Chess Piece Location Detection Using 3D Depth Information

Maxmillan Ries
8th February 2021

Abstract—Typical chess board-state recognition algorithms employ 2D RGB images for the detection of piece locations. Recognizing chess constellations using 3D point cloud information is an intriguing problem, which can serve to explore new or alternative approaches in computer vision. This paper compares, in this context, the performance between two 3D point cloud analysis solution strategies for chess board-state recognition. The two strategies considered in this paper are a segmentation-oriented approach and a clustering approach. The segmentation approach independently considers each chess cell and uses the data contained within the cell to determine if a piece is present. The clustering approach forms clusters from the 3D data, each representing a piece. It then maps the center of mass of each piece cluster as its location on the board. The presented work encompasses a rigorous analysis of accuracy and sensitivity as well as an overview of the optimization process.
The results show a superior performance of the clustering approach compared to segmentation. The best segmentation algorithm only successfully evaluated 49% of the chess boards, while the best clustering algorithm successfully evaluated 77% of the boards in our benchmark data set.

1 Introduction

The development of chess-playing automata has a rich history of over 350 years of human fascination, which has inspired several ground breaking innovations in computing science and control theory [1]. This paper evaluates two different autonomous pattern recognition algorithms for chess board-state detection. Both algorithms are based on 3D point cloud analysis derived from information of a stereoscopic 3D-camera. The work is part of a larger project developing a robotic chess-playing arm, which is able to play autonomously against a human opponent without any additional support/intervention. This encompassing project started as an extra-curricular venture, with the intent to learn and develop an interest in the field of robotics and image processing.

In 2003, David Urting and Yolande Berbers released a paper about MarineBlue, a Low-Cost Chess Robot [2]. In this paper, Urting and Berbers state that to create a chess-playing robot, three components are required: a robot control component, a chess engine component, and a computer vision component. The chess-playing robot was built with these components in mind.
To ensure that all aspects required are developed, the project was broken down into Urting and Berbers's three components. The *robot control component* of the project, represented by the Servo Control and Kinematics component in Figure 1, consists of maintaining serial communication with a series of servo motors, and calculating the inverse kinematics to allow the robot to maneuver with a full 6 Degrees of Freedom (DOF). This component was created with the aid of Peter Corke's Matlab Toolbox [3], which contains functions and classes used to represent orientation, poses in 2D and 3D space, and the full inverse kinematics solver for a 6 DOF robot arm.
The *chess engine component*, represented schematically in Figure 1, aims to provide the required functionality for analyzing the board state and for the calculation of the appropriate counter moves, as well as an opening library for a robust game-start. The state of the board is locally maintained in a Matlab class, and subsequently transferred to the open-source Stockfish Chess Engine through the use of a Matlab-Stockfish interface [4]. Similarly, the counter-move is retrieved and passed to the robot control component for the physical execution, as shown in Figure 1.
Finally, the *computer vision component* is the principal subject investigated in this project. Contrary to implementations

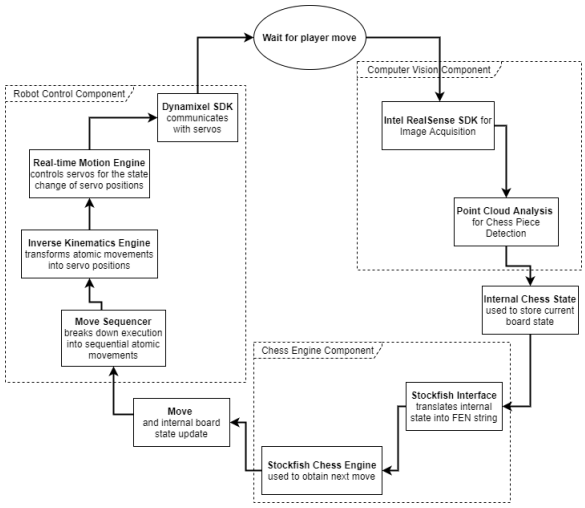

Fig. 1. Fundamentally, the chess robot can be represented as a feedback loop that retroacts based upon the opponent's move. After a move has been executed by the human layer, a depth image is taken and used to extract the chess piece locations on the board. The location information is used to update the Stockfish engine, which is used to provide a counter move against the opponent. This information is passed to the Robot Control Component, which processes the chess move and executes the obtained sequential atomic steps.

which rely on the analysis of RGB images, here a different approach has been chosen: As shown in Figure 2, the robot arm is equipped with a stereo-3D camera (Intel realsense D435), which produces a depth map of the perceived environment. The specific goal of this module is to use 3D computer vision techniques to identify the chess pieces on the board and, together with the internally maintained chess state, sense the move of the human chess opponent, which is fed into the chess engine component as shown in Figure 1. In summary, when the three modules are combined, the robot should be able to function as an independent player, by recognizing the board-state, deciding on a sensible counter-move, and physically executing the move on the real chess board. Figure 2 shows an image of the robot in the "Overwatch" position, which is centered directly above the board parallel to its surface. Since this is the most favorable position to avoid shadow casting figures, all depth maps were taken with the camera in this same position, since this also avoids additional coordinate transformations such as rotations, translations, or noise reduction.

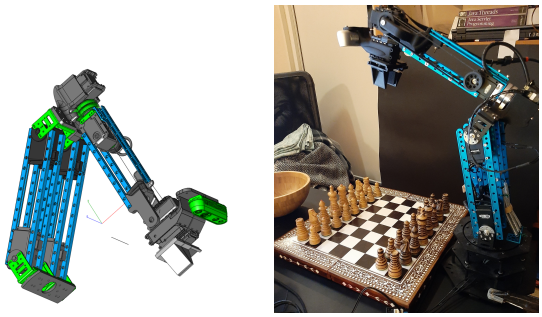

Fig. 2. Starting point for the hardware development of the project was a PhantomX Reactor Robot 5-DOF Arm Kit from Trossen Robotics [5]. Since the original arm had neither sufficient reach to cover the board, nor the required 6 DOF, the hardware setup was changed to a substantially larger variant shown above in both images. The modifications have been inspired loosely by KUKA 6 DOF anthropomorphic industrial robot designs [6] and encompass considerably stronger Servos with PID-position control, a stiffer truss-structure made from aluminium profiles and the implementation of "elbow rotation" as the additional degree of freedom. The left image displays the final computer-aided design of the robotic arm. The right image consists of the final arm, standing in the "Overwatch" position.

In the scope of this project, the following specific recognition task is investigated: Given a depth map directly obtained from the Intel RealSense camera, which chess squares are occupied by a chess piece? The principle challenge occurs when chess pieces are posed in a decentered position, unfavorably close and in tight clustered situations (and any combination thereof).

In this paper, two fundamentally different approaches are analyzed to solve the recognition task: i) a point-cloud segmentation approach, and ii) a point-cloud clustering approach. Furthermore, a set of variations in the implementation of both approaches are discussed, along with the benefits of an additional iterative refinement process.

To evaluate the quality of each algorithm, and its variations, a set of test cases are recorded on a chess board with 37mm squares. Our database of cases contains 96 depth maps containing different board states, which are broken down into three levels of difficulty. Each board is classified by both its state and its difficulty (Easy, Medium or Hard), where Easy represents a chess board with centered pieces, Medium a board with imperfect (i.e. decentered ±10mm) placement, and Hard a board (i.e. decentered ±20mm) which is difficult for even a human to evaluate.

The next section introduces prior works which have been done on the subject of chess board recognition. Section 3 describes the data generation and Section 4 describes the design of each algorithm along with the evaluation procedure. Section 5 describes the quantitative analysis of the results, and Section 6 provides a conclusion and a direction for future work.

2 Related Work

The automatic recognition of the chess board state is of importance in chess tournaments, where games are often digitized both for documentation and broadcasting purposes. Furthermore, the solution to the problem is essential for chess robots that omit the need for the player to enter the executed moves by hand.

As a consequence, this problem has been solved in the past with numerous sensing/analyzing strategies. Apart from vision processing, the simplest and most robust solution is to modify the pieces and the board by the appropriate combination of magnetic/RFID tagging of the chess pieces and integrate the corresponding sensors in the board [7].

Image-based methods are a very commonly used approach for chess recognition due to the wide availability of RGB cameras and the lack of a better alternative. In 2017, Christie et al. released a paper which discussed the use of differences between RGB frames with a given initial state to determine the current state of a board [8]. Following this, Czyzewskia et al. used a neural network on a line-detection based approach to try and solve the same problem [9].

Such approaches are however unfavorable for a robotic arm with an attached camera, since image-based recognition methods often use the differences between consecutive images or frames to detect and classify the chess piece locations.

Neural networks can compensate for these offsets. As a powerful processing technology, neural networks have shown to be very capable in the field of computer vision as a tool to mimic the way a human perceives and learns [10]. This methodology alleviates the need for a controlled environment. Nevertheless, in order to obtain an accurate neural network detector, there is often a requirement for large amounts of training data. This data requires much preprocessing and hand labelling, and can be very tedious to obtain. In 2016, Koray et al. proposed a real-time chess tracking system based on a webcam positioned above the chess board. The analysis of the project consisted of a series of detections, rectifications and adjustments carried out on each RGB video frame [11]. In the same year, SquareOff released a unique chessboard and set of pieces, which used no computer vision tools, and instead relied on a unique magnetic design [7]. These solutions require little to no data to train and efficiently recognize the board's state, but are monetarily expensive and often difficult to design and implement.

Overall, many external improvements can be made to the

board, such as marking pieces or using specific color-coded boards. These improvements however significantly decrease the versatility of the algorithms used. More generic approaches, while allowing a usage across a greater number of chess sets, tend to suffer from environmental effects, such as poor lighting. The balance between accuracy and spread is important, and it is for this reason that several attempts have been made to design dedicated hardware for supporting the detection of chessboards [12], [13].

3 Background

Most of the non-AI based methods focus on using the lighting and color information obtained from an RGB camera. This focus is primarily caused by the lack of availability in technology capable of obtaining other data formats, such as depth information.

Besides the costly LIDAR laser ranging, the first efforts to provide affordable high-resolution optical 3D sensing capabilities started around 2005, when the company PrimeSense proposed an active ranging system based on low cost stereo IR-camera setups, which were subsequently developed into the Kinect product line from Microsoft [14]. Somewhat unfortunately, the first Kinect devices were heavy/bulky for small robotic projects and Microsoft did not provide a convenient Software Development Kit (SDK) for a long time. In 2018, however, Intel released a range of products known as RealSense. These compact and affordable cameras can be used to obtain both RGB frames, along with good quality depth images and fully fleshed out SDKs, which allow interfacing with a variety of computer languages, such as Matlab. In this paper, we attempt to recognize the state of a chessboard using only the depth information obtained from one of these Intel 3D-cameras. This opens up interesting alternatives to the more established RGB-camera based approaches, in particular since the potential of full 3D object detection and 3D point cloud processing can be explored.

The Intel RealSense camera uses an RGB-D pixel system, where each pixel contains both the RGB information recorded, along with a unique depth value [15]. Depth sensing is achieved by a speckle IR-laser projector, which projects a point pattern on the scene and two stereotactic IR-cameras observing the scene. The depth information is calculated from the perspective image offset between both IR-cameras, as shown in Figure 3.

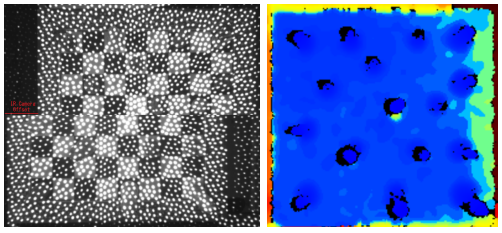

Fig. 3. The processing of depth sensing is done by the Intel RealSense camera by a speckle pattern IR-laser projector and two stereotactic IR-cameras which observe the scene. The image on the left consists of both camera views, with the upper half of the board representing the right stereotactic IR-camera and the lower half representing the left stereotactic IR-camera. Both images are aligned and have been cropped using the exact same parameters, with the red line representing the offset used to calculate the depth image (right).

4 Methodology

4.1 Preprocessing

The preprocessing/calibration step consists of a simple affine translation over the length and width of the board. As the camera is placed parallel to the chess board, no rotation transformations are required, and only a simple translation along the board is used. For this, a specific board layout is used, consisting of a single pawn placed at the very center of the board. By selecting the 3D data of the pawn in the center of the board, and shifting its central vertical axis to XY coordinates of $(0, 0)$, one can use the size of the board to accurately navigate. Saving the transformation parameters, the process can be repeated on any new incoming data, resulting in a clean and usable chess board for analysis.

Figure 4 shows two depth maps of the same board before and after calibration.

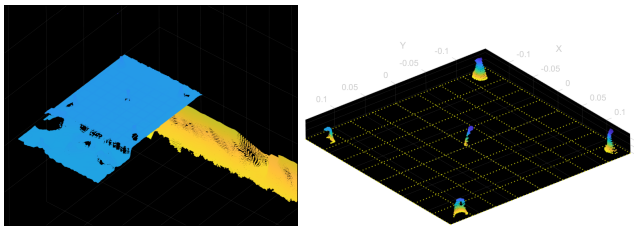

Fig. 4. On the left, the original depth information obtained from the Intel RealSense camera. The calibration is used to crop out of the full 3D-scene, which includes parts of the desk space and the perimeter of the chess-board; only the board area remains. On the right, the cropped point cloud data is displayed. Note that the chessboard's surface has been cropped out of the scene to facilitate clustering of the pieces and to reduce the size of the data.

The image on the left in Figure 4 shows the original depth information obtained, while the image on the right shows the calibrated data, where the board surface itself was removed to isolate the individual pieces. For each piece visible in both images, the missing data is a consequence of the camera positioning above the board, and the shadow-casting caused by a side of a piece occluding another. The grid on the right image was added for a visual representation of the chess grid and the cells used in both the segmentation and clustering approach.

This calibration method, while relying on the external knowledge of the chessboard size or cell size, is very accurate. The image on the right clearly demonstrates that, following the cropping, the central piece is in the center of the grid, and the four external pieces are located inside of a cell, in a similar manner to how they were on the physical board. While this process reduces the generalization capability of the algorithm, it is a minor parameter and the method of obtaining the chess data without external information was not investigated for the scope of this paper.

4.2 Main Approach

The approaches used in this project are conceptually similar to those used with RGB images. Rather than determining the piece at each square, an initial state is provided. With this information, once a move is executed on the board, the algorithm takes the difference between the two board states and determines what pieces have moved.

The representation used in this project to store the board state is a simplification of the Forsyth-Edwards notation (commonly known as a FEN string), which is used by Stockfish and many chess engines today [16]. As the algorithm does not classify the nature of the piece present, but only if a piece occupies the square, a binary FEN string is created, with the symbols for a chess piece replaced with a '1' and the symbol for an empty square being '0'.
Typically, the Forsyth-Edwards represents a series of adjacent empty squares as a single value instead of a series of '0's. As the Stockfish engine internally converts such a series into that singular value, the conversion was not considered in this paper.

4.2.1 Segmentation Approach

The segmentation processing approach consists of evaluating each square iteratively, and independently determining if a piece is present inside the cell or not.

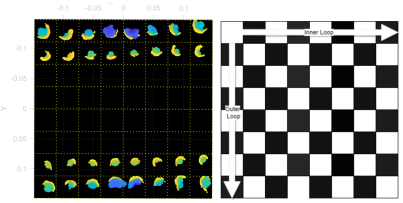

Fig. 5. On the left, the calibrated starting chess board positions along with the X and Y coordinate values. On the right, the order in which chess squares are processed.

The most basic implementation of the algorithm is simple: First, the affine transformation is applied and the data is cropped. Subsequently, the algorithm iterates over each chess cell and decides if a chess piece is present based on the presence of point cloud data in the respective field. The chess fields can easily be iterated over by storing the information pertaining to the length and width of the chess board. In order to maintain a Forsyth-Edwards notation string, the iterations were adjusted to follow the order of the representation, as to minimize the number of string manipulations throughout the project. The order of the iterations is described in the right picture of Figure 5.
An improvement to the basic algorithm would aim to resolve any ambiguity caused by pieces overlapping onto multiple squares. A simple and direct method is to introduce a shrink offset, which for every cell, dictates how much of the data should be taken into consideration. As the chess cells are rectangular, the simple approach consists of subtracting a fixed offset from each side of a cell. This method, in combination with a threshold for the minimum amount of data required for a piece, aims to reduce the ambiguity described above. Segmentation in combination with these refinements are analyzed in Section 5.1 of this report.
An additional improvement consists of using the chess board information to determine the position of the closest piece of data to the center of the current iterated square. By setting a distance threshold, the overlap can further be reduced. However, this method reduces the generalization of the segmentation algorithm as the minimum distance is manually set using the chess cell size, and is hence tied to the specific chessboard used. As such, this improvement will not be evaluated.

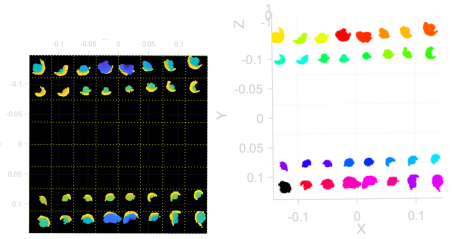

Fig. 6. On the left, the calibrated data of a standard chess starting position. Note that the color coding represents depth information (blue closer to the camera, yellow further). On the right, a color-coded image with the results of a simple clustering using a minimum euclidean distance after flattening the data. Here, the color of each 3D point indicates the membership of each respective cluster.

4.2.2 Clustering Approach

Unlike the segmentation approach which aims to interpret the data present on each square independently, the clustering approach takes a holistic look at all the available information. The fundamental principle is that the 3D-point cloud can be split into one clearly distinguishable sub-cloud for each present chess piece. The goal of the clustering algorithm is to cluster, using a minimum Euclidean distance, the entirety of the data, before finding the central vertical axis of each cluster and assigning it to a chess square. While the algorithm is more complex and computationally demanding than the segmentation approach, it does not selectively decide based on independent cases. As such, the benefit of this approach is the lack of overlap. Since clusters are formed based on a minimum Euclidean distance, the overlap of a piece on another square is predicted to have little or no effect.
A very noticeable and predictable problem with the clustering process is the lack of data for a piece. Pawns, Rooks and Bishops have very simple shapes and are 5 cm tall, while the pieces for the King and Queen are 9 cm tall. Additionally, the King and Queen pieces have rounded bulbous tops, which shadows lower parts of the figure from laser illumination due to the top-down perspective of the camera. There are two solutions to this problem. The first solution consists of removing the data present near the base of the pieces. The second solution consists of temporarily eliminating the height information, and clustering uniquely over the XY spread of the data.
Although both methods are viable, the first method removes data from the point cloud. While this may initially appear trivial, this problem is in practice considerable: In cases of proximity between multiple pieces, the shadow-casting caused by the fixed perspective may cause a crucial amount of data to be removed. Additionally, due to the multitude of clusters located along the same height axis, the process is less efficient than the second solution as these clusters need to be fused together.

Improvements which aim to improve the clustering method consist of either reducing the likelihood of pieces being clustered together, or of breaking down fused clusters into the respective components. For the former

improvement, a simple slicing of the data is sufficient. Rather than considering all data obtained from the depth map, a fraction of all data can be used, while for the latter improvement, an iterative process needs to be implemented. The basic clustering approach, along with the non-iterative improvements are discussed in Section 5.3.

4.2.3 Iterative Clustering

The idea of the iterative process is straight-forward: Given an initial clustering without knowing the number of pieces, the fused piece clusters need to be broken down as accurately as possible. The simplest method for doing so consists of reducing the minimum euclidean distance required to fuse pieces into a cluster. This process, however, cannot be applied to an entire board, due to the data error margin present from the camera and the missing data caused by the perspective choice.

As such, the iterative process iterates over each cluster obtained from the initial clustering and uses either a size threshold or the piece length information to determine whether or not a cluster consists of two or more pieces. In both cases, the deduction comes with a risk. With the former idea, the missing data caused by overlapping pieces and perspective blockage causes the amount of information per piece to be very irregular. Setting a fixed threshold is difficult, as it is equally likely for a King or Queen chess piece to hold as much data as three clustered pawns combined. With the latter idea, similar to the minimum distance with the segmentation approach, the solution becomes less generalized. This approach is evaluated and analyzed in Section 5.5.1.

The clustering approach, while computationally more demanding, is a very solid attempt at forming a generalized solution from little data. The main flaw of the approach originates with the missing information from the depth camera. In order to obtain the central vertical axis of each piece, the median or mean of a piece's cluster data must be taken. With the exception of the pieces right underneath the camera (within the four central squares of the board), each piece's data only consists of the portion facing the camera. Without recreating the piece information, the best methods which can yield good accuracy are the ones which most efficiently mitigate this slight offset or compensate for it.

The process consists of combining an iterative clustering approach with the slicing improvement over the regular clustering. During each iteration, an XY clustering is performed across the full board, before the lower third of the data along the height axis for the matching data is removed. Following that, the minimum Euclidean distance to cluster pieces together is increased at each iteration, as to prevent isolated clusters from forming. This process is a big trade-off between efficiency and reliability. While the slowest out of all algorithms, this approach prevents any data from remaining isolated or any pieces from being fused together. Additionally, this method significantly reduces the offset caused by the camera perspective, as only a fraction of the highest data is kept for each cluster. It is important to note that the amount of data kept at each iteration is large enough to allow fused pieces to contain enough data when split for the next iteration. This algorithm is further discussed and analyzed in Section 5.5.2.

In conclusion, while iterative processes increase the generalization and accuracy of the clustering algorithm, they come at a heavy computational cost. In all implementations, the iterative approaches require the repeated clustering of a data set. While the extent is based on the specific iterative method used, it stands that the time complexity of iterative clustering is significantly slower than regular clustering or segmentation approaches. The choice of use between any algorithm lies in the trade-off between time complexity and reliability, something addressed in the following section.

4.3 Evaluation Procedure

The data used to evaluate the results of both algorithms consists of a database of test cases, generated through the use of a random chess-position generator. The generator functions as follows: Given a set of pieces to use from a chess piece set, and the number of pieces desired on the generated board, the generator randomly places a selection of the pieces on the board while maintaining the two given parameters.

The generator used was developed by Bernd Will, as a gadget used to create FEN strings of chess boards with absurd positions [17]. The board constellations consist of unrealistic chess board states, which do still conform to the standard chess rules. An example of such a constellation is shown in Figure 7. The reason for this use was to allow the algorithms to be tested both in regular and irregular positions, as to ensure that they were not overly fitted to a specific series of commonly used board states.

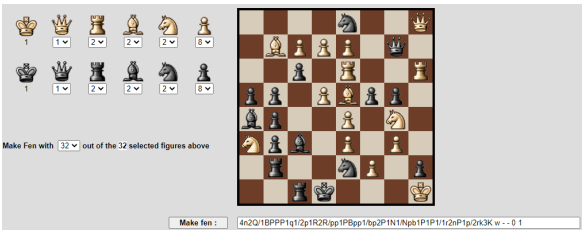

Fig. 7. Image displaying Bernd Will's random absurd chess board state generator. In the evaluation these positions have been set-up on the physical board and subsequently evaluated by the chess-robot.

There are a total of 32 cases used for testing. Each case consists of a set of three boards, each of a different difficulty (Easy, Medium and Hard). Each case uses a different number of pieces. The pieces used are fully randomized, with the addition of two states, the empty control state and the standard chess starting setup. Overall, the chess board tests obtained from this generator are all distinct, and the selection of pieces used varies from state to state. As the data is provided through a standard FEN string, it is first converted into its simplified form to ease the comparison with the result of the algorithm.

The algorithms are validated using two criteria, estimation performance and computational time. The estimation performance consists of a set of derived values from a confusion matrix: **Accuracy, Precision, Sensitivity, Specificity, Negative Predictive Value**.

The computational time is evaluated using both an analysis

of the time complexity of each algorithm along with the execution time on an **Intel Core i7-4790k 4.00GHz CPU** with **14GB of RAM**.

5 Results

This section displays the results obtained from running each algorithm and its variant on 96 board-states. Several algorithms require tuning of independent variables for optimal performance. For each parameter tuning, a Receiver Operator Characteristic (ROC) curve is provided in the Appendix to evidence the "best choice" of each parameterization for the overall comparison. The optimal parameters were found by taking the point closest to the top left corner of the ROC curve, representing a true positive rate of 1 and a false positive rate of 0.
In order to show that the differences between improved algorithms are statistically significant, a Chi Square Test of Independence is used.

5.1 Segmentation Results

The results of the basic segmentation, along with its improvements are detailed in this section.

5.1.1 Basic Segmentation

The basic segmentation approach is a very simplistic approach, based on the assumption that the pieces are perfectly centered on a square. This approach iterates over the coordinates which match each chess board cell, and decides if a piece is present based on the presence of data.

TABLE 1
Table displaying the confusion matrix obtained by running the basic segmentation approach on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1647	FP = 524
Predicted '0'	FN = 6	TN = 3967

TABLE 2
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
91.4%	75.9%	99.6%	88.3%	99.8%

Number of correctly classified boards: 16/96.

The confusion matrix and derived values in Table 1 and Table 2 show that the algorithm performs overall poorly with only 16 out of 96 boards classified correctly. While the algorithm successfully recognizes the presence of a piece with a Sensitivity of 99.6%, it fails to classify empty squares consistently, with a Precision of 75.9%.
The main flaw of the basic segmentation approach is based around the assumption of pieces being perfectly centered. In cases of overlap, with a piece partially present on a neighbouring square, the algorithm detects both squares as occupied and a false positive occurs.
The improvements discussed in the following subsections aim to reduce the basic algorithm's reliance on centered pieces and an analysis of the speed of the algorithm is discussed in Section 5.2.

5.1.2 Shrink Offset Segmentation

The shrink offset is an improvement over the basic segmentation approach. It aims to reduce the number of false positives by taking only a subset of the data present in a chess cell into consideration. It does so by only considering an inner portion of each chess square, based on the value of a shrink offset variable.
The optimization process used to find the optimal shrink offset is represented by an ROC curve in Figure 17 of the Appendix.

TABLE 3
Table displaying the confusion matrix obtained by running the shrink offset segmentation on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1575	FP = 34
Predicted '0'	FN = 78	TN = 4457

TABLE 4
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
98.2%	97.9%	95.3%	99.2%	98.3%

Number of correctly classified boards: 47/96.

When only taking the inner 40% of the data into consideration, an increase in the number of successfully evaluated cases can be observed. A sharp decrease in the number of false positive cases is present, as a considerable overlap is now required for a false positive to occur. This can be seen by the increase in Precision from 75.9% to 97.9% shown in Table 4. However, as only the innermost 40% of the cell is used, it is difficult for overlapping pieces to be considered present in the original cell, shown by the small decrease in Sensitivity of 4.3%. A prime example of such a case is shown in Figure 8.

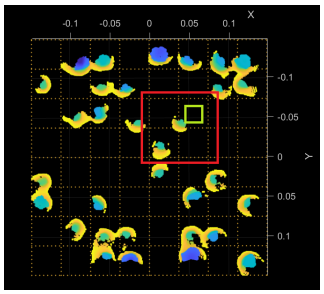

Fig. 8. Point cloud representing the typical case where the shrink offset method fails entirely. The center of the red square contains a piece which is overlapping on four different chess squares. The green rectangle represents the area evaluated with a shrink offset of 0.09m.

In order to statistically display the improvements of the shrink offset approach over the basic segmentation algo-

rithm, a Chi Square Test of Independence was performed, shown in Table 5.

TABLE 5
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic found is 522.1119 with a p-value < 0.00001.

	TP	TN	FP	FN	Row Total
Basic	1647	3967	524	6	6144
Shrink Offset	1575	4457	34	78	6144
Column Total	3222	8424	558	84	12288

The null hypothesis states that the outcome of the distribution is independent of the algorithm used. The alternative hypothesis states that there exists a difference in the distribution of responses to the outcome variable among the compared algorithms. As the p-value is smaller than 0.01, we can state that there is a difference between the distribution of the responses amongst the compared algorithms, and hence that the improvement brought forth by the introduction of a shrink offset is significant.
Overall, with a substantial increase in correctly evaluated boards and a sharp decrease in the number of false positive cases, the algorithm is a net improvement over the basic segmentation approach.

5.1.3 Threshold Segmentation

The insertion of a threshold for a minimum number of data points per piece aims to use all data while reducing the number of false positive cases while keeping the number of false negative cases as low as possible.
The optimization process used to find the best threshold is represented by a ROC curve in Figure 18 of the Appendix.

TABLE 6
Table displaying the confusion matrix obtained by running the threshold segmentation on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1644	FP = 370
Predicted '0'	FN = 9	TN = 4121

TABLE 7
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
93.8%	81.6%	99.5%	91.8%	99.8%

Number of correctly classified boards: 24/96.

With a threshold of 65 data points, while the method provides a lower reduction in the false positive count compared to the shrink offset, it benefits from low false negative count with the Sensitivity remaining at 99.5% (see Table 7). With the milder improvements, only an additional 8 boards were successfully evaluated.
The reason behind the less drastic improvement lies in the inconsistency of the data retrieval process as shown in Figure 9. In this figure, one of the evaluated chess board point clouds is shown, along with a piece representing the inconsistency in the data. Due to a higher number of data points present on the green square, a threshold which allows the segmentation algorithm to fully recognize the board cannot be properly set.

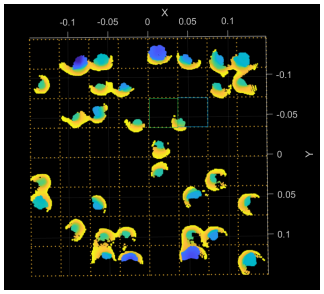

Fig. 9. Image representing the same chess configuration as Figure 12. The green and blue rectangles represent different squares of the chess board. The piece's true location lies in the blue square. In this case, the green square contains 543 data points while the blue square contains 501 data points.

In order to show that the improvements brought by the insertion of a threshold are significant, a Chi Square Test of Independence is performed, shown in Table 8.

TABLE 8
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 30.0629 and the p-value is < 0.00001.

	TP	TN	FP	FN	Row Total
Basic	1647	3967	524	6	6144
Threshold	1644	4121	370	9	6144
Column Total	3291	8038	894	15	12288

As the p-value is smaller than 0.01, the null hypothesis that the confusion matrix values are distributed independently from the algorithm use can be rejected. As such, the improvements brought forth by the usage of a threshold are significant.
Overall the insertion of a threshold into the basic segmentation algorithm is a positive and significant improvement. However, due to limitations caused by the shadow-casting of pieces, the method is not an ideal solution.

5.1.4 Combination of Shrink Offset and Threshold Segmentation Improvements

The shrink offset and threshold variants both offer different degrees of improvement to the same basic algorithm. The following approach consists of an attempt to improve the shrink offset implementation by combining it with the threshold improvement. To do so, the shrink offset value of 0.009m was chosen, and an optimum threshold value for that value was found.
The optimization process used to find the best threshold for a shrink offset of 0.009m is represented by an ROC curve in Figure 19 of the Appendix.
The combination of the shrink offset and threshold improvements show a decrease of 346 false positive cases for an increase of 13 false negative cases (see Table 9), with the Precision increasing from 75.9% to 89.2%, shown in Table

TABLE 9
Table displaying the confusion matrix obtained by running the final segmentation approach (shrink offset + threshold) on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1475	FP = 178
Predicted '0'	FN = 19	TN = 4472

TABLE 10
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
96.8%	89.2%	98.7%	96.2%	99.6%

Number of correctly classified boards: 37/96.

10.
The improvements seem mixed. While the algorithm performs substantially better than the basic segmentation algorithm, despite the same shrink offset value, it performs worse than the regular shrink offset variant, with a total of 37 out of 96 boards evaluated correctly and a lower true positive count.
An exemplary board is shown in Figure 10, consisting of a case where the addition of the threshold reduces the performance of the shrink offset.

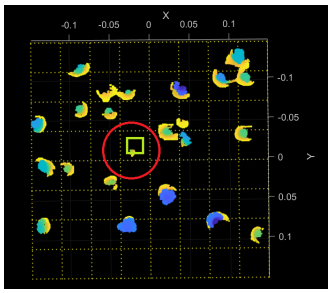

Fig. 10. An example of the negative influence brought by the introduction of the threshold is shown, circled in red. In this example, due to only a portion of the data being considered (green circle), the threshold causes the evaluation to return false. As there is a large discrepancy in the amount of data present within the center of the square, the case occurs frequently, leading to the observed decrease in the number of successfully evaluated boards.

In order to show significance in the variant's benefits, a Chi Square Test of Independence against the basic segmentation approach is, shown in Table 11.

TABLE 11
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 216.9914 and the p-value is < 0.00001.

	TP	TN	FP	FN	Row Total
Basic	1647	3967	524	6	6144
Shrink + Threshold	1475	4472	178	19	6144
Column Total	3122	8439	702	25	12288

Overall, comparing the algorithm to the basic algorithms or the threshold variant, the improvements are far more substantial and significant. While the threshold variant decreased the number of false positives by 154, the insertion of the shrink offset further decreased the false positive count by 192. Compared to the shrink offset method alone, the algorithm is not as robust of an improvement.

5.2 Segmentation Time Complexity Analysis

The idea of the segmentation approach and its variants is to iterate over each cell present on a board, and to find the data present within the boundaries of the cell. The insertion of a shrink offset or of a threshold does not affect the computational complexity:

$$\text{Time Complexity} = O(64n) = O(n),$$

where n is the number of data points. Figure 11 is a board state used to evaluate the performance of each segmentation algorithm.

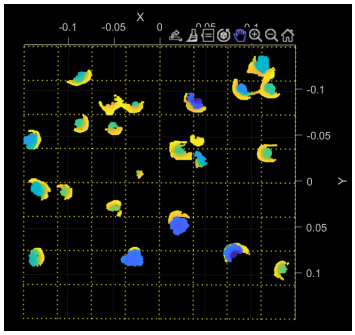

Fig. 11. Point cloud used for the testing of the time required for all three of the discussed approaches. The board was randomly chosen from the database and was kept to allow the number of data points n to remain constant.

Measuring a run from the segmentation-based algorithms, the average elapsed time of 0.023282 seconds is found. While a small discrepancy could be seen, as the methods are identical in terms of computation time, only the average is provided.

5.3 Clustering Results

The results of the basic clustering, along with its improvements are detailed in this section.

5.3.1 Basic Clustering

The clustering algorithms classify the state of the board by considering all data simultaneously. The general idea consists of applying a k-means algorithm using a minimum Euclidean distance on the point cloud data and mapping the center of mass of each cluster onto the chess board.
The optimization process used to find the best minimum Euclidean distance is represented by an ROC curve in Figure 20 of the Appendix.

TABLE 12
Table displaying the confusion matrix obtained by running the basic clustering approach on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1535	FP = 64
Predicted '0'	FN = 118	TN = 4427

TABLE 13
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
97.0%	95.9%	92.8%	98.6%	97.4%

Number of correctly classified boards: 44/96.

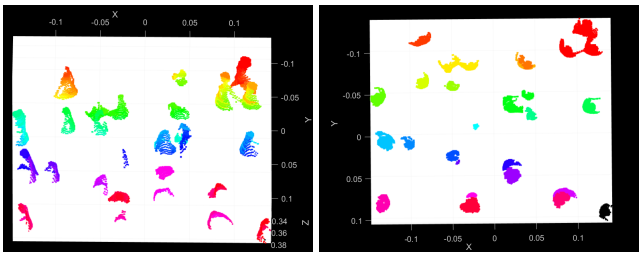

Fig. 12. Representation of the clusters formed when a poor minimum Euclidean distance is chosen. The image on the left shows a case where the minimum distance is set too high $(0.01m)$. In such a case, clusters belonging to multiple individual pieces are fused together, and the center of mass is set in the middle of all 3 pieces. The image on the right shows the case where a too small minimum distance is chosen $(0.001m)$. In such a case, one can see the "rainbow"-like pattern on several pieces caused by the piece being broken down into several clusters.

Figure 12 shows the clustering processes using two incorrect minimum euclidean distances, and illustrates the importance of the optimization process. A high minimum distance results in a large number of false negative cases, while a too low minimum distance results in an approach comparable to the basic segmentation algorithm.
The algorithm performs far better out of the box than the basic segmentation approach. All of the derived confusion matrix percentages are above 90%, with the algorithm performing similarly to the shrink offset improvement (see Table 12 and 13). In order to show that the performance difference with the basic segmentation is not due to chance, a Chi Square Test of Independence is performed, see Table 14.

TABLE 14
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 216.9914 and the p-value is < 0.00001.

	TP	TN	FP	FN	Row Total
Basic Segmentation	1647	3967	524	6	6144
Basic Clustering	1535	4427	64	118	6144
Column Total	3182	8394	588	124	12288

An additional Chi Square test, comparing the base clustering approach with the shrink offset variant was performed, returning a p-value of 0.000448.
Both Chi Square tests returned a p-value below 0.01, allowing the null hypothesis, that the results are independent of the algorithm used, to be rejected. The basic clustering improvement is a significantly better approach than the basic segmentation approach, but does not perform as well as the shrink offset improvement.

5.3.2 *Height Data Reduction Clustering*

The height data reduction improvement originates from the observation that only the chess pieces whose bases are in contact are clustered together. The height reduction improvement removes a portion of the lower piece information during the preprocessing step.
The optimization process used to find the best height cutoff is represented by an ROC curve in Figure 21 of the Appendix.

TABLE 15
Table displaying the confusion matrix obtained by running the height data removal clustering on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1614	FP = 96
Predicted '0'	FN = 39	TN = 4395

TABLE 16
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
97.8%	94.4%	97.6%	97.9%	99.1%

Number of correctly classified boards: 45/96.

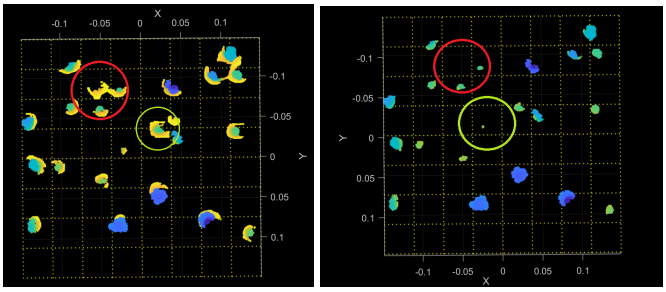

Fig. 13. In many board cases, several of the pieces lack a substantial amount of height information due to being surrounded by too many taller pieces. The red circle represents a pawn surrounded by a bishop and a rook, while the green circle represents an isolated bishop. The right and left image consist of the same point cloud, cropped at two different height values. For many cases, due to the pieces being obfuscated by others, much of the upper height information is not recorded by the camera, causing the piece to disappear entirely. In this the height data reduction value chosen was 0.37.

While the algorithm seems to be a slight improvement over the basic approach (see Table 16), it comes with a flaw. Figure 13 shows an example of the algorithm introducing a false negative, as the preprocessing step removes all data pertaining to the piece circled in red. Since the removed data

cannot be easily recovered, the algorithm is not considered to be an improvement over the basic clustering approach.

5.3.3 Height-less Clustering

The height-less clustering improvement aims to reduce to minimum distance required by omitting the height axis from the clustering process. Unlike the height reduction approach, no data is removed during the preprocessing step. The optimization process used to find the minimum euclidean distance for the height-less clustering is represented by an ROC curve in Figure 22 of the Appendix.

TABLE 17
Table displaying the confusion matrix obtained by running the height-less clustering on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1530	FP = 43
Predicted '0'	FN = 123	TN = 4448

TABLE 18
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
97.3%	97.3%	92.6%	99.0%	97.3%

Number of correctly classified boards: 53/96.

The algorithm results in a similar confusion matrix as the basic clustering approach, shown in Table 17 and 18. While only a small increase in the Precision can be seen, the height-less clustering results in an additional 9 successfully evaluated boards.
Figure 14 shows a flaw with all clustering approaches. Due to the shadow-casting of the chess pieces, the center of mass is offset towards the side of the piece facing the camera. In most cases, the difference is negligible, but the Hard cases in the database consist uniquely of scenarios with pieces located on the edge between two chess cells. As a consequence of the shadow-casting, the piece is recognized on the wrong square, causing both a false positive and false negative to occur.

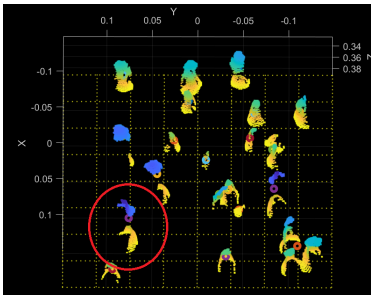

Fig. 14. A common problem encountered due to the camera perspective choice is circled in red in the image above. The purple circle represents the center of mass (CoM) of the object. The XY coordinate of the CoM is (0.076, 0.0736). While the X coordinate is well within the appropriate square, the piece itself is physically located on the cell with Y within the range 0.074-0.111m. As such the piece's center of mass is misclassified and both a false positive and false negative case occurs.

In order to show that the improvements are significant, a Chi Square Test of Independence is performed, see Table 19.

TABLE 19
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 4.7275. and the p-value is < 0.19287.

	TP	TN	FP	FN	Row Total
Basic Clustering	1535	4427	64	118	6144
Height-less Clustering	1530	4448	43	123	6144
Column Total	3065	8875	106	241	12288

As the p-value is lower than 0.01 or even 0.05, the null hypothesis, that the confusion matrix values and the algorithm choices are independent from one another, is accepted. The differences between the basic clustering algorithm and the height-less clustering variant cannot be considered significant.

5.4 Clustering Time Complexity Analysis

The basic clustering approach uses a k-means algorithm to cluster the data into respective pieces, before mapping the center of masses found onto the chess board. The time complexity is follows:

$$\text{Time Complexity} = O(n^2 + mn + 64n) = O(n^2),$$

where n is the size of the data and m the number of clusters. It is assumed that the k-means clustering used has an O(n^2) complexity. The height reduction and height-less clustering methods have the same time complexity as the basic approach, though the height reduction method has less data points due to the data being removed in the preprocessing step.
Timing the execution of each algorithm, the basic and height-less clustering algorithms both returned similar results, as the same data and clustering method are used. Their execution time averaged between 0.278012 and 0.286960 seconds. The height reduction method, with the presence of less data during the clustering process, was twice as fast, with an average execution time of 0.121484 seconds. These values were obtained from the same board in Figure 11, in order to ensure consistency throughout the time analysis.

5.5 Iterative Clustering

In this section, an analysis of the active and passive iterative clustering algorithms is provided.

5.5.1 Active Iterative Clustering

After an initial clustering, this approach iterates over each cluster and breaks down any fused clusters into their individual pieces.
The optimization process used to find the optimal evaluation metric is represented by an ROC curve in Figure 23 of the Appendix.
The algorithm performs better than the basic clustering. The confusion matrix values shows a lower false positive and false negative count, and the number of correctly evaluated boards increases from 44 to 54, see Table 21.

TABLE 20
Table displaying the confusion matrix obtained by running the active iterative clustering with optimal parameters on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1561	FP = 38
Predicted '0'	FN = 92	TN = 4453

TABLE 21
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
97.9%	97.6%	94.4%	99.2%	97.8%

Number of correctly classified boards: 54/96.

Figure 15 shows an example flaw present in the iterative process. While the re-clustering done within the fusion breaks the larger cluster down into 3 sub-clusters, the process breaks down some of the pieces incorrectly, causing the center of mass to be further offset towards a neighbouring cell.

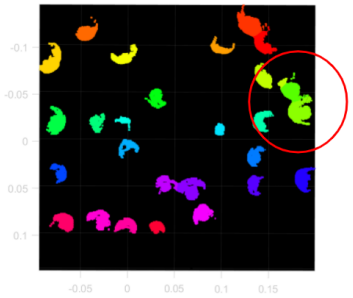

Fig. 15. An example of a flaw in the active iterative approach. The three pieces of interest are circled in red, where the lightest green cluster is broken down incorrectly. As a consequence, the center of mass of the piece is further offset towards a neighbouring cell.

In order to show that the increase and improvements are significant, a Chi Square Test of Independence is performed, see Table 22.

TABLE 22
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 10.141. and the p-value is < 0.05.

	TP	TN	FP	FN	Row Total
Basic Clustering	1535	4427	64	118	6144
Active Iterative	1561	4453	38	92	6144
Column Total	3096	8880	102	210	12288

The Chi Square test returned an independence between the values obtained and the algorithm used with a p-value of 0.017405. While the improvement is significant with a p-value < 0.05, more testing is required to properly determine the significance of the data.

Time Complexity:

$$\text{Time Complexity} = O((n^2 + imn) + 64n) = O(n^3),$$

where i is the number of iterations, n the number of data points and m the number of clusters found at each iteration. Running this iterative approach on the same board referred to in Figure 11, the average elapsed time of 0.447120 seconds is found. The algorithm is slower than the non-iterative clustering approaches, and is not noticeably better than the height-less clustering improvement.

5.5.2 *Passive Iterative Clustering*

The passive iterative approach repeatedly clusters the entire chess board, whilst removing a portion of each cluster's data after every iteration.

The optimization process used to find the optimal set of parameters is represented by an ROC curve in Figure 24 of the Appendix.

TABLE 23
Table displaying the confusion matrix obtained by running the passive iterative clustering with optimal parameters on 96 board-states. The table holds the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) counts, which represent the classification results in comparison to the ground-truth.

	Actual '1'	Actual '0'
Predicted '1'	TP = 1622	FP = 24
Predicted '0'	FN = 31	TN = 4467

TABLE 24
Table consisting of the percentages derived from the confusion matrix. The Accuracy represents the overall performance of the algorithm, while the remaining percentages represent the proportion of truth values correctly identified.

Accuracy	Precision	Sensitivity	Specificity	Negative Pred.
99.1%	98.5%	98.1%	99.5%	99.3%

Number of correctly classified boards: 72/96.

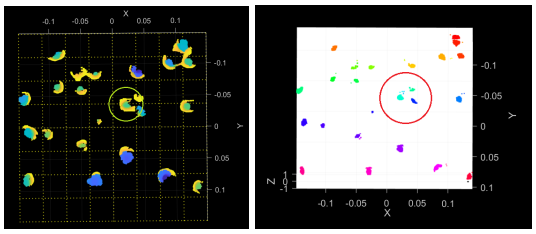

Fig. 16. A typical problem of a too low minimum clustering distance increase is shown in the red circle. With a too low value (in this case, without an increase), the piece circled in green is broken down into two separate clusters which are present on two different chess cells. This causes the algorithm to detect a False Positive.

Figure 16 above shows a case where the minimum clustering distance increase is too low, and causes a false positive to be detected. With the optimized increase of +0.00175m however, such cases do not occur in our data set.

Comparing the confusion matrix values to that of the basic clustering algorithm, one can immediately notice a substantial decrease in both the false positive and false negative cases (see Table 23). Additionally, the algorithm correctly evaluates 72 out of the 96 boards, which is an improvement over the previous 44.

The number of iterations was fixed to 5. While the algorithm performs similarly or slightly better (74 cases out of 96), with

the combination of a higher percentage of preserved data, minimum distance increase and higher iteration count, the process rapidly becomes too time consuming. Due to the O(n^2) nature of the clustering process, when placed inside of an iterative loop, the time complexity of the algorithm becomes O(n^3). If the iteration count is low, the effect is almost imperceptible, however at higher iterations, the algorithm can take up to a minute to complete.
In order to show that the increase and improvements are substantial, a Chi Square Test of Independence is performed, see Table 25.

TABLE 25
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 71.5579. and the p-value is < 0.00001.

	TP	TN	FP	FN	Row Total
Basic Clustering	1535	4427	64	118	6144
Passive Iterative	1622	4467	24	31	6144
Column Total	3157	8894	68	149	12288

The algorithm is not flawless. While the cases can be argued to be too difficult to evaluate for a human, the logic to recognize the pieces nonetheless remains the same for a computer. To improve this number using this iterative process, a solution is to decrease the minimum distance increment and percentage of data preserved, whilst also increasing the number of iterations. This solution brings back into the question the trade-off between time complexity and reliability.
Time Complexity:

$$\text{Time Complexity} = O(i(n^2 + mn) + 64n) = O(n^3),$$

where i is the number of iterations, n the number of data points and m the number of clusters found at each iteration. Running this iterative approach on the same board referred to in Figure 14, the average elapsed time of 0.755646 seconds is found. When comparing this value to the other clustering variants, a large difference can be seen. While the algorithm is around 3 times slower than the basic clustering approach, it is still low enough to be negligible when placed inside of the feedback loop described in Figure 1.

5.6 Summary Segmentation vs Clustering

Overall, both approaches perform well. While the basic segmentation is unreliable in a realistic scenario, the shrink offset and threshold improvements do enhance its performance. When comparing the performance of the best segmentation approach to the best clustering approach, a clear difference can be seen. The use of a clustering algorithm, while less time-efficient, is more suitable for this kind of problem, as all data on the board needs to be taken into account to decipher what constitutes a piece and under what piece the 3D data belongs.
To show that the clustering performance is significantly better than the segmentation performance, a Chi Square Test of Independence was performed, see Table 27.

As the p-value is smaller than 0.01, the null hypothesis is rejected, and the data of the confusion matrices is dependent on the algorithm used. In other words, there exists a significant difference between using both algorithms.
In the case of chess board recognition, while the time complexity plays a part, as the final result is the processing of a single board consisting of at most 32 pieces, the time it takes for the longest clustering variant to complete is negligible. While increasing the iteration count for that variant can increase the computation time up to a minute, the current state of the variant takes under 1 second to complete.

TABLE 26
Summary table consisting of the number of correctly evaluated boards and the time taken to evaluate the same board for all algorithms discussed in this paper.

Method	Number of Correctly Evaluated Boards	Time Taken to Evaluate a Single Board
Basic Segmentation	16 out of 96	0.023282 seconds
Segmentation with Shrink Offset	47 out of 96	0.023282 seconds
Segmentation with Threshold	24 out of 96	0.023282 seconds
Segmentation with Shrink Offset and Threshold	37 out of 96	0.023282 seconds
Basic Clustering	44 out of 96	0.278012 seconds
Clustering with Data Reduction	45 out of 96	0.121484 seconds
Clustering without Height Data	53 out of 96	0.286960 seconds
Iterative Clustering	54 out of 96	0.447120 seconds
Iterative Clustering with Data Reduction and without Height	72 out of 96	0.755646 seconds

TABLE 27
Results of the Chi Square Test of Independence obtained from the Social Science Statistics website [18]. In this test, the chi-square statistic is 22.6924. and the p-value is < 0.000047.

	TP	TN	FP	FN	Row Total
Best Segmentation	1575	4457	34	78	6144
Best Clustering	1622	4467	24	31	6144
Column Total	3197	8924	58	109	12288

6 Conclusion and Future Work

An autonomous chess-playing robot relies on a robust and accurate detection of the chess-board state to play without the need of any human support/intervention. While this problem has extensively been investigated in the past using RGB information [8], [11], 3D stereo vision introduces new possibilities. In this paper, three approaches were implemented, optimized and evaluated: a segmentation, a clustering and an iterative clustering approach. The best results were achieved using an ROC-based optimization process, which relied on the classification results of each approach compared to the ground-truth.
Unlike conventional RGB approaches, the use of 3D depth information allows a more generalizable solution for the recognition problem to be derived, at the cost of an increased computation time.

The appearance of affordable depth cameras is bound to bring forth a wave of new interest in the technology and its applications. With the appearance of PointNet [19], a neural network specialized in processing 3D data, the use of depth cameras in chess board recognition can rapidly

become a viable alternative to the classic RGB cameras.
The evaluated recognition methods are limited towards the full range of chess moves, with notable examples being castling or promotion. An improvement to all the approaches would be the addition of a chess piece recognition algorithm, which would simplify the recognition logic currently in place. An alternate improvement would be a method of compensating for the lack of data, such as an approach combining depth information obtained from multiple perspectives.

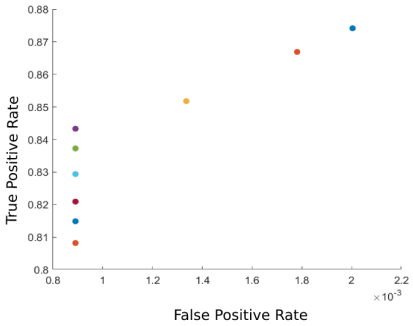

Fig. 19. ROC curve depicting the optimization process conducted to find the best best threshold for the given shrink offset. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to the threshold value of 40 data points.

APPENDIX A
OPTIMIZATION ROC CURVES

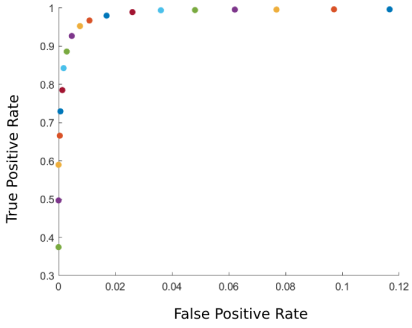

Fig. 17. ROC curve depicting the optimization process conducted to find the best shrink offset for the approach. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to the shrink offset value of 0.009m.

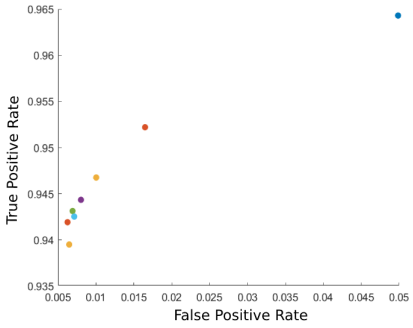

Fig. 20. ROC curve depicting the optimization process conducted to find the best minimum Euclidean distance for the basic clustering approach. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to the minimum distance of 0.006m.

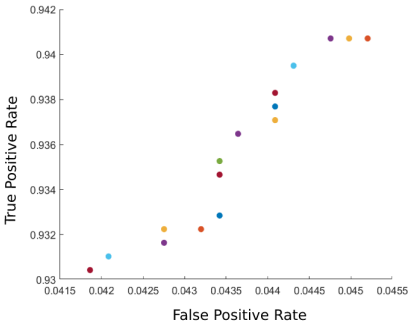

Fig. 18. ROC curve depicting the optimization process conducted to find the best threshold for the approach. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to the threshold value of 65 data points.

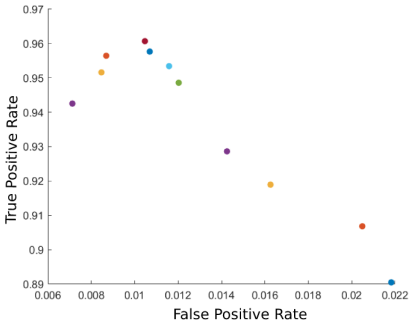

Fig. 21. ROC curve depicting the optimization process conducted to find the best height cutoff for the clustering approach variant. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to the value of 0.3875m.

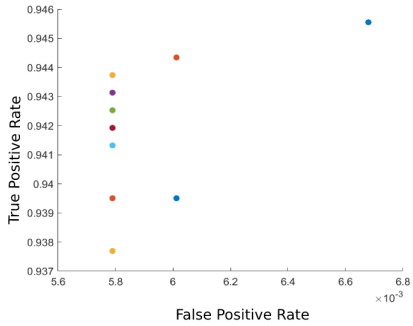

Fig. 22. ROC curve depicting the optimization process conducted to find the best minimum Euclidean distance for the height-less clustering. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to a minimum euclidean distance of 0.003m.

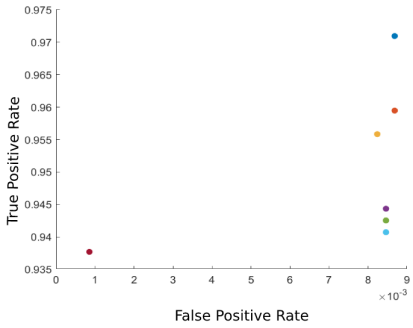

Fig. 23. ROC curve depicting the optimization process conducted to find the distance with which to consider a cluster to be a fusion. The optimal value is found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to a value of 1.3×the size of a chess cell.

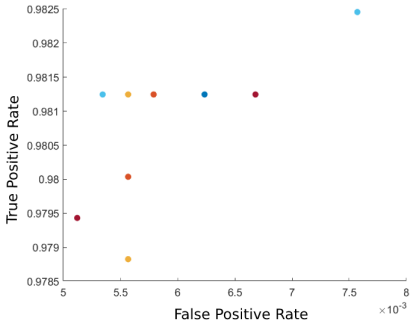

Fig. 24. ROC curve depicting the optimization process conducted to find the best parameter combination for the passive iterative clustering approach. The optimal values are found by taking the point closest to the top left corner, representing a True Positive Rate of 1 and a False Positive Rate of 0. This corresponds to an iteration count of 5, a minimum clustering distance of 0.003m, a data preserving ration of 66.6% and an increment of 0.00175m.

REFERENCES

[1] Higgins, C. (2017, July 29). A Brief History of Deep Blue, IBM's Chess Computer. Retrieved January 30, 2021, from http://mentalfloss.com/article/503178/brief-history-deep-blue-ibms-chess-computer

[2] D. Urting and Y. Berbers, "MarineBlue: A Low-Cost Chess Robot," IASTED International Conference Robotics and Applications, Salzburg, Austria, Jun. 2003.

[3] P. Corke, "Robotics Toolbox," Peter Corke, 27-Apr-2020. [Online]. Available: https://petercorke.com/toolboxes/robotics-toolbox/.

[4] "Stockfish Chess," Stockfish. [Online]. Available: https://stockfishchess.org/.

[5] "PhantomX Reactor Robot Arm Kit," PhantomX AX-12 Reactor Robot Arm. [Online]. Available: https://www.trossenrobotics.com/p/phantomx-ax-12-reactor-robot-arm.aspx.

[6] H. A. Almurib, H. F. Al-Qrimli, and N. Kumar, "A review of application industrial robotic design," 2011 Ninth International Conference on ICT and Knowledge Engineering, 2012.

[7] Name*, "Cool Gadgets, Gifts amp; Stuff," CoolThingscom Cool Gadgets Gifts Stuff, 11-Nov-2016. [Online]. Available: https://www.coolthings.com/square-off-robot-chess-board/.

[8] D. A. Christie, T. M. Kusuma, and P. Musa, "Chess piece movement detection and tracking, a vision system framework for autonomous chess playing robot," 2017 Second International Conference on Informatics and Computing (ICIC), 2017.

[9] M. A. Czyzewski, A. Laskowski, and S. Wasik, "Chessboard and Chess Piece Recognition With the Support of Neural Networks," Foundations of Computing and Decision Sciences, vol. 45, no. 4, pp. 257–280, 2020.

[10] L. Hardesty, "Explained: Neural networks," MIT News — Massachusetts Institute of Technology, 14-Apr-2017. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[11] C. Koray and E. Sümer, "A Computer Vision System for Chess Game Tracking," 21st Computer Vision Winter Workshop Luka Cehovin, Rok Mandeljc, Vitomir Struc (eds.)Rimske Toplice, Slovenia, Feb. 2016.

[12] Y. Xie, G. Tang and W. Hoff, "Chess Piece Recognition Using Oriented Chamfer Matching with a Comparison to CNN," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Tahoe, NV, 2018, pp. 2001-2009, doi: 10.1109/WACV.2018.00221.

[13] Afordable eboard with leds and piece recognition - chess forums. (n.d.). Retrieved February 07, 2021, from https://www.chess.com/forum/view/chess-equipment/afordable-eboard-with-leds-and-piece-recognition

[14] "Kinect for Windows," Kinect - Windows app development. Available: https://developer.microsoft.com/en-us/windows/kinect/.

[15] "Intel® RealSense™ Technology," Intel. [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html.

[16] S. J. Edwards, "PGN Standard : Steven J. Edwards : Free Download, Borrow, and Streaming," Internet Archive, 12-Mar-1994. [Online]. Available: https://archive.org/details/pgn-standard-1994-03-12.

[17] B. Will, "Random-FEN-Generator," Bernd's Random-FEN-Generator. [Online]. Available: http://bernd.bplaced.net/fengenerator/fengenerator.html.

[18] "Chi-Square Test Calculator," Chi Square Calculator - Up To 5x5, With Steps. [Online]. Available: https://www.socscistatistics.com/tests/chisquare2/default2.aspx.

[19] R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 77-85, doi: 10.1109/CVPR.2017.16.