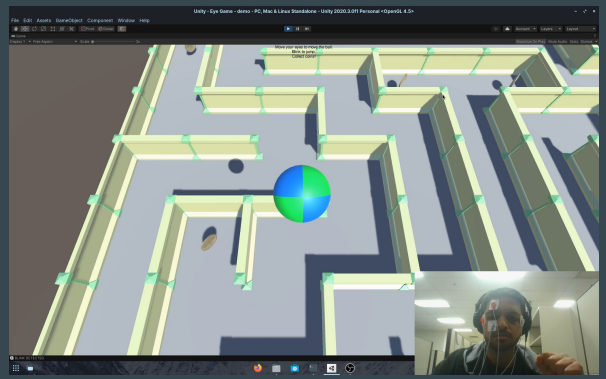
Eye-maze Venture: A project on EOG based detection and gameplay

...

Sarthak Tanwani
UID: 705627719
UCLA Integrated Sensor Lab

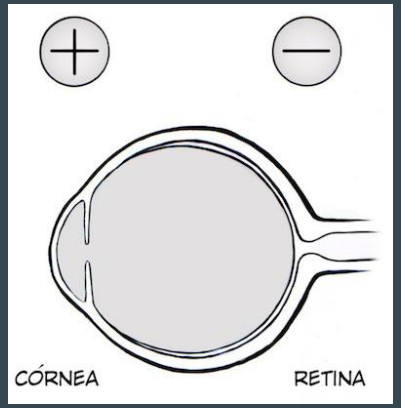
Introduction

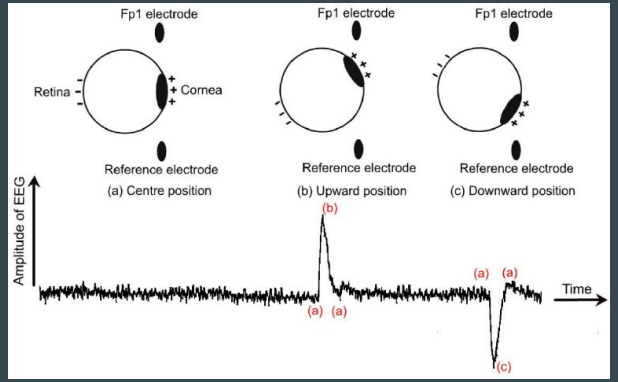
- Methods of input to traditional computer systems is usually limited to a keyboard and a mouse
  - This largely limits the usability of computers for users who do not have fully functioning hands and fingers
  - Evaluate different movement patterns for heterogeneous environments
- Camera based solutions
  - These solutions depend on sophisticated image and signal processing and are usually expensive and inaccessible to the average person
- This project introduces a method of EOG signal processing using a state machine requiring a low memory and processing footprint.
- It also demonstrates the use of these BioSignal classifications in playing a custom game called Eye Maze-Venture.



Properties of the Human Eye

- Human eye is an electrical dipole
  - Cornea is (+) charged
  - Retina is (-) charged
- Potential difference between Cornea and Retina is called Corneal Retinal Potential (CRP)
  - Range: 0.4 mV - 1.0 mV
- The changes in CRP and its trend can be used as a measure of movement and direction of the eye.



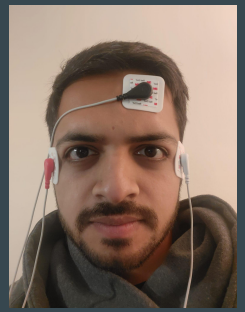


Electro-OculoGraphy (EOG)

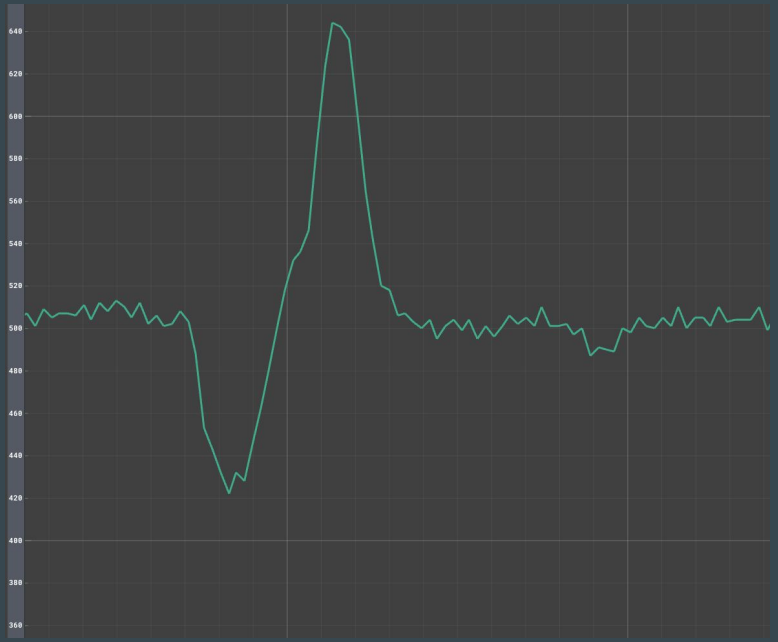
...

ElectroOculoGraphy (EOG)

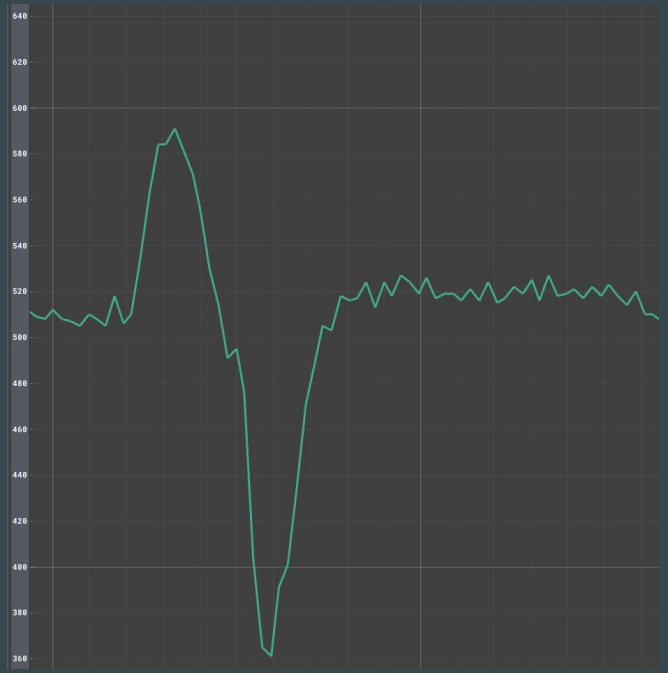
- Electro-oculography (EOG) is a technique used to measure eye movements and eye position.
- EOG works by placing electrodes on the skin near the eyes, without touching the eyes directly, to detect the electrical potential changes that occur as the eyes move.
- It is a non-invasive and painless procedure, and it provides a high-resolution measurement of eye movements and eye position.



Received Signal for Different Movements



Down or Right Movement



Up or Left Movement

Received Signal for Different Movements

Blink Movement

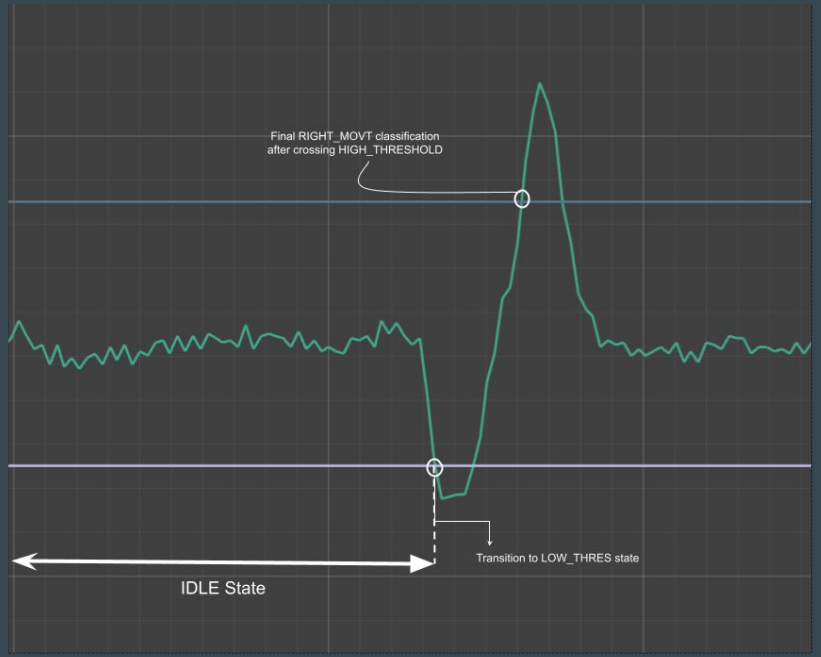
Finite State Machine (FSM)

- Since we need to detect multiple peak in a particular order, a single threshold based system would not be sufficient and we would need a Finite State Machine.
- A state machine is ideally a mathematical model in which a system's activity is shown as a series of states and transitions

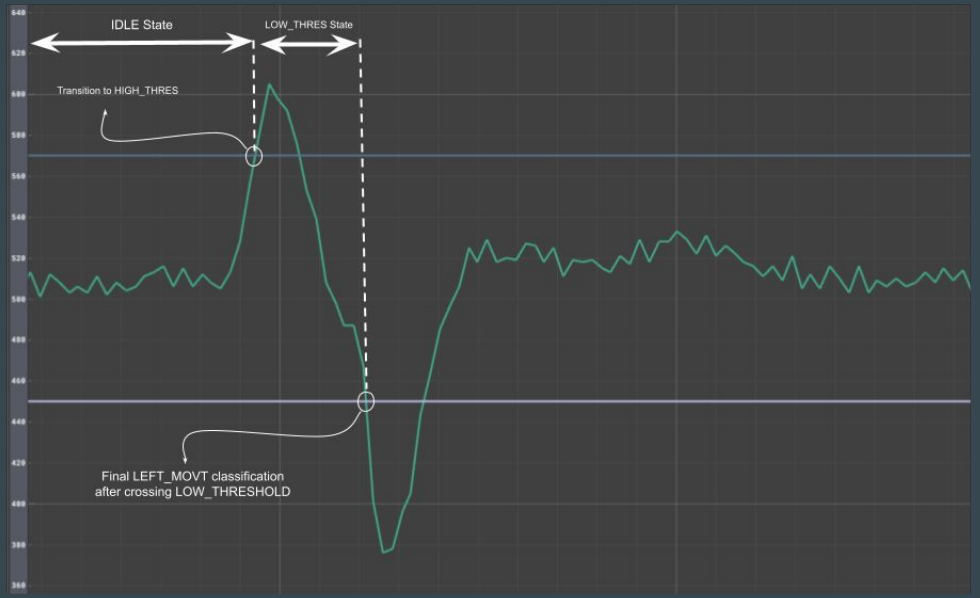
Finite State Machine: States

State	Description
IDLE	Default state with no activity
LOW_THRES	System enters the state when it crosses the low threshold
HIGH_THRES	System enters the state when it crosses the high threshold
BLINK_THRES	System enters the state when it crosses the Blink threshold
MOVT1	Final classification of Movement 1. This could be left or up depending on the placement of electrodes
MOVT2	Final classification of Movement 1. This could be left or up depending on the placement of electrodes

EOG Signal Processing

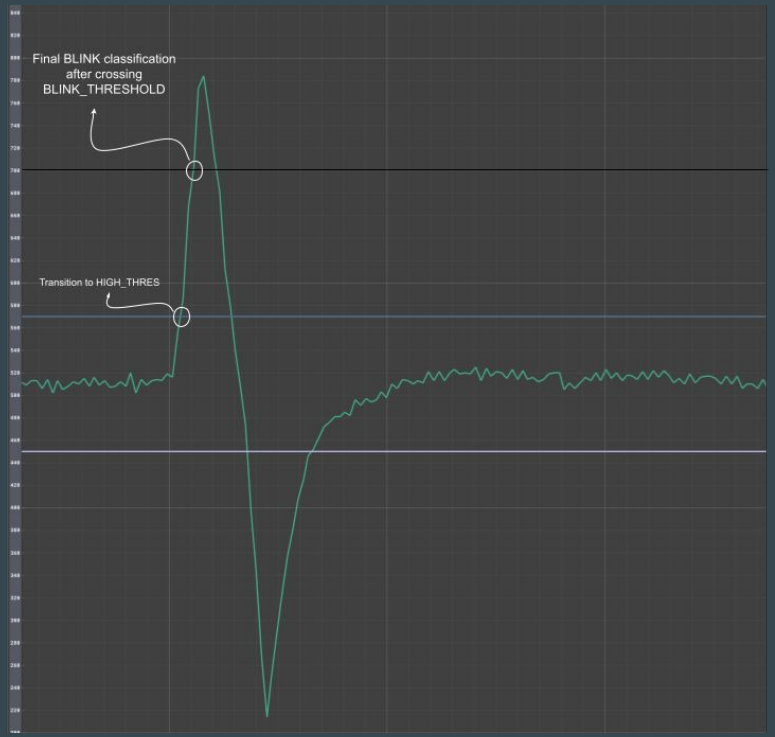


Down or Right Movement



Up or Left Movement

EOG Signal Processing



Blink Movement

State Machine Implementation

Algorithm 1 Algorithm used for the State Machine Classifier

Require: $n \geq 0$
Ensure: Serial Port is connected
  baudrate ← 115200
  while *val* ← NEW SAMPLE **do**
    if *val* is greater than BLINK_THRESHOLD **then**
      state ← *BLINK*
      Stop, Reset, and Start Timer
    else if *val* is greater than BLINK_THRESHOLD but less than HIGH_THRESHOLD **then**
      if *last_state* is LOW_THRES and time since last movement is less than 500 ms **then**
        state ← *RIGHT_MOVT*
        Stop, Reset and Start Timer
      else if *last_state* is IDLE **then**
        state ← *HIGH_THRES*
      end if
    else if *val* is less than LOW_THRESHOLD **then**
      if *last_state* is HIGH_THRES and time since last movement is less than 500 ms **then**
        state ← *LEFT_MOVT*
        Stop, Reset and Start Timer
      else if *last_state* is IDLE **then**
        state ← *LOW_THRES*
      end if
    end if
  end while

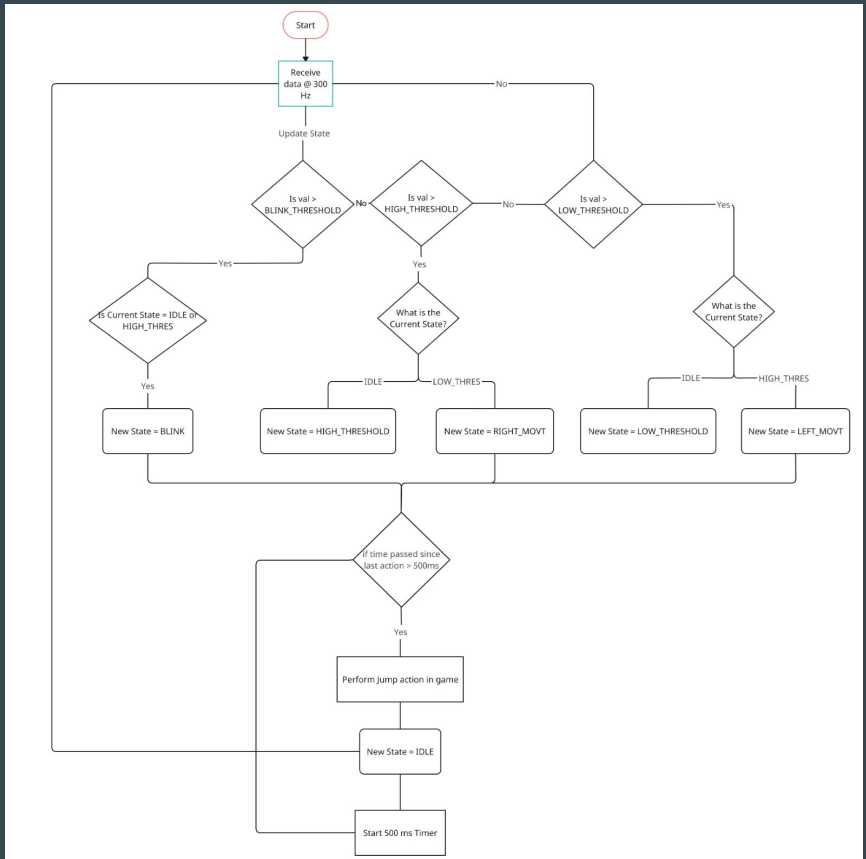
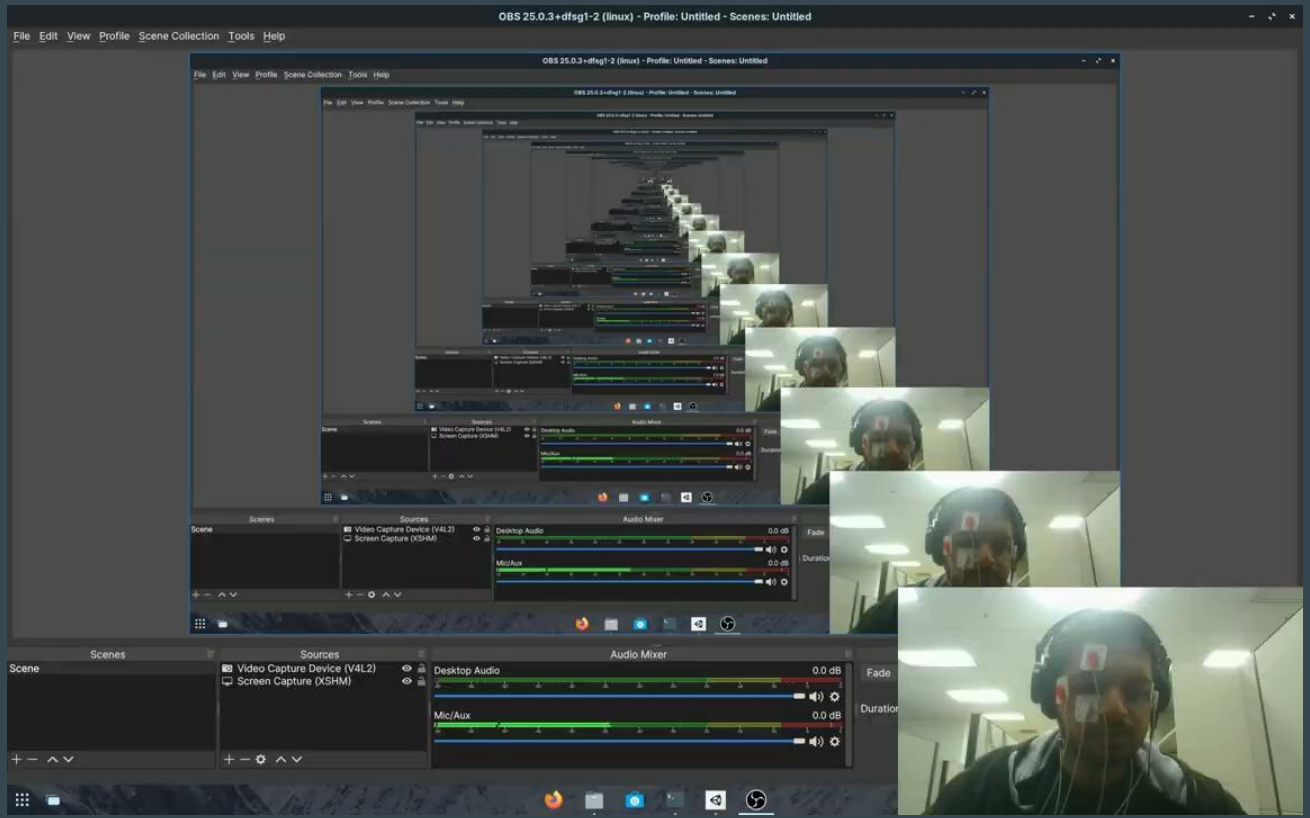


Fig. 10. State Machine Diagram

Demo Video

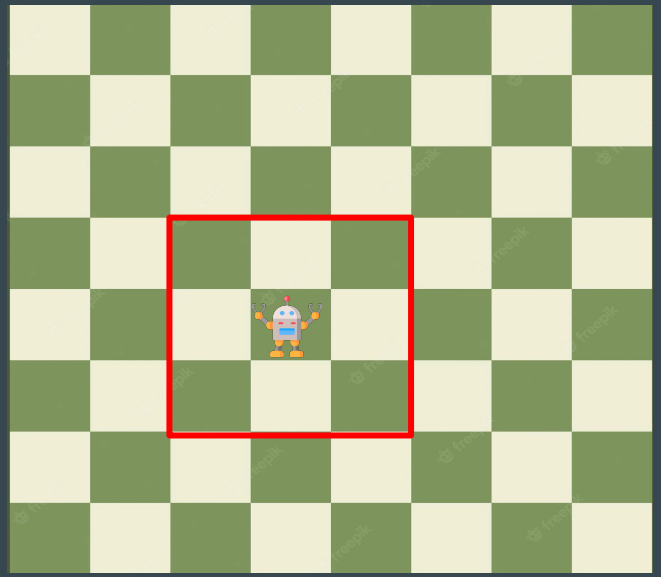
- 
OBS 25.0.3+dfsg1-2 (linux) - Profile: Untitled - Scenes: Untitled
File Edit View Profile Scene Collection Tools Help
Scenes
Scene
Sources
Video Capture Device (V4L2)
Screen Capture (XSHM)
Audio Mixer
Desktop Audio
0.0 dB
Mic/Aux
0.0 dB
Fade

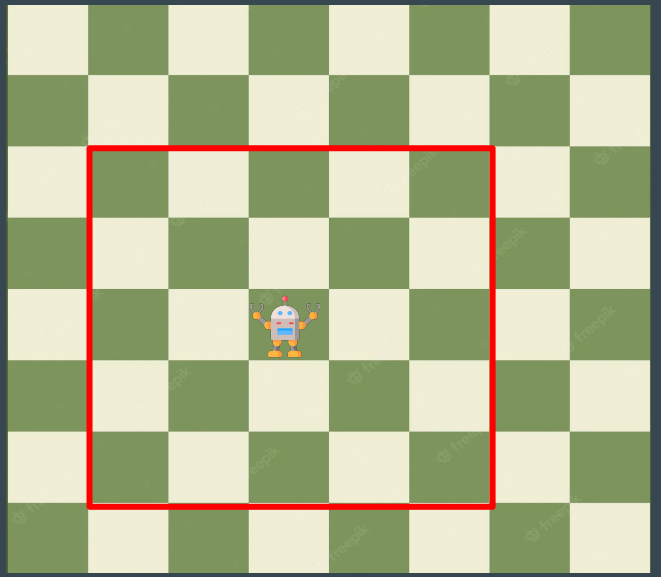
References

1. Author, F.: Article title. Journal 2(5), 99–110 (2016)
2. Bott NT, Lange A, Rentz D, Buffalo E, Clopton P and Zola S (2017) Web Camera Based Eye Tracking to Assess Visual Memory on a Visual Paired Comparison Task. Front. Neurosci. 11:370. https://doi.org/10.3389/fnins.2017.00370
3. Mehrubeoglu, Mehrube Pham, Linh Le, Hung Muddu, Ramchander Ryu, Dongseok. (2011). Real-time eye tracking using a smart camera. Proceedings - Applied Imagery Pattern Recognition Workshop. 1-7. https://doi.org/10.1109/AIPR.2011.6176373.
4. Front. Neurosci. 11:370. https://doi.org/10.3389/fnins.2017.00370
5. 3. Mehrubeoglu, Mehrube
6. Pham, Linh
7. Le, Hung
8. Muddu, Ramchan-
9. der
10. Ryu, Dongseok. (2011). Real-time eye tracking using a smart cam-
11. era. Proceedings - Applied Imagery Pattern Recognition Workshop. 1-7.
12. https://doi.org/10.1109/AIPR.2011.6176373.
13. 4. S. Yathunanthan, L. U. R. Chandrasena, A. Umakanthan, V. Vasuki and S. R. Mu-
14. nasinghe, ”Controlling a Wheelchair by Use of EOG Signal,” 2008 4th International
15. Conference on Information and Automation for Sustainability, Colombo, Sri Lanka,
16. 2008, pp. 283-288, https://doi.org/10.1109/ICIAFS.2008.4783987.
17. 5. M. Merino, O. Rivera, I. Gómez, A. Molina and E. Dorronzoro, ”A Method of
18. EOG Signal Processing to Detect the Direction of Eye Movements,” 2010 First
19. International Conference on Sensor Device Technologies and Applications, Venice,
20. Italy, 2010, pp. 100-105, https://doi.org/10.1109/SENSORDEVICES.2010.25.
21. 6. N. M. M. Noor and M. A. M. Mustafa, ”Eye movement activity that affected the eye

System Parameters - Sense

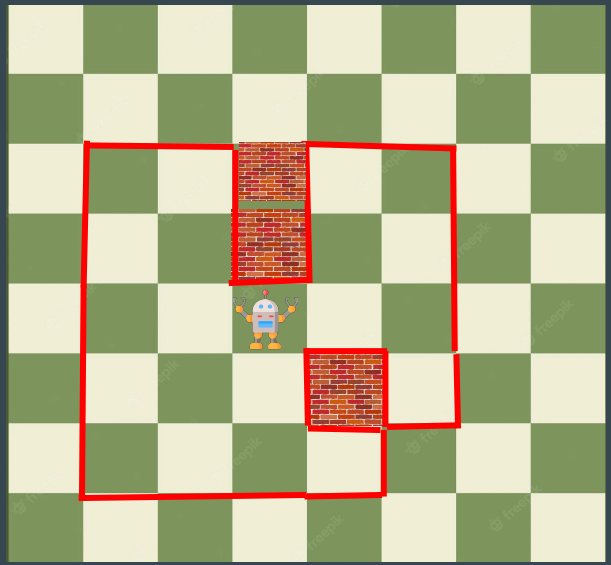
- Configurable and finite sensor window





System Parameters - Sense

- Landmarks shadowing other states in the sensor window or movement validity



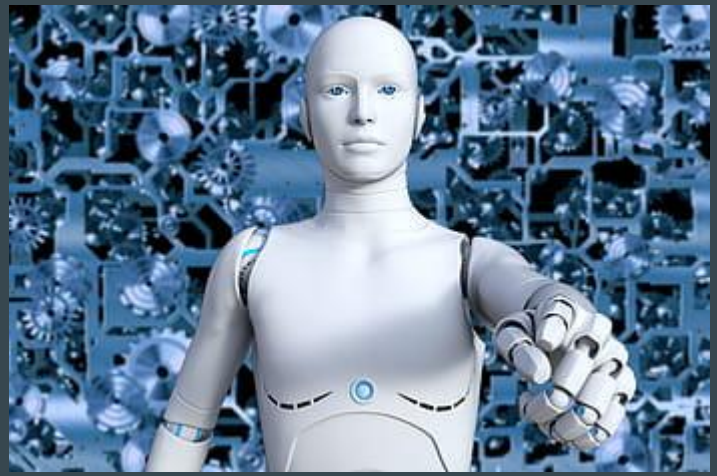
System Parameters - Sense

- Data array: stores the relative distances of landmarks from robot position and motion taken over period of time
- Use data array to reconstruct map of world: form constraint matrices and vectors in SLAM function

```
Measurements:  [[0, 2.9737532967831535, -2.2905647497524755]]
Motion:  [1, 2]
```

System Parameters - Move

- Initial configuration of movement
  - Move farthest from known states
  - Heuristic: find least amount of steps taken by robot to explore whole map
- Adaptation of initial configuration
  - Move randomly* with a 1% chance of noise in movement
  - Pairs of movement have been configured initially - robot can not move farther than two states away

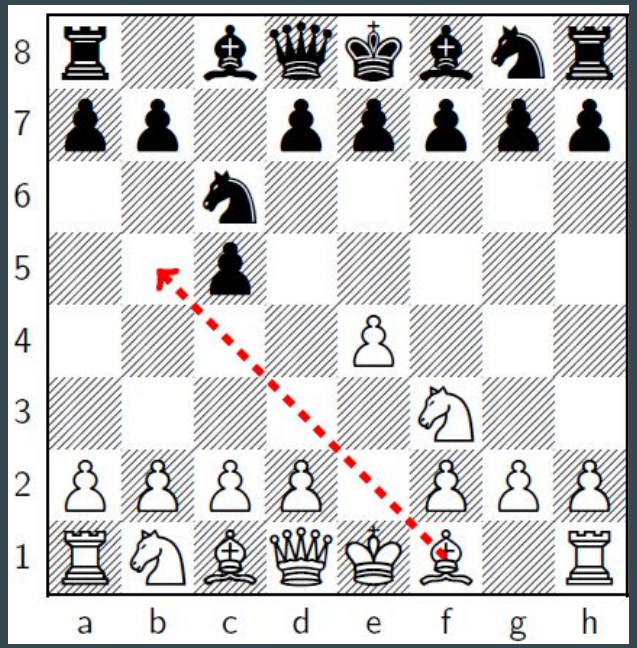


Omega Matrix and Xi Vector

- Iterate through data array => update omega and chi to account for all measurements and motions
- Omega matrix consists of all the poses and landmarks
- Xi is a column vector that includes the constraints for each motion from one state to another.
- Every movement is represented as a numerical relationship in these matrices.

$$\mu = (\Omega^{-1})\,\chi$$

Omega Matrix and Xi Vector Example



	x1	x2	y1	y2	L1	L2
x1	1	-1				
x2	-1	1				
y1			1	-1		
y2			-1	1		
L1						
L2						

Omega

4	x1
-4	x2
-4	y1
4	y2
	L1
	L2

Xi

$$\mu = (\Omega^{-1})\,\chi$$

Our Graph SLAM Results

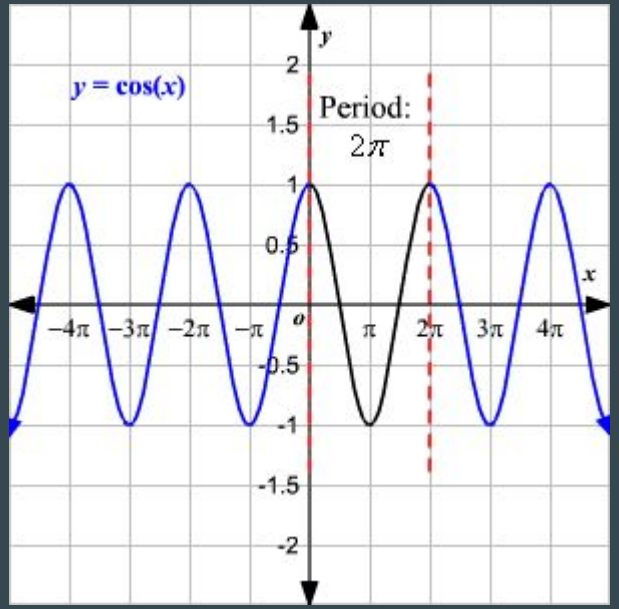
Map	True Landmarks vs Graph SLAM Estimated Landmarks
Circle	[0,0] [0,1] [0,0] [0,-1] [0,0] [1,0] [0,0] [0,0] [0,-1] [0,0] [1,1] [0,0]
Corridor	[0,0] [-1,-1] [0,0] [0,-1] [0,0] [0,-1] [0,-1] [0,-1] [0,0] [0,0] [-1,0] [0,-1] [0,0] [-1,-1]
Random	[-1,-1] [-1,-2] [-2,-1] [-2,0] [-2,0] [-2,-1] [-1,0] [-1,0] [-2,-1] [-2,-1]

Our Graph SLAM Results

Map	True Landmarks vs Graph SLAM Estimated Landmarks
Heart	[2,1] [2,1] [1,1] [3,2] [3,2] [2,1] [2,1] [2,1] [2,1] [2,1] [2,1]
Sticky	[3,1] [3,2] [3,1] [3,1] [3,1] [3,1] [2,1] [2,1] [2,1] [2,1]
Maze	[0,0] [0,0] [0,0] [0,0] [0,0] [0,0] [0,0] [0,0] [0,0] [0,0]

SLAM Algorithms - Extended Kalman Filter (EKF)

- Extension of classic Kalman Filter
  - For nonlinear systems utilizing Taylor expansion (most real world problems)
    - Versus Gaussian Distribution and Linear Functions
- Considered sensor fusion algorithm
  - Uses many inputs from different sensors that work better than the estimate obtained by only one measurement



Source

EKF SLAM

- Recursive method for state estimation in 2 steps:
  - Prediction - sensors move
  - Correction - sensors observe landmarks in environment that had been previously mapped
- Models SLAM using single EKF of poses, landmarks, and covariance between each position and landmarks
  - Single estimate of pose tracked over time => confidence in pose tracked by covariance matrix
- Predict State Update:
  - Noise is added to the control signals (linear and angular velocity)
  - Next predicted state found by: adding change in orientation in time to linear and angular velocity
  - Predict covariance: add state covariance to current uncertainty of EKF
    - Each time step uncertainty in system grows by the covariance of the pose

$$X = FX + BU$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta t cos(\theta) & 0 \\ \Delta t sin(\theta) & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} v_t + \sigma_v \\ w_t + \sigma_w \end{bmatrix}$$

EKF SLAM

- Update:
  - Observations of nearby landmarks used to correct location estimate
  - Kalman Gain is added to the pose update
    - Measurement covariance << current estimate => Kalman gain will be 1 (completely believing measurement)
    - Measurement covariance >> current estimate => Kalman gain will go to 0 (high trust in process and little trust in measurement)
  - Covariance matrix updates

$$y = z_t - h(X)$$

$$K = \bar{P}_t H_t^T (H_t \bar{P}_t H_t^T + Q_t)^{-1}$$

$$xUpdate = xEst + (K * y)$$

EKF SLAM Results

Map	True Landmarks vs EKF SLAM Estimated Landmarks
Circle	*added additional 2 landmarks [0,0] [0,0] [0,-1] [-3,-2] [-5,2] [0,2] [-1,1] [-1,0] [0,3] [-1,1] [5,6] [6,0]
Corridor	*added additional 6 landmarks [0,0] [0,0] [0,3] [2,1] [2,4] [0,-1] [0,2] [1,0] [2,4] [2,0] [2,3] [2,2] [1,2] [2,0] [7,6] [9,7] [9,8] [3,3] [2,2] [1,3]
Random	*Two landmarks mapped as two each [0,0] [0,0] [0,0] [0,0] [0,1/-1] [0,0] [0,0] [0,0] [0,0/-1] [0,0]

EKF SLAM Results

Map	True Landmarks vs EKF SLAM Estimated Landmarks
Heart	*added one additional landmark [1,-2] [1,0] [1,2] [1,2] [1,2] [0,0] [2,-1] [1,-1] [1,0] [0,0] [2,-1] [2,4]
Sticky	*Added one additional landmark [0,1] [0,1] [1,1] [1,1] [1,1] [1,1] [0,1] [1,3] [1,3] [1/4,3/2]
Maze	*Added one additional landmark [0,0] [0,0] [3,-1] [1,1] [4,-1] [0,-2] [0,-1] [-1,-2] [-2,-1] [2,-3]

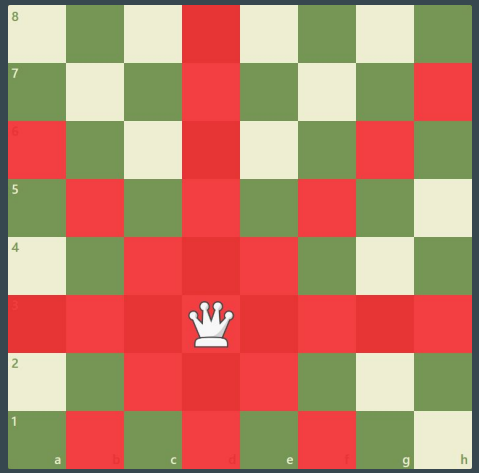
Path Planning

...

Movement Patterns

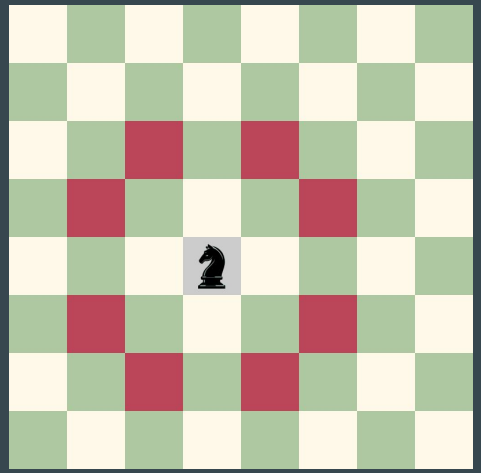
Land

- Limited by sensor window
- Cannot move if obstacle is in the way



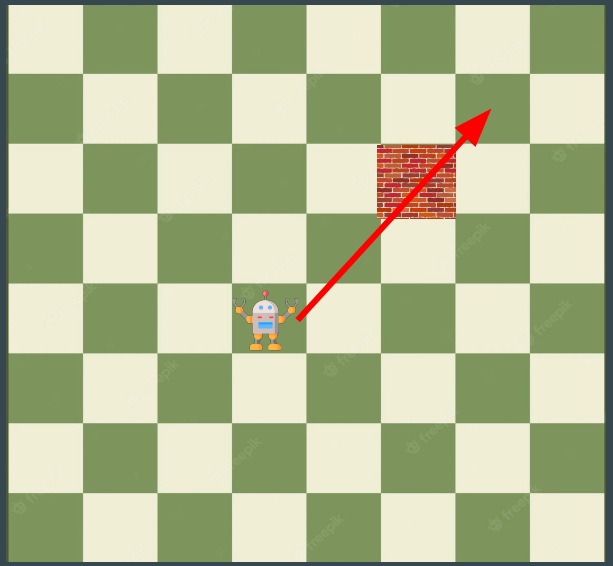
Air

- Limited by sensor window
- Can jump over obstacles



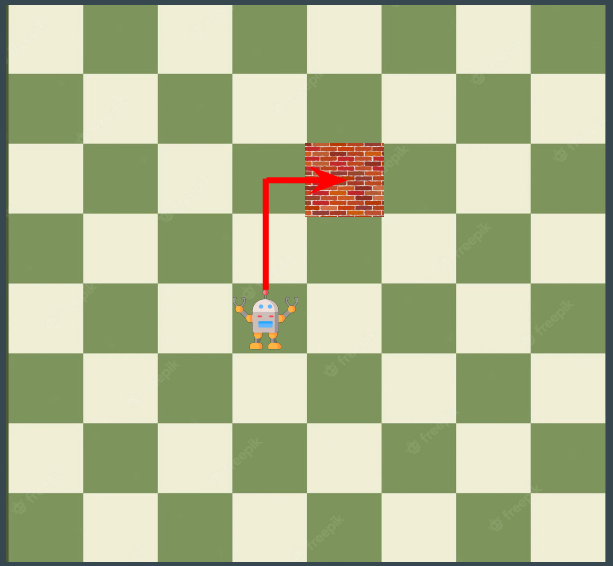
Movement Validity - Land

- Check if obstacle is in the desired path
  - Line is made, and check if obstacle is within the path



Movement Validity - Air

- Check if obstacle is in the desired destination



Transition Probability

$$P(s \mid s_0, a) = \begin{cases} 0.9 & \text{s.t } s[0] = s_0[0] + a[0] \text{ and } s[1] = s_0[1] + a[1] \\ 0.1/n & \text{s.t } n = \text{number of valid squares around destination} \\ 0 & \text{if obstacle is in the way (land) or in the destination (air)} \end{cases}$$

Reward Function

1. Goal state has an arbitrary non-zero value
2. Every other state, including obstacles, have a reward of 0

$$R(s, s_0, a) = \begin{cases} 10 & \text{s.t } s = \text{ Goal State} \\ 0 & \text{s.t } s \mathrel{!}= \text{Goal State} \end{cases}$$

Value Iteration

$$init : V_0^*(s) = 0$$

$$loop : V_{i+1}^*(s) = \max_a \sum_{s'} P(s,a,s')[r(s,a,s') + \gamma V_i^*(s)]$$

$$until\ either\ i = H,\ or\ convergence\ of\ V*\ for\ H = inf$$

- Testing for convergence by checking if the difference between the current time step's maximum value and the previous time step's maximum value is under a tight threshold
  - Signifies that hardly any improvement to the policy, if any, is possible at that point in time.

Path Planning - Testing Methodology

- Previously completed SLAM Testing steps
  - Define landmarks to describe a map with a particular characteristic
  - Run path planning on map resulted from graph slam, EKF slam, and ground truth (no slam)
- Value iteration until convergence?
- Steps taken to reach goal?

Path Planning Results

		Iterations to Converge (Land/Air)	Steps (Land/Air)
Circle	True	47/31	5/6
	Graph	47/32	7/6
	EKF	47/31	7/6
Corridor	True	48/32	8/6
	Graph	47/32	7/8
	EKF	47/32	5/6

Path Planning Results

		Iterations to Converge (Land/Air)	Steps (Land/Air)
Random	True	46/31	5/6
	Graph	47/32	6/6
	EKF	46/31	5/6
Sticky	True	47/31	6/6
	Graph	47/32	7/6
	EKF	46/31	6/6

Path Planning Results

		Iterations to Converge (Land/Air)	Steps (Land/Air)
Heart	True	47/32	7/6
	Graph	47/31	6/6
	EKF	47/31	7/8
Maze	True	47/31	7/6
	Graph	47/31	7/6
	EKF	47/32	6/6

Conclusion

- SLAM performs better when landmarks are closer to initial state of robot
  - Random movement, substantial noise are primary concerns
  - Accuracy improves when landmarks are frequently seen
- In 10x10 world, number of steps taken with either land or air movement is comparable
- Future work - create bigger worlds to better understand the benefits of each movement

		Iterations to Converge (Land/Air)	Steps (Land/Air)
Horizontal	True	49/47	25/14
Vertical	True	49/48	26/14