# Augmented Reality Tic Tac Toe

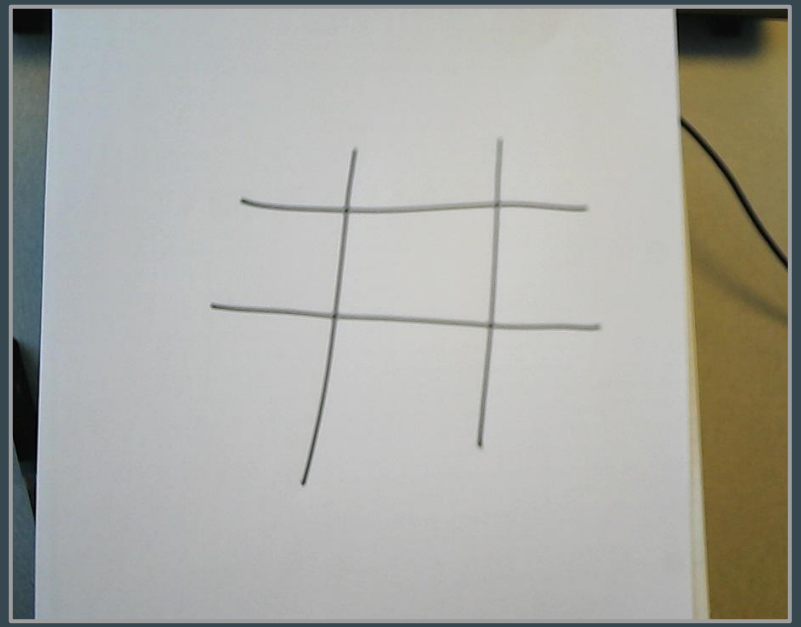
Nikesh Muthukrishnan
Ray Wen

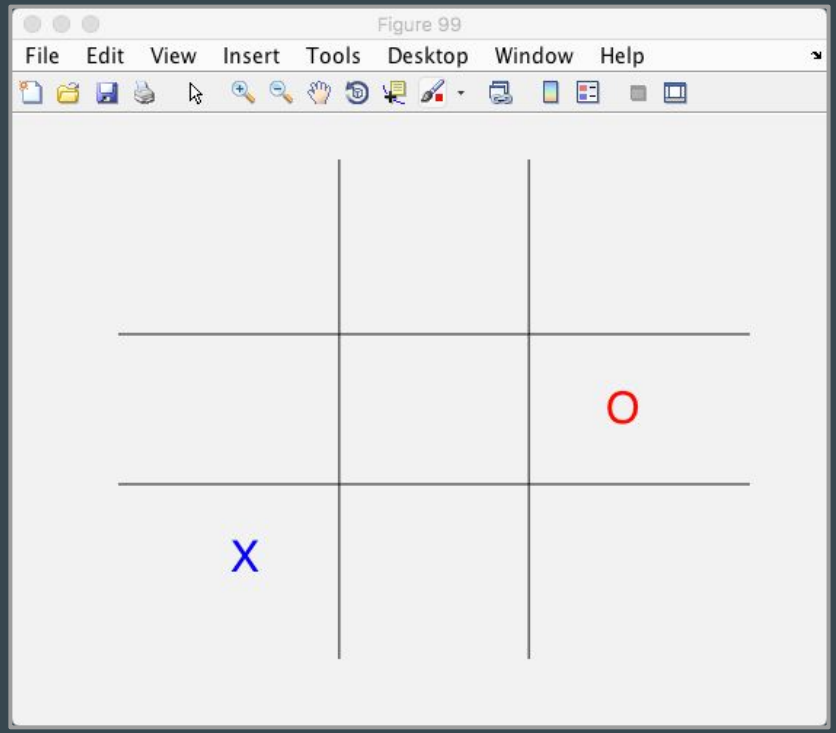
# Presentation Outline

- Background: Augmented Reality
- Our Algorithm - Theory & Results
- Limitations of Our Algorithm (Noisy Images & Filtering)

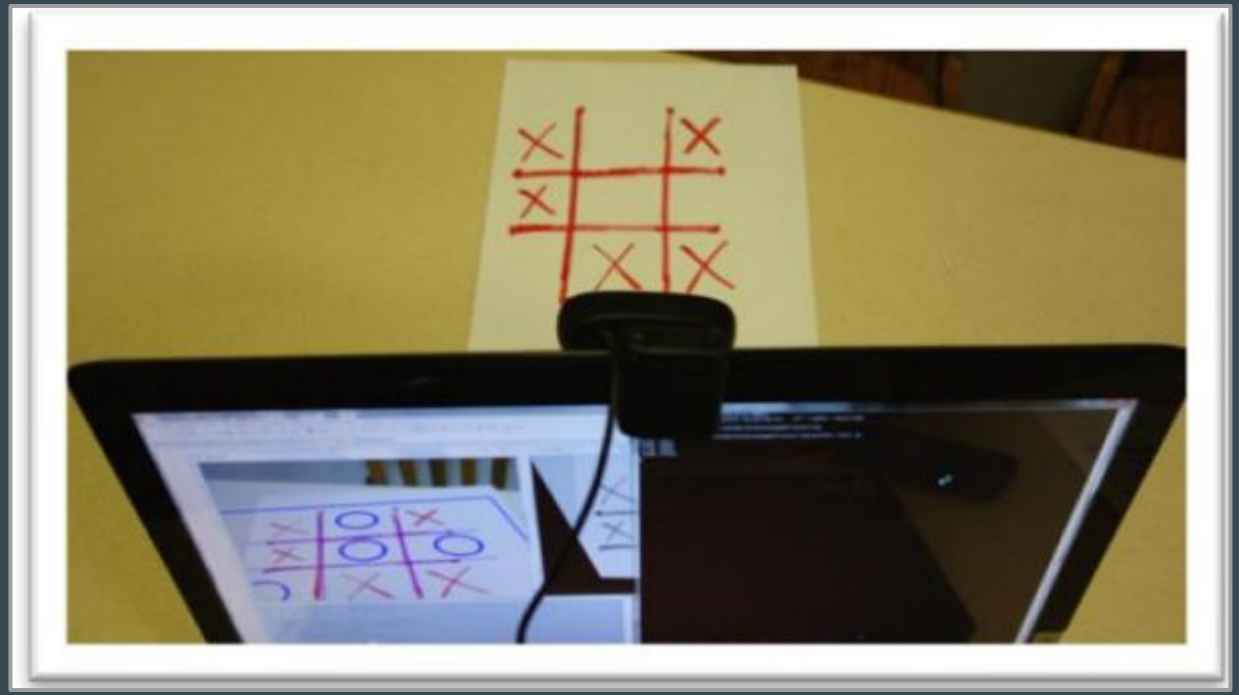
# Augmented Reality = CG Image + Real World

# Augmented Reality Tic-tac-toe: play with the computer





# Prior work: Maguire and Saltzmann



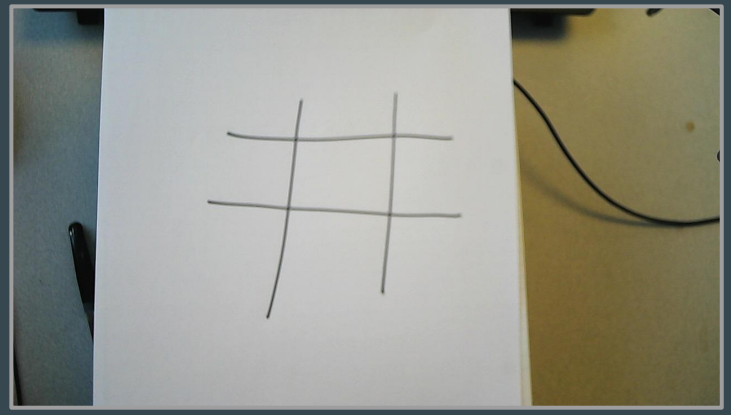
- Only detecting ‘X’
- Relatively optimal setup
- OpenCV implementation

# Our Algorithm

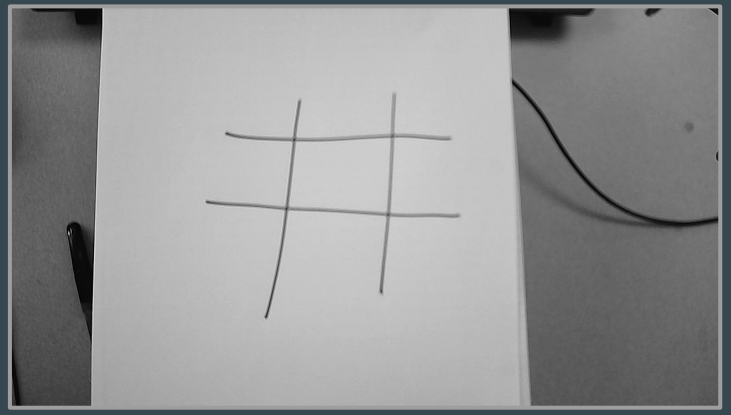
Webcam and TicTacToe Board Set up

- **GetPaper** - Focus Images on TicTacToe board
- **GetBoardDimension** - Extract center and x,y thresholds
- **CaptureMove** - Find new move and position according to thresholds
- **ReturnComputerMove** - Decide and illustrate a return move

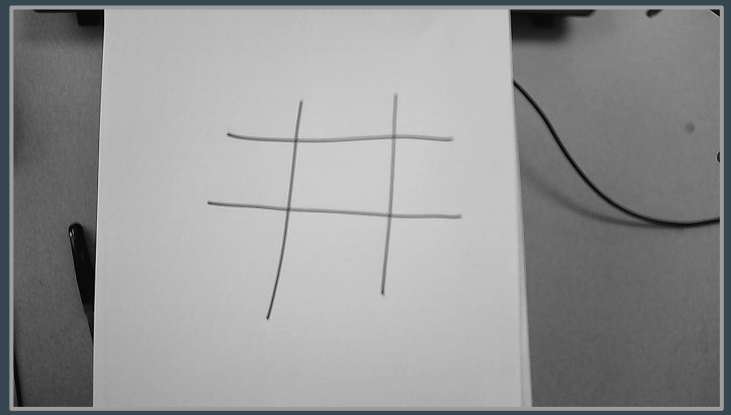
# GetPaper - Focus images on TicTacToe board



RGB2gray
Sharpen Edges



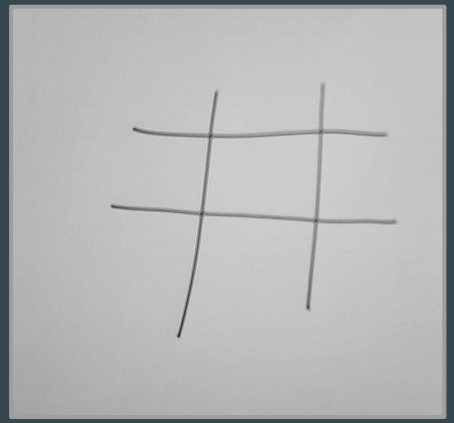
# GetPaper - Focus images on TicTacToe board

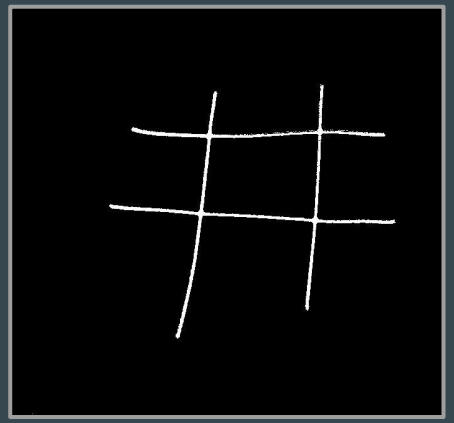


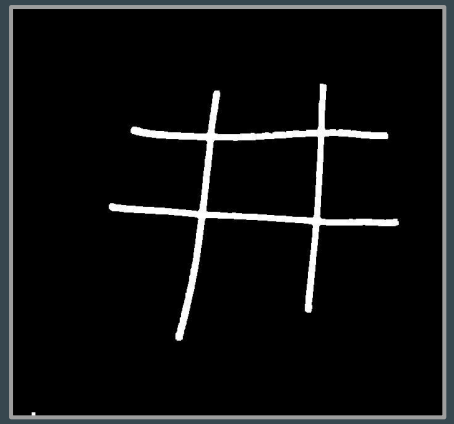
Smooth out background
Binarize

Extract Centroid and Dimensions.
& Operator and Zoom In.

# GetPaper - Focus images on TicTacToe board



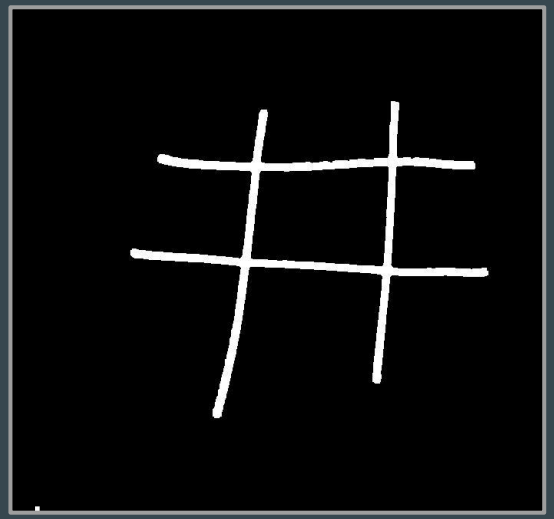




Binarize
Negative

Dilation

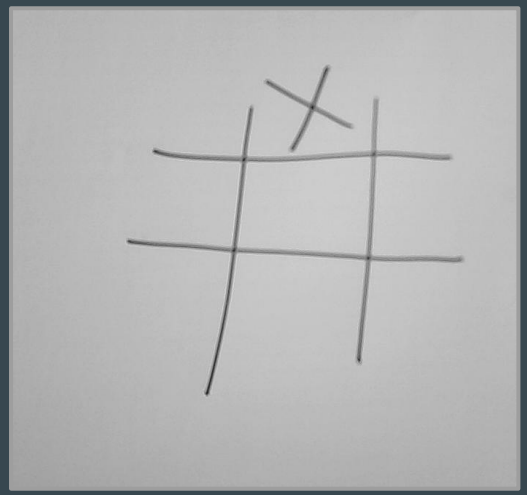
# GetBoardDimensions - Extract Center and x,y Thresholds

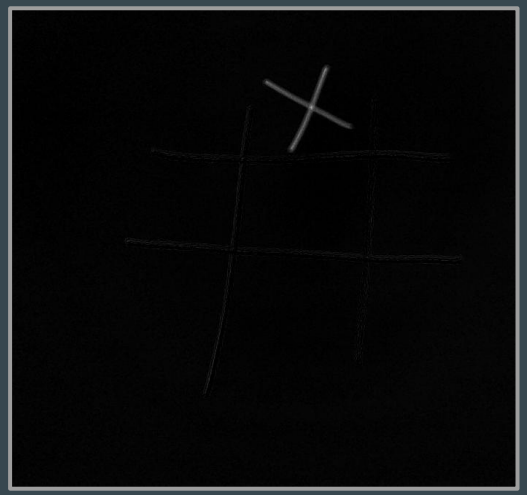


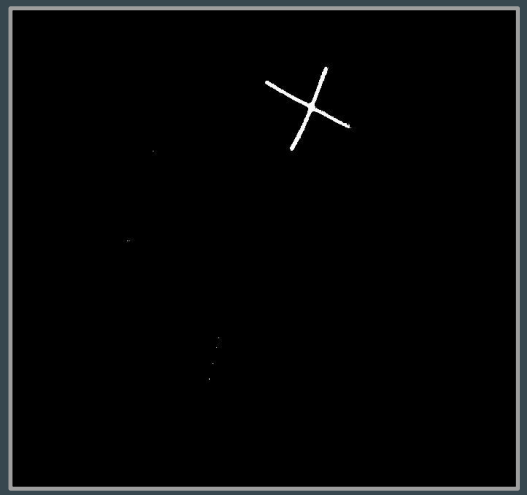
Using RegionProps, find Centroid, and Dimensions of TicTacToe region.

X_threshold = length(region)/3;
Y_threshold = height(region)/3;

# CaptureMove - Determine Piece and Co-ordinates



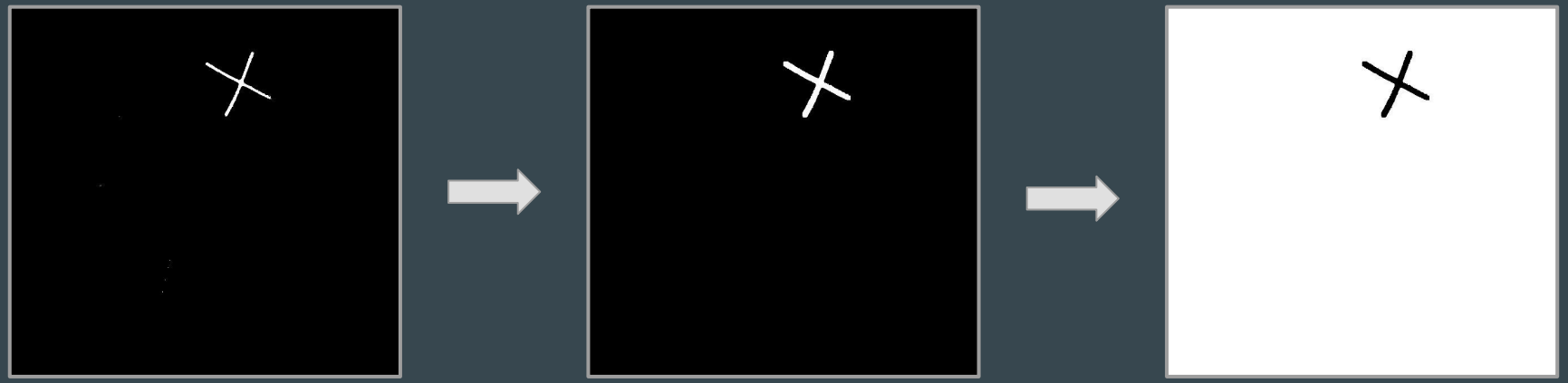




Difference between new_state and previous_state

Threshold to remove faint lines and focus piece

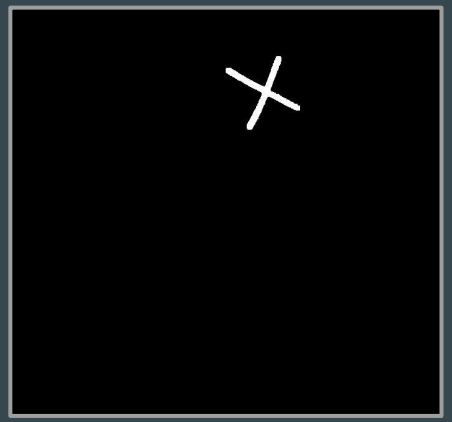
# CaptureMove - Determine Piece and Co-ordinates



Opening to remove
any unwanted artifacts

Negative.
X has 1 CC, and
O has 2 CC.

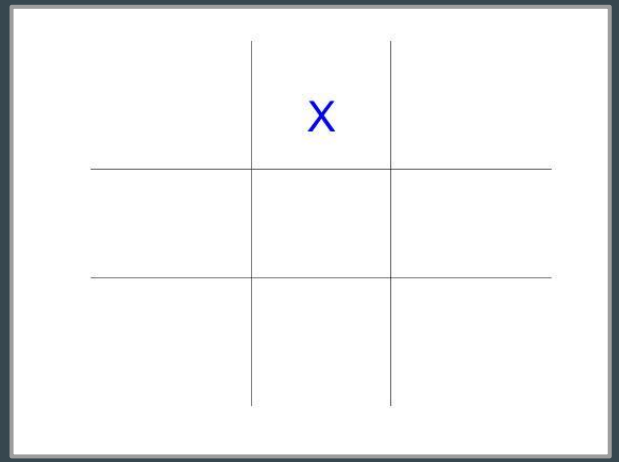
# CaptureMove - Determine Piece and Co-ordinates



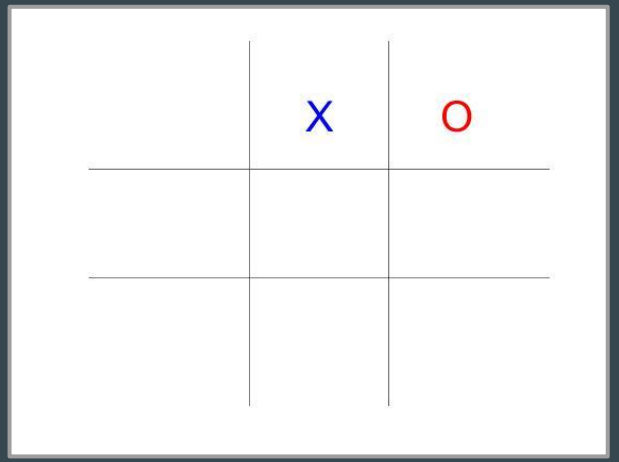
Using RegionProps, find Centroid of Piece

Determine position of piece relevant to the TicTacToe board.

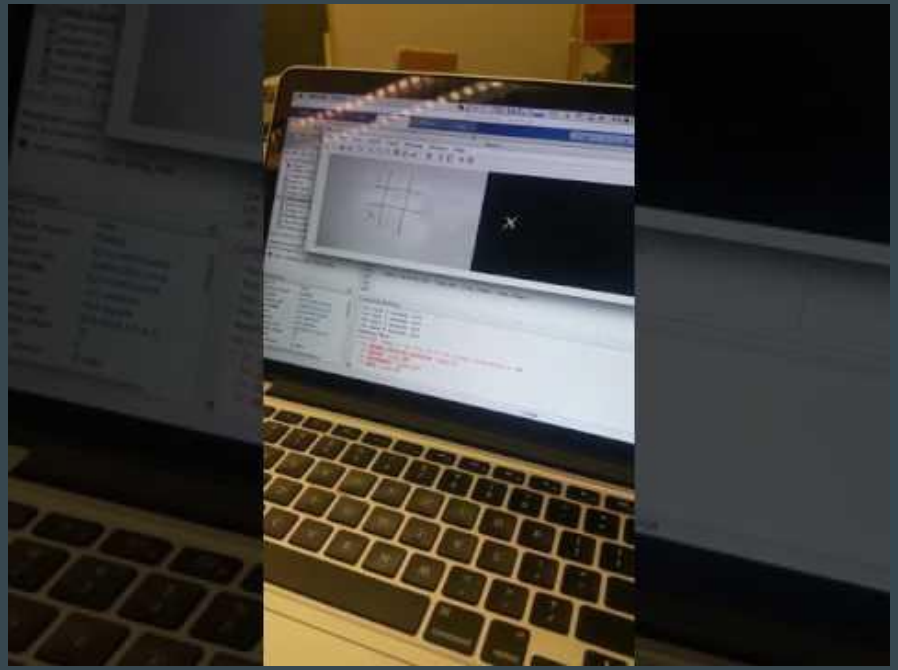
# ReturnCompMove- Find and position comp move

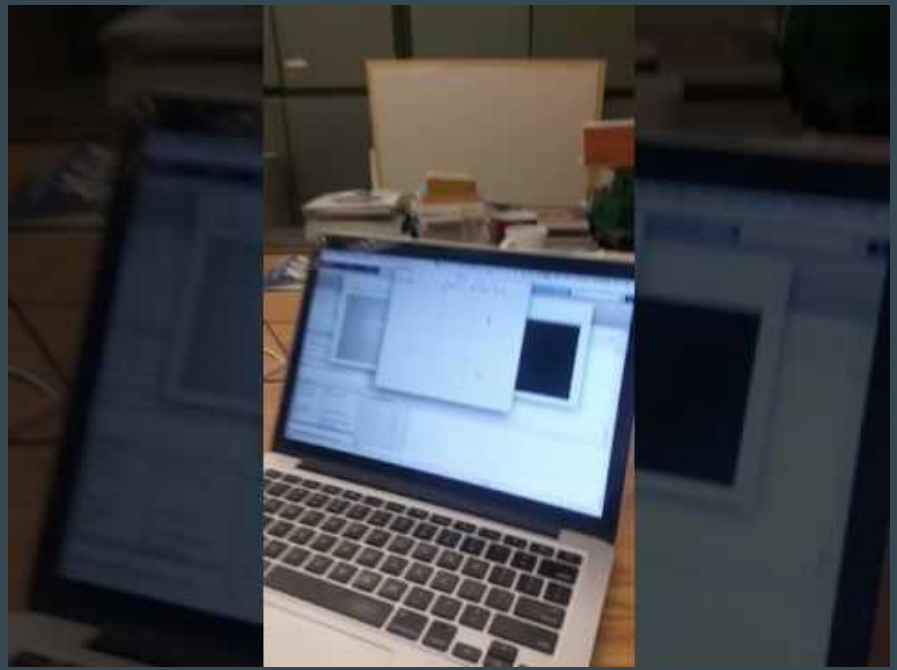



Select a random empty slot and select it for opposite piece

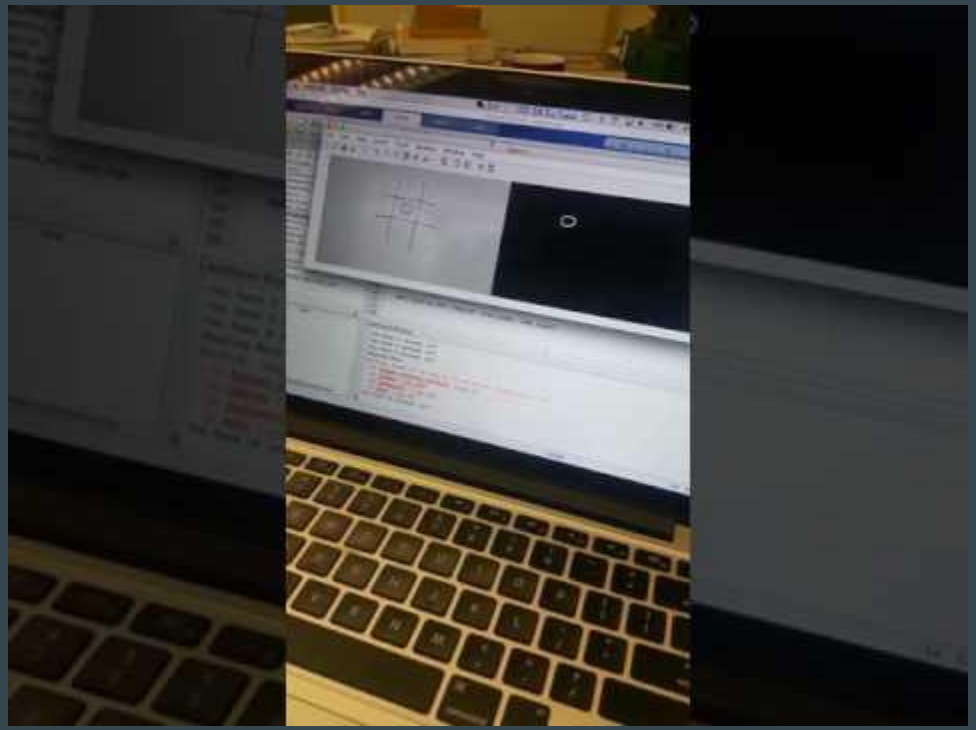


# Performance

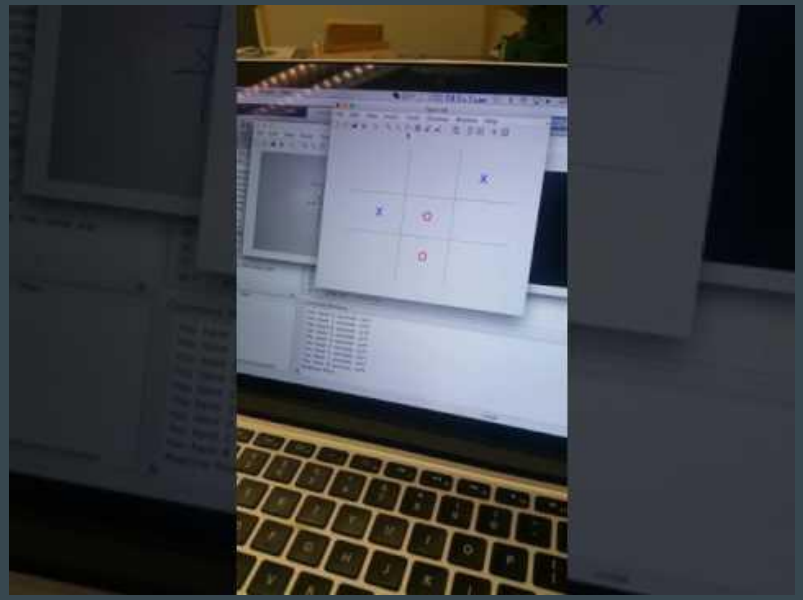


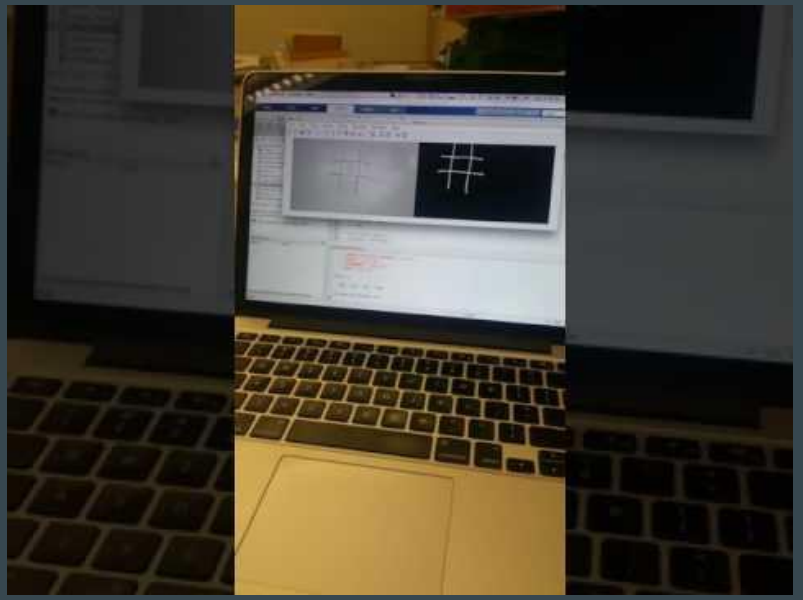


# Performance



# Limitations of Our Algorithm

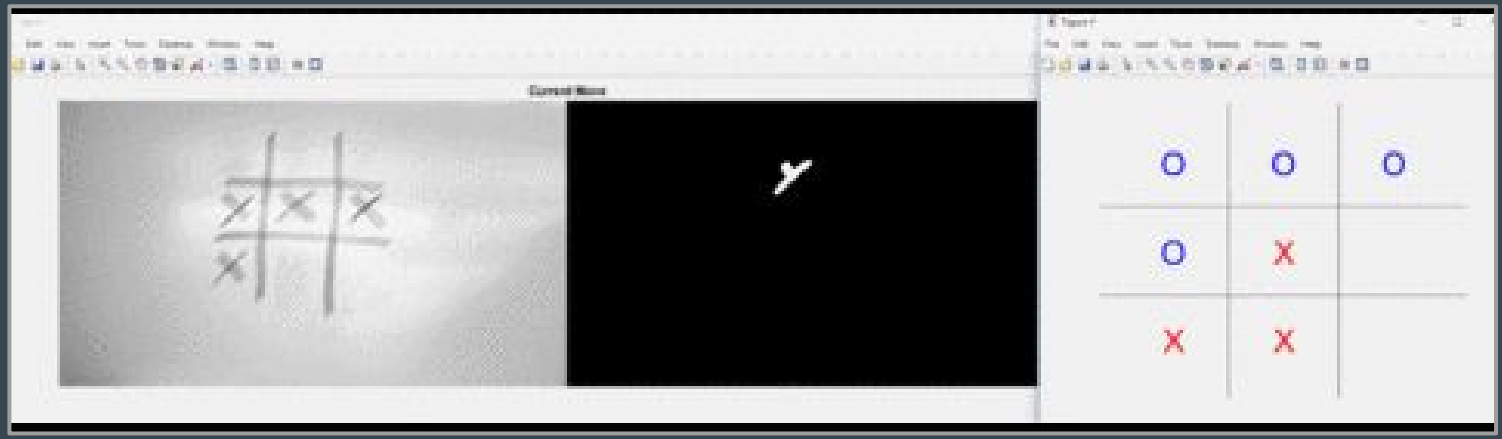




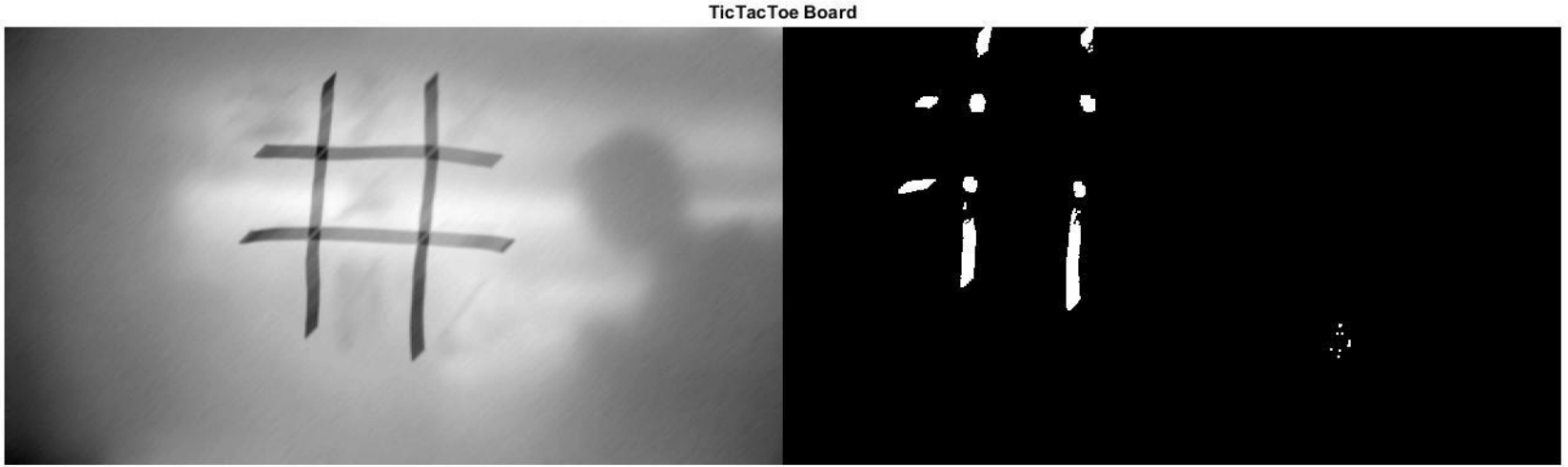
# Noisy images: Motion - 5 and 15 pixels


# Noisy images: Motion - 25 and 35 pixels


# Noisy images: Motion - 45 pixels

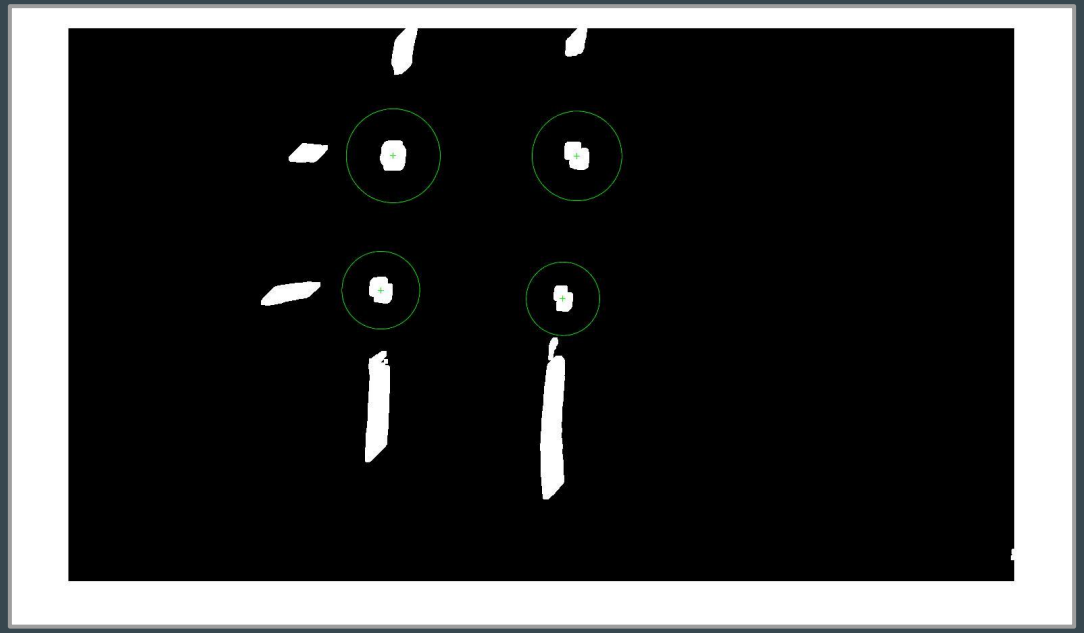


# Noisy images: Motion - 45 pixels

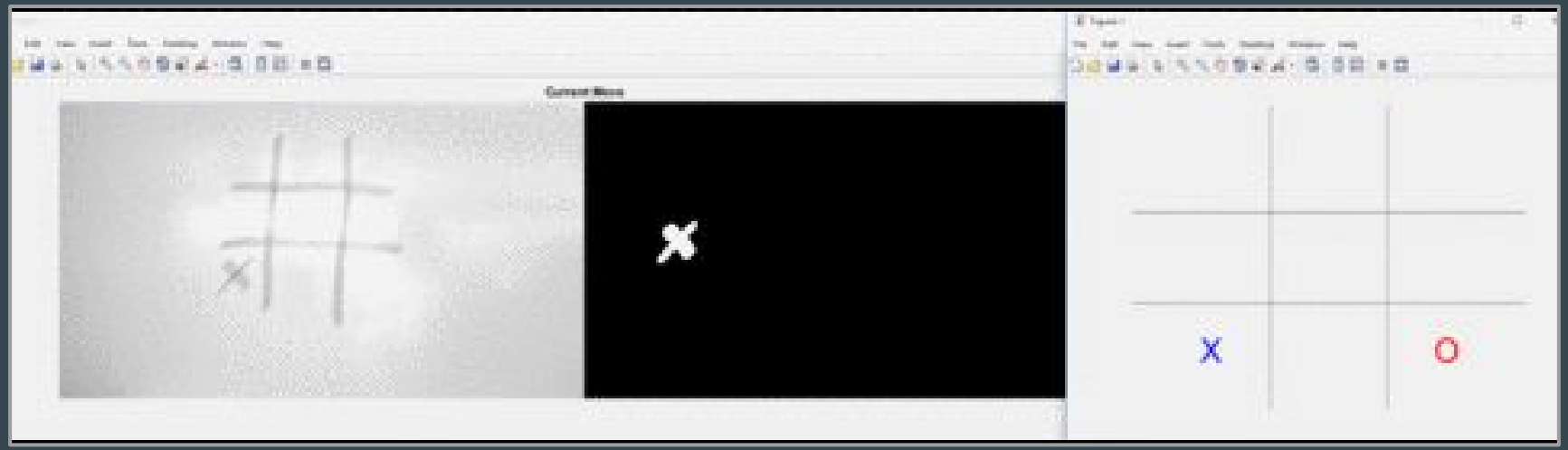


# Noisy images: Motion - 45 pixels

SURF: Speeded Up Robust Features



# Noisy images: Motion - 45 pixels

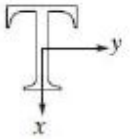
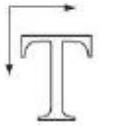


# Noisy images: Motion

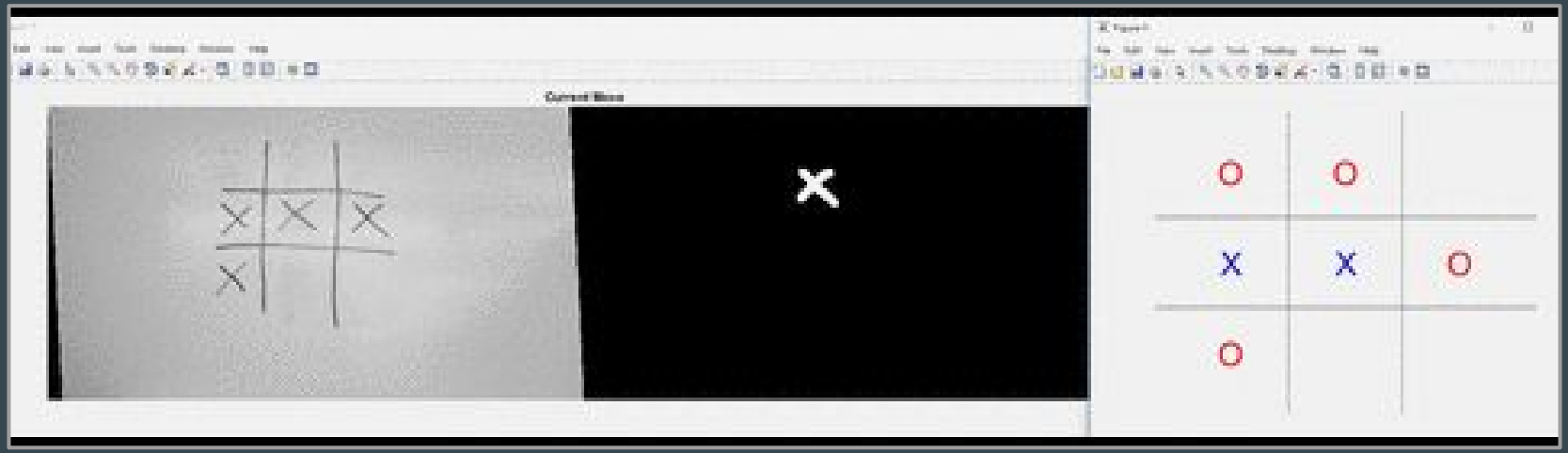
SURF Method works until 60 pixels.

Comparison between no motion and 45 pixel motion:

# Noisy images: Sheer

| Transformation Name | Affine Matrix, T | Coordinate Equations | Example |
|---|---|---|---|
| Identity | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v$<br>$y = w$ |  |
| Scaling | $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = c_x v$<br>$y = c_y w$ |  |
| Rotation | $\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v\cos\theta - w\sin\theta$<br>$y = v\cos\theta + w\sin\theta$ |  |
| Translation | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$ | $x = v + t_x$<br>$y = w + t_y$ |  |
| Shear (vertical) | $\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v + s_v w$<br>$y = w$ |  |
| Shear (horizontal) | $\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v$<br>$y = s_h v + w$ |  |

# Noisy images: Sheer 5$^0$

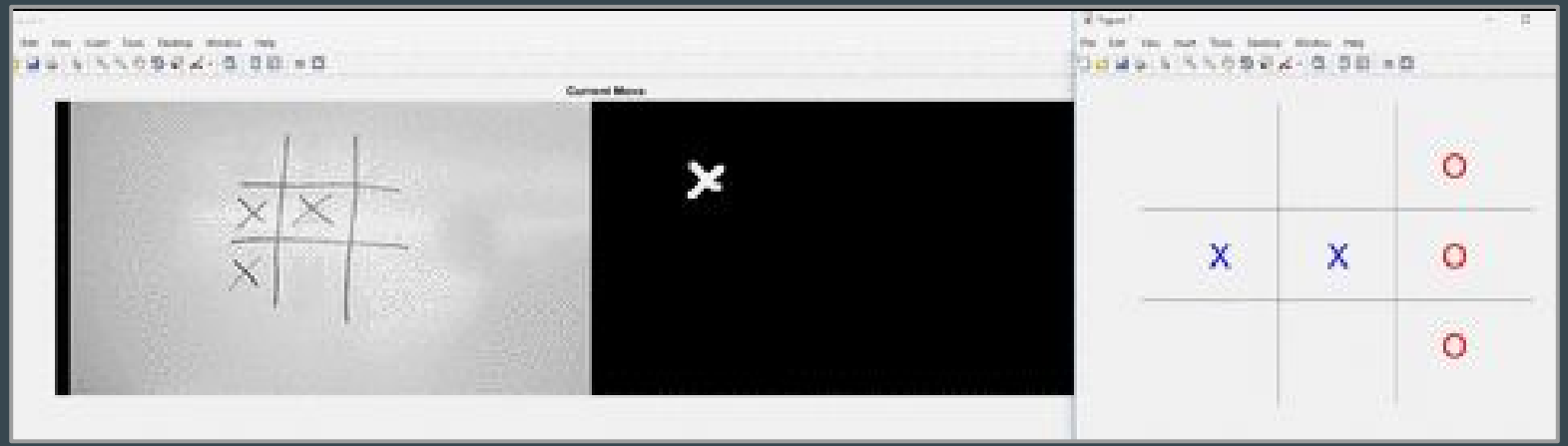


# Noisy images: Sheer $5^0$

Artifacts on board. Center is not properly defined.

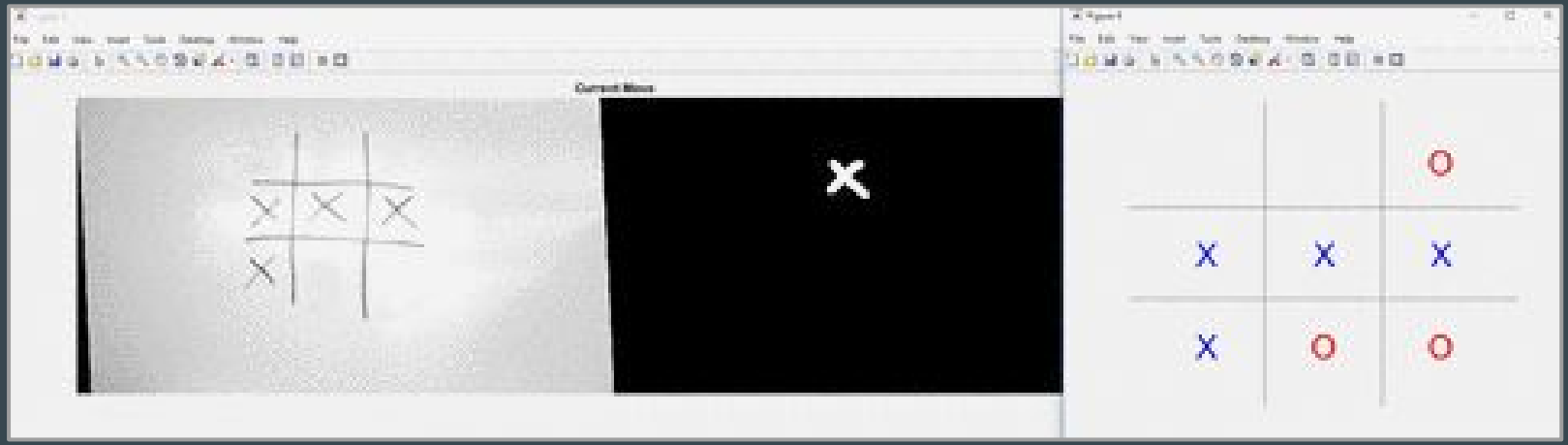
# Noisy images: Sheer 5$^0$

Reverse Sheer - Does not work



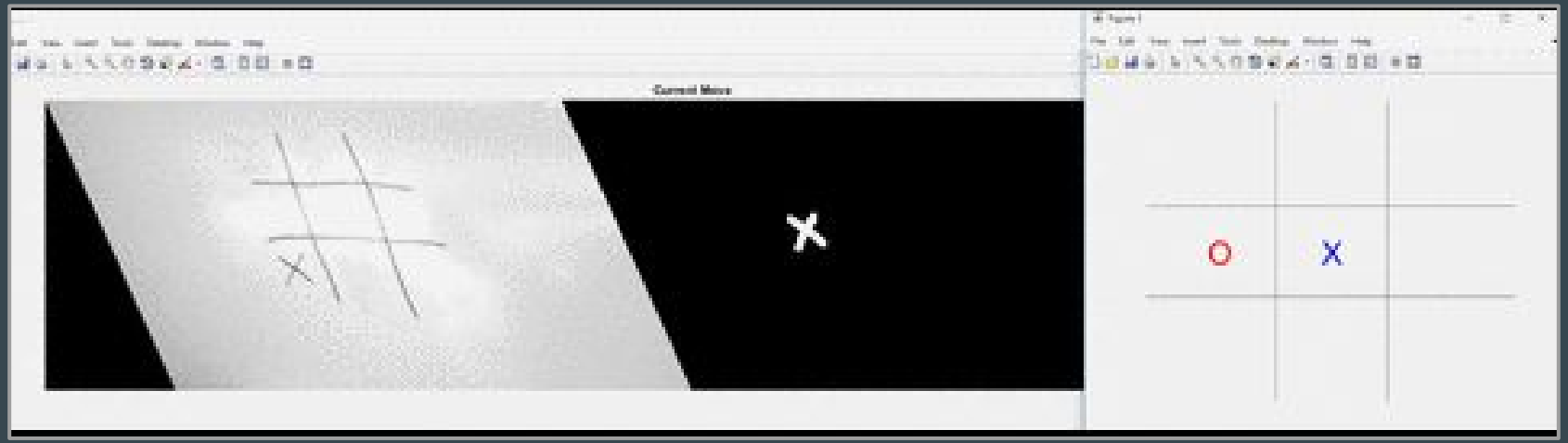
# Noisy images: Sheer 5$^0$

With 9x9 Median Filtering:



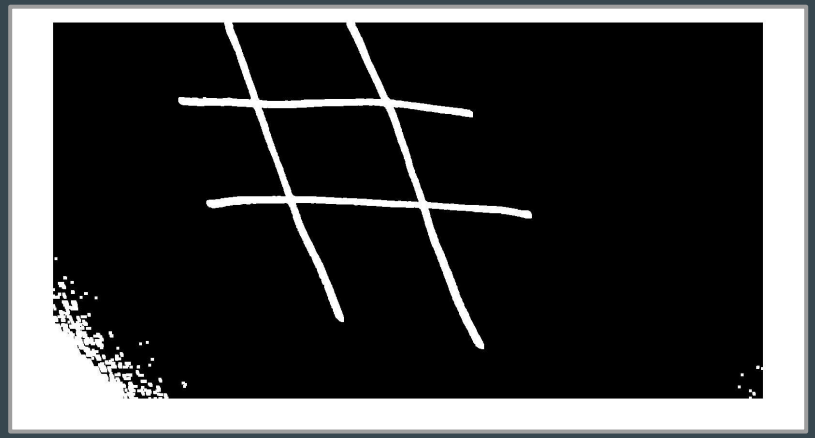
# Noisy images: Sheer $45^0$

Median Filter stops working at $45^0$



# Noisy Image: Motion and Sheer Problem

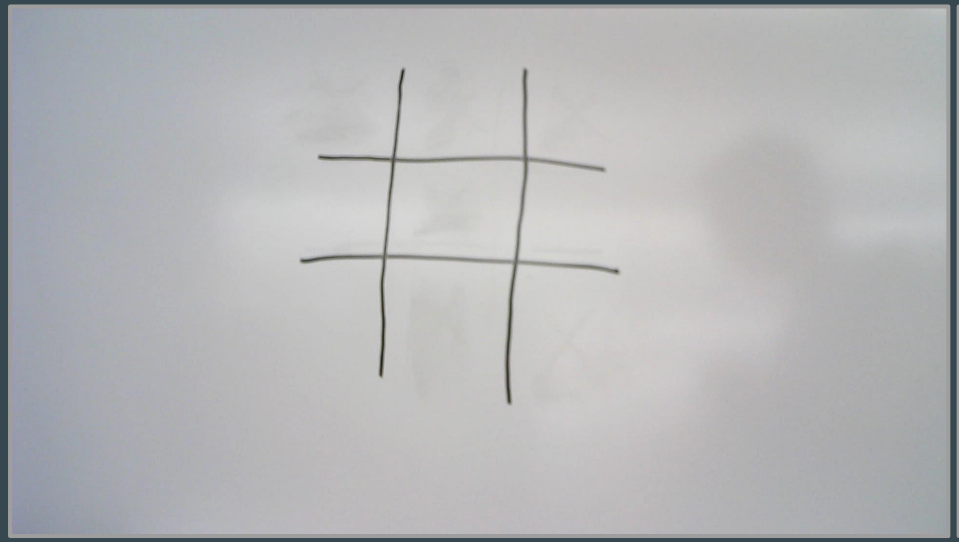
Biggest problem is defining center of the board.



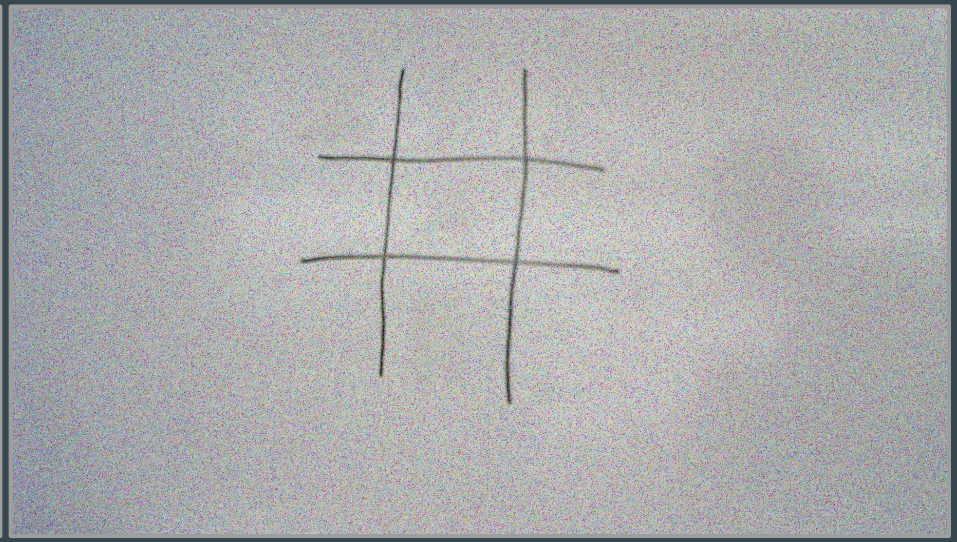
Noisy clumps of pixels appear on image.
A viable solution would be local enhancements:

$$g(x,y) = \begin{cases} E \cdot f(x,y) & \text{if } m_{S_{xy}} \le k_0 M_G \quad \text{AND} \quad k_1 D_G \le \sigma_{S_{xy}} \le k_2 D_G \\ f(x,y) & \text{otherwise} \end{cases}$$

# Noisy images: salt & pepper noise with 0.15 noise density

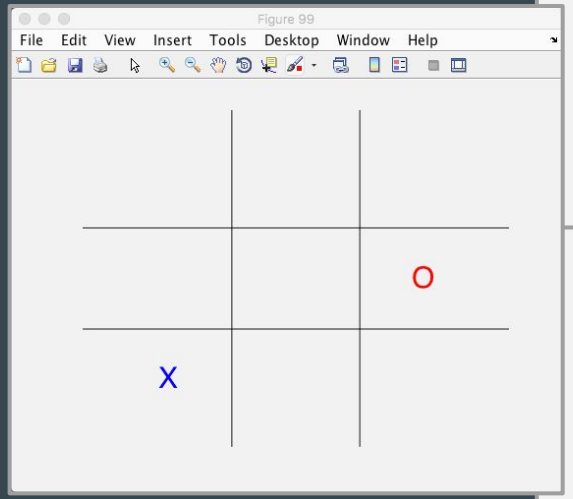


Density = 0

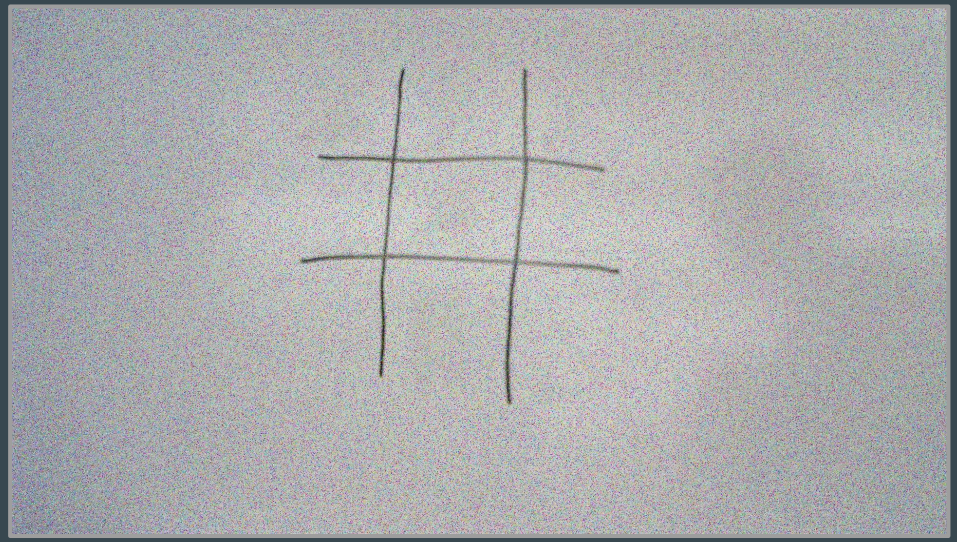


Density = 0.15

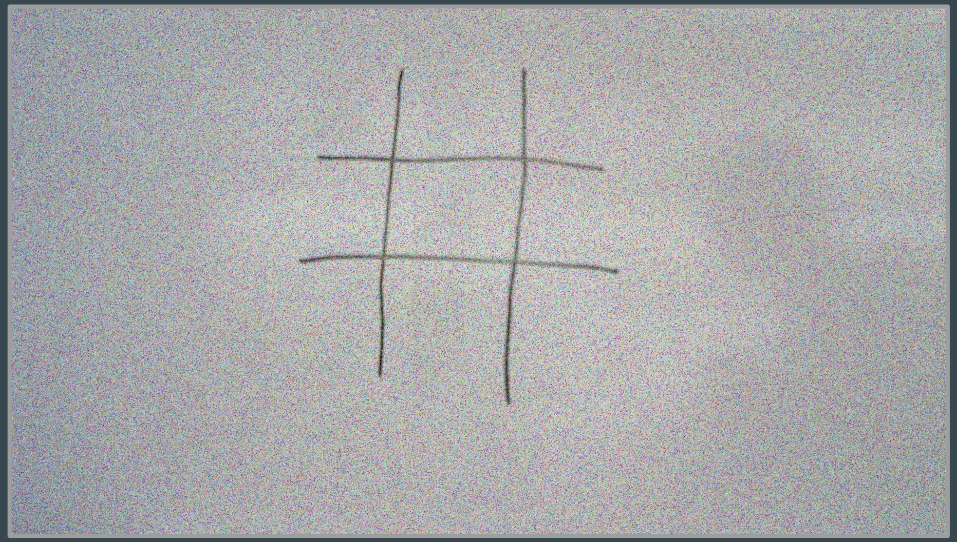
# Noisy images: salt & pepper noise with 0.15 noise density




# Noisy images: salt & pepper noise with 0.2 noise density

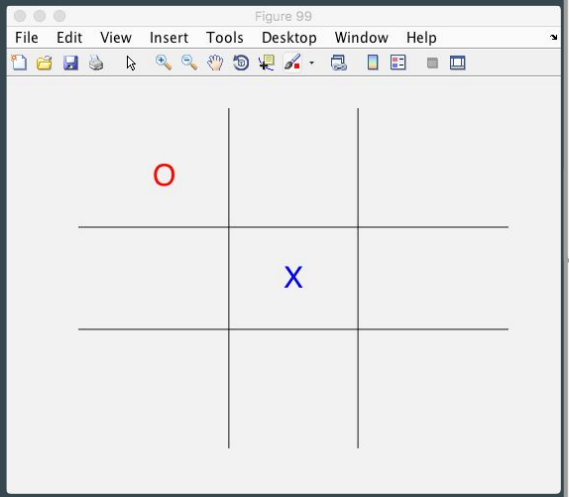


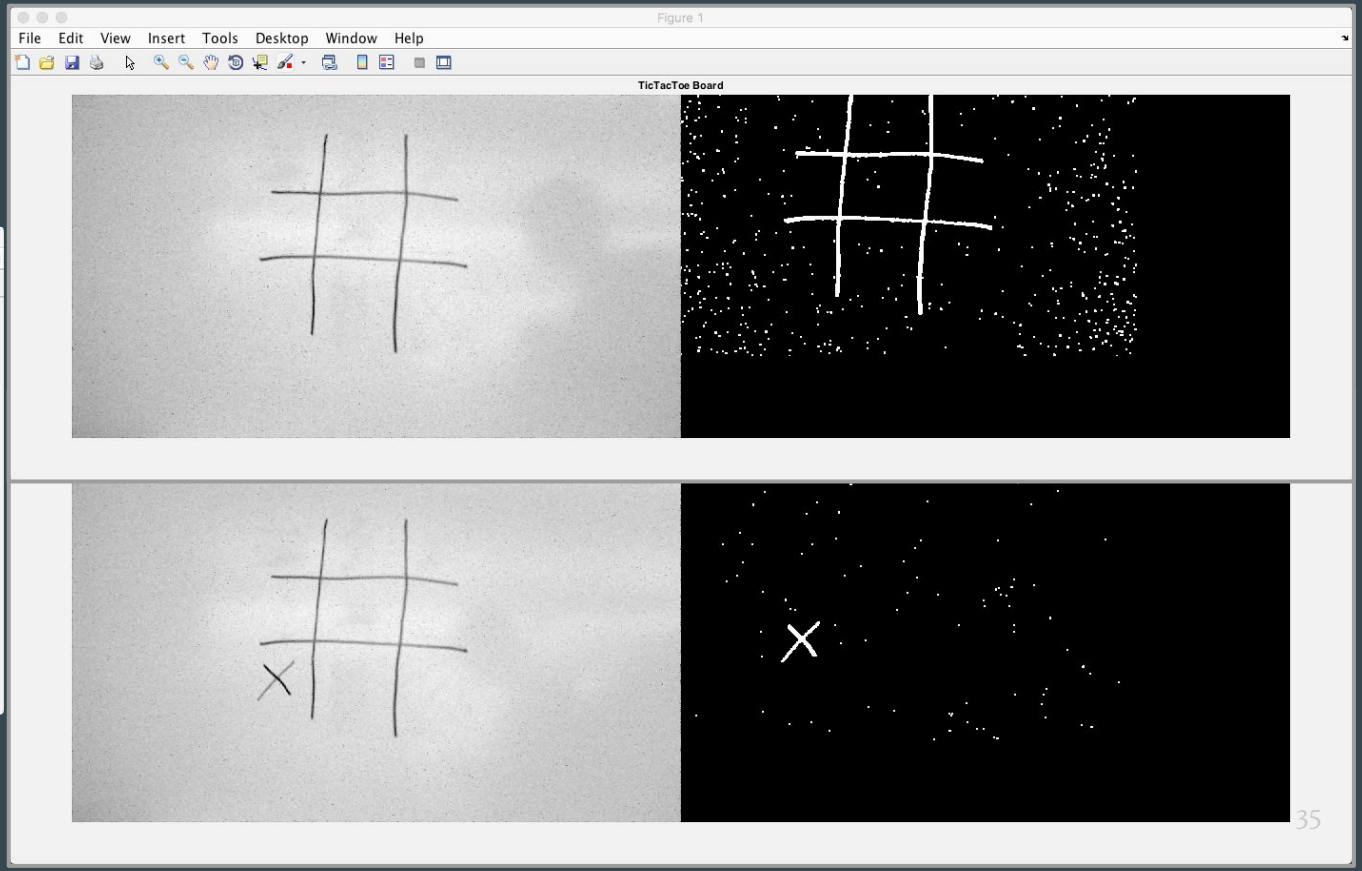
Density = 0.15



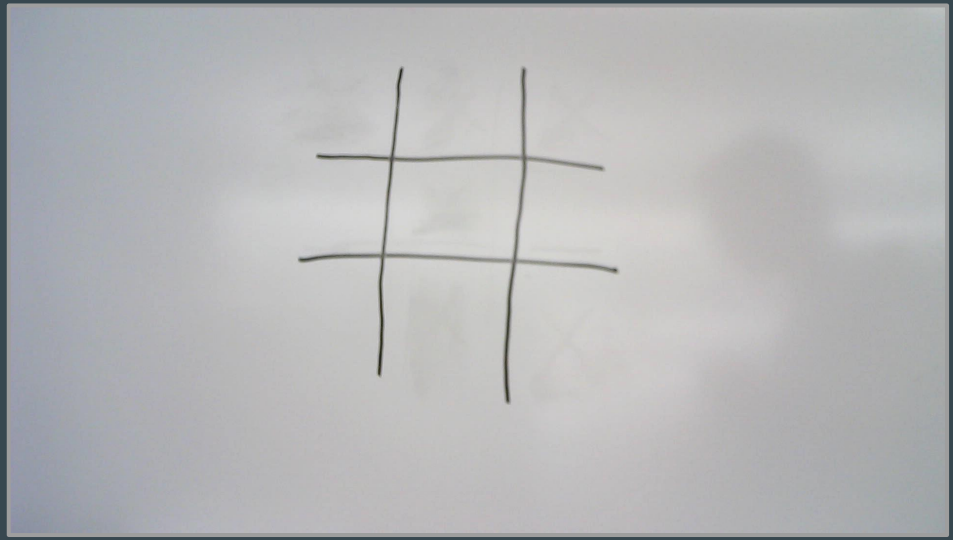
Density = 0.2

# Noisy images: salt & pepper noise with 0.2 noise density

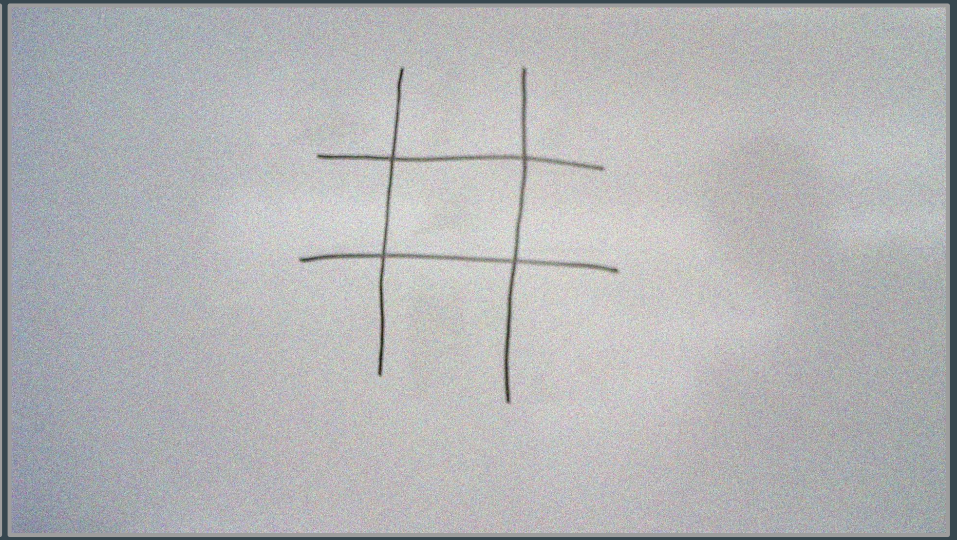


# Noisy images: Gaussian noise with 0.015 variance

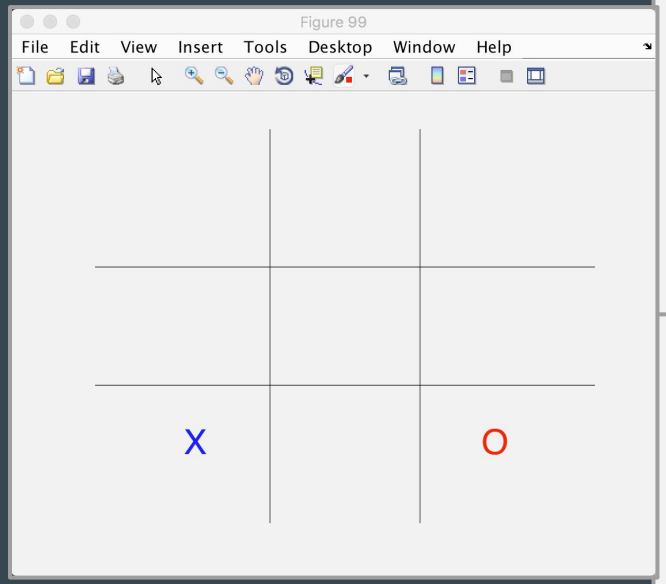


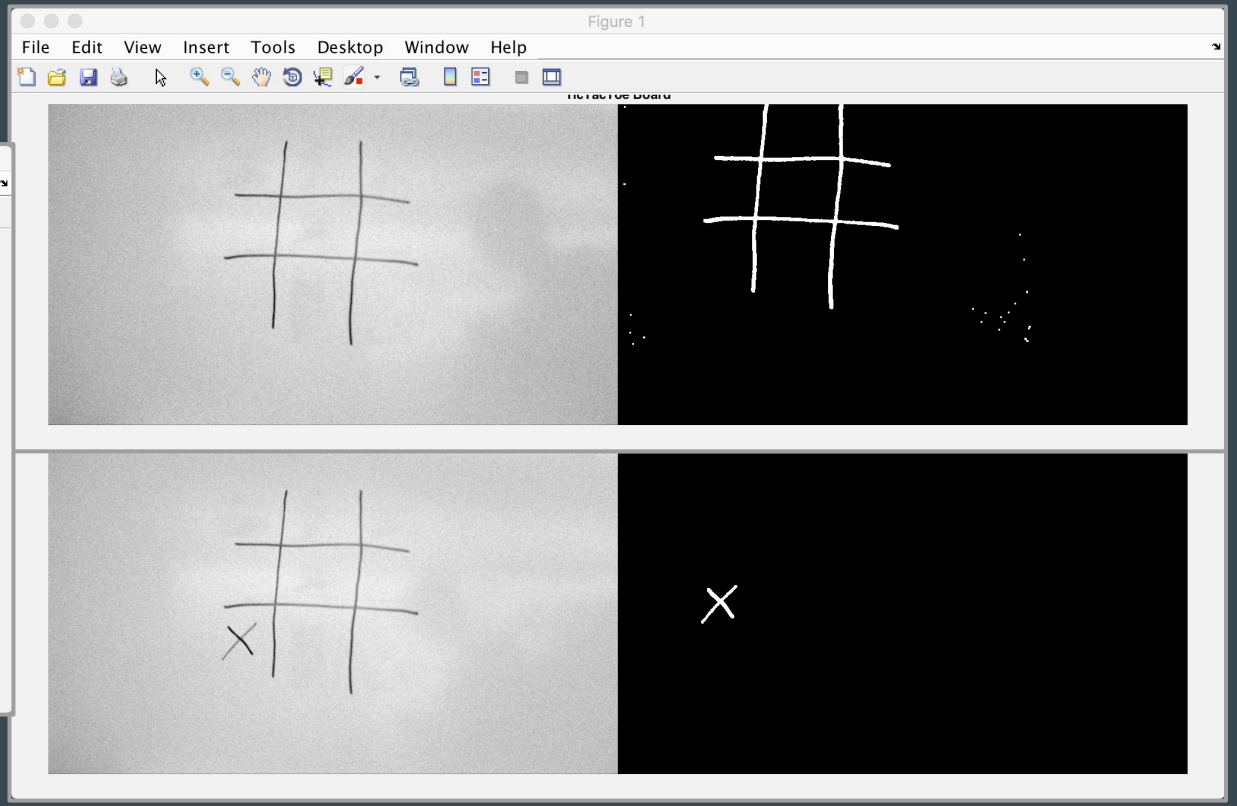
No noise



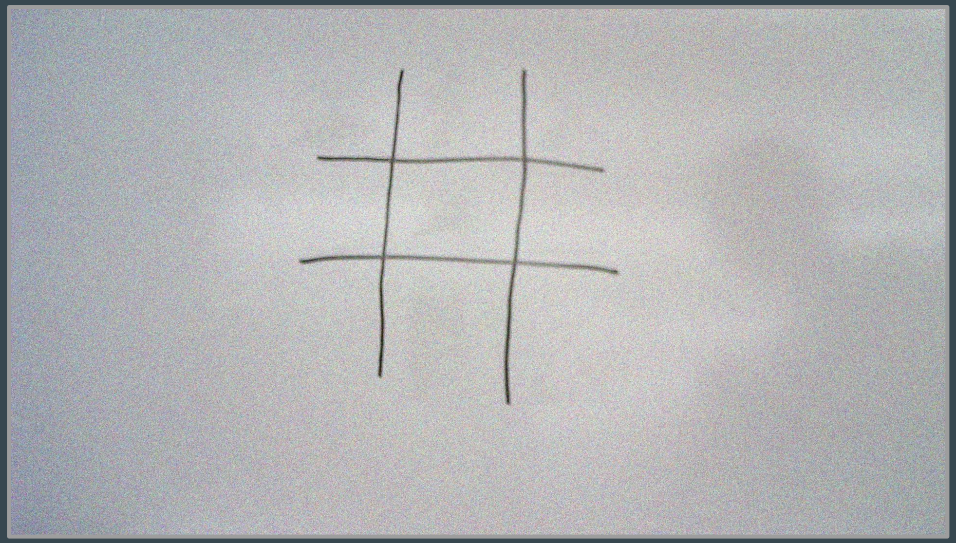
Noise = 0.015 * variance(I)

# Noisy images: Gaussian noise with 0.015 variance

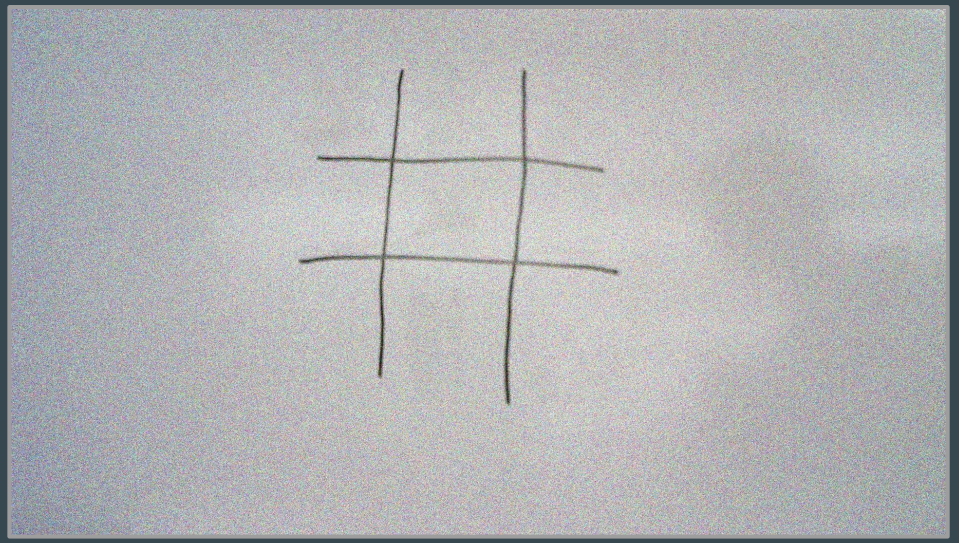


# Noisy images: Gaussian noise with 0.03 variance

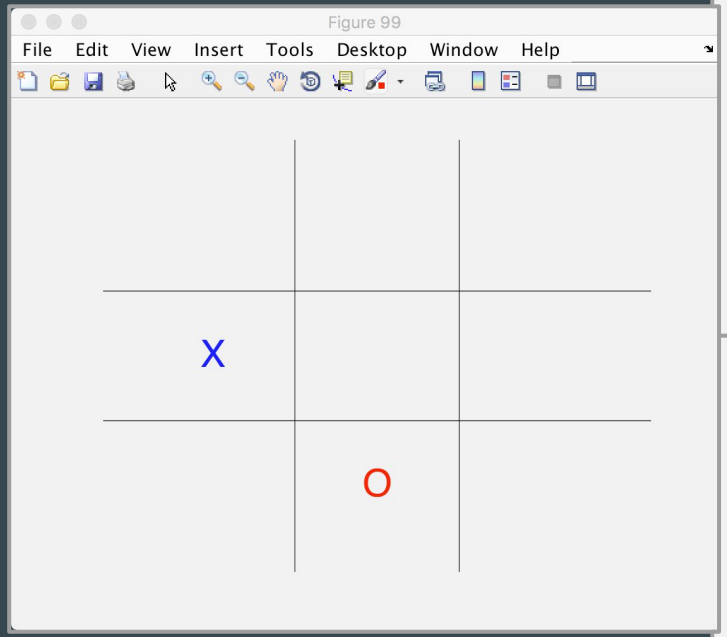


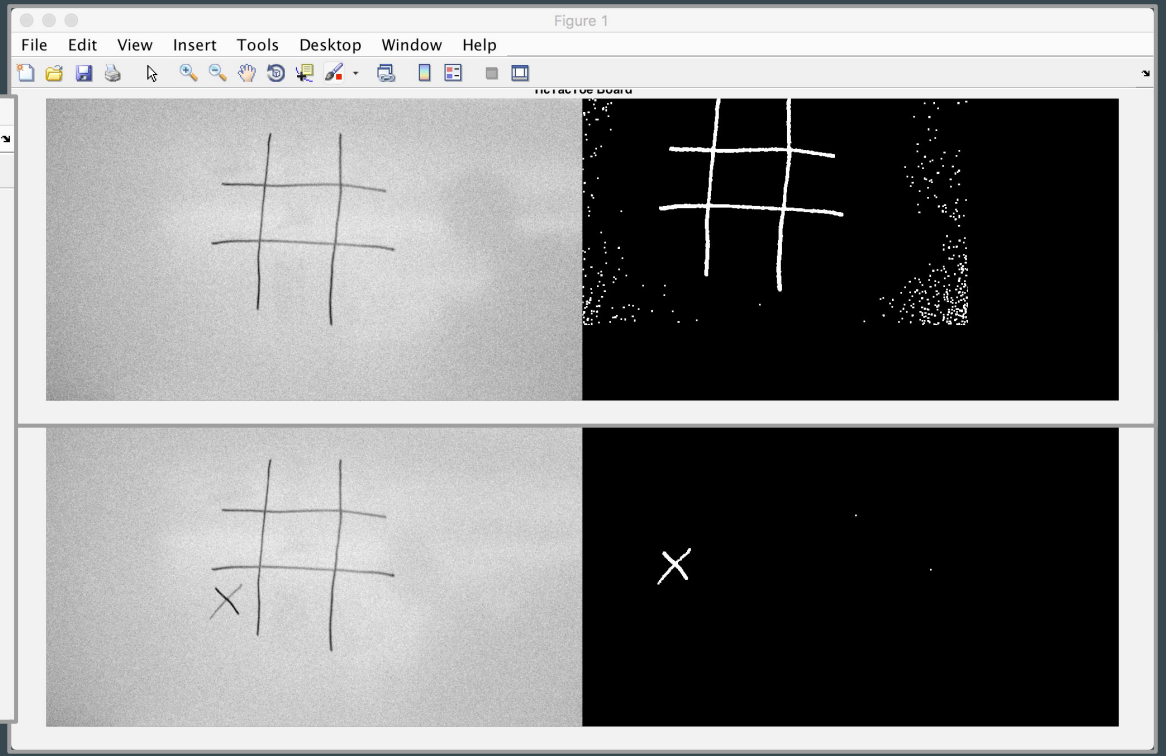
Noise = 0.015 * variance(I)



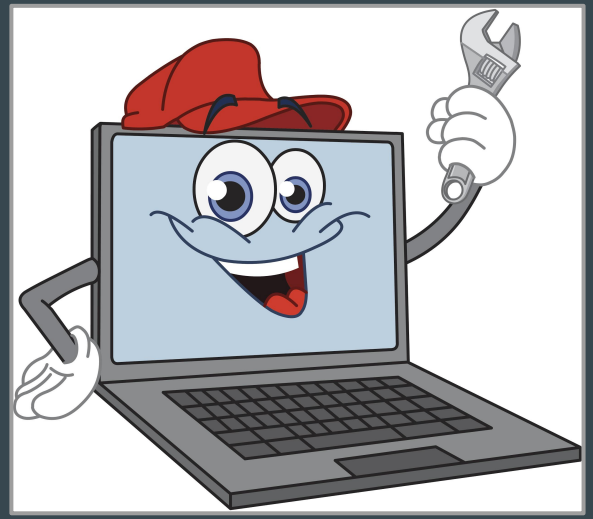
Noise = 0.03 * variance(I)

# Noisy images: Gaussian noise with 0.03 variance



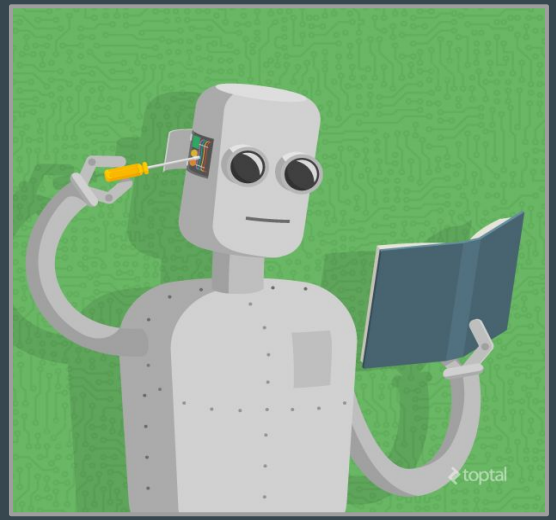



# Things that we should have done



Smarter moves by the computer

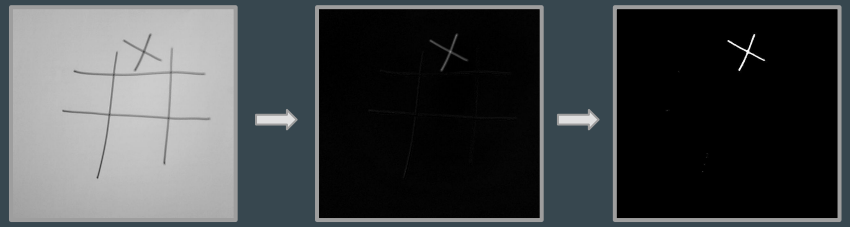

Remove the timer system for user input




- Machine learning for shape detection

## Augmented Reality = CG Image + Real World

## Optimal setup - successful result



## Noisy setup - mixed result

Nikesh Muthukrishnan

Ray Wen

first_name.last_name@mail.mcgill.ca