Go Board Reconstruction from a Single Image

Amy Zhang
ayzhang@stanford.edu
Link to project code:
https://drive.google.com/drive/folders/11p79o3kmwenHBj1TZNZJyLRtmLcKIX4G?usp=sharing

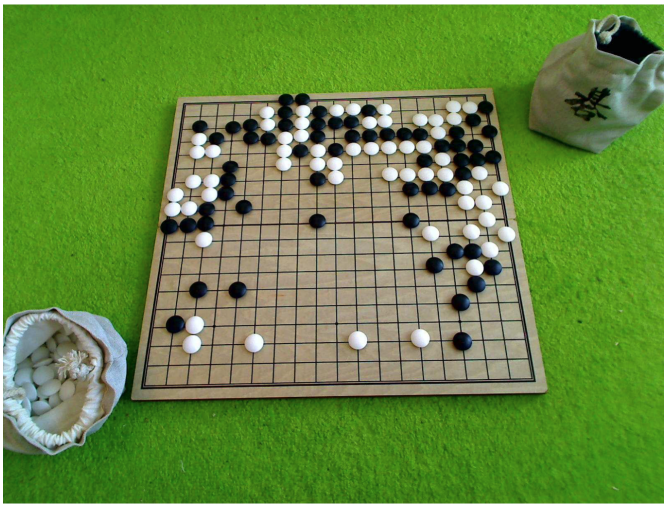

Figure 1. Traditional Go Game Played on a 19x19 Grid

Abstract

The game of Go is generally played on a 19x19 grid board and has the potential to involve up to 361 individually placed stones. In order to score a game of Go, each of these stones must be individually counted and treated using the scoring rules. This paper introduces an application of Computer Vision techniques to create a digital reconstruction of a Go board so that players can upload their boards to virtual scoring programs. The goal is to reconstruct both the game board and the stone placement from only a single input image of the Go board, instead of requiring a separate calibration image of an empty board. The performance of this approach is compared to the ground truth board and stone positions of seven input images. Analysis of these images shows that this approach is exceptionally accurate in reconstructing boards in ideal lighting conditions but not generalizable to boards in extreme lighting conditions.

1. Introduction

The game of Go is widely regarded as the oldest board game that is still played in the present day. The game is played using black and white stones on a grid board where the goal of the two players is to surround more territory than their opponent. The winner is determined after scoring the board. There are two main ways in which a Go board can be scored: area scoring and territory scoring. Both sets of scoring rules involve the two players manually counting their stones and captured spaces and removing dead stones. This process is generally quite time consuming, especially when playing on the standard 19x19 board.

This paper explores how to apply computer vision techniques to reconstruct a Go board. This digital reconstruction would then allow players to feed their board data into an automatic Go scoring program (like Crazy Stone [1]), instead of having to spend time manually scoring the game. The goal of this paper is to take an input image of a Go board (see example input in Figure 1) and simultaneously reconstruct the position of the grid and the stones. The reconstruction pipeline can be broken down into two parts: board recognition and stone recognition. Previous works on the topic of Go board reconstruction include work done by Musil [2], Srisuphab et al. [3], and Corsolini et al. [4]. All of these approaches, however, assume an input of at least two images: one of the empty board for grid detection and another of the board in the same position but now covered with stones. This paper will be adapting these approaches to a single image reconstruction approach.

2. Related Work

This section will focus mainly on the three works mentioned above: Tomas Musil's "Optical Game Position Recognition in the Board Game of Go", Corsolini et al.'s "A New Approach to an Old Problem: the Reconstruction of a Go Game through a Series of Photographs." and Srisuphab et al.'s "An Application for the Game of Go: Automatic Live Go Recording and Searchable Go Database".

The approach by Musil uses filtering, Hough transforms

and RANSAC to detect an empty grid and K-means clustering to detect stones. The implementation first converts the image to gray-scale and then uses a Laplace operator and high-pass filter to preprocess the image. The image is then passed through a Hough transform and another high-pass filter. Then RANSAC is used to find the two sets of parallel lines that determine the grid. Then RANSAC is used again to find the candidate diagonals of the grid which are then used to compute the positions of the corners of the grid. The candidate diagonals each generate a candidate board and then the best board is selected. Next is the stone reconstruction step. Since the intersection of the grid-lines are already known, this approach takes the average color in a 5x5 pixel square around each intersection and uses kMeans clustering to cluster the colors into 3 clusters (black, empty, and white). Then each stone is assigned to a cluster and their color is then known. This is the approach that will be used for our implementation of stone recognition as it takes into account variability in lighting conditions since it does not involve setting hard boundaries or RGB thresholds.

Srisuphab et al. use Canny edge detection, Hough transforms and rectification to detect the empty grid and simple RGB averages to detect stones. This approach first uses a Canny edge detector described by J.F. Canny [5]. Then a Hough transform is applied to better refine the detected edges. This results in well-defined lines; however, there may still be some errors in this reconstructed board. To fix these errors, this approach performs a rectification step where the hypothesized intersections are compared against a virtually generated board. To reconstruct the stones, Srisuphab et al. use a circular boundary around each intersection and calculate the average color value of each boundary. It then uses the HSV color model to determine whether each boundary is white, black or empty.

Corsolini et al. use Hough transforms to detect the empty grid and the difference between game images to detect stones. This approach uses two runs of Hough transforms and error checking to reconstruct the grid. This approach tracks a game starting from an image of an empty board and has input images as each stone is placed. It is therefore able to use a very quick approach of calculating the difference between consecutive images to reconstruct the stones.

Besides the academic literature surrounding the problem of Go board reconstruction, there are also public implementations such as Kifu Snap and Imago which are capable of Go board reconstruction [6] [7]. As of June 2020, Kifu Snap can be used online to reconstruct most single images of boards. Imago, the public implementation of Musil's approach, is capable of analyzing a whole game with a 76% success rate and takes 30 seconds per photo to perform the analysis [3]. A common next step from board recognition is tracking an entire game of Go from a video of the game board. Programs like Igoki and the implementation by Srisuphab et al. are capable of performing such video tracking.

3. Approach

3.1. Determining the Pipeline Structure

As mentioned in the introduction, the overall reconstruction pipeline can be broken down into two main pipelines: grid reconstruction and stone reconstruction. Given that the input image has both grid and stone data, there was flexibility in choosing which pipeline to begin with.

The process of first generating the stone reconstruction and then generating the grid reconstruction was explored first. An initial approach to this problem was to use circular Hough transforms and color thresholding to detect the circular stones on the board. The detected stones could then also potentially be used to aid in grid reconstruction since all the stones must be placed at the intersections of grid lines.

This approach was not successful due to several factors. The first factor was that the surface of the stones are non-lambertian and therefore, even with color thresholding, it was difficult for the circular Hough transform to correctly identify the boundaries of stones. The second factor was that the images of the Go boards often had very different intensities and warmth values. This made it difficult to generalize an approach that could always detect stones when for example, a white stone in more yellow lighting might blend in considerably with the wooden board. The third factor was the human error involved with placing the stones. Often the stones were placed significantly off-center from the actual intersection of grid lines and as a result, the idea that the stones could then be used to aid in grid reconstruction would not have been feasible. As a result of these factors, the final reconstruction pipeline first tackles grid line reconstruction and uses that to aid in stone reconstruction.

3.2. Overview

The reconstruction pipeline begins with an image preprocessing step where the image is thresholded and the actual game board is extracted from the image and rectified. Then the pipeline proceeds with grid reconstruction. The grid reconstruction is then used to perform the stone reconstruction step, as all of the stones must at least partially lay on an intersection of grid lines.

An important note is that this project is operating under the assumption that all grid lines are at least partially visible in the image, ie. there are no rows or columns that are entirely occluded by stones.

Figure 2. Example Input Image

3.3. Image Preprocessing

The dataset of images used in this project involve images of Go boards taken from differing angles and distances (see Figure 2). As a result, it was necessary to first extract out the board from the image and rectify it so that the following reconstruction pipelines could be more generalizable. This was accomplished by first manually selecting the four corners of the game board in the given image and using those four corners to compute a perspective transformation. This transformation was then applied to the area contained in the selected four points (ie. the un-rectified board). The result from this step is a square image of the rectified game board without any surrounding environment. (This step has been newly added after the final presentation.)

This rectified image then went through a thresholding step where any pixels above a set white color threshold were turned completely white and any pixels below a set black color threshold where turned completely black. These thresholds were manually tuned with the white color threshold being [160, 160, 160] and the black threshold being [50, 50, 50]. This step was necessary in order to decrease the variance of the differing shades of white and black in the image. As a result of this step, the stones and stone colors can be much more easily separated out in the later stone detection step. Furthermore, the dark grid lines are also more apparent for the grid detection step (see Figure 3).

3.4. Grid Reconstruction Approach

Before the image goes through the grid reconstruction pipeline, it must go through an extra step of processing where it is converted to grayscale, thresholded again and blurred. This intensity thresholding step removes any gray tones that are still present from the wooden board background by turning them white. This helps reduce noise for the subsequent line detection step. The intensity threshold was manually tuned to be 90. In the blurring step, the image is passed through a Gaussian filter which smooths the image and reduces any Gaussian noise. The width and height

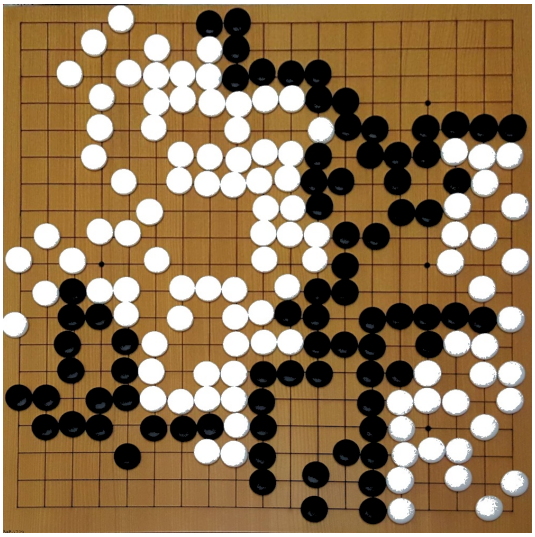

Figure 3. Example Input Image After Image Processing Step

of the Gaussian kernel were manually tuned to be 11x11.

The next step in the grid reconstruction pipeline involves detecting lines in the image. My original approach had used Hough transforms and probabilistic Hough transforms to detect the grid lines on the board; however, both of these approaches were too sensitive to the lines being partially occluded by stones. Furthermore, both of these approaches required the specification of a minimum line length which was hard to tune given the variety of different line lengths. I discovered that the `FastLineDetector` function from the CV2 library was much more successful at detecting grid lines and went with this approach instead. Before running this function, the image is first passed through a Laplacian edge detector, which uses the second order derivative of the pixel intensity to detect edges. These edges are then fed into the `FastLineDetector` function. This line detector uses Canny edge detection to detect lines. The first Canny hysteresis threshold was manually tuned to be 25 and the second Canny hysteresis threshold was manually tuned to be 50. These detected lines are then used to estimate the corners of the grid, which are used in the next step to fill in the missing coordinates of points defining clustered lines.

Figure 4 illustrates the example input image after this line detection step. Note that although many of the detected lines are from the actual grid lines, there are also many lines that are actually a part of the stones. These lines should not be counted in the grid reconstruction pipeline. To address this issue, it was decided that applying clustering to all of the horizontal and vertical detected lines would be the best approach. Clustering would allow us to find the 19 most dominating horizontal lines and 19 most dominating vertical lines which would serve as the grid lines for the board. This relies on a core assumption that the amount of detected lines from the grid lines outweigh the number of detected

lines from the stones.

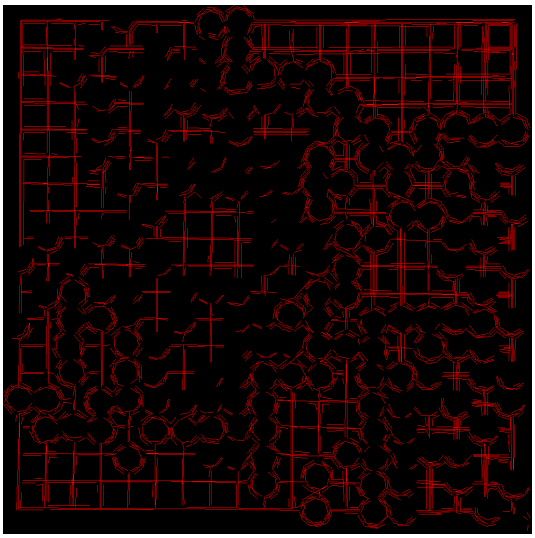

Figure 4. Example Input Image After FastLineDetector

Therefore, all of the horizontal and vertical lines are then extracted from these detected lines and each clustered into 19 clusters. These clusters serve as the 19 horizontal and 19 vertical lines that define the 19x19 grid board. More specifically, for horizontal lines we run `kMeans` clustering to define 19 clusters using the y values associated with the first point defining each extracted horizontal line. These 19 cluster centers are taken to be the y coordinates for each of the horizontal lines. The x coordinates of the horizontal lines are the x coordinates associated with the top right and top left corners calculated in the previous step. This gives us all of the 19 horizontal grid lines. For vertical lines, we run `kMeans` clustering to define 19 clusters using the x values associated with the first point defining each extracted vertical line. Similarly to the horizontal lines, these 19 cluster centers are taken to be the x coordinates of each of the vertical lines. The y coordinates of the vertical lines are the y coordinates of the top right and bottom right corners. This gives us all of the 19 vertical grid lines and the grid reconstruction is now complete as illustrated in Figure 5.

3.5. Stone Reconstruction Approach

The image after the initial image processing step (section 3.3) is used as the input to this recognition pipeline. This image is then further processed through a color thresholding step that takes any pixel that is neither black nor white and turns it blue. This ensures that any noise present in the board due to the wooden texture is eliminated and allows the stones to be easily distinguished from the board.

The next step involves looking at the pixels within a given radius surrounding each intersection and is a direct implementation of the stone recognition pipeline used by Musil [2]. More specifically, since we know that all valid

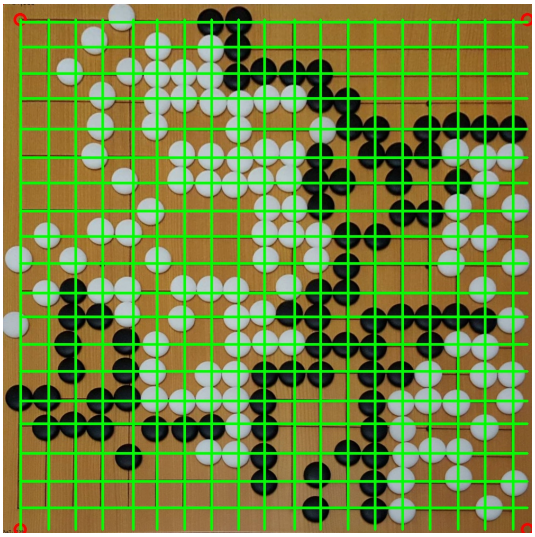

Figure 5. Example Input Image After Grid Reconstruction Pipeline

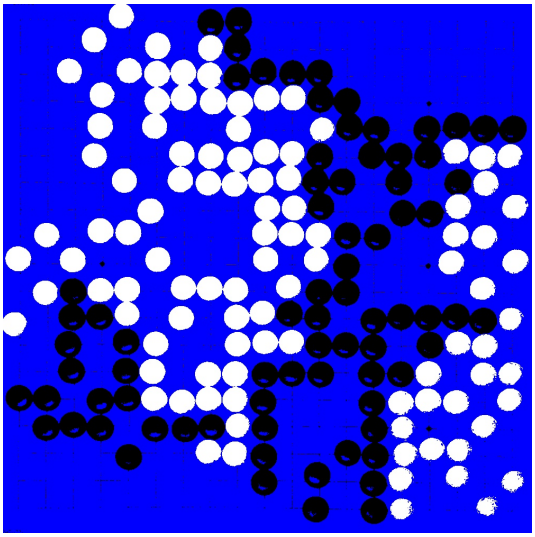

Figure 6. Example Input Image After Color Thresholding

Go stones must be placed at a grid line intersection, we can use the grid lines that were reconstructed in the previous step to define and search these intersections for stone data. Again, there is a certain element of human error that must be accounted for here. In searching the radius around an intersection it could be possible that a stone is just barely placed over an intersection in which case part of the region may be empty if the search radius is too large. If the search radius is too small, the stone estimates are more prone to possible noise from the board or even grid lines. The search radius was manually tuned to be 10 pixels. We then aggregate the pixel data within the 10 pixel radius surrounding each intersection and use `kMeans` clustering to cluster this pixel data into three clusters. These three clusters represent black intersection regions, white intersection regions and empty

intersection regions. From `kMeans` clustering, each intersection is now labeled according to its cluster center and the label (white, black or empty) can be taken to be the stone data of the board. A major advantage of using clustering is that no RBG or intensity thresholds had to be tuned so this approach is much more robust to noise and can handle a variety of lighting conditions. The stone recognition pipeline is complete, as we now know at which intersections each of the white stones and black stones are located. The resulting reconstruction is visualized in Figure 7.

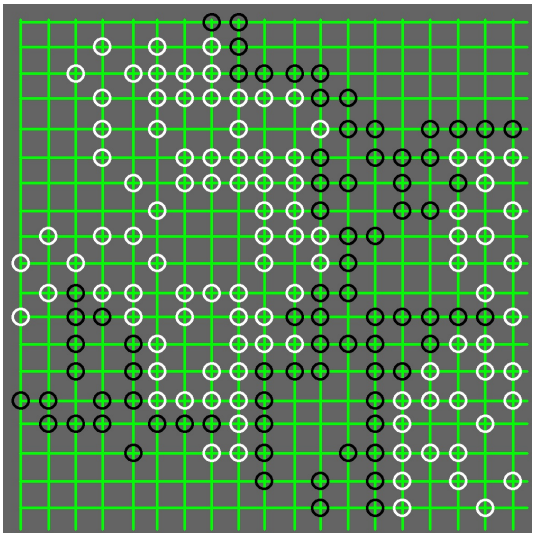

Figure 7. Final Visualized Reconstruction of Example Image

4. Experiment

In order to better mimic a wide range of input images, the dataset used in this experiment was pulled directly from Google Images. The dataset consists of seven images of entire Go boards with stones placed on them. As mentioned previously, a core assumption in this problem is that all grid lines are at least partially visible so this constraint played a factor in the dataset. Furthermore, any image with a partially occluded board could not be considered.

4.1. Results

The results from this project are compared to the ground truth grid lines and stone placement of the depicted game boards. This algorithm was able to successfully recover 100% of the grid structure for four out of the seven input images. Of these four images, two images had a 100% accurate stone reconstruction, one image had a 99.4% accurate stone reconstruction (see Figure 7) and the last image had a 50% accurate stone reconstruction.

The three images that had unsuccessful grid reconstructions were not counted in the stone reconstruction accuracy rate as the grid reconstruction was used to reconstruct the stone placement, so these stone results were not indicative of the actual accuracy of the stone recognition pipeline. This approach was unable to recover the grid for these three images due to several factors. The first factor is that even if a very small portion of the corner of the board is occluded in the image, the image will end up being warped in the initial rectification step. This will result in the horizontal and vertical lines being warped and these lines will therefore not be picked up by the line detectors (see Figure 8). The second factor is that some of the boards in the images were non-lambertian and therefore sometimes had portions that reflected light. This would result in the grid lines being hidden by the reflection and as a result, would go undetected by the line detectors (see Figure 9). We can see that this is indeed the issue in Figure 9 as the grid lines on the part of the board that is not reflecting light were successfully reconstructed. Figure 9 also illustrates the dependency that the grid lines have on the correct estimation of the corners of the grid.

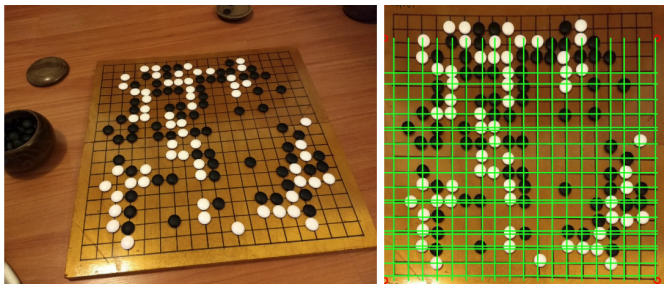

Figure 8. Occluded bottom right corner resulting in warped grid lines

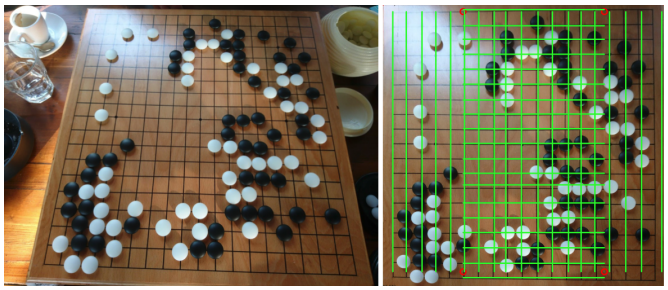

Figure 9. Reflective upper left corner of board resulting in undetected grid lines

The 99.4% stone reconstruction accuracy rate was due to the fact that there was one stone that was placed significantly off center from the intersection of the grid lines. As a result, the stone was assigned an empty value instead of a white value (which was the ground truth) (see Figure 10).

This is due to the human error involved with placing the stones but could be addressed in future work, potentially with the use of CNNs to learn stone placement patterns. The 50% stone reconstruction accuracy rate was due to the fact that this particular input image had very warm lighting. As a result, the white stones on the board looked very yellow and were not picked up in the white thresholding. The black stones, however, were successfully reconstructed. This could be addressed in future work with potentially using clustering to set the white and black thresholds (this would be more robust to different lighting situations similarly to the clustering application in regards to stone recognition).

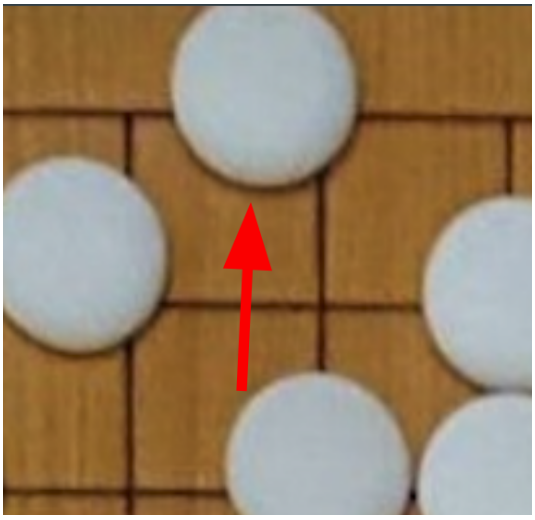

Figure 10. Upper stone placed significantly off center from intersection

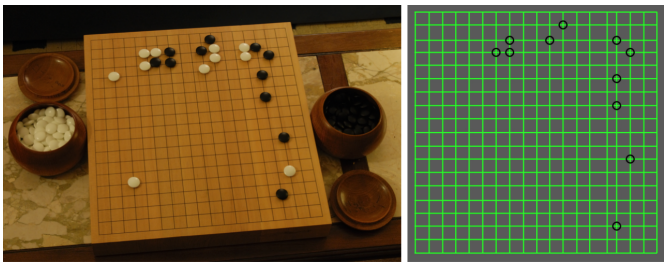

Figure 11. Warm lighting making yellow stones go undetected by white thresholding

5. Conclusion

As with any application of computer vision techniques, the issue of building in generalizability to extreme lighting conditions was the main obstacle to building a robust approach. Despite this obstacle, however, this approach of thresholding, line detection and clustering was very accurate for well-formed and well-lighted images. These concepts are excellent candidates to address Go board reconstruction and this approach would serve as a robust foundation for future work into the topic.

As mentioned above, there are two main areas for future work. The first being building a more robust approach to extreme lighting conditions. This could be done by using clustering to generate thresholds, instead of manually setting and tuning thresholds. Furthermore, the issue of grid lines being covered by reflective areas of the board could be addressed using simple inference methods that leverage the portion of the board that is not covered by a reflection. The second main area for future work is to address the issue of human error in placing the stones. When a board is manually scored, it is much easier to discern where a player might have intended to place his or her stone; however, when this task is left up to computer vision techniques there needs to be a more robust approach than just checking the radius around each intersection. This would be a great problem to solve using learning based approaches and could involve using CNNs to learn stone placement patterns.

References

[1] Coulom, Rémi. Crazy Stone, www.remi-coulom.fr/CrazyStone/.

[2] Musil, Tomas. “Optical Game Position Recognition in the Board Game of Go.” Charles University in Prague, 2014.

[3] A. Srisuphab, P. Silapachote, T. Chaivanichanan, W. Ratanapairojkul and W. Porncharoensub, ”An application for the game of Go: Automatic live Go recording and searchable Go database,” TENCON 2012 IEEE Region 10 Conference, Cebu, Philippines, 2012, pp. 1-6.

[4] Corsolini, Mario and A. Carta. “A New Approach to an Old Problem: The Reconstruction of a Go Game through a Series of Photographs.” ArXiv abs/1508.03269 (2015): n. pag.

[5] J. F. Canny, “A computational approach to edge detection,” IEEE Transaction on PAMI, vol. 8, no. 6, pp. 679–698, November 1986.

[6] Coulom, Rémi. Kifu-Snap: Automatic Go-Board Image Recognition, www.remi-coulom.fr/kifu-snap/.

[7] Musil, Tomáš. “Imago.” Tomáš Musil, tomasm.cz/imago.