm failed is shown in Fig.8. As can be seen in this figure, it is evident that the pips detected on a sugar cude, detecting the pips on the side of the dice when the video is recorded in a slant angle, and not detecting the dice at all in noisy and blurred images are the reasons behind the inferior performance of the algorithm.



(a) 2018-10-08@15-55-36.avi (b) 2018-10-08@15-34-45.avi

Fig. 8: Example of failures.

V. Conclusion and Future Work

The proposed method can address the dice detection problem where the background are in different colors and the dice are made of different materials. Though the approach was efficient, its accuracy is affected by the video quality and the position of the camera while recording the video. However, it is seen that the proposed approach would work with high accuracy and efficiency if the camera is positioned straight above the tray, the camera is focused right on the tray such that no other background is recorded, and no other foreign objects are placed on the tray.

For the task of accurate detection of dice from the videos used in this experiment, the following avenues could be explored in future;

1) Deep learning algorithms or computer vision methods could be employed to bound and extract the tray from the background.
2) More clever algorithms to bound each and every die and extract the top face could be implemented to avoid detecting the pips on the side faces.
3) Algorithms which can detect and ignore foreign objects such as sugar cubes would be a boon to the accuracy of such a system.
4) Grey clustering approach could be tried out to improve the grouping of pips on the same dice.

ACKNOWLEDGMENT

This work is not possible without Prof. Dr. R. Herpers and his associates who gave me the opportunity to work on such a project. In addition, I would like to thank those who were involved in the recording of the videos and implementation of the evaluation software. Finally, I would like to extend my gratitude to my family and friends whose love and moral support helped me in steering this project at the right pace.

REFERENCES

[1] Huang, K.Y., 2008. An auto-recognizing system for dice games using a modified unsupervised grey clustering algorithm. Sensors, 8(2), pp.1212-1221.
[2] Chung, C.H., Chen, W.Y. and Lin, B.L., 2009. Image identification scheme for dice game. In Proc. of 2009 Int. Conf. on Advanced Information Technologies (pp. 24-25).
[3] Glenn De Backer, "Recognizing Dices", 28 September 2016, Retrieved from: https://www.simplicity.be/article/recognizing-dices/.
[4] Correia, B.A.B., Silva, J.A., Carvalho, F.D., Guilherme, R., Rodrigues, F.C. and de Silva Ferreira, A.M., 1995, March. Automated detection and classification of dice. In Machine Vision Applications in Industrial Inspection III (Vol. 2423, pp. 196-203). International Society for Optics and Photonics.
[5] Hsu, Y.T.; Lin, G.G.; Chou, Y.M. Predictions of tops and bottoms of stock market based on grey clustering method. Joint Conf. AI Fuzzy Syst. Grey Syst. 2003.
[6] Yang, C.M., Chen, J.C., Peng, L.L., Yang, J.S. and Chou, C.H., 2002. Chi-Chi Earthquake-caused Landslide: grey prediction model for pioneer vegetation recovery monitored by satellite images. Botanical Bulletin of Academia Sinica, 43.
[7] Bergant, U., Zimic, N. and Lapanja, I., 2000, January. Design considerations of an electromechanical dice gambling machine. In Industrial Technology 2000. Proceedings of IEEE International Conference on (Vol. 1, pp. 444-447). IEEE.
[8] Lapanja, I., Mraz, M. and Zimic, N., 2000, January. Computer vision based reliability control for electromechanical dice gambling machine. In Industrial Technology 2000. Proceedings of IEEE International Conference on (Vol. 2, pp. 325-328). IEEE.
[9] Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.
[10] Jones, E., Oliphant, T. and Peterson, P., 2014. SciPy: open source scientific tools for Python.
[11] Glenn De Backer, "Blob Detection Using OpenCV (Python, C++)", 17 February 2015, Retrieved from: https://www.learnopencv.com/blob-detection-using-opencv-python-c/.
[12] Tim Bock, "What is a Dendrogram?", Accessed: 16 January 2019, Retrieved from: https://www.displayr.com/what-is-dendrogram/.