

# Test documentation

## I. Introduction:

Stress is a natural response of the body to perceived threats or challenges. It is a normal part of life, but when stress becomes chronic, it can have negative effects on physical and mental health. Stress can manifest in various ways, such as through changes in behavior, physiology and physiology.

Detection of stress in human behavior is an important aspect of understanding and managing stress. By identifying stress early on, individuals and healthcare professionals can take steps to prevent or mitigate its negative effects. Identifying stress through behavior can be challenging, as it can manifest differently in different people. However, by understanding the common signs of stress, it is possible to detect stress in human behavior.

There are several reasons why detecting stress in human behavior is important. Firstly, early detection of stress can help prevent or mitigate the negative effects of stress on physical and mental health. Secondly, it can also help individuals and healthcare professionals to identify the source of stress and take steps to address it. Thirdly, it can also be used to monitor the effectiveness of interventions and treatments for stress. Finally, it can also be used in various fields such as psychology, psychiatry, human resources and security.

In conclusion, stress is a normal part of life, but when it becomes chronic, it can have negative effects on physical and mental health. Detection of stress in human behavior is an important aspect of understanding and managing stress, allowing individuals and healthcare professionals to take steps to prevent or mitigate its negative effects, identify the source of stress and monitor the effectiveness of interventions and treatments for stress.

## II. Related features:

There are several features that can be used for stress detection with computer vision, some of the most common include:

- **Facial expression analysis:** This involves analyzing the facial expressions of an individual to detect signs of stress. Features such as furrowed brows, tight lips, and increased blink rate can indicate stress.
- **Head posture and movement:** The way an individual holds their head, and how they move it can also indicate stress. Features such as tilting the head to one side or holding the head in a rigid position can be used to detect stress.
- **Eye movement and blink rate:** The way an individual moves their eyes and the rate at which they blink can also indicate stress. Features such as increased blink rate, dilation of the pupils, or rapid eye movement can be used to detect stress.

- **Body posture:** The way an individual holds their body can also indicate stress. Features such as slouching, crossing arms, or shifting weight can be used to detect stress.
- **Speech and voice analysis:** The way an individual speaks and the tone of their voice can also indicate stress. Features such as increased pitch, stammering, or a monotone voice can be used to detect stress.
- **Thermal imaging:** Thermal imaging can be used to detect stress by analyzing changes in body temperature. Increased skin temperature, especially on the face, can indicate stress.

These are just a few examples of features that can be used for stress detection with computer vision, I already used some of them in my code but. The specific features used will depend on the application and the available technology. The techniques used for stress detection with computer vision are still in the research and development phase and more sophisticated techniques are being developed.

### III. libraries and framework:

The importance of libraries for stress detection can be summarized as follows:

- They provide pre-built functionality for computer vision and machine learning tasks, such as facial feature detection, facial expression recognition, and image processing, which can be used to detect signs of stress in images and videos.
- They can be used to train custom models for specific stress detection tasks, such as identifying specific facial expressions or body postures that indicate stress.
- They can be used to process live video streams, images, and audio data, which allows for real-time stress detection.
- They can be used to support multiple platforms, such as mobile devices, web browsers, and edge devices, which allows for stress detection in various environments.
- They can be easy to use and customize, which allows developers to quickly build and deploy stress detection solutions.
- They can be used in different fields such as psychology, psychiatry, human resources and security, which can help in understanding and managing stress in human behavior.

In general, libraries for stress detection can be a powerful tool for understanding and managing stress by providing pre-built functionality for computer vision and machine learning tasks, which can be used to detect stress in images, videos, and audio data. They can also be used to train custom models for specific stress detection tasks, which allows for real-time stress detection across various platforms.

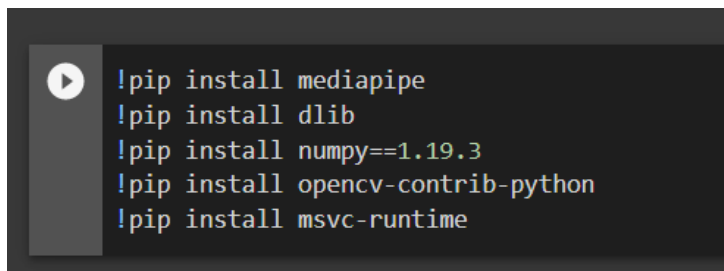
There are several libraries and frameworks that i used in my code and there are others that are useful for stress detection:

- **OpenCV:** OpenCV is an open-source computer vision library that can be used to analyze images and video for stress detection. It includes functionality for facial and body feature detection, as well as image processing and machine learning.
- **dlib:** dlib is a machine learning toolkit that can be used for facial feature detection and recognition. It includes functionality for facial landmark detection, facial expression recognition, and head pose estimation.
- **scikit-learn:** scikit-learn is a machine learning library for Python that can be used for stress detection. It includes functionality for feature extraction, classification, and model selection.
- **pyAudioAnalysis:** pyAudioAnalysis is a Python library for audio analysis that can be used for stress detection. It includes functionality for feature extraction, classification, and segmentation of audio data.
- **AffectNet:** AffectNet is a library for facial expression recognition that can be used for stress detection. It includes pre-trained models and a large dataset of facial images with labels for various expressions.
- **MediaPipe** is an open-source framework that can be used for computer vision and machine learning tasks, including stress detection. It is developed by Google and is designed to be fast and flexible, with a focus on real-time processing.

These are just a few examples of libraries that can be used for stress detection (I have already implemented some of these), depending on the type of data that you have and the specific requirements of your application. It is also important to note that stress detection is still in research phase and more sophisticated techniques are being developed.

## IV. Implementation:

### A. installation of libraries ( [link to cell](#) )



```
!pip install mediapipe
!pip install dlib
!pip install numpy==1.19.3
!pip install opencv-contrib-python
!pip install msvc-runtime
```

Figure1: installation of libraries

## B. Landmark and video reading ( [link to cell](#) )

In this code I'm using the OpenCV, MediaPipe, and os libraries to detect facial landmarks in a video.

- It starts by importing the necessary libraries
- It then loads the video from the specified path using the cv2.VideoCapture function.
- It then instantiates the mp.solutions.face\_mesh.FaceMesh() function to detect the facial landmarks, and starts reading the video frame by frame. The while loop continues until there are no more frames to be read, and inside the loop, it uses the mp\_face\_mesh.process() function to process the video frame and detect the facial landmarks.
- It then iterates over the facial landmarks points and marks them on the video frame using cv2.circle() function.
- It then shows the modified video frame using the cv2.imshow() function, and waits for 1 millisecond before reading the next frame, this is done to slow down the video.
- Finally, it releases the video capture, and closes all the windows.

Overall, this code uses a combination of OpenCV, MediaPipe, and os libraries to process a video file and detect the facial landmarks on the video frame.

## C. Eye blink detection ( [link to cell](#) )

In this code I'm using OpenCV and MediaPipe libraries to display text on an image with different background styles.

- It starts by importing the necessary libraries
- It then defines some global variables such as color values, a points\_list and 3 functions:
- drawColor(img, colors): this function is used to draw rectangles on the image with the specific color provided.
- colorBackgroundText(img, text, font, fontScale, textPos, textThickness=1, textColor=(0,255,0), bgColor=(0,0,0), pad\_x=3, pad\_y=3) : this function is used to display text on image with a colored background.
- Calculating frame per seconds FPS with this formula:  
end\_time = time.time()-start\_time  
fps = frame\_counter/end\_time
- And finally it calculate and print the blink number on each frame

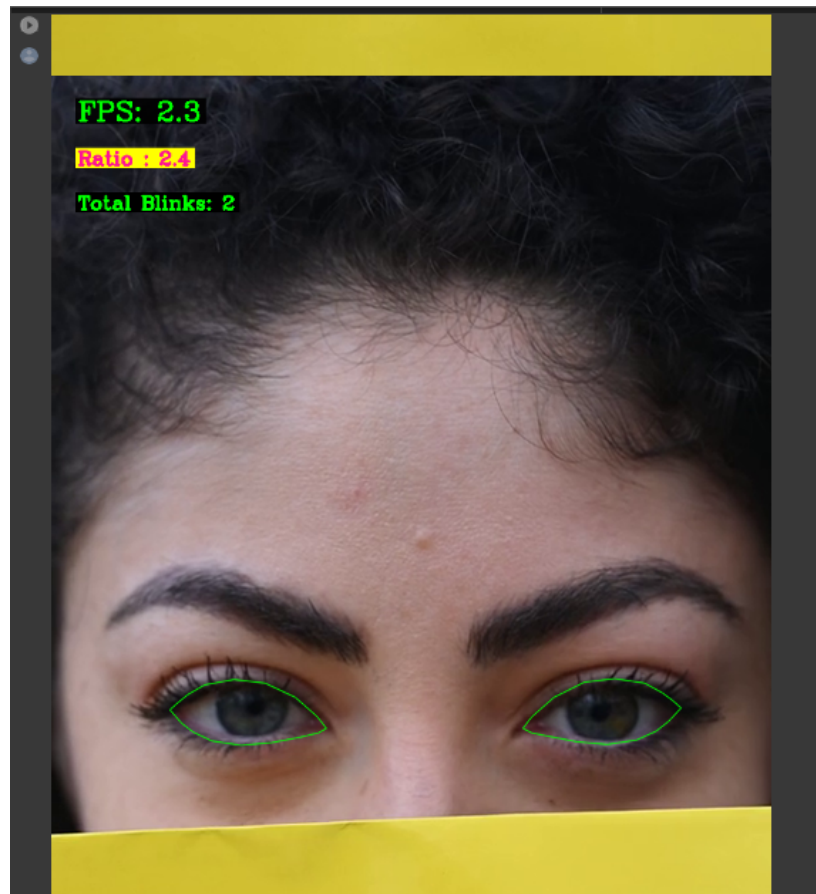


Figure2: eye blink demonstration

#### D. Head pose detection ( [Link to cell](#) )

In this code I'm using OpenCV and MediaPipe libraries to detect the head pose of a person in a video.

- It starts by importing the necessary libraries
- Then it instantiates the `mp.solutions.face_mesh.FaceMesh()` function to detect the facial landmarks, and starts reading the video frame by frame. The while loop continues until there are no more frames to be read, and inside the loop, it uses the `face_mesh.process()` function to process the video frame and detect the facial landmarks.
- It then uses the 2D and 3D coordinates of the facial landmarks and the camera matrix to calculate the angles of the head rotation using `cv2.solvePnP()` and `cv2.Rodrigues()` functions.
- It then uses these angles to check if the user is looking left, right, up or down, and display the corresponding text on the video frame using the `cv2.putText`
- Calculating frame per seconds FPS with this formula:
  - `end = time.time()`
  - `totalTime = end - start`
  - `fps = 1 / totalTime`

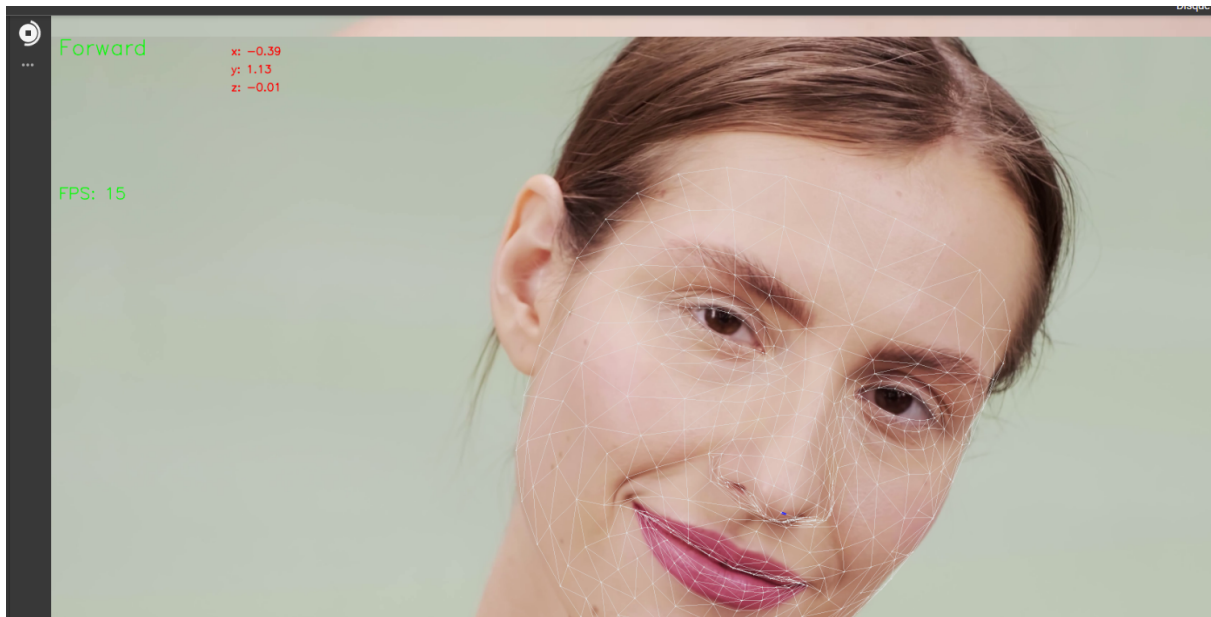


Figure3: head pose demonstration

### E. Hand pose detection ( [Link to cell](#) )

In this code I'm using the Mediapipe library to detect and track the pose of a person in a video. The video is read using the OpenCV library and the frames are passed to the Mediapipe Pose module for processing. The Pose module uses machine learning algorithms to detect and track the keypoints of the person's pose in the image. The results of the pose detection are then used to draw the pose landmarks on the image using the Mediapipe Drawing module. The resulting image with the pose landmarks is then displayed using OpenCV's cv2\_imshow function. The code also includes an optional step to flip the image horizontally for a "selfie-view" display. The loop continues until the escape key is pressed or the video ends.

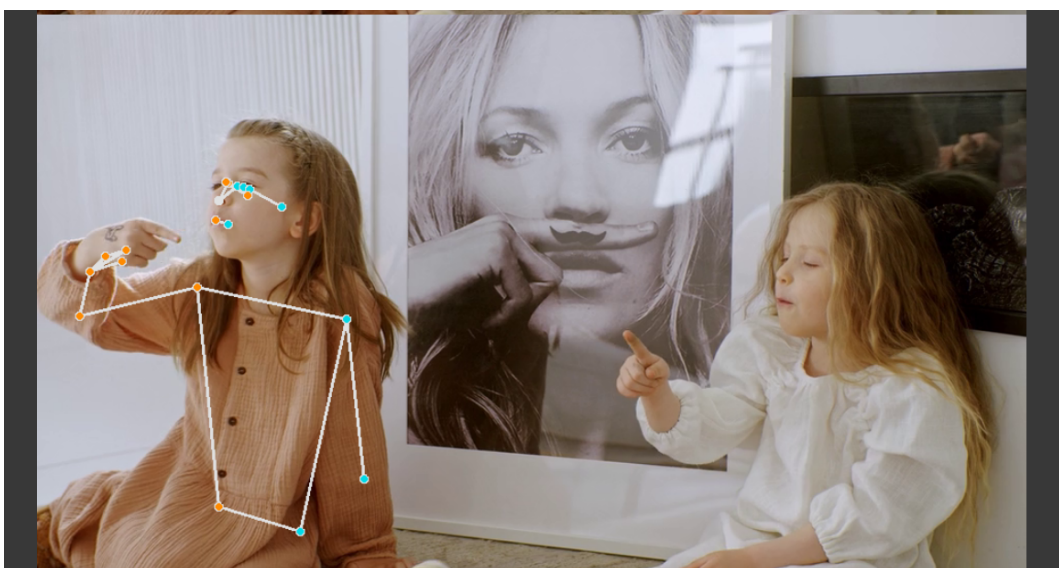


Figure4: hand pose demonstration

## F. Eyebrow with stress detection ( [Link to cell](#) )

For detecting stress levels in a person by analyzing their facial expressions and eye movements. I used a combination of computer vision techniques and machine learning models to detect stress. I used the dlib library to detect the face and facial landmarks in the video feed, then used the scipy library to calculate the distance between the two eyebrows. The emotion\_finder function uses a pre-trained deep learning model to classify the emotion of the person in the video feed. Then the normalize\_values function compares the eyebrow distance with a threshold value to classify the level of stress. The result of the stress level and the corresponding emotion is then displayed on the video feed. The code uses OpenCV library for capturing video, imutils library for resizing the video frames and matplotlib for plotting the results.

## V. FUTURE WORK ( Perspectives )

The future work for this project can include several areas of improvement and expansion. One potential direction is to increase the accuracy of the stress detection by incorporating more advanced machine learning and deep learning techniques and using larger datasets with tensorflow. Additionally, it could be useful to explore different physiological signals, such as heart rate or electrodermal activity, to improve the accuracy of stress detection. Another area of future work could be to develop an interactive system that provides real-time feedback to the user ( using reinforcement learning), such as through a wearable device or using an Nvidia Jetson nano card with raspberry pi 0. This could allow for more personalized stress management strategies and provide the user with a better understanding of their stress levels throughout the day. Additionally, it could be interesting to explore ways to integrate this stress detection system into other applications, such as in the workplace or in educational settings. Overall, there are many opportunities for future work to improve and expand upon this project and make it more widely applicable in different settings.

Last but not least I would like to express my deepest gratitude to **Captiosus Axons** for giving me the opportunity to improve my skills and knowledge through the test that was conducted last week. This experience has been truly valuable and I have learned so much from it.