**MediaPipe Face Mesh Capabilities**

**Yaakoub Chaker**

**Summary:**

**MediaPipe Face Mesh:**

MediaPipe Face Mesh is a highly efficient model for face detection, offering 468 landmarks to identify facial features. However, it lacks a specific landmark for the neck, leading to a couple of challenges:

**Problem 1: Lack of Neck Landmark**

The primary issue is the absence of a dedicated neck landmark. While it's theoretically possible to add new landmarks to the model, this process is complex and involves the creation of virtual points. It also requires retraining the MediaPipe Face Mesh model on a dataset that includes the new landmark. At present, finding a clear and comprehensive resource for adding new landmarks in MediaPipe is challenging.

**Problem 2 : Inability to Detect Modified Mallampati Score**

Within the mouth, MediaPipe Face Mesh focuses on detecting facial landmarks rather than intricate mouth shapes. As a result, it lacks the capability to recognize or assess the "Modified Mallampati Score." Since MediaPipe primarily detects facial features and not the detailed mouth structure, it cannot be used to accurately determine the Modified Mallampati Score, limiting its suitability for this specific medical assessment

“we can add a photo here of the mouth foe explain” = = > not require

**Problem 3: Calculating Real Distance Between Landmarks**

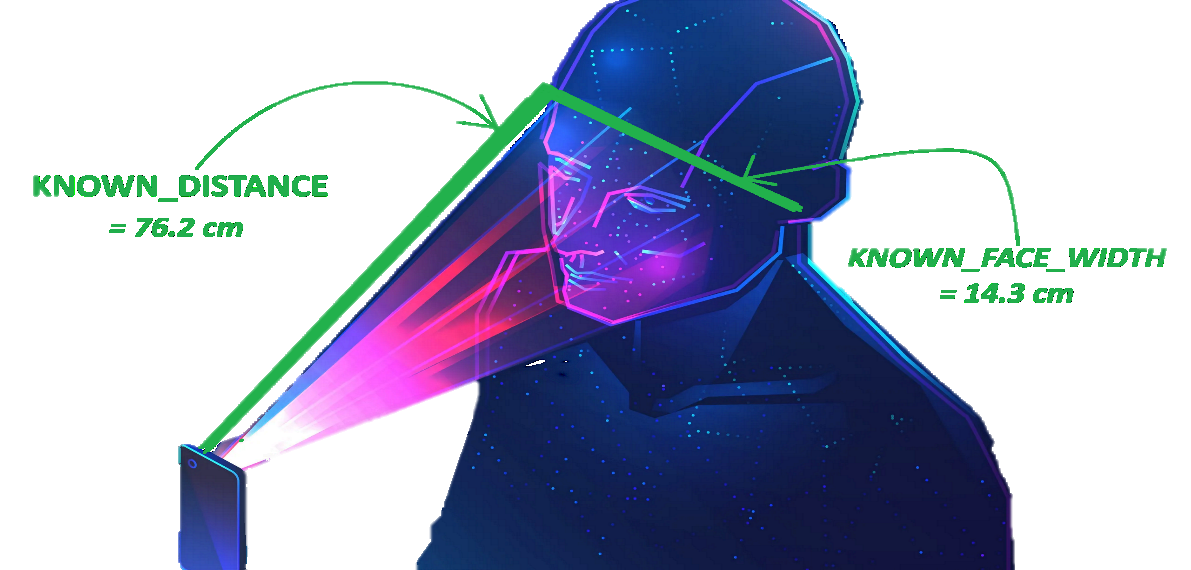
Accurately measuring distances between two landmarks using MediaPipe Face Mesh presents a challenge. MediaPipe lacks a direct, built-in method for this task. Despite extensive research, a default solution from MediaPipe remains unidentified.

**How to solve that?**

**1 - estimating the distance between camera /face**

Our workaround involves estimating distances between the camera and an individual's face using a real-world image capture as a reference with known parameters:

* KNOWN\_DISTANCE (distance between camera and face)
* KNOWN\_FACE\_WIDTH



Leveraging this reference, we estimate distances for other images with faces.

Our tests involving real-time estimations and static images displayed promising results. Validation was performed using a PC camera, with upcoming tests focused on mobile cameras. Consistent outcomes are currently achievable only with consistent image dimensions. Variations arise when image dimensions change, emphasizing the need for a mobile application with a tailored camera setup featuring fixed width and height parameters. Moreover, image usage will be limited solely to those captured within our mobile application.

3 – 4 real image estimated the distance ex :

For clarification see here:

Code … git hub

**2 How can we get more out of estimating the distance between the camera and the face :**

**2-1 Using a fixed focal length : 🡺 Correcting the result based on the error percentage :**

Currently, a practical approach involves relying on the distance measurement between the camera and the face. establishing a consistent relationship based on known measurements.

For instance, suppose we assign 1 meter as the reference distance, equating to a fixed distance of 10 cm between specific landmarks, like the distance between the right and left jaw in a reference image with standardized dimensions.

Using this reference, our program can flexibly estimate distances between the camera and the face.

For instance, if the program it estimates a distance of 15 cm between the right and left jaw based in a **fixed** focal length , considering the 1-meter reference, the actual distance would be 10 cm.

This discrepancy of 5 cm is the error in the calculation based on fixed focal length.

🡺 Conducting several experiments at different distances allows us to calculate the percentage error. Understanding this error percentage enables us to make corrective adjustments, leveraging our knowledge of the camera-to-face distance and have a formula (function , algorithm ) to have the right distance between 2 landmark .

In essence, the key lies in leveraging constants related to the distance between the camera and the face for accurate distance estimation between two points."

**- Test example (fixed focal length):**

<https://github.com/qduriani/Morpheus/tree/ChakerTest/ChakerTest/MediaPipe2>

For example, in this code, if you try to run python media.py in the terminal, the code will execute an output , this output is the distance between two points (2 landmarks in the face ).

The distance is wrong, and we will correct the result . "

+ 2 , 3 photo

**2-2 Calculating the focal length ( not tested yet / under consideration ) :**

Deriving the focal length for the camera involves a series of experiments and a deep comprehension of camera coverage. We're currently devising a plan to implement these procedures in pursuit of achieving accurate results. (We have not tried that approach yet)

**How can we estimate the distance between two landmarks in calculating the focal length??**

1. Formulate Equation: Apply the formula:

**Focal Length (in pixels/cm) = (Pixel Width \* Known Distance) / Known Width**

where:

**Focal Length :** is the focal length of the camera in pixels per centimeter

**Pixel Width :**  is the width of the captured image in pixels

**Known Distance** : is the known distance between the camera and the face in centimeters

**Known Width** : is the width of the face in centimeters

1. Normalized Distance , Conversion Factor

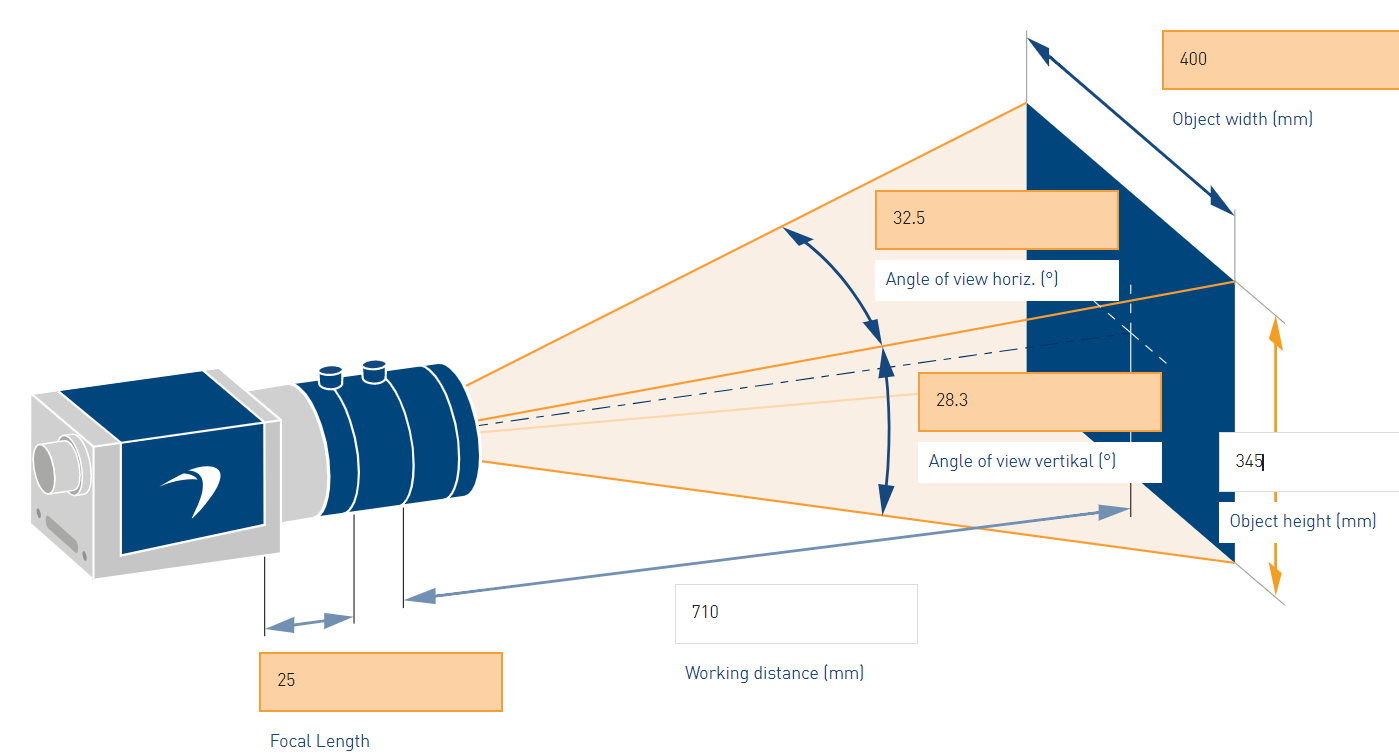
Estimated Distance (in cm) = (Distance in Pixels between 2 landmarks ) / Focal Length

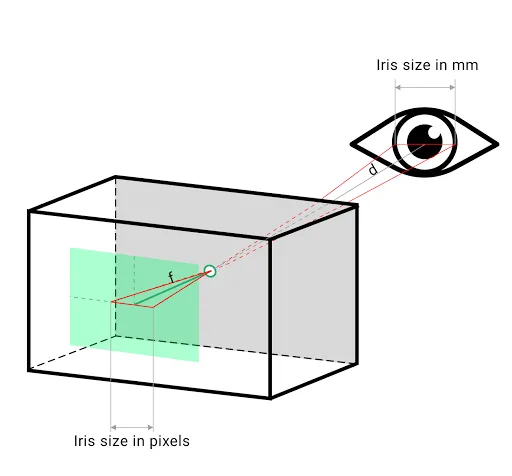
1. Considerations:

Consistency in image dimensions (width to the height ) remains crucial for accurate distance estimations in centimeters.

This revised approach enables the estimation of distances between landmarks detected by MediaPipe in centimeters by utilizing the focal length and converting pixel distances to real-world measurements.

We don't have a lot of references on that approach , and we have to learn more about this approach, and we must try it in the next chapter . It is a method that we arrived at by searching for bard and chatsGBT and researching how to calculate focal length ..





<https://medium.com/@susanne.thierfelder/create-your-own-depth-measuring-tool-with-mediapipe-facemesh-in-javascript-ae90abae2362>

**In conclusion:**

MediaPipe Face Mesh offers 468 fixed topology landmarks, without direct support for adding new landmarks due to the model's limitations. To define a custom point outside the predefined landmarks, manual frame-based location identification is necessary, involving intricate image processing techniques such as contour identification or feature recognition.

While measuring distances between existing facial landmarks using MediaPipe Face Mesh is feasible, our exploration will involve two directions: calculating the focal length and using a fixed focal length. However, challenges arise when measuring distances within the mouth or neck region. These challenges may necessitate exploring alternative methods or constructing our own model , potentially through the creation of a dedicated dataset tailored to address these specific regions.

**Useful Links :**   
<https://developers.google.com/mediapipe/solutions/vision/face_landmarker/>

<https://developers.google.com/mediapipe/api/solutions/python/mp>

<https://www.techrxiv.org/articles/preprint/Calculating_screen_to_face_distance/12951320>

**Things that can be benefited from in our project (they must be discovered) :**

* OpenPose, SMPL

- **OpenPose** is a versatile and open-source library that can detect and track 138 keypoints on the human body, including neck and face landmarks

**-SMPL (Simplified Human Model):**

SMPL is a 3D body model that simplifies the representation of the human body, making it useful for various applications. It can be utilized for estimating body shape and pose, which can be valuable for understanding human body proportions and movements.

- Determine if SMPL's capabilities align with the specific objectives of our project, such as distance estimation or body pose analysis.

- Investigate the feasibility of integrating SMPL into our project and whether it requires additional data or training.

<https://www.youtube.com/watch?v=m8i00zG6mZI&list=PLon2R-xWRkjcbEggkfl2hjtROA-uKksJ0&index=6>

**yaakoub chaker**

Nothimg just for save :  
To find the real-world distance between two facial landmarks detected by the MediaPipe FaceMesh, you need additional information about the physical properties of the camera and the image. You can estimate the real-world distance by calibrating the camera or knowing the physical size of an object in the image.

The code snippet you've used calculates the distance between two landmarks in normalized units, which isn't directly the real-world distance but a relative measure within the image. To convert this to real-world units (e.g., centimeters), you'd need more information about the scale of the image in the real world.

Here's a general approach:

Camera Calibration: If you know the camera's intrinsic properties (focal length, sensor size), you can perform camera calibration to determine the relationship between pixels and real-world units. This process typically involves capturing images of a known calibration object with defined dimensions.

Known Object Size: If you have an object of a known size in the image, you can use its size in pixels to establish a conversion factor between pixels and real-world units.

Without such calibration or known object sizes within the image, it's challenging to directly convert the normalized distance between landmarks to a real-world distance. The MediaPipe library doesn't provide direct real-world measurements; it gives normalized coordinates within the image.

If you have access to the camera's intrinsic properties or objects of known sizes within the image, I can assist in incorporating that information into the calculations for real-world distance estimation.