Feature Description of ADDI Alzheimer’s Detection Challenge dataset: Description, Methodology and Examples

CONTENTS

Input Data Format

There are two outputs that are generated from the object detection model.

The first object detection code detects they are of interest in the image (i.e., clock in this case) and subsequently crops the original image based on these coordinates.

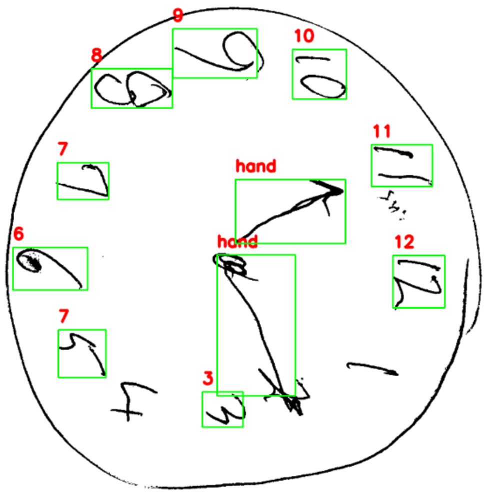
The second object detection code detects the different digits and the hands present in the cropped image. The output is an excel file consisting of the coordinates of each detected digit and the detected hands. This output file goes as an input into the Feature Extraction Code.

Clock Features

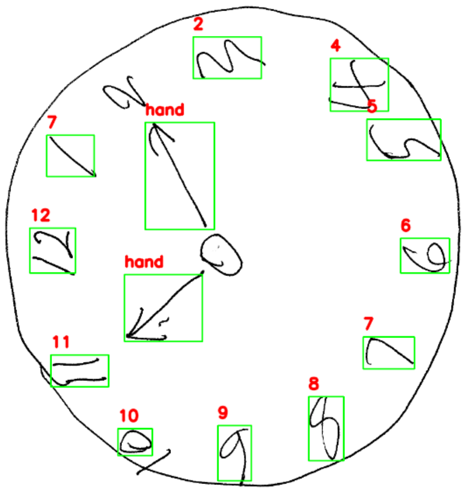
Final Rotation Angle

- **Variable:**
    - *final_rotation_angle* **--** This variable calculates the angle the image has to be rotated to make the clock straight.

- **Methodology**: Checking the positions of 3, 6, 9 and 12 and then calculating the angle the clock needs to be rotated to get 12 on top.
- **Prerequisites:** The raw annotated output is required.
- **Example:** The following two images with the id ***10000019_R8.tif*** *and* ***10000008_R7.tif*** shows clocks that are not aligned. The first clock need to be rotated by 90 degrees counter clockwise to make it straight and the second clock need to be rotated by 270 degrees counter clockwise to get 12 on top and make the clock straight.



10000019_R8.tif



10000008_R7.tif

Digit Features

Number of Digits

- **Variable:** *number_of_digits*

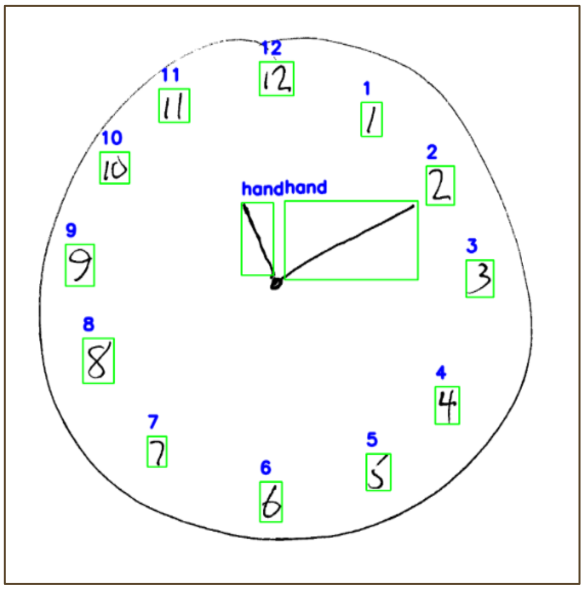
- **Description:** This variable gives the number of digits detected for a hand drawn clock image. The numbers may vary as some patients can draw one number more than once, or some numbers may not get detected. It may also happen that some numbers are detected wrong. For e.g. digit 5 is detected as digit 3, in such cases the detection confidence is considerably low (~60%-75%)

- **Methodology:** The raw annotated output is used to calculate this feature. However, before this feature is calculated, the data is first cleaned of repeated digits detected whose confidence percentage is less 90%, based on the criteria that any proper detection needs to have at least 90% confidence. For digits that are not repeated, confidence percentage less than 90% are also included.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:** The image with id ***10000056_R2.tif*** has 12 digits detected.

Figure 1: The following image has 12 digits detected



Missing Digit Dummy Variable

- **Variables:** There are twelve variables (each a dummy variable for each of the twelve digits). For each variable 0 means the digit is present, whereas 1 means it is not. The variables are:
    - *missing_digit_1 – this variable capture if the digit 1 is present or not.*
    - *missing_digit_2 – this variable capture if the digit 2 is present or not.*
    - *missing_digit_3 – this variable capture if the digit 3 is present or not.*
    - *missing_digit_4 – this variable capture if the digit 4 is present or not.*
    - *missing_digit_5 – this variable capture if the digit 5 is present or not.*
    - *missing_digit_6 – this variable capture if the digit 6 is present or not.*
    - *missing_digit_7 – this variable capture if the digit 7 is present or not.*
    - *missing_digit_8 – this variable capture if the digit 8 is present or not.*
    - *missing_digit_9 – this variable capture if the digit 9 is present or not.*
    - *missing_digit_10 – this variable capture if the digit 10 is present or not.*
    - *missing_digit_11 – this variable capture if the digit 11 is present or not.*
    - *missing_digit_12 – this variable capture if the digit 12 is present or not.*

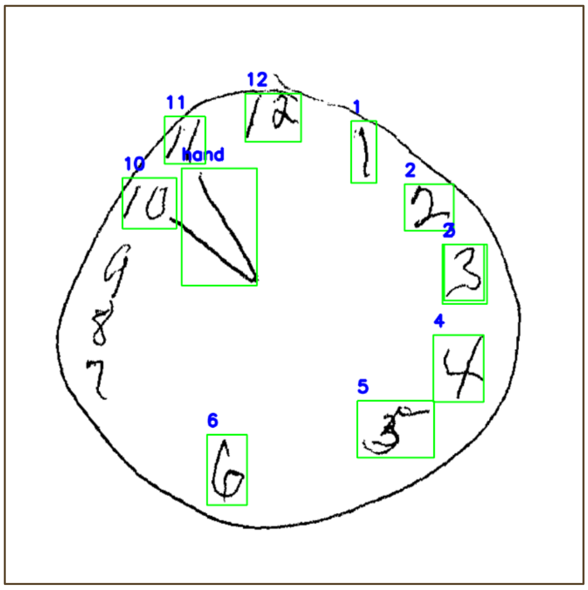
- **Description:** These variables gives the information about the digits that are present.

- **Methodology:** After the digit and hands coordinates output is generated, the result is checked for duplicate values. If there are duplicate values for a particular digit, then all those instances where the confidence of detection is less than 90%, they are removed, assuming that they have been detected wrong, while the rest are kept in the dataset.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:** The following image (image id – ***10000056_R2.tif***) shows that none of the digits have either not been detected, or they have been detected twice. Hence, the missing digit dummy for all the variables is 0.

Figure 2: For image 10003861_R1.tif, digits 7,8, and 9 were not detected



Deviation of Axis Digits (3, 6, 9 and 12) from Mid Axes

- **Variable:** *deviation_dist_from_mid_axis*

- **Description:** This variable gives the average deviation (in terms of Euclidean Distance) of the axis digits from the mid axes (i.e., vertical and horizontal line passing through 512, 512). This gives us an idea that if more than two digits are present in a quadrant, and if an axis digit is present then how far that axis digit is from its ideal position. For the time being we have average, we can also include standard deviation of distance, mean distance, etc.

- **Methodology:** The distance from the center of an axis digit's bounding box to the mid axes is calculated.

  - For digits 3 and 9, this is the difference between 512 and the y coordinate of the digit box center.
  - For digits 6 and 12, this is the difference between 512 and the x coordinate of the digit box center

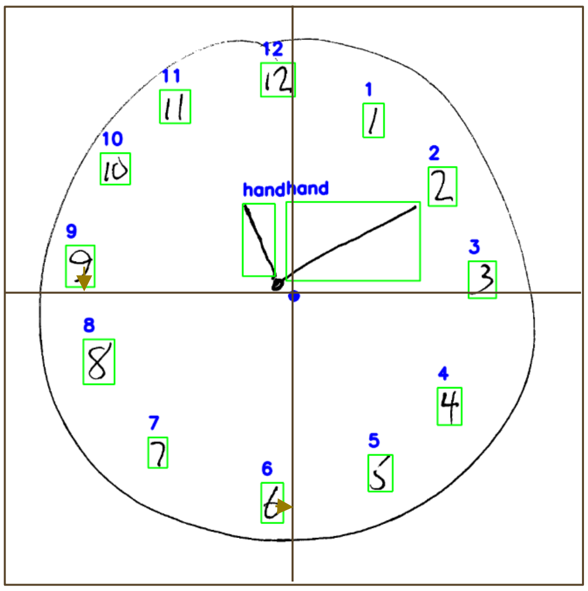
  Then the average of these distances is calculated.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:**

Figure 3: Deviation of the axis digits from the mid axes

This arrow shows the distance of deviation



Between Axis Digits Angle Metrics

- **Variable:** Based on the angle between two digits algorithm, the angle between axis digits was calculated to see how much disoriented the axis digits are relative to each other. However, since this output is at an image-digit level, the output was rolled using sum and variance. The two variables created are:
  - *between_axis_digits_angle_sum* – the sum of the angles between the axis digits.
  - *between_axis_digits_angle_var* – the variance of the angles between the axis digits,

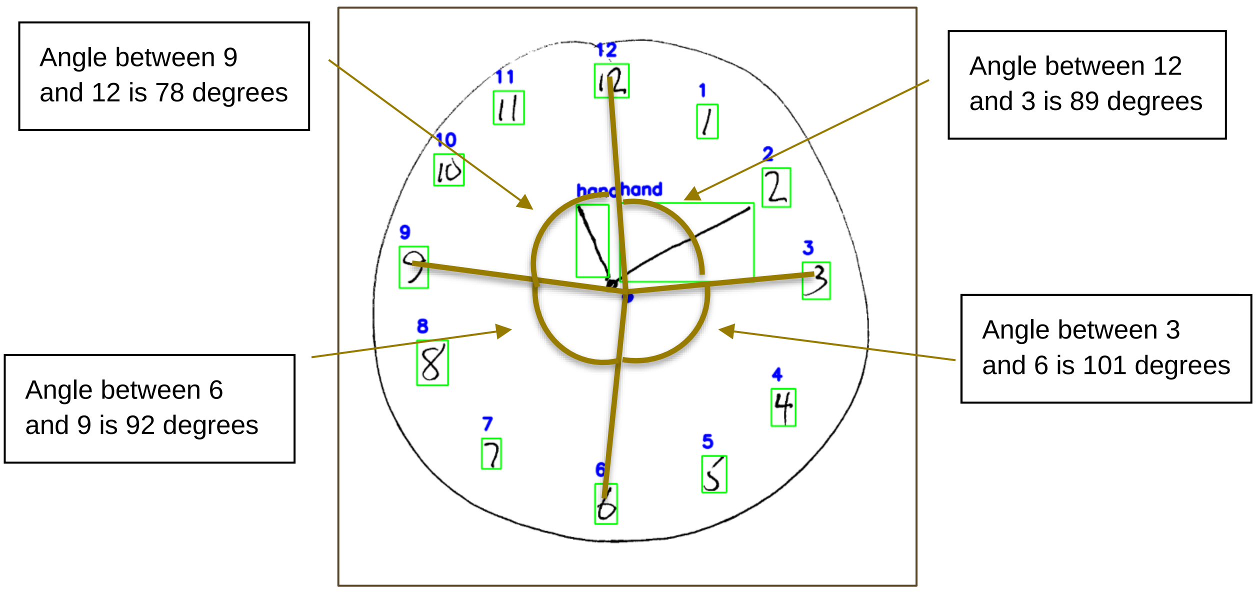
- **Description:** In a proper image or in an image where the axis digits have been detected correctly, the sum between the angles should be equal to 360. The variance in turn gives an idea about the difference in the angles. Lower this value, better is the relative position of the axis digits to each other.

- **Methodology:** Angle between digits algorithm was used with the detected digits being 3, 6, 9 and 12.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:**

Figure4: Angle Between Axis Digits


Angle between 9 and 12 is 78 degrees
Angle between 12 and 3 is 89 degrees
Angle between 6 and 9 is 92 degrees
Angle between 3 and 6 is 101 degrees
12
11
1
10
2
hand
hand
9
3
8
4
7
5
6

Between Digits Angle Metrics

- **Variable:** There are four variables that have been calculated that gives an idea about how much equidistant the detected digits are from one another in the image. The four variables are:
  - *between_digits_angle_cw_sum* – the sum of the angles between the detected digits, going in clockwise direction
  - *between_digits_angle_cw_var* – the variance of the angles between the detected digits, going in clockwise direction
  - *between_digits_angle_ccw_sum* – the sum of the angles between the detected digits, going in counter-clockwise direction
  - *between_digits_angle_ccw_var* – the variance of the angles between the detected digits, going in counter-clockwise direction

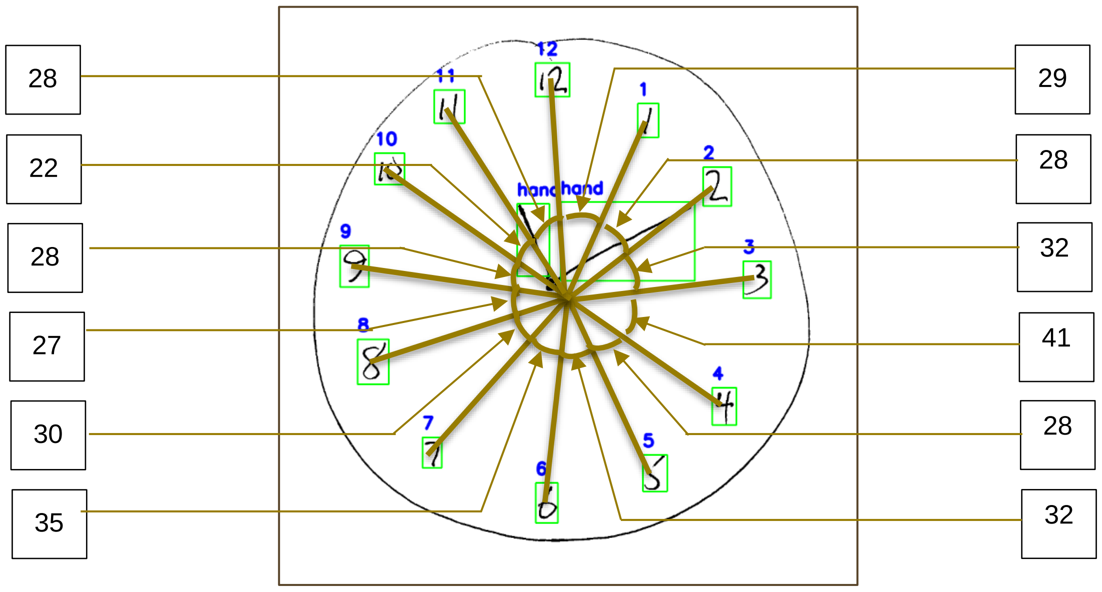
- **Description:** In a proper image or in an image where the axis digits have been detected correctly, the sum between the angles should be equal to 360. The variance in turn gives an idea about the difference in the angles. Lower this value, better is the relative position of the digits to each other. The sum of the angles will be helpful in determining the sequence of the digits, which has been explained later.

- **Methodology:** From the digit and hand annotated raw output, the digit data is filtered for an image. This data is then sorted on the basis of the digit column. Then the center of the first and second digit is calculated. The angle between the straight lines formed by the geometric center of the image and the center of the digits respectively, gives the angle between the two digits. The direction of the angle is from the first digit to the second in a clockwise direction. This step is repeated successively for the rest of the digits. Once, this output is generated, this table is rolled up to image level using sum and variance, for reasons explained above.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:**

Figure 5: Angle Between the Digits Detected in Degrees



Sequence Flag Clock Wise and Counter Clock Wise

- **Variable:** There are two variables to capture the sequence of the digits.
  - *sequence_flag_cw* – a dummy variable denotes whether the digits are in clockwise sequence or not. 0 means the digits are not in sequence, but 1 means they are.
  - *sequence_flag_ccw* – a dummy variable capturing whether the digits are in counter clockwise sequence or not.

- **Description:** These variables denote whether in an image the detected digits are in clockwise or counter clockwise sequence. They do not tell anything about the position of the digit.

- **Methodology:** After the angle between digits have calculated, the sum of all these angles is calculated. If the sum is equal to 360, then the digits are in sequence, else they are not.

- **Prerequisites:** The digit and hand annotated output

Figure 6: The detected digits are in clockwise sequence

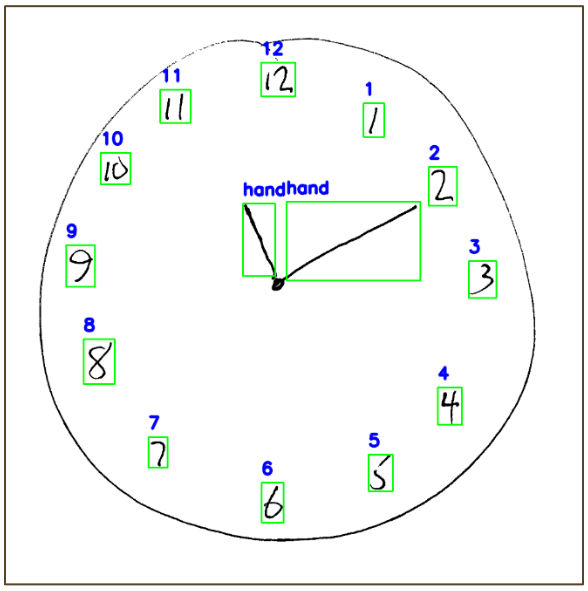


Figure 7: The detected digits are in anticlockwise sequence

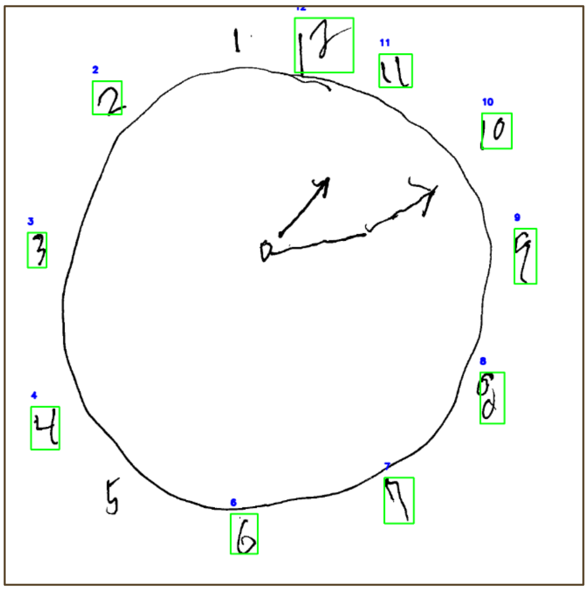
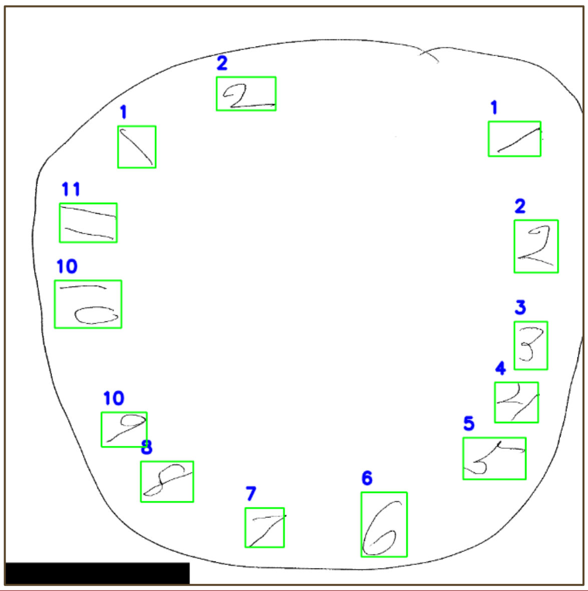


Figure 8: The numbers 1 and 2 are not in sequence



Hand Features

Number of Hands

- **Variables:**
  - *number_of_hands -- Gives us the number of hands detected*
  - *hand_count_dummy*

- **Description:** This variable gives the number of hands detected for a hand drawn clock image. The numbers may vary as some patients can draw no hands at all, one hand, more than two hands, or one of the hands may not get detected. It may also happen that some hands are detected wrong. For e.g. digit 7 is sometimes detected as a hand, in such cases the detection confidence is considerably low (~60%-75%)

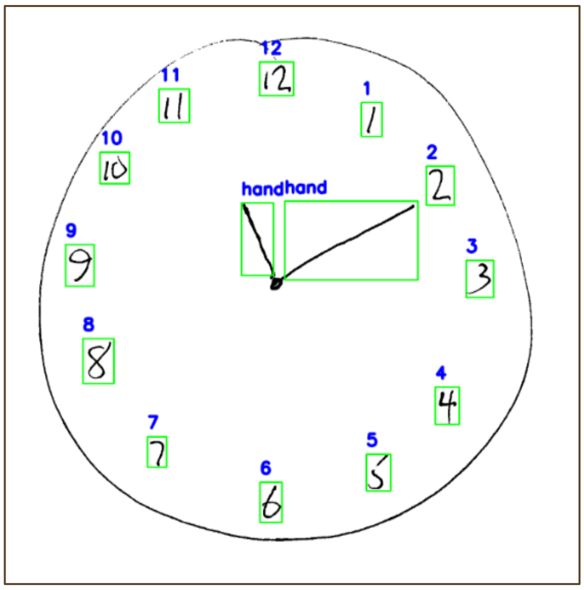
  Variable 'hand_count_dummy' takes values of 1 to 3, where 1 is assigned when a single hand is detected, 2 when two hands are detected and 3 when more than two hands are detected.

- **Methodology:** The raw annotated output is used to calculate this feature. However, before this feature is calculated, the data is first cleaned of repeated hands, if the count is more than 2, detected whose confidence percentage is less 90%, based on the criteria that any proper detection needs to have at least 90% confidence. For images where one hand has been detected, confidence percentage less than 90% are also included.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:** The image with id ***10000056_R2.tif*** has 2 hands detected.

Figure 9: The following image has two hands detected



Hand Length

- **Variables:** Five variables are created based on the number of hands detected. If the number of detected hands are either one or two, then the following variables are calculated:
    - *hour_hand_length* – if there are two hadns detected, then the one with the smaller length (i.e., the length of the diagonal of the bounding box of the hand) is termed to be the hour hand.
    - *Minutehand_length* - if there are two hadns detected, then the one with the longer length (i.e., the length of the diagonal of the bounding box of the hand) is termed to be the minute hand.
    - *Single_hand_length* – if only one hand has been detected, then the it is termed as single hand and its corresponding length falls under single hand length.
    - *Clock_hand_ratio* – the ratio of the minute hand length to the hour hand length. For single hands, this variable's value is null.
    - *Clock_hand_diff* – the difference between the minute hand length and the hour hand length. For single hand length, this value is null, as well for images where the number of hands is more than two or no no hands have been detected.

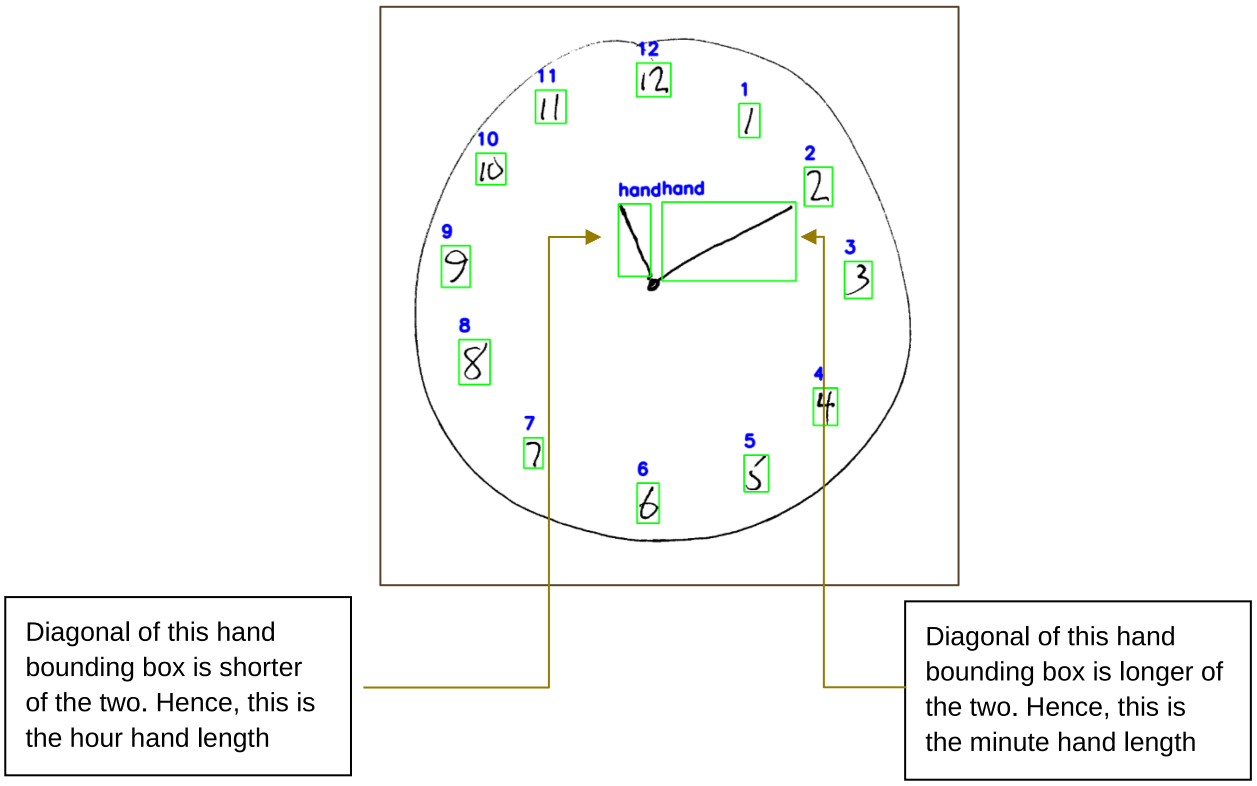
- **Description:** These variables help to identify the hands in the image. Also, a separate set of variables can be derived from these, which have been described later.

- **Methodology:** Given the digit detection raw annotated output, the length of the diagonal of the bounding boxes of the hands give the length of the hand. These features has been calculated if the number of detected are either one or two. For the rest, the value for these features is null.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:**

Figure 10: Hand Identification based on length of the hands



Diagonal of this hand bounding box is shorter of the two. Hence, this is the hour hand length

Diagonal of this hand bounding box is longer of the two. Hence, this is the minute hand length

Angle Between Hands

- **Variable:** *angle_between_hands*

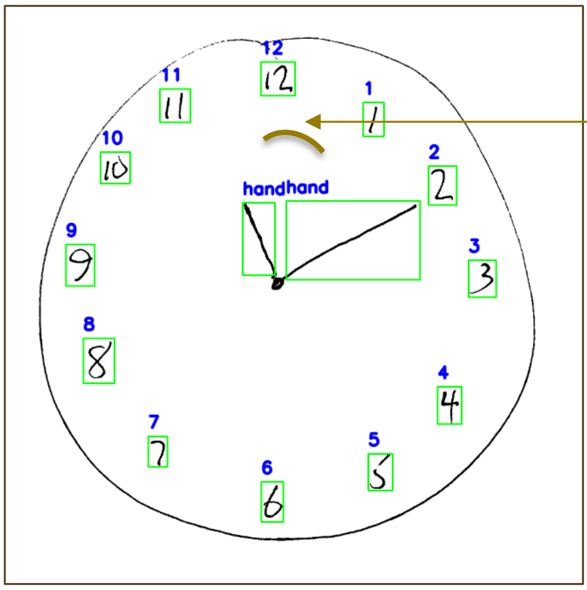
- **Description:** This variable calculates the angle between the hour hand and the minute hand. The direction of the angle is from the hour hand to the minute hand in a clock wise direction. The value for this feature if null if the number pf hands detected is not equal to 2.

- **Methodology:** The proper diagonal of the hour hand bounding box is identified, which is actually the hour hand, using the minimum of the sum of the absolute of the difference of each vertex from the center (512, 512). The vertex closest to the center and its diagonally opposite vertex forms the two coordinates of the hour hand. Similarly, for minute hand the correct diagonal vertices are found. Thus, there are two straight lines whose intersection angle is calculated. In an ideal clock, this angle should be equal to 90 degrees for 11:10.

- **Prerequisites:** The digit and hand annotated output

- **Example:**

Figure 11: The angle between the hands for 10000056_R2.tif



The angle from the hour hand to the minute hand in a clockwise direction is 86 degrees.

Deviation of Intersection Point of Hands from Geometric Centre

- **Variable:** There are two variables that have been calculated to get an idea about how far the intersection point of the hands is away from the geometric center and in which direction with respect to the center. The variables are:
  - *deviation_from_center* – this variable calculates the Euclidean distance between the intersection point of the hands and the geometric center. Higher the value, far away the intersection point is from the center.
  - *Intersection_pos_rel_center* – this variable gives a general direction of the intersection point relative to the geometric center. The values can be one of the four – top Left, Top Right, Bottom Left and Bottom Right.

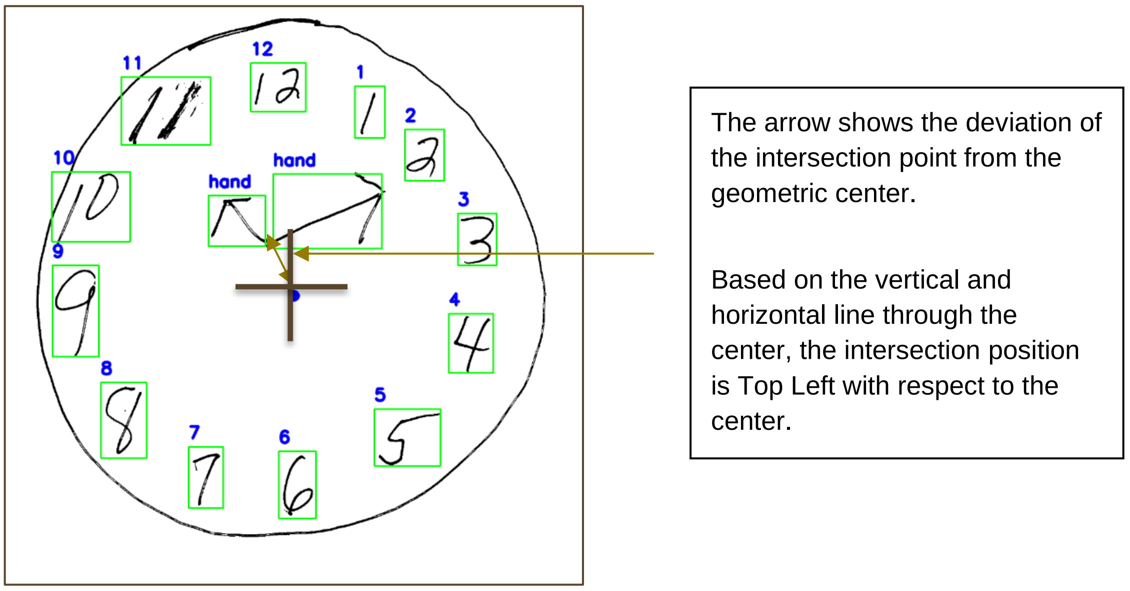
- **Description:** These variables give an important idea about how far the hands are from the center. As a result of this, the angle between the hands can get affected. This feature is available when both the hands are detected. In other cases, this value is null.

- **Methodology:** We identify the proper diagonal of the bounding boxes of the two hands, similar to *angle_between_hands* and then the intersection point is calculated using coordinate geometry formula. Post this, based on the intersection coordinates, the position of the intersection point is derived relative to the center, similar to the determination of the quadrants.

- **Prerequisites:** The digit and hand annotated output.

- **Example:**

Figure 12: Euclidean Distance between intersection point of hands and the geometric center, along with the relative position



11
12
1
2
10
hand
hand
3
9
4
8
5
7
6
The arrow shows the deviation of the intersection point from the geometric center.
Based on the vertical and horizontal line through the center, the intersection position is Top Left with respect to the center.

The Proximity of Hour and Minute from 11 and 2 Respectively

- **Variable:** There are two variables to get how far the hour hand is from the digit 11 and the minute hand is from the digit 2. The variables are:
  - *hour_proximity_from_11* – this variable gives the closeness of the hour hand to the digit 11.
  - *minute_proximity_from_2* – this variable gives the closeness of the minute hand to the digit 2.

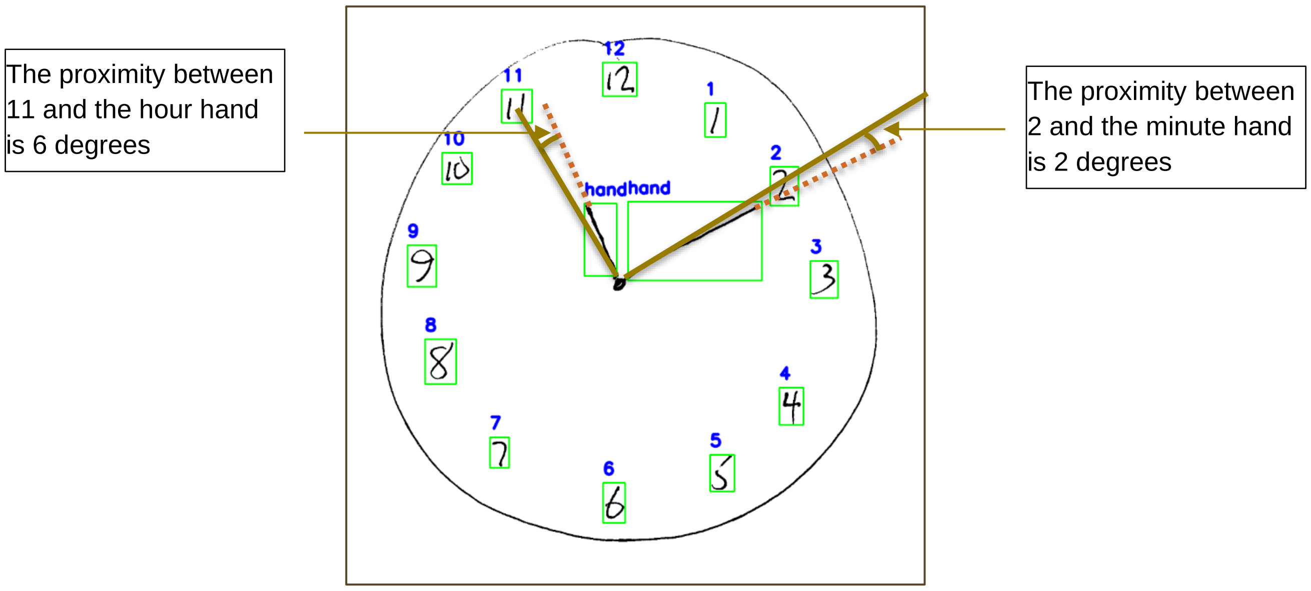
- **Description:** These two variables calculate the proximity of the hour hand to the digit 11 and the minute hand to the digit 2. Each of these variables can be capture if both the digit and the hand are identified. Hence, to get values for both the variables, the digits 11 and 2 have to be identified along with the hour and the minute hand. Other either or both the variables will be null.

- **Methodology:** For the proximity of the hour hand, a reference line is calculated from the center coordinates of the digit 11 and the vertex of the hour hand bounding box that is closest to the geometric center. Then the angle is calculated from the reference line to the hour hand in a clockwise direction. If the angle is more than 180 degrees, then it is subtracted from 360 degrees. The final angle lies between [0,180] degrees. Similar for the minute hand proximity.

- **Prerequisites:** The digit and hand annotated output

- **Example:**

Figure 13: The proximity of hour and minute hand from 11 and 2 respectively


The proximity between 11 and the hour hand is 6 degrees
The proximity between 2 and the minute hand is 2 degrees
12
11
1
10
2
hand
hand
9
3
8
4
7
5
6

Digit Pointed by Hour and Minute Hand

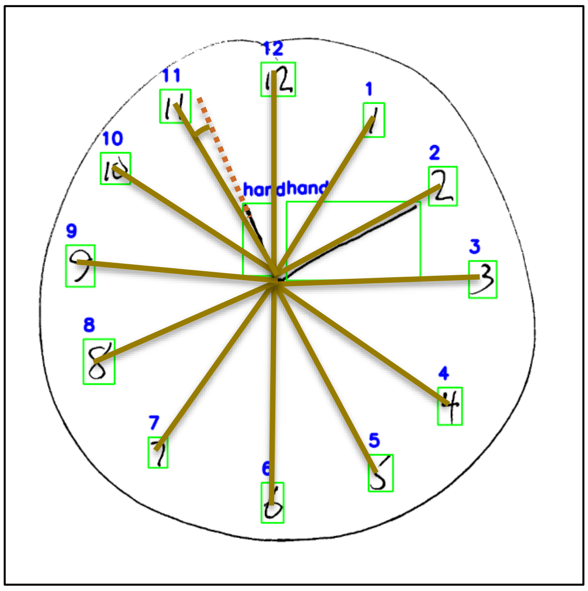
- **Variable:** There are two variables to get which digits are being pointed by the hour hand and the minute hand. The variables are:
  - *hour_pointing_digit* – this variable gives the digit that is pointed by the hour hand
  - *minute_pointing_digit* – this variable gives the digit pointed by the minute hand.
-
- **Description:** This variable calculates the time that the user has drawn. Some clock hand errors can be derived from these two features that have been described later.

- **Methodology:** Using the proximity algorithm, the angle between the hour hand and the reference line for all the detected digits is calculated. Then whichever corresponds to the minimum angle is taken to be the digit pointed by the hour or the minute hand.

- **Prerequisites:** The digit and hand annotated output

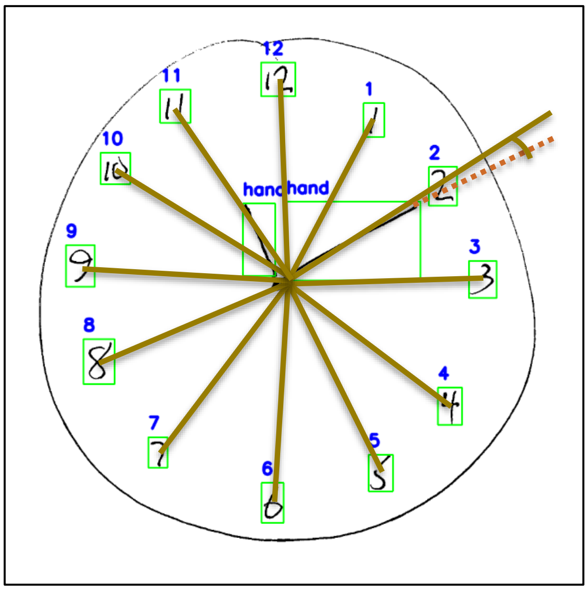
- **Example:**

Figure 14: The digit pointed by the hour hand is 11



The angle between the reference line of 11 and the hour hand is smallest. Hence, the hour hand points to the digit 11

Figure 15: The digit pointed by the minute hand is 2



The angle between the reference line of 2 and the minute hand is smallest. Hence, the minute hand points to the digit 2

Clock Hand Errors

- **Variable:** There are couple of features (dummy variables) that capture the different errors of the clock hands. They are:
  - *eleven_ten_error* – this dummy variable captures the error when the hour hand points to 11 or 10 and the minute hand also does the same.) means there is no eleven ten error, whereas 1 means there is eleven ten error.
  - *other_error* – this dummy variable captures any other form of error of the hands. This variable is 0 if the digits pointed by the hour and the minute hand are 11 and 2 respectively, else it is 1.

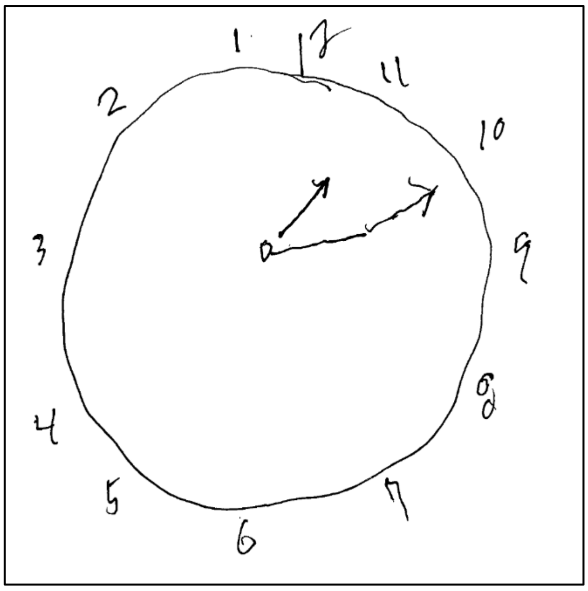
- **Description:** These variables help to determine how strong the correlation is between the class of an image and the placement of the hands in that image.

- **Methodology:** Simple if else conditions are used based on the hour hand pointing digit and the minute hand pointing digit.

- **Prerequisites:** The digit and hand annotated output

- **Example:**

Figure 16: In the below picture, the hour hand points to 11 whereas the minute hand points to 10. Hence, the value for eleven_ten_error is 1

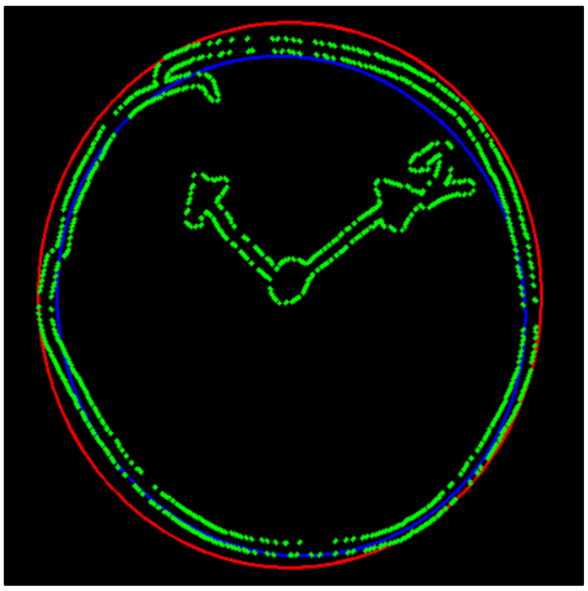


Circle Features

Ellipse to Circle Ratio

- **Variable:** *ellipse_circle_ratio*
- **Description:** This variable tells how circular the drawn clock is. This is achieved by comparing the fitted ellipse (of the contour) with the minimum enclosing circle.
- **Methodology:** An ellipse is fitted along with a minimum enclosing circle (a circle enclosing the clock that has the minimum area). Then the area of the two fiited shapes are calculated and compared as a ratio. In case of poor fitted ellipses, this could result in a situation where the ellipse has a larger area thean the minimum enclosing circle.
- **Prerequisites:** The cropped image is required.
- **Example:** In the below figure, the blue line is the fitted ellipse whereas the red line is the minimum enclosing circle.

Figure 17: The fitted ellipse and minimum enclosing circle



Predicted Tremor and the Number of Defects

- **Variables:** There are two variables that are used to get an idea about the tremors present in a clock drawing. They are:
    - *count_defects* – this variable shows how smooth the drawn circle is in a clock.
    - *pred_tremor* – this variable tries to predict the tremor class based on count_defects.

- **Description:** These variables captures the smoothness of the clock, and in turn tries to highlight the relationship between the cognitive state of a person and his motor skills. For e.g., if a person has Parkinson's, then does it affect his/her cognitive score.

- **Methodology:** It can be assumed that the roundness of the circle can be defined as a set of convex hulls. The more circular the clockface is, the more convex hulls will be present. Convex hulls here are defined as "defects". Therefore, ideally the more convex hulls, the fewer shaky lines are present in such an image.

- **Prerequisites:** The cropped image is required.

- **Example:** In both these images below, the "red-dots" signify the defects present in the image. The image on the left has more of these red dots, as it is smoother, while the image on the right has lesser number of red dots, showing that it is less smooth (shaky).

Figure 18: A smoother circle that has fewer greater number of defects

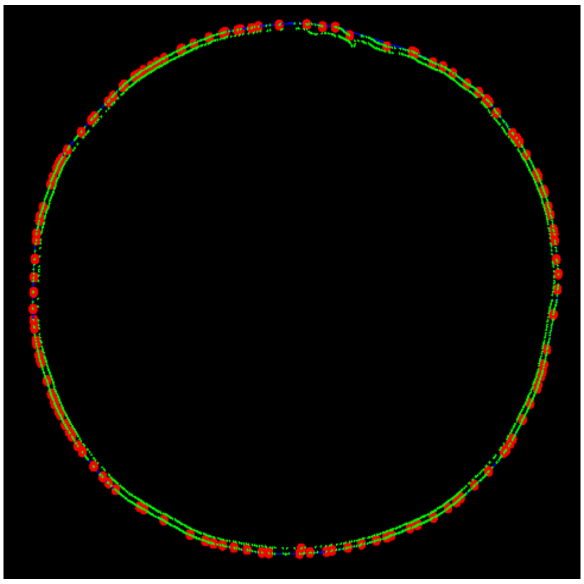
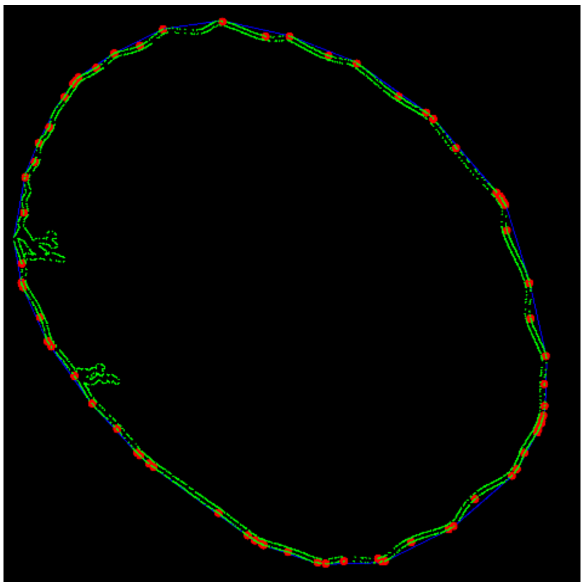


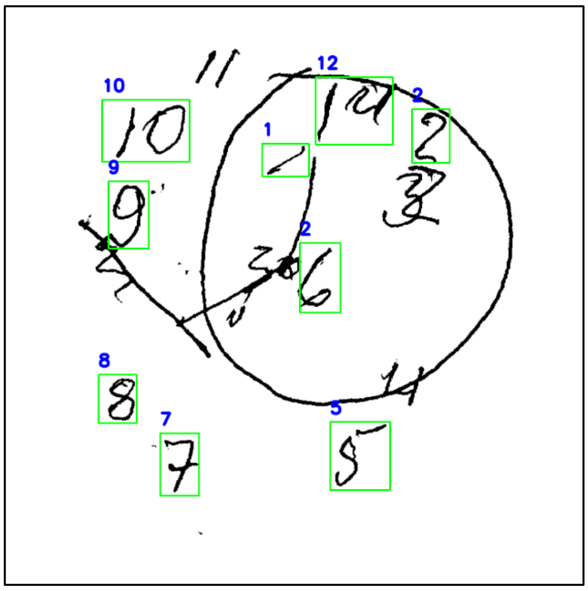
Figure 19: A shaky circle, that has a greater number of defects



Percentage of Digits inside the Clock Face

- **Variable:** *percentage_inside_ellipse*
- **Description:** This variable captures the percentage of digits and hands are inside the clock perimeter.
- **Methodology:**
- **Prerequisites:** The cropped image is required.
- **Example:**

Figure 20: Out of 9 digits detected, 4 are inside the circle perimeter



The Length of the Major and Minor Axis of the Fitted Ellipse

- **Variable:** There are two variables. They are:
  - *double_major* – this variable fives the length of the major axis of the fitted ellipse
  - *double_minor* – this variable fives the length of the minor axis of the fitted ellipse

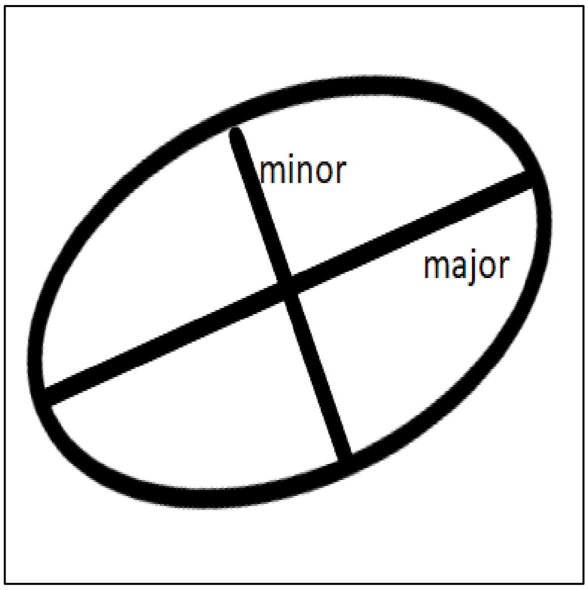
- **Description:**

- **Methodology:** The major and minor is calculated using the opencv package in python.

- **Prerequisites:** The cropped image is required.

- **Example:**

Figure 21: The major axis and minor axis of the fitted ellipse



Area of the Top, Bottom, Left and Right Hemisphere of the Circle

- **Variable:** There are four variables that gives the percentage of area of the four different halves. They are:
    - *top_area* – this variable gives the percentage area of the upper hemisphere.
    - *bottom_area* – this variable gives the percentage area of the lower hemisphere.
    - *left_area* – this variable gives the percentage area of the left hemisphere.
    - *right_area* – this variable gives the percentage area of the right hemisphere.

- **Description:** These four variables capture the roundness of the clock as well.

- **Methodology:** The image is divided into the upper and lower half by the horizontal line passing through the geometric center. Then the area of each half is calculated as percentage to the area of the whole circle. Same is done to divide the image into left and right halves, by a vertical line passing through the geometric center.

- **Prerequisites:** The cropped image is required.

- **Example:**

Figure 22: The upper hemisphere of the clock image

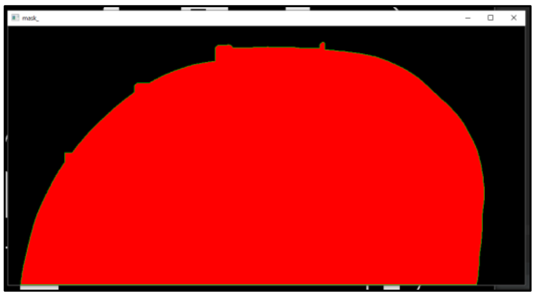


Figure 23: The lower hemisphere of the clock image

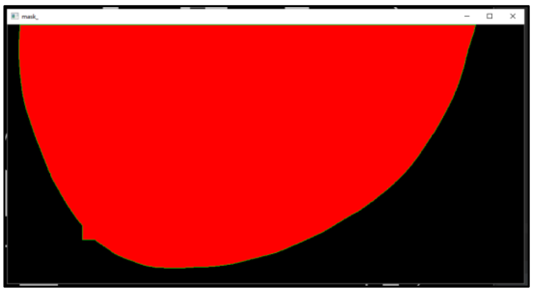


Figure 24: The left hemisphere of the clock image

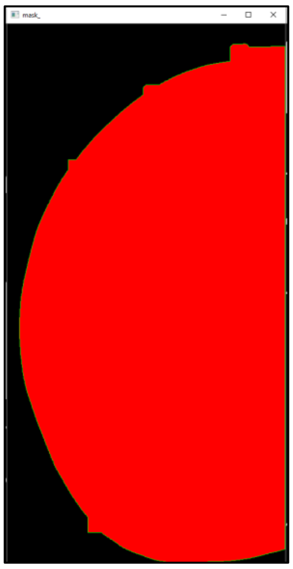
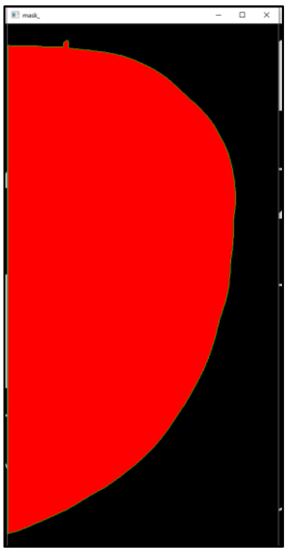


Figure 25: The right hemisphere of the clock image



Horizontal and vertical distances

- **Variable:** There are two variables, which are:
  - *Horizonal_dist – the number of selected digits which lie on the horizontal axis*
  - *Vert_dist – the number of selected digits which lie on the vertical axis*
  -
- **Description:** These two variables try to explain how disproportionate the clock face is, i.e. how far it deviates from a regular circle. Works similar to the major and minor axis.

- **Methodology:** After the ellipse is fitted to the clock face, we assume (512,512) is the center of the clock. Then two vectors are drawn, one horizontal and one vertical. The distance between intersections from the edge of the ellipse tells us roughly how deviated form a regular circle and off center this ellipse is.

- **Prerequisites:** The cropped image is required.

- **Example:**

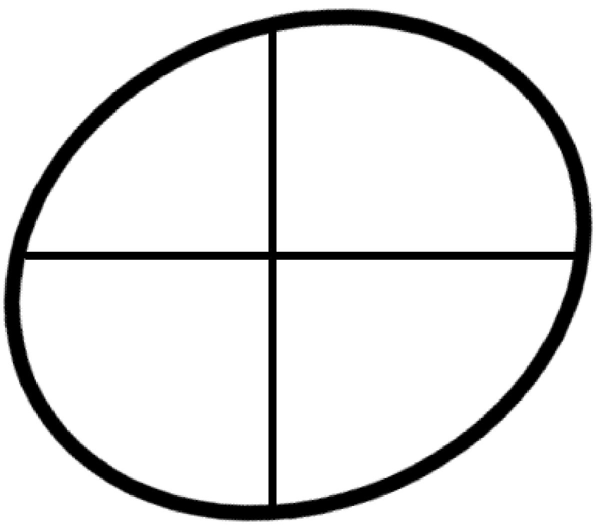


Figure 26: Fitted ellipse with intersecting horizontal and vertical vectors.

Euclidean Distance from Digits

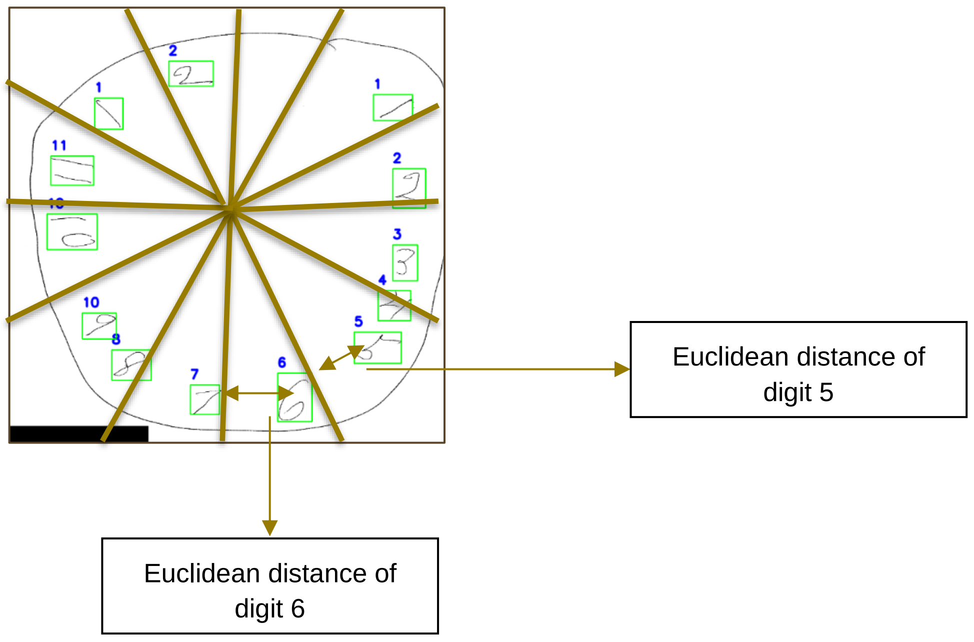
- **Variable/s:** There are a total of 12 variables
    - *euc_dist_digit_1*: Distance of digit 1 center point from its ideal location
    - *euc_dist_digit_2*: Distance of digit 2 center point from its ideal location
    - *euc_dist_digit_3*: Distance of digit 3 center point from its ideal location
    - *euc_dist_digit_4*: Distance of digit 4 center point from its ideal location
    - *euc_dist_digit_5*: Distance of digit 5 center point from its ideal location
    - *euc_dist_digit_6*: Distance of digit 6 center point from its ideal location
    - *euc_dist_digit_7*: Distance of digit 7 center point from its ideal location
    - *euc_dist_digit_8*: Distance of digit 8 center point from its ideal location
    - *euc_dist_digit_9*: Distance of digit 9 center point from its ideal location
    - *euc_dist_digit_10*: Distance of digit 10 center point from its ideal location
    - *euc_dist_digit_11*: Distance of digit 11 center point from its ideal location
    - *euc_dist_digit_12*: Distance of digit 12 center point from its ideal location

- **Description:** These variables captures the distance between the centre point of each digit drawn by a patient from its ideal position

- **Methodology:** Each digit on a clock is located at a particular angle with respect to its center. The angle between two consecutive digits is 30 degrees. Considering the centre to be passing through point (512,512) and using the angle, we get the equation of the line. This line represents us the ideal position of each digit. The center point of a digit is calculated and a perpendicular distance from this point to the line is calculated using co-ordinate geometry.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:** The image below is poorly drawn. Except 4, none of the other digits detected are located in their ideal position. The digit 6 is to the right of the vertical axis, similarly digit 5 is also located to the right of where it should actually be.



Distance of Digits from clock center

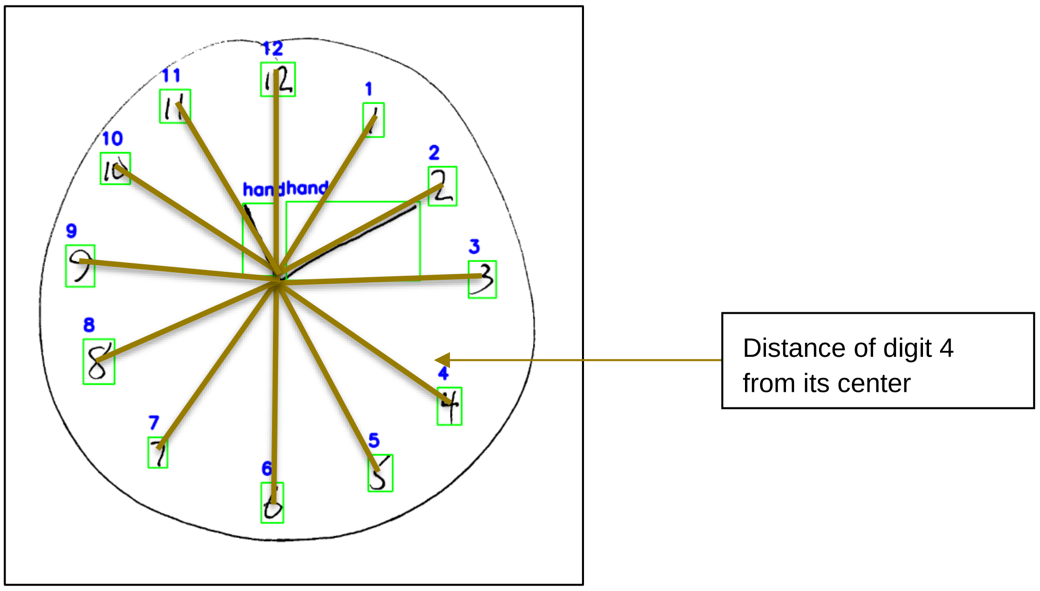
- **Variable/s:** There are a total of 12 variables
  - *1 dist from cen*: Distance of digit 1 center point from ideal clock center (512,512)
  - *2 dist from cen*: Distance of digit 2 center point from ideal clock center (512,512)
  - *3 dist from cen*: Distance of digit 3 center point from ideal clock center (512,512)
  - *4 dist from cen*: Distance of digit 4 center point from ideal clock center (512,512)
  - *5 dist from cen*: Distance of digit 5 center point from ideal clock center (512,512)
  - *6 dist from cen*: Distance of digit 6 center point from ideal clock center (512,512)
  - *7 dist from cen*: Distance of digit 7 center point from ideal clock center (512,512)
  - *8 dist from cen*: Distance of digit 8 center point from ideal clock center (512,512)
  - *9 dist from cen*: Distance of digit 9 center point from ideal clock center (512,512)
  - *10 dist from cen*: Distance of digit 10 center point from ideal clock center (512,512)
  - *11 dist from cen*: Distance of digit 11 center point from ideal clock center (512,512)
  - *12 dist from cen*: Distance of digit 12 center point from ideal clock center (512,512)

- **Description:** These variables captures the distance between the center point of each digit drawn by a patient from clock center (512,512)

- **Methodology:** Calculating the coordinates of the center of each digit and then calculating the distance between the two points.

- **Prerequisites:** The digit and hand annotated raw output table is required.

- **Example:**



Area, Height, Width of Digit Bounding Boxes Metrics

- **Variable/s:**
  - area_digit_1 – Area of digit 1 bounding box. Similarly we have 11 other variables for all the other digits (area_digit_2, area_digit_3, etc.)
  - height_digit_1 – Height of digit 1 bounding box. Similarly we have 11 other variables for all the other digits (height_digit_2, height_digit_3, etc.)
  - width_digit_1 – Width of digit 1 bounding box. Similarly we have 11 other variables for all the other digits (width _digit_2, width_digit_3, etc.)
  - variance_width – Variation in width of digits
  - variance_height – Variation in height of digits
  - variance_area – Variation in area of digits

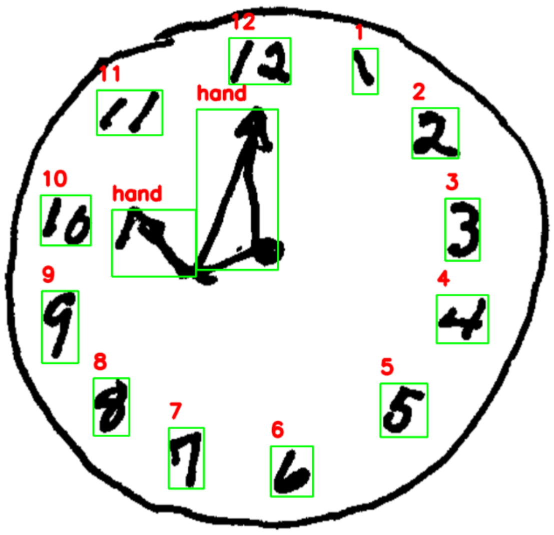
- **Description:** These variable calculates the variance of the area, height and width of the bounding boxes of the digits. This gives an idea about whether the user is able to write digits of similar dimensions.

- **Methodology:** The bounding box's height and width are calculated from the coordinates for each digit. Then the area is calculated. Since, this output is at an image-digit level, the data is rolled up to image level using the variance, to see the variability in digit dimensions. For the time being variance has been used, but other metrics can also be used.

- **Prerequisites:** The digit and hand annotated output

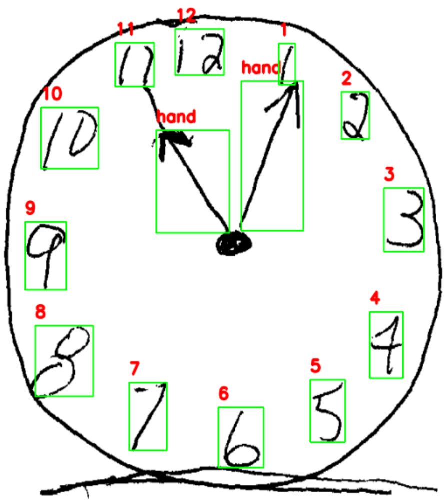
Time Features

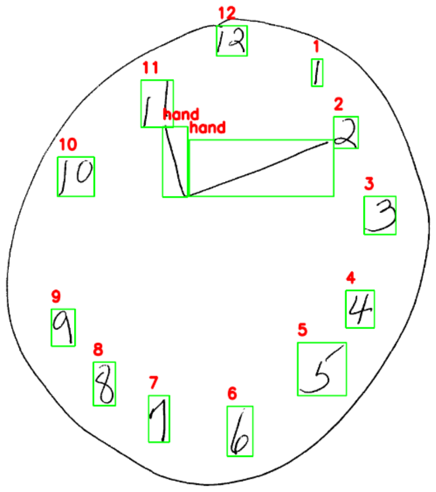
- **Variables**: Three variables are being calculated
    - *Time_diff* – If the time on the hand drawn image is different from 11.10, then the smallest time difference in minutes is calculated from 11.10.
- **Prerequisites**: The raw annotated output data is required to calculate the hour and minute pointing digits.
- **Methodology**— Keeping the actual hour and minute pointing digits to be 11 and 10 respectively, we calculate the minimum time difference in minutes from the time shown by the clock.
- **Example**— The clock below (ID: 10000044_R1) represents time 10:00, where the hour hand is pointing towards 10 and minute hand is pointing towards 12. The time difference from 11.10 is 70 minutes.

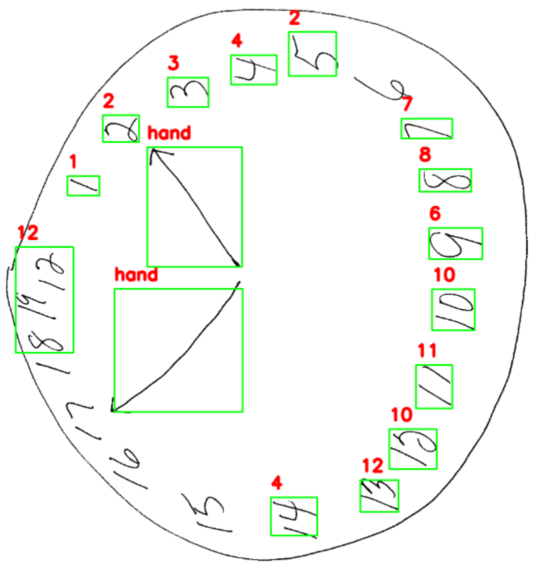


Center Dot Detection

- **Variable**:
    - *centre_dot_detect*—This variable represents if a dot is created at the intersection point of the two hands. The variable takes the values 0 and 1, where 1 represents the presence of a dot and 0 represents the absence of the dot.
- **Prerequisites**: Cropped images are required
- **Methodology**-- A model is trained to identify the center dot on top of the cropped images.
- **Examples**: The first image shows the presence of a center dot, whereas the second and third image shows the absence of a center dot.







Horizontal and vertical count

- **Variable:** There are two variables, which are:
  - *Hor_count – the number of selected digits which lie on the horizontal axis*
  - *Vert_count – the number of selected digits which lie on the vertical axis*

- **Description:** These two variables try and give us an understanding on how “good” the clock is.

- **Methodology:** As we assume the center of the clock to be (512,512), we can draw a horizontal and vertical vector through the clock. The number of intersections of each vector with the detected digits will increase the count for that vector, only the digits which are supposed to lie on each vector are counted. I.E. for the vertical vector, only digits ‘9’ and ‘3’ are considered.

- **Prerequisites:** The cropped image is required.

- **Example:**

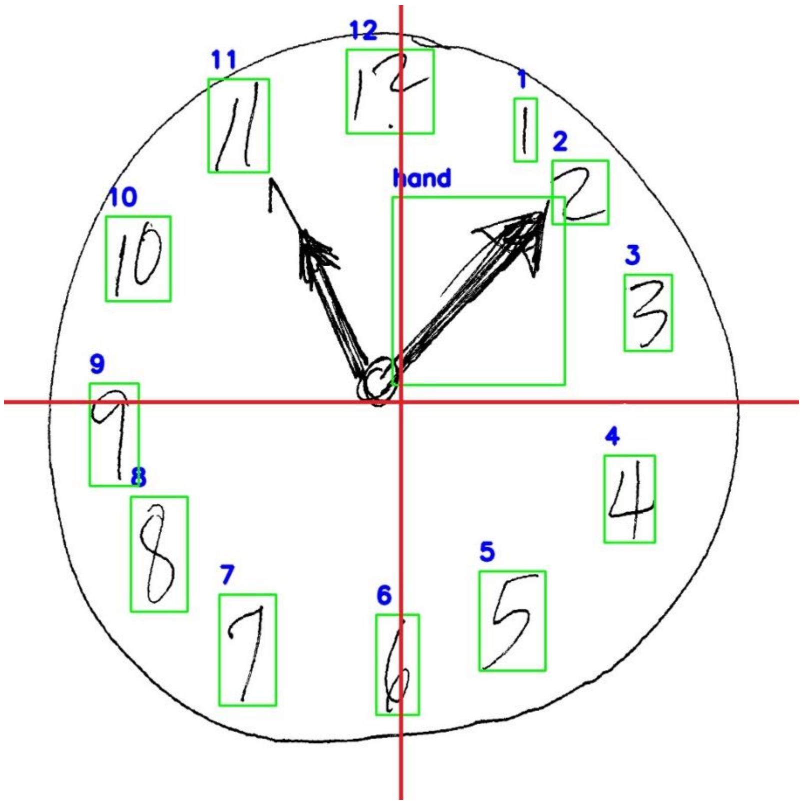


Figure 27: Horizontal and vertical vector applied. As you can see, the vertical count would be '2' and horizontal counter would be '1'.