VISTA

Vision-based Interactive System for Touch-free Applications

PROBLEM VS SOLUTION

PROBLEM: INACCESSIBILITY

- Tight spaces
- On the go
- Disability
- More intuitive and immersion compared to mainstream mouse keyboard

SOLUTION: A TOUCH-FREE INTERFACE

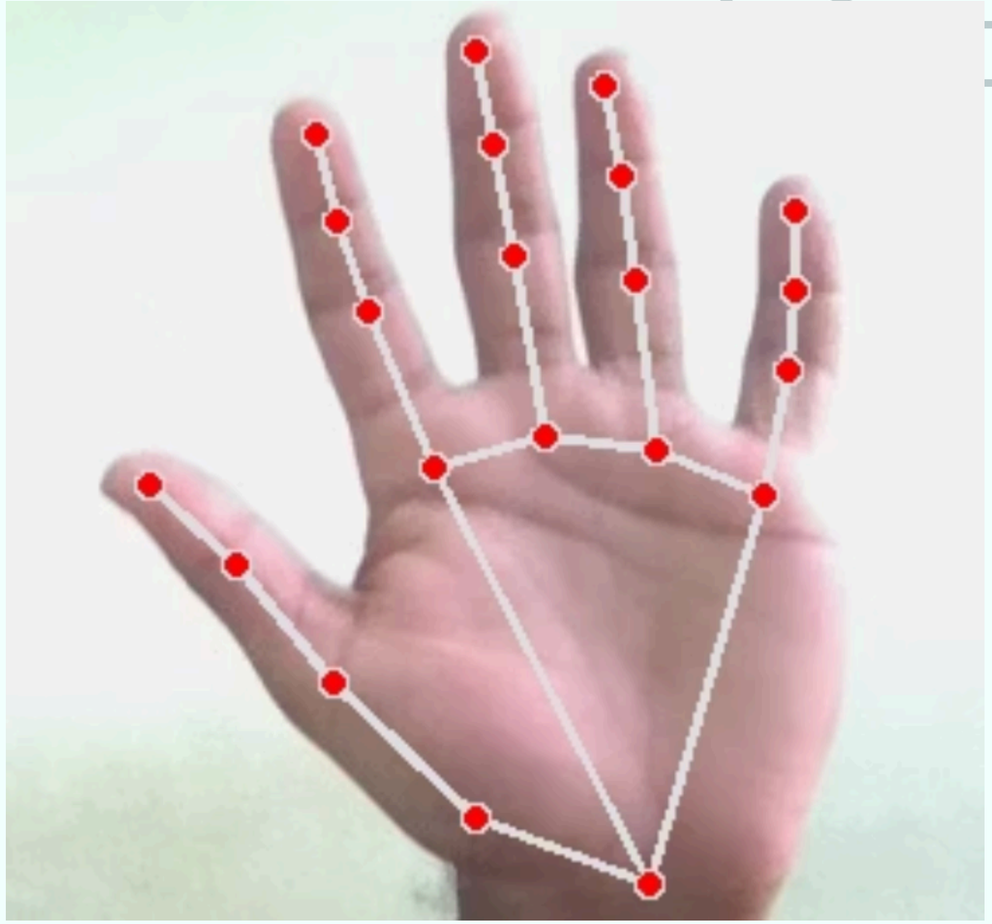
- **Hand Gesture Recognition**
  - Real-time recognition for keyboard and mouse clicks.
- **Eye Movement Tracking**
  - Gaze-based cursor control for precise pointer movement.

LIBRARIES USED

- OPENCV (CV2): PRIMARY COMPUTER VISION LIBRARY FOR IMAGE PROCESSING AND WEBCAM INTEGRATION
- MEDIAPIPE: GOOGLE'S FRAMEWORK FOR HAND TRACKING, FACIAL MESH DETECTION, AND LANDMARK ESTIMATION
- SCIKIT-LEARN: MACHINE LEARNING LIBRARY USED FOR GESTURE CLASSIFICATION WITH MLPCLASSIFIER
- NUMPY: NUMERICAL COMPUTING FOR EFFICIENT ARRAY OPERATIONS AND MATHEMATICAL CALCULATIONS
- PYAUTOGUI: SYSTEM AUTOMATION LIBRARY FOR SIMULATING KEYBOARD AND MOUSE EVENTS
- PANDAS: DATA MANIPULATION AND ANALYSIS FOR DATASET HANDLING
- JOBLIB: MODEL SERIALIZATION AND PERSISTENCE
- TIME: FOR FPS COUNT/CALIBRATION
- SKLEARN.LINEAR_MODEL.LINEARREGRESSION: FOR MAPPING GAZE TO SCREEN COORDINATES
- TRANSFORMERS{HUGGING FACE'S TRANSFORMER LIBRARY}: APPLIES DEEP LEARNING TO CORRECT GRAMMAR IN A GIVEN SENTENCE. IT IS THE FIRST STEP IN CORRECTION PROCESS.
- LANGUAGE_TOOL_PYTHON: RULE-BASED GRAMMAR AND SPELLING CHECKER. USED FOR POLISHING AFTER THE TRANSFORMER MODEL'S CORRECTION.

HOW MEDIAPIPE PLOTS THE 21 HAND LANDMARKS

1. **Detects hand** using a palm detection model (bounding box).
2. **Crops and resizes** the hand region to a fixed size.
3. **Runs a landmark model** to predict 21 (x, y, z) coordinates of the hand.
4. **Returns landmarks** in normalized values (0–1) relative to the image size.
5. **Landmarks follow a fixed index order** from wrist to fingertips.
6. **Plots landmarks and connects lines** (bones) between specific points for visualization.
7. **Supports left/right hand detection** and tracking across frames.



01.

DATA COLLECTION AND PREPARATION

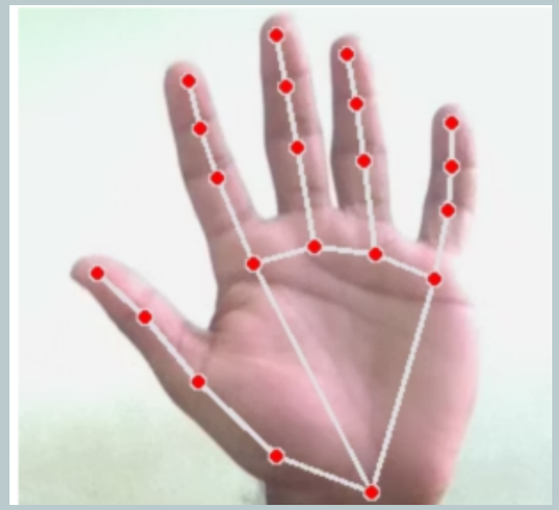
PHASE-1

DATA COLLECTION AND PREPARATION

The project implements a comprehensive data collection system with three distinct phases:

Right-Hand Gestures (A-Z):

- 26 alphabetical characters
- 150 samples per character
- Single-hand landmark detection



Left-Hand Gestures (0-9 + Punctuation):

- 10 numerical digits
- 3 punctuation marks (., , ?)
- 150 samples per gesture

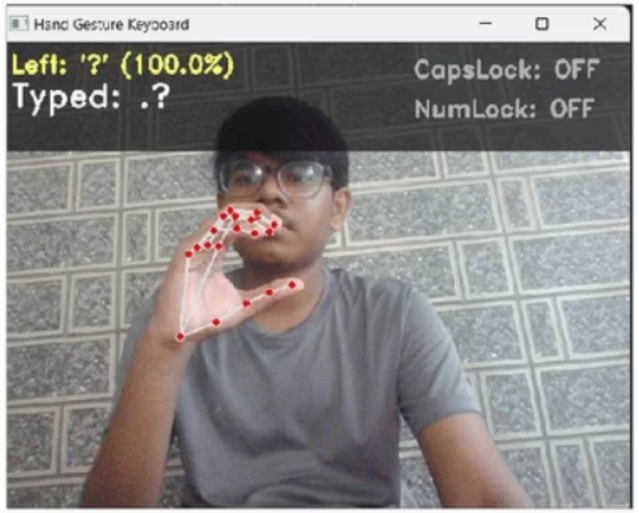
Dual-Hand Control Gestures:

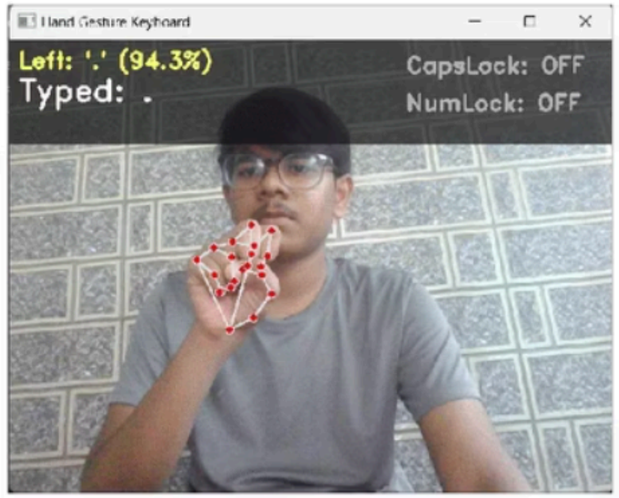
- 7 control commands: space, enter, backspace, right-click, left-click, capslock, numlock
- Simultaneous two-hand tracking required
- 150 samples per command


Dual: 'enter' (93.7%)
Typed:
CapsLock: OFF
NumLock: OFF


Right: 'Q' (99.5%)
Typed:
CapsLock: OFF
NumLock: OFF


Left: '3' (100.0%)
Typed: 3333
CapsLock: OFF
NumLock: OFF


Hand Gesture Keyboard
Left: '?' (100.0%)
Typed: .?
CapsLock: OFF
NumLock: OFF


Hand Gesture Keyboard
Left: '.' (94.3%)
Typed: .
CapsLock: OFF
NumLock: OFF

02.

GRAMMAR CHECK

PHASE-2

GRAMMAR AND SPELL CHECK

Spell Checking (raw_spell_check):

- Splits input into words and identifies misspelled words using SpellChecker.
- Replaces incorrect words with most likely corrections.

Grammar Correction (grammar_correct):

- Tokenizes the sentence using HuggingFace AutoTokenizer
- Uses pre-trained Transformer model (prithivida/grammar_error_correcter_v1) to correct grammar
- Decodes the output into a clean, corrected sentence.

GRAMMAR AND SPELL CHECK

Combined Correction (correct_sentence):

- First applies spell check.
- Then corrects grammar on the spell-checked result.

Execution in spell_check() function:

- Extracts the latest sentence (before the last dot) from typed text.
- Applies correct_sentence() on it.
- Deletes the old sentence using simulated backspaces (pyautogui.press('backspace')).
- Re-types the corrected sentence character by character.
- Ensures sentence ends with a period

03.

GAZE BASED CURSOR CONTROL

PHASE-3

GAZE BASED CURSOR CONTROL

GAZE VECTOR CALCULATION

- Uses pupil and eye corner landmarks to calculate the direction of gaze.
- Normalizes by eye width to be scale-independent.
- Applies a deadzone filter to ignore small jitters.

CALIBRATION PROCESS

- Guides the user to look at specific screen points (green dots).
- Records average gaze vectors for each known screen point.

MODEL TRAINING

- Trains two linear regression models:
    - model_x: maps gaze vectors to X screen coordinates.
    - model_y: maps gaze vectors to Y screen coordinates.

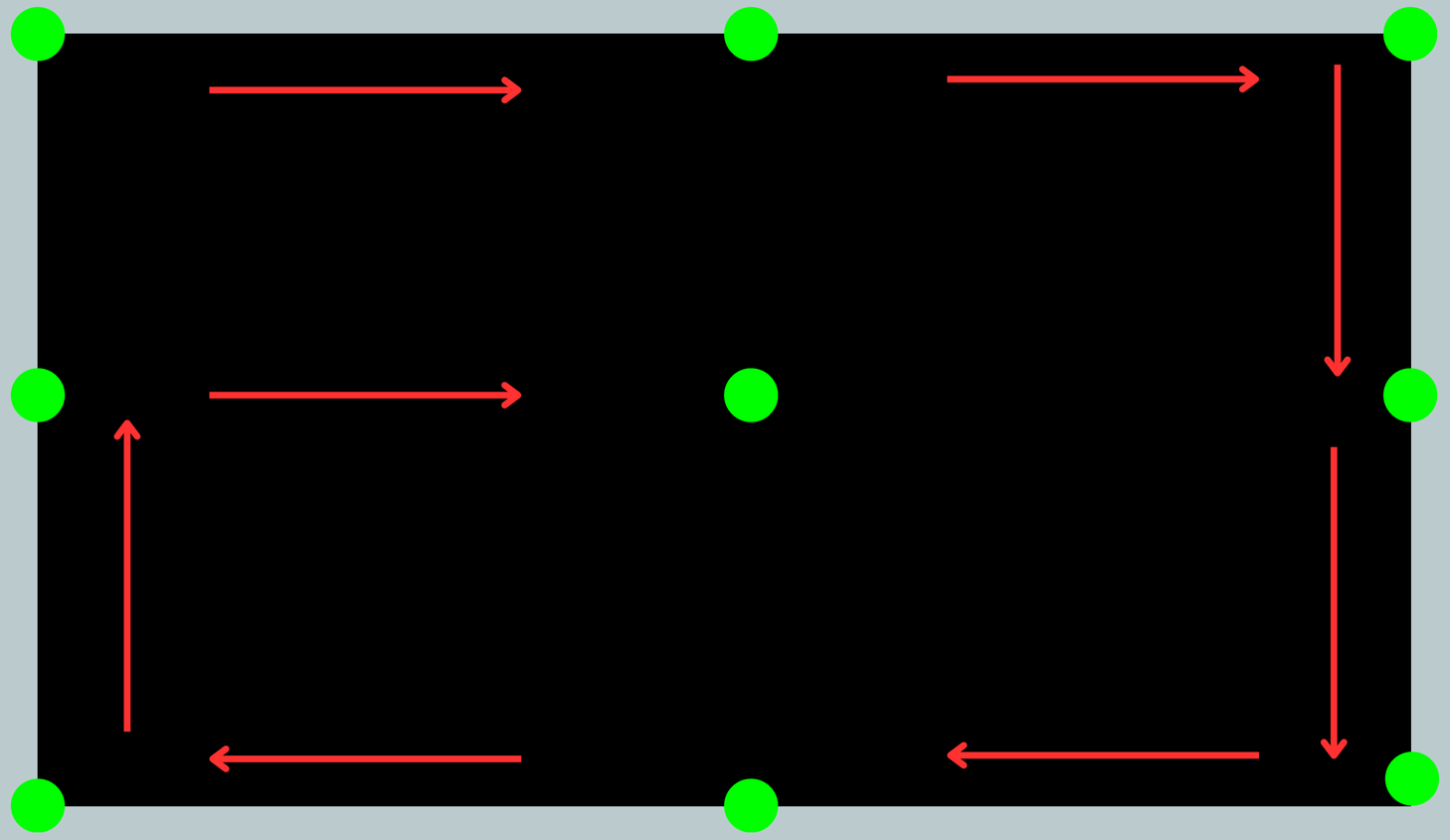
MAIN LOOP

- Captures video frames in real-time.
- Computes gaze vector → smooths it → predicts screen location.
- Moves the mouse cursor using pyautogui.moveTo().
- Overlays visual feedback:
    - A yellow dot where the model thinks you are looking.
    - A yellow line representing the direction of gaze.
    - FPS counter.

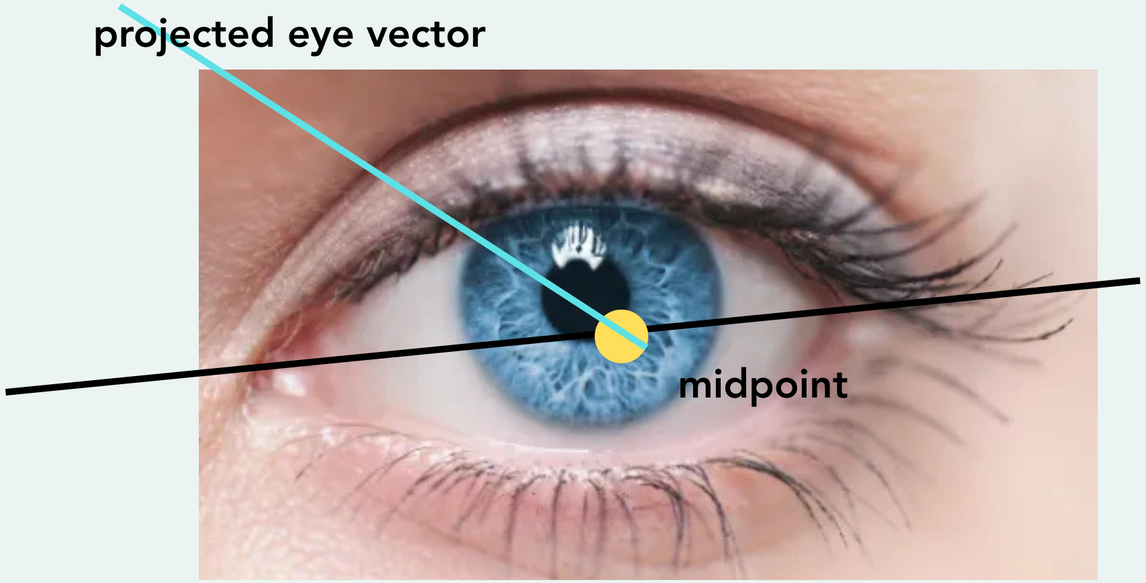
- Gaze vector - horizontal mapping
- Relative eyelid positioning - vertical mapping as the pupil doesn't have much visible vertical movement

CALIBRATION

- Guides the user to look at specific screen points (green dots).
- Records average gaze vectors for each known screen point.



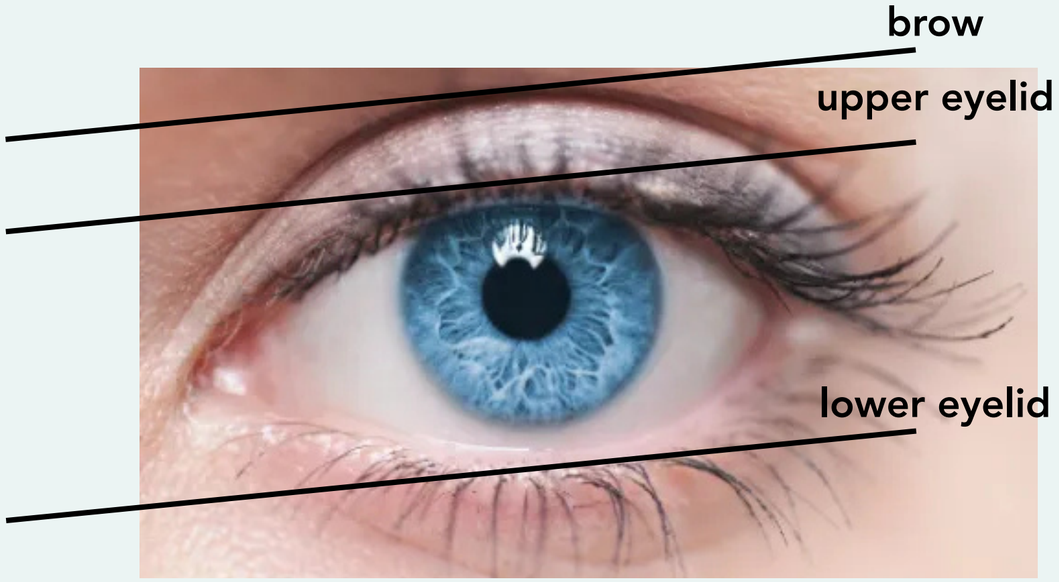
HORIZONTAL GAZE



- We are finding the mid pint of the eye by mapping the
- Left inner corner and left outer corner so pupil moves relative to centre to track the vector
- Then we can normalize wrt the eyewidth

combined gaze vector

VERTICAL GAZE



- Blinking is rejected because it occurs to quickly and fits as an outlier
- can cause cumulative error future work involves much closer eye observation and error reduction

- People's faces have different sizes, and their distance from the camera can vary.
- Normalizing eye openness by face height makes the measurement scale-independent and comparable across different frames and people.

```
brow (landmark 10)
|
|
upper eyelid (159)
lower eyelid (145)
|
|
chin (landmark 152)
```

04.

FUTURE WORK & PROBLEMS FACED

PHASE-4

PROBLEMS FACED

- Recalibration is necessary for each additional user to ensure accuracy.
- Spellcheck tends to be slower and less accurate on low-context sentences.
- Mouse positioning may suffer from cumulative drift over time.
- Calibration errors can also impact mouse accuracy.

Following shall be taken into account for future works

FUTURE APPLICATIONS

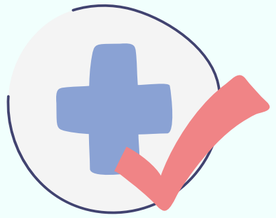
HEALTHCARE:
Hygienic interfaces in medical environments

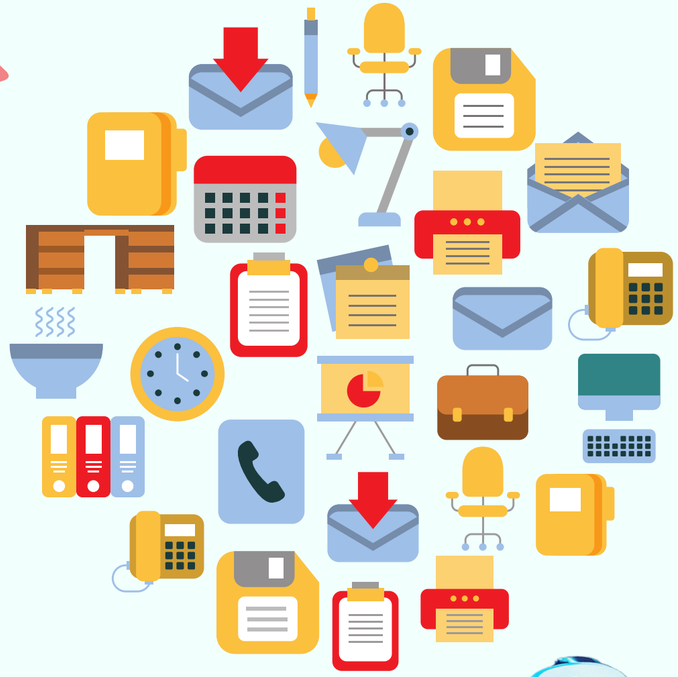
ACCESSIBILITY:
Assistive technology for disabled users

INDUSTRIAL:
Touch-free control in manufacturing environments

EDUCATION:
Interactive learning systems without physical contact

GAMING:
Immersive gesture-based gaming experiences





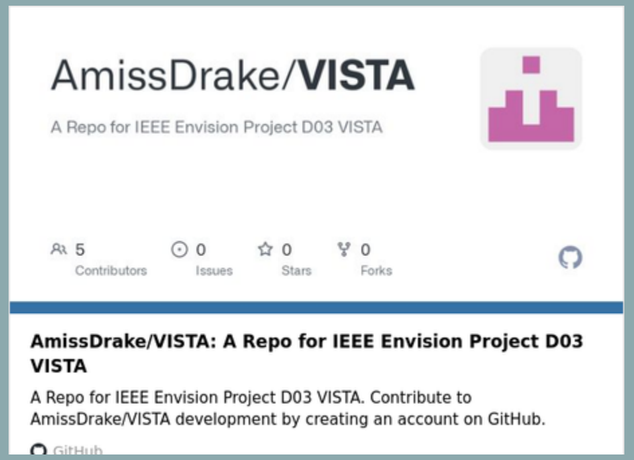
TEAM DETAILS

MENTORS

- Ashmita Das
- Ashmit R Sambrani
- Sahasra Pulumati
- Ranjit Tanneru

MENTEES

- Jashwanth R
- Jaydeep Rathva
- Rohini V
- Sai Easwar
- Sumedh V Bhat



THANK YOU

Do you have any question?

:)