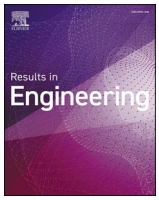
Results in Engineering 25 (2025) 104091

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering



Development of an autonomous chess robot system using computer vision and deep learning

Truong Duc Phuc *, Bui Cao Son

School of Mechanical Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam

ARTICLE INFO

Keywords:
Autonomous chess robot
SCARA robot
Computer vision
Deep learning
Image processing

ABSTRACT

In this research, a low-cost autonomous chess robot system is developed using computer vision, deep learning, and robot control. The system comprises a chessboard, a camera system, and a 4-DOF SCARA robot. The entire system is managed by software running on a computer. Additionally, a deep learning model has been created for chess piece recognition and position detection. The calculation of chess moves is performed using the minimax algorithm within the Stockfish chess engine. Results indicate that the computation time for a chess move is approximately 2 s per chess position, while the average time for the robot to execute a chess piece movement is from 20 to 90 s for one position, depending on the type of chess move. The developed chess robot system operates stably and accurately, capable of autonomously playing a complete chess game against humans or identifying chess positions for a pre-arranged setup. Moreover, the fabrication cost of the robotic arm and its control system is approximately $100, making it both affordable and suitable for training and entertainment-focused chess robot systems. The results demonstrated that the autonomous chess robot system developed in this study is feasible for real-world applications for chess playing or chess training systems.

1. Introduction

Chess serves as a powerful medium to enhance human cognition and psychology, improving mental agility and emotional well-being. This is because the game's complexity and strategic depth make it an excellent dynamic exercise for the brain. Claude Shannon [1] was the first to estimate the game tree complexity of chess at up to 10^120 possible positions, a figure exceeding the number of observed atoms in the universe. This underscores the impracticality of exhaustive search and the necessity of selective strategies. Therefore, chess games require players to analyze complex positions, anticipate opponents' moves, and develop long-term strategic plans. These activities enhance logical reasoning, problem-solving abilities, and decision-making skills, and improve cognitive abilities like memory, concentration, and logical reasoning which are essential for navigating academic, professional, and real-life challenges.

Frequently playing and practicing chess is vital for intellectual growth because it continuously challenges and enhances cognitive abilities. It was reported that expertise in chess is acquired through extensive practice, which transitions cognitive processes from deliberate reasoning to intuitive, unconscious recognition [2–4]. In addition, chess masters excel due to their ability to recognize and encode patterns, or chunks of meaningful chess configurations. Chess masters possess an extensive repertoire of familiar patterns and positions stored in long-term memory, estimated between 10,000 to 50,000, enabling them to identify and retrieve meaningful structures during play [2–4]. These patterns are crucial for quick, accurate decision-making. In addition, this repertoire is built through years of practice and exposure to chess positions [2–4]. These results lead to high demand for the research and development of chess-playing systems for intellectual entertainment, training, and practice.

For the purpose of intellectual entertainment, training and practicing, chess-playing systems can be categorized into two types. The first one is human-computer chess systems, and the other one is autonomous chess robot systems. The human-computer chess systems are software-based platforms where users interact with AI-driven chess engines on computers, tablets, or mobile devices [5,6]. These systems serve as convenient tools for training and education, allowing players of all levels to learn and practice anytime and anywhere against opponents (either a virtual opponent or a real opponent) of varying skill levels with different playing styles. In addition, it is possible to study chess-playing personalities by using analysis tools and virtual simulation inside these

* Corresponding author.
E-mail address: phuc.truongduc@hust.edu.vn (T.D. Phuc).

https://doi.org/10.1016/j.rineng.2025.104091
Received 11 August 2024; Received in revised form 12 January 2025; Accepted 19 January 2025
Available online 22 January 2025

platforms. It was reported that personality-driven strategies in chess significantly affect game dynamics, decision-making players, and performance, even among those with similar skill levels [7,8]. The use of virtual simulations inside chess playing systems provides a controlled environment for studying these factors, offering flexibility in pairing different styles and analyzing outcomes. Therefore, players can benefit from practicing against virtual opponents with varied personalities to improve not only their chess skills but also enhance their adaptability, intellectual and cognitive abilities. Moreover, together with the rapid development of artificial intelligence tools, big data technologies, cloud computing technologies, and optimization algorithms, the chess database is more and more powerful. Thereby, chess training tools can significantly enhance their functionality and performance by integrating various playing styles across different skill levels, as well as diverse conditions and scenarios to their database, offering more diverse and realistic practice scenarios.

On the other hand, autonomous chess robot systems consist of physical robots capable of playing chess autonomously by moving pieces on a real board. While human-computer chess systems are widely used due to their accessibility and convenience, the physical presence of an autonomous robot introduces unique benefits that digital-only systems cannot replicate. One key advantage of autonomous chess robot systems is the real-world, tangible interaction they provide. Unlike human-computer chess systems, where players interact through a screen, chess robots allow players to move physical pieces on a board, which creates a more realistic and competitive atmosphere. This can improve a player's spatial awareness and memory retention. Additionally, practicing with a physical robot better simulates the environment of over-the-board tournaments, where players must manage time pressure and physical interactions, helping to build psychological resilience. Normally, an autonomous chess robot system consists of a sophisticated combination of hardware (i.e., robotic arm, sensors, chessboard) and software (i.e., computer vision, chess engine, motion control). These components work together to enable real-time interaction, physical piece manipulation, and intelligent decision-making, making it difficult and challenging for research and development of autonomous chess robot systems.

Recently, several studies have been published on the development of chess-playing robots. Matuszek et al. [9] developed a 6-DOF chess-playing robot system that identifies chess positions from images using digital image processing methods combined with support vector machines (SVM). The results demonstrated that the robot could play chess autonomously and achieved a success rate of over 91 % in manipulation tasks across multiple games. Additionally, visual servoing significantly improved the accuracy of piece handling, particularly in challenging grasping scenarios. However, the robot arm used in the system is relatively expensive (around $18,000), limiting its accessibility for broader applications. Furthermore, the chess move response algorithm was not mentioned. Similarly, Varun Gupta et al. [10] presented a low-cost autonomous chess-playing robot capable of playing chess against human opponents. The system utilizes a custom-built robot with a rack-and-pinion-based linear slider system to move pieces across a standard chessboard. Moreover, a network of light-dependent resistors (LDRs) was embedded beneath the chessboard to detect the presence of pieces on each square. The results demonstrated that the robot was successfully tested in several games, showing reliable performance in recognizing board states and executing moves accurately. However, the sensors integrated into the chessboard complicate the hardware development, making it less cost-effective. Additionally, if a chess piece is moved to the wrong position (either accidentally or intentionally), all subsequent moves will be incorrect. This is because the LDRs located under each chess square only detect the presence of a chess piece, regardless of which piece it is. Furthermore, the robot is built with a wooden frame, and its design allows for translational movement along the x, y, and z axes. As a result, the robot's mechanical movement is relatively slow, requiring considerable time to complete each move, which may diminish user engagement during extended games. Luqman et al. [11] introduced a chess-playing robotic system capable of autonomously engaging in chess games against human opponents. The system features a 4-DOF robotic arm designed for manipulating chess pieces on a standard board, along with a camera to detect the chessboard and recognize piece positions from a top-down view. The results showed that the system's accuracy in piece handling and decision-making, successfully completing multiple games under various conditions. However, the system requires a predefined starting configuration, which limits its flexibility in managing irregular setups or mid-game positions. Additionally, its error recovery capabilities are constrained, struggling with scenarios such as displaced or knocked-over pieces. The high development cost of the robotic arm and vision components presents a barrier to scalability and broader adoption. Chen et al. [12] introduced an autonomous chess-playing humanoid robot equipped with advanced computer vision and human-interaction capabilities. The system employs computer vision algorithms, including a dynamic approach to the Hough line transform and a hybrid edge and morphology-based method for object detection. This innovative approach allows the robot to rely solely on visual input for game interaction, achieving a 100 % success rate in standard environments and 80 % in extreme conditions. Furthermore, the experimental results demonstrated the system's effectiveness in chess piece manipulation and reliable board recognition. However, the robotic arm operates slowly, with each move taking 45–90 s, which can disrupt the pacing of gameplay. Moreover, utilizing an expensive robot arm (specifically, the study employs a Baxter robot arm costing about $22,000) for a chess-playing system is unaffordable for entertainment purposes, limiting its practical applications. Furthermore, Kazuki Shin et al. [13] explored the feasibility of adapting a general-purpose robotic arm for autonomous chess-playing. The study focuses on integrating the arm's manipulation capabilities with computer vision and a chess engine to create a functional system. The results indicate that the robotic arm successfully demonstrates precise manipulation of chess pieces on a standard board, showcasing its ability to execute legal moves without disturbing other pieces. The computer vision system accurately detects the chessboard, identifies piece positions, and updates the game state in real-time, ensuring seamless interaction between human and robotic players. The research highlights the potential of general-purpose robotic arms for versatile applications, with chess serving as a testbed for robotic manipulation and decision-making in constrained environments. However, the robotic arm may operate at a slower pace than desired, leading to delays during gameplay. Additionally, the general-purpose nature of the robotic arm results in higher costs, which limits the practical applications of the chess-playing system.

On the other hand, extensive research on chess-playing systems highlights the importance of computer vision, particularly in chessboard detection and piece recognition. Jay Hack et al. [14] introduced a cost-effective, real-time computer vision system designed to autonomously track and record chess games using a standard laptop webcam. The system utilized Harris corner detection and heatmaps, combined with Hough transforms and SIFT descriptors, for precise corner detection in identifying the chessboard. Additionally, K-means clustering was employed for color-based piece recognition, enabling the system to identify moves by comparing board states before and after each turn. The results indicate that the system performs well under controlled lighting conditions with minimal occlusion. This approach offers a practical solution for recording chess games without the need for specialized equipment like overhead cameras or digital boards, making it suitable for real-world applications. However, it necessitates manual input from players to trigger move recording after each turn, which limits its level of automation. Furthermore, this system is sensitive to poor lighting and glare, impacting the accuracy of board and piece detection. Object detection in low-light scenarios can be enhanced through a combination of pixel-wise depth refinement and TensorRT optimization, as reported by Vinoth [15]. Moreover, detecting small,

occluded, or indistinct objects in challenging scenarios involving complex backgrounds, overlapping pieces, or varying lighting conditions can be improved by utilizing the enhanced Faster R-CNN algorithm [16]. This is because of the optimized anchor box generation strategy designed for specific object scales, which can be recalibrated for chess pieces. This ensures that the model accurately detects objects of varying sizes, such as pawns and queens, on the chessboard. Furthermore, the adaptive Feature Pyramid Network (FPN) can improve multi-scale feature extraction for chessboards, allowing the model to identify pieces even when obstructed, such as in the case of partially covered squares. Neufeld et al. [17] proposed a novel method that combines line detection, Hough transform, and probabilistic reasoning to identify chessboards from a single camera perspective akin to a human viewpoint. The results indicate that the system can accurately detect chessboards under various angles and lighting conditions. The method achieves a detection accuracy of 91.6 % with standard boards and processes approximately 26 % faster when utilizing optimized masking techniques. The algorithm effectively manages variations in board orientation and lighting by applying erosion, dilation, and Canny edge detection, followed by quadrilateral probability matching. This proposed method is efficient in identifying standard boards and lays a robust foundation for future enhancements involving chess piece recognition, which can be utilized in chess robot systems. Tam et al. [18] presented a method for accurately segmenting the grid of a populated chessboard captured from a low-angle perspective, addressing challenges posed by perspective distortion and partial occlusion of squares. This approach employs edge detection and Hough transforms to identify and extract grid lines, followed by perspective correction to create a top-down view of the chessboard. By utilizing a grid-fitting algorithm, the method achieves high accuracy in delineating individual squares, even when pieces partially obscure the board. The findings are beneficial for chessboard detection, which is applicable to automated chess analysis and chess robot systems. Bennett et al. [19] proposed a method for effectively identifying chessboard vertices by combining edge detection, geometric consistency, and a strength-based response measure. This approach emphasizes accurately detecting the intersections of the chessboard grid, even in challenging conditions such as varying lighting, noise, and distortion. The results demonstrate that the system can reliably identify chessboard corners and intersections in real-world images with diverse perspectives and moderate occlusions, making it suitable for real-time applications in chess robot systems. While significant advancements have been made in image processing and computer vision research for chessboard and chess piece recognition, integrating these technologies into fully autonomous chess-playing systems continues to present considerable challenges. Specifically, the development of low-cost, user-friendly systems with robust and reliable performance remains a complex and highly sought-after task.

In this study, the authors developed a low-cost, user-friendly, autonomous chess robot system that encompasses both hardware and software components. A 4-DOF SCARA robot has been designed to automatically pick and place chess pieces during gameplay. The cost for fabrication of the robot arm and control system is approximately $100, making it affordable and suitable for chess robot systems. Additionally, the image processing system incorporates deep learning algorithms to accurately identify chess positions. The optimal chess move is determined by selecting the highest score among all possible moves using the minimax algorithm. With this approach, the developed chess robot system can play complete chess games against humans and solve pre-arranged chess positions in a short amount of time. Moreover, this work employs open-source and free software, making the development of an autonomous chess-playing system more convenient and cost-effective, which holds potential for real-world applications.

2. Robot design and kinematic to pick and place chess pieces

Fig. 1(a) illustrates the design of a chess robot system examined in

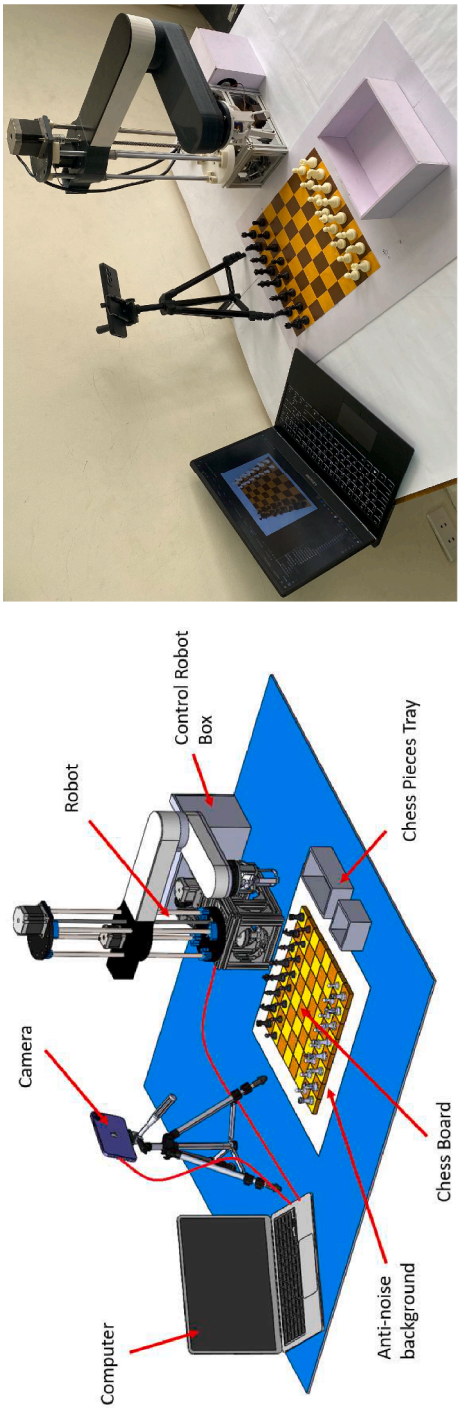


Fig. 1. Chess Robot System: (a) 3D design; (b) Manufactured system.

this study. The system comprises four main components: (1) a chessboard system; (2) a 4-DOF SCARA robot utilized for picking and placing chess pieces in desired positions on the chessboard; (3) a camera system for recognizing chess positions; and (4) control software running on a computer to manage the entire system. Fig. 1(b) presents the practical chess robot system developed in this research.

Fig. 2(a) illustrates the geometric dimension parameters of the 4-DOF SCARA robot designed for picking and placing chess pieces in this study. The robot features one translational movement along the z-axis (T1) with a stroke of 362 mm. This stroke distance is calculated to ensure the robot can pick and place all chess pieces during gameplay without any collisions. It includes two rotational movements (R1 and R2, both with rotation angles ranging from 0 to 340°) and one grip/release movement (G1, with the grasping aperture diameter varying from 18 mm to 32 mm).

Figs. 2(b) and 2(c) illustrate the relative position between the SCARA robot and the chessboard, as well as the robot's working region over the chessboard (i.e., in the *Oxy* plane). In this study, the dimensions of the chessboard are set at 320 × 320 mm, which is sufficient for accommodating each chess piece on its respective square unit. The rotation center of link 1 of the robot is positioned on the vertical symmetry axis of the chessboard (i.e., on the *y*-axis) and is located 109 mm from the lower edge of the chessboard. The robot's motion trajectory is defined by the area between the blue dashed line circles A and B, as shown in Fig. 2(c). It is calculated that the robot's farthest working range is represented by circle A, with a diameter of 916 mm, while its nearest working range corresponds to circle B, with a diameter of 196 mm.

Fig. 3 depicts the kinematic parameters of the SCARA robot in relation to the chessboard. L1, L2, φ_1, and φ_2 represent the lengths and rotation angles of link 1 and link 2, respectively. X and Y denote the coordinates of the center of the gripper link in the *Oxy* plane. The forward kinematic equations of the robot on the *Oxy* plane are computed as follows:

$$\begin{cases} X = -L1cos(\varphi 1) - L2cos(\varphi 1)sin(\varphi 2) \\ Y = -L1sin(\varphi 1) + L2sin(\varphi 1)cos(\varphi 2) \end{cases} \quad (1)$$

The inverse kinematic equations are derived by solving Eq. (1), resulting in the rotation angles for link 1 and link 2 as follows:

$$\begin{cases} \varphi_1 = f(X, Y, L_1, L_2) \\ \varphi_2 = f(X, Y, L_1, L_2) \end{cases} \quad (2)$$

These calculated rotation angles (φ_1, φ_2) serve as the output for directing the robot to pick and place a chess piece at the corresponding (X, Y) position.

The inverse kinematic equations of the robot (Eq. (2)) can be solved using several methods, including the geometric solution approach, which is applicable to simple robot structures [20], the algebraic method [21], and the numerical method [22]. These traditional solutions are often time-consuming, which limits the system's fast response. In this research, the Sympy library, a fully featured computer algebra system (CAS) written in the Python programming language [23], is employed to solve the inverse kinematic equations of the robot. This choice is made because it provides solutions quickly and accurately [24]. Additionally, it is free and open-source software.

Fig. 4(a) illustrates the robot's transmission mechanism. NEMA 17 stepper motors are employed to drive rotations R1 and R2 through two GT2 belt-pulley transmission systems. The speed reduction ratios for the belt-pulley systems are 228/11 for rotation R1 and 16 for rotation R2, respectively. A NEMA 23 stepper motor is utilized to drive the translation movement T1 along the *z*-axis via a lead screw and nut system. Translation movement T1 is guided by four guide shafts and bushings. A servo motor (SG90) operates the gripper mechanism for picking and placing a chess piece.

Fig. 4(b) illustrates the control circuit diagram of the robot. A CNC Shield V3 is employed to manage the gripper servo motor and stepper motors through the TB6600 drivers. An Arduino Uno R3 is used to oversee the entire system.

3. Chess position recognition using computer vision and deep learning algorithm

Fig. 5 illustrates the flowchart of image processing for recognizing chess positions and calculating chess moves. The chess position recognition was performed using image processing, which comprised two subprocesses: (1) chessboard detection; (2) chess piece recognition. Once the chess position is established, the chess move can be calculated to determine the optimal move for the current position.

3.1. Image processing for chessboard detection

The chessboard detection process aims to accurately identify the coordinates of the center points of the chessboard squares. These center point coordinates are utilized to control the robot's movement for picking and placing chess pieces during a game. Consequently, a moving chess piece is picked from the center point of one chessboard square and placed on the center point of another chessboard square.

Fig. 6 illustrates the process of chessboard detection through image processing. A camera system captures an image of the chessboard, which is then processed through the following steps: (a) Edge detection using the Canny algorithm; (b) Line detection employing the Hough Line Transform algorithm; (c) Detection of intersection points via a manual clustering algorithm; and (d) A flattened and orthogonal view of the chessboard using the Homography algorithm. The center points of the chessboard squares are then determined from this flattened and

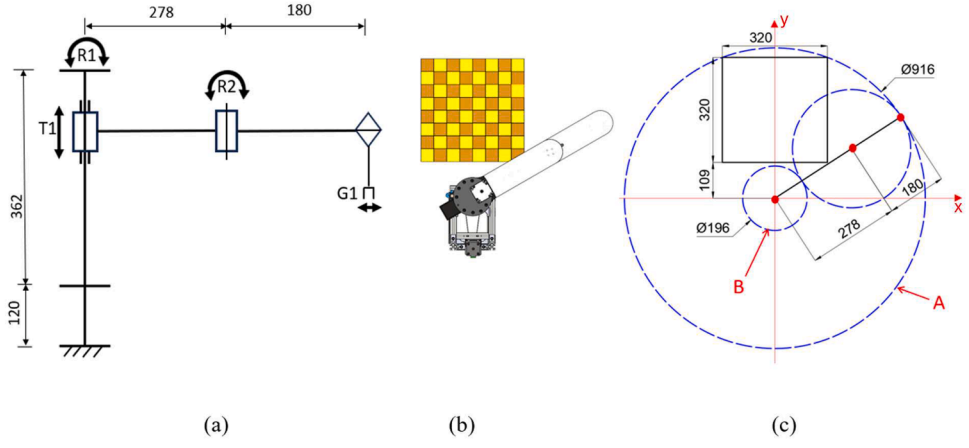


Fig. 2. (a) The robot's parameters; (b) Position of the robot relative to the chessboard; (c) Working region on the *Oxy* plane.

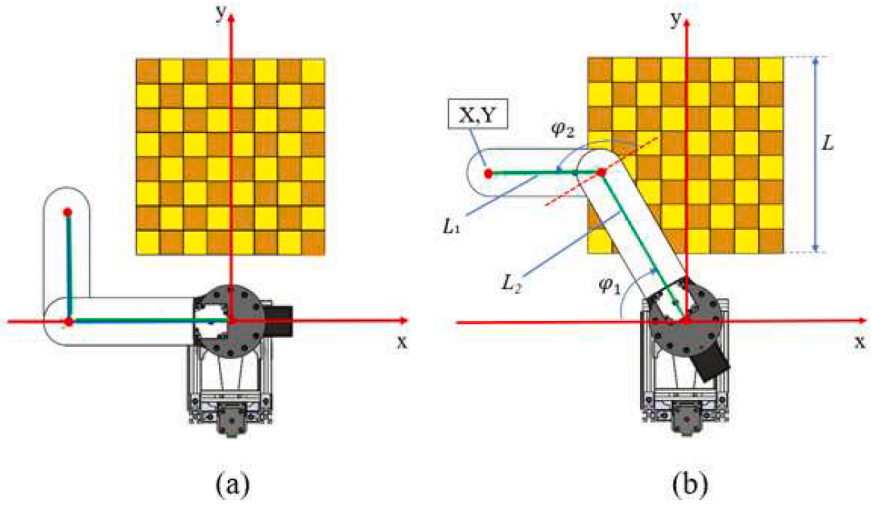


Fig. 3. Robot kinematic model: (a) Home position; (b) Arbitrary position.

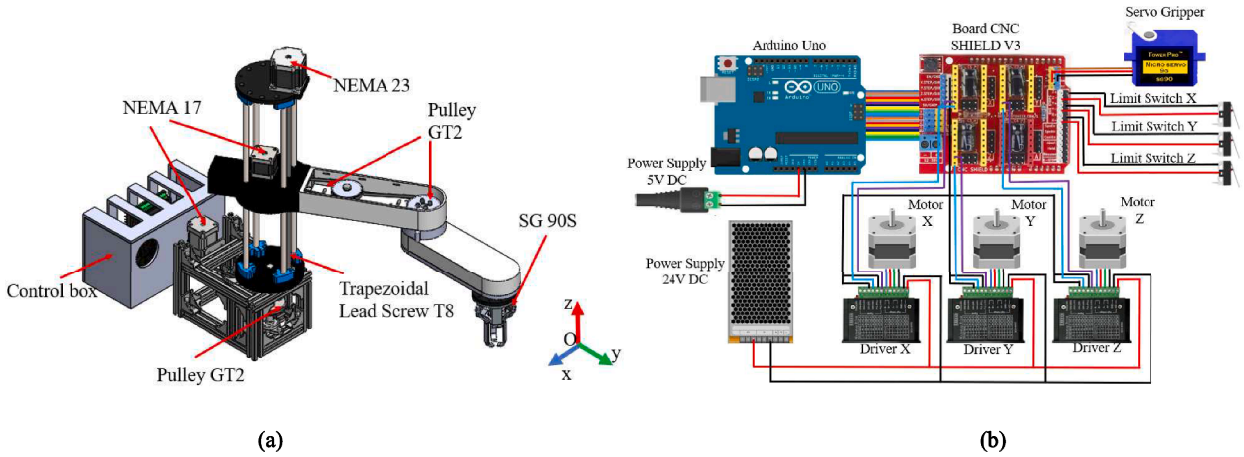


Fig. 4. Robot transmission and control systems: (a) Transmission mechanism (b) Control system.

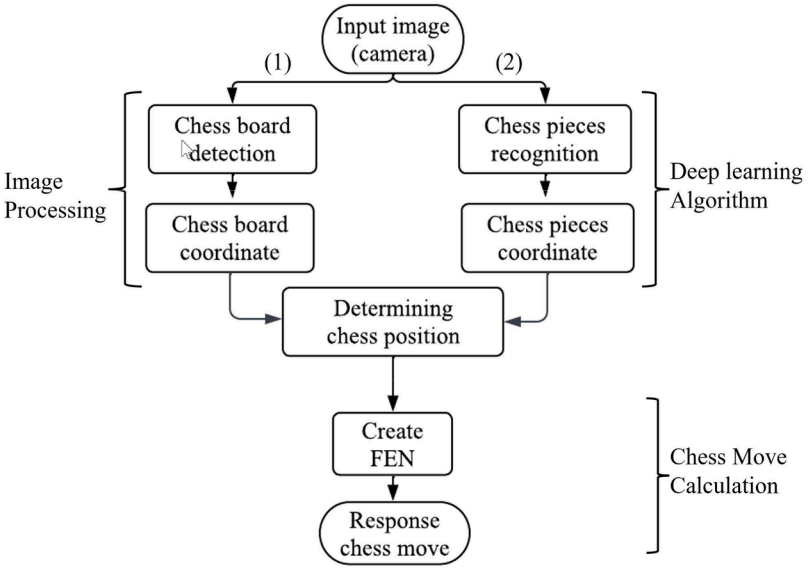


Fig. 5. Flow chart of image processing process for the chess robot system.

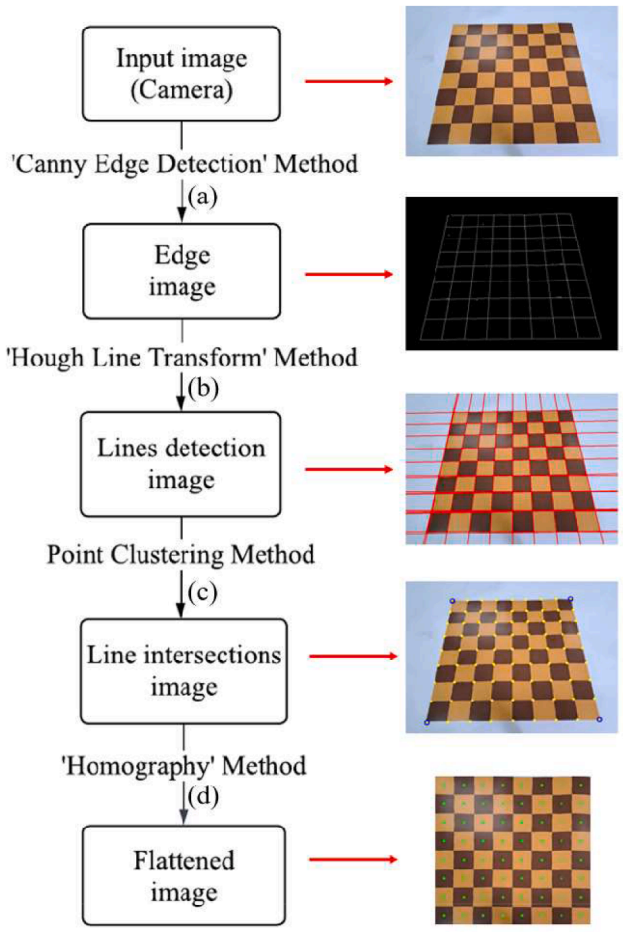


Fig. 6. The chessboard detection process.

orthogonal view. The steps in Fig. 6 are detailed in the following subsections.

(a) Edge detection using the Canny algorithm [25]: First, the captured color image of the chessboard is converted to a grayscale image. Next, the grayscale image undergoes a noise reduction step through Gaussian blur for smoothing. This is achieved using the image convolution technique with a 5 × 5 Gaussian filter kernel. The equation for a Gaussian filter kernel of size $(2k+1) \times (2k+1)$ is expressed as:

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \le i.j \le (2k+1) \tag{3}$$

Next, the gradient is calculated for the smoothed image to detect edge intensity and direction. Edges can be identified by utilizing filters that highlight intensity changes in both the horizontal direction (x direction) and the vertical direction (y direction). This is due to the fact that the gradient will change most significantly at the edge when moving in the x or y direction from one chessboard square to the adjacent ones. The magnitude of the gradient (G) and the slope (θ) of the gradient are determined as follows:

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

$$\theta = \tan^{-1}\frac{G_x}{G_y} \tag{5}$$

Where, G_x and G_y represent the first derivatives in the horizontal direction (x-direction) and vertical direction (y-direction), respectively.

Then, non-maximum suppression is applied. This process eliminates pixels that are not deemed part of an edge, ensuring that only thin lines (candidate edges) are preserved. However, these thin lines may exhibit variations in edge intensity, meaning some pixels within the lines are brighter than others. This issue is resolved in the subsequent steps.

Finally, a double threshold filter and edge tracking by hysteresis are applied. The double threshold filter aims to identify three types of pixels: strong (intensity higher than the upper threshold), weak (intensity between the upper and lower thresholds), and non-relevant (intensity lower than the lower threshold). In this paper, the threshold range used for pixel filtering is [100; 200]. It is clear that the strong pixels correspond to the edges, while the non-relevant ones are disregarded. Some of the weak pixels are converted into strong ones (indicating edges) through the hysteresis mechanism, while the others remain non-relevant (not indicating edges). Consequently, thin edges with uniformly high-density pixels are detected.

(b) Line detection using the Hough Line Transform algorithm: The detected grayscale image of the edges from the previous step undergoes the Hough Transform [26] to identify the lines that form the grid of the chessboard. This algorithm is a well-known computer vision technique for detecting shapes such as lines and circles in an image. In brief, the Hough Line Transform algorithm converts these shapes in image space into mathematical representations in a parameter space. It employs a voting mechanism in the parameter space to emphasize potential object candidates. Consequently, this facilitates the detection of shapes through pattern recognition in the transformed space. Fig. 6 (step (b)) demonstrates the line detection of the chessboard image using the Hough Line Transform algorithm. Most lines were detected with high fidelity; however, some lines (particularly in the lower half of the chessboard) were detected with noise (i.e., multiple lines detected for a single original edge). This may be due to uneven light exposure across the image, as the lower half of the chessboard is closer to the camera, resulting in a higher pixel density in this area. This issue is addressed in the following step.

(c) Intersection point detection using a proposed clustering algorithm: There are still noisy straight lines during the line detection process. All detected lines will be clustered based on edge distances and the angle between lines. Fig. 7 illustrates the point clustering proposed in this study; two adjacent lines, where the distance between them is <20 pixels or the angle between them is <60°, are merged and considered as one line. Through experiments, the proposed clustering algorithm proved effective in suppressing noise. As a result, a high-fidelity line intersection image is achieved, as shown in Fig. 6 (step (c)).

(d) Flattening the line intersection image to achieve an orthogonal view of the chessboard using the Homography transform algorithm [27]: The Homography transform seeks to convert the image from one plane to another by multiplying the coordinates of each pixel with the Homography matrix, as illustrated in Eq. (6).

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6}$$

Where, (x, y) and (x', y') are the coordinates of the pixels before and

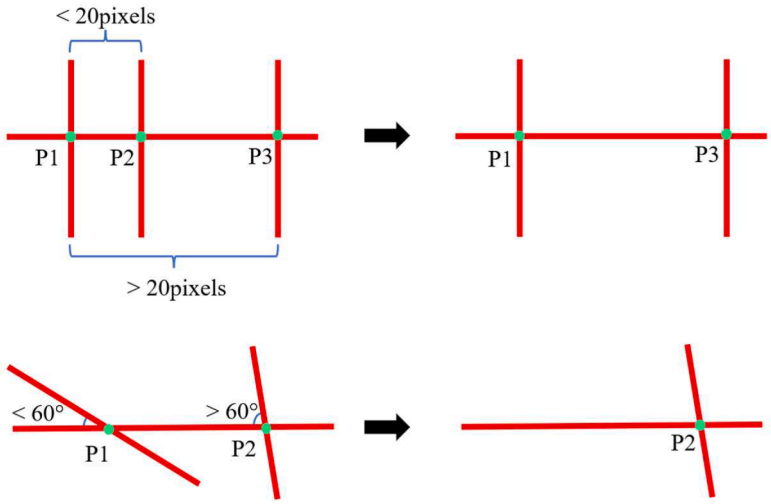


Fig. 7. Point clustering.

after transformation, respectively. H is the Homography matrix.

From the flattened and orthogonal view of the chessboard, the center points of the square chessboard cells are identified. Finally, a reverse Homography transform is employed to ascertain the coordinates of the center points of the square chessboard cells in the pre-transformed image. These center points serve as locations for the robot to pick and place chess pieces while playing chess.

3.2. Deep learning model for chess piece recognition

The purpose of the chess piece recognition process is to identify the precise position of each chess piece on the chessboard. Consequently, the current chess position is detected and analyzed to formulate strategies for subsequent moves. Chess piece recognition can be achieved through several methods. The first is known as template matching (TM) [28], a computer vision technique for locating small parts of an image that match a template image. It involves comparing the template image with various regions of the larger image to find the best match. Typically, this method demonstrates excellent performance when the template image is cropped from the original. To accurately recognize chess pieces, it requires compiling a vast database of chess position images, which is quite time-consuming. Furthermore, the chess piece recognition process is generally lengthy. Another method is known as the histogram of oriented gradients (HOG) [29,30], which counts occurrences of gradient orientation in localized portions of an image. This technique emphasizes the shape of an object, tallying the occurrences of gradient orientation in each local region. It then generates a histogram based on the magnitude and orientation of the gradient. This technique is effective for object detection; however, for chess piece recognition, it is limited by changes in scale (i.e., the size of objects in an image), rotation of the chess pieces (i.e., different orientations of the chess pieces on the chessboard), and occlusion (i.e., when an object is partially or fully obscured by another object in an image or video).

On the other hand, scale-invariant feature transform (SIFT) and speeded-up robust features (SURF) [31] are two widely used techniques for object recognition and image stitching. They involve extracting features from each image, matching these features, and determining the presence of landmarks based on the matches. However, these methods also face limitations in scalability and adaptability. Furthermore, they necessitate manual adjustments or the creation of new templates for different tasks or new object categories. Additionally, they demand intensive computation, particularly during the key point detection and descriptor generation stages.

Recently, deep learning (DL) methods [32], particularly convolutional neural networks (CNNs), have been extensively applied to computer vision, especially for object recognition [16,33]. This is due to their ability to learn and extract features from images both accurately and rapidly. They are well-suited for chess piece recognition, as the chess position can change frequently. Therefore, in this study, a deep learning method is employed for chess piece recognition.

Fig. 8 illustrates the DL model for chess piece recognition in this study. Firstly, the dataset was created by capturing 400 photos of various chess positions. Each chess piece in every photo was then manually labeled with its exact name (i.e., pawn, rook, knight, bishop, queen, king for both white and black sides). After labeling, the dataset was augmented to 1000 photos through image processing techniques such as symmetric transformation, flipping, rotation, and variations in brightness and exposure. The image augmentation techniques demonstrated significant improvement in model generalization and robustness, particularly when training on small or imbalanced datasets [34]. The complete dataset (i.e., 1000 images after labeling) is utilized for the DL model and divided into three sets: training set (60 % of data), validation set (20 % of data), and testing set (20 % of data). The YOLOv8x model [35] was employed for training the chess piece recognition in the DL model.

Fig. 9 shows the evaluation results of the DL model in chess piece recognition. The horizontal axis represents the number of epochs, which indicates the complete passes through the training dataset, while the vertical axis displays the loss function metrics as shown in Fig. 9(a) to Fig. 9(d) and precision as depicted in Fig. 9(e). It is observed that the quality index of the DL model improves rapidly (i.e., loss decreases and precision increases) as the number of epochs rises from 0 to 20. The quality index of the DL model increases more slowly when the number of epochs exceeds 20, reaching saturation when the number of epochs hits 50. Therefore, an epoch number of 50 is selected for the effective implementation of the DL model in this study.

Fig. 10 presents the results of the intersection over union (IOU) index for chess piece recognition in one chess position example using the DL model with 50 epochs. The IOU index is a crucial indicator that reflects the quality of chess piece recognition; a higher IOU value signifies better quality and greater accuracy in the recognition process [36]. As shown in Fig. 10(b), all chess pieces are recognized correctly, with IOU values ranging from 0.78 to 0.9. It is noted that the IOU value for recognizing each individual chess piece strongly depends on its shape, color, and position. This is due to the fact that different shapes and positions of a chess piece yield distinct features for their recognition. Additionally, variations in the positions and orientations of chess pieces on the board can affect the quality of the captured images. It has been reported that chess piece recognition is considered good and acceptable if the IOU value exceeds 0.5 [36]. Consequently, the chess piece recognition in the

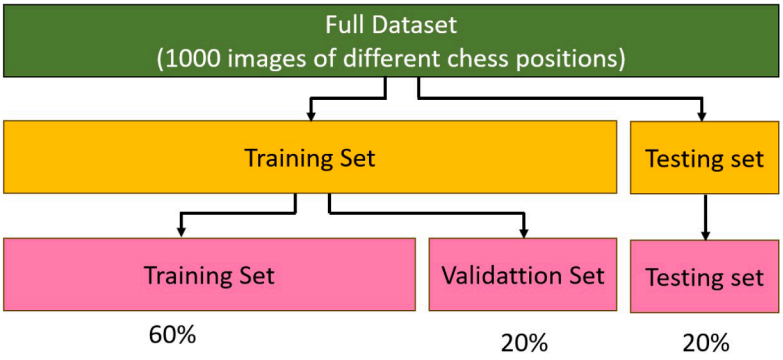


Fig. 8. Data spliting for deep learning model.

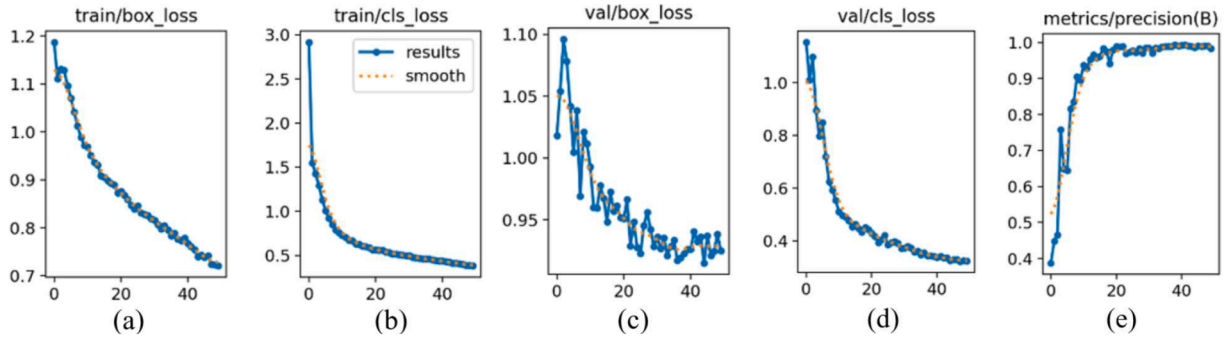
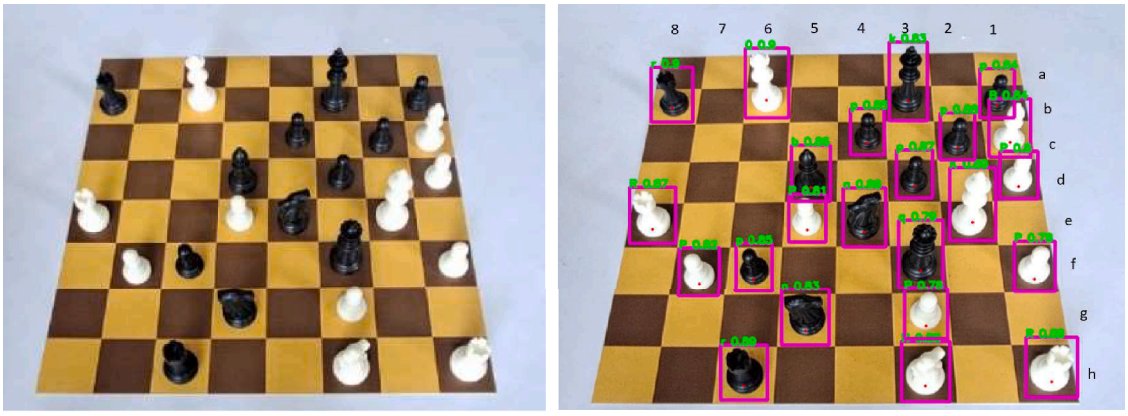


Fig. 9. Deep learning model training results.



(a) original image (b) IOU values of chess piece recognition

Fig. 10. IOU value of chess piece recognition of the deep learning model.

DL model developed in this study is both precise and accurate. Furthermore, the total processing time for determining one chess position (including chessboard detection through image processing and chess piece recognition via the DL algorithm) is under 1 second (using a GTX 1650Ti GPU). Thus, the system is suitable for real-time chess playing applications.

Fig. 11 displays the normalized confusion matrix for the chess piece recognition by the deep learning model. It is observed that the chess piece classification model demonstrates high accuracy across most pieces, particularly with the black knight, black pawn, black queen, black rook, white knight, white pawn, white queen, and white rook. However, some minor confusions persist, notably between the black bishop and black king, as well as issues with the background layer (which occurs with certain white chess pieces), where some white pieces are misidentified as background at rates ranging from 7 % to 40 %. To address this, the authors are enhancing the background data and bishop variants in the training set, along with modifying the loss function to lower the background confusion rate. These enhancements could enable the model to achieve more precise recognition and improve performance in real-world scenarios. The results will be presented in a future publication.

4. Chess move calculation using minimax algorithm

With chess piece recognition achieved, the current chess position can be analyzed to determine the next move using the Forsyth–Edwards

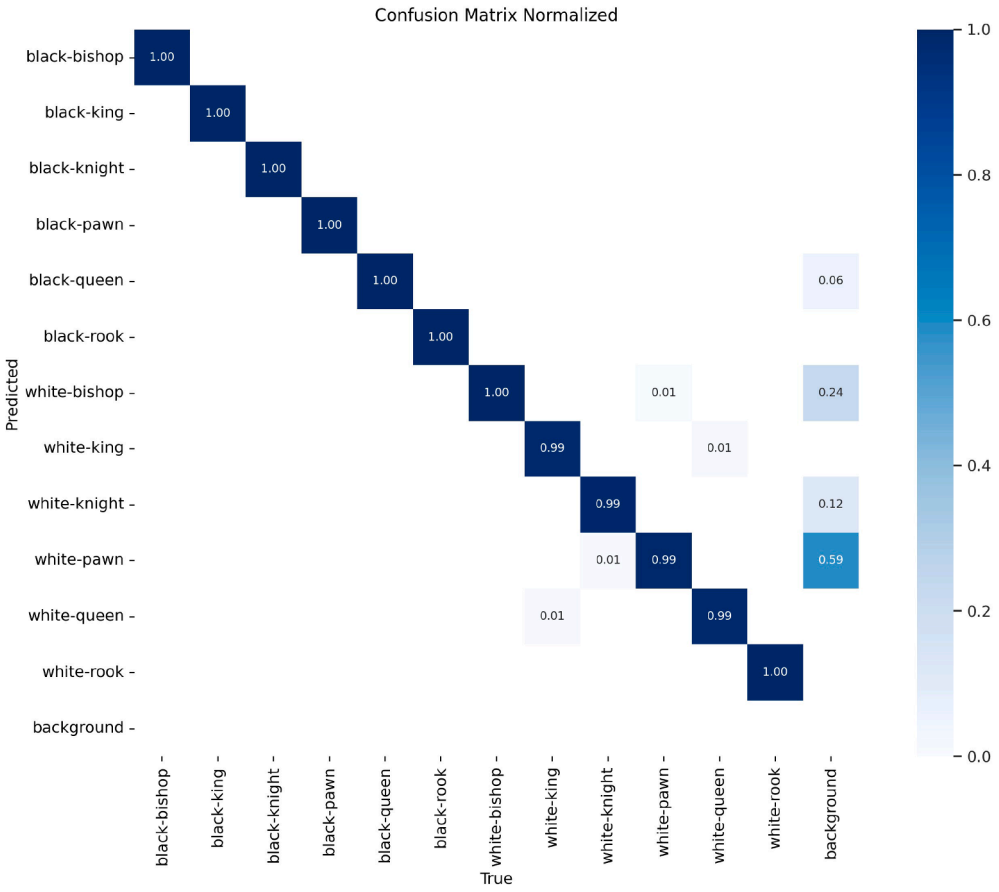
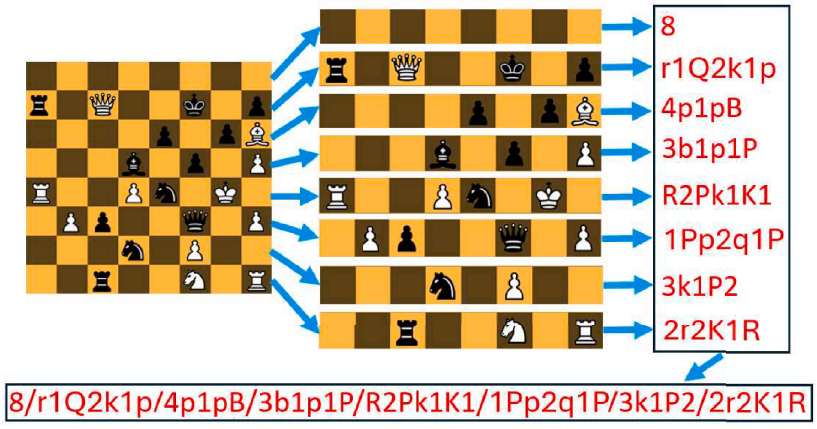


Fig. 11. Normalized confusion matrix of the chess piece recognition of the deep learning model.

(a) FEN code for chess pieces



(b) FEN code for a chess position

Fig. 12. FEN code for chess pieces and chess position.

Notation (FEN) method, which presents a chess position as a single string of characters [37]. In the FEN method, each chess piece is denoted by a letter; uppercase letters represent white pieces, while lowercase letters denote black pieces, as shown in Fig. 12(a). Empty squares are indicated by a number corresponding to the count of empty squares in each row of the chess position. The '/' symbol separates the FEN codes

for each row of the chess position. Fig. 12(b) provides an example of a chess position represented using FEN code. After determining the FEN code for a chess position, the next move is calculated using the Stockfish chess engine [38]. This engine employs the minimax algorithm [39] to evaluate potential moves by calculating scores for all possible options.

The essence of this algorithm lies in the fact that the two players are adversaries: one is referred to as MAX and the other as MIN. Each player aims to maximize their own score while minimizing their opponent's benefit. The algorithm constructs the entire game tree and applies a utility function to derive utility values for the terminal states. Subsequently, the minimax algorithm conducts a depth-first search to explore the complete game tree. It continues down to the terminal nodes and then backtracks through the tree via recursion. The advantage of this algorithm is that it recommends the next optimal chess move that yields the highest score among all possible moves.

5. Result and discussion

Fig. 13 illustrates the image processing workflow and results for a chess position example, aimed at determining the FEN code and calculating the next chess move for the chess robot system during a game. Initially, a photo of the current chess position is captured by the camera system. This photo is utilized for chess piece recognition to identify the chess position using a deep learning method. Subsequently, a FEN code is generated for the captured chess position. The FEN code is then employed to calculate the next chess move using the minimax algorithm. The result indicates that the FEN code for the next chess move is identified as "g6e5". From the next chess move location, the coordinates (X, Y) of the chess piece are determined. Consequently, the rotation angles φ_1 and φ_2 are calculated by solving the inverse kinematic equations (2), and the values of φ_1 and φ_2 are displayed in Fig. 13. Statistically, the entire image processing procedure for a chess position took approximately 2 s. It is noted that it takes approximately 20 to 30 s for a chess player to calculate a chess move in a standard chess game [40]. In addition, during a chess game between top players, especially in high-stakes moments such as endgame scenarios or pivotal positions, players tend to allocate even more time to ensure optimal decisions [40, 41]. Therefore, the system developed in this study is feasible for real-world chess playing applications.

Fig. 14 illustrates the operation of the SCARA robot after calculating the rotation angles φ_1 and φ_2 . Initially, the robot is in the home position (as shown in Fig. 14(a)); it then moves to grasp the chess piece at position e5, corresponding to $\varphi_1 = 134^\circ$ and $\varphi_2 = 208.2^\circ$ (as depicted in Fig. 14(b)). The robot releases the chess piece outside the chessboard (as seen in Fig. 14(c), since the chess piece, specifically the white pawn, is captured in the current move). Next, the robot grips the chess piece at g6 (the black knight) corresponding to $\varphi_1 = 155.7^\circ$ and $\varphi_2 = 214^\circ$ (as shown in Fig. 14(d)); it then releases the chess piece at e5, corresponding to $\varphi_1 = 134^\circ$ and $\varphi_2 = 208.2^\circ$ (as illustrated in Fig. 14(e)). Finally, the robot returns to the home position (as in Fig. 14(f)). It is found that the robot operates accurately for each chess move, and the entire chess robot system functions steadily and precisely, and the total time for the robot to finish this chess move is approximately 60 s.

It is noted that the processing time for the robot to execute a chess move can vary based on the specific move. In this study, a simple move takes about 20 s for the robot to complete, while a capture move requires

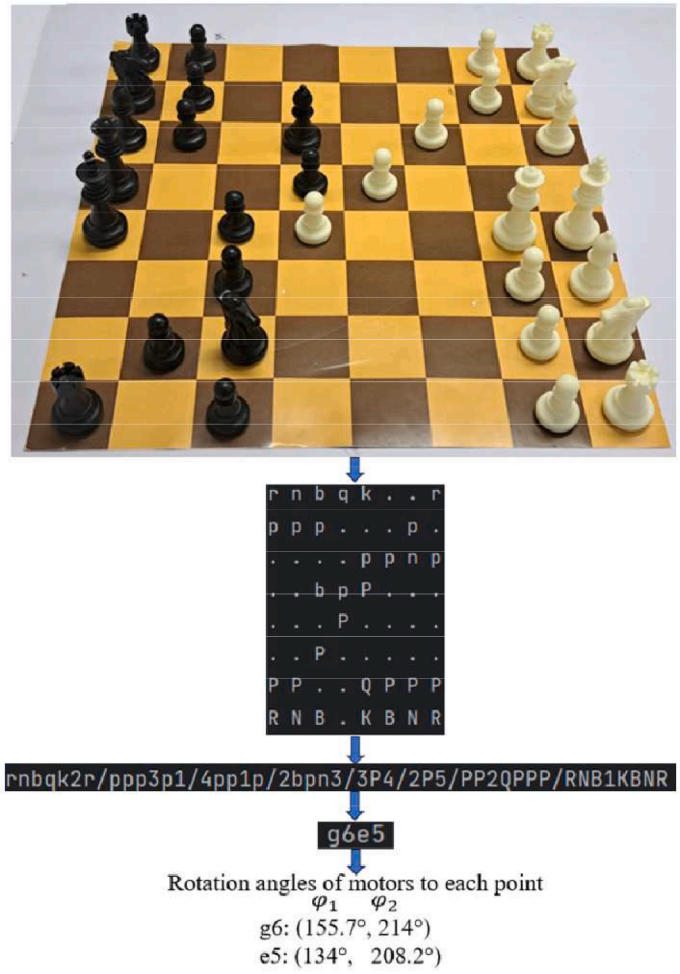


Fig. 13. Image processing process of a chess position.

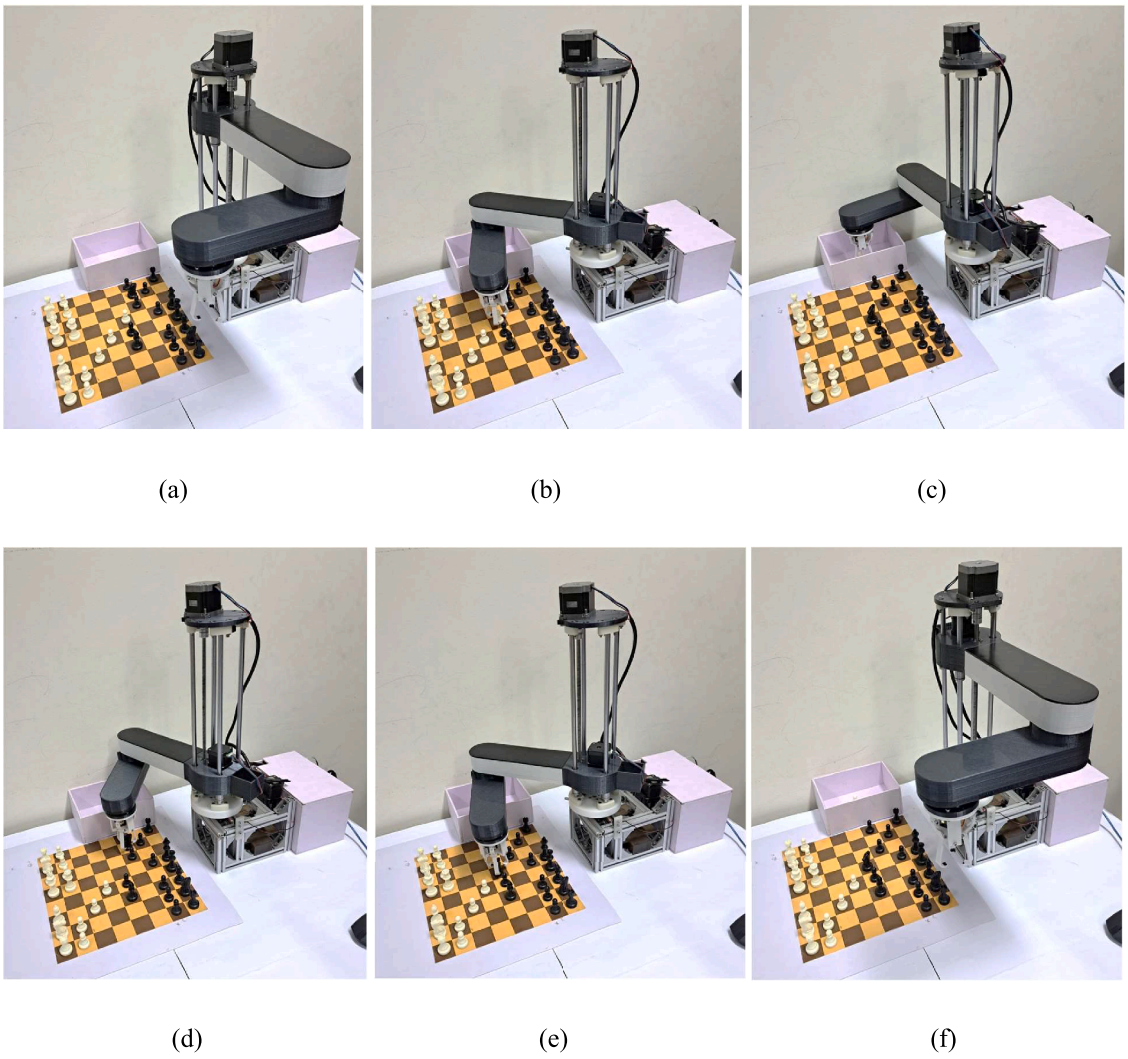


Fig. 14. Robot's movements in a chess position: (a) home position (0); (b) Gripping chess piece at position e5 (134°, 208.2°); (c) Releasing chess piece outside the board (release position); (d) Gripping chess piece at g6 (155.7°, 214°); (e) Releasing chess piece at e5 (134°, 208.2°); (f) Back to home position.

approximately 60 s, as the robot must pick up the captured piece and place it in the storage box, then retrieve and position the captured chess piece. Conversely, a pawn promotion move takes roughly 90 s, since the robot needs to return to the storage box to select the desired promotion chess piece.

It is important to note that processing time can be further reduced in for real-time chess playing if the speed of the robot's driving motor is increased. However, the robot's structure must possess sufficient durability to ensure stable operation. In this case, the robot shoud have high speed driving motors and suffient toughness structure. Additionally, one might question the positional errors of a chess piece on the board caused by the SCARA robot during the pick-and-place process. This concern is mitigated because the robot returns to its home position (i.e., the zero position) defined by a limit switch after each chess move. Consequently, accumulated errors from the control system and mechanical movement are avoided. Furthermore, the gripper mechanism features a self-centering design with a large grasping aperture diameter (ranging from 18 mm to 32 mm), which exceeds the diameter of a chess piece. If there is a minor positional error with a chess piece, the robot can still successfully pick it up and place it accurately.

The developed chess robot system demonstrates significant potential as an affordable, efficient tool for both entertainment and educational purposes. Future improvements could involve increasing motor speed, enhancing mechanical durability, and improving the vision model to handle diverse environmental conditions. These enhancements would further improve the robot's performance and broaden practical applications of the system beyond chess, such as in industrial assembly and object manipulation tasks, and industrial vision inspection systems.

The developed chess robot system demonstrates significant potential as an affordable and efficient tool for both entertainment and educational purposes. Future improvements could involve increasing motor speed, enhancing mechanical durability, and refining the vision model to handle diverse environmental conditions. These enhancements would further improve the robot's performance and expand the practical applications of the system beyond chess, such as in industrial assembly, object manipulation tasks, and industrial vision inspection systems.

6. Conclusion

In this paper, an autonomous chess robot system is successfully developed through a combination of computer vision, deep learning, and robot control. A deep learning model is implemented for chess piece recognition and chess position detection. Chess move calculations are performed using the minimax algorithm within the Stockfish chess engine. Additionally, a 4-DOF SCARA robot is designed for picking and placing chess pieces during the game. The results demonstrate that the developed deep learning model achieves high accuracy in chess piece recognition across most pieces, particularly with the black knight, black

pawn, black queen, black rook, white knight, white pawn, white queen, and white rook, with IOU index values ranging from 0.78 to 0.9. Furthermore, the computation time for a chess move is approximately 2 s per position. A simple move takes about 20 s for the robot to complete, while a capture move requires approximately 60 s. Meanwhile, a pawn promotion move takes roughly 90 s. The processing time for the robot to pick and place a chess piece can be further reduced for real-time chess playing if the robot is equipped with high-speed driving motors and a sufficiently robust structure. The results confirm that the developed chess robot system operates automatically, stably, and accurately. This chess robot system can play a complete game against a human or solve chess positions for pre-arranged setups. The success of this research paves the way for several future applications, including entertainment robot systems designed to teach beginners how to play chess. Moreover, the use of image processing for recognizing chessboards and pieces can be adapted to address challenges such as identifying bolts and nuts, as well as automating the picking of nuts or screws to tighten bolts and nuts - an application commonly found in assembly lines. Additionally, image processing can be utilized for the inspection and classification of mechanical parts, such as automatically measuring hole diameters or identifying missing components on printed circuit boards.

CRediT authorship contribution statement

Truong Duc Phuc: Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Conceptualization. **Bui Cao Son:** Software, Investigation, Formal analysis, Data curation.

Declaration of competing interest

The authors declare no conflict of interest.

Data availability

Data will be made available on request.

References

[1] C.E. Shannon, Programming a computer for playing chess, Phil. Mag. 41 (1950) 256–275.
[2] W.G. Chase, H.A. Simon, Perception in chess, Cogn. Psychol. 4 (1) (1973) 55–81.
[3] W.G. Chase, H.A. Simon, The mind's eye in chess. Visual Information Processing, Academic Press, 1973, pp. 215–281.
[4] H. Simon, W. Chase, Skill in chess. Computer Chess Compendium, Springer New York, New York, NY, 1988, pp. 175–188.
[5] https://www.chess.com/.
[6] https://lichess.org/.
[7] K. Dhou, An innovative employment of virtual humans to explore the chess personalities of Garry Kasparov and other class-A players, in: HCI International 2019-Late Breaking Papers: 21st HCI International Conference, HCII 2019, Springer International Publishing, Orlando, FL, USA, 2019, pp. 306–319. July 26-31, 2019, Proceedings 21.
[8] K. Dhou, A novel investigation of attack strategies via the involvement of virtual humans: a user study of Josh Waitzkin, a virtual chess grandmaster, in: HCI International 2020-Late Breaking Papers: Cognition, Learning and Games: 22nd HCI International Conference, HCII 2020, Springer International Publishing, Copenhagen, Denmark, 2020, pp. 658–668. July 19-24, 2020, Proceedings 22.
[9] Cynthia Matuszek, Brian D. Mayton, Roberto Aimi, Marc Peter Deisenroth, Liefeng Bo, Robert Chu, Mike Kung, Louis Legrand, Joshua R. Smith, Dieter Fox, Gambit: an autonomous chess-playing robotic system, in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 4291–4297.
[10] Varun Gupta, Amit Kumar, Saumya Jaiswal, Shilpi Agrawal, Autonomous chess playing robot, IJERT 4 (3) (2015).
[11] H.M. Luqman, M. Zaffar, Chess brain and autonomous chess playing robotic system, in: 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC), Bragança, Portugal, 2016, pp. 211–216, https://doi.org/10.1109/ICARSC.2016.27.
[12] Andrew Tzer-Yeu Chen, Kevin I-Kai Wang, Robust computer vision chess analysis and interaction with a humanoid robot, Comput. 8 (2019) 14.
[13] K. Shin, et al., Exploring the capabilities of a general-purpose robotic arm in chess gameplay, in: 2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids), Austin, TX, USA, 2023, pp. 1–8, https://doi.org/10.1109/Humanoids57100.2023.10375209.
[14] Hack, J.; Ramakrishnan, P. CVChess: computer Vision Chess Analytics, 2014. Available online: http://cvgl.stanford.edu/teaching/cs231a_winter1415/prev/projects/chess.pdf (accessed on 7 February 2019).
[15] K. Vinoth, P. Sasikumar, Lightweight object detection in low light: pixel-wise depth refinement and TensorRT optimization, Results Eng. 23 (2024) 102510, https://doi.org/10.1016/j.rineng.2024.102510.
[16] Gengyan Cui, Li Zhang, Improved faster region convolutional neural network algorithm for UAV target detection in complex environment, Results Eng. 23 (2024) 102487, https://doi.org/10.1016/j.rineng.2024.102487.
[17] J.E. Neufeld, T.S. Hall, Probabilistic location of a populated chessboard using computer vision, in: InProceedings of the IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), Seattle,WC, USA, 2010, pp. 616–619, 1–4 August.
[18] K.Y. Tam, J.A. Lay, D. Levy, Automatic grid segmentation of populated chessboard taken at a lower AngleView, in: Proceedings of the Digital Image Computing: Techniques and Applications (DICTA), Canberra,Australia, 2008, pp. 294–299, 1–3 December.
[19] S. Bennett, J. Lasenby, ChESS – quick and robust detection of chess-board features, Comput. Vis. ImageUnderst. 118 (2014) 197–210.
[20] S. Kucuk, Z. Bingul, Robot kinematics: forward and inverse kinematics. Industrial Robotics: Theory, Modelling and Control, IntechOpen, 2006. Dec.
[21] J.M.McCarthy Perez, Sizing a serial chain to fit a task trajectory using Clifford algebra exponentials, in: IEEE international conference on robotics and automation, 2005, pp. 4709–4715.
[22] Jianxin Xu, Wei Wang, Yuanguang Sun, Two optimization algorithms for solving robotics inverse kinematics with redundancy, J. Control Theory Appl. 8 (2) (2010) 166–175.
[23] M. Lutz, LearningPytho, O'Reilly Media, Inc, Sebastopol, 2013.
[24] A. Meurer, C.P. Smith, M. Paprocki, O. Čertík, S.B. Kirpichev, M. Rocklin, Am. Kumar, S. Ivanov, J.K. Moore, S. Singh, T. Rathnayake, S. Vig, B.E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, A. Scopatz, SymPy: symbolic computing in Python, PeerJ Comput. Sci. 3 (2017) e103, https://doi.org/10.7717/peerj-cs.103. PeerJ.
[25] OpenCV Docs: canny Edge Detection.
[26] OpenCV Docs: hough Line Transform.
[27] OpenCV Docs: homography Transform.
[28] M. Sultana, T. Ahmed, P. Chakraborty, M. Khatun, Md. Rakib, M. Shorif, Object detection using template and HOG feature matching, Int. J. Adv. Comput. Sci. Appl. 11 (7) (2020), https://doi.org/10.14569/ijacsa.2020.0110730. The Science and InformationOrganization.
[29] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), IEEE, 2005, https://doi.org/10.1109/cvpr.2005.177.
[30] Muhammad Asif, Mohsin I. Tiwana, Umar S. Khan, Muhammad W. Ahmad, Waqar S. Qureshi, Javaid Iqbal, Human gait recognition subject to different covariate factors in a multi-view environment, Results Eng. 15 (2022) 100556, https://doi.org/10.1016/j.rineng.2022.100556.
[31] P. Umale, A. Patil, C. Sahani, A. Gedam, K. Kawale, Planer object detection using surf and SIFT method, Int. J. Eng. Appl. Sci. Technol. 6 (11) (2022) 36–39, https://doi.org/10.33564/ijeast.2022.v06i11.008. International Journal of Engineering Applied Sciences and Technology.
[32] K. Vaishnavi, G.P. Reddy, T.B. Reddy, N.Ch.S. Iyengar, S Shaik, Real-time object detection using deep learning, J. Adv. Math. Comput. Sci. 38 (8) (2023) 24–32, https://doi.org/10.9734/jamcs/2023/v38i81787. Sciencedomain International.
[33] Ha Quang Thinh Ngo, Design of automated system for online inspection using the convolutional neural network (CNN) technique in the image processing approach, Results Eng. 19 (2023) 101346, https://doi.org/10.1016/j.rineng.2023.101346.
[34] Seunghyeon Wang, Evaluation of impact of image augmentation techniques on two tasks: window detection and window states detection, Results Eng. 24 (2024) 103571, https://doi.org/10.1016/j.rineng.2024.103571.
[35] Ultralytics YOLOv8 Docs.
[36] R. Padilla, S.L. Netto, E.A.B. Da Silva, A survey on performance metrics for object-detection algorithms, in: 2020 international conference on systems, signals and image processing (IWSSIP), IEEE, 2020, pp. 237–242.
[37] Iqbal, An algorithm for automatically updating a forsyth-edwards notation string without an array board representation, in: 2020 8th International Conference on Information Technology and Multimedia (ICIMU), Selangor, Malaysia, 2020, pp. 271–276, https://doi.org/10.1109/ICIMU49871.2020.9243487.
[38] Stockfish - Open Source Chess Engine (stockfishchess.org).
[39] W.B. Putra, L. Heryawan, Applying alpha-beta algorithm in a chess engine, Jurnal Teknosains 6 (1) (2017) 37, https://doi.org/10.22146/teknosains.11380. Universitas Gadjah Mada.
[40] Chang, Lane, There is time for calculation in speed chess, and calculation accuracy increases with expertise, Am J Psychol 129 (1) (2016) 1, https://doi.org/10.5406/amerjpsyc.129.1.0001.
[41] Yu-Hsuan A. Chang, David M. Lane, It takes more than practice and experience to become a chess master: evidence from a child prodigy and adult chess players, J. Expertise 1 (1) (2018). /June/, https://api.semanticscholar.org/CorpusID:201684343.