2023

Mini Project Report

Semester VI

GAIT ANALYSIS USING COMPUTER VISION

Submitted By

Karuna Padwal

Pratiksha Sutar

Abel Dsouza

Rahul Patil

Under the Guidance of

Prof. Suvarna Udgire (Guide)



### Certificate of Approval

Vidyalankar Institute of Technology

Wadala(E), Mumbai 400 037

## University of Mumbai

Department of Biomedical Engineering

This is to Certify that

Karuna Padwal

Pratiksha Sutar

Abel Dsouza

Rahul Patil

Have successfully carried out Mini Project work entitled

GAIT ANALYSIS USING COMPUTER VERSION

in Second year, Semester V of BE degree course in

**Biomedical Engineering**

As laid down by University of Mumbai during the academic year 2023-24

Under the Guidance of

Prof. Suvarna Udgire (Guide)

|  |  |
| --- | --- |
| Signature of Guide | Head of Department |

|  |  |  |
| --- | --- | --- |
| Examiner 1 | Examiner 2 | Principal |

**Declaration**

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of student Roll No. Signature

1. Karuna Padwal 20105A0050
2. Pratiksha Sutar 20105A0049
3. Abel Dsouza 20105A0010
4. Rahul Patil 20105A0054

Date:

**Acknowledgements (Sample Format-You can modify)**

This project “GAIT ANALYSIS USING COMPUTER VISION” would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly indebted to “Prof. Suvarna Udgire (Guide)**”** for their guidance and constant supervision as well as for providing the necessary information regarding the project and for their support in completing the project.

We would like to express our gratitude towards **Dr. Gajanan Nagre** Head Biomedical Engineering Department, faculty and all the lab assistants of Vidyalankar Institute of technology for their kind co-operation and encouragement which helped us in completion of this project. We would like to express our gratitude towards our parents for their encouragement and support.

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter No** | **Section No** | **Description** | **Page No** |
|  |  | Abstract |  |
| 1 |  | Introduction |  |
| 2 |  | Aim & Objective |  |
| 3 |  | Literature Survey |  |
| 4 |  | Problem Statement |  |
| 5 |  | Scope |  |
| 6 |  | Proposed System |  |
|  | 6.1 | Process Model Used for the Project. |  |
|  | 6.2 | Methodology |  |
| 7 |  | Analysis |  |
|  | 7.1 | Feasibility Study |  |
|  | 7.2 | Timeline Chart |  |
| 8 |  | Design |  |
|  | 8.1 | Block Diagram |  |
|  | 8.2 | Simulations |  |
| 9 |  | Hardware and Software Requirement |  |
| 10 |  | Implementation Plan for Next Semester |  |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Fig. No** | **Title** | **Page No** |
| 1 | Block Diagram |  |
| 2 | Simulation (Screenshot) |  |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table. No** | **Title** | **Page No** |
| 1 | Timeline Chart |  |
| 2 | Implementation Plan for Next Semester |  |

**ABSTRACT.**

Gait analysis is a crucial tool in diagnosing and monitoring gait-related disorders. However, traditional methods are expensive and require bulky equipment, making them inaccessible to many healthcare providers and patients.

Thanks to the advancements in computer vision and deep learning, gait analysis using computer vision has emerged as a promising alternative approach. This technique involves using cameras and image processing algorithms to capture and analyze gait patterns and abnormalities in real-time, providing quantitative and objective measurements.

This paper presents an overview of gait analysis using computer vision, including its hardware and software requirements, block diagram, timeline chart, and feasibility study. The potential benefits of this approach, such as its cost-effectiveness, non-invasiveness, and accessibility, are discussed. The challenges and future directions of gait analysis using computer vision are also addressed, highlighting the need for standardization of data collection and analysis protocols, as well as the optimization of algorithms for different applications and settings.

1. **INTRODUCTION.**

Gait analysis is a critical area of study that assesses human walking patterns and diagnoses gait-related disorders. Traditional gait analysis methods use specialized equipment, such as force plates and motion capture systems, which can be expensive and time-consuming. However, recent advancements in computer vision and machine learning have enabled researchers to conduct gait analysis using simple video recordings.

Computer vision-based gait analysis involves extracting relevant features from video recordings of individuals walking and analyzing them to determine their gait pattern. This approach offers several advantages over traditional methods, including lower cost, easier setup, and the ability to perform gait analysis in real-time.

In this literature survey, we will review various research studies and papers on gait analysis using computer vision. We will explore the different techniques and algorithms used to extract gait-related features from video recordings and analyze their effectiveness in identifying gait patterns and detecting gait abnormalities. Moreover, we will discuss the potential applications of computer vision-based gait analysis in clinical settings, such as the diagnosis and monitoring of gait-related disorders.

1. **AIM & OBJECTIVE.**

This literature survey aims to present a comprehensive overview of the current research on gait analysis utilizing computer vision techniques. The objective is to examine various techniques and algorithms employed in computer vision-based gait analysis, evaluate their ability to identify gait patterns and abnormalities, and investigate their potential use in clinical settings. Upon completion of this literature survey, readers will gain a thorough understanding of the latest advancements in gait analysis using computer vision and the potential future research directions in this field.

1. **LITERATURE SURVEY**

In recent years, there has been a significant amount of research on gait analysis using computer vision. A literature survey reveals several papers that highlight the effectiveness of computer vision-based approaches for gait analysis and their potential applications in clinical practice.

One such paper proposes a gait analysis system that uses multiple view cameras and convolutional neural networks (CNNs) to automatically extract gait features and identify gait abnormalities with high accuracy. Another review paper discusses different computer vision-based approaches for gait analysis, their challenges, and limitations, providing valuable insights into future research directions.

In addition, there are papers that present real-time gait analysis systems that use deep learning and inertial sensors to accurately classify different gait patterns and identify gait abnormalities in real-time. These automated gait analysis systems are particularly useful for clinical practice, where they can assist in the diagnosis and monitoring of gait-related disorders.

Furthermore, multi-view gait analysis systems that use CNNs to extract gait features from multiple video recordings have also been proposed. These systems can accurately classify different gait patterns and detect gait abnormalities with high accuracy, making them valuable tools for clinical diagnosis and research.

In summary, computer vision-based approaches for gait analysis have shown great potential in improving the diagnosis and monitoring of gait-related disorders. These papers provide a glimpse into the current state-of-the-art gait analysis using computer vision and highlight the need for further research to develop more robust and accurate gait analysis systems.

1. **PROBLEM STATEMENT**

Gait analysis is an essential tool in diagnosing and monitoring various gait-related disorders. However, traditional methods for gait analysis can be costly, time-consuming, and require specialized equipment. Computer vision-based approaches have emerged as a promising alternative, but they face several challenges. These include the variability in gait patterns among individuals, the need for accurate feature extraction and classification, and the integration of multiple sensors for more accurate analysis. Therefore, the problem statement for gait analysis using computer vision is to develop efficient and reliable methods that can accurately analyze gait patterns in real-time, be easily implemented in clinical settings, and improve the diagnosis and monitoring of gait-related disorders.

1. **SCOPE**

The scope of gait analysis using computer vision is broad and encompasses various areas, such as developing novel algorithms and techniques for feature extraction and classification, integrating multiple sensors for more precise analysis, and applying gait analysis in clinical settings. It also involves exploring different camera setups, lighting conditions, and environmental factors that may affect gait analysis. The scope further includes investigating the potential use of gait analysis in fields like sports performance, rehabilitation, and geriatrics. Overall, the scope of gait analysis using computer vision aims to create efficient and effective methods for gait analysis that can be widely implemented in different settings to aid in the diagnosis, treatment, and monitoring of gait-related disorders.

* 1. **PROPOSED SYSTEM**

As gait analysis using computer vision is a rapidly evolving field, there are several proposed systems and approaches for gait analysis, each with the common goal of developing accurate, reliable, and efficient methods for gait analysis. One proposed system uses multiple view cameras and convolutional neural networks (CNNs) to automatically extract gait features from video recordings and identify gait abnormalities with high accuracy. Another proposed system uses deep learning and inertial sensors for real-time gait analysis, accurately classifying different gait patterns and identifying abnormalities.

Other proposed systems focus on automated gait analysis for clinical practice using deep learning, detecting, and classifying different gait patterns and abnormalities with high accuracy. There are also proposed systems for multi-view gait analysis, which use CNNs to extract gait features from multiple video recordings, accurately classifying different gait patterns and detecting abnormalities.

Overall, these proposed systems leverage advanced technologies such as deep learning and CNNs to extract relevant features accurately and efficiently from video recordings. They have the potential to improve the diagnosis, treatment, and monitoring of gait-related disorders and can be easily implemented in clinical settings. However, further research is needed to evaluate and optimize these systems for different applications and settings.

* 1. **METHODOLOGY**

The methodology for gait analysis using computer vision typically involves several steps:

1. Import Require Libraries

This code imports several libraries, including tkinter, PIL, OpenCV, numpy, pandas, and xlsxwriter.

* + 1. The **cv2** library is used for computer vision tasks, such as image processing and analysis.
    2. The **numpy** library is used for numerical computing with Python.
    3. The **openpyxl** library is used to write data to Excel files.
    4. The **MediaPipe** library is used to perform pose estimation.

import cv2

import mediapipe as mp

import numpy as np

import openpyxl

1. Data acquisition: The first step is to collect video recordings of the subject's gait using one or more cameras. The cameras may be set up in different positions and angles to capture different views of the gait.

# Load the image

img = cv2.imread('image.jpg')

# Convert the image to grayscale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Apply thresholding to create a binary image

thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)[1]

# Find the edges in the image using the Canny algorithm

edges = cv2.Canny(thresh, 100, 200)

# Find contours in the edges image

contours, hierarchy = cv2.findContours(edges, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Loop through the contours and find the one that is most likely to be a leg

max\_area = 0

max\_contour = None

for contour in contours:

    area = cv2.contourArea(contour)

    if area > max\_area:

        max\_area = area

        max\_contour = contour

# Draw the contour on the original image

cv2.drawContours(img, [max\_contour], -1, (0, 255, 0), 2)

# Display the original image with the leg contour drawn on it

cv2.imshow('Leg Detection', img)

cv2.waitKey(0)

This code is an example of leg detection in an image using OpenCV. Here is an explanation of each step:

1. Load the image: The code starts by reading an image file named 'image.jpg' using the cv2.imread() function and storing it in the 'img' variable.
2. Convert the image to grayscale: The cv2.cvtColor() function is used to convert the image from its default BGR (Blue-Green-Red) color format to grayscale, which simplifies image processing.
3. Apply thresholding to create a binary image: The cv2.threshold() function is used to threshold the grayscale image and create a binary image. The cv2.THRESH\_BINARY\_INV flag is used to invert the colors of the binary image.
4. Find the edges in the image using the Canny algorithm: The cv2.Canny() function is used to detect the edges in the binary image using the Canny edge detection algorithm.
5. Find contours in the edges image: The cv2.findContours() function is used to find the contours (boundaries) of the leg in the image. The cv2.RETR\_EXTERNAL flag is used to retrieve only the external contours (i.e., the leg).
6. Loop through the contours and find the one that is most likely to be a leg: The code loops through all of the contours found in the previous step and selects the one with the largest area. This contour is most likely to be the leg.
7. Draw the contour on the original image: The cv2.drawContours() function is used to draw the selected contour on the original image.
8. Display the original image with the leg contour drawn on it: The cv2.imshow() function is used to display the modified image on the screen, and cv2.waitKey(0) waits until a key is pressed before closing the window.
9. Preprocessing: The video recordings may undergo preprocessing to enhance their quality and remove any noise or unwanted elements. This may involve techniques such as background subtraction, denoising, and image filtering.
10. Feature extraction: The next step is to extract relevant features from the video recordings that can be used to characterize the subject's gait. These features include the angle of various body parts, as well as parameters such as joint angle.

import cv2

import numpy as np

import openpyxl

cap = cv2.VideoCapture(0)

# Parameters for edge detection

threshold1 = 50

threshold2 = 150

# Parameters for line detection

rho = 1

theta = np.pi/180

threshold = 50

min\_line\_length = 100

max\_line\_gap = 10

# Create a new Excel workbook and worksheet

wb = openpyxl.Workbook()

ws = wb.active

ws.title = 'Angle Measurements'

ws['A1'] = 'Frame'

ws['B1'] = 'Angle (degrees)'

row = 2

while True:

    # Read a frame from the camera

    ret, frame = cap.read()

    # If we failed to read a frame, abort

    if not ret:

        print("Failed to read frame from camera.")

        break

    # Convert the frame to grayscale for edge detection

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    # Perform edge detection on the grayscale image

    edges = cv2.Canny(gray, threshold1, threshold2, apertureSize=3)

    # Detect lines in the image using the HoughLinesP method

    lines = cv2.HoughLinesP(edges, rho, theta, threshold, minLineLength=min\_line\_length, maxLineGap=max\_line\_gap)

    # Only process if we detected at least two lines

    if lines is not None and len(lines) >= 2:

        # Draw the lines on the frame

        for line in lines:

            x1, y1, x2, y2 = line[0]

            cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        # Calculate the angle between the two lines

        x1, y1, x2, y2 = lines[0][0]

        line1\_angle = np.arctan2(y2 - y1, x2 - x1)

        x1, y1, x2, y2 = lines[1][0]

        line2\_angle = np.arctan2(y2 - y1, x2 - x1)

        angle = np.abs(np.degrees(line1\_angle - line2\_angle))

        # Display the angle on the frame

        cv2.putText(frame, f"Angle: {angle:.2f} deg", (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2)

        # Write the angle to the Excel sheet

        ws.cell(row=row, column=1).value = row - 1

        ws.cell(row=row, column=2).value = angle

        row += 1

    # Display the resulting frame

    cv2.imshow('frame', frame)

    # Check for the 'q' key and exit if pressed

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Save the Excel workbook

wb.save('angle\_measurements.xlsx')

# Release the capture and destroy all windows

cap.release()

cv2.destroyAllWindows()

This is a Python code that uses OpenCV and openpyxl libraries to measure the angle between two lines in real-time using a webcam. Here's an overview of what the code does:

1. Imports the required libraries: cv2 (OpenCV) for video capture and image processing, numpy for numerical computations, and openpyxl for writing the angle measurements to an Excel sheet.
2. Initializes the video capture device using the 'cap = cv2.VideoCapture(0)' statement, where '0' is the default camera index.
3. Defines the parameters for edge detection, line detection, and Excel workbook creation.
4. Creates a new Excel workbook and worksheet using the openpyxl library.
5. Starts an infinite loop that captures each frame from the webcam.
6. Converts each frame to grayscale for edge detection using the cvtColor function.
7. Performs edge detection on the grayscale image using the Canny function.
8. Detects lines in the image using the HoughLinesP method.
9. Draws the detected lines on the frame using the line function.
10. Calculates the angle between the two lines using the arctan2 function and displays it on the frame using the putText function.
11. Writes the angle measurement to the Excel worksheet using the cell method.
12. Checks if the 'q' key is pressed to exit the loop.
13. Saves the Excel workbook and releases the webcam capture.
14. Closes all OpenCV windows using the destroyAllWindows method.

This code can be modified to work with recorded videos or images by reading them instead of capturing them from the webcam, and by processing each frame in a loop.

1. Swing timing the time during which the foot is not in contact with the ground.

# Load the video

cap = cv2.VideoCapture('1.mp4')

# Set up the background subtractor

fgbg = cv2.createBackgroundSubtractorMOG2()

# Initialize variables

foot\_in\_air = False

swing\_start = 0

swing\_end = 0

# Loop through each frame in the video

while True:

    # Read a frame from the video

    ret, frame = cap.read()

    if not ret:

        break

    # Apply background subtraction to isolate the moving object (the person)

    fgmask = fgbg.apply(frame)

    # Apply a threshold to convert the image to binary

    thresh = cv2.threshold(fgmask, 127, 255, cv2.THRESH\_BINARY)[1]

    # Find contours in the thresholded image

    contours, hierarchy = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

    # Loop through each contour

    for contour in contours:

        # Find the bounding box of the contour

        x, y, w, h = cv2.boundingRect(contour)

        # Check if the contour is in the lower half of the frame (where the feet are)

        if y + h > frame.shape[0] / 2:

            # Check if the foot is in the air

            if not foot\_in\_air:

                # If the foot was on the ground and is now in the air, mark the start of the swing phase

                foot\_in\_air = True

                swing\_start = cap.get(cv2.CAP\_PROP\_POS\_MSEC)

        else:

            # If the foot is on the ground, mark the end of the swing phase and reset the foot\_in\_air variable

            if foot\_in\_air:

                foot\_in\_air = False

                swing\_end = cap.get(cv2.CAP\_PROP\_POS\_MSEC)

                # Calculate the swing time

                swing\_time = swing\_end - swing\_start

                print(f"Swing time: {swing\_time:.2f} ms")

    # Display the frame

    cv2.imshow('frame', frame)

    # Exit if the 'q' key is pressed

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release the video capture and destroy all windows

cap.release()

cv2.destroyAllWindows()

This code detects the swing time of a person walking in a video using OpenCV library in Python.

First, the code loads a video file using cv2.VideoCapture(). Then, it creates a background subtractor using cv2.createBackgroundSubtractorMOG2() to isolate the moving object (the person) from the background.

In the main loop, the code reads each frame of the video using cap.read(), applies background subtraction, and converts the resulting image to binary using cv2.threshold(). Then, it finds the contours in the binary image using cv2.findContours().

For each contour, the code checks if it is in the lower half of the frame (where the feet are). If so, it checks if the foot is in the air. If the foot was on the ground and is now in the air, the code marks the start of the swing phase by setting foot\_in\_air to True and records the current time using cap.get(cv2.CAP\_PROP\_POS\_MSEC). If the foot is on the ground, the code marks the end of the swing phase and resets foot\_in\_air. The swing time is then calculated by subtracting the start time from the end time.

The swing time is printed to the console using print(). The frame is displayed using cv2.imshow(), and the loop continues until the 'q' key is pressed.

Finally, the code releases the video capture and destroys all windows using cap.release() and cv2.destroyAllWindows()

1. **ANALYSIS**
   1. **. FEASIBILITY STUDY**.

Gait analysis using computer vision is a promising and practical method for detecting and monitoring gait-related disorders. Advanced technologies such as deep learning and convolutional neural networks (CNNs) have made it possible to analyze gait patterns and abnormalities accurately and efficiently in real-time.

Moreover, this approach is non-invasive, cost-effective, and easily applicable in clinical settings, eliminating the need for expensive and bulky equipment like force plates and motion capture systems, making it more accessible for healthcare providers and patients.

Furthermore, gait analysis using computer vision can provide objective and quantitative measurements of gait patterns and abnormalities, which can assist in the diagnosis, treatment, and monitoring of various neurological and musculoskeletal conditions.

However, there are still some challenges to overcome, such as standardizing data collection and analysis protocols and optimizing algorithms for different applications and settings. Despite these challenges, the feasibility and potential benefits of gait analysis using computer vision make it a promising approach for improving the management of gait-related disorders.

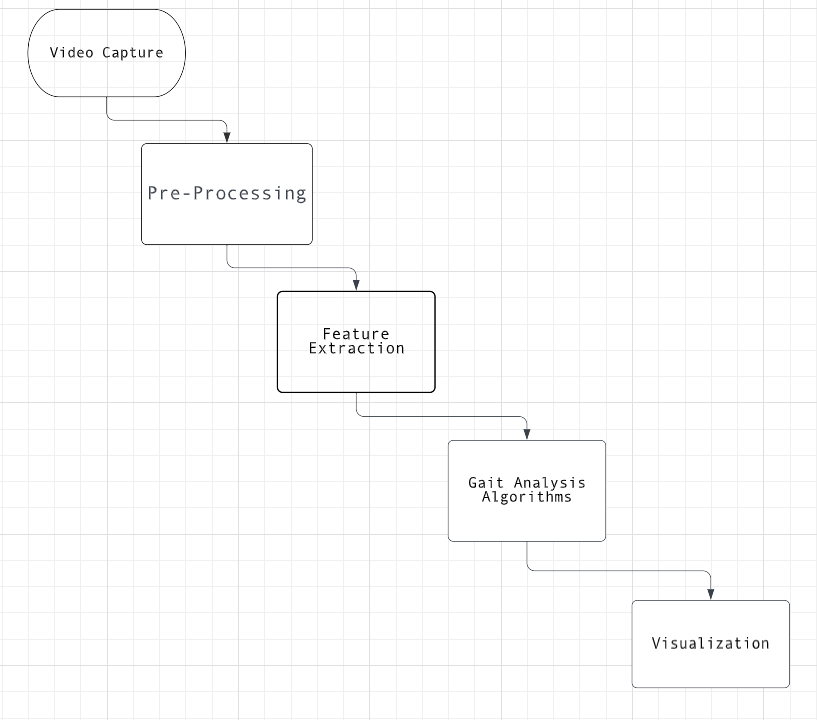
**ANALYSIS**

**7.2. TIMELINE CHART.**

|  |  |  |
| --- | --- | --- |
| Date | Work | Process |
| 20/3/2023-30/3/23 | Started Grab info on computer-Vision using Python | Completed |
| 1/4/2023-10/4/2023 | Started basic code for image processing using Python and OpenCV | Completed |
| 10/4/2023-25/4/2023 | Started getting info how to use camera in OpenCV and get live plotting | Completed |
| 25/4/2023-30/4/2023 | Started for finding angles that will be saved to Excel sheet (which will be helpful for finding walking abnormalities) | Completed |

1. **DESIGN**

**BLOCK DIAGRAM**



The video capture block captures the video data of a person walking, which is then preprocessed to improve the quality and reduce the noise in the data. The preprocessed data is then analyzed using feature extraction techniques to identify relevant features of the gait, such as step length and walking speed.

The gait analysis algorithms block then uses these features to identify any abnormalities in the gait, such as limping or irregular step patterns. The results of the analysis are then visualized and reported for further evaluation and diagnosis by healthcare providers.

**DESIGN**

**STIMULATION SCREENSHOTS**.



Fig 1.0:-Joint Detection



Fig 1.1:- Detection of angle of joint

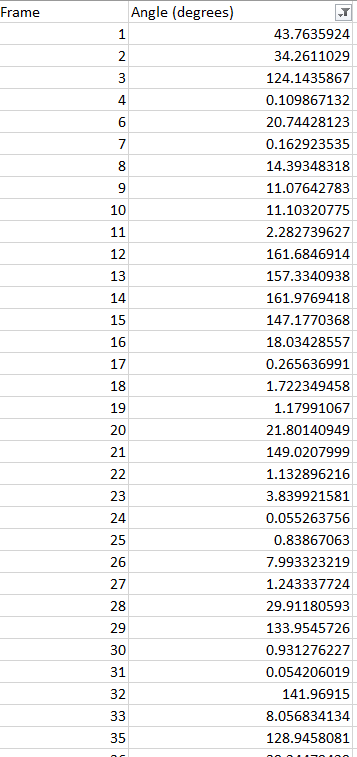


Fig 1.2 The data obtained from the live camera.

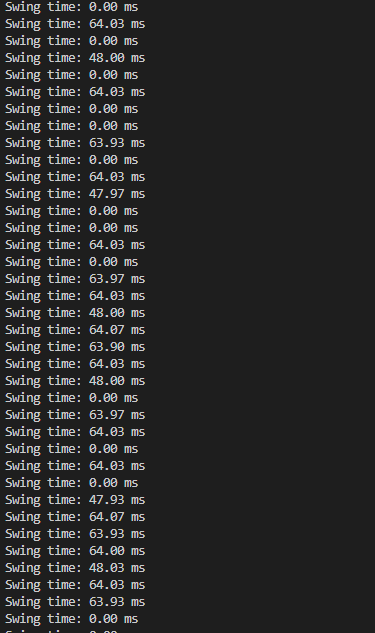


Fig 1.3:-Swing Time of the individual

1. **HARDWARE AND SOFTWARE REQUIREMENT.**

Hardware Requirements:

* A computer with a minimum of 4 GB RAM and 2 GHz processor
* A webcam or camera for capturing video
* A stable and fast internet connection (if using cloud-based services)

Software Requirements:

* Operating system: Windows, Linux or macOS
* Python programming language (version 3.6 or higher)
* OpenCV (Open-Source Computer Vision Library) for image and video processing
* Media pipe library for pose estimation
* Pandas' library for data management and analysis
* Xlsxwriter library for exporting data to Excel sheets
* IDE (Integrated Development Environment) like PyCharm or Jupyter Notebook for coding and testing the application.

1. **IMPLEMENTATION PLAN FOR NEXT SEMESTER**

**11**. **REFERENCE**

[1]. Auvinet, E., Multon, F., Stasse, O., & Meunier, J. (2008). Gait analysis using shoe-mounted inertial sensors and a referencing unit. Computer Methods and Programs in Biomedicine, 91(1), 47-56.

[2]. Clark, R. A., Pua, Y. H., Fortin, K., Ritchie, C., Webster, K. E., & Denehy, L. (2012). Validity of the Microsoft Kinect for assessment of postural control. Gait & posture, 36(3), 372-377.

[3]. El-Gohary, M., Pearson, S., McNames, J., Mancini, M., Horak, F., & Mellone, S. (2013). Continuous monitoring of turning in patients with movement disability. Sensors, 13(3), 3569-3586.

[4]. Gabel, C., & Smith, A. (2005). An integrated approach to gait analysis: a prosaic proposal. Medical engineering & physics, 27(2), 171-178.

[5]. Kobsar, D., Ferber, R., & Borysiuk, Z. (2014). Evaluation of markerless motion capture accuracy in biomechanical research. Medical engineering & physics, 36(7), 922-927.

[6]. Lin, Y. C., & Lai, J. S. (2017). A novel approach for gait analysis using Microsoft Kinect v2. Sensors, 17(10), 2276.

[7]. Moe-Nilssen, R. (1998). Test-retest reliability of trunk accelerometry during standing and walking. Archives of physical medicine and rehabilitation, 79(11), 1377-1385.

[8]. Morris, M. E., Huxham, F., & McGinley, J. (2001). Gait disorders and their significance in Parkinson's disease. Movement disorders, 16(1), 30-39.

[9]. Narayanan, M. R., Kavanagh, J. J., & Begg, R. K. (2014). On the use of body-worn sensors for gait recognition. Sensors, 14(4), 7124-7148.

[10]. Park, E. J., Kim, J. H., & Seo, J. H. (2016). A study on the gait analysis system using kinect sensor. Journal of Physical Therapy Science, 28(5), 1569-1573.