Rank Counting In A Game of Black Jack

Marcus Wong Yew Hon, Wong Shen Yang, Sim Hann Zern

BSc Computer Science

***Abstract*— In this paper we present an algorithm for playing card rank counting and winner classification in a game of black jack. This algorithm is robust enough to perform rank counting and determining the winner in a game of black jack with moderate card rotation and partial occlusion efficiently. It is achieved using perspective correction, minimum bounding box and template matching.**

***Keywords*— template matching, perspective correction, card occlusion, rank counting, winner classification**

I. Introduction

We were tasked to develop a system that would be able to detect playing cards from a set of given images and determine the winner of a game of black jack as well as counting the total ranks of each player. The system should be able to distinguish between different cards from different players and should be able to regardless of card orientation or partial occlusion.

This system could prove to be useful for industries requiring identification of an object from a top down perspective. Casinos in particular will benefit greatly from systems similar to ours as there is a need for casinos to monitor playing cards on a gambling table in real time to prevent cheaters and scammers.

Early testing of the proposed system showed promising results for high resolution images whereby the average accuracy rates for rank detection exceeded 90% even with partial occlusion of cards. However, results from low resolution images leaves a lot to be desired. On the other hand, the average accuracy rates for winner classification exceeded 70%.

II. Literature Review

Reading through several prior works in regards of card detection, the most common method in these works is a matching technique known as template matching. In [1], [2] and [3], the authors proposed algorithms for card recognition while implementing template matching with great success and accuracy. However, not much can be found in terms of card detection with occlusion with the exception of [4].

Before template matching process, M. Paulo, P. R. Luís and T. Luís [1] proposed a method of pre-processing an image with grayscale conversion, Gaussian blur and using edge detection, specifically canny edge detector, before finding the contour of the cards in an image. Perspective correction and enlargement is used later, but the end result presents with an extremely blurry image of the card. This may be due to the enlargement process and may affect detection accuracy. The perspective correction is achieved by obtaining the corner coordinates of each card. The perspective transform matrix is then calculated using the obtained corner points and a newly created destination point. The top left and bottom right corners of the card are cropped and used for template matching operations. This is a simple but efficient way for obtaining sufficient card rank and suit information. Lastly, template matching is performed using a set of pre-existing template images, where the match returning the lowest value is determined to be a match. The authors have set a matching threshold whereby any match exceeding the threshold will be disregarded as a match. This method could significantly reduce the occurrence of false positives. However, this method does not include solutions to combat card detection with occluded cards.

B. Dan and W. Hugh [2] have a similar proposed method with [1], having a slight difference in image pre-processing procedures. In [2], instead of using edge detectors, the authors proposed using binary thresholding and region filtering for card segmentation. The end result is excellent, with each segmented card being clearly distinct from one another and the image having minimal noise. However, edge detection was implemented during template matching in [2] on the cropped card corners as direct template matching without edge detection resulted in poor results. This seems to be an issue only to the authors of [2] as template matching without edge detection worked well for both M. Paulo *et al* [1] and C.B Zheng's [3] proposed algorithm. B. Dan and W. Hugh [2] have also highlighted the challenges they have faced implementing other template matching algorithms such as SIFT and SURF. Not only were the end results using these algorithms poor, the algorithms are significantly more computationally heavy compared to regular template matching as well. Similar to [1], card occlusion is not taken into consideration for this method.

C. B. Zheng and G. Richard [3] proposed an algorithm with similar core concepts to [1] and [2], but with a very different approach in implementing and executing the perspective correction process. While the algorithm for perspective correction in [3] is similar to [1] and [2], C. B. Zheng and G. Richard proposed for each detected card to be rotated at 10-degree clockwise increments, up to 180 degrees, while performing template matching. The images are scaled to multiple sizes ranging from 90% to 40% as well. The images are matched every 10-degree rotation increment when the size of the images is scaled for each iteration. In total, 108 matches will be performed for each card. The end results for [3] prove to be rather impressive with almost flawless rank detection for images at full scale. However, the suit detection is rather poor for J, Q and K cards as the suits in these cards are generally represented by an alternative suit pattern, thus harder to be recognized by the algorithm. Card occlusion detection using this system seems to be unlikely as well.

Lastly, J.P Pimentel [4] introduced a method to solve the problem of card detection for occluded cards. The method utilizes Hough transform, an algorithm that detects lines in an image. Hough transform is used to detect the lines of the occluded cards. As Hough transform is based on a voting mechanism to determine line validity in an image, even small visible edges of the partially covered card can have its lines detected. This results in many possible rectangles, or false positives, of the detected card to be created. To solve this, [4] implemented non- maxima suppression on the rectangles whereby each rectangle in close proximity of a rectangle with more votes are discarded, giving the best card representation rectangle. Steps similar to [1] are then implemented in [4], where the chosen rectangles are cropped, perspective corrected and subjected to template matching at the extracted card corner for rank and suit classification. The end results are astounding with no false positives for occluded card detection and 100% detection matching rate for images with sufficient resolution.

III. Aims & Objectives

We aim to develop a system that would be able to detect and count the ranks of playing cards in an image as well as determining a winner in a game of black jack that is robust enough to perform well with moderate rotations of cards and partially occluded cards.

Several key steps are required to achieve our aim. Our first objective is to extract and isolate the occluded cards from each player in an input image. This will allow us to identify the total amount of playing cards within an image. The next objective is to obtain the rank information of each individual card within an image. Orientation correction and cropping will be necessary to obtain rank information from the card in a consistent manner. Last objective is to count rank value of each player's cards by matching the extracted rank information with a set of training images. Rank counting is also required for determining the winner in a game of black jack.

IV. Methodology

In this segment, we will discuss in detail about the methods that we have implemented to achieve the objectives we have set in the previous segment. In a nutshell, the chronological implementation methods of our system are as follows: image preprocessing, player identification, card segmentation, orientation correction, template matching and rank counting with winner classification. Figure 1 illustrates the steps mentioned and would be further discussed in its respective sub-section.

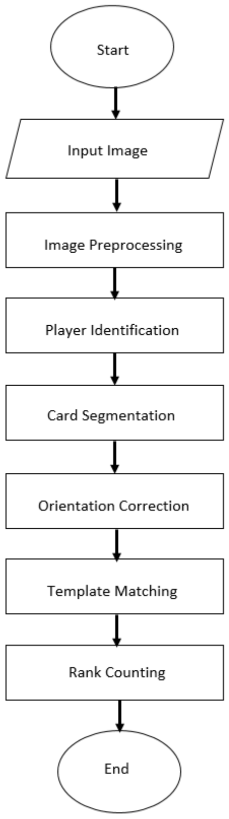


Fig. 1 Main steps in playing card suit counting.

A. Image Pre-processing

Typical in a casino table setting, where card detection and recognition are commonly implemented, playing cards are usually presented on a table where there exists a stark difference of contrast between the cards and the table surface. This brings us to utilizing thresholding to segment the cards from the background to find the card contours. However, pre-processing of the image must first be done prior to segmentation.

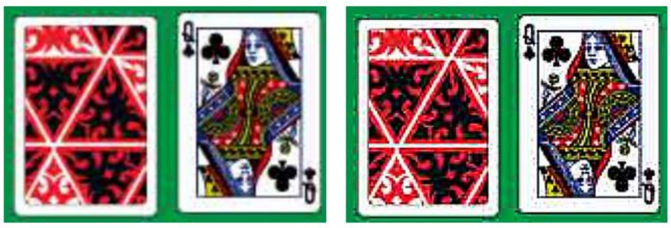
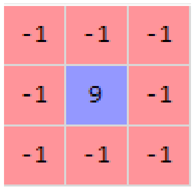


Fig. 2 Before and After Image Sharpening

The image is first resized to a fixed dimension for more consistent results in case input images differ in sizes. The images are then flipped 180 degrees for the rank information of the cards to be at the top left corner. Then, the resized images are converted from a RGB image to a grayscale image. A 2d filter will be applied to the image if the input image is of low resolution. The 2d filter will sharpen the image with the help of a sharpening structuring element as shown in figure 3. This is to increase the effectiveness of card segmentation later on as a more defined image will yield clearer edges that defines individual card segments more distinctly.



-1	-1	-1
-1	9	-1
-1	-1	-1

Fig. 3 Structuring Element used for Sharpening of Image

B. *Player Identification*

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

If g(x,y) is a thresholded version of f(x,y) at some global threshold T

The resized image is then converted into a binary image through a simple global thresholding method denoted in (1). There is no need for a complex thresholding method as the images are well perceived and contrast strongly with its background. Edge detection is not implemented as we find thresholding to be sufficient in card segmentation operations.

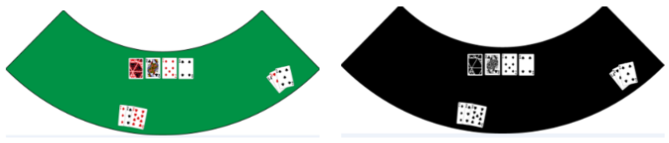


Fig. 3 Before and After Thresholding

After thresholding, each of the player's cards, which we will call hand, in the binary images would be extracted into individual components to be processed. The thresholded image will first be morphologically closed to merge small details and cards that are close in proximity together and then dilated to merge all nearby cards together to form a hand. The morphed image is then subjected to a contour finding algorithm provided in the extensive OpenCV library called cv2.findContours. The detected contours will then be drawn by implementing the cv2.drawContours function to provide visual representation of the contours as shown in Figure 4.

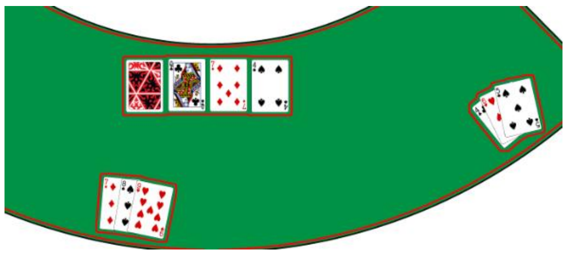


Fig. 4 Detected contours

The contours not only serve as a base for future card processing operations such as card segmentation, perspective correction and rank counting but also allow us to obtain the total player count and player type as each detected contour is regarded as an individual player. As figure 4 shows, there are contours detected that are not player card hands. To solve this, average detected contour area size is calculated, Contour with areas larger than the calculated average area will be ignored, leaving only player card hand contours as shown in figure 5.

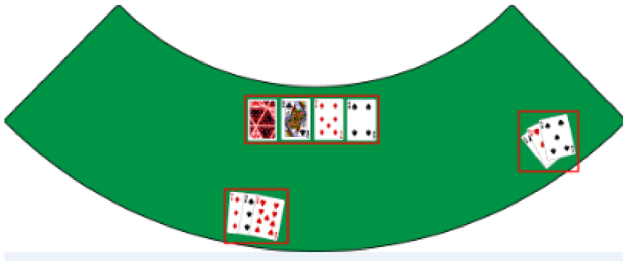


Fig. 5 Bounding rectangle of player's card hands only

A bounding rectangle will be created to enclose each contour. The bounding rectangle will provide the dimensions of each player hand segment that could be used for determining the player type. As dealers have their cards opened without occlusion, the hand segment of dealers will be of larger width compared to regular players. This allowed the system to recognize which hand belongs to which type of player.

C. *Card Segmentation*

Each player contour is then cropped into individual segments based on the dimensions of its respective bounding box. The cropped segments are then converted to grayscale and thresholded as well. The binary image is then subjected to erosion to have the overlapped cards to be distinctly separated from one another as shown in figure 6. This will be the basis of our card segmentation.

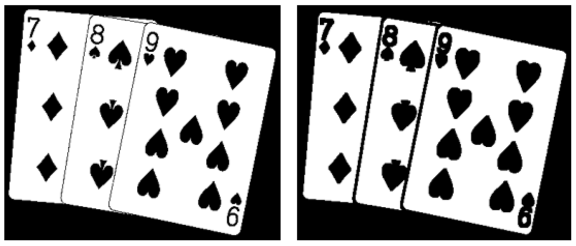


Fig. 6 Before and after erosion of binary image

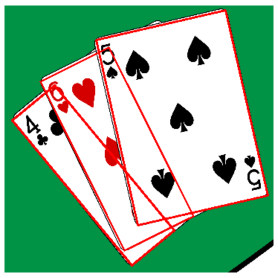


Fig.7 Minimum area rectangle for detected individual cards

The contour finding algorithm cv2.findContours is implemented once more upon the eroded image. A size threshold is added to ignore smaller contours that might be regarded as cards. This results in individual contours being detected as individual cards within a player's hand. A minimum area rectangle is drawn on each of the detected contours as shown in figure 7. Minimum area rectangle differs from bounding rectangle in the sense that it considers the enclosing object's angle from the perpendicular as well. The angle value it provides will play a huge role in orientation correction.

D. *Orientation Correction*

As the cards will need a consistent vertical orientation to proceed with rank extraction, perspective correction would have to be done on the card contours. As mentioned earlier in the previous section, the minimum enclosing rectangle provided the necessary angle information needed to rotate the card contours to its appropriate orientation.

For each card contour, a rotation matrix is calculated from the obtained angle using the cv2.getRotationMatrix2D function. The rotation matrix is a destination array that has the correct orientation already. The obtained rotation matrix is then use in conjunction with the cv2.warpAffine function to warp the extracted card segment to reflect the rotation matrix's orientation.

To crop the segment, the vertices coordinates of the card segment must have its orientation corrected as well. Utilizing the rotation matrix, a transform is applied upon the vertices coordinates to rotate the vertices points of each card contour to its upright angle using the cv2.transform function. The new vertices coordinates are then stored for cropping out the card segments. The end results can be seen in figure 8.

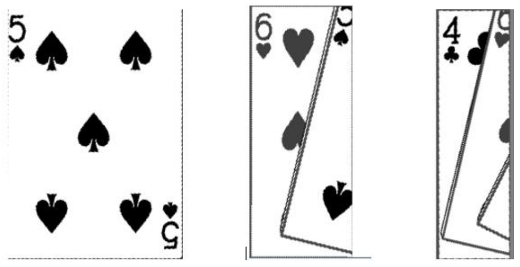


Fig.8 Cropped individual card segments

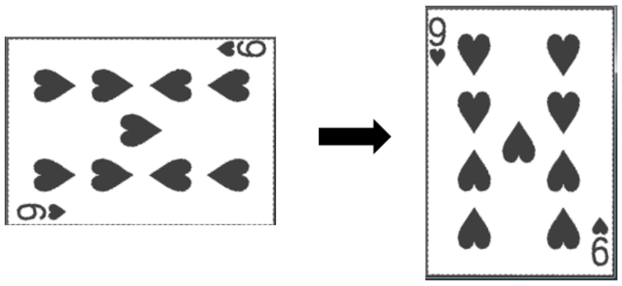


Fig. 9 Erroneous orientation correction is rotated to correct vertical orientation

However, this proposed method does not work perfectly all the time as cards are sometimes rotated erroneously 90 degrees clockwise from the intended orientation. To solve this, the width and height of each rotated card segment was stored beforehand. If the result of the orientation correction yields a width that is larger than its height, it means that the card is horizontally oriented and will then be subjected to a 270 degrees rotation; resulting in a correct vertical orientation as shown in figure 9. Now the individual card segments are ready for the next step.

E. *Template Matching*

Template matching is a method of finding small parts of an image by comparing it with a matching image of similar size, known as template image. We have decided on using template matching as it is a simple, straightforward method that is more than enough for our system needs. This means reduced computational time especially when compared to other matching algorithms such as SIFT and ORB.

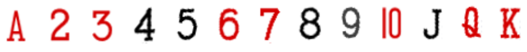


Fig. 10 Template images for card ranks

We implemented template matching by matching the extracted individual cards with an array of suit training images. The template with the best match will return the highest max value. Figure 10 shows the 13 template images used for our system's template matching operation. These template images are stored in an array with the exact same order as in figure 8 for rank counting purposes which will be explained in the next segment.

F. *Rank Counting and Winner Classification*

As there are 13 template images in the array, a counter is implemented to increment by 1 for each of the 13 iterations a card contour goes through template matching. The counter will reflect the value of the ranks in the array as the ranks inside the array are arranged in ascending order based on its value. The counter will return a value of ten for iteration number 11, 12 and 13 as J, Q and K are regarded as rank ten.

As only the best match would return the highest value, a max value threshold is set to only allow matching rates above 80% to register as a legitimate match. After a legitimate match is found, the counter that has been incrementing for each iteration of template matching will have its value appended into a list (list A) which stores all detected ranks of an individual player. If all 13 template images are iterated through and yields no matches, it will be regarded as a covered card and will be appended into list A as well.

Once all cards of an individual player have gone through template matching, the resulting individual list with all rank information will be parsed into a rank counting function that calculates the total detected ranks of an individual as well as returning the total amount of covered cards the individual player has by looping through list A. The calculated total rank and total covered cards are then appended into a list, List B.

Each individual player will have their own corresponding list B. Each list B are then appended into another list, namely list C. List C will be storing all individual player card ranks information. The highest and lowest total card ranks could be determined by iterating through list C.

The highest points possible in a game of blackjack is 21. A hand with more than 21 points will be deemed "burst" and is immediately disqualified. A hand with total rank that is the closest in value to 21 would be deemed the winner. Therefore, our winner classification algorithm determines a winning hand as having less than 22 points, having no covered cards and being the closest to 21 points. If all player hands contain at least one covered card, the hand with the lowest total rank value would be declared the winner as a higher total rank would have a higher chance to "burst" after unveiling a covered card.

G. *Testing*

Our system will have to go through two testing phases with two distinct categories for each phase. The categories being high resolution images and low-resolution images. The steps for the tests conducted will be the same for each category. We used a total of 4 images comprising of 2 high resolution images and 2 low-resolution images for the tests. Lastly, the system's execution time will be recorded for all 4 test images as well. The system is tested on a computer with a quad core Intel i5-3570K processor.

Phase 1 – Rank detection accuracy:

As the rank counting accuracy of the system correlates directly to the rank detection accuracy, rank detection rate will be chosen for testing. The system will run and return detected cards of each player within an image. The results will be presented in percentage form and the accuracy rate will be calculated as follows:

$$Rank\ detection\ accuracy = \frac{no.\ of\ correct\ detection}{total\ cards\ in\ image}\ x\ 100\%$$

Phase 2 – Winner Classification

The system will run and return the winner of the black jack game based on the input image. The results will answer the following criteria:

- Actual winner of black jack
- Detected winner of black jack
- Result

The results for the tests will be tabulated and discussed in detail in the next segment.

V. Results and Discussions

The test results are recorded and tabulated into two tables, each representing its respective category. Each of the tables are further split into two tables, one for rank counting accuracy results and another for winner classification results. In addition to that, each category would have a table for program execution time results for each test image as well. There would be 6 tables in total.

A. *High Resolution Images Results*

TABLE I. Rank Detection Accuracy Rate

Input Image	Rank Detection Accuracy		
	Total correct detection	Total cards in image	Results
1	10	10	$\frac{10}{10}\ x\ 100\% = 100\%$
2	14	16	$\frac{14}{16}\ x\ 100\% = 87.5\%$
		Average:	93.75%

TABLE II. Winner Classification results

Input Image	Winner Classification Results		
	Actual Winner	Detected Winner	Result
1	Player 2	Player 2	Successful
2	Player 4	Player 4	Successful

TABLE III. Program Execution Time

Input image	Program Execution Time (seconds)
1	0.982
2	0.89

B. *Low Resolution Images Results*

TABLE IV. Rank Detection Accuracy Rate

Input Image	Rank Detection Accuracy		
	Total correct detection	Total cards in image	Results
1	8	10	$\frac{8}{10}\ x\ 100\% = 80.0\%$
2	9	16	$\frac{9}{16}\ x\ 100\% = 56.25\%$
		Average:	68.13%

TABLE V. Winner Classification results

Input Image	Winner Classification Results		
	Actual Winner	Detected Winner	Result
1	Player 2	Player 2	Successful
2	Player 4	Player 2	Failed

TABLE VI. PROGRAM EXECUTION TIME

Input image	Program Execution Time (seconds)
1	0.358
2	0.279

C. Discussion

Based on tables I, the results show that our system's ability to detect card rank, and at the same time count ranks, is at a 93.75% rate of accuracy in high resolution images. However, for low resolution images as shown in table IV, the average accuracy rate is only 68.13%. This is extremely evident that the blurriness and lack of definition in low resolution images has severely impacted the accuracy of our algorithm even after image sharpening techniques have been applied (refer to image preprocessing section). If the source image presented are of low resolution or low quality, the extracted card segment will yield an inconclusive match during template matching, which contributes to the lowered accuracy rate.

There appears to be a dip in accuracy in general for input image 2 with a 12.5% drop in accuracy compared to input image 1 for high resolution image category and a significant drop of 23.75% compared to input image 1 for low resolution image category. This was later found to be a card contour error in one individual player's hand during the card segmentation process of the system. As card segments are extracted from a cropped image containing only an individual player's hand (figure 11), there appears to be non-card contour to be detected as one. This non-card contour is then cropped based on its minimum area rectangle (figure 12) and template matching was operated upon the erroneous minimum area rectangle section.

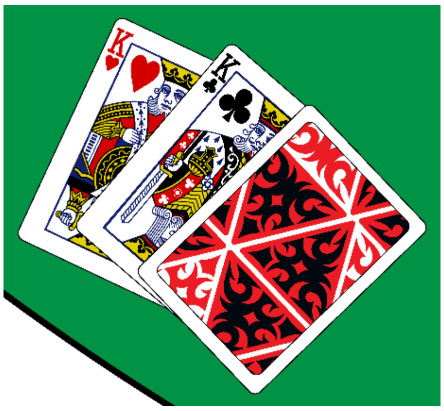


Fig. 11 Cropped segment of individual player hand

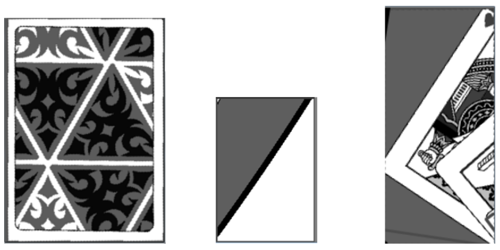


Fig. 1 Erroneous cropped minimum area rectangle section

For winner classification, table II shows our system has successfully determined the actual winner of a black jack game in both instances for high resolution images. Table V shows that our system is not as satisfactory in determining a winner for low resolution images compared to high resolution images, with 1 successful attempt and another failed attempt. This is due to the low rank detection accuracy in low resolution images as mentioned before. Even though the winner classification for the first input image in low resolution images category returned the correct winner, the system returned incorrect total rank values for most of the player hands except for the dealer. It was by sheer chance that the returned ranks have led to the same detected winner as the actual winner. Therefore, it can be said that winner classification algorithm in our system for low resolution images are extremely unreliable. That aside, overall, our proposed system returned a rate of 75% accuracy for winner classification in a game of black jack based on 4 image tests.

Based on tables III and VI, it is evident that our system's efficiency is relatively high. For high resolution images, our test results show that the time taken to process input image 1 and 2 are below 1 second, being 0.982 and 0.89 second respectively. As input image 1 has more playing cards and players compared to input image 2, the increase in processing time is to be expected. The same trend in execution time can be observed in low resolution images tests as well, with 0.358 and 0.279 second for input image 1 and 2 respectively. The vastly reduced processing time for low resolution images are in line with our expectations as there are significantly less pixels in the low-resolution images compared to high resolution images.

VI. Conclusion

In this paper, we have proposed an algorithm for a system to be able to count the number of players and determine player type in a game of black jack from an input image reliably. We have also proposed a rank identification algorithm capable of identifying and counting card ranks with the presence of occlusion at an accuracy rate exceeding 90% for high resolution images as well as being able to determine a winner in a game of black jack in a an efficient manner. However, there are flaws in our algorithm that have led to occasional decrease in accuracy for some cases due to the inconsistency of the card detection algorithm. Rank detection accuracy results for low resolution images left much to be desired as well. Thus, future enhancements to the card detection method must be done in order for the system to approach higher accuracy rates especially in low resolution images.

References

[1] M. Paulo, P. R. Luís and T. Luís, “Poker Vision: Playing Cards and Chips Identification based on Image Processing,” [Online]. Available: https://pdfs.semanticscholar.org/0077/07dded08ae36b301e35734d8136c13e821fc.pdf. [Accessed 3 May 2018].

[2] B. Dan and W. Hugh, “Texas Hold’em Hand Recognition and Analysis,” [Online]. Available: https://stacks.stanford.edu/file/druid:yj296hj2790/Brinks_White_Texas_Hold_Em_Hand_Recognition_and_Analysis.pdf. [Accessed 2 May 2018].

[3] C.B.Zheng and G. Richard, “Playing Card Recognition Using Rotational Invariant Template Matching”, [Online]. Available: https://pdfs.semanticscholar.org/d064/6c9b7c8111e1c7ed5b34827324c322c52638.pdf. [Accessed 3 May 2018]

[4] [4] J.P Pimentel, “Machine Vision in Casino Game Monitoring”, [Online]. Available: https://fenix.tecnico.ulisboa.pt/downloadFile/395142223319/resumo.pdf. [Accessed 5 June 2018]