DETECTING THE BEGINNING OF THE ACTUAL CHESS GAME IN THE INPUT VIDEO:

After reading a video of a chess game detecting the first frame of the game became the preliminary task. It is obvious to see that we get the same pattern of the board if the frame is rotated by 90 degrees. Using this method for all in upcoming frames, it is checked if they have the same edge pattern in both the configurations. Canny edge detection is procedure for that.

Detecting the first frame was so crucial that the entire pre-processing is done on the same frame and later the results of which were carried forward. In accuracy issues this stage took major part of the project.

For the given input video every frame was processed using the following algorithm at the rate equal to twice the frame rate of the video.

ALGORITHM:

*Step1: Input RGB 24 Bit image.

*Step2:Conversion to Gray Scale

*Step3: Edge Detection using the Canny Filter.

*Step 4: Plotting the Sum of Canny Edge Detected Image:
The sum of all the image pixels is taken along the columns. Here we could note that along the columns where the edges are present, maximum number of pixels were accumulated. In an image the number of peaks were calculated which correspond to edges.

*Step5: Similar procedure is followed for the same image rotated by 90 degrees.
And number of Peaks were calculated. Ideally we had to get equal number of peaks, equal to 8. But because of presence of chess pieces and noise we got a difference 2 to 3 peaks. By adjusting the threshold pseudo frames were ignored.

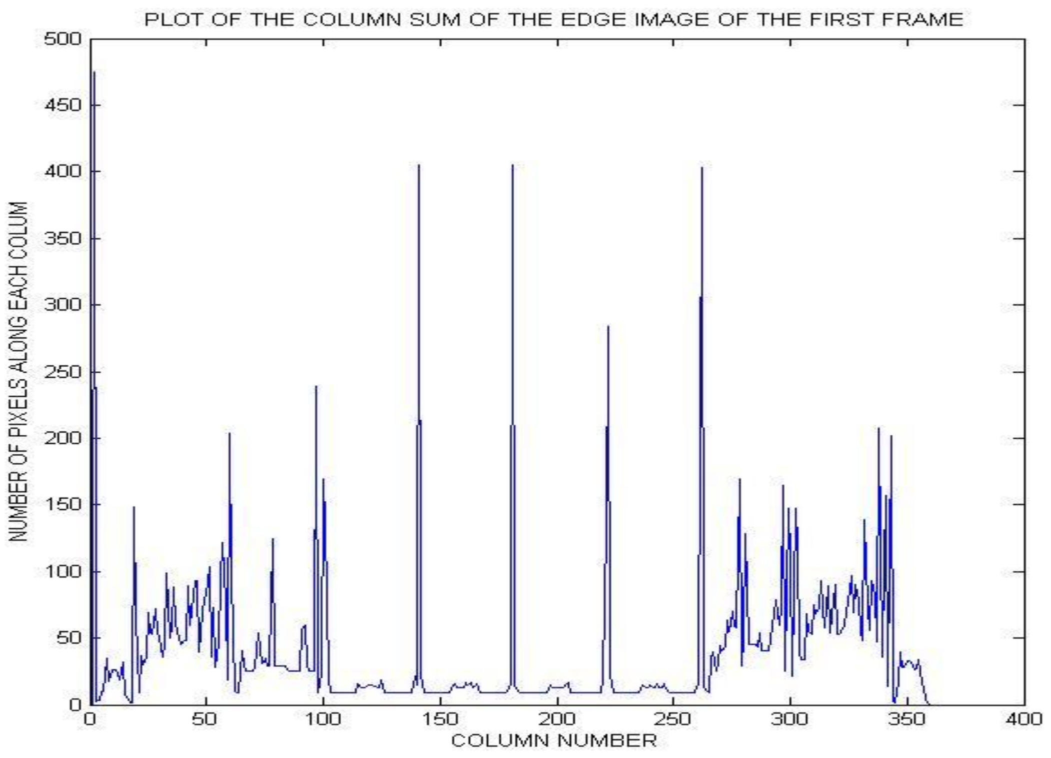
*Step6: Starting from the first frame of the image the above procedure is followed till we get the First Frame after which the actual game begin.

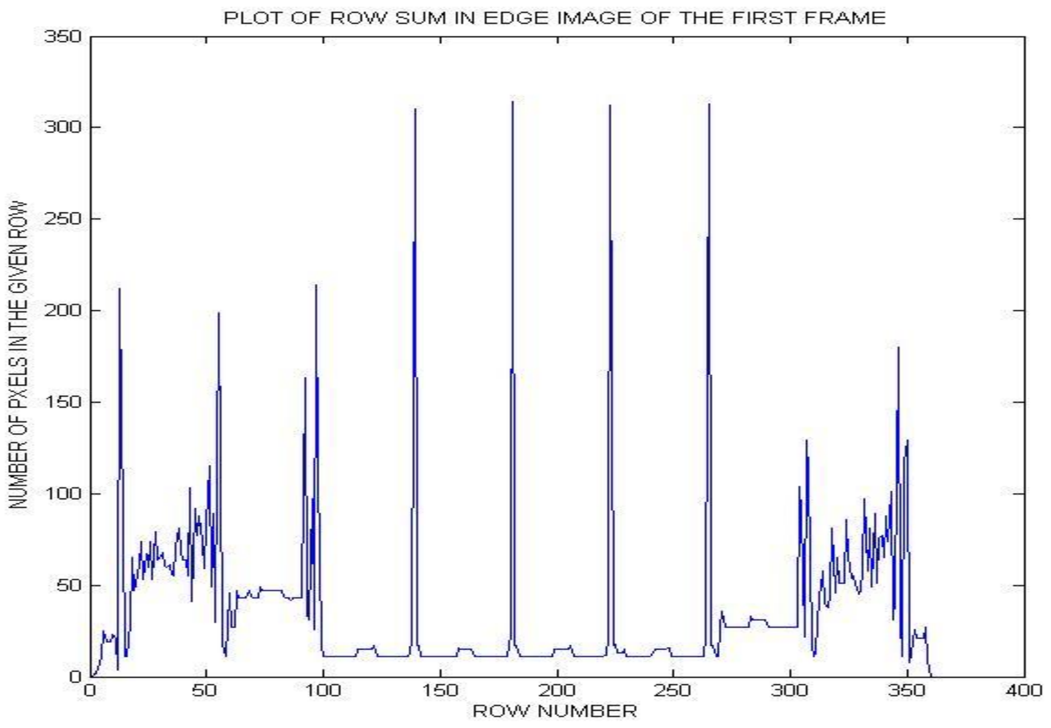
ISSUES FACED

*Initially it is decided to recognize the beginning by of the game depending upon the statistical quantities such as deviation, variance, mean etc. But the results were not reliable because of which they had to be dropped.

*Acquiring the histogram of the image was the next approach. That is by picking up the number of local maxima in a frame. We knew that this number had to be ideally 5 that is, 4 because of the pieces and squares of two colours and another one due to the surrounding background. But the presence of noise in the frame spread out the histogram spectra through the full scale from 0 to 255. This ruined the maxima and hence the procedure.

DIAGRAMATIC DESCIPTION





DESCRIPTION

The plots of the sum of the pixels corresponding to the each columns and rows are as shown in the above figure. As we can see the number of maxima are around 8 pertaining to the criteria of having more than half the dimension in that particular direction.

Depending on the graph shown above the first frame was recognised. And later the entire processing was done on this particular frame.

Below we have the image of the first frame extracted from the video. This algorithm gave accurate results for most of the videos by neglecting the initial frames having comprising of people, disturbances, etc..

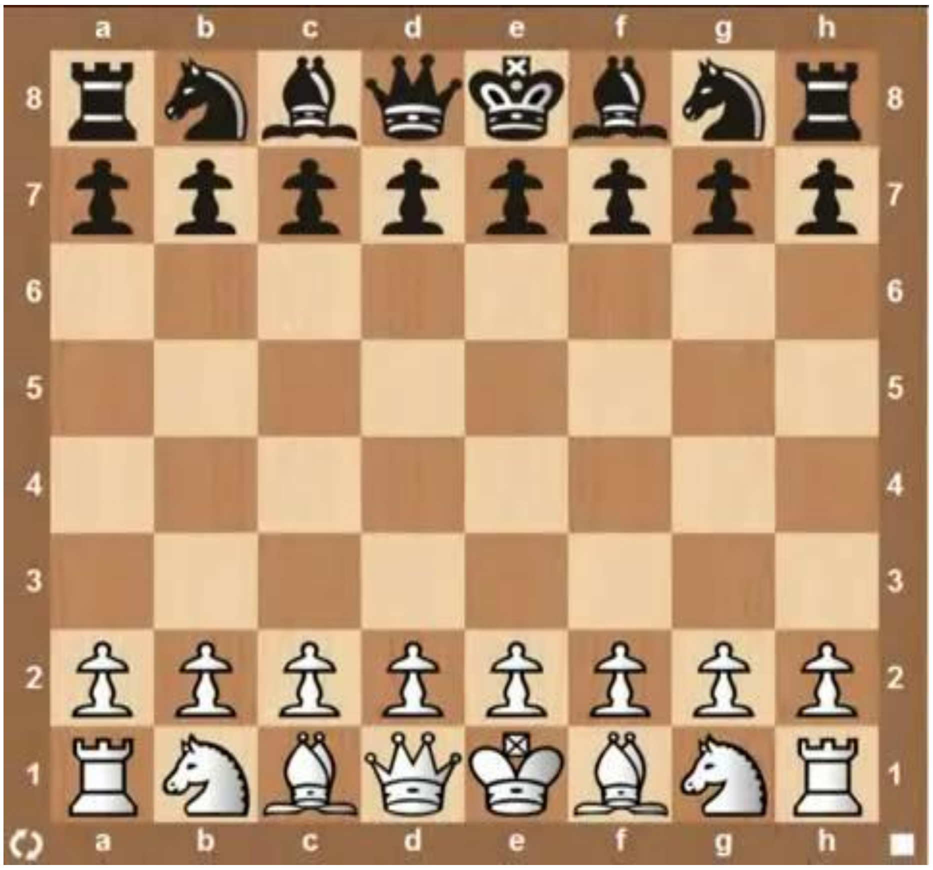


FIGURE: FIRST FRAME DETECTED FROM THE VIDEO

SIZE OF THE CHESS SQUARE

In this stage of processing the size of the 64 squares in the image is determined. Input to the process is the first frame acquired from the previous image. The detection starts from the central pixel in random.

In a chess board we know that, for the first frame extracted in the previous step the central four rows are unoccupied. Using this condition we can easily determine the dimension of each chess square.

Starting from the upper side, each pixel is checked for the gradient in the intensity value. We pursue in each direction till a pixel with new intensity is obtained.

Similarly the border of the square is detected in all four directions. After getting the coordinates of the borders, height of the square is obtained by subtracting the lower border co-ordinate from that of the upper border. In the same way, width is obtained by subtracting the left side border from the right side border. Thus we can get the dimensions of the single square.

<u>ALGORITHM</u>

*Step1: Start from the central pixel at random.

*Step2: Move upwards by decrementing the row number till a difference of intensity value is obtained. We will get the position of top border of a square.

*Step3: Similarly bottom border position is obtained by incrementing the row index. The difference of these two rows will give the height of the square.

*Step4: If we follow the Step3 and Step4 in the horizontal direction we shall get the width of the square.

<u>ISSUES FACED</u>

*In some of the videos, the shading within the square was detected as the border. In such cases the threshold of the difference had to be kept pretty high to avoid them.

*The starting point being the corner of the square is avoided by moving the the same by 5 pixels to the right.

<u>EXTRACTING THE CHESS BOARD FROM THE ENTIRE IMAGE</u>

Initially an approximate position of the row and column corresponding to the borders is estimated by simply adding the multiple dimensions of the square in each direction to the central square obtained in the previous step.

In the later part of the design, the proper edge is searched for the accuracy of the task. In the similar manner as done in detecting the beginning of the game, the extracted frame in converted into gray-scale image followed by the canny edge filter. We know that, along the borders of the chess board there will be a thick edges detected in the filter. Using this fact, we could determine the presence of the borders.

So sum of the edge image obtained in the previous stage is calculated for each columns. The column corresponding to the maximum number of pixels in the vicinity of the approximate border gave the proper edges.

The accuracy and reliability of the entire algorithm was solely dependent on the the results of this processing step. So the algorithm had to be absolutely error free.

ALGORITHM

*Step1: Input RGB 24 Bit image.

*Step2: Conversion of the image to Gray Scale

*Step3: Approximate border detection.
Upper border = Central square corner - 4 * square height
Lower border = Central square corner + 4 * square height
Right Side border = Central square corner + 4 * square width
Left Side border = Central square corner - 4 * square width

*Step3: Edge Detection using the Canny Filter.

*Step 4: Plotting the Sum of Canny Edge Detected Image:
In the neighbourhood of the above borders obtained we check for the local maxima. If any column correspond to a high number of pixels than the estimated one, then that particular column is selected to be the vertical border of the chess board.

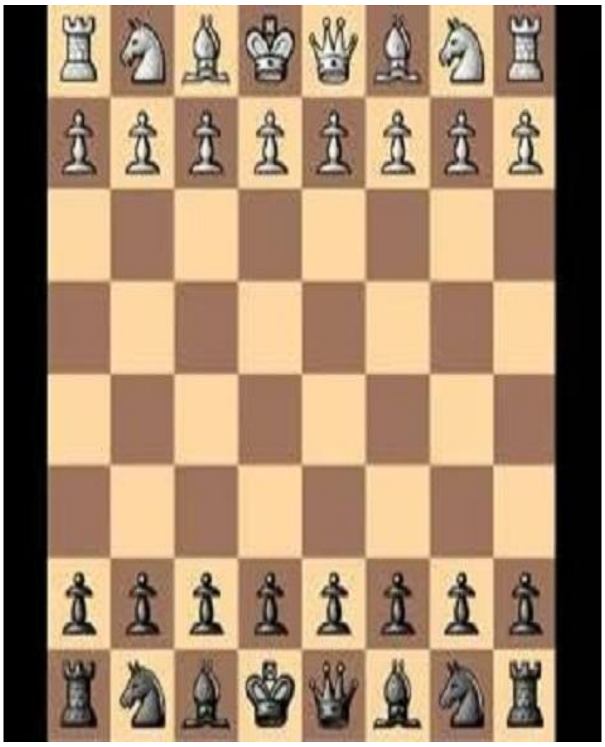
*Step5: Similar procedure is followed for the image rotated by 90 degrees.
This step gives us the horizontal borders of the chess board.

ISSUES FACED

*Determining the size of the neighbourhood for searching the edge became the major task. Some of the videos had Digits written along both the sides of board, which were predicted to be the edges. This led to wrong results.

*Some of the chess boards used had thick fancy borders. In such cases the edge closer to the central pixel was selected to be the proper border.

DESCRIPTION OF THE IMAGES:
The videos which had the background as shown in the figure had multiple borders and led to the wrong detection of the borders.



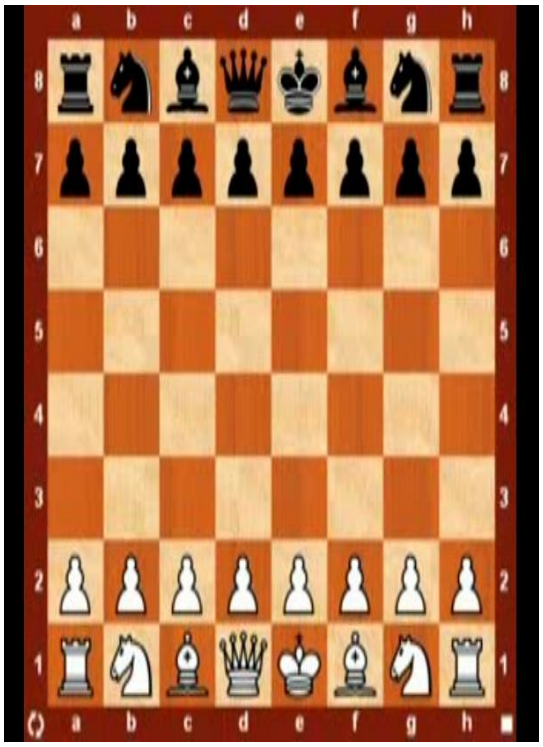
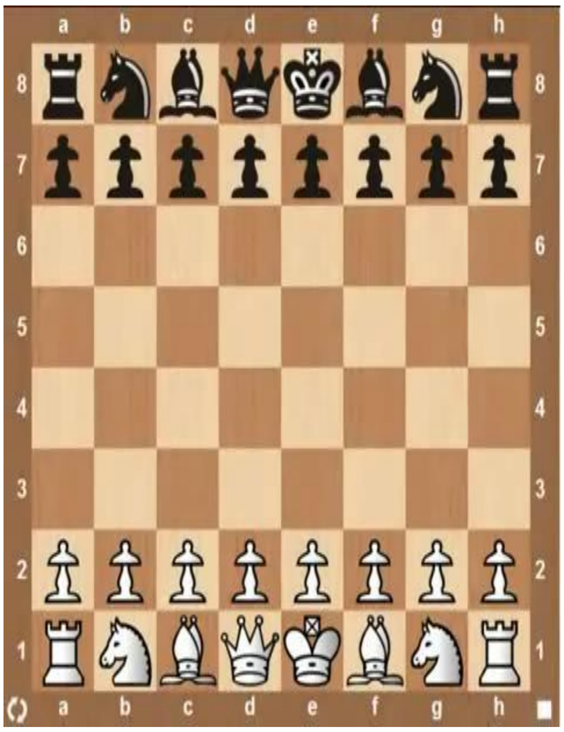


FIGURE: WRONG DETECTION OF BORDER BECAUSE OF LOT OF NOISE AT THE EDGES
FIGURE: VERY SMALL MARGIN WHICH LEADS TO THE LOSS OF PRECESSION



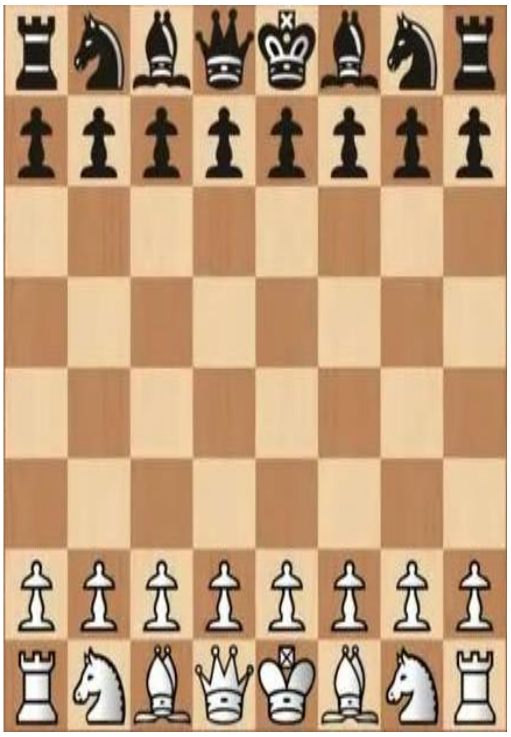


FIGURE: FIRST FRAME AND THE IMAGE AFTER REMOVAL OF BORDER

DETERMINATION OF COLOUR OF THE INDIVIDUAL SQUARES

Initially in the first frame extracted, we have all the four rows in the centre are empty. In this step histogram of the gray scale image is used for the prediction. The histogram gives us the number of pixels that correspond to each of the 256 colours.

In a chess board we know that no adjacent squares have the same colour. After the border removal we will have the coordinates of each individual squares. So in particular we know that the square corresponding to E5 position is white and in the similar way the square corresponding to the E4 position in the board is black.

The histogram of the white square is acquired. The intensity value corresponding to the maximum number of pixels is selected to be colour of the particular square. Similar procedure is carried out with an adjacent square. Now we have the two colours corresponding to two squares.

ALGORITHM

*Step1: Extract the square corresponding to e5 position.

*Step2: Get the histogram of the square.

*Step3: Find the global maxima of the square. This particular intensity value belongs to

white-square.

In the same way as above colour of the black square is obtained by checking in e4 colour.

ISSUES FACED

* In some videos the colours of the white and black squares were altered because of the errors in the co-ordinates predicted in the previous step. So, more importance was given for border removal.

*Due to noise present in the image, there was a broad spread of the histogram. So identifying the perfect intensity value of the square was crucial.

DESCRIPTION THROUGH IMAGES

FIGURE: WHITE AND BLACK SQUARES

It is the first frame from which the colour of the background is acquired. One of the shaded squares in the board are selected in the board and the histogram of the square is taken to determine the required colour intensity values.

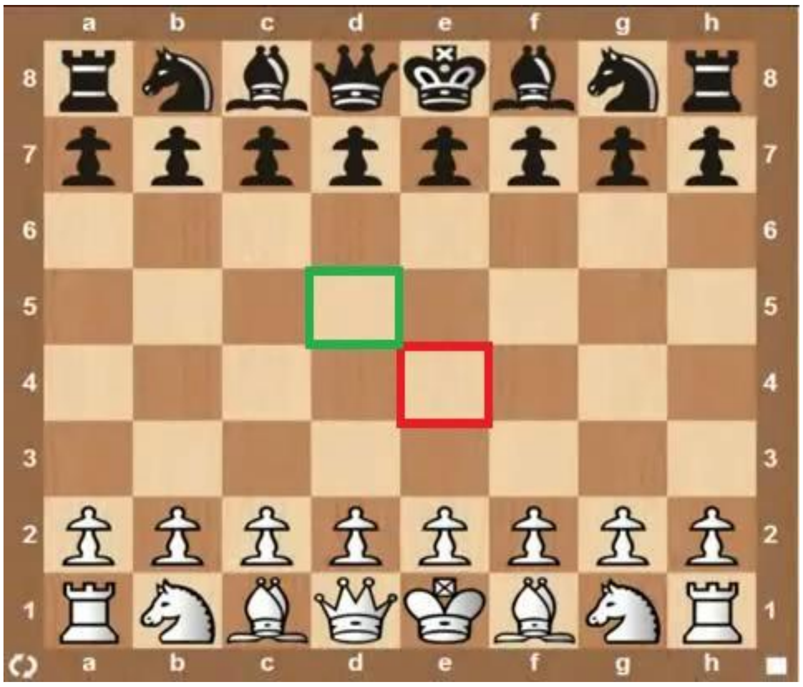
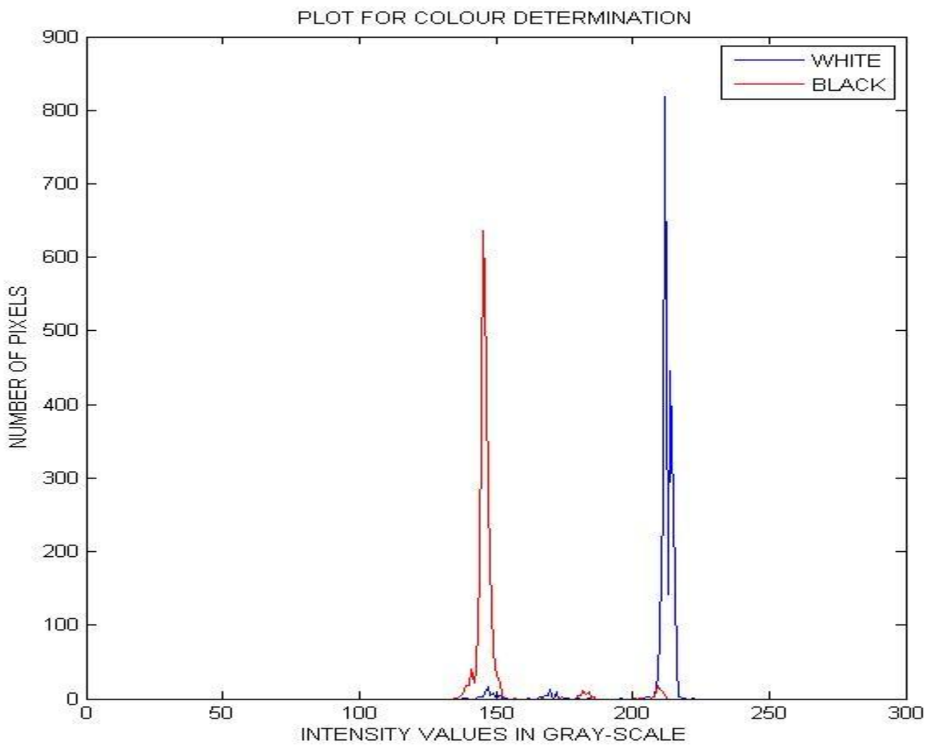


FIGURE: FINDING THE COLOUR F THE BLACK AND WHITE SQUARES



DETERMINATION OF COLOUR OF THE INDIVIDUAL PIECE

We already know the colour of the black and white squares in the chess board. So this task becomes easy now. The colour histogram interpretation is used here for analysis.

In the first frame obtained earlier, we know the position of each piece in the board. In fact, we know that in both the corners ROOK is present. So these particulars squares were separated from the entire image and analysed. The histogram is obtained for both the squares separated. Since we knew the intensities of the back ground squares, the number of pixels corresponding to those intensity values are suppressed(Put them to zero). Later, maxima is obtained and it corresponds to the colour of the piece.

In the actual manner, there should be player playing with the white pieces present in the bottom of the board and latter at the top. In this step the position of the board whether it is proper or flipped is checked based on the intensity values of the pieces.

ALGORITHM

*Step1: Using the co-ordinates of the each squares obtained while removing the border, separate the square A1 and A8 as separate images.

*Step2: Obtain the histogram of the particular square.

*Step3: Obviously the first maxima in the histogram array will correspond to the background square of the image. So obtain the second maxima by keeping an optimal threshold. The colour of the pieces is obtained like this.

*Step4: Later, the values of both the pieces is checked. The maximum of the two is assumed to be the colour of white piece and the other colour is black assumed to be black.

ISSUES FACED

* Some of the videos had pieces with shading throughout the body. In such cases determination of the exact intensity was a tough task. So average was taken among the values within the threshold of the piece colour.

*The white pieces had black coloured outlines which decreased the accuracy of the method.

DESCRIPTION

FIGURE: ROOK PIECES OF BOTH THE COLOURS FROM WHICH THE PIECE COLOUR IS EXTRACTED

DETECTION OF DIFFERENT FRAMES

We have the exact co-ordinates of the chess board. So determining the arrival of different frames which correspond to consecutive moves wasn’t a great task to be done, since we already had sufficient data acquired from various steps.

By keeping the first frame as the reference, we check for the next frame to be saved and analysed. This was done by counting the number of pixels in the two images which had difference in the intensity values.

We had to keep the optimal thresholds to recognise a pixel that a piece has been moved from that particular column to somewhere else. After that, threshold the number of pixels that will designate the change in position of pieces was also the typical task.

After analysing for different videos as well as moves, the optimal value is chosen to be 300. For the next iteration, the selected image in the previous iteration was taken as reference. Thus a set of frames were collected and stored in an array.

ALGORITHM

*Step1: Keep first frame as the reference image. Check for the upcoming frames with the criteria as defined above. That is check the corresponding pixels in both the images if the intensity has been changed drastically.

*Step2: If a new frame with significant number of pixels being changed then store this image in the array allocated earlier.

*Step3: Keeping the image stored as reference check for the next image.

*Step4: Continue the procedure for all the frames, by skipping four frames after every iteration.

ISSUES FACED:

* Some of the frames had arrow marks denoting the attack and movements of the pieces from one place to another. These frames were selected to be the distinct moves in the game, which gave wrong results while interpretation.

*Some of the videos had games played so fast that, there was every possibility that the algorithm could miss some of the frames required. The rate at which the frames are analysed had to be so precise to overcome the difficulties.

FRAMES IDENTIFIED ACCURATELY AND STORED IN YHE ARRAY

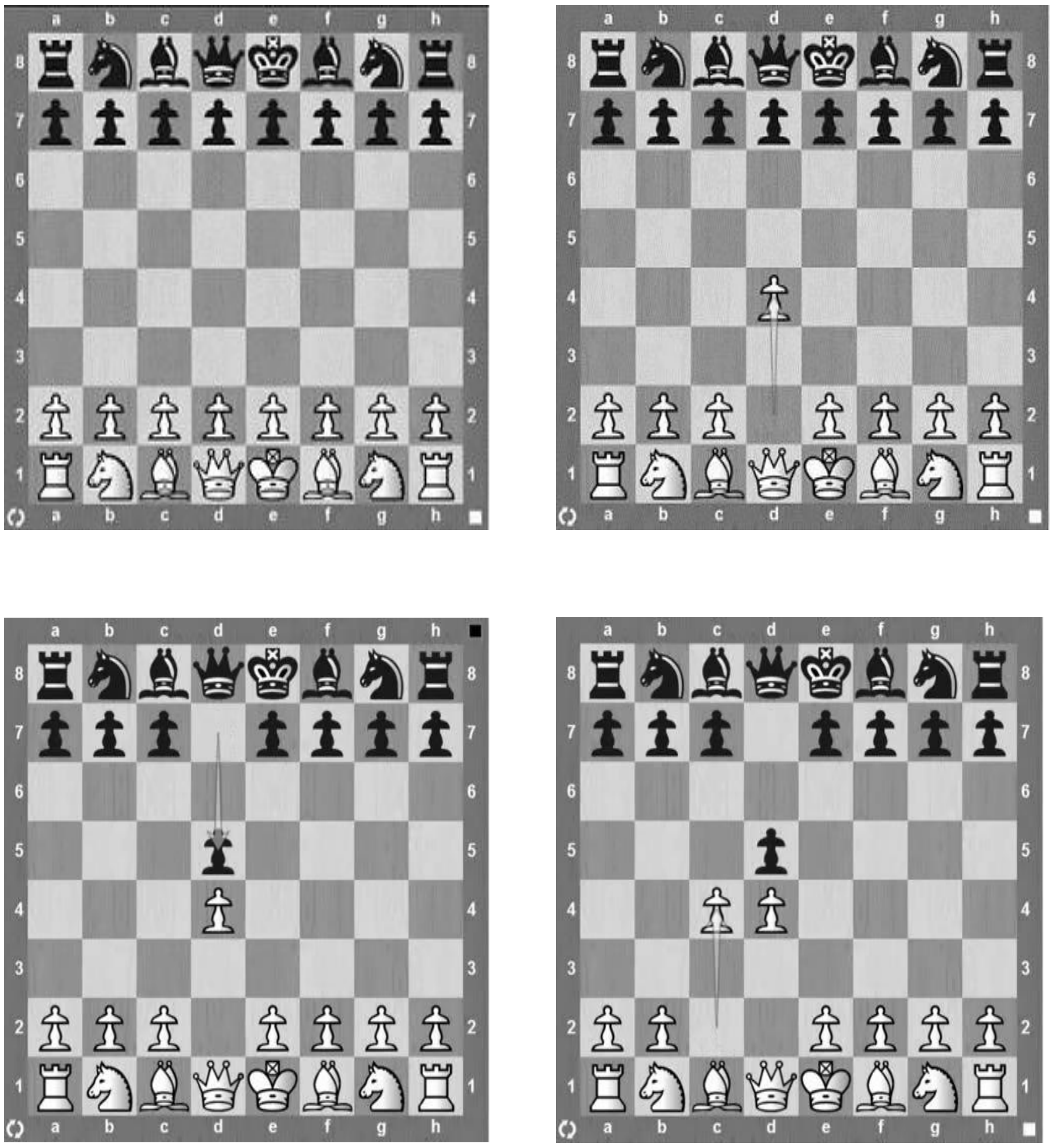


FIGURE: (CLOCKWISE) DIFFERENT MOVES IN THE GAME STARTING FROM THE FIRST FRAME.

WRONG INTERPRETATION BY THE ALGORITHM

There were frequent frames which had long explanations regarding the moves played. So the system could not differentiate between them, rather it stored them blindly following the criteria for arrival of new pixels.

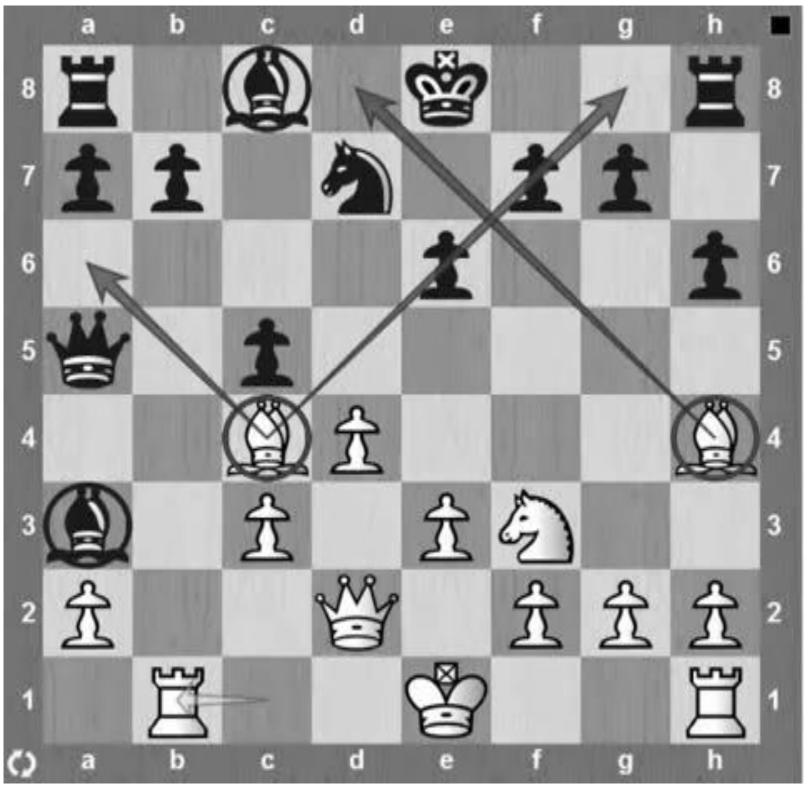


FIGURE: WRONGLY DETECTED TO BE THE DISTINCT MOVE

PIECE RECOGNITION IN A GIVEN CHESS FRAME

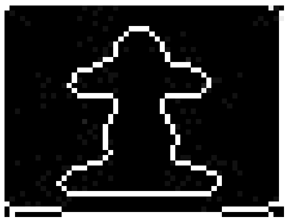
After all the pre-processing steps determining the piece configuration is required to predict the moves played in the game. Basically determining the area and perimeter of the particular piece was decided to be one of the best and efficient method.

In the starting position we know the entire configuration f the chess board. We know which squares are occupied by which pieces. So we applied this rule and obtained the area as well as the perimeter of each of the pieces belonging to black as well as white side.

The data collected in the previous step can be used to just compare and predict the configuration of the board.

Previous to this we had exact co-ordinates of each square. So in the particular square all the operations had to be performed. So the for that particular square extracted, canny filter is applied so that we can determine the exact perimeter could be obtained.

To find the area of the particular piece was a major as well as tough task. We knew the exact colours of the black and white squares as well as pieces. In the required square, we count the number of pixels which correspond to black and white piece colour. By checking the maxima among the two of the pixel count, colour of the piece was determined.


Perimeter of the pawn= 85

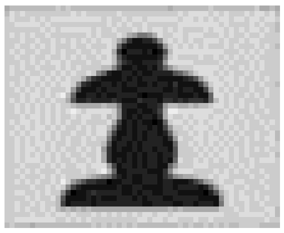

Area of the pawn= 465

FIGURE: AREA AND PERIMETER OF THE PIECE

ISSUES FACED

*Because of the noise and change in the orientation, finding the area of the piece was a hindering task. Many of the background intensity pixels contributed to the piece colour and and hence the area.

*While determining the perimeter of the piece, the border of the square came into the picture, and led the wrong results.

* There was a large gradient of intensity within the square in the chessboard. So the empty square was also recognised to be occupied with piece.

METHODS USED

Filtering:

The square extracted was filtered initially before calculating the area and perimeter of the piece. That is done by using the appropriate threshold. After the process the entire image had only four intensity values.

The method was to find the manhatten distance between the intensity values of the given pixel and all the four colours identified previously, that is namely black piece, black square, white piece and white square. The colour for which least distance was obtained is assigned to the particular pixel. In this way the entire image had a histogram corresponding to only four colours.

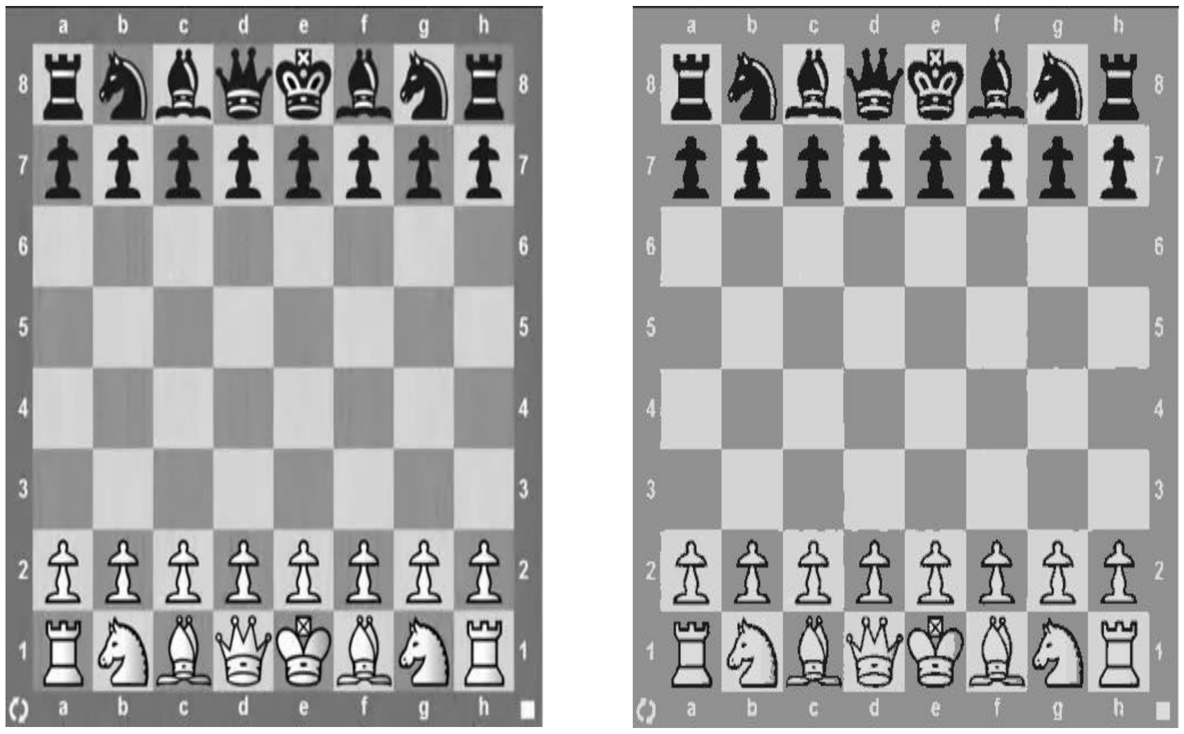


FIGURE: ACTUAL IMAGE AND FILTERED IMAGE

REGION OF INTEREST:

The area within the square divided earlier had border of the square along with it. So determining the area of interest to work or analyse was one of the crucial step. So the algorithm as used in the detection of the border of the chess board is used.

Initially the entire image is passed through canny filter. And later its been divided into required squares. Within the square, sum of the pixels is obtained along each column. We know that along the borders we would get a strong edge and hence a maximum point is obtained.

In the similar way, the analysis is done along the rows by inverting the image through 90 degrees. This step will give us the exact co-ordinates of the edges within which the perimeter has to be calculated.

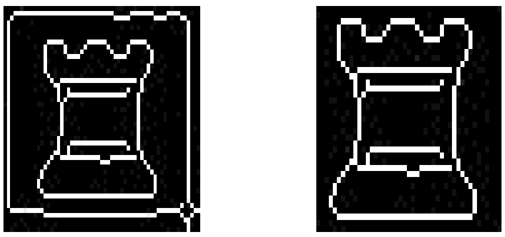


FIGURE: IMAGE OF PIECE AFTER GETTING THE PROPER REGION OF INTEREST

REGION GROWING:

In the image of a particular square this method is applied. Starting from the middle pixel this algorithm works. We determine the gradient image of the particular square. Along with that we need to calculate the angle produced by the two gradients in both the directions.

The gradient is produced using the equations as given below.

$$G_x = | f(x,y) - f(x-1,y) |$$

$$G_y = | f(x,y) - f(x,y-1) |$$

Th e angle between them is given by,

$$\theta = \tan^{-1}(G_y / G_x)$$

So, we form a region comprising of equal values of the gradient as well as the angle within a pre-defined threshold. So this will obviously increase the number of pixels corresponding to the piece colour. This operation is performed through out the image starting from the central pixel.

TRACKING THE GAME

We used an algorithm that could track the game starting from the beginning. Prior to this we have all the frames required to be processed. In each frame we can split it into a set of sixty-four squares. Analysing each square in the consecutive frames we can judge if there is any movement in the piece.

We know the initial position and hence the presence of pieces within the board. Each square present in the entire board can be analysed separately. Using this procedure each square is analysed if there is any change from the previous frame.

First checking is done if there is any new arrival of piece in any square. Then all the squares are checked for absence of piece. And thus the piece which is absent is noted and presented to be moved to the square predicted earlier.

The same procedure is followed till the end by keeping each frame as reference in each step. During every iteration, an expression is produced using the names of the pieces as well as the moves performed. So the acquired expression is stored as an array. Finally the array is stored as *PGN* in the text format using the system calls.

ALGORITHM

*Step1: Keep first frame as the reference.

*Step2: Compare each square in the next frame with the previous frame. Check from which square a piece has moved, Store the string corresponding to the particular square.

*Step3: Now see for the square in which a new piece as arrived and it was inherently absent in the previous frame. Now assign the string stored in the previous step to that particular position of the chess-board.

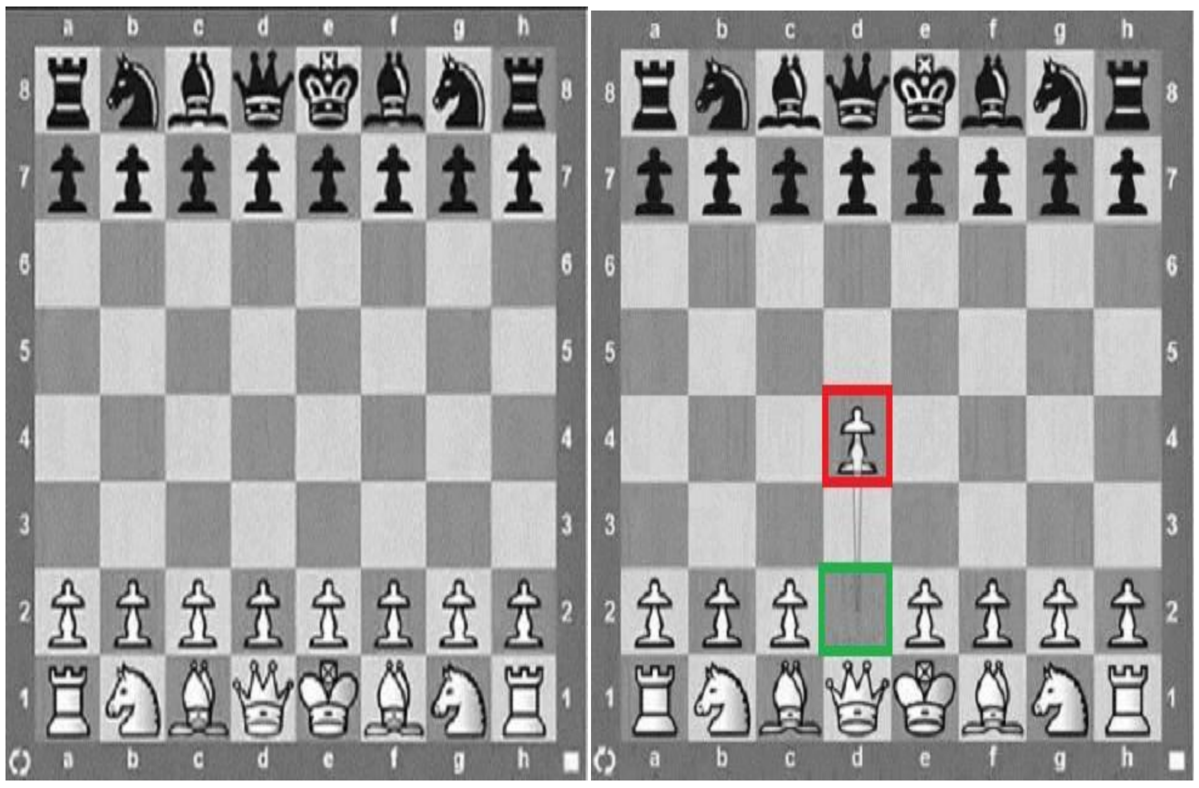
*Step4: Keep the present frame as the reference. Analyse the next frame.

*Step5: Continue the procedure for all the frames.

```
00000000
00000000
00000000
00000000
00010000
00000000
00010000
00000000
```

```
00000000
00010000
00000000
00010000
00000000
00000000
00000000
00000000
```

FIGURE: CHANGE DETECTION MATRICES FOR THE MOVES SHOWN BELOW



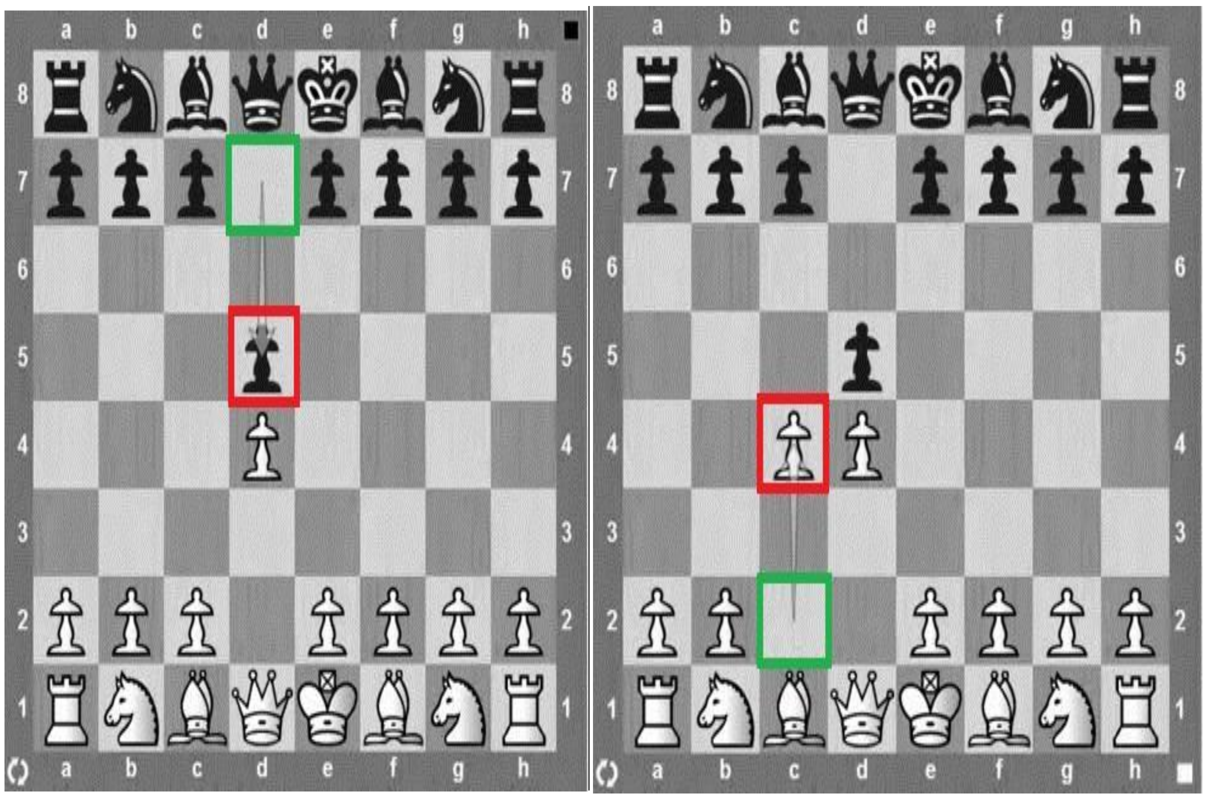


FIGURE: DETECTING THE MOVES IN THE CONSECUTIVE FRAMES

WRITING THE PGN FILE

We know that in every frame either a piece is removed or displaced with respected to the previous frame. So using this principle the piece which enters an unoccupied square is the required piece. We now have the name of the piece which is been moved.

We name each of the squares present in the entire chess board using the standard notation i.e each of the square is denoted by two specifications. First one is an alphabet followed by a number. Typically we require 8 alphabets and numbers to designate all the squares. For example a8, b5, h3 etc.

After having the name and position of the piece which has been moved from and to a square, we can write the PGN file using the adequate rules and notations prescribed in writing it as mentioned in the introduction chapters.