

Bachelor Thesis Presentation

Implementation of an Android app for the recording of Go games
by tracking their state

Tilman ADLER

June 1, 2015

Pattern Recognition Lab (CS 5)



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



Motivation

Detector Pipeline

- Overview

- Preprocessing

- Detection of Lines and Pieces

- Extrapolating the Board

- Postprocessing

Evaluation

- Image Set

- Detection Quality

- Speed

Conclusion

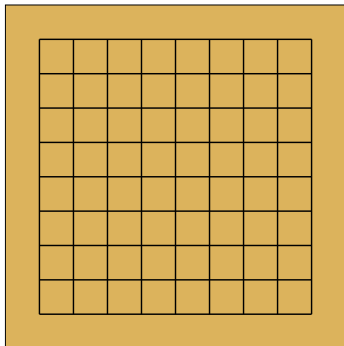
- Our Contribution

- Future Work

- Live Presentation

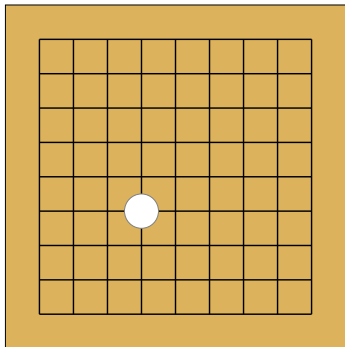
A Really Short Introduction to Go

- This is a 9x9 Go board (there's also 11x11, 13x13 and 19x19)



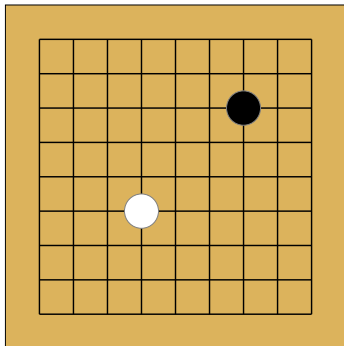
A Really Short Introduction to Go

- This is a 9x9 Go board (there's also 11x11, 13x13 and 19x19)
- Players alternately put pieces on the intersections



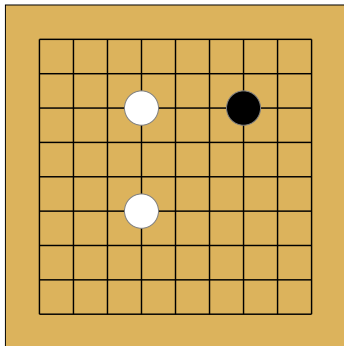
A Really Short Introduction to Go

- This is a 9x9 Go board (there's also 11x11, 13x13 and 19x19)
- Players alternately put pieces on the intersections



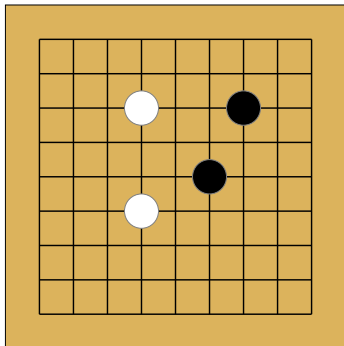
A Really Short Introduction to Go

- This is a 9x9 Go board (there's also 11x11, 13x13 and 19x19)
- Players alternately put pieces on the intersections



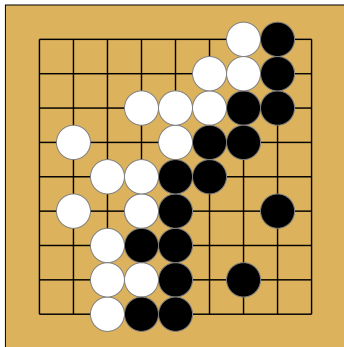
A Really Short Introduction to Go

- This is a 9x9 Go board (there's also 11x11, 13x13 and 19x19)
- Players alternately put pieces on the intersections



A Really Short Introduction to Go

- This is a 9x9 Go board (there's also 11x11, 13x13 and 19x19)
- Players alternately put pieces on the intersections
- The player with the most area wins



Reasons to Record Go Games

The famous Ear-reddening Game of 1846

- This is the game after the 25th move,...

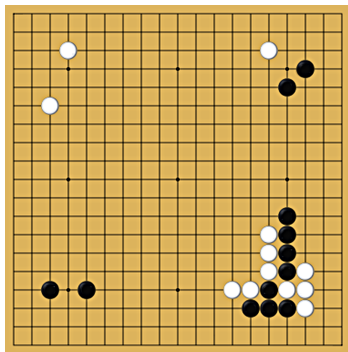


Image CC-BY-SA 3.0 by Wikipedia
http://en.wikipedia.org/w/index.php?title=Ear-reddening_game&oldid=640966258

Reasons to Record Go Games

The famous Ear-reddening Game of 1846

- This is the game after the 25th move,...
- when black made a mistake, that put white in the lead.

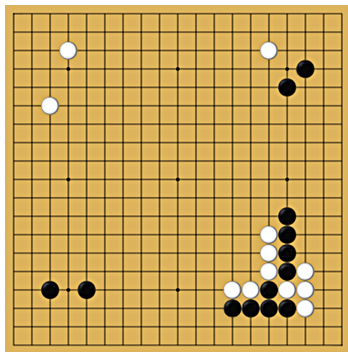


Image CC-BY-SA 3.0 by Wikipedia
http://en.wikipedia.org/w/index.php?title=Ear-reddening_game&oldid=640966258

Reasons to Record Go Games

The famous Ear-reddening Game of 1846

- This is the game after the 25th move,...
- when black made a mistake, that put white in the lead.
- The game went on to move 126...

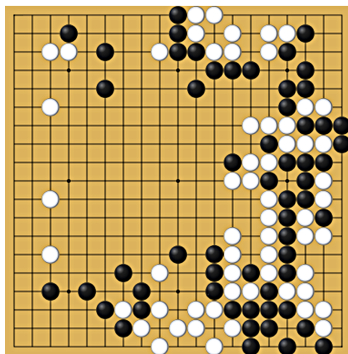


Image CC-BY-SA 3.0 by Wikipedia
http://en.wikipedia.org/w/index.php?title=Ear-reddening_game&oldid=640966258

Reasons to Record Go Games

The famous Ear-reddening Game of 1846

- This is the game after the 25th move,...
- when black made a mistake, that put white in the lead.
- The game went on to move 126...
- when this ingenious move turned the game and let the white player's ears blush. No one but the players noticed what a good move it was.

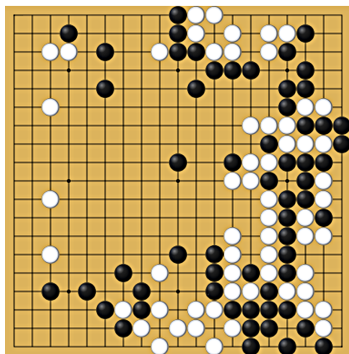


Image CC-BY-SA 3.0 by Wikipedia
http://en.wikipedia.org/w/index.php?title=Ear-reddening_game&oldid=640966258



Motivation

Detector Pipeline

- Overview

- Preprocessing

- Detection of Lines and Pieces

- Extrapolating the Board

- Postprocessing

Evaluation

- Image Set

- Detection Quality

- Speed

Conclusion

- Our Contribution

- Future Work

- Live Presentation

Overview

Most of the related work uses only line detection to find the board
⇒ This is problematic for mobile applications

Our approach is to

- use line detection for finding intersections
- detect pieces, too, for intersections
- extrapolate the board from both
- classify intersections

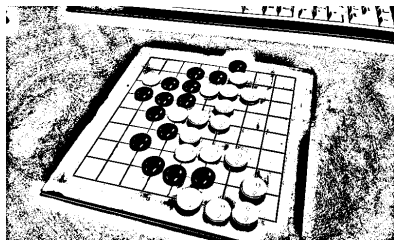
Preprocessing

Segmenting the board from the background

Segmenting the board from the background is useful for

- removal of noisy backgrounds
- improving detection speed

We use an adaptive threshold and connected-component analysis for this task



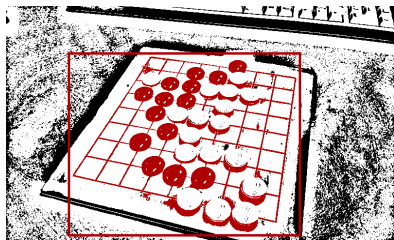
Preprocessing

Segmenting the board from the background

Segmenting the board from the background is useful for

- removal of noisy backgrounds
- improving detection speed

We use an adaptive threshold and connected-component analysis for this task



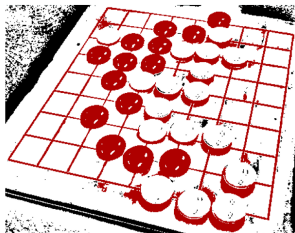
Preprocessing

Segmenting the board from the background

Segmenting the board from the background is useful for

- removal of noisy backgrounds
- improving detection speed

We use an adaptive threshold and connected-component analysis for this task

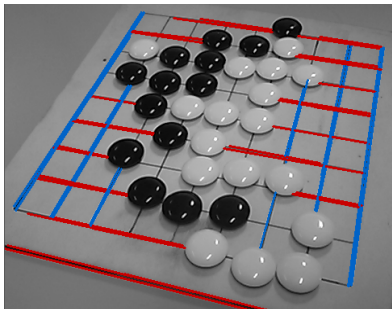


Detection of Lines

using Hough Lines Transformation

This approach is pretty straight forward:

- detect lines
- classify as horizontal/vertical
- intersect each horizontal with each vertical
- remove duplicates

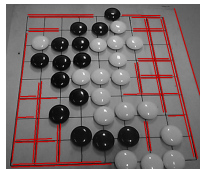
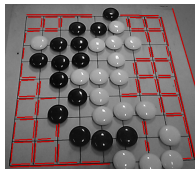
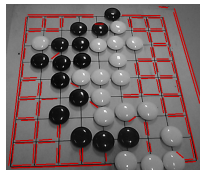
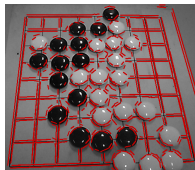


Detection of Lines

using the Line Segment Detector

Additional steps for *LSD*
detector

- filter short lines
- filter lines without parallels
- stitch lines



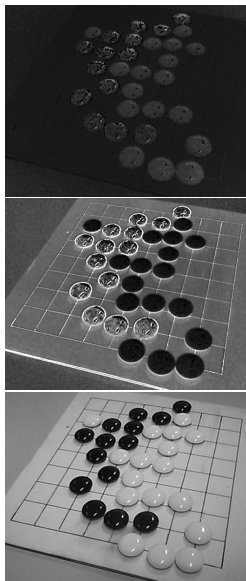
Detection of Pieces

Thresholding

To segment the tokens from the background we use thresholding

White pieces are a problem due to color aberration

⇒ Threshold in HSV space, use combined results from saturation and hue channel



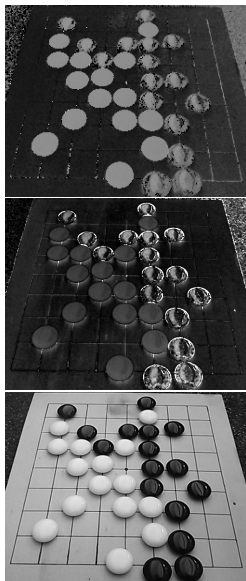
Detection of Pieces

Thresholding

To segment the tokens from the background we use thresholding

White pieces are a problem due to color aberration

⇒ Threshold in HSV space, use combined results from saturation and hue channel



Detection of Pieces

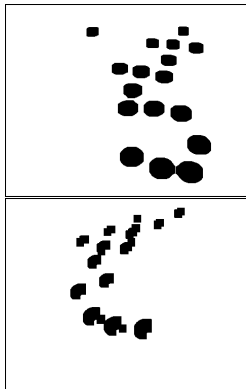
Thresholding

To segment the tokens from the background we use thresholding

White pieces are a problem due to color aberration

⇒ Threshold in HSV space, use combined results from saturation and hue channel

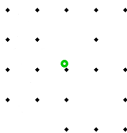
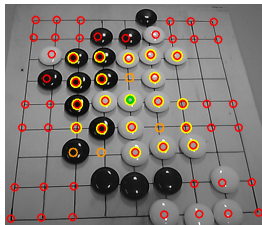
To detect the pieces we use *Hough Transformation* or fit rectangles around the blobs and filter for squares



Extrapolating the Board

We finally fill the intersections by

- rotating intersections to be horizontal
- selecting 20 around the center
- building a model from them
- detecting orientation of model with RANSAC
- using orientation to extrapolate board
- using detected intersections to refine extrapolation





Postprocessing

We try to filter out invalid detection results. We consider results invalid if one intersection is

- outside the image
- or closer than 5px to each other.

We undo the cropping from the preprocessing step.

We smooth results over time: only if a token was detected in 5 of the last 10 frames we count it as accurate.



Motivation

Detector Pipeline

Overview

Preprocessing

Detection of Lines and Pieces

Extrapolating the Board

Postprocessing

Evaluation

Image Set

Detection Quality

Speed

Conclusion

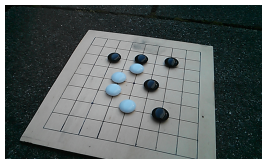
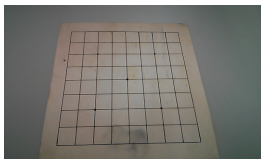
Our Contribution

Future Work

Live Presentation

Image Set

- We took 101 photos with different lighting and backgrounds
- 70 were used for empirical parameter optimization; 31 for testing



Detection Quality

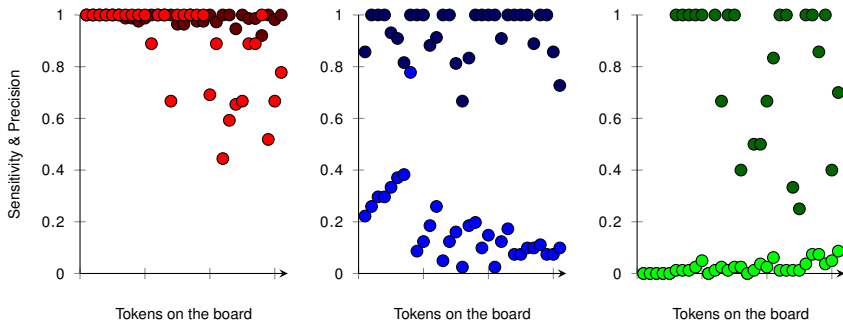
Line detection algorithms

		Hough	LSD	FAST
Empty board	Sensitivity	100%	30.3%	0.685%
	Precision	99.4%	94.2%	83.3%
Some pieces (7-13)	Sensitivity	95.6%	11.4%	1.97%
	Precision	98.6%	91.2%	53.3%
Many pieces (27-34)	Sensitivity	58.6%	9.02%	3.80%
	Precision	95.5%	94.0%	68.8%

Sensitivity and precision on our test set. We evaluated correct ($\leq 15\text{px}$ off) intersections.

Detection Quality

Line detection algorithms



Sensitivity (light color filled circles) and precision (dark color filled circles) per image in our test set.

Detection Quality

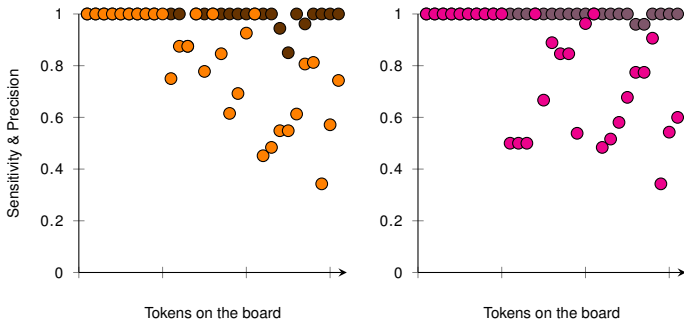
Piece detection algorithms

		Contour	HOUGH
No pieces	Sensitivity	N/A	N/A
	Precision	1 false positive	N/A
Some pieces (7-13)	Sensitivity	86.7%	75.90%
	Precision	98.6%	100%
Many pieces (27-34)	Sensitivity	40.0%	70.6%
	Precision	99.2%	99.2%
Total	Sensitivity	71.6%	71.6%
	Precision	97.8%	99.3%

Sensitivity and precision on our test set. We evaluated correct ($\leq 15\text{px}$ off) piece location and roughly correct piece size.

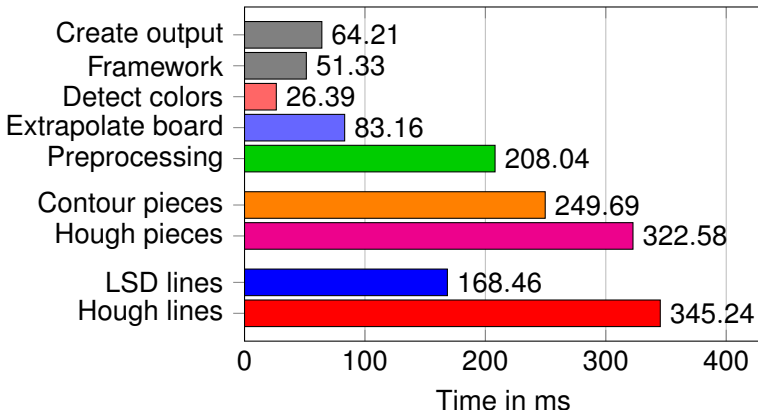
Detection Quality

Piece detection algorithms



Sensitivity (light color filled circles) and precision (dark color filled circles) per image in our test set.

Speed



Average duration of parts of the pipeline over 100 iterations on a
Nexus 4



Overall Quality

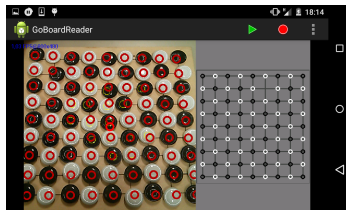
Accuracy of our final detector on our image sets

	Trainingset	Testset
Number of images	70	31
Discarded results	5	2
Accuracy localizing	99.1% (5217/5265)	100% (2349/2349)
Accuracy classifying	98.0% (5112/5217)	98.6% (2317/2349)
Accuracy for both	97.1% (5112/5265)	98.6% (2317/2349)

Similar quality is shown in tests where real games were played and recorded.

Our Contribution

- We present a reliable method for the detection of Go boards
- Our method is fast enough to be used on smartphones
- We can detect boards with dense token distribution
- We showcase a possible approach for an end user product





Future Work

Possible areas of future work could be:

- extending the approach to a 19x19 grid or grids of arbitrary size
- autonomously detecting the center of the board
- improving correction of specular highlights



Live presentation

Because seeing is believing

If there is still time...



Thank you for your attention.

Questions?