Ultimate Chess

Team B1: Yoorae Kim, Demi Lee, Anoushka Tiwari

Application Area

- Enjoy physical chess during pandemic
- Help the elderly who aren’t comfortable with apps still enjoy chess
- Learn to be better at chess from practicing chess with AI
- Areas Covered:
  - Signals, Software, Circuits

Solution Approach

- Computer Vision
  - Webcam placed on top of the chessboard
  - Detect player’s move using OpenCV
- Software
  - Check if player’s move is valid or not by implementing chess game logic
  - Use existing chess AI engine to come up with next move
- Hardware
  - Display human player and AI’s move using LEDs
  - Player presses push button:
    - After making their move -> Signals camera to take picture of board
    - When wrong LEDs light up for user move -> Max 2 retries CV detection

Complete Solution (Board detection)

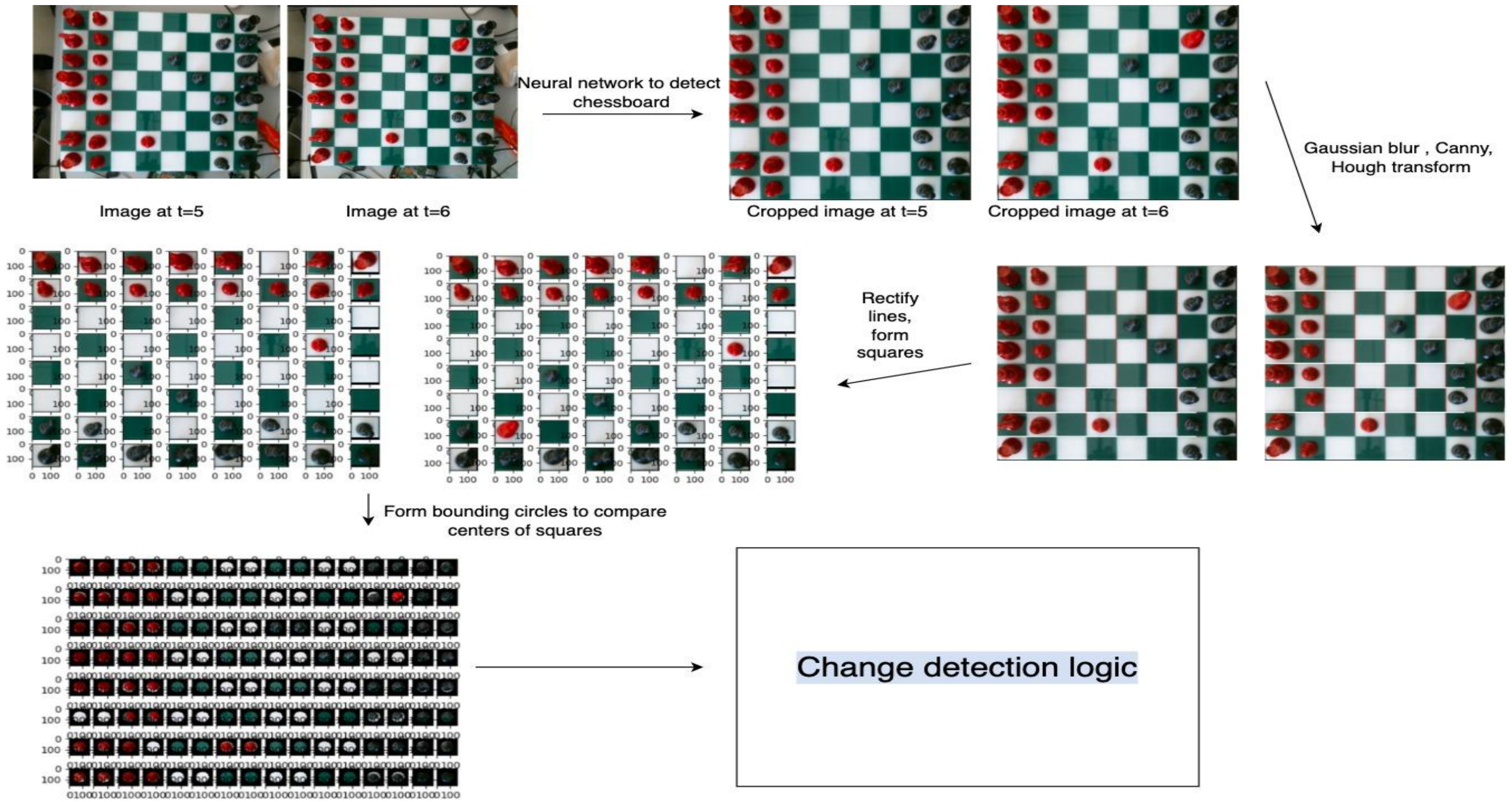
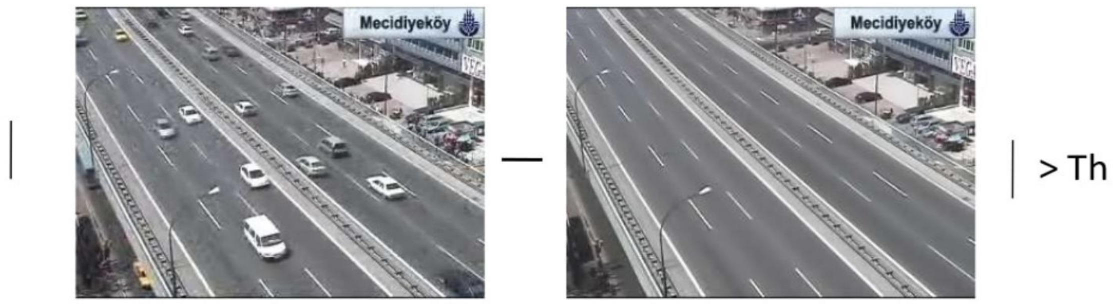


Image at t=5
Image at t=6
Neural network to detect chessboard
Cropped image at t=5
Cropped image at t=6
Gaussian blur , Canny, Hough transform
Rectify lines, form squares
Form bounding circles to compare centers of squares
Change detection logic

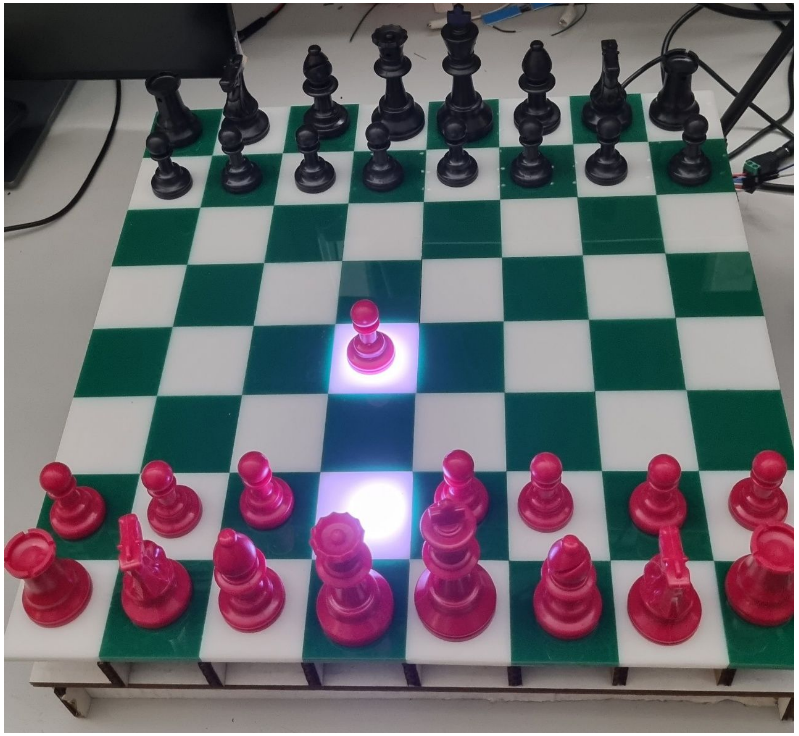
Change detection logic

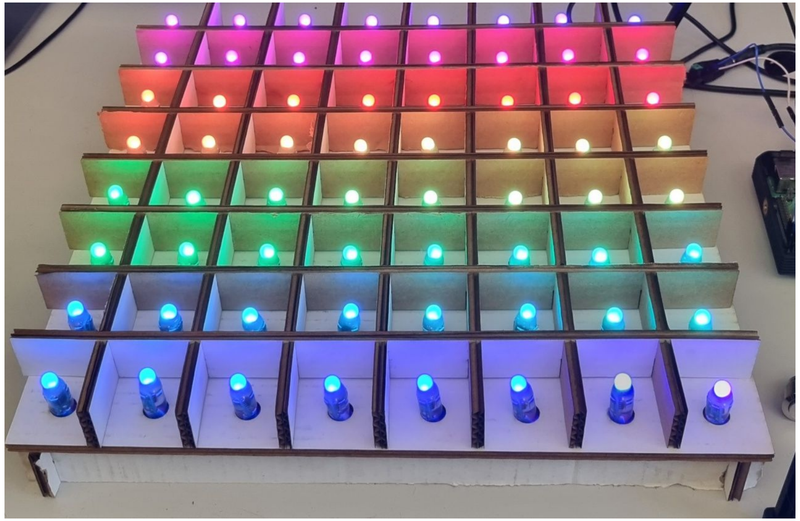
Used 'frame difference' algorithm of background subtraction



1. Set a bounding circle mask on each square to reduce the error
2. Compute cumulative absdiff value on RGB for each square (higher value means more change in color has occurred)
3. Output the coordinates with the computed value greater than threshold.
4. Out of two coordinates, determine which piece moved to where from the previous board state list.

Complete Solution (Hardware)





- Individually addressable LED strip
- Button to press when turn is over / incorrect CV detection

Metrics - Computer Vision

Requirement	Expected result	Result	Testing strategy
Move detection time	<24s	Avg 8.5s	Use Python Timeit library
Move detection accuracy	99%	26/27	Measured as the % of player moves correctly detected
Distance of chess piece from center	D <= 1.875 - radius of piece	TBD	

Metrics - Valid Logic

Testing strategy: tested detection of 10 legal moves, 10 illegal moves, and 10 capturing moves per piece type

Piece type	Expected Result	Result	Description
King	100%	100%	
Queen	100%	83%	Error in the diagonal move detection, error fixed and updated
Rook	100%	100%	
Bishop	100%	57%	Error in the diagonal move detection, error fixed and updated
Knight	100%	100%	
Pawn	100%	100%	

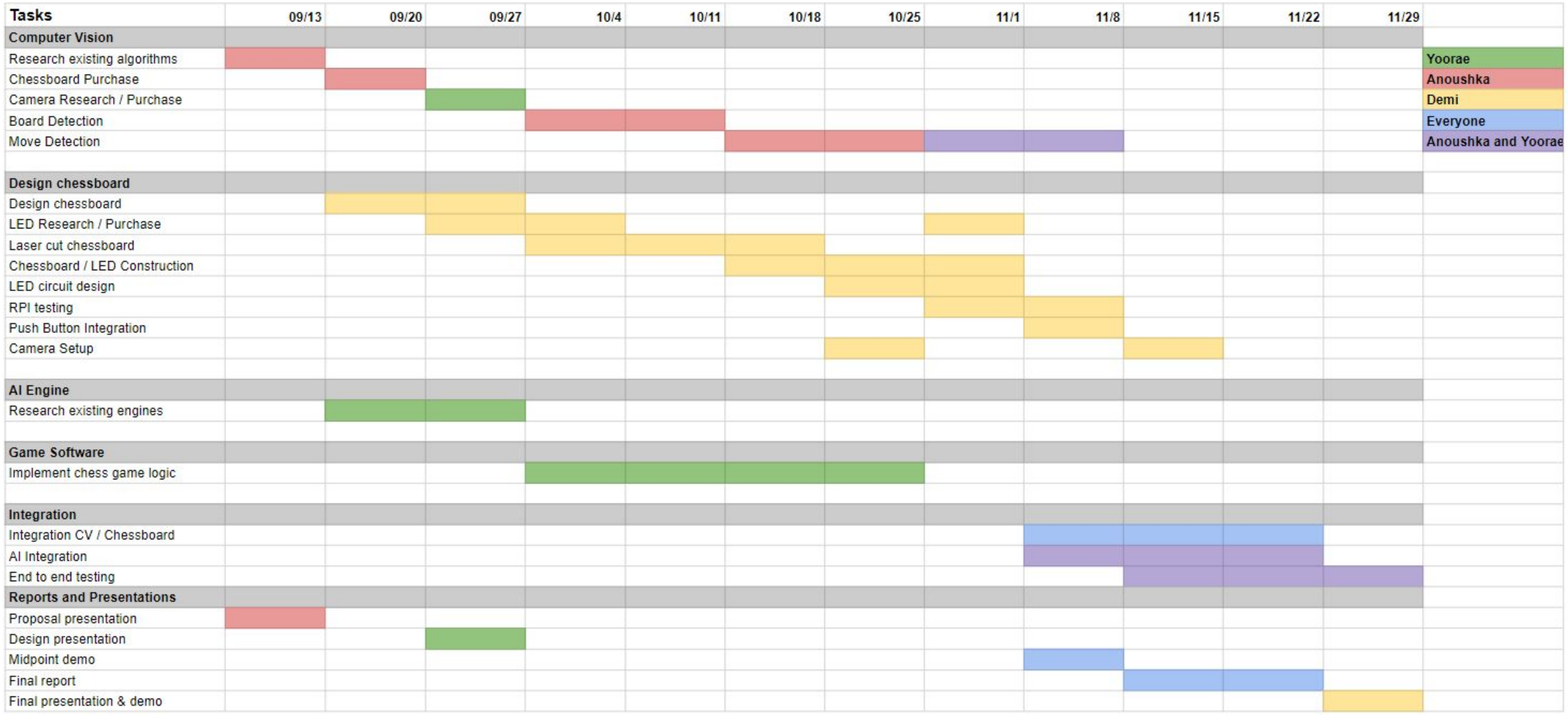
Metrics - LEDs

Requirement	Expected Result	Result	Testing Strategy
LED Code Execution Time	< 100ms	30 tests Avg. 28ms Max 31ms	Measured the time necessary to parse the algebraic chess notation into LED index and light up the corresponding LEDs
LED Correctness	100% Correct	30/30 tests passed	Given a coordinate and color, visually confirmed that the correct LEDs light up

Trade-offs

- Red vs White chess pieces
  - Additional cost switching from white pieces
  - Easier to distinguish between white square and red piece
- Turn-based vs Real-time
  - User confirms that they are done making their move
  - Less smooth, but speeds up CV because it doesn’t have to figure if the move is done
- Move detection validation
  - User verifies the move was detected correctly by pressing or not pressing a button
  - Ease of use v.s. Accuracy tradeoff

Schedule



Tasks	09/13	09/20	09/27	10/4	10/11	10/18	10/25	11/1	11/8	11/15	11/22	11/29
Computer Vision												
Research existing algorithms	🟥											
Chessboard Purchase		🟥										
Camera Research / Purchase			🟩									
Board Detection				🟥	🟥							
Move Detection						🟥	🟥	🟪	🟪			
Design chessboard												
Design chessboard		🟨	🟨									
LED Research / Purchase			🟨	🟨				🟨				
Laser cut chessboard				🟨	🟨	🟨						
Chessboard / LED Construction						🟨	🟨	🟨				
LED circuit design							🟨	🟨				
RPI testing								🟨	🟨			
Push Button Integration									🟨			
Camera Setup							🟨			🟨		
AI Engine												
Research existing engines		🟩	🟩									
Game Software												
Implement chess game logic				🟩	🟩	🟩	🟩					
Integration												
Integration CV / Chessboard									🟦	🟦	🟦	
AI Integration									🟪	🟪	🟪	
End to end testing										🟪	🟪	🟪
Reports and Presentations												
Proposal presentation	🟥											
Design presentation			🟩									
Midpoint demo									🟦			
Final report										🟦	🟦	
Final presentation & demo												🟨

🟩	Yoorae
🟥	Anoushka
🟨	Demi
🟦	Everyone
🟪	Anoushka and Yoorae

Work Remaining

- Game
  - Test full game
  - Implement retries
- CV
  - Handle Castling
- Final Video / Final Poster
- Final Report