JOIV : Int. J. Inform. Visualization, 9(2) - March 2025 699-707

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Bike Fitting System Based on Digital Image Processing on Road Bike

Tigor Hamonangan Nasution [a,*], Andreas Sitohang [a], Seniman [b], Soeharwinto [a]

[a] Department of Electrical Engineering, Universitas Sumatera Utara, Medan, Indonesia
[b] Department of Technology Information, Universitas Sumatera Utara, Medan, Indonesia
Corresponding author: *tigor.nasution@usu.ac.id

***Abstract*—This research aims to develop a bike fitting system based on digital image processing for road bikes. The method used in this study involves using the OpenCV and MediaPipe libraries in the Python programming language to detect the rider's body pose from a video stream captured using a webcam. The body pose data is then used to calculate important angles such as elbow, hip, knee, and ankle range related to the correct riding position for road bikes. In this research, a comparison is made between the body angles obtained and the angle range considered ideal for bike fitting on road bikes. If the body angles fall within the desired range, the system will label it as "Fit"; if the body angles are outside the selected range, the system will label it as "Not Fit." The results of this study indicate that the bike fitting system based on digital image processing using a webcam can provide helpful visual feedback in improving the rider's body position for road bikes. By observing the body angles produced and seeing the "Fit" or "Not Fit" label, cyclists can adjust their position to match the ideal position in bike fitting. The system test results show a low error rate, with elbow angle having an average error of 0.81%, hip angle of 1.37%, knee angle of 0.83%, and ankle range of 1.76%. Thus, this research contributes significantly to supporting cyclists in achieving a position appropriate to their inseam height.**

***Keywords*—Bike; bike fitting; MediaPipe; OpenCV; road bike.**

Manuscript received 8 Jun. 2024; revised 29 Aug. 2024; accepted 6 Mar. 2025. Date of publication 31 Mar. 2025.

I. INTRODUCTION

Bicycles are a means of transportation that is widely used throughout the world. Besides being used for sports activities, bicycles are also an environmentally friendly transportation alternative and can help reduce traffic jams in big cities [1], [2], [3]. They are increasing public interest in cycling as a recreation and a routine activity to maintain body health. However, many still make mistakes in determining the correct position when cycling. This can result in discomfort when cycling and even injury to the body [4], [5], [6].

The basic principle of balancing a bike fitting from a side perspective is determining the engine room's height and the cockpit's position so that the rider's tilt is optimal [7]. If this is wrong, the racer will lean too far forward or back, as seen in Fig. 1 [8]. The body position appropriate for the bike fitting based on the average measurements and range of the bike fitting can be seen in Fig. 2 [8]. One solution to overcome this problem is to use a bike fitting system, which can help choose the right bicycle size. The bike fit process adjusts the bicycle to the user's body to determine whether the body and bicycle position suit the user so that the user gets a comfortable position when cycling.

Fig. 1 Comparison of Body Positions in Bike Fitting

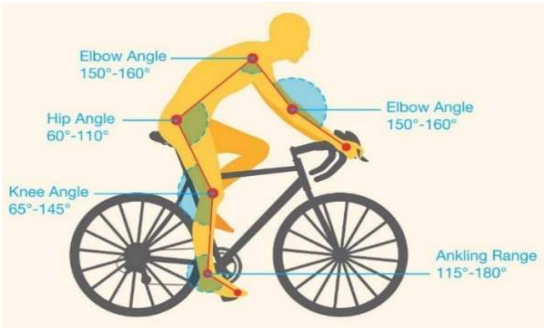

Fig. 2 Body position according to bike fitting

However, the bike fitting system requires special equipment and is relatively expensive [9]. In previous research, an appropriate bike system was also carried out using digital image processing. However, in this research, the bike-proper system still uses unique cameras and sensors in the digital image processing of the bike-appropriate system [10]. In this research, a bike fitting system is developed to make it more accessible and affordable by utilizing digital image processing. Using a webcam, this system took an image of the user on a road bike and then analyzed the image to measure the angle of several parameters in body position on the bicycle being used[8]. The results of this system are expected to help users increase comfort and effectiveness when cycling. A bike fitting system using digital image processing is a technology that can help improve efficiency and accuracy in the bike fit process. Hopefully, this research can contribute to developing digital image processing technology that can be applied in various fields and facilitate using bicycles as a more environmentally friendly and healthy transportation alternative.

II. Material and Methods

A. System Overview

This research generally requires a computer to run the bike fitting system using digital image processing. In this research, the system uses the Python 3.11.3 programming language using the OpenCV library in the video image processing process. OpenCV has been widely used in research related to Computer Vision in recent years [11], [12], [13]. The system block diagram can be seen in Fig. 3.

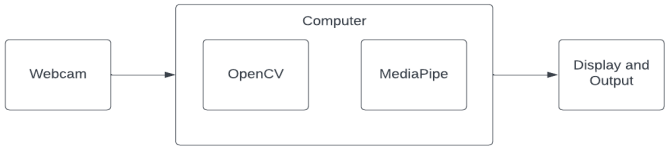

Fig. 3 System Block Diagram

In designing a bike fitting system using image processing on a road bike, several things need to be prepared, such as a webcam which is tasked with taking a video stream of the bicycle user when using the road bike, which the bicycle could later use a trainer to facilitate the data testing process, OpenCV as the library used to help, a webcam can be connected to a laptop and take a video stream and Mediapipe is tasked with detecting poses for bicycle users and assisting in measuring angles for bicycle users. Several libraries were used in system design, namely Open CV OpenCV (Open Source Computer Vision Library), a popular library for image and video processing [14], [15], [16].

In a bike fitting system based on digital image processing on road bikes, OpenCV is used to read and display video frames, change image color schemes, write text on images, save images, and draw landmark poses using functions such as cvtColor, imwrite, putText, imshow, and others. MediaPipe is a media processing library developed by Google [17], [18], [19]. MediaPipe provides tools and models for object detection and tracking in imagery and video. In a bike fitting system based on digital image processing on road bikes, MediaPipe is used to detect human poses in images using the mp_pose component, which provides a pose estimation model, and the mp_drawing component, which is used to draw pose landmarks[20].

B. System Design

The stages in designing a bike fitting system based on digital image processing on road bikes are divided into several stages. The system design begins with taking a video stream, which could later be processed on a computer with the help of several libraries, such as open CV and media pipe. These libraries were used to obtain angle values for several parts of the bicycle user, which follow predetermined measurement references. Before designing a bike fitting system based on digital image processing on road bikes, several Python libraries are needed to make it easier for researchers to call functions that support this image processing. The libraries needed are:

a. OpenCV (Open Source Computer Vision Library) is a popular image and video processing library [11], [12], [21]. In a bike fitting system based on digital image processing on road bikes, OpenCV is used to read and display video frames, change image color schemes, write text on images, save images, and draw landmark poses using functions such as *cvtColor*, *imwrite*, *putText*, *imshow*, and others.
b. MediaPipe is a media processing library developed by Google[18], [22], [23]. MediaPipe provides tools and models for object detection and tracking in imagery and video. In a bike fitting system based on digital image processing on road bikes, MediaPipe detects human poses in images using the mp pose component, which provides a pose estimation model, and the mp_drawing component, which is used to draw pose landmarks.
c. Math is a standard Python library for mathematical operations[24], [25]. In a bike fitting system based on digital image processing on road bikes, math is used to perform mathematical operations, such as calculating arctans and converting angles from radians to degrees.
d. NumPy is a popular library for numerical computing in Python[26], [27]. In a bike fitting system based on digital image processing on road bikes, NumPy performs mathematical operations on arrays, such as calculating angles and applying an average filter to angle data.

1) Video Stream Capture:

Video stream capture uses a camera connected to the IP Webcam system. Video capture is carried out by recording video of the cyclist while he is in a relevant position for angle measurements. The program code snippet in Fig. 4 shows the use of OpenCV to set the video source as an external webcam with index 1. In the first line, use the cv2. *VideoCapture()* function to create a stamp object that could be used to access the video source. In this case, an argument of 1 is passed to this function, indicating that it wants to use an external camera with index 1.

```
cap = cv2.VideoCapture(1)
```

Fig. 4 Program code snippet for the use of OpenCV

2) Changing Image Format to RGB:

Next, the image taken with the webcam could be converted to RGB using the OpenCV library. Fig. 5 shows a snippet of

program code in Python that uses the OpenCV (cv2) library to change the image color format.

```
# Recolor image to RGB
image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
image.flags.writeable = False
```

Fig. 5 Program Code Snippets for Converting Images to RGB

In the first line, the image whose color format you want to change is stored in the frame variable. Then, the second line explains the process of changing the image color format from BGR (Blue-Green-Red) to RGB (Red-Green-Blue). The BGR color format is the default color format used by OpenCV, while the RGB color format is more commonly used in image processing and visual displays. In this example, the *cv2.cvtColor()* function converts color format with the parameter *cv2.COLOR_BGR2RGB* indicates the conversion from BGR to RGB. In the third line, *image.Flags.Writeable=False* is used to change the writeable status of the image variable to non-writable. This optimizes memory usage because the image variable could not change during subsequent processing. The system can efficiently manage the image variable's memory by changing the writeable state to False. Thus, the program above wholly changes the image color format from BGR to RGB using OpenCV so that the image can be processed and displayed in the appropriate color format.

3) Pose Detection Using MediaPipe:

Fig. 6 shows the landmarks on the media pipe, which in this study used several landmark points in detecting and measuring body angles for bicycle users[28]. This research used several libraries, namely Math, a standard Python library for mathematical operations. In a bike fitting system based on digital image processing on road bikes, math is used to perform mathematical operations, such as calculating arctan and converting angles from radians to degrees[29]. NumPy is a popular library for Python numerical computing. In a bike fitting system based on digital image processing on road bikes, NumPy performs mathematical operations on arrays, such as calculating angles and applying an average filter to angle data[30], [31].

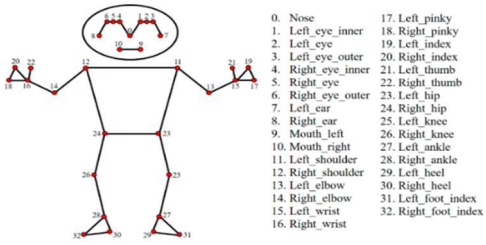

Fig. 6 Landmarks in MediaPipe Pose

At this stage, pose detection was performed on the cyclist who is being recorded using a webcam connected to a computer, as shown in the program code snippet in Fig. 7. Fig. 7 shows the process of setting up or initializing an instance of the MediaPipe library. First, import the MediaPipe library using the import *mediapipe* as *mp* command. Then, use the *mp_pose* and *mp_drawing* objects to access the functions and utilities provided by the MediaPipe library.

```
# Setup instance mediapipe
mp_pose = mp.solutions.pose
mp_drawing = mp.solutions.drawing_utils
```

Fig. 7 Program Code snippet for Pose Detection Using MediaPipe

The mp_pose object is an instance of the *mp.solutions.pose* class is used to detect human poses in videos or images. These objects provide a method for recognizing and tracking landmarks or essential points on the human body, such as hands, feet, elbows, etc. The mp_drawing object is an instance of the *mp.solutions.drawing*_utils class that provides utilities for drawing landmarks and lines associated with human poses. This object can easily remove and connect landmarks to form a detected human pose. By setting up this MediaPipe instance, you can use the functions and utilities provided by MediaPipe to see, track, and draw human poses on your video or image stream.

4) Landmark Extraction:

At this stage, landmark extraction is performed to identify relevant body points for measuring angles in the bike fitting system. In this study, the points identified included the elbows, hips, knees, and ankles. The program code snippet in Fig. 8 is used to access and obtain pose landmark coordinates from pose detection results using the MediaPipe library. Each pose landmark is an essential point on the human body detected in an image or video.

```
#Getcoordinates
right_hip=[landmarks[mp_pose.PoseLandmark.RIGHT_HIP.v
alue].x,landmarks[mp_pose.PoseLandmark.RIGHT_HIP.valu
e].y]
right_elbow=[landmarks[mp_pose.PoseLandmark.RIGHT_ELB
OW.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_ELBO
W.value].y]
right_knee=[landmarks[mp_pose.PoseLandmark.RIGHT_KNEE
.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.v
alue].y]
right_ankle=[landmarks[mp_pose.PoseLandmark.RIGHT_ANK
LE.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_ANKL
E.value].y]
right_shoulder=[landmarks[mp_pose.PoseLandmark.RIGHT_
SHOULDER.value].x,landmarks[mp_pose.PoseLandmark.RIGH
T_SHOULDER.value].y]
right_wrist=[landmarks[mp_pose.PoseLandmark.RIGHT_WRI
ST.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_WRIS
T.value].y]
right_foot_index=[landmarks[mp_pose.PoseLandmark.RIGH
T_FOOT_INDEX.value].x,landmarks[mp_pose.PoseLandmark.
RIGHT_FOOT_INDEX.value].y]
right_heel=[landmarks[mp_pose.PoseLandmark.RIGHT_HEEL
.value].x,landmarks[mp_pose.PoseLandmark.RIGHT_HEEL.v
alue].y]
```

Fig. 8 Program Code snippet for Landmark Extraction

In this example, take several vital landmarks related to body position when cycling, such as the right hip (RIGHT_HIP), right elbow (RIGHT_ELBOW), right knee (RIGHT_KNEE), right ankle (RIGHT_ANKLE), and right toe (RIGHT_FOOT_INDEX). By accessing each landmark's x and y coordinates, we can further analyze the body positions and angles required for the bike fitting system. For example, by measuring the angles between the hip, elbow, and knee points, we can evaluate the suitability of the body position

with the bicycle. These landmark coordinates help obtain relevant data to determine whether the cyclist is in the correct position or needs adjustments. Thus, using the code above provides the ability to access and utilize landmark pose data in implementing a bike fitting system based on digital image processing.

5) Angle Measurement:

Once the body points are identified, the angles between the relevant points are measured. The angles between elbows, hips, knees, ankles, and toes are calculated using the calculate_angle function defined in the program code. This angle measurement provides information about the cyclist's posture. The program code snippet in Fig. 9 implements the calculate_angle function, which calculates the angle between three points based on each point's x and y coordinates. This function utilizes the NumPy and math libraries to perform mathematical operations. First, point a point b, and suggest c coordinates are converted into a NumPy array to facilitate processing. Then, the angle is calculated using the *math.atan2* function to calculate the arctan from the difference in y and x coordinates between points c and b, minus the arctan from the difference in y and x coordinates between points a and b. Next, the angle in radians is converted to an angle in degrees by multiplying by the constant 180.0/np.pi. This is done because of the *math.atan2* function produces angles in radians.

```
def calculate_angle(a, b, c):
    a = np.array(a)
    b = np.array(b)
    c = np.array(c)
    radians = math.atan2(c[1]-b[1], c[0]-b[0]) -
math.atan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle > 180.0:
        angle = 360 - angle
```

Fig. 9 Function to calculate the angle between three points

A check is carried out using the if condition to ensure the resulting angle is 0-180 degrees. If the angle is more than 180 degrees, it is corrected by subtracting it from 360 degrees. Finally, the calculated angle is returned as the function's output. With the *calculate_angle* function, we can calculate the angle between three points, which can later be used to analyze posture and body position in the bike fitting system.

6) Average Filter:

An average filter is applied to the measured angle data to produce more stable angle data. This average filter uses an adjustable *filter_factor* to regulate how previous angle data affects the measured results. This program code snippet can be seen in Fig. 10.

```
def apply_average_filter(data, prev_data,
filter_factor):
    filtered_data = (data * filter_factor) +
(prev_data * (1 - filter_factor))
    return filtered_data
```

Fig. 10 Function to apply an average filter to corner data

The *apply_average_filter* function is used to apply an average filter to the obtained angle data. This average filter aims to smooth out gradual changes in angle values and reduce noise that may be present in the data. This function accepts three arguments: data, which is the angle value to be filtered; *prev_data*, which is the previous angle value before the filter is applied; and *filter_factor*, which is the filter factor. In the process of using the average filter, the angle value to be filtered (data) is multiplied by *filter_factor*, while the previous angle value (*prev_data*) is multiplied by (1 - *filter_factor*). Then, the two multiplication results are combined to produce the filtered angle value (*filtered_data*). Using this average filter, changes in angular values could occur slowly and in a controlled manner, resulting in smoother and more stable angular movements. This is useful in reducing fluctuations in angle values that may occur due to noise or insignificant changes. Thus, we can get more accurate and consistent information from the filtered angle data.

7) Labeling Angle Measurement Results:

Based on the angle range the bike fitting expert determines, a label "Fit" or "Not Fit" is given for each angle measurement. In the program code snippet in Fig. 11, there is a function called *get_fit_label*, which is used to provide a label "Fit" or "Not Fit" based on the given angle value. This function uses four parameters, namely *elbow_angle*, *hip_angle*, *knee_angle*, and *ankle_range*, which are angles that have been calculated previously. In this function, several conditions must be met for someone to be categorized as "Fit." If the hip_angle angle value is in the range between 60 to 110 degrees, *elbow_angle* is in the range between 150 to 160 degrees, *knee_angle* is in the range between 65 to 145 degrees, and *ankle_range* is in the range between 115 to 180 degrees, then the label given is "Fit." However, if one or more of the four angle values does not meet the specified conditions, the label is "Not Fit." The *get_fit_label* function shows whether someone is considered "Fit" or "Not Fit" based on the measured angle value.

```
def get_fit_label(elbow_angle, hip_angle,
knee_angle, ankling_range):
    if 60 <= hip_angle <= 110 and 150 <= elbow_angle
<= 160 and 65 <= knee_angle <= 145 and 115 <=
ankling_range <= 180:
        return "Fit"
    else:
        return "Not Fit"
```

Fig. 11 Function to label " Fit" or "Not Fit"

8) Displaying Frame Image with Angle and Labeling Information:

Frame images that have passed the image processing stage could display the results of angle measurements and labeling. Fig. 12 shows the bike fitting system using digital image processing on a road bike. It was designed using Python programming language and the OpenCV and MediaPipe libraries to detect human body pose in an image or video stream and display angle information on the image. In the initial part of the program, several conditions are checked to determine whether the angles at the elbows, hips, knees, and ankles are within the desired range. If the angles are within the selected range, the text containing the angle values will be

green. Otherwise, the text will be displayed in red. Overall, this system detects the pose of the human body, calculates the angles at the elbows, hips, knees, and ankles, and displays this angle information on the image, as well as visualization of body pose landmarks. This system helps monitor human body movements.

Fig. 12 View of the Bike Fitting System Using Digital Image Processing

9) Take and Save Images:

At this stage, the image is processed, and angle measurements and labeling are obtained, taken, and saved in a computer/laptop file. A snippet of program code is shown in Fig. 13.

```
mp_drawing.draw_landmarks(image,results.pose_landmar
ks,mp_pose.POSE_CONNECTIONS,mp_drawing.DrawingSpec(c
olor=(245,117,66), thickness=2,
circle_radius=2),mp_drawing.DrawingSpec(color=(245,6
6,230), thickness=2, circle_radius=2))
cv2.imshow('Mediapipe Feed', image)
# Add logic to take an image when the 's' key is
pressed
        key = cv2.waitKey(1)
        if key == ord('s'):
            save_image(image)
        if key == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

Fig. 13 Program Code Snippet "Take and Save Image"

Fig. 13 is the final part of the Python code, which uses the OpenCV and MediaPipe libraries to detect human body pose in a video stream. In this section, the program adds logic to take a picture when the 's' key is pressed. The code *cv2.waitKey(1)* waits for a keyboard key to be pressed for one millisecond and returns a value representing the key packed. If the 's' key is pressed (represented by *ord('s')*), then the program will call the *save_image()* function to save the currently displayed image. Next, the program checks whether the 'q' key is pressed. If the 'q' button is pressed (represented by *ord('q')*), then the program will exit the main loop and proceed to the following line of code. Once the loop is complete, the program will free the video stream resources using *cap.release()* and close all windows opened by OpenCV using *cv2.destroyAllWindows()*. Overall, the program provides additional functionality for the user, namely the ability to capture images from the video stream when the 's' key is pressed and the ability to stop the program by pressing the 'q' key.

III. Results and Discussion

In this research, a bike fitting system based on digital image processing on a road bike was tested. Later, data will be tested from road bike cyclists who will take angle measurements using digital image processing that has been designed; the results of the angle measurements will later be adjusted to the measurement reference from literacy. After measuring the angle and the results are not close to the measurement reference, the bicycle will be adjusted to obtain test results that match the measurement reference so that the bicycle user can get an efficient and comfortable position when using the road bike.

C. Preparatory Phase

The following things must be prepared before image processing on the bike fitting system.

a. Prepare the testing requirements for the road bike, webcam, laptop, trainer, goniometer, and laptop.
b. Setting the position of the webcam camera on a road bike
c. Adjusting the position of road bike users to make it easier to capture video streams.
d. Running Programs

The first thing that needs to be prepared is a road bike installed with a trainer and webcam for recording video streams and a laptop for image processing. Then, set the webcam position with the road bike. The following is a measurement reference for a bike fitting system using digital image processing on a road bike.

The bike fitting system uses digital image processing to measure the angles of several body points. The measurement results could be adjusted to the angle measurement reference to get the results 'Fit' or 'Not Fit.' Several points on the body are used as measurement references, namely as follows.

a. Elbow Angle has an angle range of 150 to 160 degrees.
b. Hip Angle (Hip Angle) has an angle range of 60 to 110 degrees.
c. Knee Angle (Elbow Angle) has an angle range of 65 to 145 degrees.
d. Ankling Range has an angle range of 115 to 180 degrees.

D. Data Testing Results Using Image Processing

Data testing using image processing on the bike fitting system uses a laptop and several other supporting tools such as a webcam, trainer, and road bike. The display of data testing using image processing can be seen in Fig. 14.

Fig. 14 Data Testing Using Image Processing

E. Data Testing Results with Manual Measurements

Data testing using manual measurements on the bike fitting system uses a measuring instrument to measure angles, namely a goniometer. A goniometer measured angles on body parts according to the measurement reference. The data testing using manual measurements can be seen in Fig. 15.

Fig. 15 Testing Data Using Manual Measurements

F. Data Before Fit

They are testing data on the bike fitting system based on image processing on road bikes when the bike condition is not yet fit, whereas road bikes have the following frame sizes.

a. Vertical Tube: 46 cm
b. Horizontal Tube: 50 cm
c. Saddle height: 72 cm
d. Power length: 12.5 cm
e. Handlebar: 40 cm

At this stage, data was collected on image processing of the road bike fitting system. The data test results can be seen in Table 1. In Table 1, the data testing results using image processing were obtained by testing ten times, and the test results were labeled "Not Fit," which provides information on whether the bicycle frame used is unsuitable for bicycle users. Hence, it is necessary to change the size of the bicycle frame.

TABLE I
DATA USING IMAGE PROCESSING

Data	Elbow Angle	Hip Angle	Knee Angle	Ankling Range	Labeling
1	165.00	112.80	154.90	124.90	Not Fit
2	164.00	110.00	151.00	125.00	Not Fit
3	165.00	110.00	148.00	126.00	Not Fit
4	165.00	110.00	150.00	125.00	Not Fit
5	165.00	112.00	153.00	131.00	Not Fit
6	170.00	110.00	153.00	131.00	Not Fit
7	165.00	111.00	153.00	130.00	Not Fit
8	161.00	110.00	151.00	129.00	Not Fit
9	166.00	108.00	151.00	127.00	Not Fit
10	161.00	106.00	150.00	132.00	Not Fit

In the next stage, data was collected using measurements of the manual road bike fitting system. The test results are in Table 2. In Table 2 above, the results of data testing using image processing were obtained by testing ten times, and where the test results showed that some data was labeled "Not Fit". The test results were marked "Not Fit", which provides information if the bicycle frame used is unsuitable for the bicycle user, so it is necessary to change the size of the bicycle frame.

TABLE II
DATA USING MANUAL MEASUREMENTS

Data	Elbow Angle	Hip Angle	Knee Angle	Ankling Range	Labeling
1	165.00	112.80	154.90	124.90	Not Fit
2	164.00	110.00	151.00	125.00	Not Fit
3	165.00	110.00	148.00	126.00	Not Fit
4	165.00	110.00	150.00	125.00	Not Fit
5	165.00	112.00	153.00	131.00	Not Fit
6	170.00	110.00	153.00	131.00	Not Fit
7	165.00	111.00	153.00	130.00	Not Fit
8	161.00	110.00	151.00	129.00	Not Fit
9	166.00	108.00	151.00	127.00	Not Fit
10	161.00	106.00	150.00	132.00	Not Fit

G. Data After Fit

Data collection for the bike fitting system is based on image processing on the road bike when it is in a fit condition. The saddle height of the road bike will be changed to 68.874 cm, which is obtained by multiplying the cyclist's inseam height, namely 78 cm and 0.883, so that the bike has the following frame size.

a. Vertical Tube: 46 cm
b. Horizontal Tube: 50 cm
c. Saddle height: 68.874 cm
d. Power length: 12.5 cm
e. Handlebar: 40 cm

At this stage, data was tested on image processing of the road bike fitting system. The data testing results using image processing can be seen in Table 3. In Table 3, the data testing results using image processing were obtained by carrying out the test 10 times; the test results included some data labeled "Fit," which provides information about whether the bicycle frame is suitable for the bicycle user.

In the next stage, data testing was carried out using manual road bike fitting system measurements. The data test results are shown in Table 4.

TABLE III
DATA USING IMAGE PROCESSING

Testing	Elbow Angle	Hip Angle	Knee Angle	Ankling Range	Labeling
1	156.88	97.11	129.58	126.9	Fit
2	156.48	96.96	129.22	126.18	Fit
3	156.54	97.95	128.97	127.31	Fit
4	155.45	98.67	131.32	127.1	Fit
5	156.53	99.3	132.48	127.4	Fit
6	156.71	99.93	132.58	125.93	Fit
7	156.19	99.26	132.8	125.16	Fit
8	158.3	98.41	131.64	127.49	Fit
9	153.61	98.68	131.94	126.22	Fit
10	153.25	98.01	131.35	126.59	Fit

TABLE IV
DATA USING MANUAL MEASUREMENTS

Testing	Elbow Angle	Hip Angle	Knee Angle	Ankling Range	Labeling
1	156.00	99.00	130.00	123.00	Fit
2	157.00	99.00	129.00	123.00	Fit
3	156.00	99.00	126.00	122.00	Fit
4	155.00	101.00	130.00	125.00	Fit
5	155.00	102.00	133.00	124.00	Fit
6	156.00	104.00	134.00	124.00	Fit
7	155.00	102.00	135.00	123.00	Fit
8	160.00	100.00	130.00	125.00	Fit
9	154.00	99.00	133.00	121.00	Fit
10	153.00	100.00	130.00	123.00	Fit

In Table 4, the results of data testing using image processing were obtained by testing ten times, and where the test results obtained some data labeled "Fit".

H. Discussion

In data analysis, a comparison was made between measurements using image processing and manual measurements. The following are the things that need to be analyzed to measure accuracy between measurements using image processing and manual measurements. After taking data on the road bike fitting system, at this stage, the absolute difference between the manual measurement results and the image processing results for each sample was measured. After calculating the fundamental difference, the percentage error was measured by calculating the percentage error between the manual measurement results and the image processing results for each sample. Next, a comparison was made between manual measurements and image processing on the bike fitting road bike system by comparing the angle results at the elbow angle of bicycle users, which can be seen in Table 5.

TABLE V
COMPARISON RESULTS OF MANUAL MEASUREMENTS WITH IMAGE PROCESSING RESULTS AT ELBOW ANGLE

Testing	Elbow Angle			
	Image processing	Manual	Absolute Difference	Error Percentage
1	166.26	165.00	1.26	0.76
2	161.83	164.00	2.17	1.32
3	163.58	165.00	1.42	0.86
4	162.69	165.00	2.31	1.40
5	161.5	165.00	3.5	2.12
6	166.78	170.00	3.22	1.89
7	164.11	165.00	0.89	0.54
8	161.28	161.00	0.28	0.17
9	164.47	166.00	1.53	0.92
10	159.53	161.00	1.47	0.91
11	156.88	156.00	0.88	0.56
12	156.48	157.00	0.52	0.33
13	156.54	156.00	0.54	0.35
14	155.45	155.00	0.45	0.29
15	156.53	155.00	1.53	0.99
16	156.71	156.00	0.71	0.46
17	156.19	155.00	1.19	0.77
18	158.3	160.00	1.7	1.06
19	153.61	154.00	0.39	0.25
20	153.25	153.00	0.25	0.16
Mean Percentage Error (MPE)				1.46

Comparing manual measurements with image processing results at right angles obtained an average error percentage of 1.46%. The most considerable absolute difference occurred in test 5 with a difference of 3.5, and the most minor fundamental difference appeared in test 20 with a difference of 0.25. The following are the results of a comparison between manual measurements and image processing on the bike fitting road bike system by comparing the results of the hip angle of bicycle users, which can be seen in Table 6.

Comparing manual measurements with image processing results on hip angles obtained an average error percentage of 1.37%. The most considerable absolute difference occurred in test 16, with a difference of 4.07, and the smallest one in test 3, with a fundamental difference of 0.13. Table 7 shows the comparison results between manual measurements and image processing on the road bike fitting system by comparing the results of the hip angle of bicycle users.

TABLE VI
COMPARATIVE RESULTS OF MANUAL MEASUREMENTS WITH IMAGE PROCESSING RESULTS AT HIP ANGLE

Testing	Hip Angle			
	Image processing	Manual	Absolute Difference	Error Percentage
1	111.81	112.80	0.99	0.88
2	110.38	110.00	0.38	0.35
3	109.87	110.00	0.13	0.12
4	110.45	110.00	0.45	0.41
5	110.89	112.00	1.11	0.99
6	110.64	110.00	0.64	0.58
7	110.49	111.00	0.51	0.46
8	110.13	110.00	0.13	0.12
9	109.24	108.00	1.24	1.15
10	107.92	106.00	1.92	1.81
11	97.11	99.00	1.89	1.91
12	96.96	99.00	2.04	2.06
13	97.95	99.00	1.05	1.06
14	98.67	101.00	2.33	2.31
15	99.3	102.00	2.7	2.65
16	99.93	104.00	4.07	3.91
17	99.26	102.00	2.74	2.69
18	98.41	100.00	1.59	1.59
19	98.68	99.00	0.32	0.32
20	98.01	100.00	1.99	1.99
Mean Percentage Error (MPE)				1.37

TABLE VII
COMPARISON RESULTS OF MANUAL MEASUREMENTS WITH IMAGE PROCESSING RESULTS AT KNEE ANGLE

Testing	Knee Angle			
	Image processing	Manual	Absolute Difference	Error Percentage
1	151.73	154.90	3.17	2.05
2	150.95	151.00	0.05	0.03
3	149.68	148.00	1.68	1.14
4	150.63	150.00	0.63	0.42
5	152.2	153.00	0.8	0.52
6	151.97	153.00	1.03	0.67
7	151.93	153.00	1.07	0.70
8	150.82	151.00	0.18	0.12
9	150.38	151.00	0.62	0.41
10	150.74	150.00	0.74	0.49
11	129.58	130.00	0.42	0.32
12	129.22	129.00	0.22	0.17
13	128.97	126.00	2.97	2.36
14	131.32	130.00	1.32	1.02
15	132.48	133.00	0.52	0.39
16	132.58	134.00	1.42	1.06
17	132.8	135.00	2.2	1.63
18	131.64	130.00	1.64	1.26
19	131.94	133.00	1.06	0.80
20	131.35	130.00	1.35	1.04
Mean Percentage Error (MPE)				0.83

Comparing manual measurements with image processing results on knee angles obtained an average error percentage of 0.83%. The most considerable absolute difference occurred in test 1 with a difference of 3.17, and the most minor fundamental difference appeared in test 2 with a difference of 0.05. Table 8 compares manual measurements and image processing on the road bike fitting system by comparing the angle results on the ankle range of bicycle users.

TABLE VIII
COMPARISON RESULTS OF MANUAL MEASUREMENTS WITH IMAGE PROCESSING RESULTS AT THE ANKLING RANGE

Testing	Ankling Range			
	Image processing	Manual	Absolute Difference	Error Percentage
1	123.55	124.90	1.35	1.08
2	124.51	125.00	0.49	0.39
3	124.23	126.00	1.77	1.40
4	124.75	125.00	0.25	0.20
5	129.41	131.00	1.59	1.21
6	130.86	131.00	0.14	0.11
7	129.93	130.00	0.07	0.05
8	127.22	129.00	1.78	1.38
9	125.14	127.00	1.86	1.46
10	130.86	132.00	1.14	0.86
11	126.9	123.00	3.9	3.17
12	126.18	123.00	3.18	2.59
13	127.31	122.00	5.31	4.35
14	127.1	125.00	2.1	1.68
15	127.4	124.00	3.4	2.74
16	125.93	124.00	1.93	1.56
17	125.16	123.00	2.16	1.76
18	127.49	125.00	2.49	1.99
19	126.22	121.00	5.22	4.31
20	126.59	123.00	3.59	2.92
Mean Percentage Error (MPE)				1.76

Comparing manual measurements with image processing results at the ankle angle obtained an average error percentage of 1.76%. The most considerable absolute difference occurred in test 13, with a difference of 5.31, and the minor fundamental difference appeared in test 7, with a difference of 0.07.

IV. CONCLUSION

From the results of testing and analysis in this research, it can be concluded that the bike fitting system that uses image processing has succeeded in providing objective information regarding angle measurements at elbow angle, hip angle, knee angle, and ankle range for cyclists. Image processing and pose analysis methods have helped determine whether the rider has achieved the ideal position for comfort and efficiency. The system test results show a low error rate, with elbow angle having an average error of 0.81%, hip angle of 1.37%, knee angle of 0.83%, and ankle range of 1.76%. Thus, this research contributes significantly to supporting cyclists in achieving a position appropriate to their inseam height, as illustrated by the example of a cyclist with an inseam height of 78 cm and a suitable bicycle specification. In conclusion, this image processing-based bike fitting system has great potential to improve the comfort and efficiency of bicycle riding.

ACKNOWLEDGMENT

This research was supported by the Universitas Sumatera Utara through the 2024 TALENTA Research Grant Applied Research scheme, Contract Number 143/UN5.2.3.1/PPM/KP-TALENTA/R/2023.

REFERENCES

[1] S. Sidjabat, "Sepeda sebagai alat transportasi ramah lingkungan," *J. Manaj. Bisnis Transp. Logistik*, vol. 3, no. 1, pp. 117–122, 2016.

[2] L. Pearson, B. Gabbe, S. Reeder, and B. Beck, "Barriers and enablers of bike riding for transport and recreational purposes in Australia," *J Transp Health*, vol. 28, p. 101538, Jan. 2023, doi:10.1016/J.JTH.2022.101538.

[3] L. Pearson *et al.*, "The potential for bike riding across entire cities: Quantifying spatial variation in interest in bike riding," *J Transp Health*, vol. 24, p. 101290, Mar. 2022, doi:10.1016/J.JTH.2021.101290.

[4] J. I. Priego Quesada, P. Pérez-Soriano, A. G. Lucas-Cuevas, R. Salvador Palmer, and R. M. Cibrián Ortiz de Anda, "Effect of bike-fit in the perception of comfort, fatigue and pain," *J Sports Sci*, vol. 35, no. 14, pp. 1459–1465, Jul. 2017, doi:10.1080/02640414.2016.1215496.

[5] J. I. Priego Quesada, Z. Y. Kerr, W. M. Bertucci, and F. P. Carpes, "The association of bike fitting with injury, comfort, and pain during cycling: An international retrospective survey," *Eur J Sport Sci*, vol. 19, no. 6, pp. 842–849, Jul. 2019, doi:10.1080/17461391.2018.1556738.

[6] J. I. Priego Quesada, Z. Y. Kerr, W. M. Bertucci, and F. P. Carpes, "The association of bike fitting with injury, comfort, and pain during cycling: An international retrospective survey," *Eur J Sport Sci*, vol. 19, no. 6, pp. 842–849, Jul. 2019, doi:10.1080/17461391.2018.1556738.

[7] G. Millour, A. T. Velásquez, and F. Domingue, "A literature overview of modern biomechanical-based technologies for bike-fitting professionals and coaches," *Int J Sports Sci Coach*, vol. 18, no. 1, pp. 292–303, Feb. 2023, doi: 10.1177/17479541221123960.

[8] P. Burt, *Bike Fit Optimise Your Bike Position For High Performance and Injury Avoidance*, First edition. London: Bloomsbury Publishing Plc, 2014.

[9] J. Braeckvelt, J. De Bock, J. Schuermans, S. Verstockt, E. Witvrouw, and J. Dierckx, "The need for data-driven bike fitting: Data study of subjective expert fitting," in *Proc. 7th Int. Conf. Sport Sci. Res. Technol. Support (icSports)*, Vienna, Austria, Sep. 20–21, 2019, pp. 181–189.

[10] R. D. Scoz *et al.*, "Validation of a 3d camera system for cycling analysis," *Sensors*, vol. 21, no. 13, p. 4473, Jul. 2021, doi:10.3390/S21134473/S1.

[11] R. TH. Hasan and A. B. Sallow, "Face Detection and Recognition Using OpenCV," *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, pp. 86–97, Oct. 2021, doi: 10.30880/jscdm.2021.02.02.008.

[12] J. Sigut, M. Castro, R. Arnay, and M. Sigut, "OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts," *IEEE Transactions on Education*, vol. 63, no. 4, pp. 328–335, Nov. 2020, doi: 10.1109/TE.2020.2993013.

[13] M. F. Wicaksono and M. D. Rahmatya, "3D Geometric Shape and Colors Interactive Learning Media using Raspberry Pi, OpenCV, and TensorFlow Lite," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 13, no. 5, pp. 1710–1718, Oct. 2023, doi: 10.18517/ijaseit.13.5.18443.

[14] P. Chaudhari, "Skin Cancer Classification Application Using Machine Learning," International Journal of Data Science, vol. 2, no. 1, pp. 47–55, Sep. 2021, doi: 10.18517/ijods.2.1.47-55.2021.

[15] T. H. Nasution, A. Simon, F. Fahmi, K. Tanjung, M. Zarlis, and N. Noer, "Analysis of the use of CMUcam5 Pixy camera in wheeled soccer robots," *IOP Conf Ser Mater Sci Eng*, vol. 851, no. 1, p. 012034, May 2020, doi: 10.1088/1757-899X/851/1/012034.

[16] K. Kounlaxay, Y. C. Yoon, and S. K. Kim, "Vehicle License Plate Detection and Recognition using OpenCV and Tesseract OCR," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 14, no. 4, pp. 1170–1177, Aug. 2024, doi: 10.18517/ijaseit.14.4.18137.

[17] F. Zhang *et al.*, "MediaPipe Hands: On-device Real-time Hand Tracking," Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.10214

[18] A. S. B. Pauzi *et al.*, "Movement Estimation Using Mediapipe BlazePose," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13051 LNCS, pp. 562–571, 2021, doi:10.1007/978-3-030-90235-3_49.

[19] O. Z. Jie, L. T. Ming, and T. C. Wee, "Biometric Authentication based on Liveness Detection Using Face Landmarks and Deep Learning Model," *JOIV : International Journal on Informatics Visualization*, vol. 7, no. 3–2, pp. 1057–1065, Nov. 2023, doi:10.30630/JOIV.7.3-2.2330.

[20] A. Latreche, R. Kelaiaia, A. Chemori, and A. Kerboua, "Reliability and validity analysis of MediaPipe-based measurement system for some human rehabilitation motions," *Measurement (Lond)*, vol. 214, Jun. 2023, doi: 10.1016/j.measurement.2023.112826.

[21] R. Madhumitha, "Efficient Vehicle Registration Recognition System: Enhancing Accuracy and Power Efficiency through Digital Image Processing," *International Journal of Advanced Science Computing and Engineering*, vol. 6, no. 2, pp. 74–79, Jul. 2024, doi:10.62527/ijasce.6.2.206.

[22] B. Kim, M. Chae, Y. Kim, I. Jeong, and H. Lee, "Forward Head Posture Classification Using Deep Learning Models on Facial Recognition Data," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 14, no. 2, pp. 805–810, Apr. 2024, doi: 10.18517/ijaseit.14.2.18519.

[23] A. Al-Khulaqi, N. Palanichamy, S. C. Haw, and S. C. Raja, "Evaluating Machine Learning and Deep Learning Algorithms for Predictive Maintenance of Hydraulic Systems," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 15, no. 1, pp. 52–59, Feb. 2025, doi: 10.18517/ijaseit.15.1.12407.

[24] B. Heinold, "A Practical Introduction to Python Programming," 2021, Accessed: May 31, 2024. [Online]. Available: http://dlib.hust.edu.vn/handle/HUST/21775.

[25] J. Suzuki, "Kernel Methods for Machine Learning with Math and Python," *Kernel Methods* for *Machine Learning with Math and Python*, 2022, doi: 10.1007/978-981-19-0401-1.

[26] C. R. Harris *et al.*, "Array programming with NumPy," *Nature 2020 585:7825*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi:10.1038/s41586-020-2649-2.

[27] C. R. Harris *et al.*, "Array programming with NumPy," *Nature 2020 585:7825*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi:10.1038/s41586-020-2649-2.

[28] A. Gupta, P. L. Shrestha, B. Thapa, R. Silwal, and R. Shrestha, " Knee Flexion/Extension Angle Measurement for Gait Analysis Using Machine Learning Solution 'MediaPipe Pose' and Its Comparison with Kinovea ® ," *IOP Conf Ser Mater Sci Eng*, vol. 1279, no. 1, p. 012004, Mar. 2023, doi: 10.1088/1757-899x/1279/1/012004.

[29] A. Meurer *et al.*, "SymPy: Symbolic computing in python," *PeerJ Comput Sci*, vol. 2017, no. 1, 2017, doi: 10.7717/peerj-cs.103.

[30] W. Schmidt and M. Völschow, "Numerical Python in Astronomy and Astrophysics," 2021, doi: 10.1007/978-3-030-70347-9.

[31] M. Arman, H. Sujon, and H. Mustafa, "Comparative Study of Machine Learning Models on Multiple Breast Cancer Datasets," *International Journal of Advanced Science Computing and Engineering*, vol. 5, no. 1, pp. 15–24, Jan. 2023, doi: 10.62527/IJASCE.5.1.105.