

DEVELOPMENT OF A MARKERLESS MOTION CAPTURE SYSTEM BASED ON OPENPOSE

THESIS

**Submitted in partial fulfillment of the requirements
for the degree of Master of Engineering from the Institut Teknologi Bandung**

**By
Alvin
NIM: 23121017
Faculty of Mechanical and Aerospace Engineering**



**INSTITUT TEKNOLOGI BANDUNG
December 2022**

ABSTRAK

PENGEMBANGAN SISTEM PENANGKAP GERAK TANPA PENANDA BERBASIS OPENPOSE

Oleh

Nama Alvin

NIM: 23121017

Fakultas Teknik Mesin dan Dirgantara

Dalam penelitian sebelumnya, telah dibuat beberapa model untuk memprediksi gaya reaksi tanah secara 3D dengan metode LSTM (*Long Short-Term Memory*). Model tersebut bekerja dengan menerima data kinematika gerak yang didapatkan dari penangkap gerak sebagai inputnya. Akan tetapi, terdapat beberapa kelemahan pada sistem penangkap gerak yang telah dikembangkan oleh Lab Biomekanika ITB yang diantaranya ialah gerak harus dilakukan dalam ruangan gelap dan penanda cenderung bergoyang saat subjek bergerak. Untuk mengatasi kelemahan diatas, metode estimasi pose manusia yang berdasar pada *Deep Learning* dan *Computer Vision* dapat diterapkan sebagai sistem penangkap gerak tanpa penanda. Pada tahun 2020 terdapat kemajuan yang pesat saat model OpenPose dirilis karena dapat memprediksi pose manusia secara instan meski terdapat banyak subjek dalam gambar. Namun, telah dibuktikan bahwa OpenPose tidak memiliki akurasi yang cukup untuk menggantikan sistem penangkap gerak tanpa penanda. Oleh sebab itu, pada penelitian ini dilakukan *fine-tuning* terhadap OpenPose agar dapat memprediksi dengan lebih akurat untuk tugas penangkapan gerak berjalan atau berlari manusia. Namun untuk dapat melakukan *fine-tuning* tersebut, diperlukan data gambar yang dapat menunjukkan subjek dan penanda dengan jelas. Lalu dilakukan juga *inpainting* untuk menghilangkan penanda dari gambar subjek sehingga didapatkan gambar subjek tanpa penanda dan posisi penanda tersebut sebelum dihilangkan. Setelah dilakukan pelatihan ulang, ditemukan bahwa akurasi model dinilai cukup dengan kesalahan rata-rata 17-pixel.

Kata kunci: Penanda, *fine-tuning*, OpenPose, penangkap gerak

ABSTRACT

DEVELOPMENT OF A MARKERLESS MOTION CAPTURE SYSTEM BASED ON OPENPOSE

By

Alvin

NIM: 23121017

Faculty of Mechanical and Aerospace Engineering

In previous research, several models to predict the 3D Ground Reaction Forces of running had been created using LSTM (Long Short-Term Memory). The models take kinematic data obtained through motion capture as input. However, there were disadvantages to the currently developed method in ITB Biomechanics Lab such as the need for a dark room, or the tendency of markers to shake while the subject is moving. To overcome this, human pose estimation based on deep learning and computer vision can be applied as a markerless motion capture system. There was a huge leap in this field back in 2020, by a bottom-up approach model OpenPose. This model can predict human pose in real-time even for multi person problem. But recently, it was reported that OpenPose is not as accurate as marker-based or sensor-based motion capture. Hence, to alleviate this inaccuracy, it is proposed to fine-tune the original OpenPose with motion capture data so that it will learn and perform better for motion capture task especially for running or gait. But to get the fine-tuning done, the subject and marker should be clearly visible. Later, the frames will be inpainted to hide the marker shown in the image leaving only the subject for the input data. Lastly, after training it was discovered that the model has good accuracy with the average of 17-pixel error.

Keywords: marker, fine-tuning, OpenPose, motion capture

APPROVAL SHEET

DEVELOPMENT OF A MARKERLESS MOTION CAPTURE SYSTEM BASED ON OPENPOSE

By
Alvin
NIM: 23121017
Faculty of Mechanical and Aerospace Engineering
Institut Teknologi Bandung

Approval on
Date: 12th December 2022

by
Supervisor



Prof.Ir. Andi Isra Mahyuddin Ph.D.

Co-Supervisor 1



Ferryanto, ST., MT.

Co-Supervisor 2



Pramudita Satria Palar, S.T, M.T, Ph.D.

ACKNOWLEDGEMENT

First, I would like to express my gratitude and sincere thanks to my respected supervisors Prof.Ir. Andi Isra Mahyuddin Ph.D., Ferryanto, ST., MT., and Pramudita Satria Palar, S.T, M.T, Ph.D. for the guidance, insight, and positive criticism they gave throughout the process of this work. Next, I would also like to thank my friends Nardo Rizaldy and Nico Jackson that helped me during the data acquisition process because without them it would be impossible to finish in a month. Lastly, I would also like to express my gratitude for all the subjects that has been willing to come to ITB during Covid-19 pandemic and without them there would be no data for this work to be done.

CONTENT

ABSTRAK	i
ABSTRACT	ii
APPROVAL SHEET	iii
ACKNOWLEDGEMENT	iv
CONTENT	v
LIST OF FIGURES	vi
LIST OF TABLES	vii
Chapter I Introduction.....	1
I.1 Research Background.....	1
I.2 Research Background.....	3
I.3 Research Purposes.....	3
I.4 Problem Limitation.....	3
I.5 Methodology.....	4
Chapter II Literature Review	7
II.1 Motion Capture.....	7
II.2 Convolutional Neural Network (CNN)	8
II.3 OpenPose	12
II.4 Lightweight OpenPose	13
II.5 Transfer Learning	13
II.6 HSV Color Space	14
II.7 Image Inpainting.....	15
II.8 COCO Dataset Format	16
II.9 FFmpeg.....	16
II.10 Git.....	17
Chapter III Data Acquisition and Processing.....	18
III.1 Data Acquisition.....	18
III.2 Data Processing	21
Chapter IV MODEL TRAINING AND EVALUATION.....	26
IV.1 Model Training.....	26
IV.2 Model Evaluation	28
Chapter V APPLICATION DEVELOPMENT	32
V.1 Application Design.....	32
V.2 Installation Guide	33
V.3 User guide.....	35
V.4 Development guide.....	38
CHAPTER VI SUMMARY AND FURTHER RESEARCH.....	39
VI.1 Summary.....	39
VI.2 Further Research.....	39
REFERENCES	40

LIST OF FIGURES

Figure I.1 Research flowchart.....	5
Figure I.2 Research flowchart (continued)	6
Figure II.1 FFNN input data feeding illustration [13]	9
Figure II.2 CNN input data feeding illustration [13]	9
Figure II.3 Feature extraction illustration [12]	10
Figure II.4 OpenPose's Pipeline [7, 18, 19].....	12
Figure II.5 OpenPose's Architecture [18].....	13
Figure II.6 Transfer learning in VGG-16 for semantic segmentation [13, 27].....	14
Figure II.7 RGB vs HSV color space [28].....	15
Figure II.8 LaMa Inpainting [36].....	16
Figure III.1 Marker variant created for HSV motion capture system	19
Figure III.2 Final Setup.....	19
Figure III.3 Subjects' BMI distribution.....	20
Figure III.4 HSV motion capture result	22
Figure III.5 Image inpainting process	24
Figure III.6 COCO Dataset Format	25
Figure IV.1 Training loss history extracted from model training log	27
Figure IV.2 First model result.....	28
Figure IV.3 First model error on predicting video without markers.....	29
Figure IV.4 Median blur effect	30
Figure IV. 5 First and second model evaluation comparison	30
Figure V.1 Project structure	33
Figure V.2 Docker desktop User Interface	34
Figure V.3 Launching the web app.....	35
Figure V.4 Web app User Interface	36
Figure V.5 Markerless motion capture User Interface.....	37
Figure V.6 Data visualisation UI	38

LIST OF TABLES

Table I.1 Model's performance [1].....	1
Table II.1 Activation Function in deep learning [15]	11

Chapter I Introduction

This research was proposed as continuation of the final project “3D Running Ground Reaction Forces Prediction using LSTMs based on kinematics parameters”. There are several considerations made during decisions making and will be explained in this chapter.

I.1 Research Background

In the final project, LSTM (Long Short-Term Memory) based models had been created and successfully predicted the 3D Ground Reaction Forces of running with decent accuracy [1]. These models take kinematics data, position, velocity, and acceleration, as their input, which can be obtained by using motion capture on a subject running on a treadmill. The accuracy was evaluated using the test dataset which is presented in Table 1.1. The Ground Reaction Forces (GRF) prediction was motivated by how overpriced an instrumented treadmill are. But to be able to pass inputs to the models, a robust method on capturing running motion was needed.

Table I.1 Model's performance [1]

Speed	GRFX (Anteroposterior)		GRFY (Vertical)		GRFZ (Mediolateral)	
	Mean RMSE (\pm S.D.)	Mean Coeff. Correlation	Mean RMSE (\pm S.D.)	Mean Coeff. Correlation	Mean RMSE (\pm S.D.)	Mean Coeff. Correlation
2,5 m/s	1,87E-02 \pm 7,71E-03	0,990	4,66E-02 \pm 4,12E-02	0,998	7,00E-03 \pm 2,50E-03	0,987
3,5 m/s	1,92E-02 \pm 6,33E-03	0,994	4,38E-02 \pm 2,35E-02	0,999	7,50E-03 \pm 4,60E-03	0,983
4,5 m/s	1,93E-02 \pm 4,90E-03	0,996	4,30E-02 \pm 1,34E-02	0,999	8,00E-03 \pm 2,40E-03	0,987
Average	1,91E-02 \pm 6,31E-03	0,993	4,45E-02 \pm 2,60E-02	0,999	7,50E-03 \pm 3,17E-03	0,985

There have been several motion capture methods that were developed in ITB Biomechanics Laboratory. The first one is the optical marker-based motion capture which use LED (Light Emitting Diode) lights as markers [2]. This method was used

most often because of its simplicity in the image segmentation method. Like many other active marker-based motion capture methods, this method should be conducted in a dark room to contrast markers against the background so that it will be easy to segment the markers from the background. But this method has several shortcomings like the markers tend to shake while the subject is moving, it takes time to attach all the markers, and lastly the darkroom itself.

The other method is the markerless motion capture that was developed in 2020 [3]. This method allows us to do motion capture in a bright room without any marker which overcomes most of the problems stated before. However, the method created at that time was not able to analyze 3D motion as it depends on the sagittal view of the subject. Moreover, it is computationally costly because it used Particle Swarm Optimization to adjust the position of the human model to the silhouette at every single frame.

To overcome the disadvantages of the developed method by Lab Biomekanika, the author looks for another alternative that can analyze human pose on every motion and every is angle view free. In the year around 2010 there was a popular method called human pose estimation. This method relies on Artificial Intelligence especially Deep Learning as its base model for predicting the joint position of human pose. It started from tree structured model [4] to CNNs model based [5] for reliable local observations on the image. Moreover, the method also evolved from single person pose estimation to multiple persons pose estimation by changing from top-down strategy to bottom-up approach. The bottom-up approach was proved to have much less computational cost as it did not need to predict every person at first which solve the multiple person complexity in a single frame [6]. As an addition, PAFs (Part Affinity Field) was also introduced so the model can differentiate between a person and another accurately. This was proposed by research called the OpenPose model [7]. This model was using the PAFs as vector to encode unstructured pairwise relationships between body parts and heatmaps as the joint position probability itself. It could even handle multi-person prediction in real-time. Although it is still 2D, this can be used to substitute the early stage of marker-based

motion capture up to the segmentation method. But because it was trained by using a random and general human pose dataset, it was not accurate enough for motion capture purposes compared to the conventional marker-based motion capture or even inertial sensors [8, 9].

To overcome OpenPose shortcomings, in this research, the original OpenPose will be fine-tuned using motion capture data. By fine-tuning, the parameters inside the original OpenPose model will be adjusted to fit the motion capture task and thus have better accuracy. In addition to that, the motion capture data should consist of both images of the subject in a bright room that will clearly show the subject and the marker positions. This kind of data can be obtained by using color-based motion capture data that will also be developed in this research.

I.2 Research Background

The problems identified for this research consist of:

1. There has not been any developed motion capture method in ITB Biomechanical Laboratory that can capture subject in bright room using marker.
2. OpenPose's accuracy is not satisfactory for motion capture task compared to marker-based or inertial sensors-based motion capture.

I.3 Research Purposes

This research's purposes can be divided into two main purposes, which are:

1. To create a color-based motion capture method for collecting the data required for fine-tuning.
2. To fine-tune OpenPose using motion capture data so that it will have better accuracy in motion capture tasks.

I.4 Problem Limitation

In this research, several limitations are applied to simplify the problem. These limitations consist of:

1. The motions that will be analyzed are only walking and running.

2. There will be only one subject in the frame.

I.5 Methodology

This research pipeline can be divided into two main parts. The first one is to create color-based motion capture and the other is to fine-tune the OpenPose using the data collected from color-based motion capture. These processes can be represented using a flowchart as shown in Figure I.1.

Color-based motion capture will be created using Python and OpenCV library in Anaconda3. The process starts with reading the video and dividing it into frames. The OpenCV will read the image in BGR (Blue Gray Red) format by default, so next, it needs to be converted into HSV (Hue Saturation Value) format. After converting the frames, thresholding can be done by selecting the parameters, H, S, and V value to segment the specified color resulting blobs. Lastly, the centroid of each blob will be detected and exported in a CSV (Comma Separated Value) file. The marker used was created by 3D printing and then it will be painted green.

To fine-tune the OpenPose, some data should be collected using the previously created color-based motion capture. But these data should be the images showing only the subject without the marker and the position of the marker. To obtain the images without any marker, preprocessing should be conducted to the original images captured by the camera. This preprocessing step was called inpainting, a process to infer back the part of a picture using deep learning. After the markers were hidden out, the images and corresponding marker positions then will be paired to be a dataset. Next, this dataset was split into the train set and test set. The train set was used to fine-tune all the parameters inside the original OpenPose, and the test set was used to evaluate the fine-tuned OpenPose. The model then was tested on a video without any marker for checking whether the model really worked as expected or not. Moreover, deployment pipeline was also built to deliver this model so that it can be used in the future. In Machine Learning Operations (MLOps), creating a model was usually called research phase and deploying the model for use is usually called the deployment phase. The deployment phase usually included the

Application Programming Interface (API), Docker and many other services. Although in this research, the author has not planned until the monitoring phase but the deployment should be sufficient for further research. Therefore, a project repository was also built along the way by using GitHub.

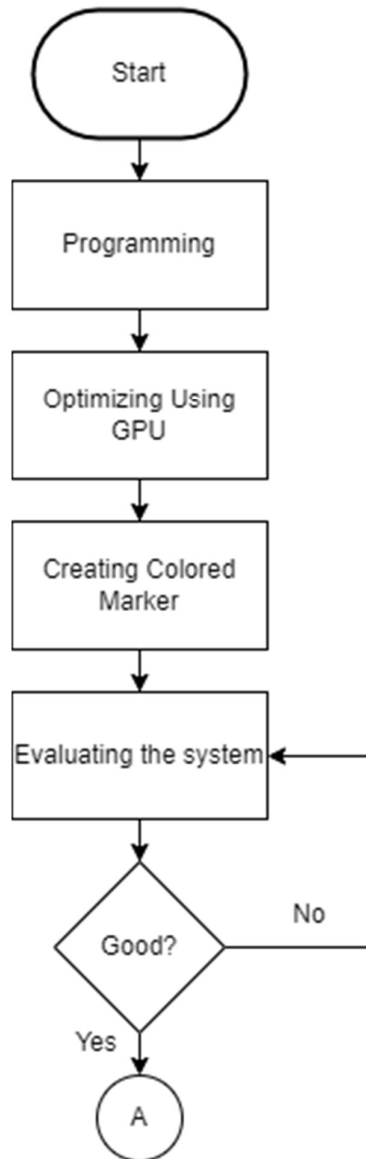


Figure I.1 Research flowchart

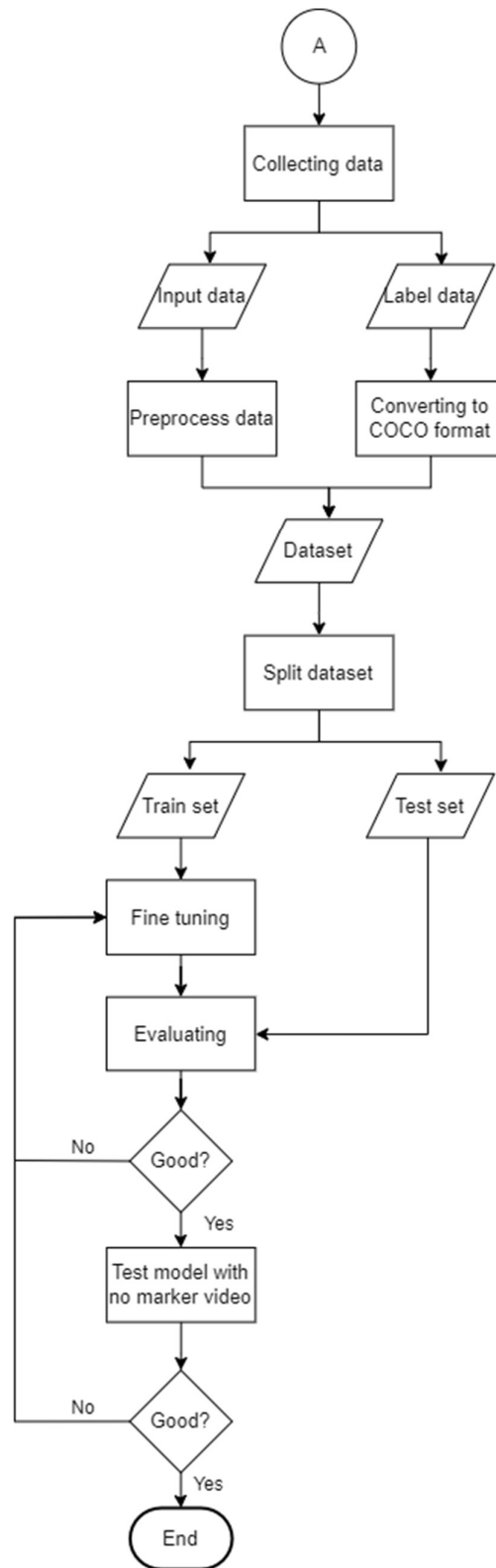


Figure I.2 Research flowchart (continued)

Chapter II Literature Review

In this chapter, topics related to the research question which are motion capture, Convolutional Neural Network (CNN), OpenPose, transfer learning and fine-tuning, and lastly the HSV color space are discussed. Other than that, some additional information regarding image inpainting, COCO dataset, and ffmpeg are also discussed.

II.1 Motion Capture

Motion Capture is the process of tracking some points' position using a camera or other sensors. There have been some existing motion capture methods developed in ITB Biomechanics Lab. All of them are using a different strategy to track the position, especially the image segmentation process. The first 3D optical motion capture developed in ITB Biomechanics Lab is using LED lights as the active markers [2]. This method is efficient if it was conducted in a dark room as the lights will contrast against the background and made it easier to segment. But because of the dark room itself, there were some disadvantages like safety and motion blur due to lowlight recording because optical devices like camera need sufficient light to work optimally. Next is the markerless motion capture developed back in 2020. This method is done without any marker attached to the subject. The segmentation was done by contrasting the background against the clothes worn by the subject. In addition, there will be a human model generated manually using pixel label and its position will be adjusted to follow the silhouette obtained after segmentation [3].

All the methods mentioned above have their advantages and disadvantages but overall, all the optical motion captures will have the same pipeline as follows:

1. Camera Calibration

This process's purpose was to obtain the intrinsic and extrinsic parameters of our camera. The extrinsic parameters consist of the translation and rotation of the camera, while the intrinsic parameter consists of the focal point, skewness, and scale. These parameters will be used to convert all pixel coordinates to the global coordinate in the reconstruction process.

There are some methods to conduct the calibration and one of them is by using the checkerboard method. There is also a library in MATLAB for this checkerboard method.

2. Video Capture

In this process, the subject's motion will be recorded by using the same calibrated camera. It should be kept in mind that no repositioning is allowed after the calibration is conducted.

3. Position tracking

After recording, the position can be tracked by using several methods based on the case. For the active marker-based motion capture, the frame can simply be formed into binary images by setting the threshold level. In color-based motion capture, we can threshold the HSV images by selecting the color region. This will be explained in more detail in Chapter 2.6. On the other hand, markerless motion capture has much more complicated process to track the position in the frame. The markerless motion capture developed in ITB Biomechanics Lab, it is using a human body segment model generated by using pixel label. This model orientation then will be adjusted to follow the silhouette obtained from the image segmentation. The adjustment is done by using Particle Swarm Optimization.

4. Reconstruction

Lastly, all the positions in pixel coordinate obtained in the previous step will be converted into global coordinates using the intrinsic and extrinsic camera parameters.

II.2 Convolutional Neural Network (CNN)

CNN is one of the deep learning architectures created to overcome the shortcomings of FFNN (Feed-Forward Neural Network), the simplest deep learning architecture when dealing with image data [10, 11]. The main problem that arises in the FFNN is the high computational cost when calculating all the parameters needed only to receive one image with a resolution of 480x640 pixels [12]. The FFNN will interpret all the pixel values in the images as individual data, and each will have its parameter. Other than that, the FFNN will also have no ability to learn the spatial

pattern contains in the image. It was because to feed the image as input data in FFNN, it should first be flattened into a vector matrix. The FFNN's process when it takes input is shown in Figure II.1.

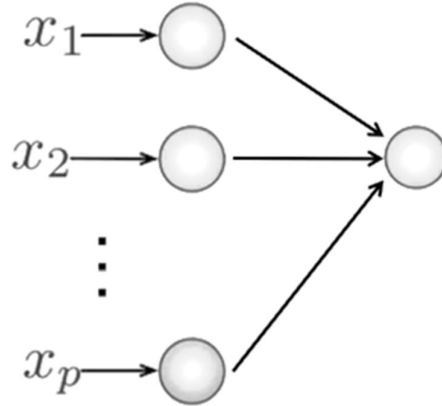


Figure II.1 FFNN input data feeding illustration [13]

Different from FFNN, CNN will take the images as input data batch by batch. This way CNN can still maintain the spatial pattern in the images and make it robust against local variation. The illustration can be seen in Figure II.2. In addition to that, the number of parameters that should be learned also decrease significantly compared to FFNN [11].

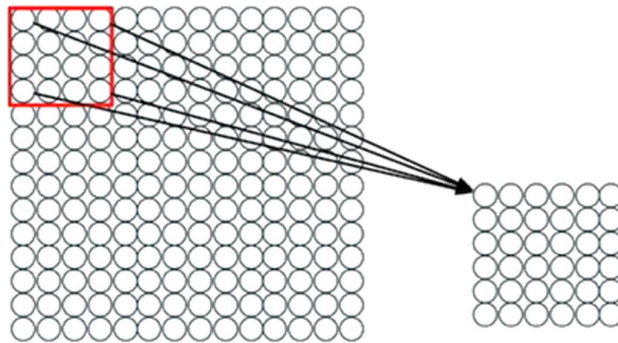


Figure II.2 CNN input data feeding illustration [13]

CNN process can be divided into three main processes which are feature extraction, weighting and applying non-linearity, and lastly down sampling.

1. Feature extraction using Convolution

Feature extraction is the process to extract patterns or information in the images. The features learned is similar to how human firstly infer an image. This extraction can be done by filtering the image, doing the element-wise multiplication, and summing all the results. These processes then were named convolution. After that, a feature map then will be obtained. Figure II.3 shows how a filter can extract information or feature from an image and as shown it has successfully extracted the edge of the image. This filter then was named Sobel Filter in image processing.

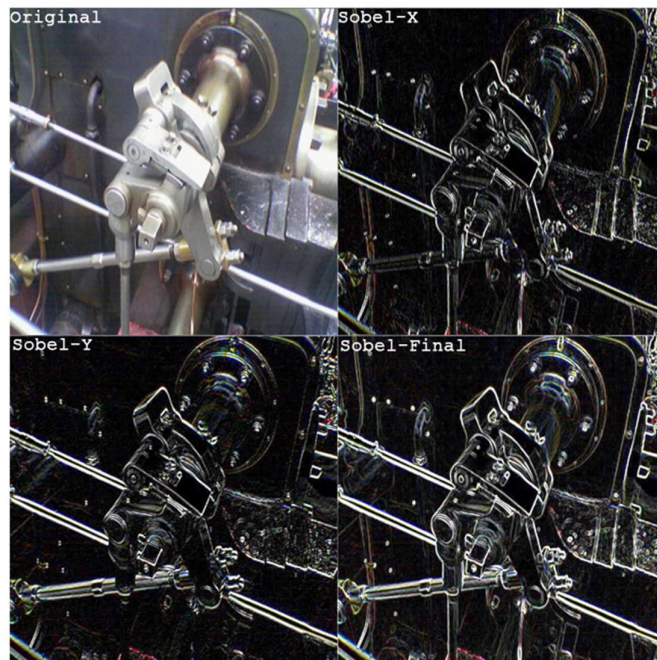


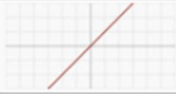


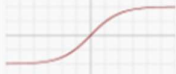


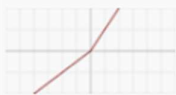


Figure II.3 Feature extraction illustration [12]

2. Weight and nonlinearity

This process is like all other deep learning architectures, where all feature maps will be weighted using random parameters which will be optimized to be a deep learning model. There are also nonlinearities applied by using the

activation functions like sigmoid, tanh, ReLU, and many others [14]. Some of the activation functions can be seen in Table II.1.

Table II.1 Activation Function in deep learning [15]

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$

3. Downsampling/Pooling

Pooling is a process to reduce the size of the spatial dimensions and the number of parameters inside the model [16]. Usually, the farther it goes in the pooling process, the more complex the features to be learned by the model will be. One of the pooling techniques is Max pooling which takes the maximum value from each small part of the Feature Map. This process also makes the model robust to spatial variation [17].

II.3 OpenPose

OpenPose is a deep learning model created to perform the Human Pose Estimation task. This model uses the bottom-up approach where each body segment of all humans that appears in the image will be predicted first before identifying the human[18]. By using this approach, the inference time is not linearly proportional to the number of people shown in the frame. OpenPose also utilizes the PAFs (Part Affinity Fields) to represent segments and the Confidence Map to represent the location of important points on the human body in the image. It was also proven that PAFs can significantly increase the accuracy of the Confidence Maps [7, 18, 19]. An illustration of the OpenPose pipeline is shown in Figure II.4.

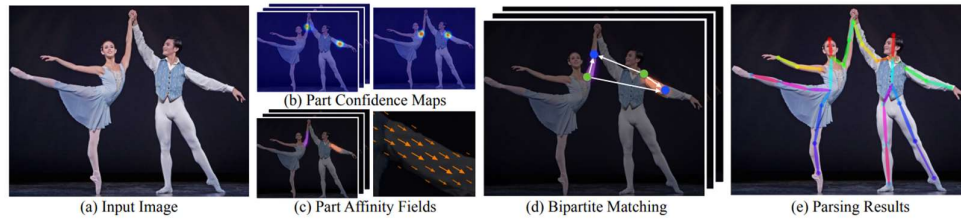


Figure II.4 OpenPose's Pipeline [7, 18, 19]

OpenPose architecture was constructed by the first tenth layer in VGG-19 [20] and then followed by multi-stages CNN. The VGG19 was a pretrained model which means that it was trained using numerous amount of images and was usually used for classifying images. Each successful stage will improve the prediction accuracy where L2-loss will be used to optimize all the parameters. The OpenPose architecture is shown in Figure II.5. The first half was to predict the PAFs which was in charge in differentiating between a person and the other while the last half was to predict the confidence map to evaluate the joint position probability. This architecture is a bit different compared to the previous OpenPose model as it was more computationally efficient [18]. This was accomplished by modifying the kernel size so that the total learnable parameter decreased.

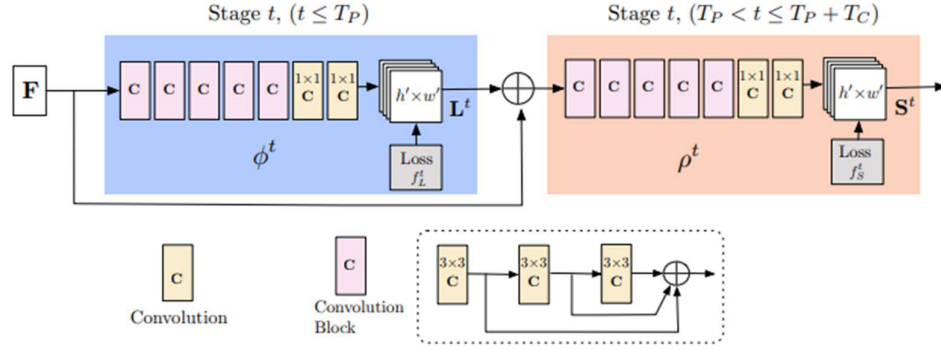


Figure II.5 OpenPose's Architecture [18]

OpenPose is evaluated by using mean Average Precision (mAP) metrics which are based on Precision and Recall [21, 22]. Precision is the comparison between correct guesses (TP) and total guesses that were made (TP+FP). Meanwhile, Precision is the comparison between correct guesses and all the correct answers (TP + FN) which means that is more focusing on multi-person prediction. Ideally the result should be balanced because good precision with bad recall can be easily obtained by just doing small number of predictions. While bad precision with good result can also be easily obtained by consistently predicting one class besides of the other.

II.4 Lightweight OpenPose

This method was a lightweight version of OpenPose developed by Intel [23]. The decrease of computational load was accomplished by replacing the backbone from VGG [20, 24] to MobileNet [25], an efficient Convolutional Neural Networks for that was famous for mobile vision application. By doing this, the complexity ratio was increased in more than 6.5 times without significant drop of accuracy and the speed can reach to around 28 fps Intel® NUC 6i7KYB mini-PC and 26 fps on Core i7-6850K CPU.

II.5 Transfer Learning

Transfer Learning is the process of applying a pre-trained model for a different task by replacing the outermost layer, which is responsible for producing output. By replacing the outermost layer, the model can be retrained for different tasks at a

much faster rate than retraining a model from scratch [26]. This is because the initial layer of the neural network usually interprets basic information, especially in CNN. By retaining the initial layer, the model will also retain the basic knowledge that may also be needed for the other tasks. This transfer learning process can be illustrated in Figure II.6. In this example, the transfer learning process was conducted on VGG-16, which was initially trained to classify images into 1000 classes, to perform semantic segmentation or pixel-wise classification. This can be achieved by maintaining all the initial layers of VGG-16 up to the outermost layer (softmax layer), which is in charge of producing the probability values of each class and replacing that last layer with a De-conv layer which is in charge of reconstructing the original image with classified pixels values. These two tasks are certainly different, but both still require the same basic knowledge, which is distinguishing two or more objects in the image.

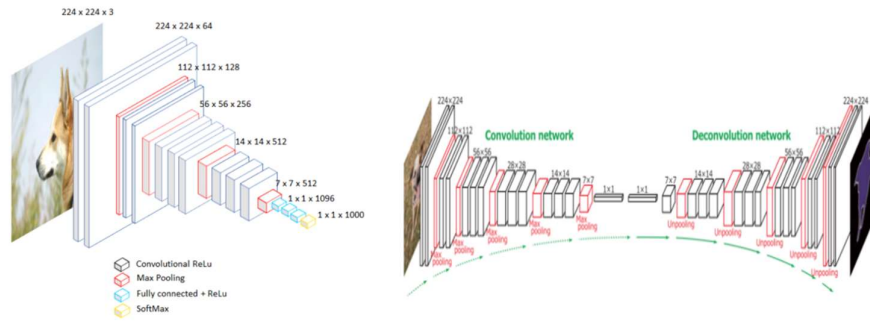


Figure II.6 Transfer learning in VGG-16 for semantic segmentation [13, 27]

Fine-tuning is a method derived from transfer learning where the difference between these two methods lies in how to maintain the initial layer. In transfer learning, all initial layers that are frozen, while in Fine-tuning all initial layers will still be trained but at a much slower learning rate than the outer layers.

II.6 HSV Color Space

There are several formats for saving images, two of which are RGB (Red Green Blue) and HSV (Hue Saturation Value). RGB images are images that consist of three-color channels, red, green, and blue. This RGB format is also the most

common format when it comes to displaying images. On the other hand, an HSV image consists of three components which are Hue, Saturation, and Value. Different from RGB, the HSV system is more similar to how humans interpret colors [28, 29]. Hue is the component that represents color based on the angle value with a range of 0-360°. Red values fall in 0-60°, green in 121-180°, and blue in 241-300°. Furthermore, Saturation is the component that represents the amount percentages of gray color in the image. Lastly, Value is the component that represents the brightness of the image in percent where the value 0 means black and 100 means the entire original color. An illustration of the comparison of RGB and HSV can be seen in Figure II.7. While doing motion capture, we usually will set the hue to around 50. This could be changed based on the condition of the lighting.

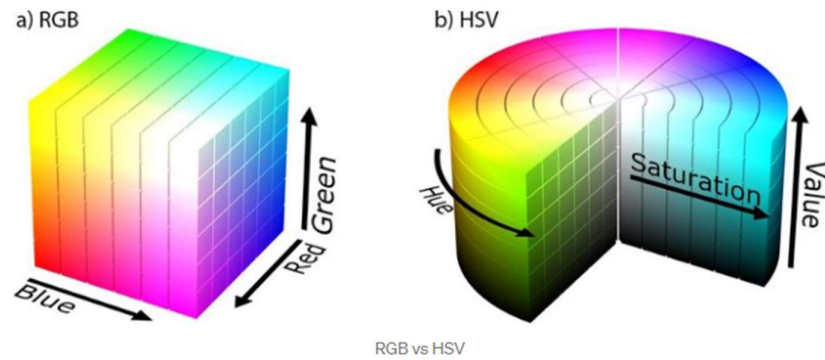


Figure II.7 RGB vs HSV color space [28]

II.7 Image Inpainting

Image inpainting is a problem of filling a missing part of an image realistically. The solution of this problem has developed in years from using Navier-stokes [30], Fast Marching Method (OpenCV) [31], GAN-based inpainting [32], Shift-Net [33], Partial-Convolution [34], to the newest method Fourier Convolutions Large Mask inpainting [35]. With Fourier Convolution, it improves both perceptual quality and parameter efficiency of the network. This method was developed by Samsung AI Center.

To use this method, an original image and a mask image should be provided. Next by selecting a suitable model, a brand-new image would be generated. This can be seen in Figure II.8.



Figure II.8 LaMa Inpainting [36]

II.8 COCO Dataset Format

COCO (Common Object in Context) is a well-known dataset format for object detection, segmentation, and caption generation. It stores all the information regarding the dataset like object classes, bounding boxes, and key points in JSON (JavaScript Object Notation) format. The structure of COCO dataset contains info, licenses, category, images, and annotations [37].

II.9 FFmpeg

FFMPEG is an open-source software for handling multimedia files. It can be used for decoding, encoding, transcoding, muxing, demuxing, stream, and even filter pretty much everything [38]. This software will be used in this research for compressing and removing audio from GoPro video. Compressing is needed for reducing the video size from almost 1 GB to only less than 100 MB. Removing audio from GoPro video is also needed so that OpenCV can open the video.

II.10 Git

Git is a version controlling tools which helped developers to create a group project or maintaining a long term project. The repository to store all the project files was usually GitHub or GitLab. The Git can be accessed by using the command prompt or using the GitHub interface via browser. The terms usually used in Git were:

1. Clone, the action of duplicating the project from remote repository to local repository. This action basically downloads all the data and apply the version control on the local computer.
2. Add, the action of adding the files to be staged.
3. Commit, the action of staging all changed files. This also usually comes with messages which describe the summary of the changes made to the project.
4. Pull, the action of updating the local repository to the latest version of remote repository
5. Push, the action of updating the remote repository by uploading and implementing changes that has been staged in local repository.
6. Branch, a new or separate version of the main repository. This can be used so that temporary changes will not affect the main repository at all.
7. Checkout, the action of switching between branch while restoring the state of the repository in local.

Chapter III Data Acquisition and Processing

In this chapter, the data acquisition pipeline and processing will be discussed. In general, the data acquisition can be divided to setting up experiment, gathering subjects, marker placement and motion capture using HSV method. The data processing pipeline can also be divided into removing audio from the video, marker tracking, image inpainting and creating dataset.

III.1 Data Acquisition

This process starts from setting up all the required equipment which are:

1. Treadmill,
2. 4 GoPro Hero 8,
3. Markers for HSV
4. 2 softboxes,
5. speedlite,
6. Black Legging

In this experiment a treadmill was used so that the subject will always be inside the camera view range. In this case numerous frames with the subject inside can be captured without redoing the video recording. Next, 4 GoPro Hero 8 was used to capture both sides of the subject. Each two of the cameras will be responsible for stereo vision which can result a pair of frames that will be able to be reconstructed. Then, the markers used in this experiment was design to have green color so that it can be easily segmented from the environment color and was created using 3D printing as shown in Figure III.1. There were three types of these markers. First is the normal ones which just consist of the half-globe shape. The second type was the one with black square aside of the half-globe. These should be placed on hip and metatarsal to prevent the camera seeing another side of the marker. Last, the third type were the ones with additional rod to support the half-globe. These should be placed on mid-thigh and mid-calf. The markers were attached to the subject by using double tape and an addition of a medical band was also needed for the mid thigh and calf so that it dampened the vibration of the long markers.



Figure III.1 Marker variant created for HSV motion capture system

In addition, 2 softboxes were added to lighten up the markers that are covered by shadow because of the poor lighting condition in the laboratory. Softboxes are an equipment that consists of lightbulbs and a cloth for diffusing the light. This equipment is often used in photostudio to produce light exposure of light to the subject. Next, speedlite was used to indicate the starting frame on each of the camera so that the frame taken will be from the same time. Lastly, the subject would always wear black legging to minimize the relative motion from the markers. The final set up can be seen in Figure III.2.



Figure III.2 Final Setup

After the set up was complete, 50 subjects with different class of BMI were gathered. The chart shown in Figure III.3 below show the distribution of the subject to the BMI classes that was included in the trainset. The variation of the BMI is

needed so that the model would have been trained in various visualization of the subject. This would make the model robust to different visualization of the subject as the model would be deployed to make inference on numerous subject later on. The overall protocol on the data acquisition were as follows.

1. Subject will be given a consent form
2. Subject will measure the body weight and height
3. Subject will change to black legging and all the markers will be placed
4. Subject will get the anthropometry data measured
5. Subject will try to walk so that the gait will be as natural as possible during recording
6. Subject will also make himself or herself comfortable by walking on the treadmill for a few minutes
7. Treadmill speed will be adjusted to the preferred speed
8. Recording starts
9. Speedlite flashed indicates the beginning of the data acquisition
10. Subject will walk for around 1 minutes
11. Speedlite flashed indicates the end of the data acquisition
12. Recording stop and the treadmill will be turned off
13. Subject will remove the legging and all the markers
14. Subject will get voucher as gift

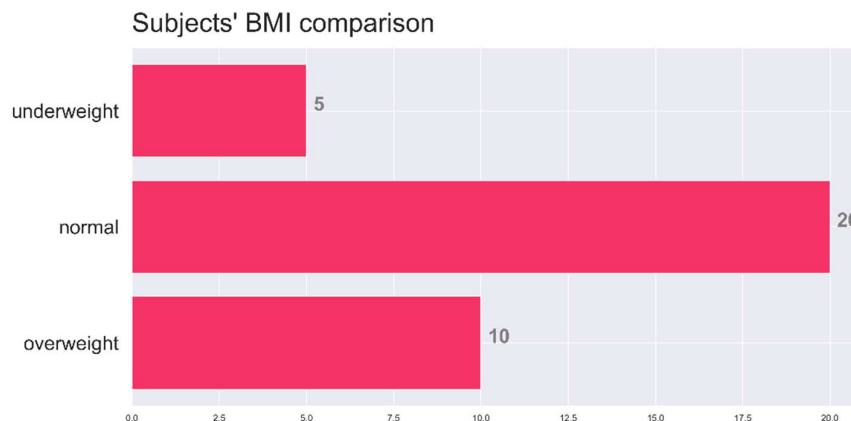


Figure III.3 Subjects' BMI distribution

III.2 Data Processing

After conducting motion capture from all subjects, the data processing will start from removing all the audio from all the videos using FFMPEG. This should be done because OpenCV library doesn't support the output video format from GoPro. The command for removing the audio would look like this:

```
ffmpeg -i video.mp4 -crf 23 "video_compressed.mp4"
```

The command should be executed through command prompt or windows powershell. Another option is just using the batch file provided in the project repository named *compress_video.bat*. This will automatically remove the audio from all videos in the exact same folder as in the bat file.

Next marker tracking will be done using the HSV color segmentation method. This was done using the OpenCV library from python. The result of the marker tracking process was the position of each marker inside the frame which can be seen in Figure III.4. The most important part in this step was selecting the best H, S, and V parameters for segmenting the color. If the parameters chosen were not suitable, then the result would have some missing markers or noises. This would become more bothersome in data processing which is why eliminating this factor in the beginning by inspecting the result video would be best. Based on experience the lighting in the data acquisition room was not ideal as the floor color was a little bit greenish. The main lights were also not bright enough that two softboxes are still needed. Because of these issues, noises appeared frequently in the middle of the video. The noises here are the dark green color which parameters sometimes collide with the parameters used for extracting the light green color. Therefore, in further research, it is recommended to move to a better room with more decent light condition. To get experience on exploring the effect of HSV parameters, a python notebook named *Find_HSV.ipynb* has been created. From the notebook, by specifying the input video, user can adjust the HSV parameter by moving the slider bar and observe the changes on the video live. To segment and track the markers the user needs to run a different python notebook named the *HSV_Mocap.ipynb*. This notebook consists of the tracking function till the post-processing function which was interpolating the missing data due to the swinging motion of the subject

hand. On some cases the user would also like to check whether the numbering of the markers is matching with all of the 4 videos from each camera. This checking should be done because the numbering on which markers should be numbered as 1 or 2 would depend on which is the lowest at the moment the data was recorded. Therefore it was recommended to run all the notebook all the way to the end as it would automatically check the data according to the angle. Keep in mind that the video name should follow the naming rule which are “Sxx_yy.mp4” as xx is the number of subject and yy is the angle of the camera as listed below:

1. BR for back right,
2. BL for back left,
3. FR for front right, and
4. FL for front left

Other alternative was by using the GUI made by the author as it would have automatically track, interpolate missing data, and check all the numbering. Therefore, the user wouldn't need to be bothered to create every environment and no coding experience would also be needed.



Figure III.4 HSV motion capture result

The main process of the HSV motion capture can be divided to:

1. Breaking down the video to frames or images using the `cv2.VideoCapture` function
2. Converting each of the frames to HSV color format by using the `cv2.cvtColor` function
3. Segmenting the colors using the HSV ranges specified using `cv2.inRange` and `cv2.bitwise_and`
4. Converting to binary images
5. To get a better result some morphological function like `cv2.opening` and `cv2.closing` was used to cover holes in extracted blob and reducing noises
6. Find contours by using the `cv2.findContours` as well as `imutils.grab_contours`
7. Calculating the centroid
8. Exporting the collected raw data to csv format
9. Create a new video displaying the data live on the original video.
10. Process the data by interpolating the missing markers position and trimming the data with the brightness average peak value.
11. Ensure the numbering of the markers by conducting some tests.
12. Export the processed data to a csv file
13. Create a new video with the processed data displayed
14. Compress all the video using the FFmpeg

Next, after finish tracking and checking all the videos, there would be images and the corresponding csv file that stores all the markers' position. Based on the marker position, all frames were inpainted to remove all the markers shown in the frame. This was be done by using the LaMa inpainting method. But to get the frames inpainted, mask image should be provided. These were created by creating binary images that consists of circles which centers are based on the markers' positions. The process can be seen in Figure III.5. All the inpainting were done by using local machine with GPU acceleration. One thing to note here was the process of inpainting requires large computational resources especially GPU memory and speed. While the author is using personal laptop for doing the inpainting process, it

is observed that all frames for a single subject will take around 20 hours to finish. The GPU used at that time was Nvidia RTX 3060 with 6GB VRAM. Another thing was because inpainting process took hours, it is recommended to give additional cooling system to the machine or just move to machine to a room provided with an Air Conditioner because the temperature of the GPU can rise to 90-degree Celcius.

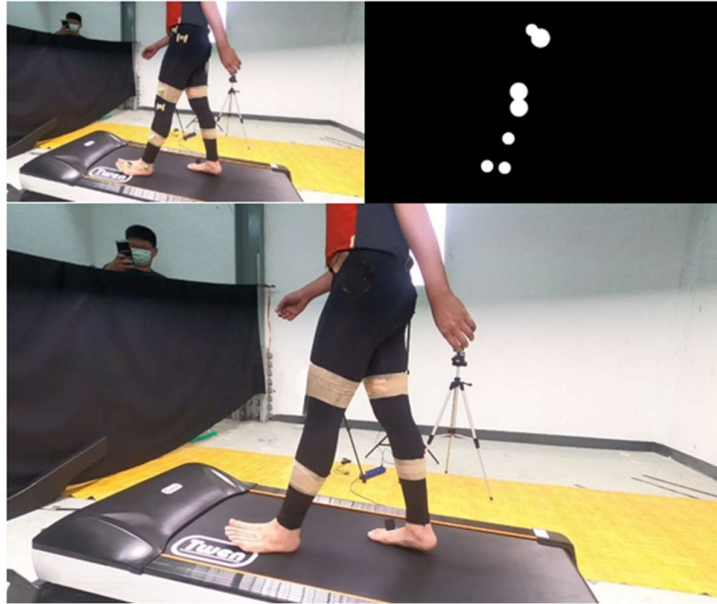


Figure III.5 Image inpainting process

While inpainting the images needed GPU access for the program to work, some drivers should be installed which are:

1. Nvidia driver, this can be checked by running `nvidia-smi` command in the command prompt. If it doesn't show the status of the GPU, then the driver should be downloaded and install first from the nvidia official website
2. Nvidia Cuda Toolkit, the software needed for GPU computing. Can also be downloaded from the Nvidia official website. The version of cuda toolkit should also suit the machine's GPU because some GPU doesn't support high cuda computability. Therefore, the nvidia toolkit version shouldn't be higher than what it was supported.

It was highly recommended to conduct the inpainting in linux operating system as the original code was made for linux and would be faster in linux as well, but for the Biomechanics user sake, the code has been modified for Windows user to used as well. The code can be found at the project repository with the name *inpainting.py*. The user just needs to install all the requirements and the python environment as well as changing the path specified in the python file to the exact path associated in the users drive. For the guides to install the environment, check out the LaMa official repository.

Finally, after all frames were inpainted, training and validation dataset will be created in the format of COCO dataset. First, all the inpainted image should be placed in a single folder. But unfortunately, because there were limit on how many items a folder can hold, the frames then were downsampled every 30 frames. Then the artifact like bounding box and keypoints are stored in JSON format as shown Figure III.6.

```
{
  "info": {
    "description": "Lower body image Dataset on human walking motion",
    "url": "https://pm.ftmd.itb.ac.id/id/biomekanik/",
    "version": "1.0",
    "year": 2022,
    "contributor": "Lab Biomekanika ITB",
    "date_created": "2022"
  },
  "licenses": [],
  "images": [
    {
      "license": "",
      "file_name": "S28_BL_257.jpg",
      "coco_url": "",
      "height": 1080,
      "width": 1920,
      "date_captured": "2022-08-16 18:38:14",
      "flickr_url": "",
      "id": "282257"
    },
    {
      "license": "",
      "file_name": "S28_BL_357.jpg",
      "coco_url": "",
      "height": 1080,
      "width": 1920,
      "date_captured": "2022-08-16 18:40:40",
      "flickr_url": "",
      "id": "282357"
    },
    {
      "license": ""
```

Figure III.6 COCO Dataset Format

Chapter IV MODEL TRAINING AND EVALUATION

In this chapter, the model training, evaluation, and analysis will be discussed. The model training section will cover from the equipment, parameter, and training procedure. The model evaluation section will cover the model evaluation procedure and all the decision taken during evaluation.

IV.1 Model Training

The model will be trained with local machine using the modified training script from the official Lightweight OpenPose Github. Modification should be done because of some differences between this research and the lightweight OpenPose, where in the lightweight OpenPose the position predicted are the joint position but in this research the position is the markers position. This makes the total number of predicted points as well as the PAFs differ. Moreover, the visualization and evaluating script should also be modified. All the scripts can be accessed in the GitHub link provided in Chapter 5.1. Also, there was a guide too for custom training provided in the Lightweight OpenPose Github. Hyperparameters such as learning rates, batch size, learning rate decay rate, and number of refinement stages can be set in the training script as configuration. In this experiment, the author is using the exact same parameters used in the original Lightweight OpenPose except for the batch size as there are limitation of the hardware.

Before the data was fed into the model, an additional process that was usually called as data augmentation was added. Through this process, the data would be augmented via rotation and cropping so that more variation of data could be provided to the model for training. The machine used during training are equipped with:

1. Intel I9 12th Gen
2. Nvidia A4000 with 16GB of VRAM
3. 64 GB RAM

Using this machine, the training batch size can be set to 16 which will occupy all the GPU Memory. The training will also be divided into two stages. In the first stage, training will be done from the pre-trained model Mobilenet. In the second stage, the refinement stage which was to increase the model accuracy. The training was completed in around 24 hours using Pytorch framework. The model training history was logged to a text file and then extracted to plot the model loss history which can be seen in Figure IV.1. This can be done by finding patterns in the log file and do some string processing to slice all the information about the loss or accuracy. Then this information was converted into pandas dataframe before it was plotted using matplotlib and seaborn library. In the training history we can observed that indeed the trend was going down and the training has stopped before overfitting. Overfitting phenomena usually can be seen if the validation loss trend of the model goes up as the iteration increased. This phenomenon should be avoided since it would make our model cannot predict well in general dataset. After confirming by inspecting the model history, the model then can be staged to the next step which was model evaluation.



Figure IV.1 Training loss history extracted from model training log

IV.2 Model Evaluation

At first, the model evaluation was done by observing the model performance in predicting the validation dataset. This can be done while training or even after training by observing the evaluation log. After the first training, the model accuracy across all the markers can reach to the average of 5.5-pixel error (distance of the predicted and groundtruth position). This result was considered very good that almost all the marker position match with the ground truth. But very good result usually indicates poor performance in predicting unseen data. Therefore, it was concluded that further testing of the model is needed before staging the model to deployment or production. This can be seen in Figure IV.2.

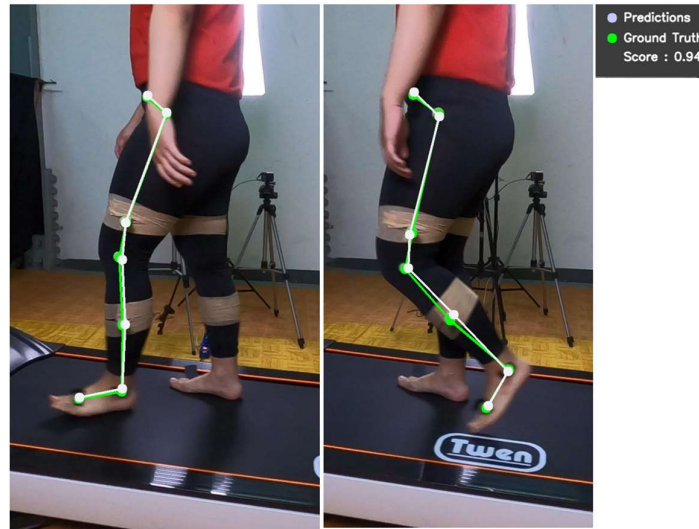


Figure IV.2 First model result

To finalize the model, a brand-new video without equipping any markers was being fed to the model, but unfortunately the model fails to predict the marker position as in the test data as shown in Figure IV.3. All the predicted marker position were (0,0) as there were dots on the top left side of the figure. This was caused by the features or defect created during image inpainting and misled the model into searching for those certain features. Although the features cannot be seen through human eyes, the story was completely different for machines which sees a picture as numbers or pixels value. The author believed that there were some patterns created by the inpainting process which were discovered by the machine and when

the model was being fed by a brand new images which hasn't been going into the inpainting process, there were no such pattern and the model finally return the (0,0) result.

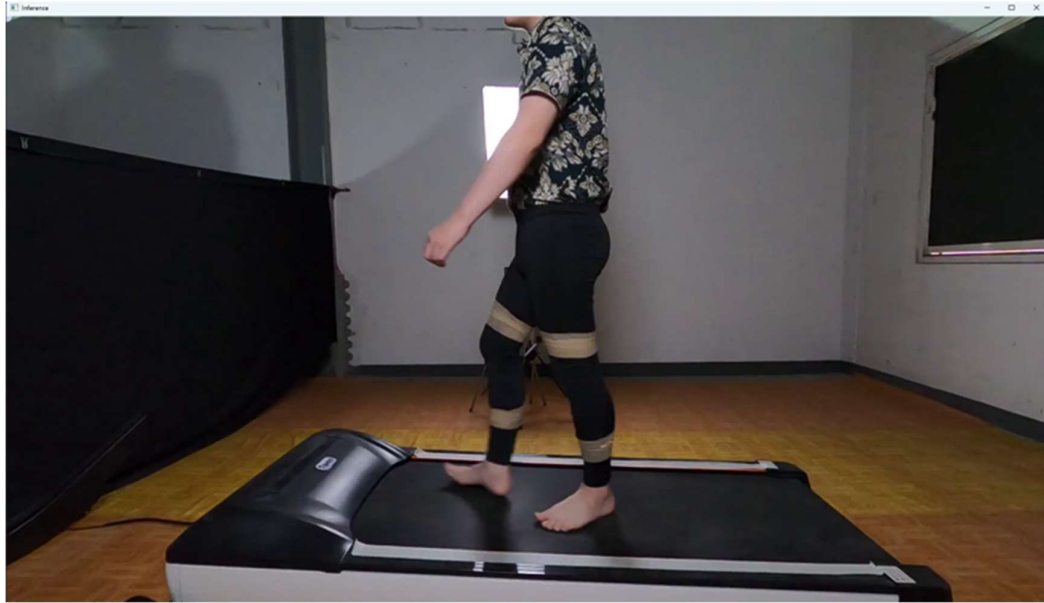


Figure IV.3 First model error on predicting video without markers

To deal with this problem, a new preprocessing step was done before feeding the image into the model during training. The preprocessing added was blurring the image using median blur filter from OpenCV library and converting the image to grayscale image. This filter was chosen because it can eliminate the defects (artifacts) on the image after the inpainting. The filter effect can be seen in Figure IV.4. Lastly the training was repeated using the exact same machine and procedure and resulting in average of 17-pixel error. Although this accuracy is far from the first training result, the performance of this model in predicting a new video with no markers was acceptable which can be seen in Figure IV. 5. Although by adding the median blur solved the issues of fail on predicting the position because of defect in the images after inpainting, the computational load for inference goes up to because implementing the median blur on the images also took times. In further research, it would be best if the preprocessing part can be omitted if there is better inpainting method which doesn't create defects on the images. Some alternatives

that can be tested are the stable diffusion models which happened to be quite popular nowadays in text to image generation task.

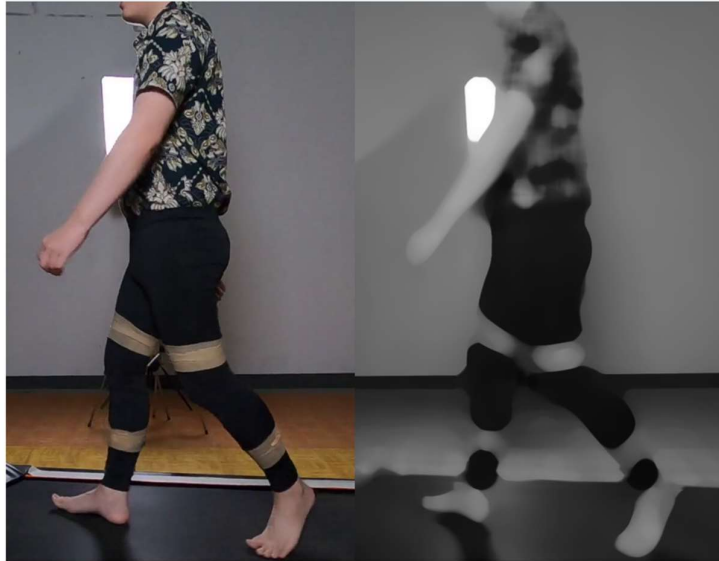


Figure IV.4 Median blur effect



Figure IV. 5 First and second model evaluation comparison

There are two ways to improve the accuracy of this model. First is to refine the HSV motion capture method so that the error from data collection can be minimize. Second is to increase the batch size in the training parameter which will requires more GPU memory than the currently available in the Biomechanics Laboratory. This should be done because using larger batch size means taking more data during learning which will give us a result closer to the global minima. But there was a drawback in increasing the batch size which was computational resources and poor generalization. Referring to the official Lightweight OpenPose, the model was train by using batch size of 80 which is far more than the one used in this experiment. Using the reference above, it would be safe to assume that the batch size parameter can still be increased to 80. This can be done in future research if the computational resource is available.

For future used a python notebook was also created if users needed to do inferenced on GPU because the application build was based on CPU. The python notebook can be run by using a text editor like the Visual Studio Code (VSCode) or Jupyter Notebook. The notebook is an interactive python script which can be run cell by cell while each cell can contain several lines of code. There will also be helper functions which are the functions to load the model checkpoint, inference function, visualization function, and filtering function to reduce the noise made by the model. Running the notebook will be as simple as creating a python environment and executing cell by cell by pressing the shift+enter button on keyboard. The python environment itself can be build by using built-in python program or using third party software like anaconda. This environment was made so that the libraries installed for the project would be isolated and not affected by any other projects in the machine. The libraries which should be installed in the library can be installed by running the `pip install -r requirements.txt` command in the command prompt. One thing to note was because the model would be loaded on GPU, the kernel of the notebook should be restarted if the users want to load to another model. This would prevent the program errors when the GPU memory needed exceeds what was available in the machine.

Chapter V APPLICATION DEVELOPMENT

In this chapter, the application created to deploy the model and to conduct HSV motion capture will be discussed. This section will cover from application design, a guide to install this application pretty much on any machine and any OS (Operating System), and lastly how to use the application itself.

V.1 Application Design

The application created to conduct either the marker-based HSV motion capture and markerless motion capture is a web-based application powered by Streamlit which was an open-source library. Streamlit is a tool for developer to easily create a modern-looking python-based webapp without having to learn Front-end development which covers HTML (HyperText Markup Language), CSS (Cascading Style Sheets), and JavaScript.

To give a clear idea on the project, the project structure can be seen in Figure V.1. There are three folders inside the project directory which each of its stores data as follows:

1. /figure : Stores all the graph created while using the application
2. /mocap : Consists of three more directories
 - a. /Data : Stores all the raw and processed data
 - b. /VideoRaw : Stores the original video that will be tracked inside the app
 - c. /VideoTracked : Stores all the tracked video to visualize the tracking result.
3. /src : Stores all the source code for the application. Not that this will only be used for development and not for deployment.

All folders would be created automatically when cloning the project from GitHub, so the user would not need to worry about creating or initializing the project. Later, all the users need to do is just cloning the project, install the application and use the web application. Another thing that could be done are creating a VM (Virtual Machine) or just deploying this application on one of the computers in Biomechanics laboratory and the user can accessed the web application through the

API (Application Programming Interface) endpoint which is <http://x.x.x.x:8501>. Although for doing that we need to set a static IP (Internet Protocol) for the computer connection and the user must connect through the OpenVPN provided by ITB.

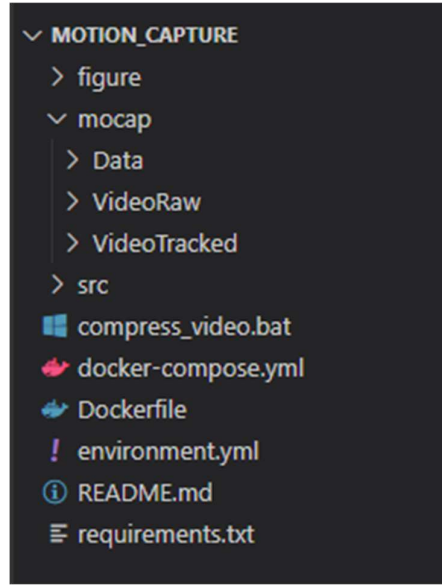


Figure V.1 Project structure

The file named `app.py` was the file which contains the Streamlit main program. This is the program that will act as the front end of the project. Next, all the function beside the `app.py` was the helper function needed for the back end of our project. This is the function to conduct HSV segmentation, saving videos, graphs, and inferring from our trained model. Lastly, to make it easier for user to install the application, the webapp was deployed using Docker. All the code was stored in the GitHub as the version control service. The GitHub can be accessed via this [link](#) (credential required).

V.2 Installation Guide

There are several steps that should be followed to install the application. First Docker should be installed in the target machine and using these steps for Windows.

1. Open Windows Powershell and run “`wsl --install`” command.

2. If the help text is seen, then just proceed to install the Linux distro by running “wsl –install -d Ubuntu-20.04” and follow the user creation process.
3. Then upgrade the wsl version to wsl2 by running “wsl –set-version Ubuntu 20.04 2”. If an error occurred then it is most likely because a kernel update package needs to be installed. Just go to this [link](#) to download the kernel update package.
4. Next, install Docker Desktop and make sure that it is running using WSL2 backend.

All the steps above were the prerequisite and only need to be done once before installing any software or application that was deployed using Docker.

Next follow these steps to install the application for the project:

1. Clone the project by running “git clone link”
2. Simply build the docker image and run the container by executing “docker-compose up -d --build” command in the command prompt inside the project folder.

If the application was successfully created, then the container can be seen in the Docker Desktop application as shown in Figure V.2.

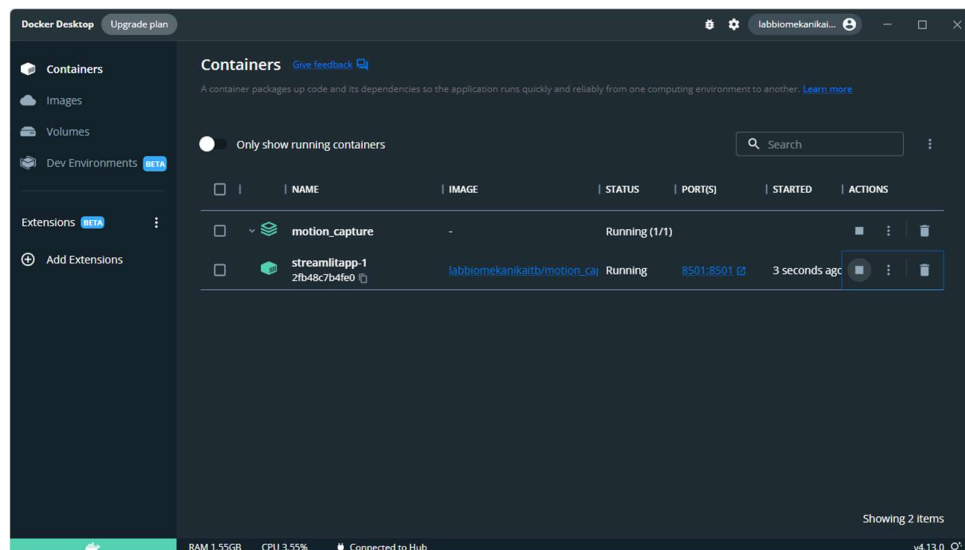


Figure V.2 Docker desktop User Interface

V.3 User guide

To run the application, simply click the play button in the docker desktop container if the container isn't currently running. Next, to show the webapp click the port to open a browser which will directly go to our webapp address (localhost:8501). This can be done alternatively by just entering the address inside the browser after the container has been up. The process can be seen in Figure V.3.

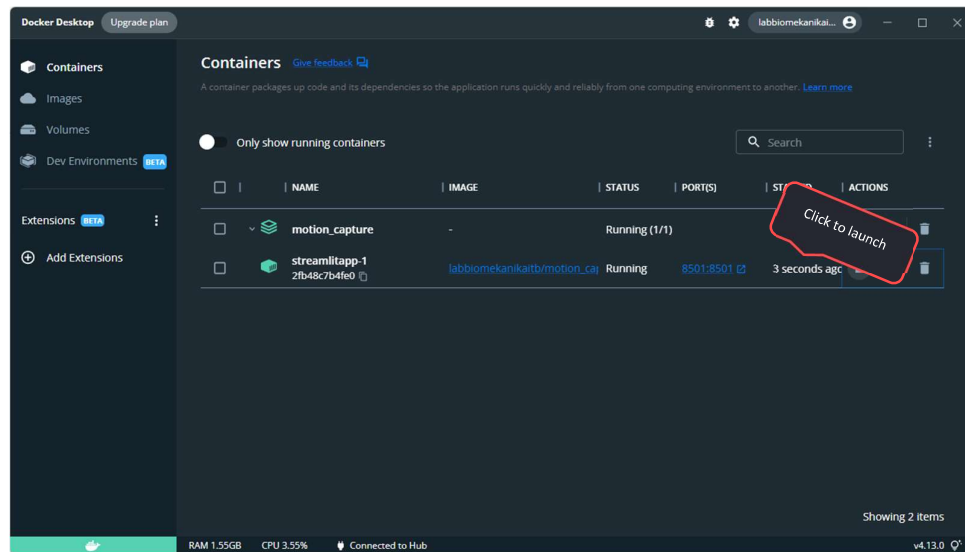


Figure V.3 Launching the web app

After the application shows up, it requires credentials before entering the application. Just input **admin** as the username and **admin** also for the password. These credentials can be change by adding other strings in the streamlit authentication program. But before deploying or publishing the application it would be safe to not explicitly show the password inside the program by using the encrypted password instead. After that the application will shows as presented in Figure V.4. In the application, there is main menu that consists of four options which are Tracking, Data, Calibration, and Reconstruction. Although the Calibration and Reconstruction still cannot be accessed, it is there to make it easier for further development.

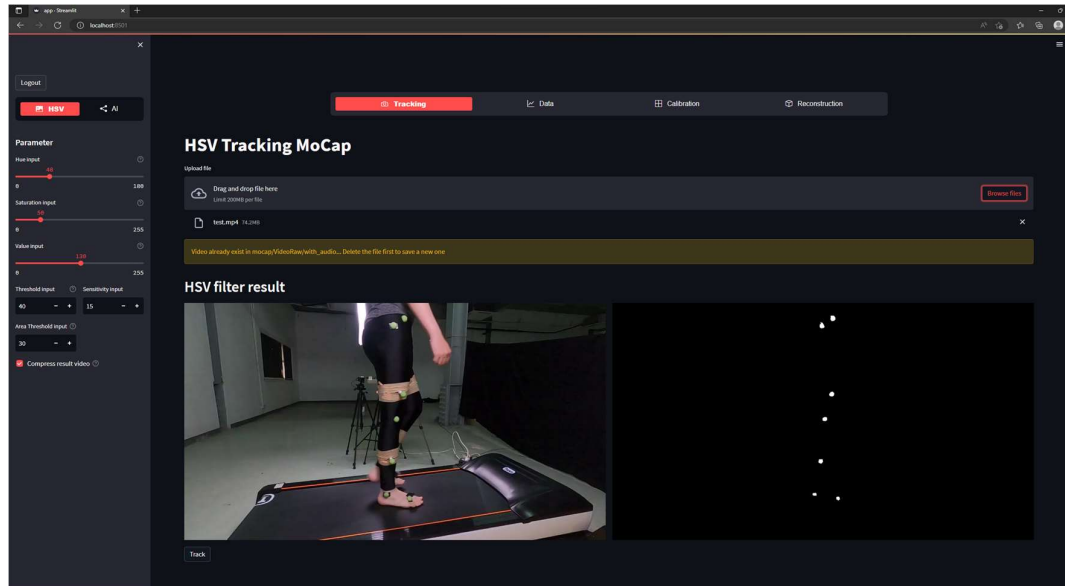


Figure V.4 Web app User Interface

Now, go to the Track menu and there will be two more options available in the sidebar. The first option HSV is for conducting the HSV marker-based motion capture. To track a video using this HSV color tracking method, just need to upload a video file which should be first compressed by running `compress_video.bat` available after cloning the project before. Just put the video beside the program and run the program it will automatically compress the video without any significant quality loss (FFMPEG required). After successfully uploading the file, the application will show the color segmentation result on the first frame of the video using the parameter specified in the sidebar. Play around the parameter to get the suitable color segmentation result. Lastly, press the Track button to run the tracking on the entire video. This will also trim the data according to the flashlight that was signaled during video recording. After the processed is complete you should be able to find all the result in the mocap directory inside the project directory.

The second option AI (Artificial Intelligence) is for conducting markerless motion capture. Same as previous, a video will be uploaded and then the model that has been trained will infer or predict all the marker position. Although the result would be quite noisy, as it can be seen in the result video, it can be filtered using the next main menu which is Data menu. HSV and AI menu can be seen in Figure V.5.

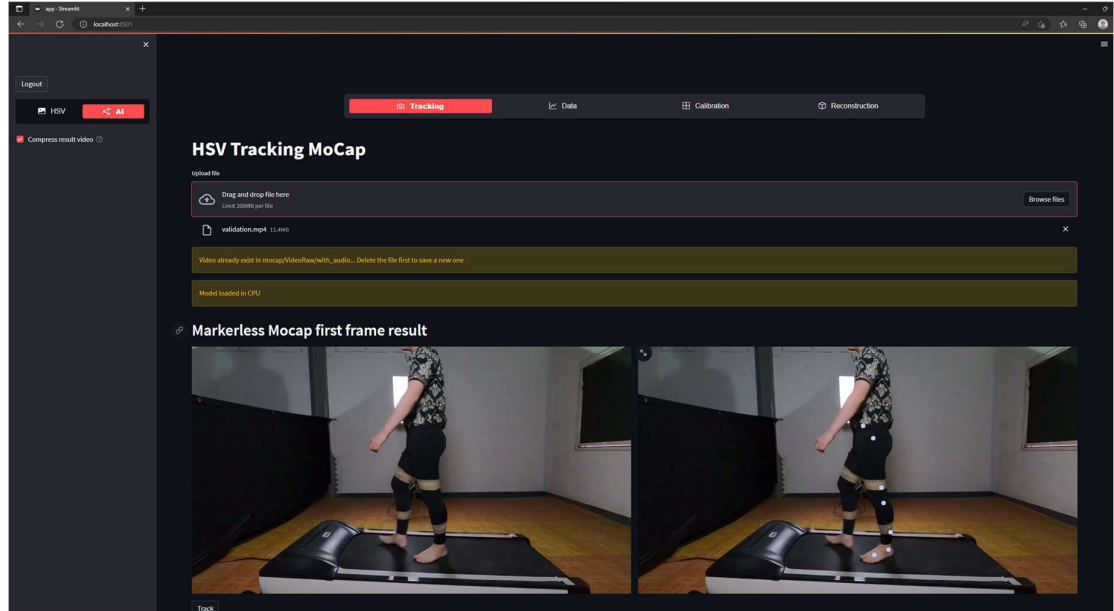
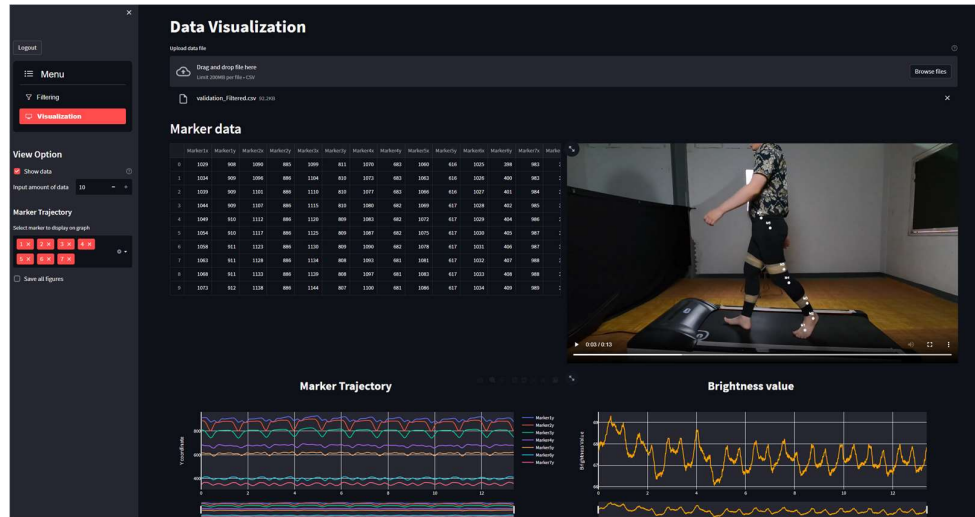


Figure V.5 Markerless motion capture User Interface

Inside the Data menu we can conduct filtering to denoise the result data by uploading the csv created inside the /mocap/Data/Processed directory. Just play around the parameter and hit the Filter button and it will show the result along with the new tracked video. Lastly, go to the Visualization tab in the sidebar and by uploading the filtered csv file, the app will visualize the marker trajectories along with a table which display the head of the data and the result video. This can be seen in Figure V.6. Some things like fullscreen mode, panning in or out the data could be done to the chart as the chart was made using plotly which are interactive charts that were displayed on HTML.

There are also some options like compress the output video which should be checked so that the output video will be only around 100MB instead of several GBs. Another option is to save all the figure displayed by checking the save figures option. This would automatically save all the figures inside the project folder. Note that users can also download any files displayed in the app manually by clicking the download button around the widget. All files that was required would be downloaded via browser where the app is currently up.



What more to improve in the app are implementing the calibration and reconstruction in the menu to get the 3D joint position data and improve error message by adding the error function at any places needed.

V.4 Development guide

Before making changes to the source code, usually it was best to create a new branch from the master branch and start making the changes from there. This will prevent any errors made in the future affecting the current deployed version of the project. In the DevOps it is usually called by version control. The steps that should be made are as follows:

1. Creating a development branch
2. Make changes
3. Ensure that the application works fine by rebuilding the app
4. Adding the changed files to staging phase
5. Stage all the changed files by committing the changes
6. Push to the remote repository
7. Push new the docker images to docker hub
8. Merge the master and development branch if needed

CHAPTER VI SUMMARY AND FURTHER RESEARCH

After conducting analysis on the research, a few summaries can be concluded and there were some recommendations for the further research.

VI.1 Summary

From this research, there are 3 points that can be concluded which are:

1. There are fifty motion capture data collected using HSV motion capture method
2. A decent model has been created using Transfer Learning technique based on the OpenPose pipeline.
3. An early webapp has been created for conducting HSV motion capture and markerless motion capture.

VI.2 Further Research

Based on the results of this work, following are several recommendations for further research:

1. Retraining the model using more accurate motion capture data by refining the HSV motion capture method.
2. Changing the inpainting method to better methods like stable diffusion so that there would be no need of additional process before the images are fed into the model.
3. Retraining the model using larger batch size parameter if the computational resource is available.
4. Combining this research with the 3d reconstruction algorithm that has been developed by ITB Biomechanics Laboratory.
5. Improving the webapp by adding the calibration and reconstruction support and improving the error handling by providing error message.

REFERENCES

- Alvin, "Prediksi Gaya Reaksi Tanah 3D Dari Gerakan Berlari Berdasarkan Parameter Kinematik Menggunakan Metode LSTM," 2021.
- S. Mahradi, T. Dirgantara, and A. I. Mahyuddin, "Development of an optical motion-capture system for 3D gait analysis," in *2011 2nd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering*, 2011, pp. 391-394: IEEE.
- A. P. Khariza, "Pengembangan Sistem Optical Motion Capture Tanpa Penanda untuk Analisis Gerakan Berlari."
- M. Andriluka, S. Roth, and B. Schiele, "Monocular 3d pose estimation and tracking by detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 623-630: Ieee.
- Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1212-1221.
- E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepcrut: A deeper, stronger, and faster multi-person pose estimation model," in *European conference on computer vision*, 2016, pp. 34-50: Springer.
- Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291-7299.
- E. D'Antonio, J. Taborri, E. Palermo, S. Rossi, and F. Patane, "A markerless system for gait analysis based on OpenPose library," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2020, pp. 1-6: IEEE.
- N. Nakano *et al.*, "Evaluation of 3D markerless motion capture accuracy using OpenPose with multiple video cameras," vol. 2, p. 50, 2020.
- V. Zhou. (2019). *CNNs, Part 1: An Introduction to Convolutional Neural Networks*. Available: <https://victorzhou.com/blog/intro-to-cnns-part-1/>
- V. Zhou. (2019). *CNNs, Part 2: Training a Convolutional Neural Network*. Available: <https://victorzhou.com/blog/intro-to-cnns-part-2/>
- Callyberz, "Sobel Filter using C++," 2017.
- H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520-1528.
- C. Nwankpa, W. Ijomah, A. Gachagan, and S. J. a. p. a. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.
- S. Sharma. *Activation Functions in Neural Networks*. Available: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
- N. Akhtar, U. J. N. c. Ragavendran, and applications, "Interpretation of intelligence in CNN-pooling processes: a methodological survey," vol. 32, no. 3, pp. 879-898, 2020.

- Savyakhosla. *Introduction to Pooling Layer*. Available: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>
- Z. Cao, G. H. Martinez, T. Simon, S. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*," 2019.
- CMU-Perceptual-Computing-Lab. *OpenPose*. Available: <https://github.com/CMU-Perceptual-Computing-Lab/openpose>
- K. Simonyan and A. J. a. p. a. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.
- J. Hui. (2018). *mAP (mean Average Precision) for Object Detection*. Available: <https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173>
- D. M. J. a. p. a. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020.
- D. J. a. p. a. Osokin, "Real-time 2d multi-person pose estimation on cpu: Lightweight openpose," 2018.
- S. J. I. J. o. S. Tammina and R. Publications, "Transfer learning using vgg-16 with deep convolutional neural network for classifying images," vol. 9, no. 10, pp. 143-150, 2019.
- A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.
- F. Zhuang *et al.*, "A comprehensive survey on transfer learning," vol. 109, no. 1, pp. 43-76, 2020.
- V. Roman. (2020). *CNN Transfer Learning & Fine Tuning*. Available: <https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2>
- J. A. Abenza. (2018). *HSV color model*. Available: <https://medium.com/neurosapiens/segmentation-and-classification-with-hsv-8f2406c62b39>
- J. H. Bear, "The HSV Color Model in Graphic Design," 2020.
- M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, vol. 1, pp. I-I: IEEE.
- A. J. J. o. g. t. Telea, "An image inpainting technique based on the fast marching method," vol. 9, no. 1, pp. 23-34, 2004.
- Y. Chen *et al.*, "Research on image inpainting algorithm of improved GAN based on two-discriminations networks," vol. 51, no. 6, pp. 3460-3474, 2021.
- Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 1-17.
- G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 85-100.
- R. Suvorov *et al.*, "Resolution-robust large mask inpainting with fourier convolutions," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2149-2159.

- s. mdal. *LaMa: Resolution-robust Large Mask Inpainting with Fourier Convolutions*. Available: <https://github.com/saic-mdal/lama>
- T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014, pp. 740-755: Springer.
- F. Developers. *FFmpeg*. Available: <http://ffmpeg.org/>