**Title:**

**NewsEye Navigator: An Interactive Eye-Tracking and Gesture-Controlled Global News Explorer.**

**Project Overview:**

NewsEye Navigator is an innovative application that combines eye-tracking and gesture-tracking technology with global news exploring. The application displays a world map and uses eye gaze and hand gesture detection to allow users to interact with the map and explore news from different continents. While eye-tracking and gesture-based technologies exist in various domains, the combination of these technologies for global news exploration is relatively novel. The unique aspect of NewsEye Navigator lies in its integration of eye-tracking and hand gestures specifically for geographical news exploration, providing an intuitive and immersive user experience.

**Inputs:**

* Eye movement data captured by a camera
* Hand gesture data captured by a camera
* Real-time news data fetched from RSS feeds

**Outputs:**

* Selected continent based on eye gaze
* News headlines for the selected continent

**Contribution:**

|  |  |  |
| --- | --- | --- |
|  | Sankeerth’s Contribution | Rohith’s Contribution |
| Eye-gaze detection |  | 100% |
| Hand-gesture detection | 100 % |  |
| Python Application | 50% | 50% |

**Approach:**

**Share the details of the algorithms you used.**

**Eye Tracking Algorithm**

1. Face Detection: Used dlib's frontal face detector to locate faces in the video frame.
2. Facial Landmark Detection: Employed dlib's shape predictor to identify 68 facial landmarks, particularly focusing on the eye regions.
3. Eye Aspect Ratio (EAR) Calculation: Computed the eye aspect ratio to determine if eyes are open or closed, which is used for blink detection and click actions.

**Hand Tracking Algorithm**

1. Hand Detection: Utilized the MediaPipe Hands solution for hand detection (Blazepalm model for palm detection, hand landmark detection model processing the cropped image region detected by the palm detector)
2. Hand Landmark Detection: Employed Mediapipe’s machine learning models to detect hands and estimate 21 3D landmarks of each hand.
3. Gesture Detection: From the calculated landmark points, angle and distance is calculated between points for unique gestures.

**Architecture**

**A diagram of a product

Description automatically generated with medium confidencePalm Detection Model Hand Landmark Model**

(Zhang A diagram of a number

Description automatically generatedet al., 2020)

**What aspects of the algorithms you have coded on your own:**

1. **eye\_aspect\_ratio method:** This function calculates the eye aspect ratio, which is used to determine if an eye is open or closed.
2. **detect\_face\_and\_eyes method:** This function detects the face and eyes in a given frame, using dlib's face detector and shape predictor.
3. **move\_cursor method:** This function calculates the cursor movement based on the detected face center and moves the cursor accordingly.
4. **process\_frame method:** This is the main processing function that combines face detection, eye tracking, blink detection, and cursor movement. It also handles the visualization of eye landmarks and text overlay on the frame.
5. **blink detection logic:** The code for detecting left and right eye blinks, including the duration calculation and click actions, is manually implemented within the process\_frame method.
6. **get\_angle:** This function calculates the angle between three points (a, b, c) in 2D space.
7. **get\_distance:** This function calculates the Euclidean distance between two points and normalizes it
8. find\_finger\_tip: This method locates the tip of the index finger from the processed hand landmarks.
9. move\_mouse: This method moves the mouse cursor based on the position of the index finger tip
10. is\_left\_click: This function detects a left-click gesture based on hand landmark positions. (Angle of the index finger is bent, angle of the middle finger is straight and distance between thumb and index finger are apart)
11. is\_scroll\_up: This function detects scroll up based on hand landmark positions (Index and middle fingers are folded in a small angle and thumb and index finger are apart)
12. is\_scroll\_down: This function detects scroll down based on hand landmark positions ( all fingers including the thumb is bent i.e the distances are small)
13. detect\_hand\_gesture: Using pyautogui and pynput, the mouse actions are performed based on the specific gesture when trues
14. All the workings of the ContinentNewsFetcher and ContinentMapApp

**What aspects of the algorithms have you used from the online resources? Cite them appropriately.**

**Dlib's face detector and shape predictor:** The project utilizes dlib's frontal face detector and shape predictor for facial landmark detection.

Mediapipe Hands: Mediapipe’s hand model is used for detecting 21 landmark points present in the hand for detection

**Evaluation of Success**

**Success is evaluated qualitatively through the following metrics:**

1. Accurate face, eye, and hand detection
2. Responsive cursor movement based on face and hand positions
3. Correct detection of eye blinks (left and right)
4. Correct detection of different hand gestures with precision
5. Accurate triggering of mouse clicks based on blink duration

**Computing Resources**

* A computer with a webcam
* CPU capable of running real-time computer vision tasks

**Results:**

**Qualitative Results**

**The project demonstrates several successful features:**

1. **Real-time News Headlines:** For each selected continent, the app fetches and displays relevant news headlines from multiple RSS feeds.
2. **Dual Control Modes:**
   * Eye-tracking Control: Users can navigate the interface using eye movements and blinks.
   * Hand Gesture Control: The app recognizes hand gestures for cursor movement, clicking, and scrolling.
3. **Automatic Mode Switching:** The system automatically switches between eye-tracking and hand gesture control based on hand detection in the camera frame.

**Analysis of Algorithm Limitations and Advantages**

**Advantages**

**1.Real-time Processing:**

The use of OpenCV and Dlib allows for efficient real-time processing of video feeds, enabling the application to detect faces and track eye movements or hand gestures with minimal latency.

**2.Dual Control Modes:**

The integration of both eye-tracking and hand gesture recognition provides users with flexibility in interaction methods. This duality enhances accessibility for users with different needs, making the application more inclusive.

**3.Automatic Mode Switching:**

The system's ability to automatically switch between control modes based on hand detection improves user experience by eliminating the need for manual toggling, allowing for a smoother interaction flow.

**Limitations**

**1.Dependence on Lighting Conditions:**

**2. Gesture Recognition Complexity:**

While hand gesture recognition is effective, it can be sensitive to variations in user behavior (e.g., different hand sizes, speeds of movement), which may lead to inconsistent performance across different users.

**3.Potential Misinterpretation of Inputs**:

Eye blinks can sometimes be misinterpreted due to rapid eye movements or other factors like fatigue, leading to unintended clicks or actions.

**Lessons Learned**

1. Integration of Multiple Technologies: The project demonstrated the power of combining various technologies such as OpenCV, MediaPipe, and feedparser to create a comprehensive user experience. This integration showcased how different libraries and APIs can work together seamlessly.
2. Real-time Processing Challenges: Implementing real-time video processing for both eye tracking and hand gesture recognition highlighted the importance of efficient algorithms and optimization techniques to maintain smooth performance.

**Future Extensions**

* Augmented Reality: Implement AR features to overlay news information directly onto a physical map or the user's environment.
* Improved Gesture Recognition: Enhance the hand gesture recognition system to support a wider range of gestures and improve accuracy in various lighting conditions.

**Bibliography**

1. **Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector.** [**https://arxiv.org/abs/1512.02325**](https://arxiv.org/abs/1512.02325)
2. [**https://mediapipe.readthedocs.io/en/latest/solutions/hands.html**](https://mediapipe.readthedocs.io/en/latest/solutions/hands.html)
3. **Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., & Grundmann, M. (2020). MediaPipe Hands: On-device Real-time Hand Tracking. arXiv.** [**https://arxiv.org/abs/2006.10214**](https://arxiv.org/abs/2006.10214)