Extracting Go Game Positions from Photographs

Teemu Hirsimäki*
Laboratory of Computer and Information Science
Helsinki University of Technology

February 1, 2005

Abstract

Go is an ancient oriental board game played by two players who place white and black stones alternately on the intersections of a rectangular grid. With digital image processing and computer vision methods available today, it should be possible to devise a system, which follows the game with a video camera and analyses where the players place new stones or remove stones.

In this article, I present an automatic system for analysing still images of go boards. The system is able to locate the playing grid and the stones from the image. The performance of the system is demonstrated with a few test images, and the possibilities to extend the analysis to video streams is discussed.

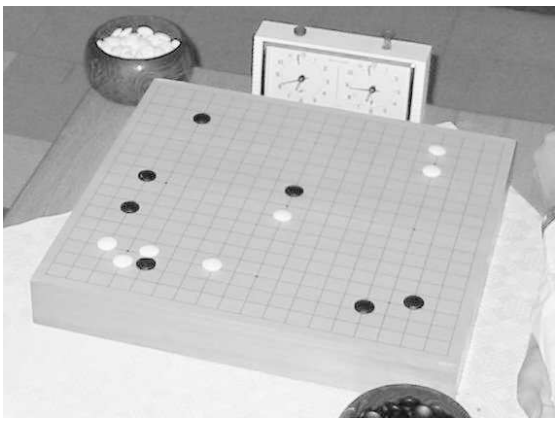

Figure 1: A go game after seven moves by both players.

1 Introduction

Go is a two-player board game which is popular in oriental countries, but not very widely known in Europe or America. The game is played by placing white and black stones alternately on the intersections of a rectangular 19×19 grid, and trying to capture opponent's stones while also surrounding as much territory as possible. Figure 1 shows a photograph of a go game in progress.

Recording the moves of a game between two players is traditionally done manually with a pencil by marking the number of each move on a grid on a paper. Nowadays, it is also common to use a computers to mark the moves with a mouse, or a pen of a hand-held computer. However, with the digital image processing and computer vision techniques available today, one might consider building a system, which uses a video camera to capture the whole game and a computer program to analyse the video and infer where each stone was placed during the game.

In the following sections, we learn how the first steps of such automatic recording system can be built with relatively simple digital image processing and optimisation techniques. Instead of dealing with video stream, the task is here limited to analysing still images of a go games and trying to locate the playing grid and stones from the images.

2 Image analysis

2.1 Properties of a go board

Let's look at Figure 1 and think a little what are the most important features that make us think it as an image of a go board? The strongest cues are evidently the dark lines that form the playing grid and especially the strict regularity of the grid. Also, the grid lines are usually on a relatively homogenous background, but it would be easy for humans to spot the playing grid even

*E-mail: teemu.hirsimaki@iki.fi

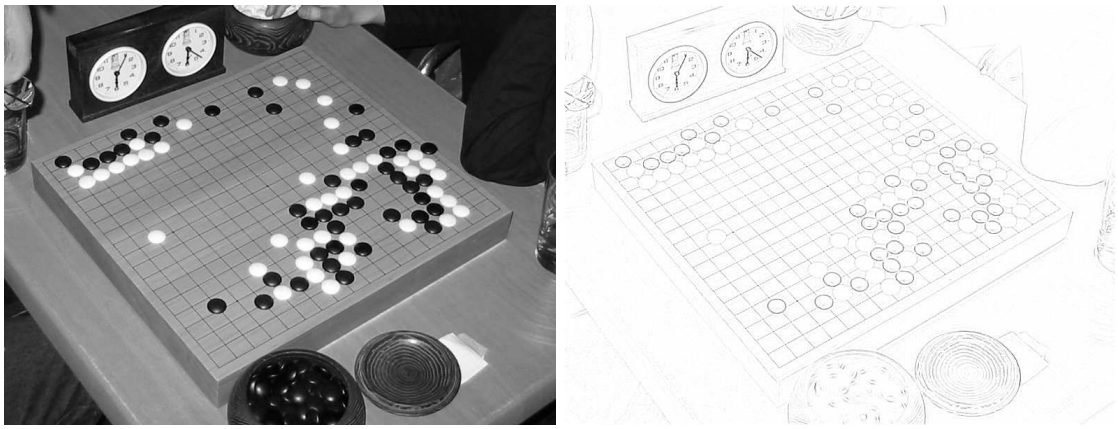

Figure 2: Left: original image I. Right: resulting image I_L after filtering with Equation 2 and cutting negative pixels (dark colours correspond to high intensity).

if it floated in air in front of a coloured background. A more important thing is that there are no other strong dark lines intersecting the grid lines.

There is one problem with relying on the lines, however. When a go game progresses, stones are placed on the board, and the grid lines will be obscured more and more. When the board is almost completely covered with stones, it would probably be more sensible to start the analysis by locating stones instead of lines. However, considering the target application, which records go games from the beginning to the end, it makes sense to start with lines, because the board is always empty at the beginning of the game. Also, if we can assume that the board and the camera do not move much during the game, and the grid is located already at the beginning of the game, it is probably not necessary to locate the grid from scratch later.

2.2 Detecting the line pixels

The first step of our image analysis is locating pixels that form the lines of the grid. A common digital image processing technique called *linear filtering* can be used to examine the local properties of the image pixels. In linear filtering, an *output image* O is created from an *input image* I by computing each output pixel $O(x, y)$ as a linear combination of the corresponding input pixel $I(x, y)$ and its local neighbourhood:

$$O(x,y) = \sum_{(x',y')\in\mathcal{N}} H(x-x', y-y')\, I(x', y') \quad (1)$$

where the local neighbourhood $\mathcal{N}$ and the coefficients H define the filter. It is also common to use rectangular neighbourhoods with odd number of pixels horizontally and vertically, so that the filter can be centred around one pixel. [Sonka et al., 1998, pp. 68–69]

Since the grid of a go board consists of dark lines on a lighter background, we base our analysis on this assumption. Pixels that are darker than their local neighbourhood can be found with a filter, that has a negative peak in the centre, and small positive values around the peak. Setting the filter values so that the their sum is zero, the filter becomes insensitive to the absolute values of the pixels and responses only to the value differences. For example, a simple 5×5-filter can be defined as follows:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -24 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (2)$$

The effect of the above filter is shown in Figure 2. On the left, is the original image I, and on the right is the resulting image I_L, which we refer to as the *line image*. In addition to filtering with H, the negative values have been set to zero in order to show the positive responses better. The remaining values have been normalised between zero and one.

Equipped with the line image I_L, and knowing that we are looking for a projection of a rectangular 19×19 grid, it would be possible to formulate the task of locating the grid as a direct optimisation problem. We could try fit-

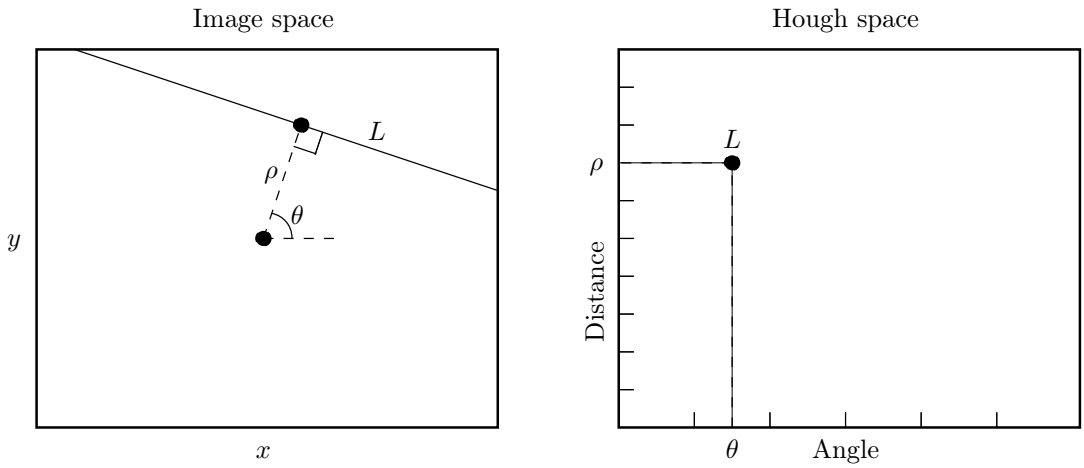

Figure 3: Line L in the image space maps to a point in the Hough space.

ting a projected grid on the image and computing, for example, the negative sum of pixel values along the lines of the grid in I_L. Then the task would be finding a grid position that minimises the sum. However, it would be very hard to find the global maximum without additional information. Thus, before thinking about the optimisation problem more, let's introduce another well-known image processing technique that helps us in our task.

2.3 Hough transform

Locating pixels that form straight lines can be done conveniently with the *Hough transform*. If we consider a pixel in the filtered image I_L for a moment, we know that the pixel can only be part of lines that cross the location of the pixel. This leads to the idea of the Hough transform: count how many pixels contribute to each possible line in order to find the strongest lines. All possible lines in the input image are represented with two parameters: the angle θ of the normal of the line, and the distance ρ between the line and the centre of the image. Figure 3 illustrates the parameters and the connection between the image space and the Hough space. Each line in the image space corresponds to a unique point in the Hough space. [Sonka et al., 1998, pp. 164–173]

In practice, the Hough transform is computed using a accumulator array $A(\theta, \rho)$, which quantizes the Hough space. For each non-zero input pixel (x, y), we consider every quantized angle θ' at time, and compute the corresponding distance $\rho(\theta')$ that makes the line go through the pixel (x, y). Then the corresponding cell in the accumulator array, $A(\theta', \rho(\theta'))$, is increased by the input pixel value at (x, y). After the whole image is processed, the strongest lines in the image form local maximums in the Hough space.

The left side of Figure 4 shows the Hough transform I_H of the image I_L of Figure 2. The parallel grid lines of the original image can be seen in the Hough space as a vertical series of peaks. The lines in the series have almost the same angle, but not exactly due to the perspective projection.

With the Hough transform image of Figure 4, we can start thinking how to actually locate the playing grid. Since the grid consists of 38 intersecting lines, the first idea might be finding the 38 strongest maximums in I_H. However, the grid lines are often quite thin and weak compared to other lines present in the image and, even though we know that the distance between parallel grid lines is regular, there might be other lines just outside the grid that would fit perfectly in the series of lines: edge of the board and lines in the tablecloth, for example. The problem is that the Hough space only contains information about the angle of the line and its distance from the centre of the image. It does not give any hints where the lines start and where they end in the original image. Thus, in order to separate the grid lines from other lines, the key is to combine the information in the Hough image I_H and the filtered image I_L.

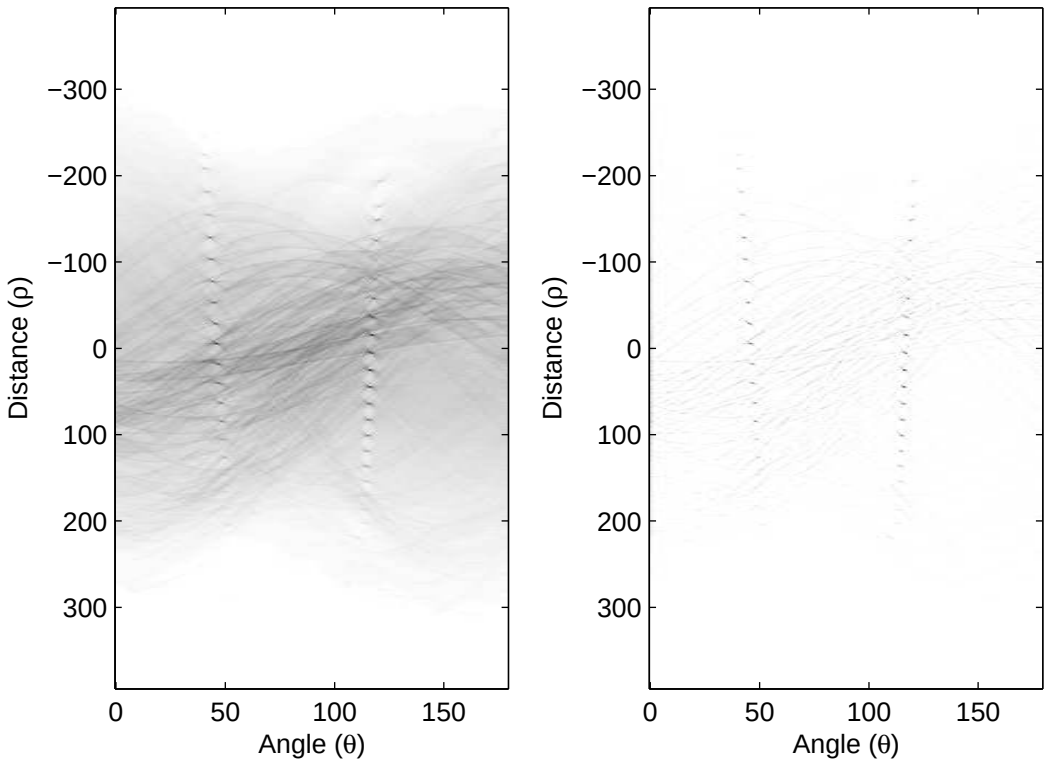

Figure 4: Left: Hough transform I_H computed from the line image I_L. The parallel grid lines of the original image form vertical series of peaks in the Hough space. Right: Filtered version of the hough transform of the enhanced line image I'_L.

3 Searching the grid

As briefly mentioned in Section 2.2, our task of locating the grid from the image can be formulated in a minimisation problem. One difficulty in finding the global minimum is that we do not have any good model for things that can appear outside the go board. There may be bowls and a clock near the board, and part of the players might be visible, as in Figure 1. The table and the table cloth can have arbitrary colours and textures. If our grid candidate is off the board, we have little clue in which direction we should move the grid to approach the desired location.

To alleviate this problem, we do not directly fit the full 19×19 to image I_L, but start with a smaller grid instead. Then it is easier to find a good initial guess for the grid position, ensure that the small grid is within the board, and to fit the grid with the edge image by minimising the cost function.

3.1 Making the first guess

3.1.1 Preprocessing

Since we are trying to record go games, we assume that most of the centre area of the original image consist of the go board and hope to find a good initial grid candidate around the centre of the image. To follow this assumption, we do an additional operation to the line image I_L before the Hough transform. After filtering the original image I with the filter H, and setting the negative values to zero, we apply a Gaussian weight function to give more weight to the centre pixels giving us the image $I_{L'}$:

$$I_{L'}(x,y) = I_L(x,y) \cdot \exp\left\{\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right\} \quad (3)$$

where (x_0, y_0) is the centre of the image. Setting the variances σ_x^2 and σ_y^2 to quarter of the width and height of the image seems to be a reasonable choice.

After transforming $I_{L'}$ to Hough space we also

filter the Hough image with a peak filter similar to Equation 2 to enhance the peaks. This time we use a positive peak because we are interested in locally high values. The resulting image $I_{H'}$ is shown at the right side of Figure 4.

3.1.2 Locating the peak series

The parallel grid lines of the original image form almost vertical series of peaks in the Hough image. First we want to find out the approximate location of the two peak series in the Hough image, i.e., what is the general orientation of the parallel lines. We do the following steps.

1. Blur the Hough image $I_{H'}$ temporarily with a horizontal filter $\begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}$, and weigh the columns of the image with a vertical Gaussian using tenth of the image height as the variance.
2. Compute the sums of the columns to get a function $S(\theta)$ and find its maximum. The location θ_1 of the maximum corresponds to the approximate location of the first peak series.
3. Remove the first maximum by setting $S(\theta)$ to zero between $\theta_1 - 25°$ and $\theta_1 + 25°$. Then, find again the maximum of the function. The corresponding θ_2 is the approximate location of the second peak series.

3.1.3 Selecting the middle peaks

After locating the approximate angles of the peak series, θ_1 and θ_2, we locate 12 peaks from each series around the centre, and choose the 5 at the middle as follows.

1. Take the columns $(\theta_i - 10°), \ldots, (\theta_i + 10°)$ from the Hough image $I_{H'}$, and weigh the patch with a horizontal Gaussian using half of the patch width as the variance.
2. Compute the maximum of each row $M(\rho)$, and the corresponding angle $M_\theta(\rho)$.
3. Then find 12 maximums by doing the following steps 12 times:
   (a) Find the maximum of $M(\rho)$, and set its location to ρ_i.
   (b) Remove the maximum by computing the median of $M(\rho)$ for $|\rho - \rho_i| < 10$, and zeroing values around ρ_1 to both directions until the value of the function drops below the median.
4. Then select the 5 median values of the values $\rho_1, \ldots, \rho_{12}$, and get the corresponding angles with the function $M_\theta(\rho)$.

This procedure seems to give reasonable line candidates for a small 5×5 grid. Sometimes, however, the procedure might miss a line or two because some grid lines may be very thin. This can be fixed by computing the distances between neighbouring lines, and checking if some of the distances is larger than 1.5 times the smallest distance. If this is the case, the gap can be filled by interpolating lines.

3.2 Tuning the grid

We can be now quite sure that the initial grid is within the actual board, and each of the grid lines is relatively near to its correct position. This means that a minimisation method can be used to tune the positions of the lines to match the pixels in the line image I_L exactly. The closest local minimum should be enough.

3.2.1 Cost function

There are many ways to choose the cost function for the minimisation. As mentioned earlier, we are planning to combine information from the Hough image and the gradient image, so one possibility is to define the cost function C as a weighted sum of two terms C_L and C_H, the costs computed from the line image and the Hough image respectively:

$$C(\alpha) = wC_L(\alpha) + (1 - w)C_H(\alpha) \qquad (4)$$

where α represents the parameters defining the grid location, and weight w controls the relative importance of the terms. The first term $C_L(\alpha)$ is computed by taking pixels that fall under the grid lines in I_L, and summing the square roots of the pixel values. Taking the square root dampens the very bright values caused by the stone edges that may have much stronger contrast than the grid lines. To compute $C_H(\alpha)$, we normalise $I_{H'}$ between zero and one, and take the pixel value corresponding to the line in question. For w, we use 0.9, which has been empirically found to be a reasonable choice.

Now the best grid location $\hat{\alpha}$ can be found by minimising the cost function:

$$\hat{\alpha} = \arg\min_\alpha C(\alpha) \qquad (5)$$

$$= \arg\min_\alpha \big(wC_L(\alpha) + (1 - w)C_H(\alpha)\big) \qquad (6)$$

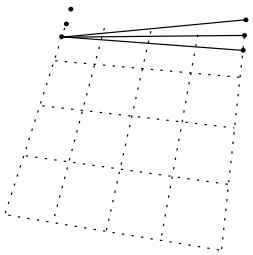

Figure 5: Optimising the position and angle of the topmost horizontal line. Each line is optimised separately by considering a few end points along the perpendicular border lines.

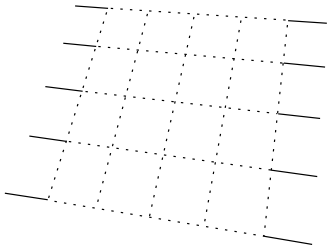

Figure 6: When grid is grown, the lines are extrapolated in opposite directions (solid lines). The goodness of a direction is evaluated by looking how well the extrapolated segments fit to the line image I_L.

Note that the above cost function does not set any restrictions for the shape of the grid as long as the grid matches the line image and the Hough image well. Surely, it is possible to define an additional term that would give penalty if the lines of the grid deviate too far from a projection of a rectangular grid. That would probably make the algorithm more robust in difficult situations. However, as we will see later, the above cost function works well for our purposes.

3.2.2 Finding the minimum

Defining the cost function is not enough as such. We also have to decide how we are going to find the minimum. Simply optimising each line in turn does the trick. Figure 5 shows how the end points of a line are varied along the perpendicular lines. For each line candidate, the value of the cost function is computed, and the best line location is selected. A reasonable range for the end points is half of the distance to the next line.

3.3 Growing the grid

After finding a reasonable initial guess and getting the small grid aligned nicely with the grid lines, there is still one thing to do: to grow the grid to full size. When we grow the grid, we have to take care that the grid does not fall off the board. The cue for detecting the edge line of the grid is that perpendicular lines do not continue over the edge line. This can be checked by extrapolating the grid in opposite directions, as illustrated in Figure 6, and computing which direction matches better with the line image I_L. Simply taking average of the pixels that are covered by the new line segments (solid line segments in the figure) and choosing the direction giving larger average will prevent the grid from growing outside the board.

Note, that average is only computed over the short line segments shown in the figure. The perpendicular line, that would be added to the end of the line segments, is not included in the computation for two reasons. If we simply extrapolate the position of the new line without tuning its position, we might miss the position of the line. Also, if we already are at the edge of the grid, the extrapolated line will most likely fall off the board, and we have no idea what lies outside the board. With bad luck, there might be something that actually gives a good response, and we might make a bad decision.

Now we have all pieces together for locating the full grid: the line image I_L, the enhanced Hough image $I_{H'}$, initial guess for the grid candidate, an algorithm for tuning the grid lines to match the image lines exactly, and an algorithm for growing the grid step by step. After each step of growing the grid, it is necessary to repeat the tuning, because the position of the new lines are computed by extrapolating the old ones, and especially with perspective distortion, the extrapolation errors accumulate easily.

3.4 Finding the stones

Assuming that the grid can be located even if there were stones on the board, we may lastly try deciding if there are any stones on the board, and where they are. Here we settle for a very simplistic approach. For each intersection (i, j) of the grid, we examine the original image I, in a 5×5 window centred at the position of the intersection (x_i, y_j). From that window we compute the median of the pixel values, getting a bright-

ness value $B(i,j)$. For each intersection, we also compute the median of the brightness values in a local 5×5 neighbourhood: $M(i,j)$. Then the stones $S(i,j)$ can be decided as follows:

$$S(i,j) = \begin{cases} \textit{white} & \text{if } \frac{B(i,j)}{M(i,j)} > T, \\ \textit{black} & \text{if } \frac{M(i,j)}{B(i,j)} > T, \\ \textit{empty} & \text{otherwise.} \end{cases} \quad (7)$$

A good value for the threshold T can be determined empirically. We will set T to 1.2 for now.

4 Tests and discussion

4.1 Tests

The system described above was implemented in MATLAB.[1] Figures 7–12 show six test images demonstrating how the system works. In each figure, the original image is shown at top, and the analysis of the system is shown below. The white stones are marked with circles and black stones with asterisks. The images in Figures 7–10 have been analysed correctly, one black stone has been missed in Figure 11, and Figure 12 shows an image on which the system failed, because it could not find sensible initial lines from the Hough image, shown also in the figure.

4.2 Problems in detecting lines

As can be seen from the test images, the system finds the playing grid quite well even if there are some stones on the board, as long as the most of the lines are visible. Also, low viewing angles are tolerated. Figure 12 shows perhaps the biggest weakness of the current approach. The analysis fails simply because there are too many stones obscuring the centre lines. The peaks into Hough image are too weak and the algorithm fails to find them.

There are a few techniques that might improve the Hough image. If the local line directions were estimated for each pixel in the line image I_L, the information could be taken into account in the Hough transform, so that only part of the array cells need to be updated [Sonka et al., 1998, pp. 169]. That would probably reduce the noise in the Hough image. There are also many variations and extensions of the Hough transform presented in the literature, that might be more robust to noise. See, for example, the survey given in [Kälviäinen et al., 1995].

Whatever method for finding lines is used, the lines can not be found if the board is completely covered by the stones. Considering the end application, if we can assume that the board and the camera are not moving during the game, it is enough to locate the grid in the beginning of the game and perhaps to only fine tune grid during the game. However, if the application should tolerate moving of the camera, an analysis based on finding stones would be needed.

4.3 Problems in detecting stones

In Figure 11, for example, we see that the stones are seldom located exactly at the intersections. Because the stones are lens shaped, and their thickness can be even half of their radius, the centres of the stones are elevated from the board. Additionally, the stones might have been placed carelessly off the intersections, which can make the effect even stronger. The current procedure for finding stones is quite simple, but seems to work well enough for the test images. It might be possible to try finding the centre of the stone before analysing the colour of stone from a small window, but then it should be decided if there is a stone on the intersection in the first place.

The other difficulty in detecting stones is that the lighting conditions can vary in different parts of the board. For this reason, trying to set simple threshold values for white and black stones is not very robust. The intensity of the board surface on a brighter part may be quite close to the intensity of the white stones in a darker part. Also, black stones may have surprisingly bright reflections from lamps above the board. The current median based analysis seems to be quite robust, but assumes that there are not too many stones of one colour in the local neighbourhood. The most straightforward way to improve the stone detection algorithm is probably to use colour images instead of gray-scale images.

4.4 From still images to video

So far we have analysed only still images, but in the end application which would analyse video stream, it would probably be useful to analyse consecutive frames, too, in order to detect if

[1] The source code of the system is available for download at http://www.cis.hut.fi/thirsima/gocam/

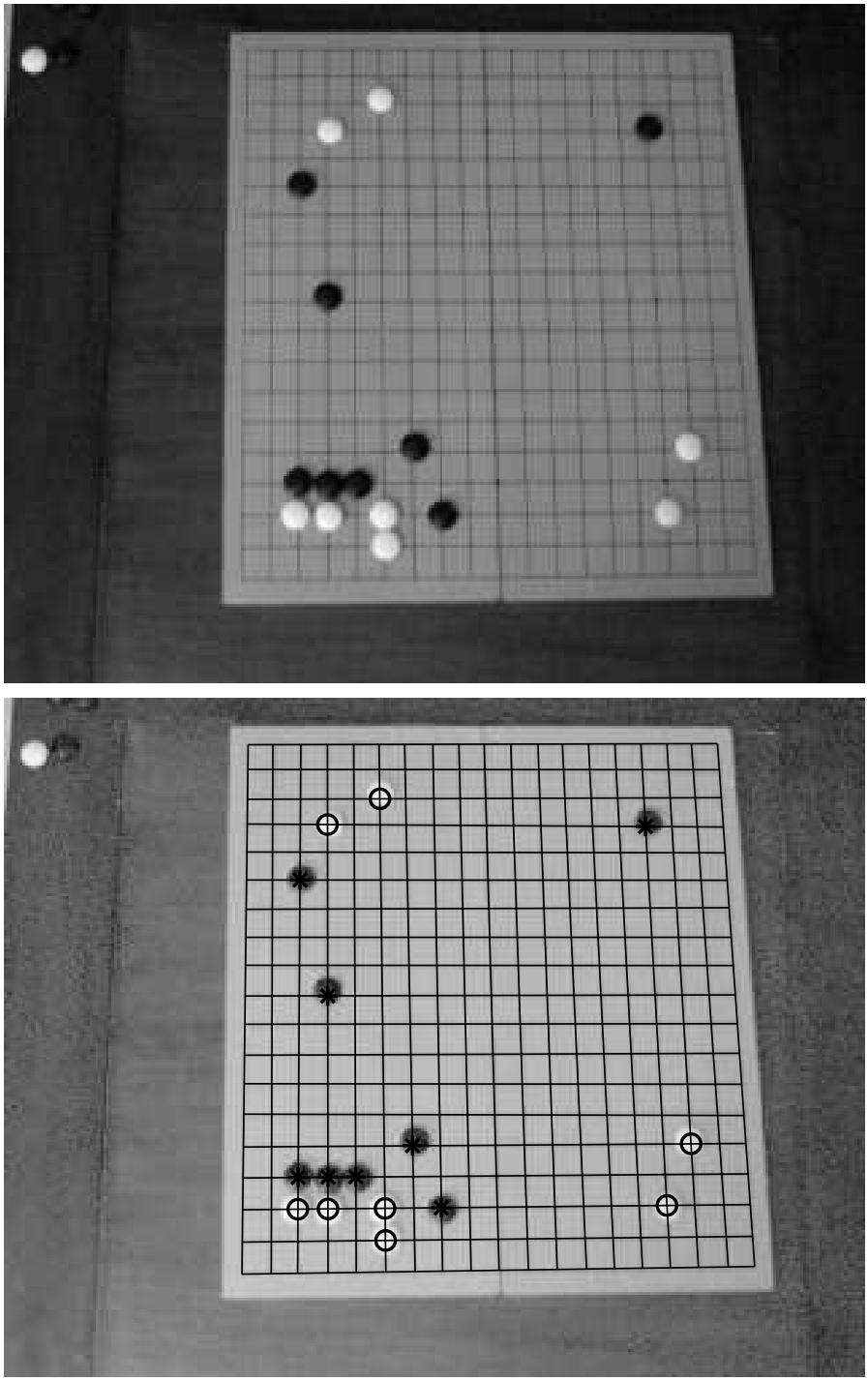

Figure 7: Test image 1 (320×240): Top: The original image. Bottom: The analysis of the system. Black stones are marked with asterisks, white stones with circles.

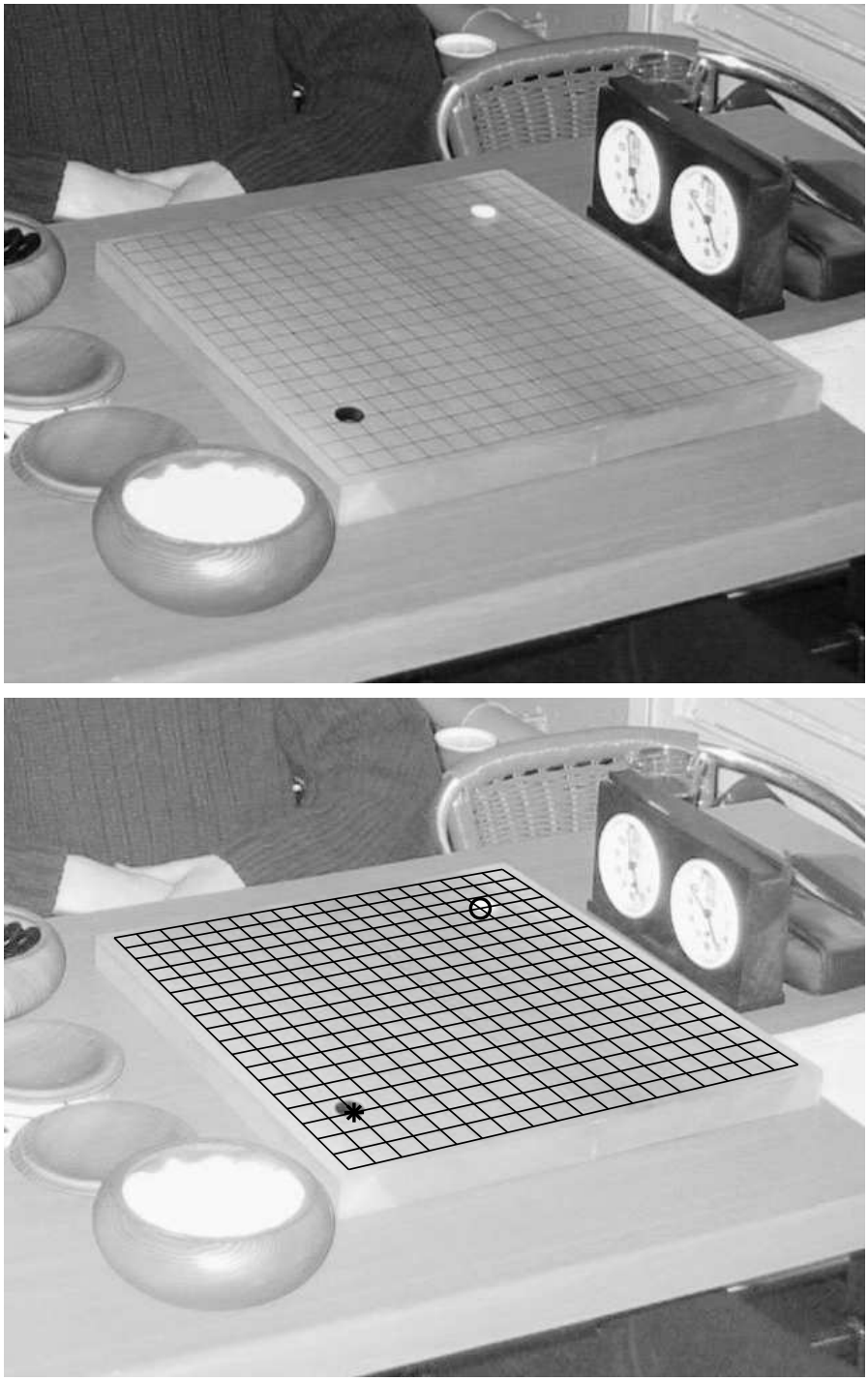

Figure 8: Test image 2 (640×480): Top: The original image. Bottom: The analysis of the system. Black stones are marked with asterisks, white stones with circles.

Figure 9: Test image 3 (350×280): Top: The original image. Bottom: The analysis of the system. Black stones are marked with asterisks, white stones with circles.

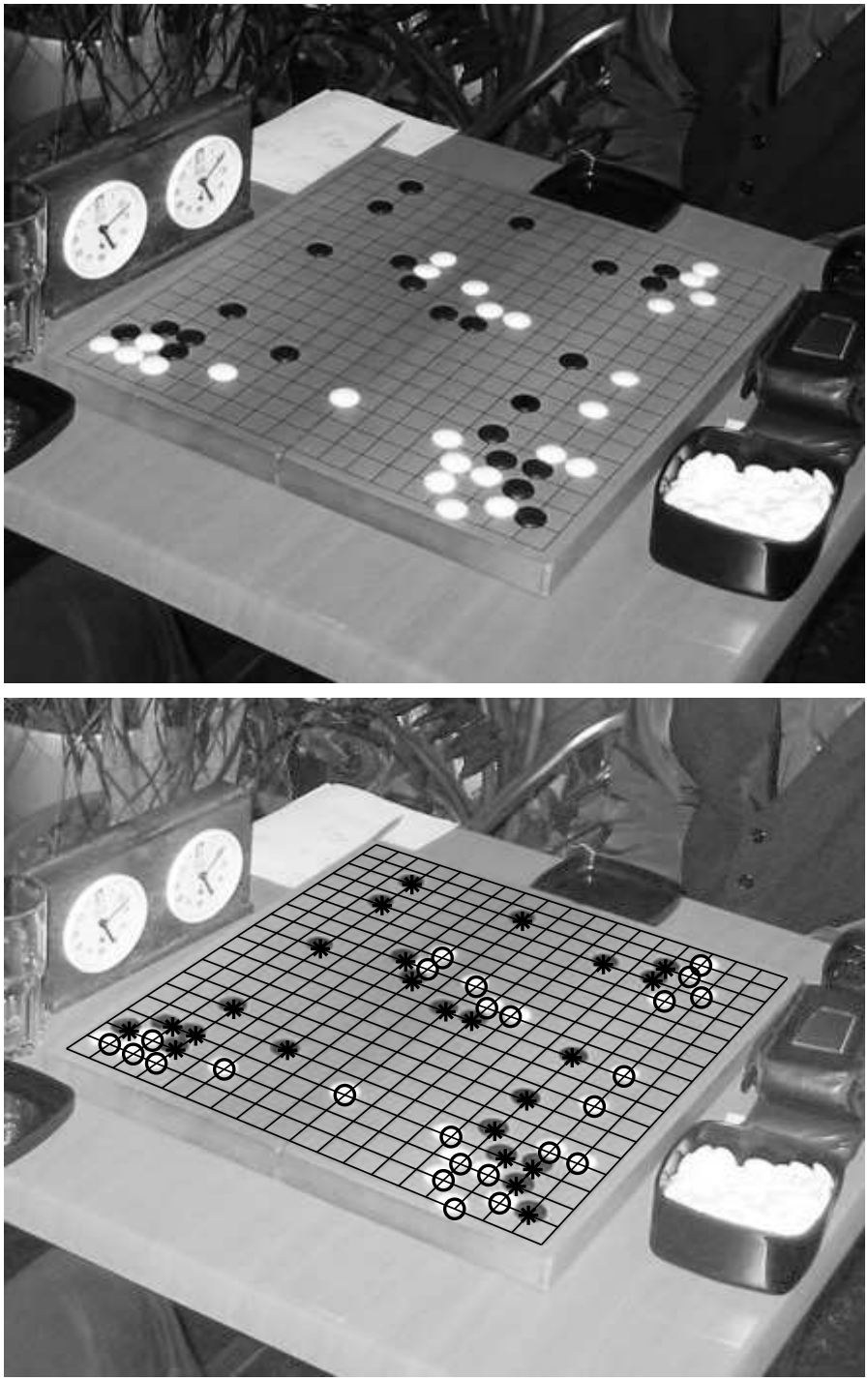

Figure 10: Test image 4 (440 × 373): Top: The original image. Bottom: The analysis of the system. Black stones are marked with asterisks, white stones with circles.

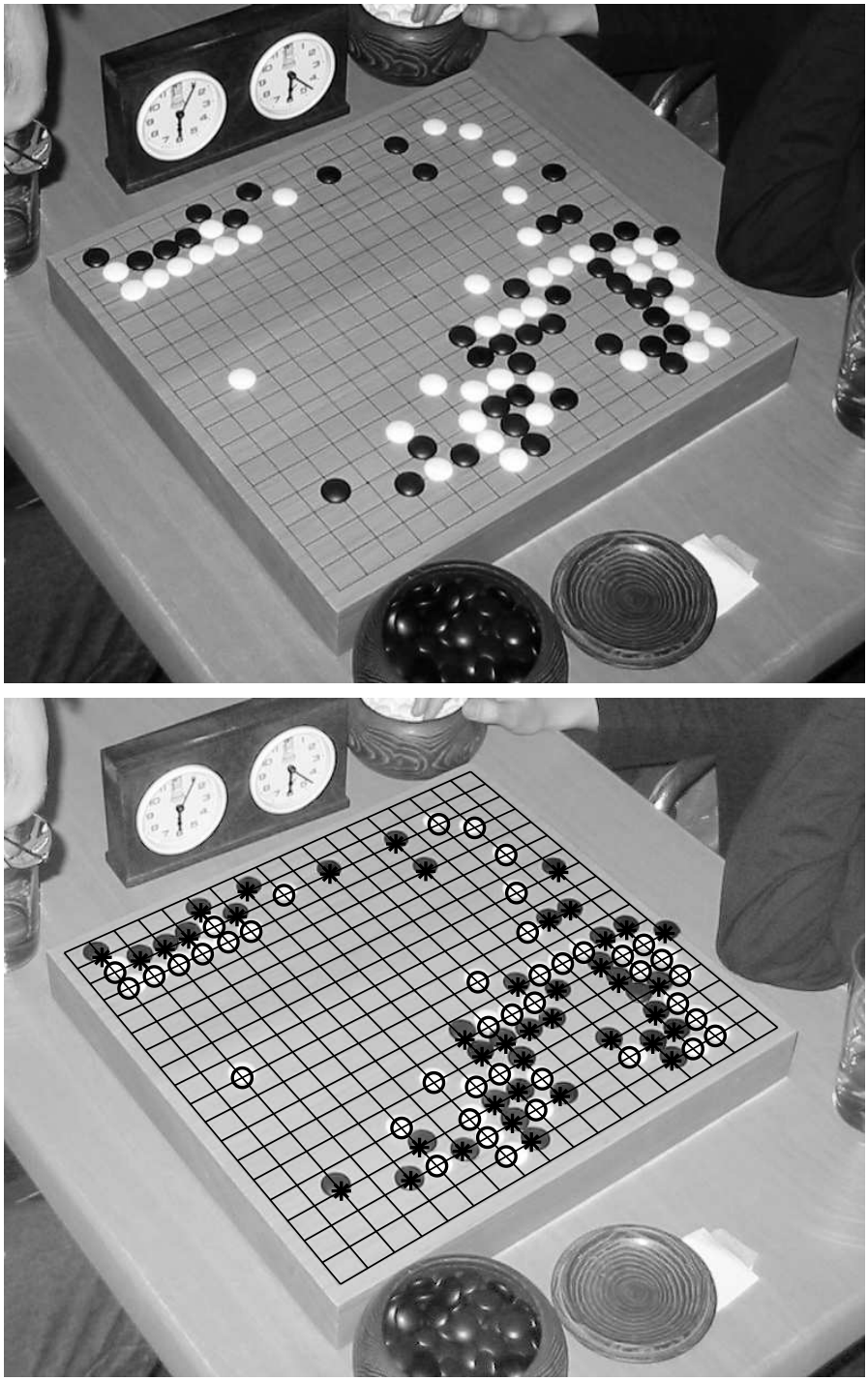

Figure 11: Test image 5 (790 × 600): Top: The original image. Bottom: The analysis of the system. Black stones are marked with asterisks, white stones with circles.

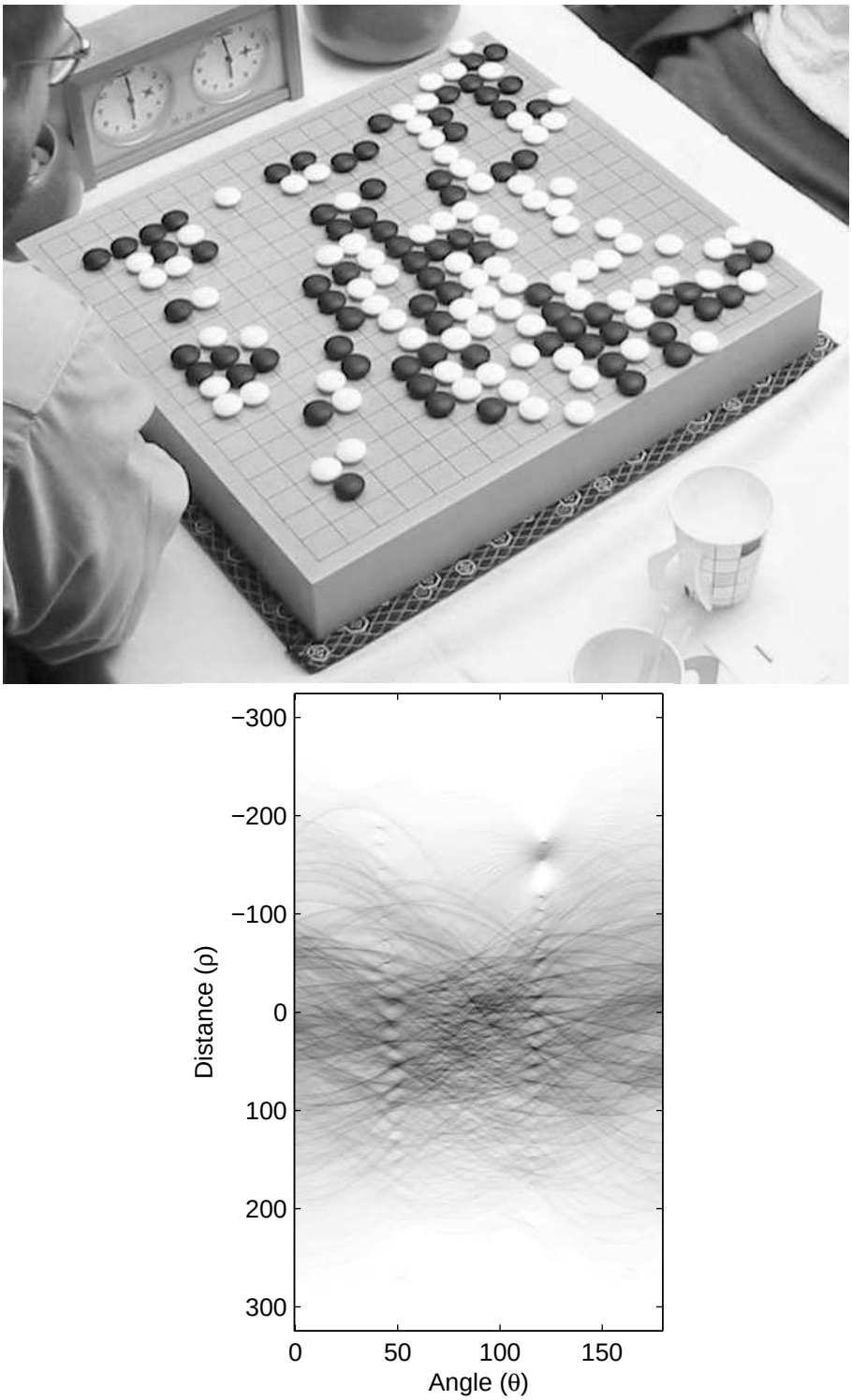

Figure 12: Test image 6 (650×522): Top: The original image, which the system could not analyse correctly. Bottom: The Hough image.

the colour around each intersection has changed. Also, it is perhaps not necessary to do the full procedure of finding lines and stones for every frame, if it can be assumed that the board and the camera do not move much during the recording. Then a separate fast analysis would be necessary to decide if the position on the board has changed. Also, the system should be able to decide if it is worthwhile to analyse the position at all. When a player is placing a stone on the board, the board is partly obscured.

5 Conclusion

All in all, the presented system for analysing photos of go boards and reading the position on the board seems to work well. It finds the grid lines using linear filtering and Hough transform, and is quite robust if there are not too many stones on the board. Also, the colours of the stones can be decided quite well. The current system is only able to analyse still images, but extending the system to analyse videos of complete go games seems possible.

References

[Kälviäinen et al., 1995] Kälviäinen, H., Hirvonen, P., Xu, L., and Oja, E. (1995). Probabilistic and non-probabilistic Hough transforms: overview and comparisons. *Image and Vision Computing*, 13(4):239–315.

[Sonka et al., 1998] Sonka, M., Hlavac, V., and Boyle, R. (1998). *Image Processing, Analysis, and Machine Vision*. International Thomson Publishing Inc., 2nd. edition.