Department of Information Engineering and Computer Science

Project of Signal, Image and Video:

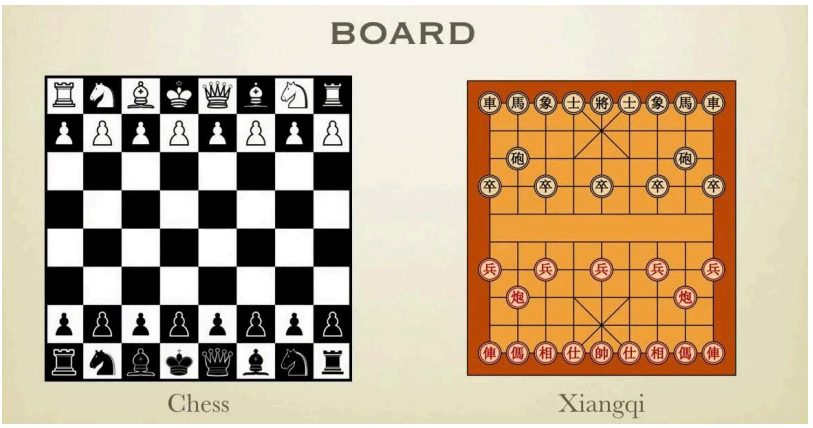
Chinese Chess Character Recognition

Silvano Vento Maddonni

Academic year 2023-2024

Introduction

Chinese chess, also known and referred to as Xiangqi, is a very popular strategic board game in many Asian countries. It is played between two players, each in control of one side, black or red. Xiangqi is played on a board that has 9 horizontal lines and 10 vertical lines and, as in the game Go, the pieces are placed on the intersections.



Other than the different board grid and rules, the big difference with respect to regular chess is the shape of the pieces.


Western Chess

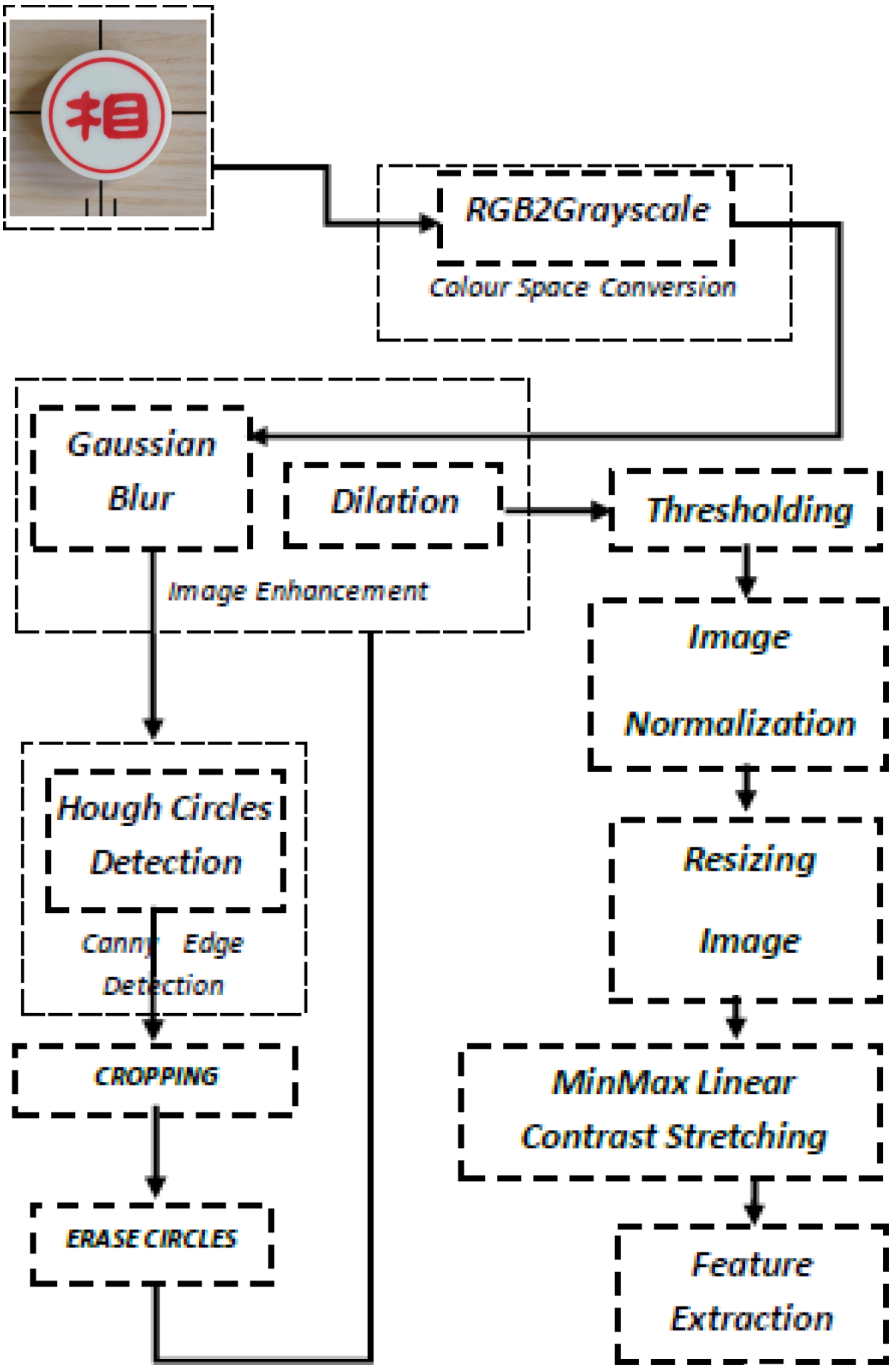
Chinese Chess

Each player controls an army of 16 pieces of 7 different types. Pieces are flat circular disks labeled or engraved with a Chinese character identifying the piece type.

Procedure

In this project we develop a procedure that is able to detect and locate pieces on the board using various image processing techniques.

The most important feature of the Xiangqi pieces is that they are circular, so the main idea of the procedure is to detect circles in the image and once individuated, extract the features inside them. This revolves around the Hough Circle Detection (using OpenCV).



Pre-Processing

Starting from the input image, several pre-processing methods have to be used in order to correctly prepare the image for the circle detection step and to be able to extract the features.

Grayscale

The first step in the procedure is the conversion to grayscale.
After this, the grayscale image is used as a starting point for many other modifications.

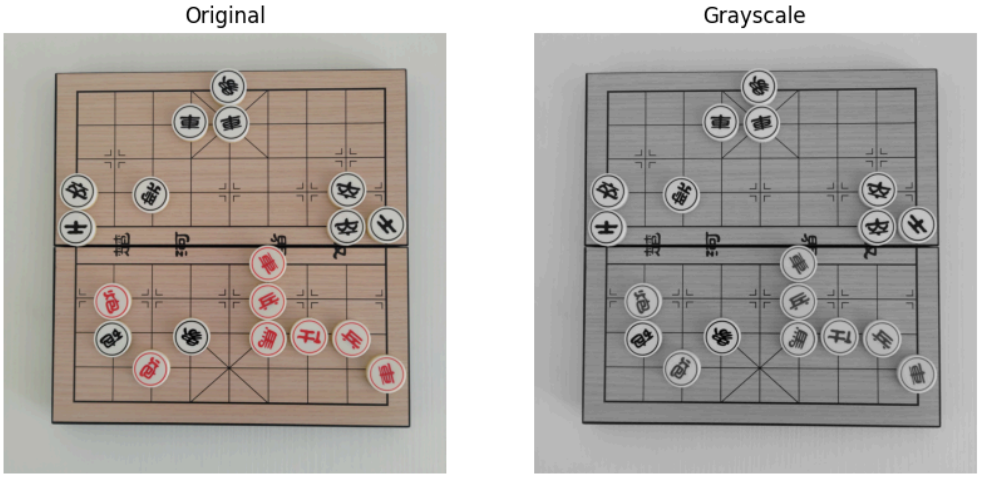


Image Smoothing (Blurring)

Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noise. In practice it removes high frequency content from the image, so the edges are blurred a little bit in this operation.

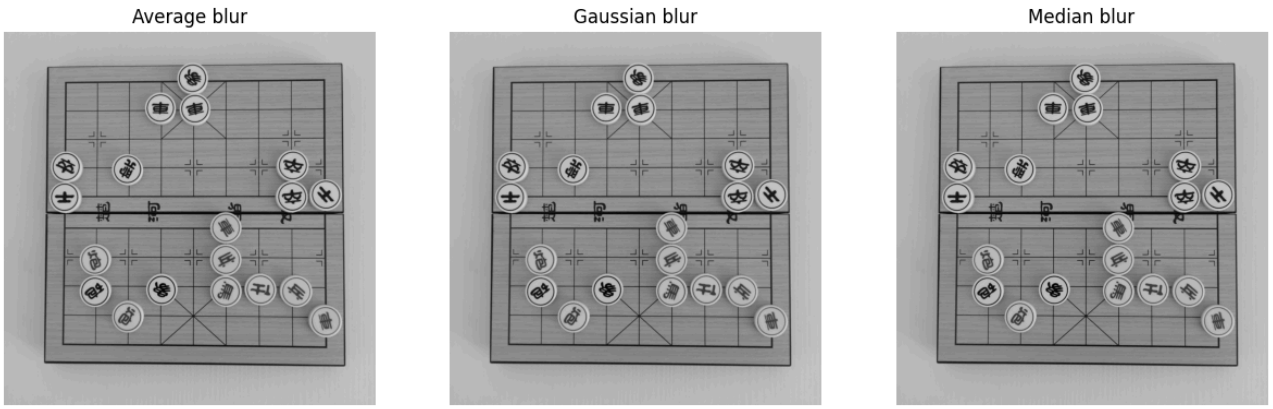
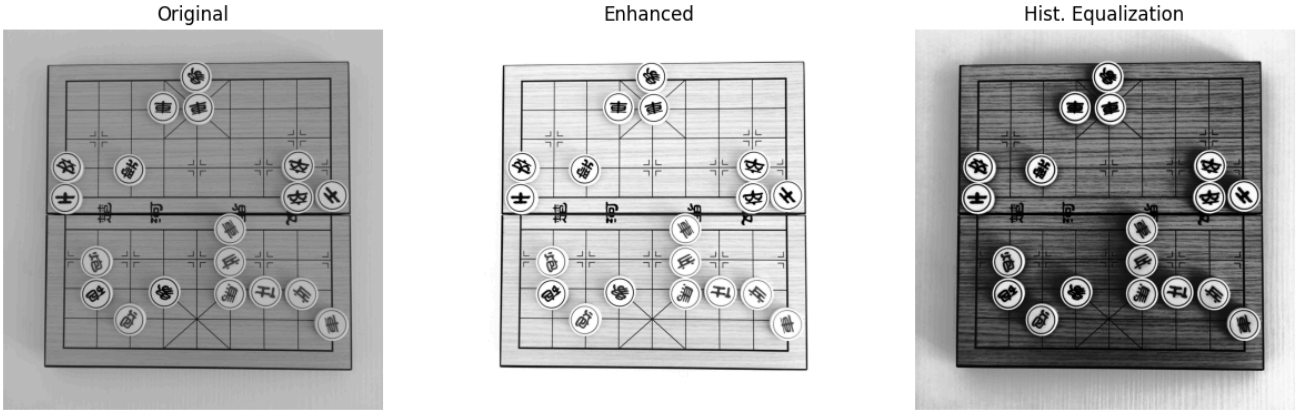


Image Enhancement and Histogram Equalization

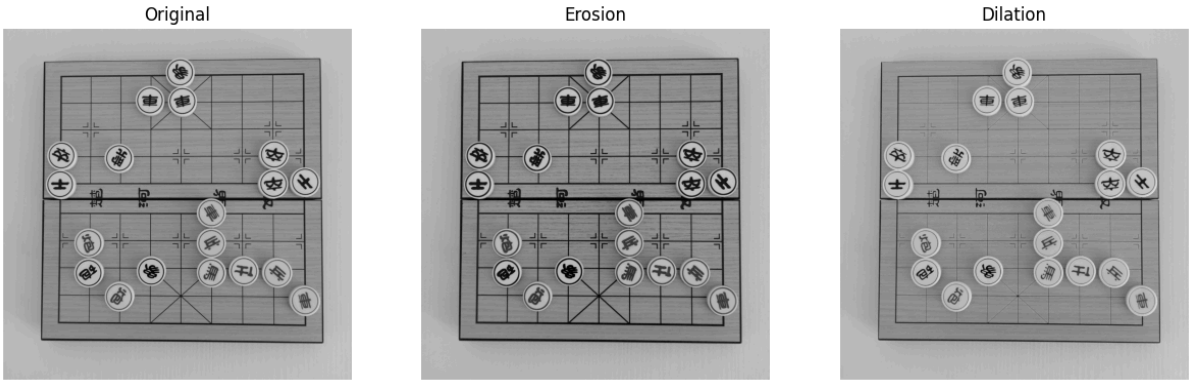
These two techniques are used to better highlight some characteristics of the image by working on brightness and contrast. In the histogram equalization what we do is stretch out the intensity range.



Morphological Transformations - Erosion and Dilation

Morphological operations are a set of operations that process images based on shapes by applying a structuring element to an input image and generating an output image. The most basic morphological operations are: Erosion and Dilation. They are used to remove noise, isolate individual elements or join separate components (for example fill little holes in the image).

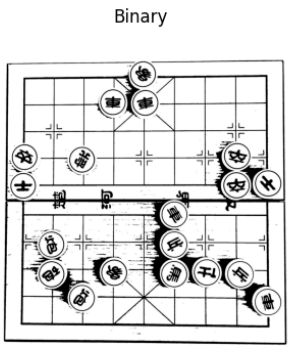
This operation consists of convolving an image A with some kernel (B) and then compute the maximal pixel value overlapped by B for the Dilation, minimal pixel value for Erosion.

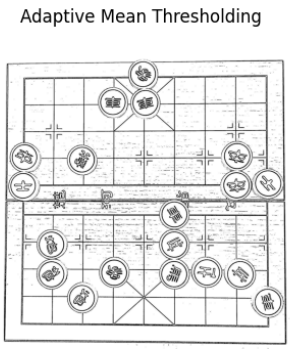


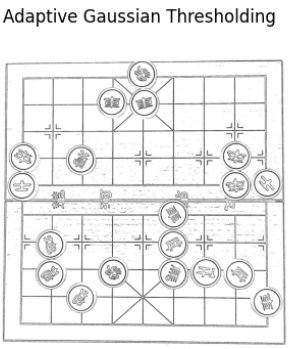
Later in the analysis of single smaller characters we also used Opening which is an erosion followed by a dilation and Closing which is a dilation followed by an erosion.

Thresholding

Thresholding is one of the most common and basic segmentation techniques in image processing and it allows us to better separate the objects that we are interested in from the rest of the image.
In cases (like ours) where an image has different lighting conditions in different areas, it could be better to use adapting thresholding where the threshold for a pixel is based on a small region around it rather than only on the pixel.

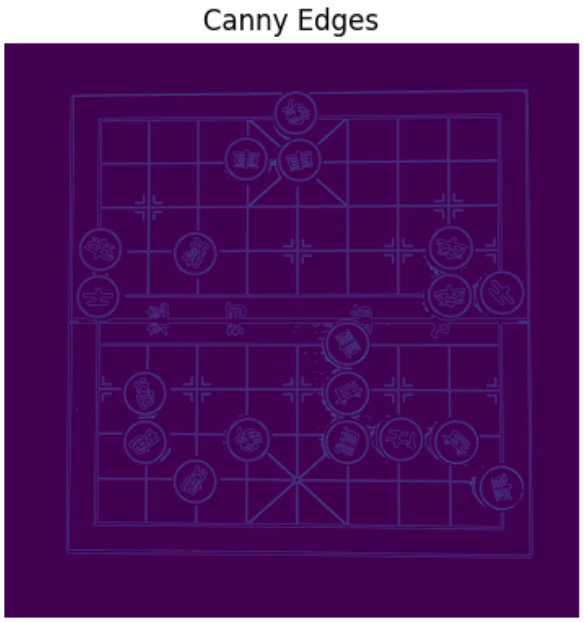


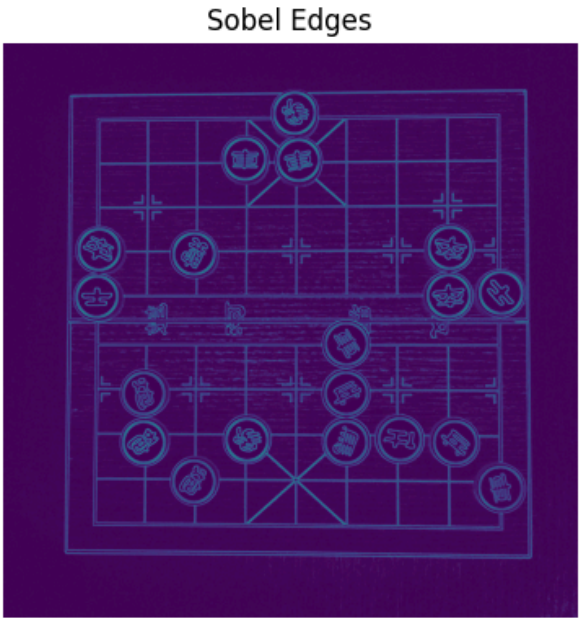




Edge detection

Edge detection is an image-processing technique that is used to identify the boundaries (edges) of objects or regions within an image.
Sudden changes in pixel intensity characterize edges. We need to look for such changes in the neighboring pixels to detect edges. Two important edge-detection algorithms have been used: Canny Edge Detection and Sobel Edge Detection.

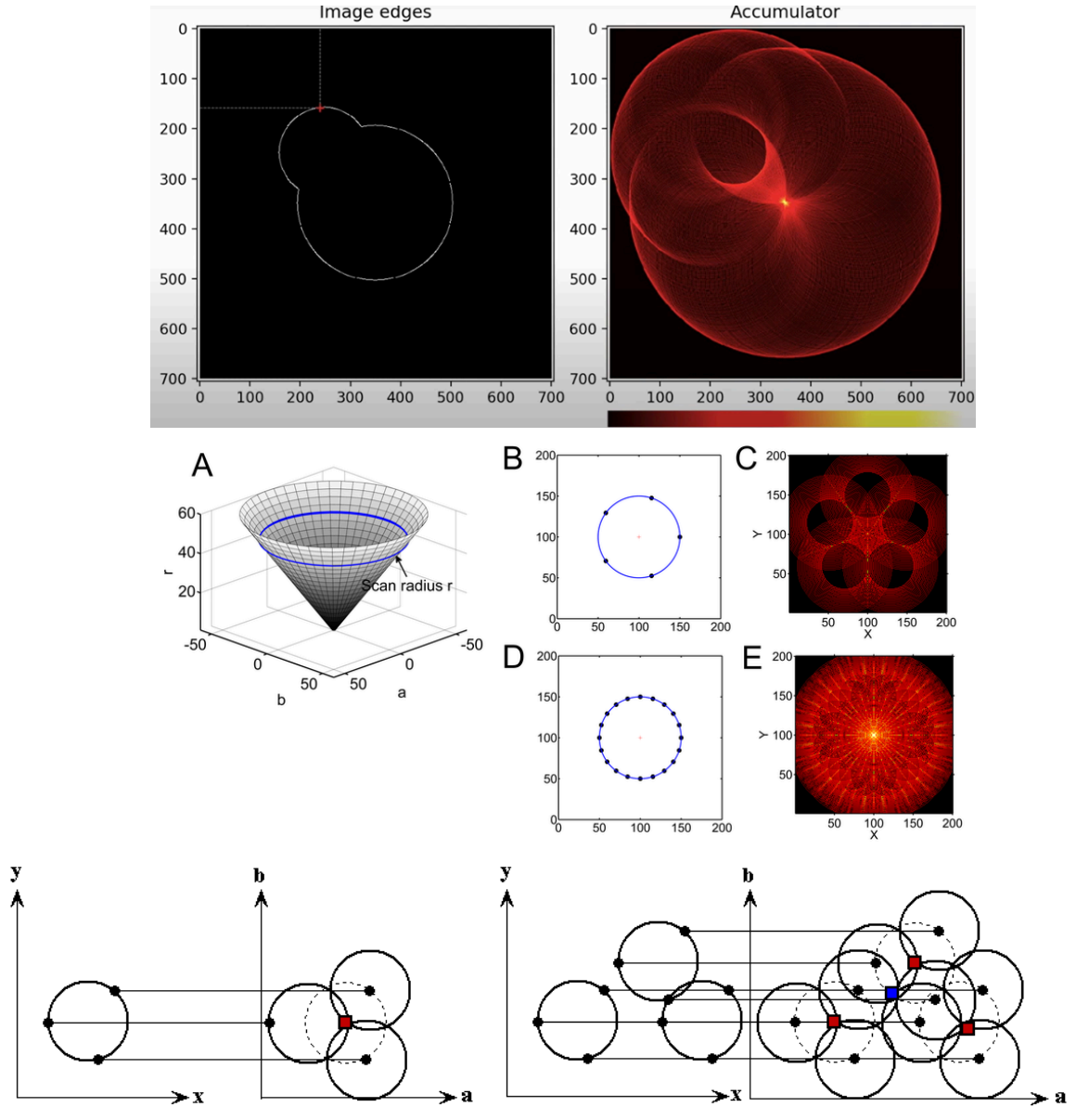




Hough Circle Transform

The Hough transform is a feature extraction technique that is used to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.
The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses.

Hough circle transform draws circles at a certain radius by traversing the edges of the input image with the help of an accumulator with the background. The point where all these circles intersect gives us the center of the circle.



Hough Circles Parameters and finding bests for our purpose.

cv.HoughCircles(img, **method**, **dp**, minDist, **param1**, **param2**, minRadius, maxRadius)

method

This is the mathematical formula used to find circles. The formula used is cv2.HOUGH_GRADIENT: this method takes into account the gradient information and it will increment the accumulator cells in the gradient direction of each edge pixel.
This algorithm consists of 2 steps: first, it finds all the plausible candidates for the circle centers, then for each candidate center, it finds the best radius.

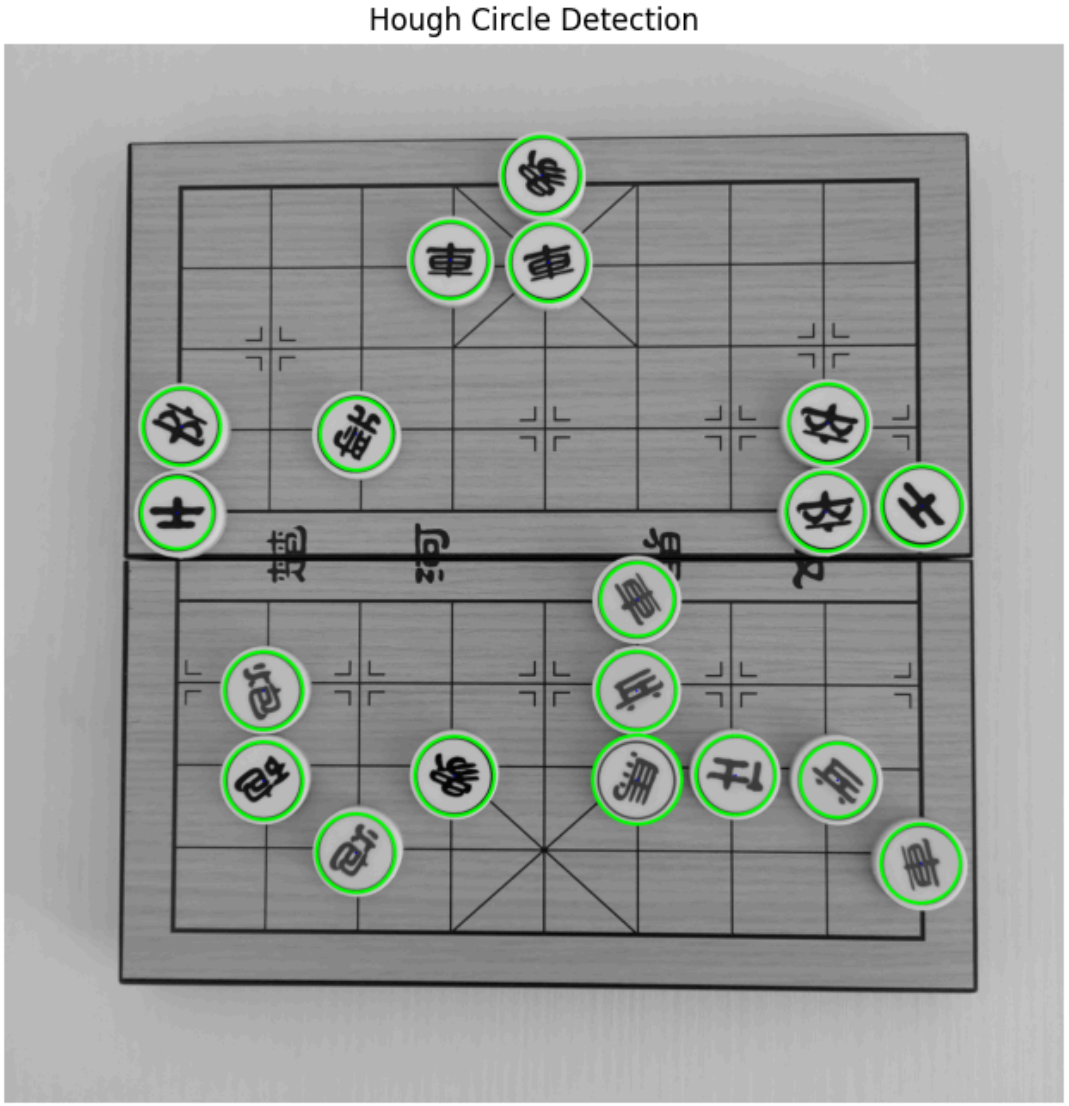
dp

dp represents the Inverse ratio of the accumulator resolution to the image resolution. During Hough transformation the input image is transformed into the so-called Hough space. It is 3-dimensional while trying to find circles: the three dimensions are the coordinates of the circle center and the radius. During the transformation each edge pixel in your input image votes for all possible circles on which the pixel could lie.
The voting works as increasing multiple values within a 3-dimensional matrix (Hough space). After voting, you search for the highest value within this matrix and read off the circle center and its radius.
The bigger the matrix (compared to your input image) , the smaller your *dp* and the higher the resolution of your voting. The higher the resolution, the more accurate the circle detection.
However, the more accurate the detection, the more likely it is to miss slightly degenerated circles or detect multiple circles instead of one with a big edge.

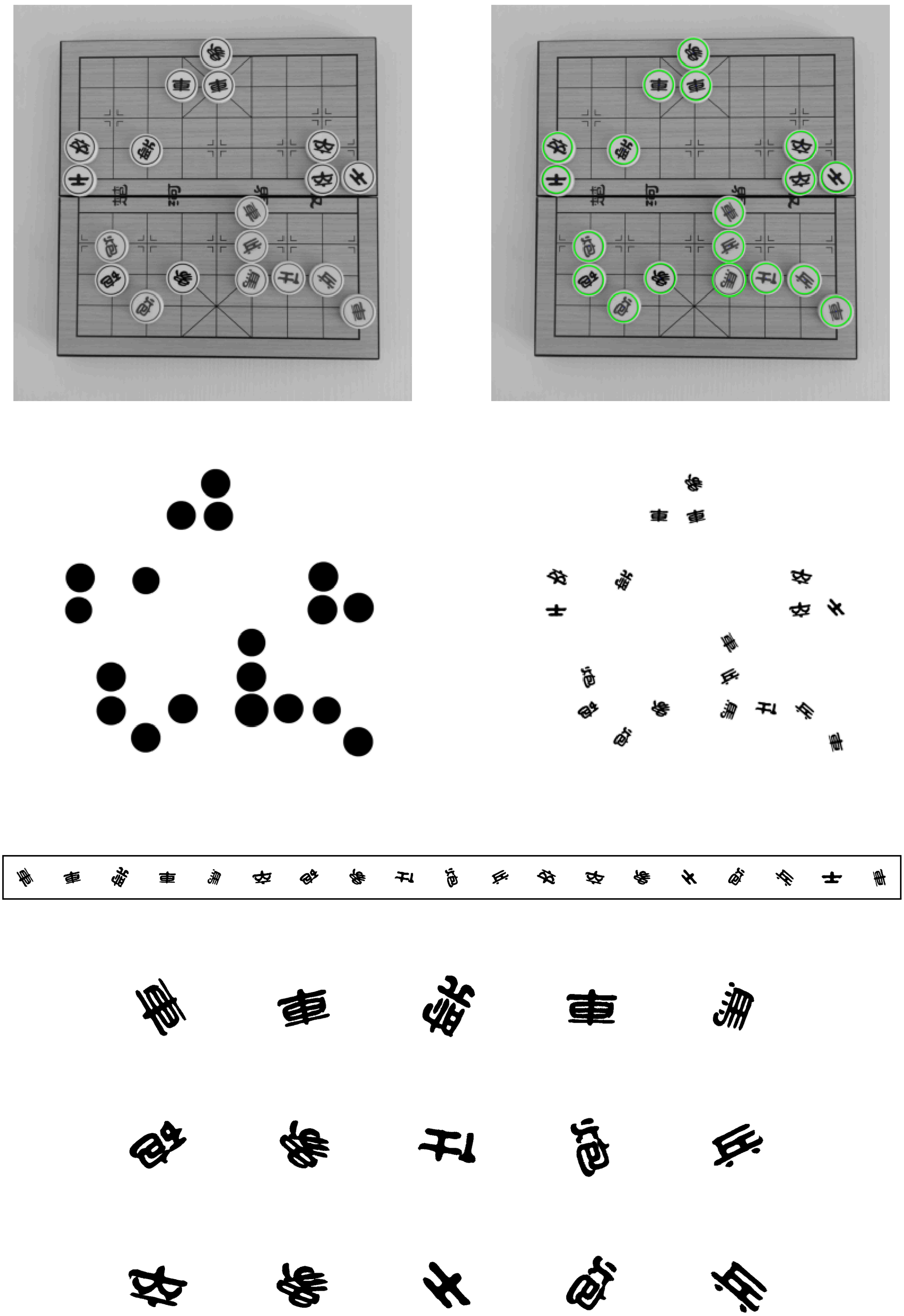
param1 and param2

HoughCircles() internally uses Canny() function. These parameters specify how aggressively you want to detect the edges.
param1 = First method-specific parameter. In case of HOUGH_GRADIENT , it is the higher threshold of the two passed to the Canny edge detector.
param2 = Second method-specific parameter. In case of HOUGH_GRADIENT , it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected. Circles, corresponding to the larger accumulator values, will be returned first.

The thresholder used in the Canny operator uses a method called “hysteresis”. Most thresholders used a single threshold limit, which means if the edge values fluctuate above and below this value the line will appear broken (“streaking”). Hysteresis counters streaking by setting an upper and lower edge value limit. If a value lies above the upper threshold limit it is immediately accepted, if the value lies below the low threshold it is immediately rejected. Points which lie between the two limits are accepted if they are connected to pixels which exhibit strong response.

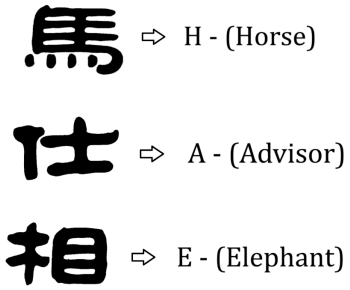


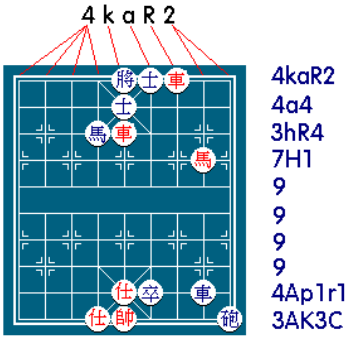
In our program we pass to the HoughCircles function our image pre-processed and enhanced. After the circle detection, using a mask, we extract all the single characters. Having precedently applied binarization and blurring/correction on the image, we are able to get a clean output of the extracted black character on white background.



Character Identification

After we detect the circles and we extract each single character, we need a way to identify it. Ideally at the end what we want is to be able to assign to each extracted fragment its relative corresponding letter, in order to convert the position in FEN string. FEN is "Forsyth-Edwards Notation"; it is a standard for describing chess (and Chinese chess) positions using the ASCII character set.
A single FEN record uses one text line of variable length composed of six data fields. Each field is composed only of non-blank printing ASCII characters. Adjacent fields are separated by a single ASCII space character.





4kaR2/4a4/3hR4/7H1/9/9/9/9/4Ap1r1/3AK3c

To do this we use a composite method of findContours and matchShape. It is generally a weak approach for recognition but in this case it works because we have clean characters and also one important thing is that it is rotation invariant.

For each fragment (each single char) we use this contours-matchShape function to see if it matches one of the reference characters in our characters' database.
The function works in this way:
1) Identify contours in the target image.
2) Identify contours in the template images.
3) For good candidates the template's contours are drawn over the input image and the area inside it gets erased.
4) On the erased image we calculate the percentage of black pixels that has been removed by this operation. Over a certain threshold we declare the char recognized.