BSc AI Thesis
Artificially Intelligent Go Annotation
Dynamic background correction and move detection

B. Stoeller, bstoelle@science.uva.nl

Supervised by L. Dorst

June 27th 2008

Abstract

AIGOA - Artificially Intelligent Go Annotation. Annotating a Go game using a webcam is not as trivial as it might seem. There are a lot of issues in the video frames that may influence the result. We assume that there are frames available containing only the Go board, the Go stones and a small part of the table at one or two of the edges of the Go board. That is due to the work of T. Kostelijk [1]. This report describes the methods that have been used to perform an accurate background estimation in order to determine what area of the board is covered by a hand, an arm or temporary heavy shadows or reflexions. Using this information, it is possible to create a video stream that contains only the board and the stones and not the hands and arms of the players. And subsequently the stones can be recognized and the moves can be annotated. The goal of AIGOA is to automatically annotate Go games from camera images in a robust way.

1 Introduction

One year ago a group of students of the University of Amsterdam [2] attempted to do the same. On of the main differences with their project is that they had a couple of manual thresholds that are now automated in AIGOA . The board finding in AIGOA is more precise, accurate and fully automatic. AIGOA has an occlusion detection which allows us to watch the Go board constantly and not only when it is completely visible. Therefore AIGOA does not need any Go logic as implemented by [2], although this might make the system more robust when Go games are played real fast or a lot of noise occurs in the video frame.
AIGOA is split up in to two parts. Part one is done by T. Kostelijk [1]. He will discuss developing a robust method to to calibrate a 19×19 Go board under general tournament conditions. This calibration includes detecting the location and orientation of the board and crosses (e.g. the locations where the stones will be expected). This report describes part two of the AIGOA project.

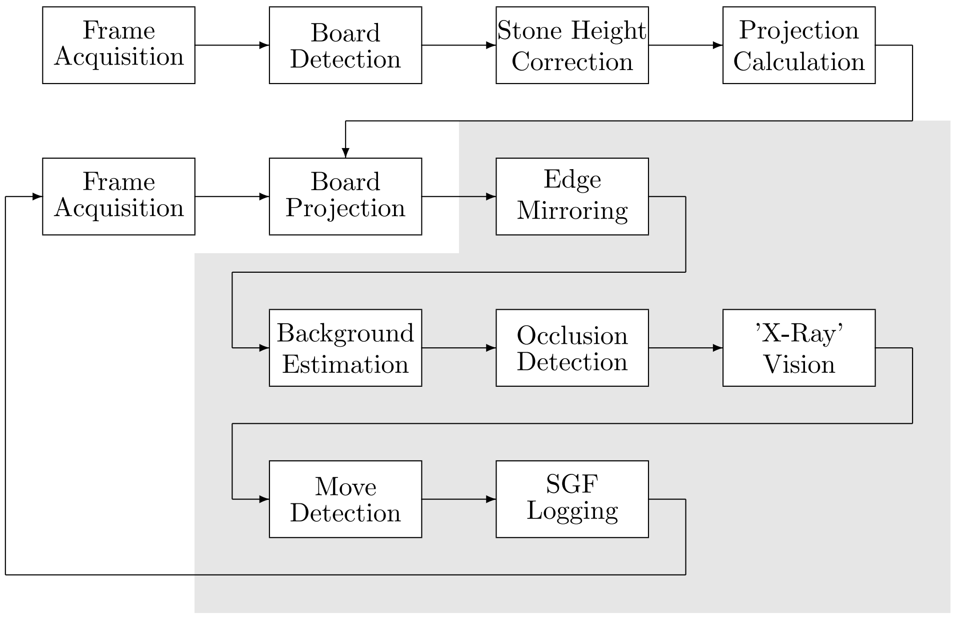


Figure 1: The modules in the gray area are implemented by B. Stoeller and will be discussed in this report, the rest is implemented and discussed by T. Kostelijk [1].

2 Structure

The structure of the complete system is shown in Figure 1. The modules are briefly described below and discussed in more detail further in this report.

Frame Aquisition This is a method to capture a frame from a video stream. This may be a webcam, an internet video stream, a stored video or whatever video stream one can imagine.

Board Detection Using an edge detector and a Hough lines algorithm the borders of the go board are found. When the borders are found it is possible to determine the position of the corners of the Go board, because these would be at the crossings of infinite line describing the borders.

Stone Height Correction Because we are not looking at the board from straight above, the sides of the stones are also visible. Projecting the video frame focusing on the board gives some errors. The stones seem to be moved up to half a board position or even further. Especially in the rear of the Go board. So focusing on a virtual plane slightly above the board at say $\frac{3}{4}$ times the stone height gives a better projection, thus stones can be detected more accurately.

Projection Calculation When the four corners of the virtual stone plane are found, it is possible to calculate a 3×3 projection matrix H that maps all the pixels in the original frame onto a new image called the board image.

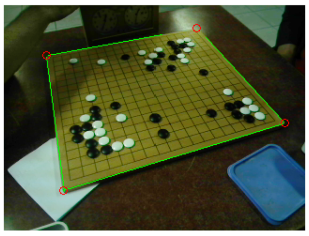


Figure 2: Board detection by T. Kostelijk

Board Projection Using a projection matrix H it is possible to extract the Go board from the video frame. This means all further calculations can be done on a square image containing only a part of the frame. This board image is always the same size regardless of the position, panning, tilting and zooming of the camera.

Edge Mirroring Because the virtual stone plane is projected instead of the Go board itself, a small stroke of the table might also be visible in the board image. To remove this stroke the edges around the board are cut of and filled like there were mirrors around the edges.

Background Estimation This is a system to estimate the background (e.i. the Go board and the stones on it) regardless of hands and arms that may be seen in the video stream. It updates constantly to take the changes in illuminance into account and to let the newly occurred stone absorb into the background.

Occlusion Detection When the background is known (i.e. estimated) it is possible to determine the occlusions by objects other than the go board and the stones using the difference between the estimated background and the last acquired board image. When a region is known to be occluded, the board and stones behind that object cant be seen. So the stone detection should not look at that part of the Go board.

'X-Ray' Vision Given the new board image and a description of the occluded region of the image, it is possible to create a video stream that displays only the Go board and the stones, but not the hands and arms that are blocking the view. The result looks like you are seeing through someones arm and hand, or 'X-ray' vision.

Move Detection Information about the positions where stones may occur is given by the Crossing Detector of T. Kostelijk. Combining this with the result of the X-Ray Viewer gives a probability for each crossing that there is a white stone, a black stone or no stone at all. Knowing this, it is possible to detect the introduction or removal of stones.

SGF Logging When a stone is placed on the board this means a move has been made and can be logged in an SGF file. The SGF format is a well known standard for several board games.

3 Edge Mirroring

Projecting the virtual stone plane onto the board image results in a small border around one or two of the edges of the Go board. To fix this a mirroring method is used. The board positions on the edge are constructed by replacing the outer half of the position with a mirrored image of the inner half. See Figure 3. This prevents the system to classify the edge as black or white stones when it is just the table it is watching.

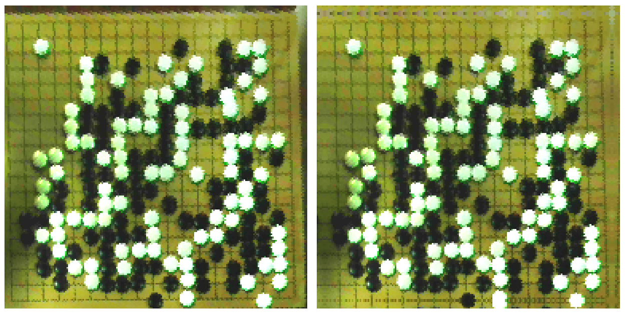

(a) Normal edges (b) Mirrored edges

Figure 3: The result of the edge mirror.

4 Background estimation

For the detection of occluding objects, it is necessary to perform an accurate background estimation. Comparing the background with a newly acquired board image makes it possible to determine occluding objects and subsequently remove the objects from the video stream. The frames on which this part of AIGOA works consist of board images only. Board images are the projected area of an original video stream that contain only the Go board. These are always square and objects outside the board area cannot be seen. These images are created by the Board Projection in the first part of AIGOA [1].

Stauffer and Grimson [3] described a very useful approach for background estimation in an environment that is constantly changing over time. Their adaptive background mixture model is used a the basis of our approach for the background estimation.
The main difference is that Stauffer and Grimson use their system in an outdoor environment with moving objects that should be seen as background like waving trees and the specularities on the surface of water, while we use it in an indoor situation focusing only on the Go board.

The main similarities are that there are also objects being introduced and removed from the scene during a Go game and that we are facing changes in global lighting over time.

Although one might assume that calculating the background automatically results in a video stream containing only the Go board and the stones, this is certainly not the case. More details on this issue will follow in Section 5.

4.1 Regional illuminance

Due to fuzzy cast shadows and reflections the regional illuminance of the board constantly changes over time. The global lighting also changes over time due to clouds drifting over, sunrise, sunset etc. To correct this, the red, green and blue channel are summed up into a new 'image'. This summed image is resized to an image of 4×4 pixels and then back to the normal size using anti-aliasing and bilinear interpolation. This is a very fast and easy way to create a blurry image representing the regional illuminance of the image. Finally the original image is divided by the illuminance image. This results in an image containing less variance in illuminance. See Figure 4. Several approaches have been tested [4] and this one seemed to be the best method that is fast enough to use in Matlab.

(a) Board image (b) Illuminance filter (c) Normalized image

Figure 4: Illuminance normalization

4.2 Adaptive mixture of gaussians

Stauffer and Grimson [3] described an iterative method that uses a mixture of multiple normal distributions (approximately 3 to 5) per pixel to model the background. Each normal distribution is described by a Gaussian function $\eta(X, \mu, \Sigma)$ and a weight ω. The red, green and blue pixel values are assumed to be independent. This is certainly not the case, but experiments show that the effect of a real covariance matrix is minimal but much more computational. Stauffer and Grimson [3] propose to define the standard deviation as $\Sigma = \sigma^2\mathbf{I}$. Our approach uses this same simplification and therefore σ is directly used in our formula's and calculations.

Each Gaussian is updated every iteration. When the newly retrieved pixel color is close to the mean μ of the Gaussian, the standard deviation σ will become smaller meaning that the distribution is more accurate. And vice versa σ will get larger when the new value is further away from the mean (Equation 2). The mean μ itself also adapts to the new value by moving a little bit toward it (Equation 3). In fact this little bit is defined by ρ which is an interpretation of the learning rate α combined with distance between the new image color and the estimated background color (Equation 4).

Because of the fact that the Go board is a reasonably static background there is no need for a very adaptive background estimation as in the examples with waving trees and such. The only dynamic part is the introduction and removal of stones and the movement of occluding objects that should never be seen as background (i.e. hands and arms). Therefore a mixture of only two Gaussians ($n = 2$) is used to model the background.

Knowing all this, the equation of $\eta(X, \mu, \Sigma)$ Stauffer and Grimson can be simplified to Equation 1 which is used in AIGOA an which is referred to as $\mathcal{N}(X|\mu, \sigma)$.

$$\mathcal{N}(X|\mu,\sigma) = \frac{1}{2\pi\sigma} e^{\frac{-1}{2\sigma^2}(X-\mu)^T(X-\mu)} \tag{1}$$

$$\sigma_t = \sqrt{\sigma_{t-1}^2 + \rho((X_t - \mu_t)^T(X_t - \mu_t) - \sigma_{t-1}^2)} \tag{2}$$

$$\mu_t = \mu_{t-1} + \rho(X_t - \mu_{t-1}) \tag{3}$$

$$\rho = \alpha\mathcal{N}(X|\mu,\sigma) \tag{4}$$

The Gaussians are initialized at the first acquired frame. This feels intuitively a bit tricky because it is possible that occluding objects are present in the first frame and we would not want them to be a model for our background. We could also start wit a completely black, gray or white image. But this would mean our background estimation has even more errors in the beginning. Presuming that the board image is presented by the first part of AIGOA [1] we may assume that the first frame is not occluded since the Board Detection requires the Go board to be clear during the initialization.

As mentioned before, each Gaussian has a weight. The weight determines what portion of the data is accounted for by the Gaussian. These weights are normalized per pixel so they add up to one. They are used to determine the probability that a pixel belongs to the background. See Equation 5. A pixel is classified as background when $P(X_t) > Q$, where Q is a threshold whose value depends on the amount of noise in the video stream.

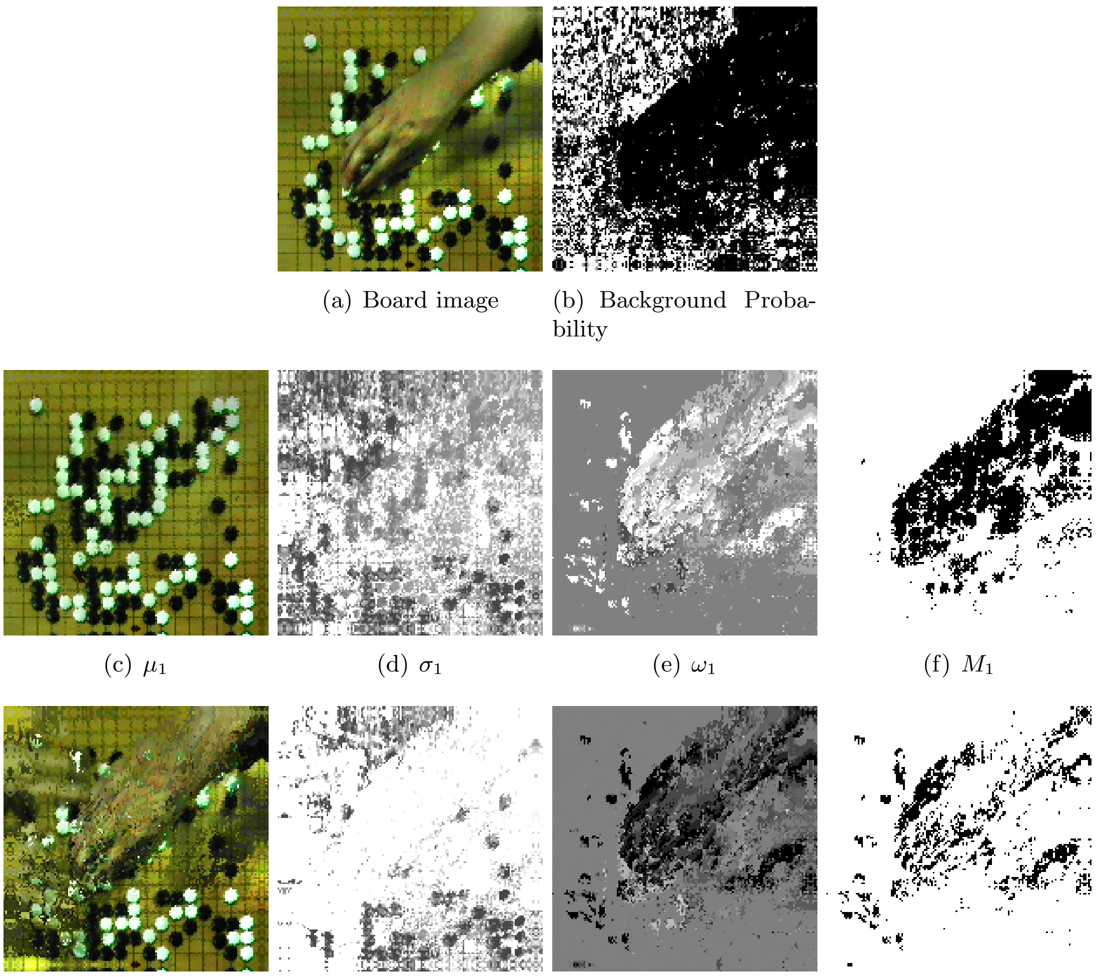
$$P(X_t) = \omega_{1,t} \times \mathcal{N}(X_t|\mu_{1,t}, \sigma_{1,t}) \tag{5}$$

A pixel is said to match the Gaussian when the difference between the pixel color and one of the two Gaussians of that pixel is smaller then 2 times the standard deviation. In this case M is *true*, otherwise it is $false$. Note that to match the Gaussian is essentially something different than to to be classified as background. The update rules described above are only applied to the Gaussains that the pixel matches.

$$M = (|X - \mu| < 2\sigma) \tag{6}$$

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t}) \tag{7}$$

The weights are updated each iteration dependent on whether the pixel matched the Gaussian or not. When a pixel does not match one of the two Gaussians, the Gaussian with the smallest weight will be replaced by a new Gaussian, centered (μ)on the pixel color, with a large standard deviation (σ) and a weight (ω) of 0.
Although the mixture of Gaussians is described per pixel of course in the Matlab implementation of AIGOA these calculations occur for all the pixels at once. For the convenience of the reader, a table with the used parameters and variables is shown in table 1 (Appendix A).



(a) Board image (b) Background Probability

(c) μ_1 (d) σ_1 (e) ω_1 (f) M_1

(g) μ_2 (h) σ_2 (i) ω_2 (j) M_2

Figure 5: Background estimation results

As you can see in Figure 5 there is a lot of noise. This is because the video that is used to test AIGOA was shot at relatively bad lighting and it was stored using PNG compression. The expectation is that the results are better using non compressed video stream under the right lighting conditions. You can also see that the color of an arm is almost the same as the color of Go board. This makes it even more difficult to distinguish foreground from background. But maybe even these noisy images are enough to estimate detect the occluded part of the board... This will be discussed in the next section.

5 Occlusion detection

When we take a look at Figure 5.c it appears that this image shows only the Go board and the stones, but not the occluding object. Yet this is not entirely true. Because of the necessary learning rate the image in Figure 5.c is a bit delayed but we don't know exactly how much, because that depends on several things like the lighting, the video noise, the previous color of the pixel, occluding objects etc.

Therefore we do use the image created in the best Gaussians (Figure 5.c) and compare it with the new frame by calculating the Euclidean distance between the estimated background and each pixel in the new frame. A threshold is set to determine for each pixel if it is background or foreground. This results in a mask like Figure 6.c. When a dilation is followed by en erosion (with a slightly larger structuring element) most of the false negative and false positives are removed. And there is even an extra margin around the occluding object to make sure we have the complete object and its cast shadow are covered. (Figure 6.d) But one of the most important advantages is that a stone that has been introduced is so small that it will disappear from the mask during the dilation of the background.

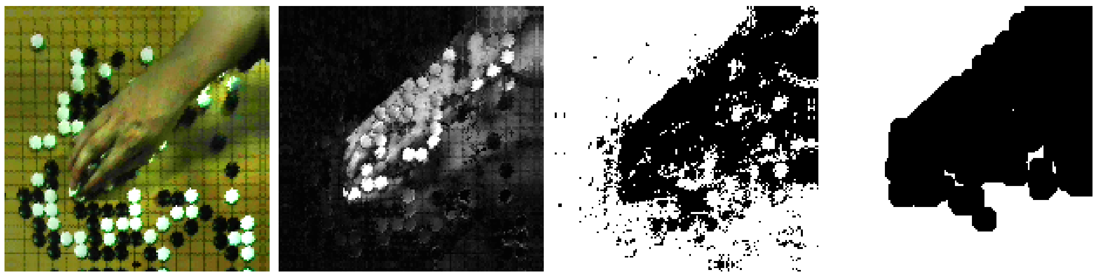

(a) Board image (b) Euclidean distance to the background (c) Mask using a threshold (d) After dilation and erosion

Figure 6: Occlusion detection results

Thus when a player holds a stone and reaches out to place it, his arm, hand an the stone are masked. But from the moment the player places the stone on the board and pulls back his hand, only his arm and hand are masked, but not the stone. The next session will discuss why this is useful.

6 'X-Ray' vision

The X-ray vision initializes on the first frame it receives. Just like the background estimator. The X-ray vision updates its vision every iteration using to the newest board image and the the occlusion mask. It updates its vision only at the regions that are not occluded. Resulting in a video stream that does not contain any hands, arms or other obstacles. This actually means that the region that is occluded at time t will be filled up with data from the history. Now only the Go board and small obstacles that are introduced to its surface, like Go stones appear in the pseudo live stream. This is a good basis for the move detection. (Figure 7);

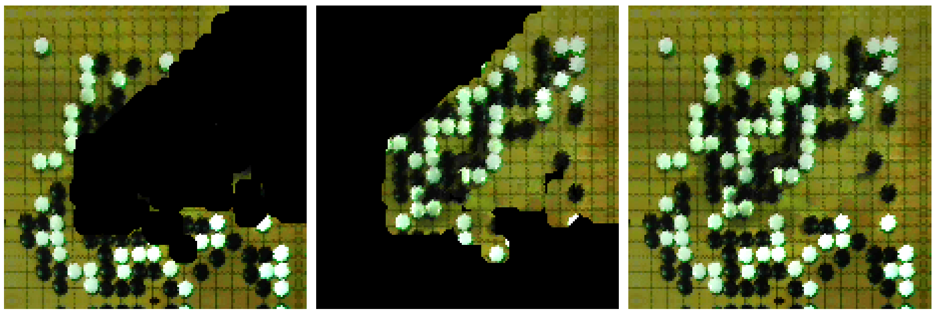

(a) New board image with masked occlusion (b) History at unknown region (c) Combination of new and known information

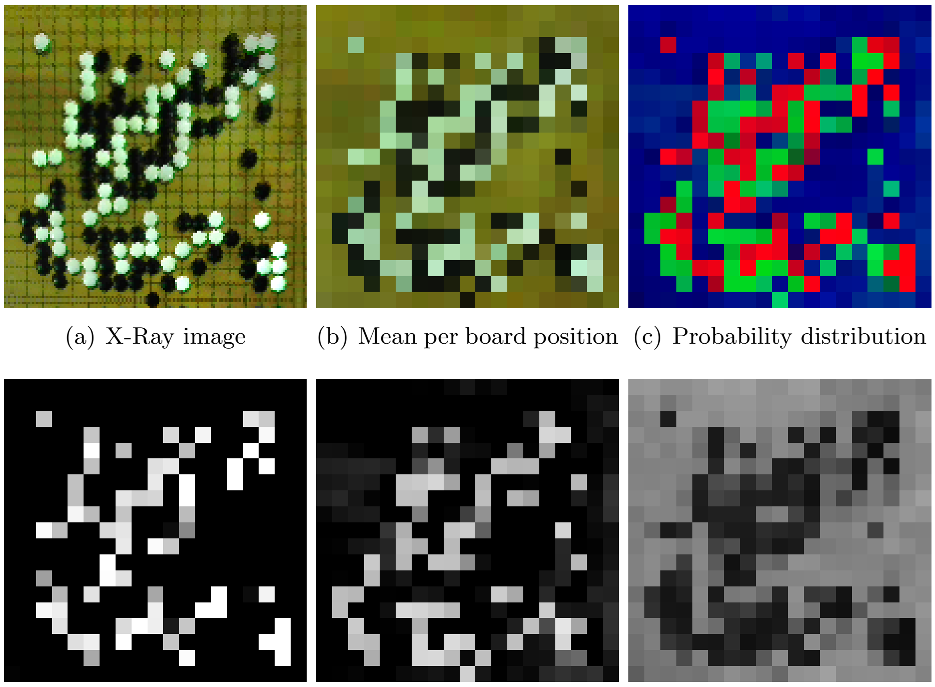
Figure 7: Combining the new data with the data already known to create X-Ray vision.

7 Move detection

The goal of AIGOA is to automatically annotate Go games from camera images in a robust way. Using the results of the X-Ray vision, it is possible to detect the stones at the board positions. Nevertheless there is still a possibility that some noise will occur in the X-Ray vision which might result in incorrect pixel values. It is also possible that the stones are not exactly placed on the crossings. At the end of this section some possible refinements will be discussed.

7.1 A priori stone detection

There are three a priori estimations made. One for the white stones (8), one for the black stones (9) and one for the board (10). There are all based on the RGB values of the means of the board positions. These means are calculated by centering a normalized Gaussian filter on each of the board position and add up the values. This effectively means that the pixels at the center of a board position (i.e. the crossings) has more influence on the mean then the pixels at the border of that particular board position. This is because the center of each board position is expected to be the color of a stone (if there is one) even if the stone is not exactly on the crossing.


(a) X-Ray image (b) Mean per board position (c) Probability distribution

(d) P_{white} (e) P_{black} (f) P_{stone}

Figure 8: Occlusion detection results

$$P_{white} = 2 \times \left(B_i, -\overline{B}\right) \tag{8}$$

$$P_{black} = 1 - \frac{G_i}{\overline{G}} \tag{9}$$

$$P_{board} = \frac{\sqrt{(R_i - B_i)^2 + (G_i - B_i)^2}}{R_i + G_i + B_i} \tag{10}$$

Where R_i is the red channel of the mean color at board position i and $\overline{R}$ is the mean of the red channel of all the pixels in the board image. In the same way G and B are respectively the green and blue values.

8 Future work

8.1 Using a history for move detection

There was not enough time to implement a more robust stone estimator. But one approach might be to keep track of an history containing the possibility for each board position to be a white stone, a black stone or an empty board position. Looking at the difference with the recent history could give a better estimation of the last move that has been made.

8.2 Using Go logic for move detection

It would also be a good idea to use some Go logic in case the system is not completely certain. For example when it registers that two black stones appeared right next to each other at the same time, it is likely that only one stone is placed a bit sloppy, at the edge of two board positions. By checking which of the two board positions has the highest probability it should be possible to eliminate the wrong detected move. The system should know these kind of things because it is not possible that two black stones appear at the same time. The Go logic of [2] might be a good method.

8.3 SGF logging

When the moves are detected, they should be logged in a well known format like SGF, which is used for many board games including Go. There is a lot of Go software available that can read and write SGF files. It is an ASCII based format so it is possible to create SGF files with Matlab or any programming language.

References

[1] T. Kostelijk. Aigoa - seeing the board. 2008.

[2] J. van Velzen C. Walraven J. de Groot, G. Molenaar. Go-viewer. 2007.

[3] W.E.L. Grimson C. Stauffer. Adaptive background mixture models for real-time tracking. 1998.

[4] Th. Gevers. Color feature detection: An overview. 2006.

Appendices

A Parameters and Variables

α	Learning rate	$[0,1]$
e	Natural logarithmic number	$2.718281828459045\ldots$
H	Projection Matrix	$\begin{bmatrix} \mathbb{R} & \mathbb{R} & \mathbb{R} \\ \mathbb{R} & \mathbb{R} & \mathbb{R} \\ \mathbb{R} & \mathbb{R} & \mathbb{R} \end{bmatrix}$
M	Match	$\{0,1\}$
μ	Mean of the distribution	$\begin{pmatrix} [0,1] \\ [0,1] \\ [0,1] \end{pmatrix}$
ω	Weight of a distribution	$[0,1]$
n	Number of Gaussians	2
η	The Gaussian distribution defined by Stauffer and Grimson [3]	$\langle 0,\infty]$
$\mathcal{N}$	The Gaussian distribution defined in this report	$\langle 0,\infty]$
σ	Standard deviation of the distribution	$\langle 0,\infty]$
Σ	Covariance matrix of the distribution	$\begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma \end{bmatrix}$
t	Time step	$\mathbb{N}$
X	Pixel value	$\begin{pmatrix} [0,1] \\ [0,1] \\ [0,1] \end{pmatrix}$

Table 1: Parameters and variables