# Application of Smoothed Facial Motion Capture System for Live2D VTuber Model

* Subiyanto[1] and Izza Azzurri[1]

[1]Universitas Negeri Semarang Jurusan Teknik Elektro

August 22, 2023

**Abstract**

The rapid growth of the live streaming industry brought about the VTuber trend, where content creators use character avatars to stream. One of the most accessible way to move a character in real time is by using a vision-based facial motion capture technique. However, previous works still suffer from jittering issues, which hinder the quality of the character's movements. This work aims to develop a smoothed facial motion capture system that works with a live2D VTuber model. The system combines Mediapipe Face Mesh and OpenCV solutions to capture facial landmarks, which are then used to calculate head pose estimation using the Perspective-n-Point (PnP) function. In addition, the system uses EAR and MAR functions to detect facial features. The motion values obtained from this process are then filtered using a Kalman filter. Finally, the filtered motion data is sent to the Unity engine, which drives the Live2D VTuber model by adjusting the character's motion parameters. The developed system successfully captures and drives the Live2D VTuber model with smoother motion, overcoming the jitter problem prevalent in previous facial motion capture approaches. The system's improved motion capture quality makes it a more viable option for a wide range of potential uses.

**Application of Smoothed Facial Motion Capture System for Live2D VTuber Model**

**Izza Azzurri[1] and Subiyanto[2*]** [1,2]*Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang, Indonesia*

**Data Availability Statement:**

The data sets used in this study are available upon reasonable request from the corresponding author.

**Funding Statement:**

This research was not funded by any party.

**Conflict of Interest Disclosure:**

The authors declare no conflicts of interest related to this research.

**Ethics Approval Statement:**

Ethical approval for this study was obtained from UNNES Electrical Engineering Students Research Group (UEESRG)

**Patient Consent Statement:**

Informed consent was obtained from all human participants included in the study.

**Permission to Reproduce Material from Other Sources:**

Permission to reproduce materials from other sources has been appropriately obtained and credited in the manuscript.

**Clinical Trial Registration:**

This study is not a clinical trial and therefore not registered in a clinical trials database.

**Author** : Izza Azzurri Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang, Indonesia.

Email: izzaazzurri@students.unnes.ac.id

***Corresponding author** : Subiyanto, Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang, Indonesia.

Email: subiyanto@mail.unnes.ac.id Tel.: +62248508104; Fax: +886-5-6314486

**Application of Smoothed Facial Motion Capture System for Live2D VTuber ModelIzza Azzurri[1] and Subiyanto[2*]** *[1,2]Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang, Indonesia *corresponding author. Email: subiyanto@mail.unnes.ac.id*

**Abstract**The rapid growth of the live streaming industry brought about the VTuber trend, where content creators use character avatars to stream. One of the most accessible way to move a character in real time is by using a vision-based facial motion capture technique. However, previous works still suffer from jittering issues, which hinder the quality of the character's movements. This work aims to develop a smoothed facial motion capture system that works with a live2D VTuber model. The system combines Mediapipe Face Mesh and OpenCV solutions to capture facial landmarks, which are then used to calculate head pose estimation using the Perspective-n-Point (PnP) function. In addition, the system uses EAR and MAR functions to detect facial features. The motion values obtained from this process are then filtered using a Kalman filter. Finally, the filtered motion data is sent to the Unity engine, which drives the Live2D VTuber model by adjusting the character's motion parameters. The developed system successfully captures and drives the Live2D VTuber model with smoother motion, overcoming the jitter problem prevalent in previous facial motion capture approaches. The system's improved motion capture quality makes it a more viable option for a wide range of potential uses.**Keywords:** VTuber, Motion Capture, Mediapipe, Kalman Filter

# Introduction

Live streaming, particularly on platforms like Twitch, has become increasingly popular, with virtual content creators or "VTubers" gaining significant traction even compared to their real-life counterpart. VTubers use character avatars to engage with their audiences, which has several advantages, including increased privacy by the anonymity provided and the ability to create a unique personal brand that in turn can generate more income for the content creator. VTubers can also be utilized outside of the entertainment industry, such as in education or as virtual moderators. VTubers usually stream with 2D anime-styled character created with tools such as Live2D Cubism. Live2D Cubism is a software developed by Cybernoid that allows 2D drawings to move like a 3D model by splitting the drawing parts into layers and deforming them accordingly. There are various motion capture methods that allow for real-time animation of virtual characters that ideal for live streaming applications. One of these method is a marker-less computer-vision based motion capture system that is more cheaper and accessible since it doesn't require additional sensor or hardware like other methods. Research by Wang et al. implemented machine learning as a way to perform facial expression capture using the HRNetV2-W18 model to obtain 98 high-resolution facial landmarks that can drive virtual character expressions. However, this method only captures facial expressions and does not track head motion and facial movements in real time. For head motion capture, Ariz et al.[12] and Ye et al. performed a head pose estimation using the PnP method. Meanwhile, The study conducted by Hammadi et al. evaluated several state-of-the-art algorithms for head pose estimation in clinical scenarios. The study found that the 3DDFA_V2 model is superior to OpenFace2.0 and MediaPipe in terms of estimating head

pose, as it resulted in a mean error lower or equal to 5.6*, depending on the plane of motion. However, the study also acknowledges that the Mediapipe face mesh model is a more robust solution for overall face detection, as it considers a geometric approach to estimate a total of 468 facial landmarks in three dimensions compared to OpenFace2.0 and 3DDFA_V2 which use 68 landmarks to estimate the pose. This approach makes the Mediapipe face mesh model more reliable in the presence of outliers when computing the homographic parameters. It is worth noting that the 3DDFA_V2 model does not feature facial expression and eye gaze analysis, which may be important for other applications. Mediapipe itself is a pipeline that can be used in various applications including face landmark detection and segmentation, where Mediapipe can register a detailed 3D mesh template to a human face on an image. To move a character model face, Wang et al. transmit expression parameters based on the movement of the recognized landmarks to Unity to drive the virtual character face for generation of real-time expression animation. On the other hand, Takács calculates important movements such as head rotation, mouth opening distances, blink detection, and line of sight based on the mediapipe face mesh landmark. These calculations are sent to Unity to move the character model based on their corresponding parameters. However, Takács also stated that landmarks tend to jitter from frame to frame when using mediapipe face mesh. This issue can be solved by implementing a filter such as the Kalman filter to estimate the motion. There's also another way to detect blink and measure how much the mouth is opened more concisely by using Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) formula respectively. These formulas are commonly used for drowsiness detection and still based on 68 landmarks Dlib face model, which need further adjustment with the 468 landmarks of mediapipe face mesh model. Ultimately, there is a lack of a comprehensive system that integrates all these essential components to reduce the jittering issue and overall improve the character's motion quality in real-time facial motion capture for Live2D VTuber models. This study aims to utilize mediapipe face mesh with both the EAR and MAR formulas, and the Kalman filter to develop a smoother real-time facial motion capture system for a Live2D VTuber model. Therefore, this paper contribution is to develop a system that combines the Mediapipe Face Mesh and OpenCV solutions to capture facial landmarks and perform head pose estimation using the Perspective-n-Point (PnP) function. Building on this, the system introduces EAR and MAR functions based on the Mediapipe facemesh landmarks to track eye and mouth movements. Another important addition is the implementation of a Kalman filter to reduce jitter, and providing smoother motion data. This seamless integration can hopefully enhance the virtual character's realism and dynamic interactions.

# Method

FIGURE 1: Motion Capture Pipeline Based on Takács's previous work, the motion capture system consists of three main parts, namely face tracking, TCP connection, and character animator. Based on this foundation, the system has been enhanced by introducing a Kalman filter to filter the movement data to make it smoother. In addition, the previous 3D character animator has been replaced with the Live2D animator to use the Live2D character model. Furthermore, the TCP connection has been simplified, choosing a direct approach instead of a WebSocket server due to the absence of web technologies in the setup. These adjustments, shown in blue in Figure 1, have resulted in an expanded and more linear flow chart, providing a clearer explanation of how the motion capture system works, as depicted in Figure 2. FIGURE 2: Flowchart of the developed motion capture system The system is designed to perform real-time facial motion capture by tracking the face and moving the character model synchronously. The system design includes initialization of both tracking and character controller modules, video capture, face tracking using mediapie face mesh, and calculating motion values for head pose and facial expression, which are then stabilized using Kalman filter, and finally sent to the character controller module in Unity to change the character parameters so that the character moves according to the user's movement in real-time.

## Facial Motion Capture

The first step for real-time face tracking or facial motion capture is to recognize and capture human faces from the video input. To achieve this, the Python OpenCV library is used to read the image frame by frame from the selected video input. The frames are processed sequentially, starting with flipping the image horizontally and converting the color space from BGR to RGB to match the configuration required by the Mediapipe face mesh. After pre-processing, the human face in the image is processed using the Mediapipe Face Mesh solution. The Mediapipe Face Mesh solution estimates the geometry of the surface by identifying 468 landmarks (478 landmarks if iris landmarks are included). Each of these landmarks corresponds to a vector point in 3D space and is assigned a specific number, which can be used for later calculations. These landmarks can also be visualized as separate layers over the image by drawing them using face mesh tessellation, thus displaying the geometry of the face mesh as depicted in Figure 3. This visualization helps in understanding the structure of the face and can facilitate further analysis and processing of the captured human face data. FIGURE 3: Drawn face mesh tessellation After obtaining the landmarks and their positions, the next step involves pose estimation to determine the movement. Pose estimation in computer vision refers to determining the relative orientation of an object with respect to the camera. This problem is commonly known as the Perspective-n-Point (PnP) Problem, which estimates the head pose from a set of 2D-3D face point correspondences . The head pose itself consists of six degrees of freedom (DOF), which include rotation (roll, pitch, and yaw), and the 3D translation of the head as shown in Figure 4. FIGURE 4: Head pose orientation in 3D space To solve the PnP problem, Guobing created an approach that utilizes the solvePnP function of the OpenCV library, which is also used in this system. This function, along with related functions, estimates the head pose based on a set of landmark points, the corresponding image projection, and the camera's intrinsic matrix and distortion coefficients. But, instead of using the previous 68 landmark models, this system now uses 468 3D model points (excluding the iris) obtained from the canonical face model file provided by Mediapipe. The solvePnP function returns rotation and translation vectors that approximate the head pose. These vectors are essential in converting 3D points expressed in the world frame into the camera frame. As a result, a 3D annotation box representing the 3D orientation of the face can be drawn, as illustrated in Figure 5, which enables a visual representation of the head pose based on the landmark data. FIGURE 5: Drawn Annotation Box to show face 3D orientation In this system, facial motions such as blinking, mouth opening, smile/mouth form, and iris position are also tracked in real-time. The necessary landmark points for these features are also obtained from the mediapipe face mesh. Instead of using all the landmarks, only a few landmarks that correspond to the facial features are utilized based on the EAR and MAR formulas . FIGURE 6: Eye Landmarks for EAR The Eye Aspect Ratio (EAR) is used to calculate the ratio of the vertical and horizontal length of the eye. The center position of the landmark points is used to determine the vertical length, while the distance between the eye corners is used to calculate the horizontal length. Based on the mediapipe face mesh landmark point in Figure 6, the values of left and right EAR are calculated by the following equation:

$$\text{EAR}=\|p_{160}-p_{144}\|+\|p_{158}-p_{153}\|\frac{}{2\|p_{33}-p_{133}\|} \quad (3.1)$$

$$\text{EAR}=\|p_{387}-p_{373}\|+\|p_{385}-p_{380}\|\frac{}{2\|p_{263}-p_{362}\|} \quad (3.2)$$

FIGURE 7: Mouth Landmarks for MAR Similar to the Eye Aspect Ratio (EAR), the Mouth Aspect Ratio (MAR) is the ratio of the vertical length between a pair of lips and the horizontal length at the corner of the lips. Based on the mediapipe face mesh landmark point in Figure 7, the values of MAR are calculated by the following equation:

$$\text{MAR}=\|p_{81}-p_{178}\|+\|p_{13}-p_{14}\|+\|p_{311}-p_{402}\|\frac{}{2\|p_{78}-p_{308}\|} \quad (3.3)$$

For iris position, the refined face mesh has 478 landmarks including the iris landmark, 468 for the left iris and 473 for the right iris. We can then measure the distance of these landmarks to the eye corner to find

the x-rate and the highest and lowest eyelid landmark points for the y-rate. Besides that, mouth corner distance is used to determine whether the user is smiling or not. After obtaining all the movement values, a Kalman filter implementation in OpenCV based on the work of Fischer is used in this system to stabilize all the movement values before sending them to the character animator in Unity.

### Animating Live2d Character model in real time

In this system, the face tracking module is written in Python and the character controller inside Unity is written in C#, so an intermediary is needed to connect these two separate modules. To achieve this, a TCP connection is used, where the Unity character animator module acts as the server and the face tracking module acts as the client. This connection allows the character controller to receive real-time motion values from the face tracking module and use them to control the corresponding movement parameters of the character in real-time. FIGURE 8: Niziiro Mao character model The character model used in this system is named Niziiro Mao (Figure 8), which is a sample model from the official Live2D website. This sample model has already been rigged and is ready to be animated. To use this Live2D model in Unity, the Live2D Cubism SDK is also required. When placed inside Unity, this character model will have parameters that can be used to move or animate the character. This model specifically has 113 parameters, but only 11 parameters will be used that correspond to our pre-captured movement values, such as the model head XYZ angle with the head pose yaw, pitch, and roll angle values. These angle values were also used to change the body movement parameters accordingly, so the character looked more lively. Other than that, EAR and MAR values correspond to both eye and mouth opening movement parameters respectively, while iris x-rate and y-rate correspond to eyeball parameter and mouth distance to mouth corner parameter. with some adjustment to scale the values in Unity the character model are now animated in real time according to user movement.

## Result

FIGURE 9: Initial system test This system was tested on a Windows 10 machine with an Intel(R) Core(TM) i5-9300H 2.40 GHz processor, 8.00 GB of RAM, and NVIDIA GeForce GTX 1650. Tests were also conducted using two different camera options, an integrated webcam (USB 2.0 HD UVC WebCam, 0.92 MP) and an Android camera (Infinix Note 11 Pro, 64 MP, f/1.7 Quad Bayer main camera) connected via USB. Initial system tests show that the system works well, with all features functioning as expected as shown in Figure 9, where users can control the character models using their head and face movements that are indicated by the face mesh and two 3D annotation boxes that each represent raw (white) and filtered (purple) movement. The rendering FPS (frames per second) averaged around 138, indicating a smooth and responsive experience. Based on direct observation, there were noticeable differences in jitter between the filtered and unfiltered movements captured by both the built-in webcam and the Android camera. Though, the jittering difference was more pronounced with the built-in webcam compared to the Android camera, where the differences were less noticeable. FIGURE 10: Testing of varying user attributes The system also underwent tests to capture users with various attributes such as gender, age, and glasses, as seen in Figure 10. The results showed that there was no noticeable difference in tracking performance, as all participants were able to control the character's head and expression as desired. FIGURE 11: Predefined animation test To further test the system's performance, a predefined animation is generated using Blender and tested as seen in Figure 11. This animation is then played back through the OBS virtual camera, simulating real-time video capture. The purpose of this test is to evaluate the system's accuracy and smoothness by comparing both raw and filtered output to a known and true values provided by the predefined animation. FIGURE 12: Predefined animation test movement comparison graphs.**a.** Roll movement. **b.** Yaw movement **c.** Pitch movement The test results given in Figure 12 depict graphs representing roll, yaw, and pitch motion, respectively. In the roll movement graph (Figure 11.a), a positive angle indicates that the head is tilting or rolling to the left, while a negative angle indicates that the head is tilting to the right (from the character's perspective). Similarly,

in the yaw motion graph (Figure 11.b), positive values indicate that the head is facing left, while negative values indicate that the head is facing right. As for the pitch movement graph (Figure 11.c), positive values indicate the head is facing up, and negative values indicate the head is facing down. The graph also shows that the filtered data exhibits fewer pronounced spikes, confirming the previous observation that the filtered movement appears smoother. In addition, measurements were also taken to evaluate the performance of the system. These measurements include standard deviation, standard error, and mean absolute error, which serve to show the accuracy of the system, as can be seen in Table 1 It is worth noting that certain motions affect the angle of other motions. Moreover, the filtered pitch movement angle consistently starts at -90 and requires several frames to adjust to the correct movement angle. Therefore, the first ten frames were not included in the measurements. Nevertheless, despite the smoother motion, the measurements showed little to no improvement in accuracy after filtering the data. Next, the system is tested using different camera options to verify the previous observations. The test is performed by connecting an Android device via USB and recording videos simultaneously using both the Android camera and the integrated camera at the same distance, using OBS split camera to ensure consistent movement values. The recorded video is then played back in the same manner as the predefined animation test. This test resulting in Figure 11, that shows each camera capture, character movements and the corresponding yaw movement angle. FIGURE 13: Integrated (top) and Android (bottom) Camera test results within the same distance. From the test shown in Figure 13, a clear similarity in performance can be observed between the two camera options. In addition, the standard error values provide further insight, where the raw integrated camera has a value of 2.800 while the filtered integrated camera has a slightly lower value of 2.660. Similarly, the raw Android camera has a value of 2.859, and the filtered Android camera has a value of 2.735. These values show a close similarity in the performance of the two camera options. However, it is important to consider that the Android camera has a wider angle of view compared to the integrated camera. As a result, faces may appear smaller in the input video, affecting the distance between landmarks. Despite having a higher resolution, this difference causes the Android camera to show the same or lower accuracy than the integrated camera at the same distance, which also affects the smoothness of the character's movements. On the other hand, when the test is conducted with the Android camera placed closer to the subject, approximately matching the angle of view of the integrated camera as seen in Figure 14, the character movement appears to be smoother and more consistent. The resulting graph displayed smoother movements and less noticeable spikes compared to the integrated camera. These results also correspond to the initial direct observation and show that the resolution and distance of the camera can also affect the tracking smoothness. FIGURE 14: Integrated (top) and Android (bottom) Camera Test Results with a similar angle of view.

# Conclusion

The developed system has shown significant improvement from previous works, providing users with effective control over Live2D character models through head and face movements with a smooth and responsive experience. The implementation of the Kalman filter has successfully improved the smoothness of the motion tracking, although it has not improved the accuracy. While there is no noticeable difference in tracking performance based on user attributes, it is noticeable that the camera option has an impact on the smoothness of the motion. The Android camera, despite having a higher resolution, produced lower accuracy compared to the integrated webcam at the same distance due to the wider viewing angle. However, the Android camera showed smoother and more consistent motion when positioned closer to the subject. With the system's improved motion capture quality, it became a more viable option for a wide range of potential uses, even outside of the entertainment industry. The findings show that this system, utilizing the Android camera, offers a promising setup for reducing the cost of motion capture while still providing smooth tracking for users. Moving forward, future work could explore alternative face tracking algorithms or filters to further refine the system. With further improvements and wider implementation, this system has the potential to enhance a wider range of applications that require motion tracking capabilities.

## Acknowledgements

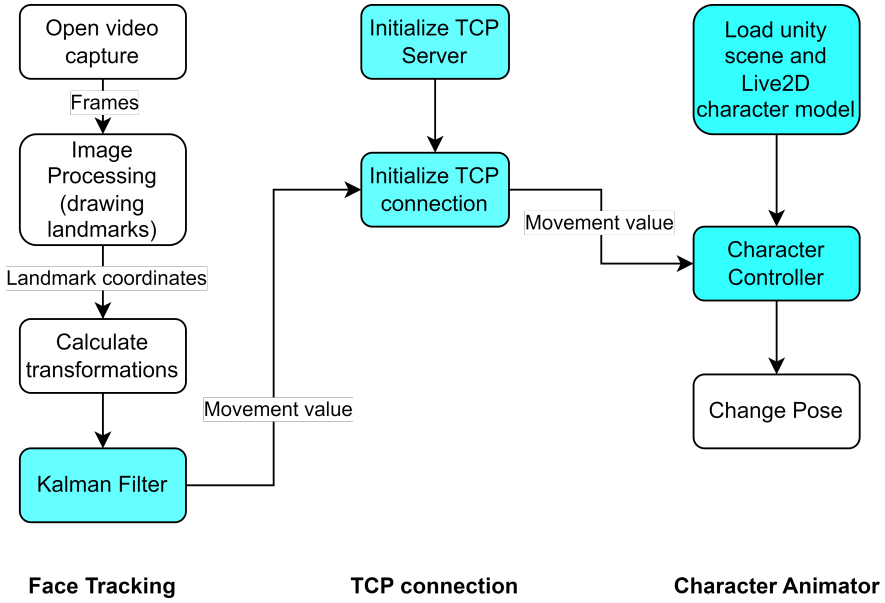
This work has been supported in part by UNNES Electrical Engineering Students Research Group (UEESRG), Universitas Negeri Semarang

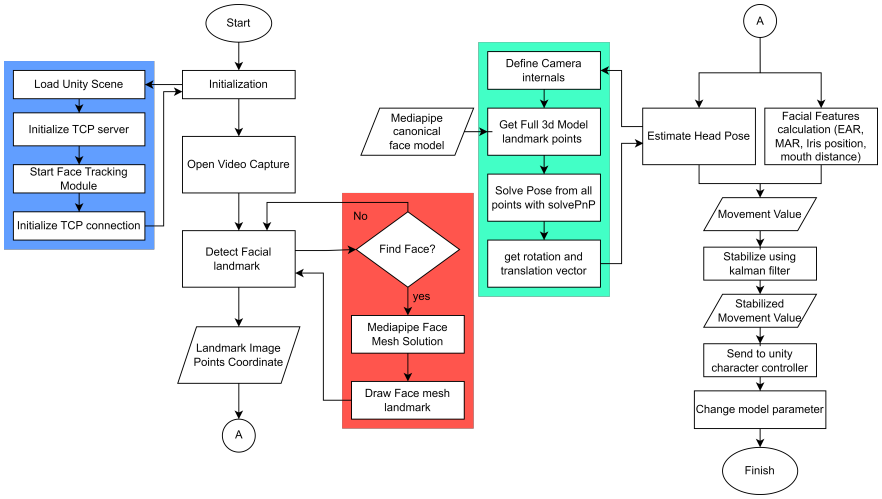
## Reference

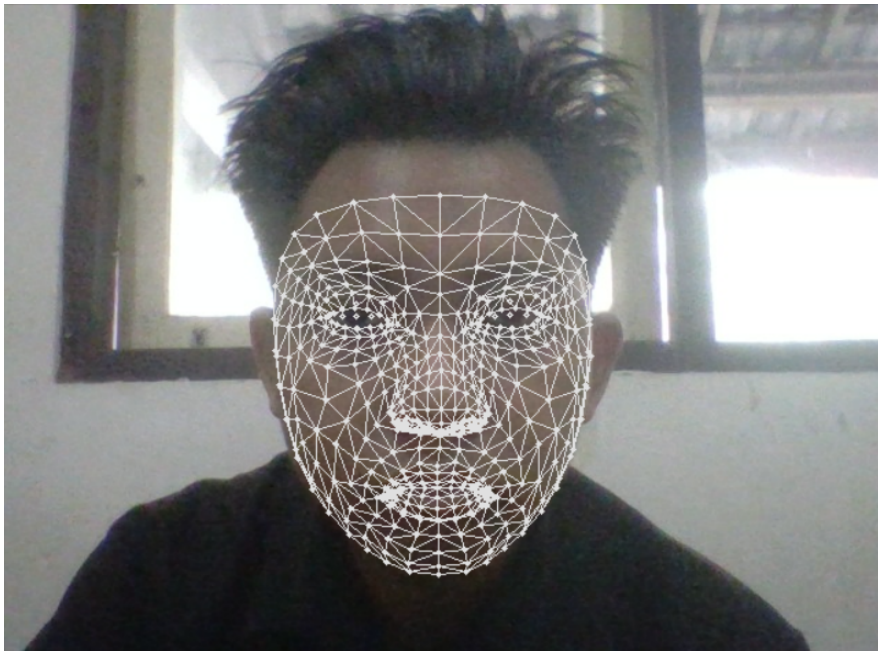
TABLE 1: Movement accuracy measurements

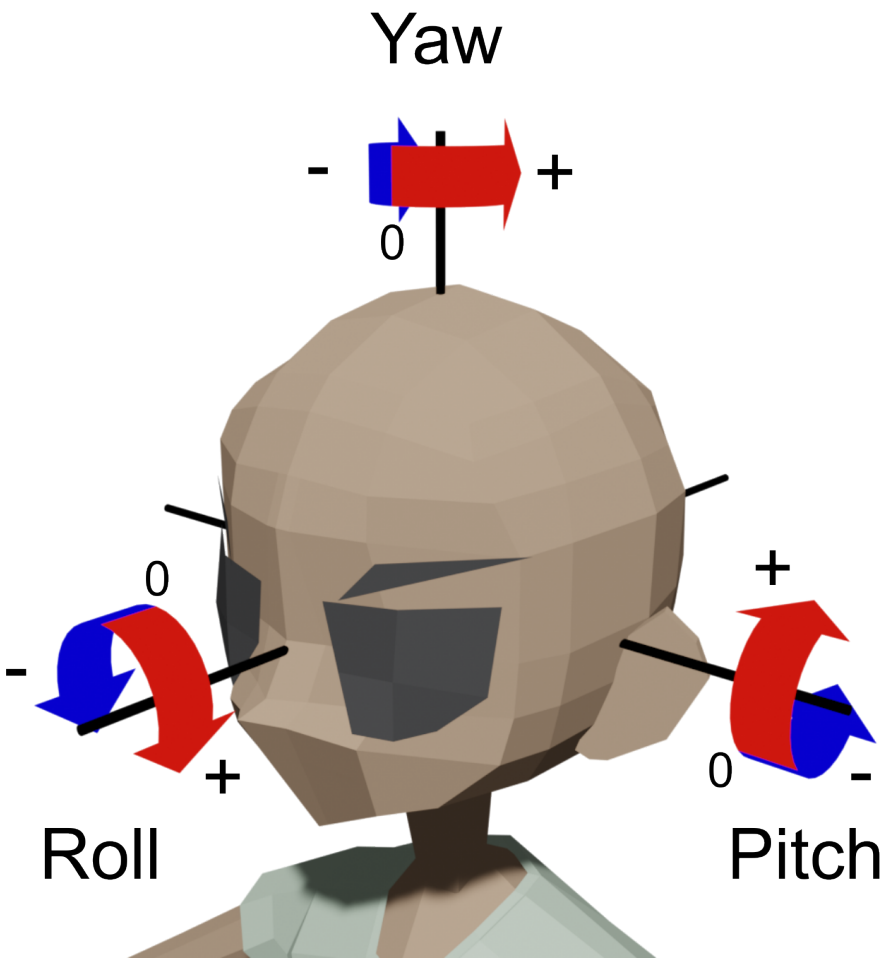
| | Pitch raw | Pitch filtered | Roll raw | Roll filtered | Yaw raw | Yaw filtered |
|---|---|---|---|---|---|---|
| Mean | -5.157678 | -5.15736 | -5.61493 | -5.61511 | 2.734456 | 2.736926 |
| Standard deviation | 10.94191 | 10.93055 | 25.16633 | 25.14577 | 15.30166 | 15.28227 |
| Standard error | 0.4008806 | 0.400464 | 0.922023 | 0.92127 | 0.560609 | 0.559899 |
| Mean absolute error | 7.815397 | 7.815075 | 5.724477 | 5.747263 | 3.242209 | 3.303836 |

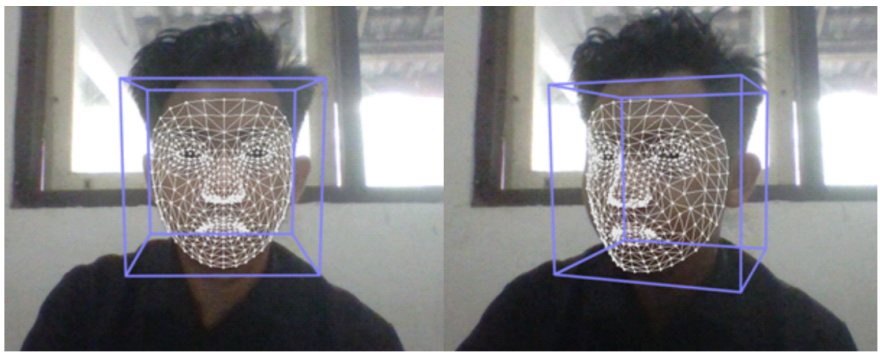
AUTHOR BIOGRAPHY**Izza Azzurri** is currently a bachelor student at the Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang, Indonesia. He is interested in research related to virtual world and artificial intelligence system. He is a member of UNNES Electrical Engineering Students Research Group (UEESRG).**Subiyanto** received B. Eng. from Universitas Diponegoro, Indonesia in 1998 and M. Eng. from Univesitas Gadjah Mada, Indonesia in 2003 and a Ph.D. degree from Universiti Kebangsaan Malaysia in 2012. He has been awarded the designation as an Insinyur (Ir.) by Institution of Engineers Indonesia. His thesis of Bachelor, Master and Ph.D. are always about the application and development of Artificial intelligence. He is currently a professor at the Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Semarang, Indonesia. He is interested in research related to electrical engineering, Intelligent Systems and their application. He is a member of IEEE, and coordinator of UNNES Electrical Engineering Students Research Group (UEESRG).

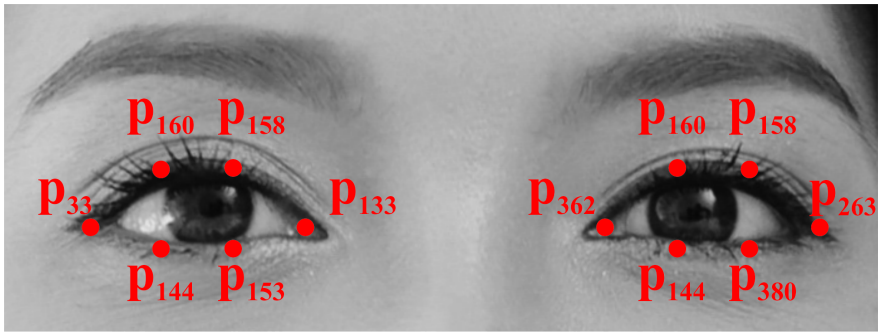


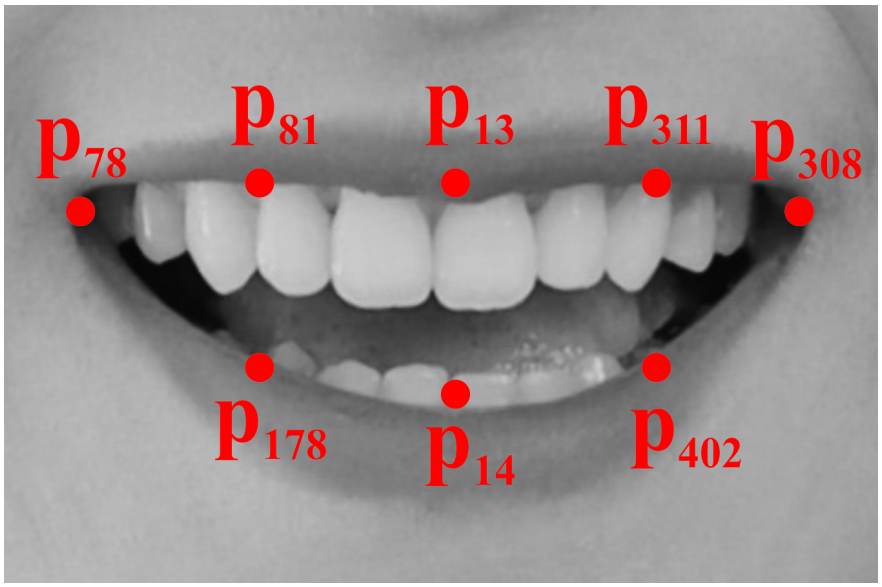


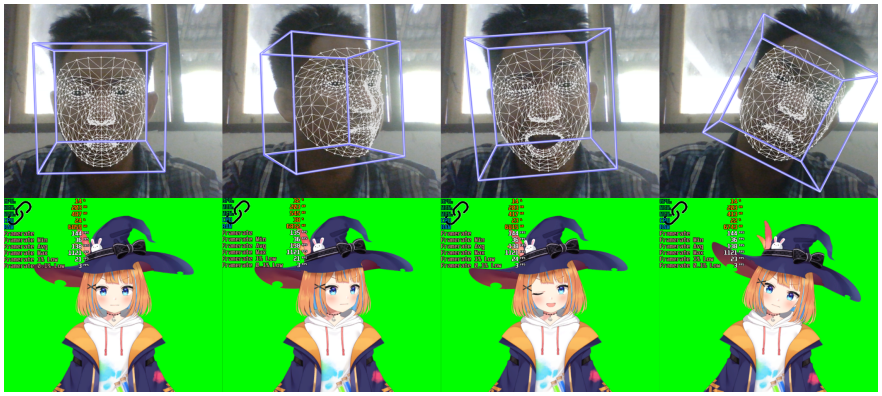



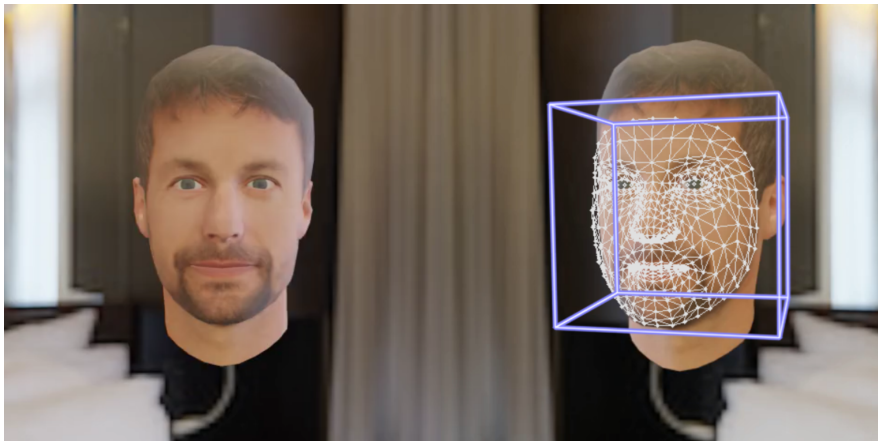


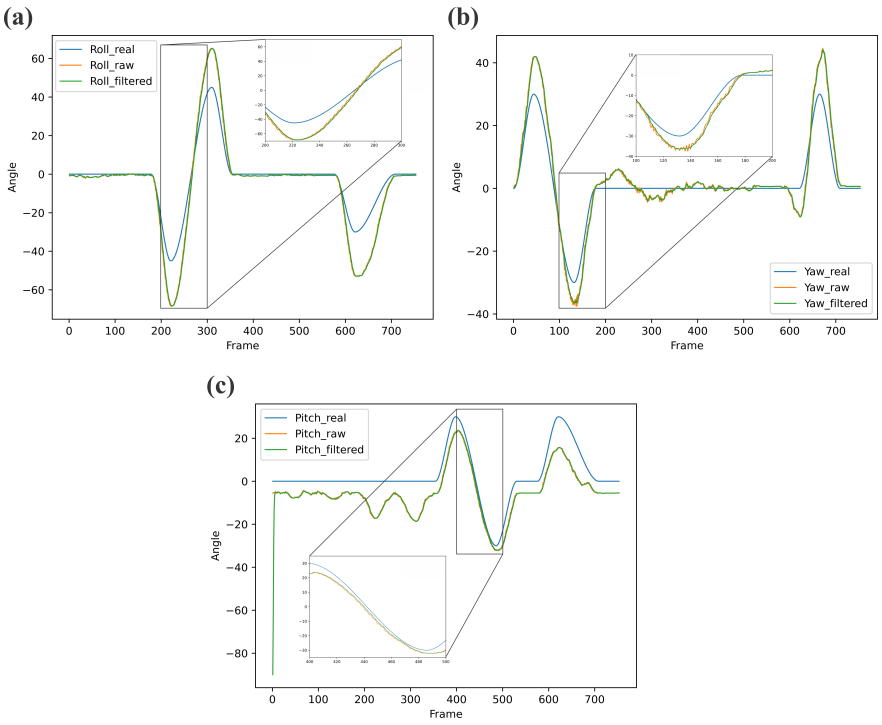



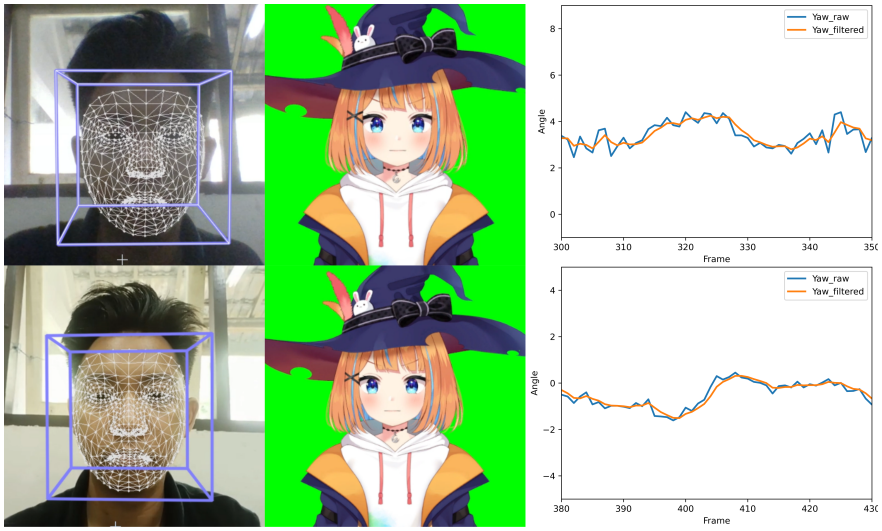









### Hosted file

`Table.docx` available at https://authorea.com/users/655316/articles/661171-application-of-smoothed-facial-motion-capture-system-for-live2d-vtuber-model