Automating animation with AI

BBCS Group 5: CGA

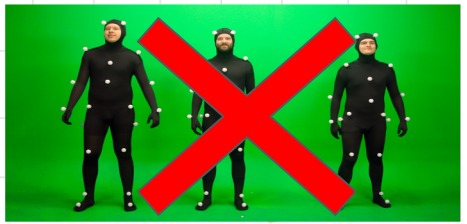
00

PURPOSE

Purpose of CGA

As a group our goal was to make improvements on the art of motion capture to make it more efficient , less time consuming and accessible to a larger audience. Through we hope that people would have a better experience of motion capture, and encourage others to try it to .

This CGA (computer generated animation) does not require the use of labour intensive animation processes or mocap suits used in CGI. However it still functions similarly by using computer vision technologies that pinpoints the motion using the landmarks of your body such as the cheek bones. Hence this makes it more accessible and usable to a wider community. It also makes the process of animation more cost effective and allows for higher quality of animation.



Team CGA aims to make CGI:

✅ Easily accessible

✅ Cost effective

01

PROTOTYPE

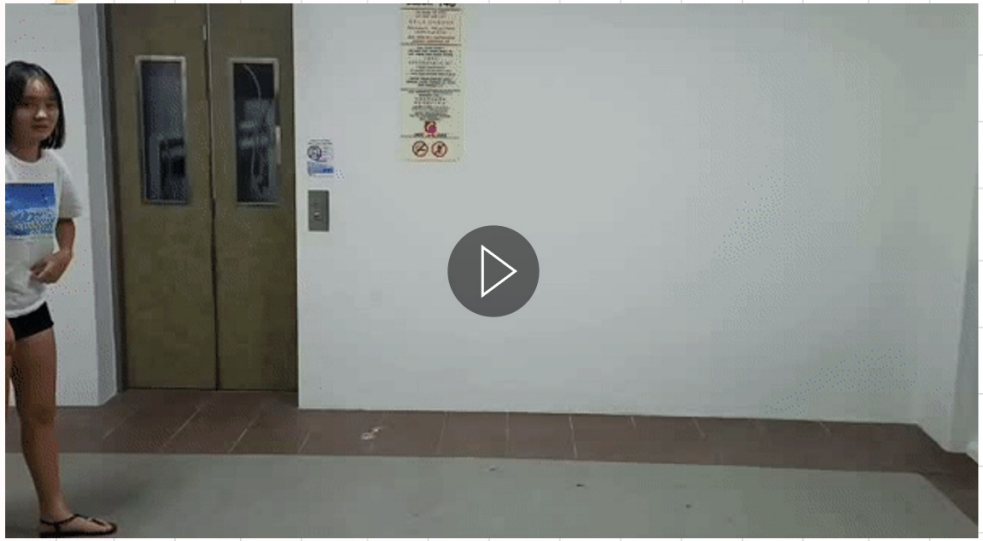
An overview

Introducing... the CGA!

Use your camera to take a video of the scene

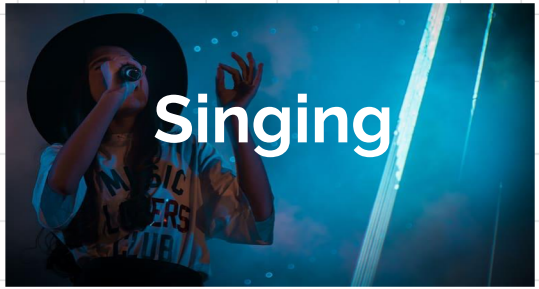
CGA

Instantly generates the animation of a character with the exact same movements!



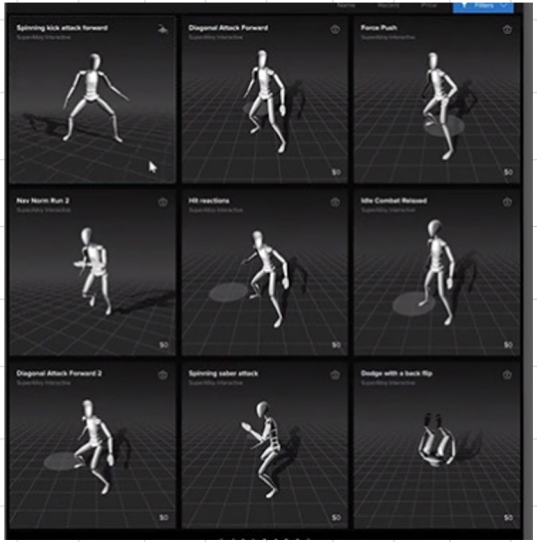
These videos can be found in
prototype_demo_shortvid.mp4

Search Activity in Database



And Many More!

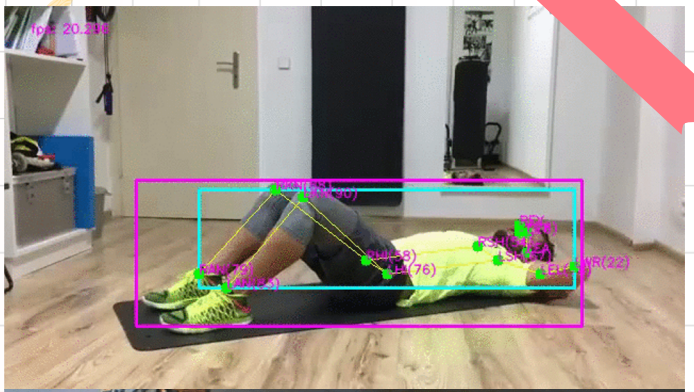
By creating this character just one time, this character will be able to perform any of the activities found in the Database

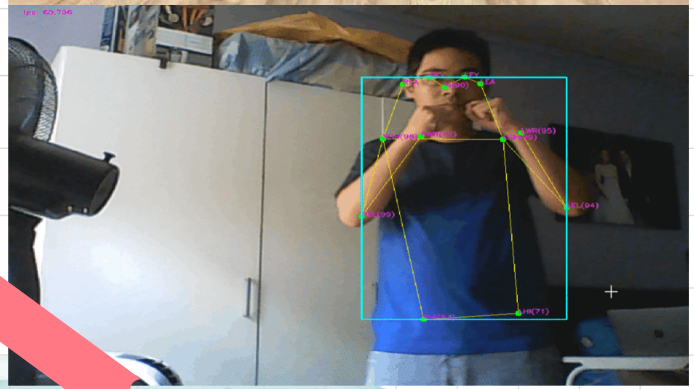


02

CONCEPTUAL -ISATION

AI BRICKS





Pose Estimation

- The detection and localisation of various joint/skeletal points on images.
- We can use this concept to estimate the pose of the person, to model the AI character.

Activity Classification

- Identification of activities using data driven approaches.
- We can have a library of activities of the character so that it will be at the disposal of the animators

AI Face Body and Hand Pose Detection with Python and Mediapipe

#0. Install and Import Dependencies

#1. Get Real time Webcam Feed

#2. Make Detections from Feed

#3. Apply Styling

#4. Capturing Animation Video

- Detect Facial Landmarks
- Detect Hand Poses
- Detect Body Poses

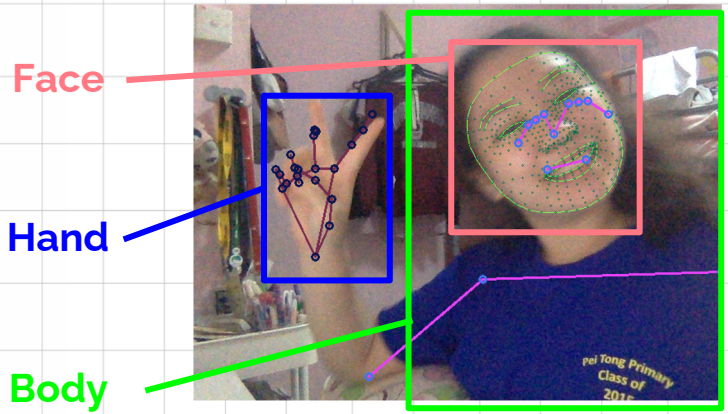
Full code can be found in
Media Pipe BBCS 2021.ipynb

How it works?

MediaPipe Hands and Face Mesh Solutions

What's this?
A machine Learning pipeline!

Identifying Region of Interest (ROIs)



How it works?

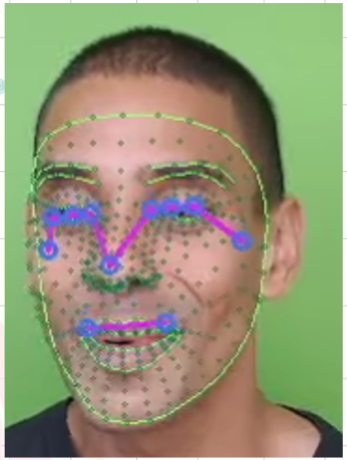
https://google.github.io/mediapipe/solutions/pose

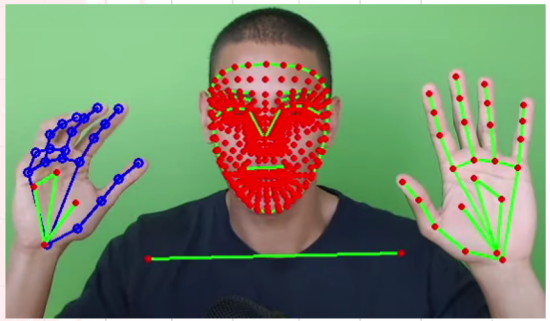
1. Locate the person within the frame
2. Predicts the pose landmarks
3. Connects landmarks according to categories (e.g the shoulders in body pose)

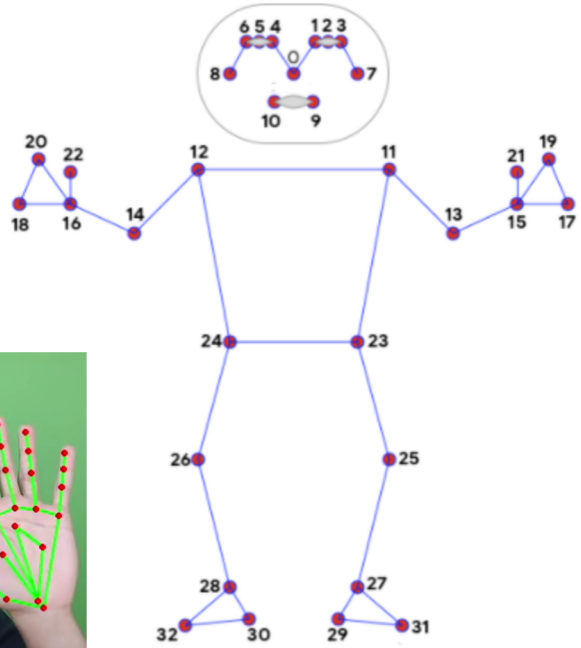
It consists of:

- Face detection
- Face mesh
- Hands
- Pose

BlazePose 33 keypoint topology





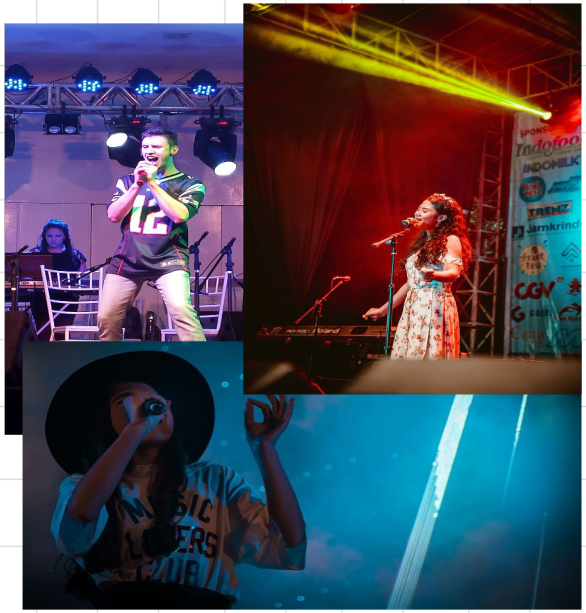


0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Activity Classification

Models are trained with many videos of similar activities

Fighting



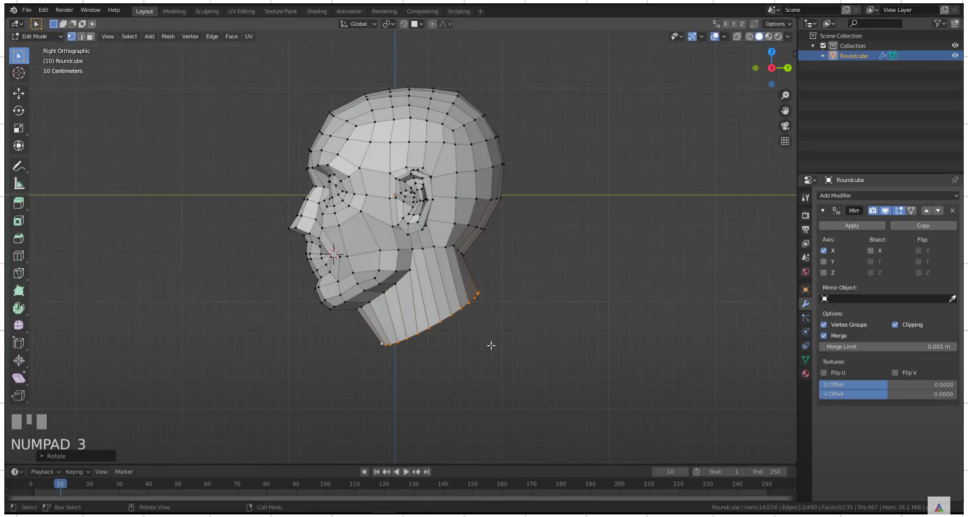
Singing

AI Face Body and Hand Pose Detection with Python and Mediapipe

- Using Mediapipe Library and a webcam, we can have:
  - Body pose detection
  - Facial landmark estimation
  - Hand pose detection

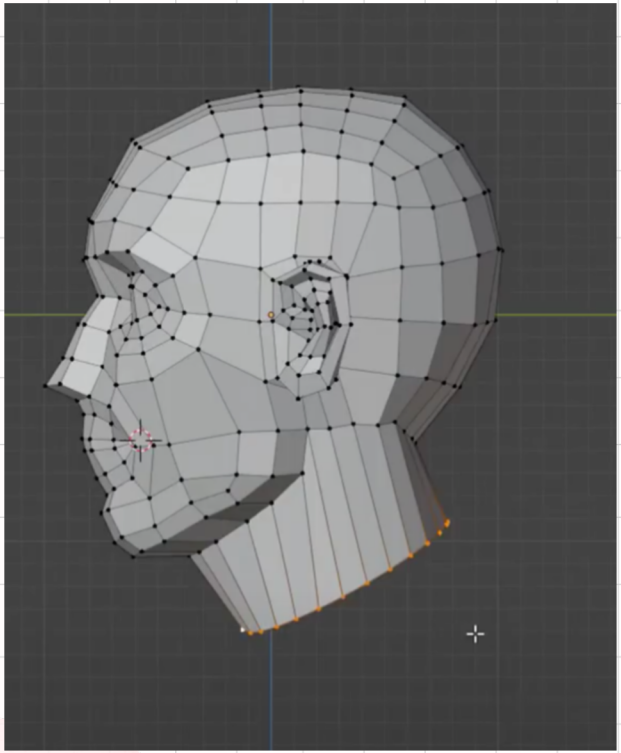
- These detections are given as points which we can then map to our animated figure

Application to Animated Figures

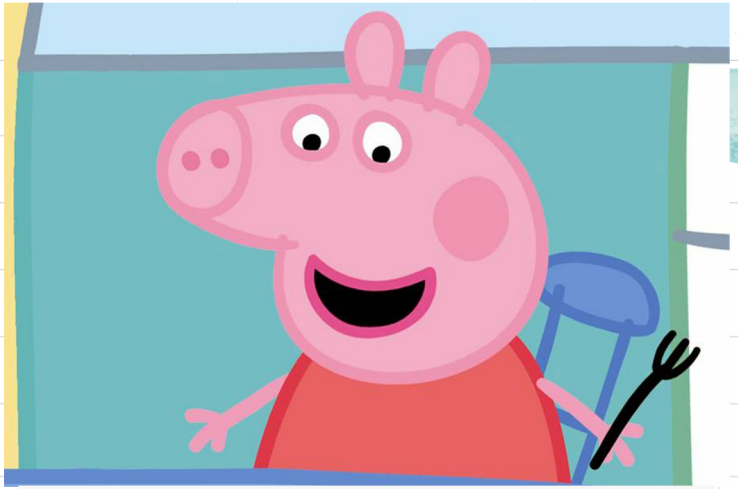


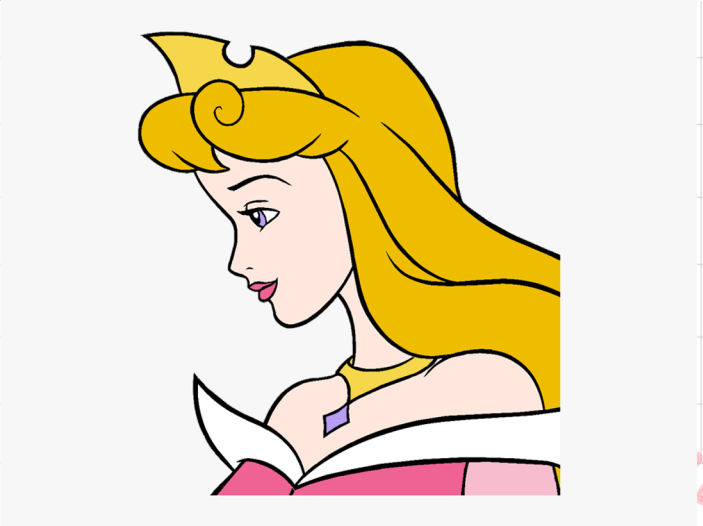
Making use of animation tool BLENDER

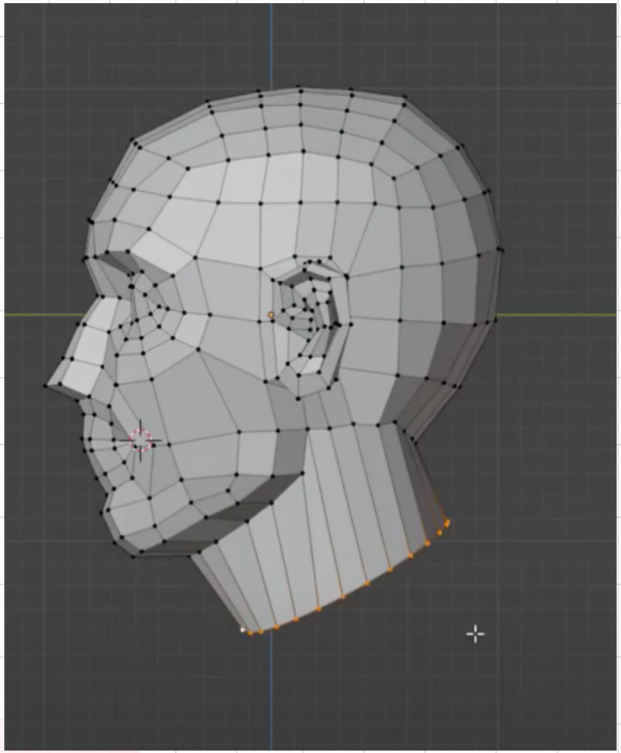
Create Characters using these feature maps



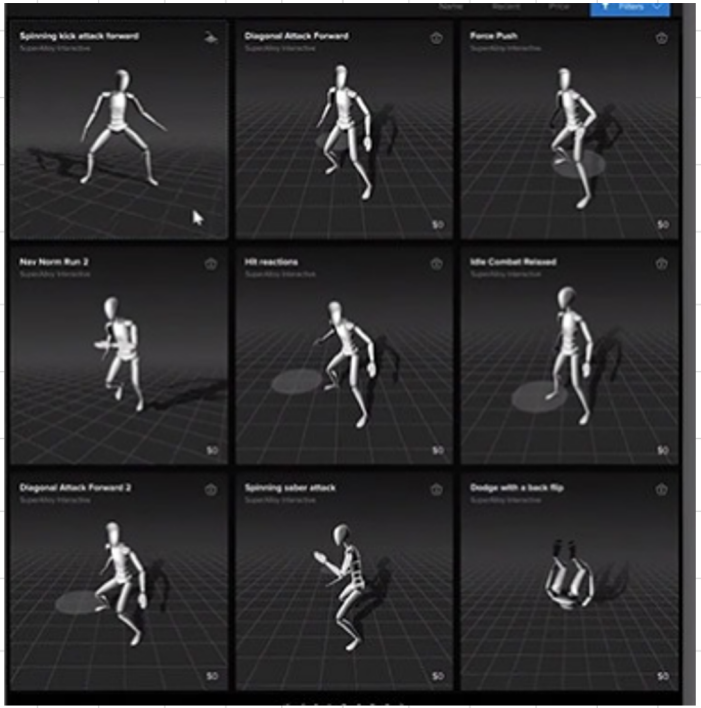
Apply to any other character by creating these characters with similar feature maps







Character can be made to do any activity after obtaining feature maps



03

CODE

The code (full version on GitHub)

```
import mediapipe as mp
import cv2 as cv2
from google.colab.patches
import cv2_imshow
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:

    #ret, frame = cap.read()

    frame = cv2.imread("Unknown.jpg")

    # Recolor Feed
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Make Detections
    results = holistic.process(image)
    # print(results.face_landmarks)

    # face_landmarks, pose_landmarks,
left_hand_landmarks, right_hand_landmarks

    # Recolor image back to BGR for rendering
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    # 1. Draw face landmarks
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                              )

    # 2. Right hand
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                              )
    # 3. Left Hand
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                              )

    # 4. Pose Detections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                              )

    cv2_imshow(image)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

The code (full version on GitHub)

```
cap = cv2.VideoCapture(0)
# Initiate holistic model
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:

    #ret, frame = cap.read()

    frame = cv2.imread("Unknown.jpg")

    # Recolor Feed
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    # Make Detections
    results = holistic.process(image)
    # print(results.face_landmarks)

    # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks

    # Recolor image back to BGR for rendering
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

04

SOURCES

Sources

AIBricks: https://aisingapore.org/ai-bricks/

Mediapipe: https://mediapipe.dev/

Mediapipe tutorial: https://github.com/nicknochnack/Full-Body-Estimation-using-Media-Pipe-Holistic

Blender Animations: https://youtu.be/fFxkTY4Z_lM , https://youtu.be/WlaMfIgS2ns


CGA