An automated approach for Billiard balls recognition problem

Tan Ngoc Pham
University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
19520925@gm.uit.edu.vn

Nguyen Thi Minh Phuong
University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
19522065@gm.uit.edu.vn

Dzung Tri Bui
University of Information Technology
Vietnam National University
Ho Chi Minh City, Vietnam
19521386@gm.uit.edu.vn

***Abstract*—Automation in billiard games, i.e. labour-free billiard referees or robotic billiard supervisors, is still a limitation despite the era of Machine Learning nowadays. In this work, we propose an end-to-end approach to detect and classify billiard balls using diverse image processing techniques and multiple Machine Learning algorithms.**

***Index Terms*—Billiard balls recognition, Machine Learning, Image processing**

I. Introduction

Billiard is now a fascinating sport throughout the world and has got its name as a new category in the 2024 Olympics game. Additionally to its amusement on playing, the motion law of billiard balls is also an interesting subject in research that catches eyes on many professionals in Mathematics and Computer Science. However, self-operating billiard games is still a limitation despite the rise of Machine Learning nowadays.

Throughout this report, we would propose an end-to-end method to detect and classify billiard balls with image processing and Machine Learning algorithms. This is the basis problem to deploy on larger-scale automated system to support in later applications. Our problem takes an input of the 8-pool ball billiard table and gives out an output of the original image with detected and classified balls (Figure 2). This is a 16-label classification problem with respect to 16 different types of balls (from 0 to 15). Each ball is labeled as its own numbers. The cue ball is marked as 0 since it has no number on it.

Fig. 1. All billiard balls in the billiard game

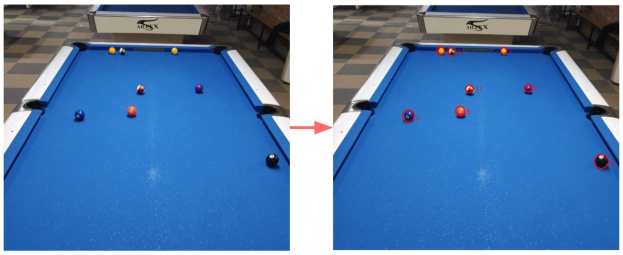

Fig. 2. An illustration on our problem

II. Dataset

A. Data collection

We have collected 120 photos with billiard tables attached from two sources which are in real life (41 photos) and screenshots from billiard playing videos on Youtube (79 photos).

Fig. 3. Our collected dataset
Taken in real life (left), screenshot from Youtube (right)

Constraints: We have set some constraints for the input images while collecting data which are 4 corners of the table have to be clear, photos must be taken in a good light condition to avoid the color changing in balls, the shooting perspective has to look down from the top to keep away from balls' closely grouping, and the distance from the shooting position to the table has to be just sufficient (not too far and not too close) to make the balls homogeneous in their sizes.

B. Data labeling

In order to evaluate the accuracy of the Machine Learning system, we conduct labeling our data. Our labeling tool that we use is the *labelImg* [1]. We use this tool to draw bounding

boxes outside the billiard balls in the photos to save the position of those balls, then we attach to them a correct class corresponding to the groundtruth label. The final result after having been labeled images is a file in which contains n rows with respect to n billiard balls that appears in the image, and every row in n rows contains information about the correct label, the central coordinates, height, and width in YOLO format.

Fig. 4. Demonstration on data labeling process using labelImg tool

III. Method

In the section below, we will talk over on how we propose to approach this problem. Our problem is defined as receiving an input of a photo and returning the result of that input photo but with detected and classified billiard balls. Overall, we divide our big problem into four primary phases which are detecting billiard table, detecting billiard balls, extracting features, and classifying billiard boards. Our full approach is demonstrated in Figure 5.

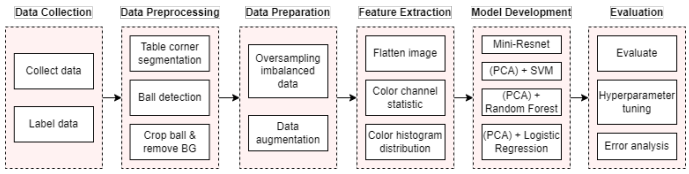

Fig. 5. Step-by-step on detecting billiard balls

A. Billiard table detection

In order to increase the precision in detecting billiard balls, as well as locating billiard balls better for the classification phase, we propose to use image segmentation and quadrilateral estimating to segment billiard table so as to remove background outside the segmented area.

Image segmentation: Billiard tables normally have green or blue color with different properties on brightness and saturation. Hence, we conclude that image segmentation based on *Hue Saturation Value* (HSV) color model [2] works better and more stable than *Red Green Blue* (RGB) color model does. Simple image segmentation method often limits blue and green color range based a defined range of colors on HSV channels.

Let denote the color value of the i-th pixel in an image as $p_i = (h_i, s_i, v_i)$, and the respective minimum and maximum value at each color channels in HSV model as $h_a, h_b, s_a, s_b, v_a, v_b$. The image segmentation algorithm would create a new binary image by the following equation:

$$p'_i = 1 \Leftrightarrow \begin{cases} h_a \leq h_i < h_b \\ s_a \leq s_i < s_b \\ v_a \leq v_i < v_b \end{cases}$$

For a much easier approach, we choose to narrow down the billiard boards colors into green and blue only. On green billiard boards, we setup the Hue range as $h_a = 80, h_b = 130$ and $h_a = 40, h_b = 70$ on blue ones. The Value parameters could also be set as $v_a = 60, v_b = 255$ since there is a similarity on the range of brightness in a typical image from our defined constraints.

However, the saturation range is different between objects in any images and it is not good settling down a hard threshold on the Saturation value. We apply histogram statistical method and billiard table's color distribution estimating based on *Gaussian Mixture Model* (GMM) [3]. GMM could be simplified as a generalizing *K-means clustering* [4] to embrace information on the data's covariance structure and the latent Gaussians centric points [5]. In details, the input image would choose significant pixels after being segmentated on the above Hue and Value threshold. Then, we count and do statistics on the set of segmentated pixels to get the histogram of the blue ones. We could use GMM to estimate the distributions if colors in a photo appear as a set of Normal distributions. We use $k = 2$ for Normal (Gaussian) distributions in this context and we choose the rightmost distribution in the histogram with the fundamental that billiard tables often have darker blue and more noticeable than the foreground.

Quadrilateral estimation: The most important and decisive step in the table detection is to identify the billiard table's corners. We use *Suzuki algorithm* [6] to find the appropriate borderlines to approximate the segmentated billiard table. Then, we use *Sklansky algorithm* [7] to do the convex hull towards the billiard table and *Douglas-Peucker algorithm* [8] to estimate the selected polygons into a quadrilateral with a minimum difference of ϵ between the original polygons and the detected quadrilateral.

B. Billiard balls detection

Billiard balls have to be detected as correctly as possible before moving to the classification step and should there be any mistakes, i.e. detecting holes instead of balls, our classification algorithms would not work effectively. We define the billiard balls detecting problem as receiving an input of the billiard board with the coordinate of four corners and an output of detected balls coordinates.

Preprocessing: We propose a way to keep the surface of the billiard table only by removing the background of the original image using the tables' four corners coordinates. This would help our algorithm avoid round but non-ball objects.

Ball detection: We use *Hough transformation* [9] to detect round objects, or billiard balls in this context, and only detected balls are then classified.

C. Feature extraction

Detected billiard balls before being classified have to be feature extracted first and performance from classification phase depends a lot on this step. We propose three different feature extracting methods throughout this work which are Flatten, Color channel statistic, and Color histogram distribution. The reason on using two later methods is based on the fact that the primary difference among to-be-classified billiard balls is the color feature (Figure 1) rather than the others, i.e. corner features.

Flatten: Flattening is a casual technique that are used in almost every Computer Vision problems. An elementary understanding of flattening is that we transform our two (grayscale image) or three (RGB or HSV image) dimensional image representative matrix into a single dimensional vector. And that vector is used as the feature vector of our image.

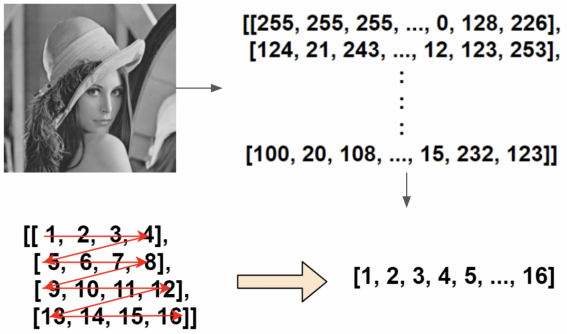

Fig. 6. Flattening procedure

Color channel statistisc: Using mean and standard deviation value from each color channel then change them into a feature vector is the fundamental of color channel statistics. The method counts the frequency of different colors (red, green, blue in the RGB color system or hue, saturation, value in the HSV color system). Then, a feature vector v is created using the mean and standard deviation based on the frequency, or $v = [\mu_{c1}, \sigma_{c1}, \mu_{c2}, \sigma_{c2}, \mu_{c3}, \sigma_{c3}]$ such that $c1, c2, c3$ is the ordinal color from the present color system.

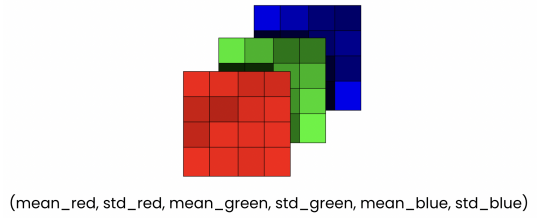

Fig. 7. An example of color channel statistic method

Color histogram distribution: Unlike Color channel statistics that only counts mean and standard deviation value from image pixels, Color histogram distribution utilise histograms to statisticize different color values as well as their frequencies and then a histogram is built. The histogram image is then flattened and becomes the feature vector for that image.

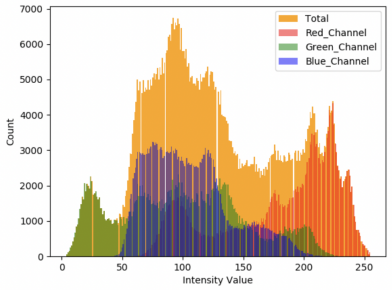

Fig. 8. An example of color histogram

D. Billiard balls classification

We choose to use different approaches on classifying billiard balls in order to give out a minor comparison on performance among various Machine Learning algorithms and Deep Learning architecture. In this work, we utilise *Support Vector Machine* (SVM) [10], *Random Forests* (RF) [11], *Logistic Regression* [12] and miniResNet [13].

Logistic Regression: Back into the early days of Machine Learning, there are not many algorithms that are able to handle both regression and classification problems. Logistic Regression, also known as Logit Regression, is one of the scarce algorithms that could be use to estimate the probability of an instance belonging to one class and thus identify the truth label for that instance.

The fundamental below Logistic Regression is elementary. It works the same way as Linear Regression does which is to compute a weighted sum of the input features with a bias coefficient, but it output the *logistic* of the result instead of a direct result as in the Linear does [14]. The Logistic Regression's estimated probability for an instance is demonstrated as:

$$\hat{p} = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

where $\sigma(\cdot)$ is a sigmoid function to create the probability for the output and $\sigma\left(t\right) = \frac{1}{1+e^{-t}}$.

Support Vector Machine: Support Vector Machine, or SVM, is one of the most powerful and versatile Machine Learning model which could be capable of performing both linear and nonlinear classification, regression, and outlier detection [14]. The basic fundamental behind SVM is to predict a decision boundary that separates classes among each other, and this boundary not only divide classes but also stay as far from the closest training instances as possible. In soft margin case, the additional objective for the SVM is to find a good balance between restricting the margin violations and obtaining the largest possible margin.

Random Forests: Ensemble learning is built with the idea that a solution given by a crowd is usually more accurately than that of a single person. And Random Forests is a supervised learning algorithm constructed with the decision-tree-based ensemble learning method. RF often consists of a set of many Decision Trees [15] and those trees are trained with bagging (or pasting) techniques. In RF, each instance is

evaluated through many Decision Trees, then the best value is chosen to be the final result of the model. Hence, more trees would help us predict better and avoid overfitting better. However, more trees also means more training time to take.

MiniResNet: To cope with the situation that deep neural networks meets a huge difficulties in the training phase, *deep Residual Networks* (ResNet) were proposed to make the training process easier and faster but still maintain a considerable accuracy on testing phase [13] thanks to the skip connection layers. miniResNet, or ResNet18, is a smaller version of the vanilla ResNet which is 18 layers deep only. Thanks to this shallow architectures, it could be trained on our networks without being largely overfitted.

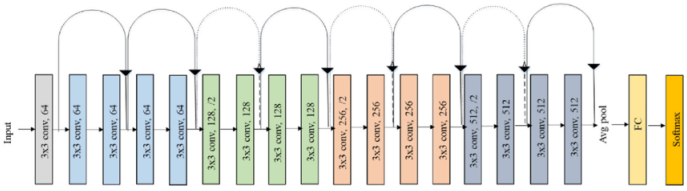

Fig. 9. miniResNet's architecture

IV. Experiments

In this section, we would discuss further on how we setup our experiments to compare among different methods on classifying billiard balls.

A. Metrics

We consider a primary metric on measuring how well our algorithms perform which is *F1 macro score*.

F1 macro score: F1 score could be understood as the harmonic score between the precision and recall (or the precision/recall tradeoff), in which the minimum value of F1 score is 0 and the maximum is 1. Whereas, the regular mean treats precision and recall equally which leads to the fact that the higher F1 score, the higher values from both recall and precision. F1 score is represented as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{\text{FN} + \text{FP}}{2}} \quad (1)$$

s.t. TP, FN, and FP are True Positive, False Negative, and False Positive, respectively.

F1 macro score is derived from F1 score but with a minor substitution that it does not take imbalance into account by calculating metrics for each label and find their unweighted mean [14].

B. Implementation details

Data preprocessing: In this step, we would process the input image of a billiard table photo, and give out an output of cropped billiard balls with the following procedures. Firstly, we would use segmentation technique to segment the billiard table and remove background. We then apply detection method to detect billiard balls on the segmentated tables. And finally, we would use coordinates and radius from identified balls at the previous step to crop balls into independent ones, then resize them into the 32×32 resolution as well as remove the balls' background which is to help classification models not be affected by the color of the table.

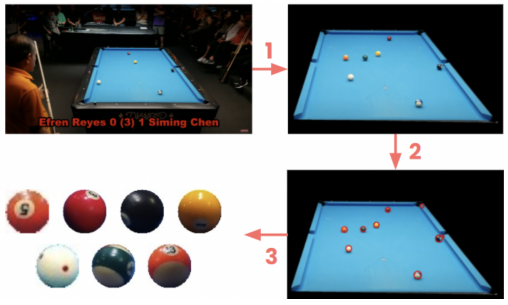

Fig. 10. An illustration on data preprocessing step

After pre-processing on 120 images in our dataset and putting balls into their respective folders, we preliminary evaluate about distribution on the number of billiard balls in the dataset. The distribution is shown as in Figure 11. The categorical number on each class is not uniformly distributed which ranges from 34 to 106. Hence we got an imbalanced dataset.

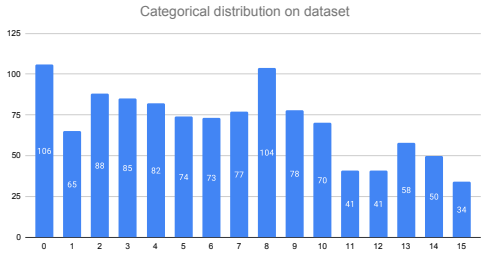

Fig. 11. Distribution on balls' categories.

Data preparation: Before training our models, we divide our dataset into train set and test set with 80% for training and 20% for testing. Oversampling and augmentation are applied to resolve the problem of imbalanced data and to get more effective machine learning models. The augmentating operations we used in this context are vertical flip, horizontal flip, rotation in range 10, and shear in range 0.2. Besides, the oversampling method we applied is to randomly duplicate instances from the minority classes and add them to the train set.

Having finished augmentation and oversampling, we got 3 different train set for the training phase which are (a) the original train set, (b) oversampled train set, and (c) train set with augmentating and oversampling. Detailed distribution is shown in Figure 12.

Experimental settings: We would use a forementioned supervised learning which are SVM, RF, Logistic Regression, and *Principal Component Analysis* (PCA) [16] to enhance Machine Learning models with three feature extraction methods running on three independent training datasets settings: raw

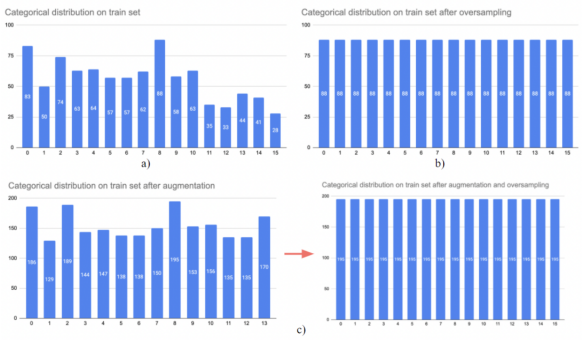

Fig. 12. Distribution on different train sets.
(a) the original train set, (b) train set with oversampling, and (c) train set with augmentating and oversampling

(train set (a)), oversampling (train set (b)), and oversampling with augmentation (train set (c)). F1 macro score is utilised as the primary metric for this work to compare and rank our models since this is an appropriate metric for imbalanced problems. After conducting the above experiments, we would pick out the best performance within our experiment and do hyper-parameters tuning using Grid Search method [17]. Besides, we also use a Deep Learning architecture that is miniResNet to compare with the supervised algorithms.

Hyper-parameters: The hyper-parameters that we have used to tune the model with the best performance, which is Random Forests, are shown in Table I.

TABLE I
HYPER-PARAMETERS VALUES USING IN RANDOM FOREST

Hyper-parameters	Values
class_weight	'balanced'
criterion	'gini'
max_depth	32
max_features	'auto'
min_samples_leaf	2
min_samples_split	6
n_estimators	500

C. Results

The results obtained in our experiments are demonstrated in Table II.

As we can see, Random Forest and color histogram distribution are respectively the best model and the best feature extracting method throughout our work with the superior performance on F1 macro score.

Principal component analysis helps Support Vector Machine, oversampling data helps Support Vector Machine and Logistic Regression improve its final accuracy, but they makes other algorithms become weaker.

Augmentation reduces performance of our models. We hypothesize that augmentation operations using in this work is not strong enough to make the original dataset become distinguishable. Thus, our models become overfitted.

On the contrary to our initial hypothesis, Deep Learning using in this work could not be a superior method. The F1 score from miniResNet is only relative compared to other supervised learning methods, and it is even worse than some of those. We think that our datasets are not diverse enough and the miniResNet is deeper than our needs for this problem. Hence, miniResNet could not perform its power completely.

D. Error Analysis

We have analysed errors on our models by observing the confusion matrix in Figure 13 and the folder containing classified results. There are certain errors in our approaches which are balls with the same colors but different stripes, e.g. balls number 1 and 9 are often misclassified, or stripes with white color towards the camera have tendency to be misinterpreted as ball number 0 (cue ball).

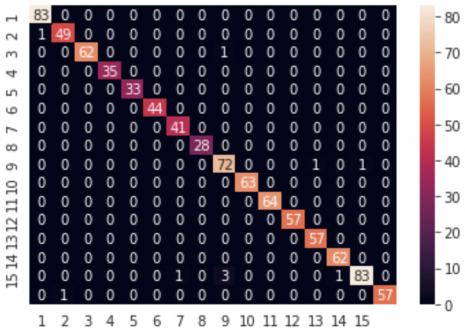

Fig. 13. Confusion matrix

V. CONCLUSIONS

In this report, we used image processing techniques to detect billiard balls from a photo containing a billiard table and classify billiard balls into their respective classes using Machine Learning algorithms. Despite current limitations and incomplete techniques, among all of our proposed approaches, Random Forest and color histogram distribution is experimentally by far the best method for this problem.

VI. FUTURE WORK

For the time being, we would explore and find more data to reinforce our problem better, as well as make our models achieve a more superior performance on the testing phase. Besides, we would also try more appropriate Deep Neural Networks towards our problem to avoid overfitting as in our work. And finally, we would look for some of more advanced feature extracting techniques than what we had used in the future.

ACKNOWLEDGEMENTS

The authors would like to thank the support from PhD. Ngan Luu-Thuy Nguyen, MSc. Hao Duong Ngoc, MSc. Duc-Vu Nguyen, and MSc. Son T. Luu on teaching us Statistical Machine Learning as well as supervising us on this project.

VII. WORK ASSIGNMENT

Our work assignment is provided in Table III.

TABLE II
EXPERIMENTAL RESULTS

Dataset	Model	Flatten	Color channel distribution	Color histogram distribution
Dataset (a)	SVC	0.6633	0.5763	0.6873
	PCA + SVC	0.7507	0.5614	0.7550
	Random Forest	0.7517	0.7607	0.8323
	PCA + Random Forest	0.6974	0.7615	0.8313
	Logistic Regression	0.7656	0.5435	0.5397
	PCA + Logistic Regression	0.7892	0.5051	0.5397
	Mini-Resnet	0.76106		
Dataset (b)	SVC	0.7556	0.6105	0.7794
	PCA + SVC	0.7736	0.6070	0.7931
	Random Forest	0.7550	0.7365	0.8290
	PCA + Random Forest	0.6575	0.7111	0.7928
	Logistic Regression	0.7236	0.5458	0.6310
	PCA + Logistic Regression	0.7400	0.4916	0.6381
	Mini-Resnet	0.72123		
Dataset (c)	SVC	0.6937	0.5987	0.7884
	PCA + SVC	0.6812	0.5692	0.7902
	Random Forest	0.6602	0.5944	0.7882
	PCA + Random Forest	0.5136	0.6713	0.7510
	Logistic Regression	0.6012	0.4922	0.6832
	PCA + Logistic Regression	0.5767	0.5112	0.6698
	Mini-Resnet	0.54867		
Grid Search		0.8514		

TABLE III
WORK ASSIGNMENT

Student ID	Name	Tasks
19522065	Nguyen Thi Minh Phuong	- Collect and label data. - Preprocess data (detect billiard table, balls). - Make slides, present, and write report.
19520925	Pham Ngoc Tan	- Prepare datasets (augmentation, oversampling data). - Extract features with three approaches. - Build miniResNet model. - Make slide, present, and write report.
19521386	Bui Tri Dung	- Train multiple models (SVM, Random Forest, Logistic Regression, PCA). - Tune hyperparameters, analyze error. - Make slide, present, and write report.

REFERENCES

[1] Tzutalin, "Labelimg," Free Software: MIT License, 2015. [Online]. Available: https://github.com/tzutalin/labelImg

[2] "Hsv color space," in *Encyclopedia of Microfluidics and Nanofluidics*, D. Li, Ed. Boston, MA: Springer US, 2008, pp. 793–793. [Online]. Available: https://doi.org/10.1007/978-0-387-48998-8_656

[3] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds. Boston, MA: Springer US, 2009, pp. 659–663. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_196

[4] X. Jin and J. Han, "K-means clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[7] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognit. Lett.*, vol. 1, pp. 79–83, 1982.

[8] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international journal for geographic information and geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[9] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: https://CRAN.R-project.org/doc/Rnews/

[12] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[14] A. Geron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017.

[15] W. A. Belson, "Matching and prediction on the principle of biological classification," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 8, no. 2, pp. 65–75, 1959.

[16] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[17] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.