**Isolating game board**

1. Locate game board? (initially & between turns)
   1. game board might require static location (i.e. Placed flat on table, directly in front of robot at a set distance)
2. *ALPhotoCapture* will obtain image from camera
3. Use OpenCV to convert image to binary (1-bit monochrome)
4. Use OpenCV to remove unwanted noise from image
   1. Threshold for noise will be determined through testing. Using a pre-made game board will simplify this process.
5. Use OpenCV to find *extreme points* to isolate the game board. Would require that the entire field of view of the robot would be a mostly white background – a static board location would allow for hard-coded calibration, making this much simpler.

**Determine state of board**

1. Isolate individual spaces on the board; two options:
   1. Perform a Hough Line Transformto calculate positions of line intersections, defining individual cells
      1. Very generic, would likely pick up previously-played X’s as well
   2. Find contours using OpenCV with retrieval mode “RETR\_EXTERNAL”, which only draws the outer-most contours (The bounding lines separating individual spaces)
      1. Extremely easy to implement using OpenCV, but would rely on proper isolation of the game board in order to properly bound and isolate outer cells
      2. External retrieval mode will ignore any contours within these bounds, such as an “X”
2. Analyze each space individually; three options:
   1. Train a neural network to differentiate between X’s and O’s
      1. Due to binary outcome as well as significant differences between character shapes, should be fairly accurate
   2. Use OpenCV to find contours. Differentiate by determining solidity (contourArea/convexHullArea)
      1. Theoretically, O’s should have a solidity of 1, while X’s will likely be somewhere between 0 and 0.5; exact thresholds can be determined through testing.
      2. Significantly less complex than implementing a neural network
   3. Use Google’s text-recognition API’s
      1. Same as option ‘a’, except *much* simpler and, most likely, more accurate

**Notes**

1. Placing the game board in a static, pre-defined location relative to the robot would remove the need to locate the game board upon initialization, allowing for hard-coded calibration. With this method, the user could still pick up the board to make a move, as long as it is returned to approximately the same location once their turn is complete.
2. Using a specific, pre-made game board with static and well-defined boundaries (i.e. a white board with black electrical tape separating cells, rather than a hand-drawn board) would also allow for further hard-coded calibration, ensuring consistency
   1. Identifying the game board and isolating individual spaces is expected to be significantly more difficult than identifying game pieces within cells (X’s and O’s). Though using pre-made game pieces was previously discussed, identifying hand-written symbols (on a whiteboard, for example) may be more trivial than expected, and should be explored.
3. Due to the turn-based nature of the game, computer vision processing should be feasible locally using only the robot’s internal resources. However, if external processing ends up being necessary for any reason, serverless computation via AWS Lambda would be viable as the game does not require any real-time processing, removing the need for setting up and maintaining a physical or cloud-hosted server.
   1. AWS Lambda also has an easy-to-use python API, further simplifying things