MARKER-LESS MOTION CAPTURE SYSTEM USING OPENPOSE

by

Bin Feng

A Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

Major: Electrical and Computer Engineering

The University of Memphis
December 2021
Acknowledgments

I would like to thank Drs. Ana Doblas and Douglas Powell for being my mentors and providing guidance throughout the whole project. I am also very thankful for having the opportunity to work in the Optical and Imaging Research Laboratory (OIRL). In addition, I would like to thank Dr. Douglas Powell for allowing me access to his lab to record data. I would also like to thank Dr. Ana Doblas, Dr. Douglas Powell, Dr. Aaron Robinson, and Dr. Madhusudhanan Balasubramanian for taking the time to serve on my committee. Next, I would like to thank Brian Bogue for giving valuable advice when I faced challenges. Last but not least, I want to give thanks to my family. I can achieve this much only because of their unlimited support throughout my whole life.
Abstract

Motion capture systems are widely used for measuring athletic performance and as a diagnostic tool in sports medicine. Standard motion capture systems record body movement using: (1) a set of cameras to localize body segments; or (2) specialized suits in which inertial measurement units are directly attached to body segments. Major drawbacks of these systems are limited portability, affordability, and accessibility. This contribution presents a markerless motion capture system using a commercially available sports camera and the OpenPose human pose estimation algorithm. We have validated the proposed markerless system by analyzing the human biometrics during running and jumping movements. The findings of this study demonstrate that pairing a low-cost sports camera with artificial intelligence allows for high-quality analysis of human movement.
# Table of Contents

*List of Figures* ......................................................................................................................... v

*Abbreviations* .......................................................................................................................... vi

**Chapter I: Introduction** .............................................................................................................. 1

1.a Topic Introduction .................................................................................................................... 1

1.b Thesis research overview ....................................................................................................... 1

1.c Objectives/Aims ..................................................................................................................... 3

1.d Benefits/Impact ...................................................................................................................... 3

**Chapter II: Background** ............................................................................................................ 4

2.a Motion Capture System ......................................................................................................... 4

2.b Biomechanics ......................................................................................................................... 5

**Chapter III: Methodology** ........................................................................................................ 10

3.a Experimental Equipment ....................................................................................................... 10

3.b Experimental Protocol .......................................................................................................... 11

3.c Data Processing .................................................................................................................... 12

   i). OpenPose Pose Estimation ................................................................................................. 12

   ii). Angles datasets ................................................................................................................ 15

   iii). 2D video to 3D skeleton Conversion ............................................................................... 20

   iv). Time-series classification with dynamic time wrapping .................................................. 21

3.d Analysis of results ................................................................................................................. 23

   i). Mean absolute error (MAE) ............................................................................................. 23

   ii). Root Mean squared error (RMSE) ................................................................................ 24

   iii). Confusion matrix ........................................................................................................... 24

**Chapter IV: Results** .................................................................................................................. 26

4.a Running analysis results ....................................................................................................... 26

4.b Jumping analysis results ...................................................................................................... 32

**Chapter V: Conclusions and Future Work** ............................................................................. 35

*Reference* .................................................................................................................................. 37

*Appendix* ................................................................................................................................... 39

*Python Code:* ............................................................................................................................ 39

   a. Aim1 ........................................................................................................................................ 39

   b. Aim2 ......................................................................................................................................... 51
List of Figures

Figure 1. Biomechanics of a normal running cycle (taken from reference below) .................. 7
Figure 2. Movement of the lower limbs during running .......................................................... 7
Figure 3. Lower limbs movement during jumping .................................................................... 9
Figure 4. Keypoints Detected by OpenPose Algorithm (Cao, 2017) ......................................... 13
Figure 5. Network Architecture of OpenPose Algorithm (taken from Ref below) ................. 14
Figure 6. OpenPose algorithm output examples (left is the accurate example; right is the inaccurate example) .................................................................................................................. 15
Figure 7. Demonstration of lower limb key points estimated by the OpenPose algorithm. The blue lines are related to the left leg and the green-cyan lines denote the right leg ............. 16
Figure 8. Experimental angles for the right knee (top row) and right angle (bottom row) (a) with and (b) without applying the low-pass Savitzky-Golay filter ............................................. 18
Figure 9. Illustration of the angle ratio plot (a) before and (b) after downsampling the experimental data ..................................................................................................................................... 19
Figure 10. Demonstration of lower limb key points (taken from Reference below) .............. 20
Figure 11 Dynamic Time Wrapping Vs Euclidean Distance .................................................... 22
Figure 12. Illustration of the performance of a KNN Classification method to discriminate good (yellow circles) and bad (black triangle) jumping. The performance of the method depends on the k parameters, which is related to the size of the circle around the target data. 23
Figure 13. Right knee angle plot (left is the result from the golden system; right is the result from our system) ......................................................................................................................... 26
Figure 14. Right knee angle plot (left is the result from the golden system using 2D gait data; right is the result from our system) .......................................................................................................................... 27
Figure 15. Right ankle angle plot (left is the result from the golden system; right is the result from our system) ......................................................................................................................... 29
Figure 16. Examples of bad detection on big toes ................................................................. 30
Figure 17. Generated 3D skeleton animation from the single video ....................................... 31
Figure 18. Generated 3D skeleton animation from the single video ....................................... 32
Abbreviations

ACL: Anterior Cruciate Ligament
CMJ: Countermovement Jump
CNN: Convolutional neural network
CSV: Comma-separated Values
DTW: Dynamic Time Wrapping
FPS: Frames Per Second
GAST-Net: Graph Attention Spatio-temporal Convolutional Nets
GCN: Graph Convolution Networks
JSON: JavaScript Object Notation
KNN: K-nearest Neighbor
LED: Light-emitting Diode
MAE: Mean Absolute Error
MAL: Musculoskeletal Analysis Laboratory
OIRL: Optical and Imaging Research Laboratory
PAFs: Part Affinity Fields
RGB: Red Green Blue
RMSE: Root Mean Square Error
VFX: Visual Effects
2D: Two Dimensions or Two-Dimensional
3D: Three Dimensions or Three-Dimensional
Chapter I: Introduction

1.a Topic Introduction

Gait analysis is a systematic study concentrating on human motion using the observation and measurement of body movements, body mechanics, and muscle activities. There are many different formations of gait analysis systems in the current market, generally categorized as marker-based and marker-less systems. So far, the most common gait analysis systems are marker-based, which means the system will require markers to be attached to users’ bodies to capture the movements. However, there are some disadvantages for marker-based gait analysis systems compared to marker-less systems. For instance, the marker-based gait analysis systems usually require a long preparation period to place the markers. Marker-less systems do not require any markers during movement, so the preparation period can be significantly shorter compared to marker-based systems. In addition, since the marker-less systems do not require users to have any external devices on their bodies, the naturalness of body movements can be captured and analyzed. There are many advantages for marker-less systems, but one of the main barriers for the marker-less system to become more commercialized is accuracy. The marker-based systems typically have good accuracy of the data analysis results compared to marker-less systems. Therefore, the main goal for many current research teams is to increase the accuracy of the current marker-less gait analysis system to a comparable level with marker-based systems. Especially with the rapid growth of machine learning technology, the needs and research interests of the marker-less gait analysis system have also increased.

1.b Thesis research overview

This Master thesis research aims to provide a potential approach for a marker-less motion capture system using the OpenPose human pose estimation algorithm (Cao, 2017). The project
contains two main phases, which are the data collection phase and the data analysis phase. For the data collection phase, a GoPro camera is used to capture videos of running and jumping of participants. After the videos are captured, they will be extracted into a certain number of frames based on the frame rate of the videos. The two frame rates used during video recording are 60 FPS and 120 FPS. The frames are then fed to the OpenPose algorithm to estimate the key points of the body joints, such as hip, knees, ankles, etc.

This Master thesis research will focus on the movement of the lower limbs, including the key points: the hips, knees, ankles, and toes. Once the locations of key points are found, they will be used for knee angle and ankle angle calculation based on triangle mathematical rules. Since there are two different types of movement (running and jumping), this thesis project provides two different approaches for data analysis. For running data, the main goal is to compare the accuracy of the OpenPose based marker-less system with the Qualisys marker-based system. Therefore, the data analysis step will require the running data from two systems to be cleaned, pre-processed, and aligned. The matrices used for difference measurements are mean absolute error (MAE), and root mean squared error (RMSE). On the other hand, the main goal for jumping data analysis is to develop a classification algorithm that can differentiate the bad jump styles from normal jump styles only using the side view of the jumping movements. Similarly, all jump data also needs to be cleaned, pre-processed, and aligned, so we will have a collection of jump cycles with different knee placements, but with the same amount of data points in one jump. The classification algorithm used for jump data is a K-nearest neighbors (KNN) classifier with Dynamic Time Warping (DTW). The results evaluation is measured by the Confusion Matrix.
1.c Objectives/Aims

Aim #1: Investigation of the performance of the proposed marker-less system to the marker-based motion capture system, Qualisys, for kinematical data

Hypothesis #1: We predict that our marker-less motion capture system can achieve similar gait analysis results as the marker-based system using only one camera.

Aim #2: To develop a classification model to determine if a jump cycle is a normal jump or an abnormal jump with a bad knee position using kinematic data from a marker-less motion capture system

Hypothesis #2: We predict that we can develop a classification model for different normal jumps or abnormal jumps with bad knee positions based on kinematic data.

1.d Benefits/Impact

A particular application of our motion capture system is in sports medicine. We predict that our system will benefit multiple professional settings, improving overall an individual’s well-being. For example, physical therapists could track their patient’s recovery process and compare the data against an expected rate. Researchers could work to develop more realistically moving prostheses for amputees. High-performance athletic trainers could use this system to help analyze their player’s movements and increase the efficiency of their training regimen. Importantly, this system will be relatively inexpensive for the average professional. Finally, since the user would only need a single all-in-one package to analyze biometric data, we predict that this system will be cost-effective.
Chapter II: Background

2.a Motion Capture System

A motion capture system is a type of system that can record the movement of a person or an object. Motion capture was originally used for the movement analysis and studies of living animals. But, it is now adopted in many different fields, such as VFX studios, sports therapists, neuroscientists, and computer vision and robotics (Baker, 2020). In the past decades, motion capture systems have been heavily investigated and developed. The common systems for motion capture systems are magnetic, mechanical, and optical systems (Baker, 2020). Optical systems can be categorized as passive, active, or marker-less systems (Baker, 2020). For this thesis, we will focus on the optical system.

The application of optical motion capture systems has become increasingly widespread in other fields, such as entertainment, biomechanics, and sports sciences (Nagymate, 1970). As we mentioned earlier, optical motion capture systems have three major categories: passive, active, and marker-less systems. The difference between those three systems is the usage of markers during the motion capture process. A passive system uses the makers coated with reflective material, so light can be reflected and captured by the camera lens. The active system uses either LEDs or sensors to locate the specific body parts location, so the location is recorded in the software system. Last but not least, the marker-less motion capture systems do not use any attachment or wearables to assist in locating a human’s pose, but complex information processing technology is required to recognize human poses from images (Nakano, 2020).

There are advantages and disadvantages for motion capture systems, either with or without markers. One of the main advantages of motion capture systems with markers is the high accuracy. There was a study that has been done to examine the accuracy of the optical motion
capture system. The research team measured the accuracy of the OptiTrack motion capture system in a large capture volume (>100m³) (Aurand, 2017). They discovered that the error was smaller than 200 mm in 97% of the capture area when all 41 cameras were used (Aurand, 2017). Even with only half of the cameras, 91% of the capture area can maintain error below 200 mm (Aurand, 2017). This accuracy is sufficient to measure full-body human kinematics with skin-mounted markers in a large capture volume (Aurand, 2017). On the other hand, it is hard for a marker-less system to achieve the same level of accuracy for multiple reasons. In 2010, Microsoft released the Kinect sensor, which is used for capturing human pose using RGB and depth images (Desmarais, 2021). However, this type of sensor is still used mainly for entertainment applications and is not well-suited for outdoor usage (Desmarais, 2021). Therefore, most commercial motion capture systems need to rely on reflective markers that are placed on users’ bodies to track the movements with high accuracy (Desmarais, 2021). Meanwhile, there are advantages of marker-less motion capture systems compared to marker-based systems. For instance, marker-based motion capture systems require users to equip wearables or attachable markers, and that process usually takes a long time to set up. On the other hand, a marker-less motion capture system can significantly reduce the preparation time before recording since there are no markers needed (Ceseracciu, 2014). In addition, without the markers, the naturalness of a subject’s movement cannot be modified (Ceseracciu, 2014).

2. b Biomechanics

Biomechanics is defined as the study of the movement of living things using the science of mechanics being the latter the branch of Physics that explains motion and how forces create motion (Arus, 2018). It is challenging to describe human motion in a precise way. However, biomechanics provides conceptual and mathematical tools for researchers to understand and
study how living things move (Arus, 2018). For example, biomechanics allows us to study human movement performance and enhance it based on the knowledge of biomechanics (Arus, 2018). The biomechanics of running have been worldwide studied in such a way that now coaches can improve a runner’s technique by matching its profile to the one from athletes (Arus, 2018). Another example of biomechanics application is injury prevention and treatment (Arus, 2018). Many sports medicine professionals study and analyze injury data based on the foundation of biomechanics to determine the potential causes of disease or injury (Arus, 2018).

On the other hand, biomechanics can also help the physical therapist, which rehabilitative exercises, assistive devices, or orthotics are recommended for enhancing movement performance (Arus, 2018). During the body injuries recovery process, many therapists perform a qualitative analysis of a patient’s gait to determine whether the patients’ muscular strength and control are sufficient for safe or cosmetically normal walking (Arus, 2018). In this Master's thesis, we are focusing on investigating a low-cost marker-less motion capture system to understand the biomechanics of running and vertical jumping.

As we know, running is one of the most popular and everyday exercise. Although many people run for exercise, especially over the last decade, an improper technique can lead to muscle and joint injuries. Therefore, it is crucial to study the biomechanics of running and determine potential human movements that increase performance and reduce the likelihood of common injuries. A running gait cycle is divided into a stance and swing phase. Whereas the stance phase is further divided into absorption and propulsion phases, the swing phase is divided into initial and terminal swing phases (Thordarson, 1997). Since it exists the period called “double float” (i.e., time in which both feet are off the ground during the beginning and end of each swing phase), the stance phase accounts for less than 50% of the whole gait cycle to allow
the airborne period (Thordarson, 1997). When the runner increases the velocity, the stance phase reduces while both the swing phase and double float time increase (Thordarson, 1997). **Figure 1** shows the summarization of a normal running cycle.

![Figure 1](image1.png)

**Figure 1. Biomechanics of a normal running cycle (taken from Arus, 2018)**

**Figure 2** represents the runner’s body movement in a full running cycle. During running, the runner’s joint repeats movements periodically, being the temporal period dependent on the runner’s velocity. For example, the knee and ankle always flex during the absorption phase of
stance (Thordarson, 1997). At heel strike, the ankle dorsiflexes rapidly while flexion occurs to the knee and the hip (Thordarson, 1997). At the mid-stance phase, both knee and ankle motion reverse and begin to extend, leading to the propulsion phase (Thordarson, 1997). Once toe-off occurs, the runner’s body prepares to go into the double float phase (Thordarson, 1997). The hip, knee, and ankle continue to extend until the feet leave the ground (Thordarson, 1997). During the double float phase, the hip, knee, and ankle are at their maximal extension (Thordarson, 1997).

The second movement this thesis focuses on is vertical jumping. Vertical jumping is not as popular as running for an exercise routine, but it is commonly combined with other movements in many different sports and exercises. For example, basketball players jump up to 50 times in one game (Pliauga, 2015). Research around vertical jump biomechanics can be easily found as many athletics are finding ways to improve their performance and avoid injuries. For instance, anterior cruciate ligament (ACL) injury is a common injury where the tissue that connects the thigh bone to the shinbone at the knee gets torn. Over 120,000 people suffer an ACL injury every year in the United States, and most ACL injuries that occur during certain sports require rapid body movements (Bates, 2013). In basketball, the most common mechanism of ACL rupture is rebounding, which involves a rapid and unstable vertical jump (Bates, 2013). Jump landings create a sudden ground reaction force that can translate into large external torques at the knee (Bates, 2013). By performing several rebounding techniques in each game for many years, many basketball players suffer knee injuries. Studying the biomechanics of vertical jump, especially landing, can be essential for sport scientists to understand the correct form of vertical jump and provide a good technique to minimize the injury risk.
In a full jump cycle, the first part is the takeoff phase, in which the person flexes their lower limb muscles for jump preparation. One of the most common vertical jump preparation techniques is the countermovement jump (CMJ) (Linthorne, 2001). At the beginning of CMJ, the jumper drops the arms, flexes at the hip, knee, and ankle to create a small downward movement (Linthorne, 2001). Then, a rapid extension of the legs and an upward swing of the arms occur at takeoff (Linthorne, 2001). Once the body is in the air, the knee and ankle extend to their maximum. The second part of the jump cycle is landing. Many studies have shown a relationship between ACL injuries and abnormal landing techniques (Wong, 2020). By studying the biomechanics of vertical jump landing, new landing techniques have been developed to reduce the risk of knee injuries, such as ACL injury (Wong, 2020). The research team discovered that when female athletes are under muscle fatigue conditions, the risk of ACL injury can be reduced if they can appropriately increase their knee flexion angle during landing (Wong, 2020).
Chapter III: Methodology

3. Experimental Equipment

The running data was recorded by two different motion capture systems. The first system, the “golden system”, is the Qualisys AB (240 Hz, Qualisys Inc., Goteburg, Sweden) system which is a marker-based motion capture system using 8 cameras. 14 Retro-reflective markers were attached on runner’s lower limb joints areas: the hips, the knees, the ankles, and the foot. Qualisys Track Manager (QTM), which is the software provided by Qualisys Inc., generates precise body kinematic information using the skeletal data captured by the system.

Our motion capture system uses only one video camera to record all needed data. For running data, the camera we used is GoPro HERO8 (120 Hz, GoPro Inc., California, the United States). For jumping data, we use the Akaso EK7000 action camera (60Hz, Akaso, Washington DC, the United States) to record.

Once the data recording step starts, both the golden system and our system simultaneously record the participant running on a treadmill. The treadmill used in the experiment is the Bertec instrumented treadmills (Bertec Corporation, Ohio, the United States).
3. b Experimental Protocol

The running data is collected at the Musculoskeletal Analysis Laboratory (MAL). Dr. Douglas Powell is one of the principal investigators of the MAL at the University of Memphis. First, the participant arrived at the MAL at the University of Memphis to prepare the experiment. Since Qualisys motion capture system is a marker-based system, the participant was required to have the markers attached to his/her body. In this experiment, the markers were used for the lower limb joints, including the right and left hips, the right and left knees, the right and left ankles, and the left and right toes. Foil tapes were used to secure the markers on the participant’s body during running. Before the recording session began, the participant warmed up their body for a few minutes until they were ready to run. Once the recording process began, the participant was required to run on a treadmill for 10 minutes. During the 10 minutes of running, the treadmill stayed at a constant speed of 3 miles per hour. One GoPro camera kept recording on the right side of the treadmill while the eight cameras of Qualisys motion capture systems were recording. The recording rate was 120 FPS, and the resolution was 1920 by 1440 pixels.

The vertical jumping datasets were captured in the Musculoskeletal Analysis Laboratory and a courtyard. The participant first arrived at the lab before the recording session to warm up their body. While the participant was getting ready to start, the camera was set up in one open area (either an open space in the lab or the courtyard). Once the recording session started, the participant was required to stand in front of the camera and adjust the position according to the captured frame of the camera. After we made sure that the camera could always capture the full body of the participant, the jump recording session started. The participant was required to jump in different styles with normal and abnormal knee positions. In this experiment, we have collected a total of 46 jump cycles, including 25 normal jumps, 12 abnormal jumps with inner
knee position, and 9 abnormal jumps with outer knee position. The camera we used for jumping datasets is the Akaso EK7000 action camera. The recording rate is 60 FPS, and the resolution is 1920 by 1080 pixels.

There was a total of five participants recruited for this study, three males and one female, with an age range between 25 to 30. All participants for this experiment were physically healthy and did not have recent lower limb injuries.

3.c Data Processing

i). OpenPose Pose Estimation

With the growing need for computer vision and machine learning applications, many 2D human pose estimation libraries are built. However, many current available 2D body pose estimation libraries require their users to implement most of the pipeline (Cao, 2017). Those libraries are normally limited to specific operating systems and hardware setups (Cao, 2017). Compared to other available libraries, the OpenPose algorithm can run on different platforms and provide support for different hardware setups (Cao, 2017). OpenPose algorithm can also use input from different resources, including images, videos, webcam, and IP camera streaming (Cao, 2017). Besides being user-friendly, the OpenPose algorithm outperforms all state-of-the-art methods while preserving high-accuracy results (Cao, 2017). To quantify the comparison, the research team compare the OpenPose algorithm with other state-of-the-art bottom-up approaches by measuring the mean Average Precision (mAP) of all body parts. From the results on the MPII multi-person dataset, the mAP obtained from the OpenPose model outperforms other approaches by 8.5%. Meanwhile, the inference time is 6 orders of magnitude less compared to other approaches.
OpenPose algorithm detects the 2D pose of multiple people in real-time imaging (Cao, 2017). As **Figure 4** shown, the OpenPose algorithm can detect up to 25 key points on the human body, including the eyes, ears, mouth, and joints on the arms, shoulders, legs, foot, among many other key points (Cao, 2017). To visualize the outcome, the OpenPose algorithm plots a dot on the picture for each key point identified as a key body part. One of the advantages of the OpenPose ML algorithm is that the users can select a broad number of key points, making it suitable for multiple applications. For example, in our particular study, we have chosen the lower limb key points to perform deeper analysis and study.
OpenPose algorithm uses a multi-stages convolution Neural network (CNN) to give an accurate pose estimation based on input images. As Figure 5 shown, there are two stages in the network architecture. The first stage provides a prediction on Part Affinity Fields (PAF), and the second stage predicts confidence maps. In the original model, the network architecture included several 7×7 convolutional layers (Cao, 2017). The new approach modifies the network architecture to include 3 layers of convolutions of kernel 3×3 and the output of each layer is concatenated (Cao, 2017). Therefore, the network can keep both lower-level and higher-level features (Cao, 2017). The OpenPose model is trained using two different datasets: 1) MPII multi-person Dataset, which consists of 3,844 training and 1758 testing groups of multiple interacting individuals with 14 body parts; and 2) COCO keypoint challenge dataset, which consists of over 100K person instances labeled with over 1 million key points.

In the OpenPose algorithm, the accuracy of foot detection is not consistent. As Figure 6 shown, the images represent two examples of the image output from OpenPose. The left image represents the example of accurate estimation of the body key points in the image. The right image represents the example of inaccurate estimation of the body key points in the image. As
we can see from the right image, the left foot of the runner is on the treadmill, but the OpenPose
algorithm did not detect the correct location of the left foot since it is blocked by the right foot.
There are similar errors in other image frames from OpenPose output. Therefore, to avoid the
error, we will focus on the side in the front. For instance, we will use the right-side limb for the
below images to obtain more accurate results.

Figure 6. OpenPose algorithm output examples (left is the accurate example; right is the
inaccurate example)

ii). Angles datasets

We have used the OpenPose algorithm to detect human body key points in a set of time-
series images that captures the running or jumping movements of the participant. In this step, we
are tracking the gait of the lower limb (a total of 8 key points, including the left/right hip, the
left/right knee, the left/right ankle, and the left/right big toe) during running or jumping. As
shown in Figure 7, we calculate the knee angles and the ankle angles for each image in the
image set.
Figure 7. Demonstration of lower limb key points estimated by the OpenPose algorithm.

The blue lines are related to the left leg, and the green-cyan lines denote the right leg.

The knee angles include the right knee angle and the left knee angle. The right/ left knee angle was calculated by the right/left hip, the right/left knee, the right/left ankle location. Assume the hip, knee, ankle locations are \((x_1, y_1), (x_2, y_2), (x_3, y_3)\). We found the distance between those three key points as \(d_1, d_2, \) and \(d_3\) using the distance formula (Lubis, 2020).

\[
\begin{align*}
    d_1 &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}, \\
    d_2 &= \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2}, \\
    d_3 &= \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2},
\end{align*}
\]

(1) \hspace{1cm} (2) \hspace{1cm} (3)

After finding the distances of the three points, we can use the cosine rule to calculate the knee angle.

\[
\theta_{knee} = \cos^{-1}\left(\frac{d_1^2 + d_2^2 - d_3^2}{2 \cdot d_1 \cdot d_2}\right),
\]

(4)
Similarly, we can calculate the right/left ankle angles by using the location of the right/left knee, the right/left ankle, the right/left-right big toe. Assume the big toe location is \((x_4, y_4)\). We can calculate the ankle angles using the following formulas.

\[
d_4 = \sqrt{(x_3 - x_4)^2 + (y_3 - y_4)^2}, \quad (5)
\]

\[
d_5 = \sqrt{(x_2 - x_4)^2 + (y_2 - y_4)^2}, \quad (6)
\]

\[
\theta_{\text{ankle}} = \cos^{-1}\left(\frac{d_2^2 + d_4^2 - d_5^2}{2 \cdot d_2 \cdot d_4}\right), \quad (7)
\]

After the knee and ankle angles for each frame are being generated, they are stored in CSV files based on the jump and running cycles. The data stored in CSV files need to be cleaned and pre-processed. The clean process consists of two steps. The first step is performed manually. Since one CSV file represents all collected information from a running or jumping video and each video contains multiple running or jumping cycles, we need to find the start and end time points for each gait cycle. The second cleaning step is applying a low-pass filter to the angle’s information. Since the results from the OpenPose estimation algorithm still contain errors, the angle calculations contain noise as well. Applying a low-pass filter can remove the outliers and present a more precise result. In this Master thesis, we have applied a Savitzky-Golay filter to the knee angle data from our system to remove the outliers. Savitzky-Golay filter applies to a set of discrete data, calculates a polynomial fit with a given order, and creates a smoother curve for the dataset (Luo, 2005). As Figure 8 shown, the two panels on the left represent the knee angle plot and ankle angle plot without the Savitzky-Golay filter, while the right two panels show the same information after applying the Savitzky-Golay filter. Compared to the plots without filters, the
plots with filters have removed many noises and lost some details. For instance, in the top-right plot, the troughs between each cycle are not as low compared to the top-left plot.

![Graphs showing knee angle before and after applying filters](image)

**Figure 8.** Experimental angles for the right knee (top row) and right angle (bottom row) (a) with and (b) without applying the low-pass Savitzky-Golay filter

The Qualisys motion capture system has a recording frequency of 240 Hz, which means there are 240 data points per second in the golden system results. However, the camera we used for running data collection is a GoPro camera with 120 frames per second (FPS) rate. The camera used for jumping data collection is the AKASO camera, which is a 60 FPS rate. To accurately compare our running analysis results with the golden system, we need to resample the data points in the golden system to 120 FPS.

For jumping data, the pre-processing step is also necessary. The camera used for all jump cycles is the same one, so the recording rate is constant. However, the number of data points contained in one jump cycle sample can vary based on the jump height. It is challenging to have the participant jump at the same height every time, so the numbers of data points in different
jump cycles are different from each other. To build a classification algorithm for time-series data, we need to ensure each sample contains the same number of time-series points. Therefore, the pre-processing step for the jumping datasets is finding the beginning and end of each jump cycle and resampling the number of data points to a fixed amount for each cycle. After the pre-processing step, each jump cycle contains 135 time-series data points, and five features have been recorded for each data point. Like the running datasets, our system also calculated the left/right knee angle and the left/right ankle angle for the jumping datasets. In addition, to consider both knee and ankle movements, we have developed a new engineered feature: the angle ratio. The left/right angle ratio is calculated by the left/right knee angle divided by the left/right ankle angle. Figure 9 represents the angle ratio changing in a complete jump cycle. The left panels show the angle ratio plot with 155 data points, and the right panels show the angle ratio plot for the same jump cycle, but after resampled to contain 135 data points. As we can see from Figure 9, the left plot and the right plot have the same shape with the only difference on the x-axis scale.

![Figure 9. Illustration of the angle ratio measurement (a) before and (b) after downsampling the experimental data](image)

Figure 9. Illustration of the angle ratio measurement (a) before and (b) after downsampling the experimental data
iii). **2D video to 3D skeleton Conversion**

One of the goals for running data analysis is building 3D skeleton animation using the recorded 2D gait videos. This project aims to perform gait analysis using only one camera, so we cannot use the traditional 3D reconstruction approach since it requires multiple image views for reconstructing the 3D view (Liu, 2020). Therefore, we have used the graph attention Spatio-temporal convolutional nets (GAST-Net) algorithm that can provide depth information using only one image view (Liu, 2020).

![Graph Attention Spatio-temporal Framework](image)

**Figure 10. Demonstration of lower limb key points (taken from Liu, 2020)**

Figure 10 demonstrates the pipeline of the GAST-Net model. The GAST-Net model takes 2D videos as inputs that contain the human body in the image frames and then generates a 2D pose estimation for body key points in the image frames (Liu, 2020). The 2D pose estimation in the GAST-Net model is generated by the HRNet Human Pose Estimation model (Liu, 2020). The model can achieve similar accuracy as the OpenPose algorithm. Based on the accuracy-test on the COCO dataset, the model can achieve 79% accuracy on human pose estimation (Liu, 2020). Thus, we used the pre-trained model from HRNet, instead of using OpenPose algorithm results for simplicity of the process (Liu, 2020). Next, the 2D key point sequences are fed to a temporal convolutional model. The temporal convolutional model takes 2D key points sequences
as input, and the output is the 3D pose estimation. The estimated 3D pose includes two graphs which are the local attention graph and global attention graph. The local attention graph is used for visualizing joints, including local kinematic dependencies and symmetric relations. The global attention graph includes the information for posture semantics (Liu, 2020). Last, the local and global attention mechanisms were combined to generate the 3D skeleton models (Liu, 2020).

The GAST-Net model is trained by two datasets which are Human3.6M and HumanEva-I (Liu, 2020). Human3.6M dataset contains about 3.6 million video frames with 11 professional subjects performing 15 daily activities (Liu, 2020). HumanEva-I dataset contains 7 calibrated video sequences with 3 subjects performing 6 common actions. To qualify the performance of the model, the research team calculates the mean per joint positioning error (MPJPE) between the predicted 3D coordinates results and the ground truth. Based on the calculated MPJPE results on Human3.6M dataset, the GAST-Net model shows competitive performance compared to state-of-the-art results.

iv). Time-series classification with dynamic time wrapping

The number of data points in one jump cycle is different from each other and depends on the maximum jump height of that jump cycle. It is impossible to control the jump height of each cycle during the recording phase, so our datasets need to be pre-processed before any model training. As we mentioned in section 3.4.c, all datasets for different jump cycles are resampled to contain exactly 135 data points. Once we make sure that our datasets are uniform, we can move to the model training phase.

The jump datasets are time-series data, and it is difficult to determine the exact start and end of a jump cycle. Thus, the different jump cycles will have a small misalignment. If we
calculate the difference between two jump cycles using Euclidean distance, we will have inaccurate results because of misalignment. Unlike traditional classification algorithms, our system will calculate the dynamic time wrapping (DTW) distance instead of the Euclidean distance. As Figure 11 shown, the Euclidean distance method calculates the distance one by one between two data sequences, but DTW aligns time-series data before calculation to generate a more precise result (Hsu, 2015).

![Dynamic time wrapping vs Euclidean Distance](image)

**Figure 11 Dynamic Time Wrapping versus Euclidean Distance**

In the jump dataset, we have five recorded features for each data point: 1) the left knee angle; 2) the right knee angle; 3) the left ankle angle; 4) the right ankle angle; 5) the left leg angle ratio. We have a total of 46 jump cycles in the dataset, so we used about 20% for testing (10 jump cycles) and 80% for training (36 jump cycles). We use the five features individually to calculate the DTW distance matrix between the training dataset and the testing datasets. Based on the calculated DTW distance for each testing jump cycle, the algorithm assigns the test data to a particular label based on the K nearest neighbors. Figure 12 demonstrates the basic idea of a KNN classification algorithm. In this thesis, we use a KNN algorithm to predict the testing jump cycles with $K = 2$ using a single feature as the input of the algorithm. We have tested other K
values, and the prediction results suggest that the best performance occurs when \( K = 2 \). Since this Master's thesis aims to differentiate normal jumps and abnormal jumps with bad knee position, I have classified all jump cycles to good and bad jumps, which is indicated by 1 or 0.

![Illustration of the performance of a KNN Classification method to discriminate good (yellow circles) and bad (black triangle) jumping. The performance of the method depends on the k parameters, which is related to the size of the circle around the target data.](image)

**Figure 12. Illustration of the performance of a KNN Classification method to discriminate good (yellow circles) and bad (black triangle) jumping. The performance of the method depends on the k parameters, which is related to the size of the circle around the target data.**

### 3.d Analysis of results

To quantify the results, metrics were used to calculate the accuracy of our system. The goal for the running analysis is to compare the difference between the results from our system and the results from Qualisys marker-based motion capture system.

i). Mean absolute error (MAE)

Mean absolute error is a common metric for error measurements. The two inputs for the calculation should have the same length and be aligned. MAE can be calculated with the below formula:
where $x_i$ and $y_i$ are two paired sequences and $n$ is the length of two sequences.

**ii). Root Mean squared error (RMSE)**

Similarly, root mean squared error (RMSE) needs the paired inputs for calculation as well. RMSE can be calculated based on the following formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - x_i)^2}{n}},$$

(9)

where $x_i$ and $y_i$ are two paired sequences and $n$ is the length of two sequences.

MAE and RMSE are both common in the statistics for error measurements. Compared to MAE metrics, RMSE metrics will square the error in the calculation, so it normally puts a bigger weight on the outliers with big errors. However, the RMSE can be a more appropriate error measurement method when the errors follow a normal distribution (Chai, 2014). To get a full understanding of the results generated by our system, we have reported both error measurement metrics.

**iii). Confusion matrix**

In aim 2 of this thesis project, the goal is to develop a classification algorithm to differentiate jump cycles based on the recorded kinematical information. A confusion matrix is commonly used for the performance measurement of a classification model. The confusion matrix can be demonstrated as **Table 1**:  

**Table 1. Confusion Matrix**
<table>
<thead>
<tr>
<th></th>
<th>Actually positive (1)</th>
<th>Actually negative (0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted positive (1)</td>
<td>True positives</td>
<td>False positives</td>
</tr>
<tr>
<td>Predicted negative (0)</td>
<td>False positives</td>
<td>True negatives</td>
</tr>
</tbody>
</table>

Besides using the confusion matrix, we have calculated the accuracy score for our classifier. The accuracy score is calculated by the following formula:

$$\text{Accuracy} (y_{\text{predict}}, y_{\text{label}}) = \frac{1}{N_{\text{samples}}} \sum_{i=0}^{N_{\text{samples}}-1} 1(y_{\text{predict}} = y_{\text{label}}). \quad (10)$$

Our classifier has two output which is normal or abnormal jump based on the knee position. Therefore, the accuracy scores are also equal to the sum of the true positive rate and the true negative rate.
Chapter IV: Results

4. a Running analysis results

The main goal of Aim 1 in this thesis project is to perform gait analysis on the running datasets and compare the results with results from the golden system, Qualisys marker-based motion capture system. Since the right side of the runner was captured during running, we only used the right-side limb data for comparison for better results. The kinematical information for the right-side limb includes the right knee angle and the right ankle angle.

![Knee Angle Measured by Golden System](image1)

![Right Knee Angle Measured by Our System](image2)

**Figure 13. Right knee angle plot (left is the result from the golden system; right is the result from our system)**

The first part of aim 1 is analyzing the knee angle results of two systems. First, we can look at the analysis result from the golden system for the knee angles. As the left panel in **Figure 13** shows, the plot for the knee angle shows a periodical change in four running cycles. There are two peaks in one jump cycle, and the angle value range is between -90 to -10 degrees. The figure on the right presents the knee angle changes calculated by our system for the same four running cycles. The knee angles values measured by our system are in the range of -80 and -10 degrees. There are two clear peaks in the right plot as well, but unlike the results from the golden system, the second peaks are not lower than the first peak in one running cycle. We have measured the
angle difference between these two maximum peaks to be found equal to 7.66 (mean) ± 1.56 (std) degrees for the golden system, and 1.28 ± 1.31 degrees for the proposed method. Based on these values, we conclude that the proposed method presents some difficulty tracking the difference between two knee flexions in a running cycle. The cause for the performance difference between the golden system and our system is the dimension difference between the two datasets. The golden system uses a 3D motion capture system, so the angle values are calculated based on the 3D coordinate values. However, our system uses a single camera, and the depth information of the knee is lost during recording. In addition, compared to the golden system, our system contains noise. By analyzing the error in the outcome from our system, we noticed that the noise was caused by the failure or incorrectly estimated body joints from OpenPose.

**Figure 14. Right knee angle plot (left is the result from the golden system using 2D gait data; right is the result from our system)**

The left plot in **Figure 14** shows the right knee angle plot generated from the golden system using 2D gait data while the right plot shows the knee angle plot from our system. After transferring the 3D gait data to 2D gait data, we noticed that the knee angle plot from the golden
system shows more similarity with our system results. For example, as the red line shown in the left plot, the two peaks in each running cycle have similar height levels as our system plot suggests. Therefore, we conclude that the accuracy of the knee angle values from our system is sufficient in a 2D perspective. If we want to improve the accuracy of our system to the next level, we will need to introduce a new dimension to our gait dataset.

To qualify the overall performance of the analysis results from our systems, we have measured the error using the golden system as the ground truth and calculated the mean absolute error (MAE) and mean squared error (MSE) for the results from our system. To include the potential effect of applying low-pass filters to our datasets, we include evaluation results for both filtered and non-filtered datasets.

**Table 2. Right Knee Angle Comparison between Our System and Golden System**

<table>
<thead>
<tr>
<th></th>
<th>Before Applying Filters (%)</th>
<th>After Applying Filters (%)</th>
<th>Difference between Applying and Not Applying Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MAE</strong></td>
<td>11.16</td>
<td>11.07</td>
<td>-0.09</td>
</tr>
<tr>
<td><strong>RMSE</strong></td>
<td>15.71</td>
<td>15.59</td>
<td>-0.12</td>
</tr>
</tbody>
</table>

The results for the right knee angle comparison between our system and the golden system has shown in **Table 2**. For the datasets without applying a low-pass filter, we get the MAE value of 11.31% and RMSE value of 15.20%. Meanwhile, we calculate the MAE and RMSE results for the datasets after applying the low-pass filter, and we get 11.07% and 15.59%, respectively. From Table2, we can see that the evaluation results get better after applying the low-pass filter, but not much.
The second part of aim1 is analyzing the ankle angles generated by both systems. In Figure 15, the left plot shows the ankle angle values measured by the golden system for the same four running cycles as above. Meanwhile, the right plot shows the ankle angle values calculated by our system. The golden system results show that the right ankle angle increases at the beginning of the running cycle, the value changes from around 40 degrees to 65 degrees. Then, the value stays around 65 degrees for a period before it increases again. Next, the ankle angle value increases until it reaches its only peak in that cycle, which the value is around 90 degrees. Last, the ankle angle decreases to the original position at the end of the running cycle. However, the ankle angle results from our system did not present a satisfactory outcome like the golden system. The ankle angle plot from our system shows a similar pattern as the ankle angle plot from the golden system. However, there are still many differences between the two results. The first one is the angle value range. For the golden system, the range of the values is between 40 to 90 degrees, but the range for our system is between 55 to 95 degrees. The second difference between the two results is the height of the small peak in each jump cycle, which is the area highlighted by the orange box. The peak in the golden system plot has an average height.
of 4.94 ± 0.75 degrees. However, the average height for the small peak in the ankle angle plot from our system is 10.11 ± 3.75 degrees. This difference may be because our angles are measured using a single camera instead of a three-dimensional view as in the ground truth system. Our system's overall performance for the ankle angle measurement obtained an MAE result of 18.94% and an RSME result of 25.45%. Based on the mentioned differences between the two results and the error values, we can see that the ankle angle measurement from our system contains a lot of noise, even more than the knee angle measurements from our system.

![Image of ankle angle measurement]

**Figure 16. Examples of bad detection on big toes**

After checking the outcome from OpenPose, we noticed that the OpenPose did not perform well on the big toe detection in the running images. The incorrect big toe detection led to the incurred ankle angle values since we calculated the angle values using the knee, ankle, and big toe locations. One of the potential improvements is to use a higher-quality video camera for recording. When the running velocity is high, the captured images can be blurry since our camera recording rate is only 120 FPS. That causes the OpenPose to fail to give a precise detection on the big toe consistently. For example, **Figure 16** shows one of the bad big toe detection examples from the OpenPose algorithm. Compared to the left on the treadmill, the
accuracy of the right foot detection is much worse. The second potential reason for the inaccurate detection of big toes could be rooted in the training dataset of OpenPose. As we mentioned, OpenPose used a separate foot dataset for their foot detection training, and the foot dataset is significantly smaller than the training dataset for other body joints. That could be one of the causes for the inaccurate detection on the big toes. Therefore, the second improvement could be to find a new approach for detecting the big toe and replace the big toe detection results with the results from the new approach. Therefore, if we can create an object detection algorithm for the shoes instead of the foot, the detection result can be more precise. Next, we will focus on finding the location of the shoe toe in the given shoe images. That can be achieved by either temple matching or other regression algorithms. With the time-strain, the improvement of the results will be completed in the future.

Finally, the last task for aim 1 is to create a 3D skeleton animation of the running video for a demonstration. I used the pre-trained GAST-Net model (Liu, 2020) to generate the animation. Figure 17 shows the example of the outcome.

![Generated 3D skeleton animation from the single video](image)
4.b Jumping analysis results

Aim 2 of this Master thesis is to develop a classification algorithm to differentiate the bad knee position from normal jumping using the collected features. As we have mentioned in the methodology, we have collected a total of 46 jump cycles, including 25 normal jumps, 12 abnormal jumps with inner knee position, and 9 abnormal jumps with outer knee position.

![Generated 3D skeleton animation from the single video](image)

**Figure 18. Generated 3D skeleton animation from the single video**

To include both movements of the knee and the ankle, we used the engineered feature angle ratio. The above plots in **Figure 18** are some examples of the angle ratio changes in different jump cycles. The left two columns of plots are the angle ratio plots from normal jump samples. The middle two columns of plots are from abnormal jump samples with inner knee position. The last two columns of plots are from abnormal jump samples with outer knee position.
Table 3. Evaluation of the KNN Classification using a single feature to predict good and bad jumping. The accuracy scores are proportional to true positive rate and true negative rate.

<table>
<thead>
<tr>
<th>Features</th>
<th>True Positive Rate</th>
<th>False Positive Rate</th>
<th>False Negative rate</th>
<th>True Negative rate</th>
<th>Accuracy Score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right Leg Angle Ratio</td>
<td>0.383</td>
<td><strong>0.050</strong></td>
<td>0.100</td>
<td>0.467</td>
<td>85.00</td>
</tr>
<tr>
<td>Right Knee Angle</td>
<td>0.383</td>
<td><strong>0.050</strong></td>
<td>0.133</td>
<td>0.433</td>
<td>81.67</td>
</tr>
<tr>
<td>Left Ankle Angle</td>
<td>0.317</td>
<td>0.117</td>
<td><strong>0.067</strong></td>
<td>0.500</td>
<td>81.67</td>
</tr>
<tr>
<td>Left Knee Angle</td>
<td>0.383</td>
<td><strong>0.050</strong></td>
<td>0.15</td>
<td>0.417</td>
<td>80.00</td>
</tr>
<tr>
<td>Left Leg Angle Ratio</td>
<td>0.350</td>
<td>0.083</td>
<td>0.217</td>
<td>0.350</td>
<td>70.00</td>
</tr>
<tr>
<td>Right Ankle Angle</td>
<td>0.383</td>
<td><strong>0.050</strong></td>
<td>0.333</td>
<td>0.233</td>
<td>61.67</td>
</tr>
</tbody>
</table>

Table 3 shows the evaluation results for our KNN classifier. We used five different features individually to train the KNN classification. We can notice that the highest accuracy (85%) of the classifier occurs when we use the right leg angle ratio as the training feature. The lowest accuracy is 61.67% when we use the right ankle angle as the training feature. If our classifier works well, the false positive and false negative rates should be as close to 0 as possible. Our classifier achieves a 0.07 false-positive rate and a 0.18 false negative rate on average. The lowest positive rate is 0.050, and it is achieved by using the right leg angle ratio, the left knee angle, the right knee angle, and the right ankle angle as training features. The false-
negative rate has a minimum of 0.067 which is achieved by using the left ankle angle as a training feature.

The classification accuracy can be improved by expanding the datasets. We only recorded 46 jump cycles because of time strain. The dataset is small and normal jump and abnormal jump are not significantly different from each other, so we need to record more jump cycles to improve the accuracy of the classifier. As for now, the KNN classifier uses only one feature every time for training. Suppose we can introduce more than one feature (such as combining the left/right knee angles and the left ankle angle) to the classification algorithm for every training phase. In that case, the prediction results could be improved. One of the drawbacks of this approach is that the measured values contain noise, so introducing more features might decrease the accuracy because it might also introduce more noise.
Chapter V: Conclusions and Future Work

In this Master thesis, we have represented a potential approach for a marker-less motion capture system using only one camera. Our method provides a more portable and affordable system compared to the marker-based and marker-less motion capture systems. For instance, Qualisys motion capture system can achieve high accuracy for the gait analysis using eight motion cameras and many reflective markers. However, the initial cost for the Qualisys system can be as high as 50,000 dollars, and the system requires ample space to mount the cameras, which are not portable. Since our method uses only one camera for data recording, the affordability and portability of our system are satisfied with the expectation. In addition, to determine the accuracy of our system, we have tested our system by analyzing two basic gaits: running and vertical jumping. To sum up this thesis, we highlight the main achievements as following:

- We have tested the accuracy of the OpenPose algorithm and verified its functionality for gait analysis purposes. Overall, the OpenPose algorithm can maintain good accuracy for most lower limb key points except the foot. There are two potential causes for this issue: 1) the quality of our camera is not good enough. When the velocity of the foot movement is high, the camera does not have a sufficient FPS recording rate to capture clear images for the foot; and 2) the OpenPose algorithm training was not focused on detecting foot data.

- We have performed a detailed data analysis on the running dataset and compared the accuracy of our system with the Qualisys motion capture system. To quantify the evaluation, we have calculated the MAE and RMSE results of the knee angle values, and the ankle angle values measured by our system and the Qualisys system. The MAE and RMSE results for the knee angle values shows that the difference between our system and
the Qualisys system is small enough for our system to perform sufficient gait analysis on the knees. However, with the low accuracy of the big toe detection from OpenPose, the collected ankle angle dataset contains many noises. The gait analysis results for the ankle angle do not satisfy our expectations.

- We have also generated a 3D skeleton animation as one of the outputs from our system to visualize the gait for our collected running dataset. The 3D skeleton estimation is achieved by applying the GAST-Net model.

- For the jumping dataset, we have developed a classification algorithm for bad knee position detection using collected kinematic data. Overall, we have identified normal versus abnormal jumping using a KNN classification model and dynamic time wrapping. The classification model achieves a maximum of 85% accuracy using the right leg angle ratio as our training engineering feature. One of the drawbacks for the classifier is that the model does not perform well on the abnormal jump cycles with inner knees. Because of time strain, we only gathered a relatively small training dataset that contains only 46 different jump cycles. We expect that a bigger dataset could lead to improving the accuracy of our system.

The proposed low-cost method does not track the depth information of the knee, which affects the accuracy of our system. Future work for the running gait analysis could be focused on investigating a new approach to either measure or estimate the depth information of key points to increase the dimension of our dataset. Also, we will improve the accuracy of the foot detection to perform a more precise gait analysis related to foot movement. Regarding the jumping data, we will collect more data to improve the accuracy of the classification model potentially.
Reference


Appendix

Python Code:

a. Aim 1
Running Analysis

Background:

We have collected two running datasets. One dataset is collected using our system (marker-less gait analysis system), and another dataset is collected by Qualisys (marker-based gait analysis system).

Datasets:

All kinematical data are stored in csv files. For our system, we collected four angles (right knee angle, right ankle angle, left knee angle and left ankle angle). For golden system, it stored the right knee angle and right ankle angle.

Analysis Goal:

We want to compare the accuracy of our system using the Qualisys system as golden rule.

compare the knee angle measurements from two systems
compare the ankle angle measurements from two systems
create visualization for both systems and angles

1. Import needed libraries and datasets

In [1]:
   a. # import needed library
   b. import pandas as pd
   c. import matplotlib.pyplot as plt
   d. import seaborn as sns
   e. import json
   f. import numpy as np
   g. from scipy.signal import savgol_filter
   h. from sklearn.metrics import mean_absolute_error as mae

In [2]:
   i. # load the csv files to dataframe for use
   j. our_sys = pd.read_csv('..input/processed-running-data-0720/right_camera_0.20.csv')
   k. gold_sys = pd.read_csv('..input/processed-running-data-0720/golden_system.csv')

2. Take a first look at the dataframes our system and the golden system

2.1 check the loaded dataframes

In [3]:
   l. # show the dataframe of our system
```python
m = our_sys

Out[3]:

<table>
<thead>
<tr>
<th>current_frame</th>
<th>Angle1</th>
<th>Angle2</th>
<th>Angle3</th>
<th>Angle4</th>
<th>probLtoe</th>
<th>probRtoe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>169.05</td>
<td>167.92</td>
<td>97.46</td>
<td>92.95</td>
<td>0.423761</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>169.10</td>
<td>168.85</td>
<td>96.34</td>
<td>92.59</td>
<td>0.256683</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>171.72</td>
<td>170.62</td>
<td>105.72</td>
<td>99.59</td>
<td>0.434617</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>170.45</td>
<td>170.25</td>
<td>93.34</td>
<td>93.27</td>
<td>0.280547</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>169.54</td>
<td>169.67</td>
<td>96.88</td>
<td>94.67</td>
<td>0.340081</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2395</td>
<td>2395</td>
<td>134.30</td>
<td>95.97</td>
<td>88.41</td>
<td>114.06</td>
<td>0.840123</td>
</tr>
<tr>
<td>2396</td>
<td>2396</td>
<td>143.69</td>
<td>99.09</td>
<td>97.13</td>
<td>100.71</td>
<td>0.856330</td>
</tr>
<tr>
<td>2397</td>
<td>2397</td>
<td>145.12</td>
<td>103.29</td>
<td>117.78</td>
<td>111.59</td>
<td>0.617615</td>
</tr>
<tr>
<td>2398</td>
<td>2398</td>
<td>137.40</td>
<td>103.62</td>
<td>93.07</td>
<td>104.38</td>
<td>0.504945</td>
</tr>
<tr>
<td>2399</td>
<td>2399</td>
<td>136.80</td>
<td>101.51</td>
<td>99.61</td>
<td>108.01</td>
<td>0.362365</td>
</tr>
</tbody>
</table>

2400 rows × 7 columns

In [4]:
```
n.  # show the dataframe of golden system
o.  gold_sys

Out[4]:

<table>
<thead>
<tr>
<th></th>
<th>right knee angle</th>
<th>right ankle angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-81.53249</td>
<td>54.53057</td>
</tr>
<tr>
<td>1</td>
<td>-80.78425</td>
<td>55.62489</td>
</tr>
<tr>
<td>2</td>
<td>-80.28616</td>
<td>56.21629</td>
</tr>
<tr>
<td>3</td>
<td>-78.91374</td>
<td>57.14418</td>
</tr>
<tr>
<td>4</td>
<td>-77.90425</td>
<td>58.31234</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>7195</td>
<td>-20.30280</td>
<td>51.49057</td>
</tr>
<tr>
<td>7196</td>
<td>-20.39163</td>
<td>49.78628</td>
</tr>
<tr>
<td>7197</td>
<td>-20.77135</td>
<td>48.72185</td>
</tr>
<tr>
<td>7198</td>
<td>-21.38792</td>
<td>49.16248</td>
</tr>
<tr>
<td>7199</td>
<td>-22.11065</td>
<td>47.81673</td>
</tr>
</tbody>
</table>

7200 rows × 2 columns

3. Select a section of the running data for detail analysis
3.1 The Analysis for the knee angles

3.1.1 Check the knee angle changes during running in golden system

In [5]:
   p. plt.figure()
   q. test_gold = gold_sys[163:857].reset_index()
   r. sns.lineplot(data = test_gold, x = test_gold.index, y = 'right knee angle')
   s. plt.title('Knee Angle Measured by Golden System')
   t. plt.xlabel('frame number')
   u. test1 = gold_sys[800:900]
   v. test2 = gold_sys[200:400]
   w. x. # p1 = test1['right knee angle'].idxmin()
   y. # p2 = test2['right knee angle'].idxmin()

In [6]:
   z. # show golden system
   aa. a = 2
   bb. b = 2
   cc. plots_num = a * b
   dd. cycle_length = 173
   ee. fig = plt.figure(figsize=(15,15))
   ff. for i in range(plots_num):
      gg. if i == 0:
         hh. start_idx = 0
         ii. end_idx = cycle_length + start_idx
         jj. plt.subplot(a,b,i+1)
         kk. plt.title('Knee Angle Plot from Golden System before Resampling(cycle {})'.format(i+1))
      ll. plt.xlabel('frame number')
      mm. plt.ylabel('Angle values (degrees)')
      nn. sns.scatterplot(data = test_gold[start_idx:end_idx], x = test_gold[start_idx:end_idx].index,y = 'right knee angle')
      oo. sns.lineplot(data = test_gold[start_idx:end_idx], x = test_gold[start_idx:end_idx].index,y = 'right knee angle')
      pp. else:
         qq. start_idx += cycle_length
         ss. end_idx += cycle_length
         tt. plt.subplot(a,b,i+1)
         uu. plt.title('Knee Angle Plot from Golden System before Resampling(cycle {})'.format(i+1))
      vv. plt.xlabel('frame number')
      ww. plt.ylabel('Angle values (degrees)')
      xx. sns.scatterplot(data = test_gold[start_idx:end_idx], x = test_gold[start_idx:end_idx].index,y = 'right knee angle')
      yy. sns.lineplot(data = test_gold[start_idx:end_idx], x = test_gold[start_idx:end_idx].index,y = 'right knee angle')

As we can see from the plots, the knee angle changes periodically over time. There are two peak in one cycle and the range of the angle should be within [-90, -10]
Since the method of ankle angle reference in our system is slightly different from the golden system, we need to adjust the system to the same scale.

- First, we manually selected the running cycle in our system that represent the same cycles in the golden system
- Second, knee angle of our system = knee angle of golden system - 180 degrees
- Third, ankle angle of our system = 180 - ankle angle of golden system
- Last, reset the index and represent the new dataframe of our system

In [7]:
zz. right = our_sys[814:1180]

aaa. right['right knee angle'] = right['Angle2'] - 180

bbb. right['right ankle angle'] = 180 - right['Angle4']

ccc. right = right.reset_index()

Out[7]:

<table>
<thead>
<tr>
<th>index</th>
<th>current_frame</th>
<th>Angle1</th>
<th>Angle2</th>
<th>Angle3</th>
<th>Angle4</th>
<th>probLtoe</th>
<th>probRtoe</th>
<th>right knee angle</th>
<th>right ankle angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>814</td>
<td>143.83</td>
<td>102.92</td>
<td>100.40</td>
<td>94.41</td>
<td>0.857391</td>
<td>0.857391</td>
<td>-77.08</td>
<td>85.59</td>
</tr>
<tr>
<td>1</td>
<td>815</td>
<td>141.49</td>
<td>106.02</td>
<td>102.01</td>
<td>94.02</td>
<td>0.846180</td>
<td>0.846180</td>
<td>-73.98</td>
<td>85.98</td>
</tr>
<tr>
<td>2</td>
<td>816</td>
<td>140.48</td>
<td>105.58</td>
<td>103.51</td>
<td>100.28</td>
<td>0.787463</td>
<td>0.787463</td>
<td>-74.42</td>
<td>79.72</td>
</tr>
<tr>
<td>3</td>
<td>817</td>
<td>138.36</td>
<td>110.96</td>
<td>102.48</td>
<td>99.47</td>
<td>0.549786</td>
<td>0.549786</td>
<td>-69.04</td>
<td>80.53</td>
</tr>
<tr>
<td>4</td>
<td>818</td>
<td>140.46</td>
<td>111.88</td>
<td>102.33</td>
<td>97.88</td>
<td>0.474370</td>
<td>0.474370</td>
<td>-68.12</td>
<td>82.12</td>
</tr>
</tbody>
</table>

... ... ... ... ... ... ... ... 

361 1175 1175 132.55 109.11 90.07 92.94 0.801312 0.801312 -70.89 87.06

362 1176 1176 136.76 105.34 103.49 97.98 0.825796 0.825796 -74.66 82.02
<table>
<thead>
<tr>
<th>index</th>
<th>current_frame</th>
<th>Angle1</th>
<th>Angle2</th>
<th>Angle3</th>
<th>Angle4</th>
<th>probLtoe</th>
<th>probRtoe</th>
<th>right knee angle</th>
<th>right ankle angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>363</td>
<td>1177</td>
<td>1177</td>
<td>151.57</td>
<td>106.33</td>
<td>111.25</td>
<td>96.55</td>
<td>0.807722</td>
<td>-73.67</td>
<td>83.45</td>
</tr>
<tr>
<td>364</td>
<td>1178</td>
<td>1178</td>
<td>147.10</td>
<td>108.83</td>
<td>105.66</td>
<td>101.87</td>
<td>0.766664</td>
<td>-71.17</td>
<td>78.13</td>
</tr>
<tr>
<td>365</td>
<td>1179</td>
<td>1179</td>
<td>144.33</td>
<td>112.23</td>
<td>104.05</td>
<td>102.53</td>
<td>0.687616</td>
<td>-67.77</td>
<td>77.47</td>
</tr>
</tbody>
</table>

366 rows × 10 columns

3.1.2 Check the knee angle changes in our system

Steps taken:

Apply savgol filter to clean the noisy in our system

Create a plot for all four running cycles

Create four seperate plots for each running cycle

Plot the knee angle data points and the smoothed data in the same plot to visualize the difference

In [8]:

```python
ddd.  plt.figure(figsize = [15,5])
eee.  plt.subplot(1,2,1)
fff.  sns.lineplot(data = right, x = right.index,y='right knee angle')

```  

```python
hhh.  plt.title('Right Knee Angle Plots before applying filter')

```  

```python
iii.  plt.xlabel('frame number')
jjj.  plt.subplot(1,2,2)

```  

```python
kkk.  sns.lineplot(data = right, x = right.index,y='right knee angle')

```  

```python
lll.  smoothed = savgol_filter(right['right knee angle'], 25, 2)
mmm.  plt.plot(right.index, smoothed)

```  

```python
nnn.  plt.title('Right Knee Angle Plots after applying filter')

```  

```python
ooo.  plt.xlabel('frame number')
ppp.  right['filtered data'] = smoothed

```  

In [9]:

```python
qqq.  right.to_csv(r'.//oursys.csv',index = False)

```

In [10]:

```python
linkcode

```  

```python
rrr.  plt.figure()

```  

```python
sss.  sns.scatterplot(data = right, x = right.index,y='right knee angle')

```  

```python
bbb.  smoothed = savgol_filter(right['right knee angle'], 25, 2)
```
Section conclusion:

As above figures shown, the plots in our system also indicate two peaks in one cycle. The angle range is [-80, -10]. However, we can also see that our system contains more noisy than the golden system. With the smoothed filter, the noisy of our running dataset has been reduced.

3.2 Comparision of the knee angle from our system and the golden system

pre-process of the data, so two dataset can be aligned and resampled

resample the golden system dataset to the same amount of data points as our system

use dynamic time warping to align the running cycles from two different datasets

3.2.1 resample the golden system dataset to the same amount as our system

In [11]:
xxxx. # resample the golden system dataset to the same amount as our system.
yyyy. from sklearn.utils import resample
zzzz. # show golden system
aaaaa.  $a = 4$

bbbb.  $b = 1$

cccc.  $\text{plots}_\text{num} = a \times b$

dddd.  $\text{cycle}_\text{length} = 92$

eeee.

fffff.  # resample the golden system dataset
ggggg.  resample_data = resample(test_gold, n_samples = 366, replace = False, random_state = 0).sort_index()
hhhhh.  resample_data = resample_data.reset_index()
iiiii.  smooth_knee = savgol_filter(resample_data['right knee angle'], 11, 2)
jjjjjj.  smooth_ankle = savgol_filter(resample_data['right ankle angle'], 11, 2)
kkkkk.  plt.plot(resample_data.index, smooth_knee)
llllll.  plt.plot(resample_data.index, smooth_ankle)

mmmmm. resample_data['golden sys filtered data'] = smooth_knee
nnnnn. resample_data['golden sys ankle filtered data'] = smooth_ankle

In [12]:

ooooo. resample_data

Out[12]:

<table>
<thead>
<tr>
<th>level_0</th>
<th>index</th>
<th>right knee angle</th>
<th>right ankle angle</th>
<th>golden sys filtered data</th>
<th>golden sys ankle filtered data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>163</td>
<td>-89.18333</td>
<td>-89.334556</td>
<td>47.927196</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>164</td>
<td>-89.09293</td>
<td>-89.028450</td>
<td>49.389785</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>166</td>
<td>-88.97738</td>
<td>-88.507656</td>
<td>50.748284</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>169</td>
<td>-87.24503</td>
<td>-87.772174</td>
<td>52.002691</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>170</td>
<td>-86.68006</td>
<td>-86.822002</td>
<td>53.153009</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>687</td>
<td>850</td>
<td>-80.44536</td>
<td>-80.455620</td>
<td>48.469378</td>
</tr>
<tr>
<td>level_0</td>
<td>index</td>
<td>right knee angle</td>
<td>right ankle angle</td>
<td>golden sys filtered data</td>
<td>golden sys ankle filtered data</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>362</td>
<td>688</td>
<td>-81.74565</td>
<td>49.67009</td>
<td>-82.177748</td>
<td>49.767025</td>
</tr>
<tr>
<td>363</td>
<td>690</td>
<td>-83.47525</td>
<td>52.01656</td>
<td>-83.509427</td>
<td>51.150575</td>
</tr>
<tr>
<td>364</td>
<td>692</td>
<td>-84.31090</td>
<td>52.79615</td>
<td>-84.450656</td>
<td>52.620030</td>
</tr>
<tr>
<td>365</td>
<td>693</td>
<td>-85.07222</td>
<td>53.49356</td>
<td>-85.001434</td>
<td>54.175388</td>
</tr>
</tbody>
</table>

366 rows x 6 columns

In [13]:

```python
fig = plt.figure(figsize=(15,15))
a = b = 2
cycle_length = 92
plots_num = range(1)

for i in range(plots_num):
    if i == 0:
        start_idx = 0
        end_idx = cycle_length + start_idx
    else:
        start_idx += cycle_length
        end_idx += cycle_length

temp = resample_data[start_idx: end_idx]
plt.subplot(a,b,i+1)
plt.title('Knee Angle Plot from Golden System after Resampling(cycle {})'.format(i+1))
plt.xlabel('frame number')
plt.ylabel('Knee Angle values (degrees)')
sns.scatterplot(data = temp, x = temp.index, y = 'right knee angle')
sns.lineplot(data = temp, x = temp.index, y = 'golden sys filtered data')
```

3.2.2 Apply DTW (dynamic time wrapper) for two dataset

Future work: apply DTW for alignment of two datasets

3.2.3 Analysis result analysis

Use mean absolute error and mean squared error matrices to measure the accuracy of our system comparing to the golden system

In [14]:

```python
from sklearn.preprocessing import MinMaxScaler
```
```python
from sklearn.metrics import mean_squared_error as mse

scaler = MinMaxScaler()

dftest = pd.concat([resample_data['right knee angle'], right['filtered data']], axis = 1)

test_data = scaler.fit_transform(dftest[['right knee angle', 'filtered data']])

print('mean absolute error is: {:.2f}%' .format(100 * mae(test_data[:,0], test_data[:,1])))

print('mean square error is: {:.2f}%' .format(100 * mse(test_data[:,0], test_data[:,1], squared = False)))
```

mean square error is: 15.20%

### 3.3 Comparision of the ankle angle from our system and the golden system

#### 3.3.1 Check the ankle angle for golden system

```py
In [15]:
df = pd.concat([resample_data['right knee angle'], right['right knee angle']], axis = 1)
test_data = scaler.fit_transform(dftest[['right knee angle', 'right knee angle']])

print('mean absolute error is: {:.2f}%' .format(100 * mae(test_data[:,0], test_data[:,1])))

print('mean square error is: {:.2f}%' .format(100 * mse(test_data[:,0], test_data[:,1], squared = False)))
```

mean square error is: 15.20%

<table>
<thead>
<tr>
<th>index</th>
<th>right knee angle</th>
<th>right ankle angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-89.18333</td>
<td>47.89883</td>
</tr>
<tr>
<td>1</td>
<td>-89.09293</td>
<td>48.67561</td>
</tr>
<tr>
<td>2</td>
<td>-88.96581</td>
<td>49.55359</td>
</tr>
<tr>
<td>3</td>
<td>-88.97738</td>
<td>51.14827</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>right knee angle</td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>4</td>
<td>167</td>
<td>-88.0835</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>689</td>
<td>852</td>
<td>-82.1676</td>
</tr>
<tr>
<td>690</td>
<td>853</td>
<td>-83.47525</td>
</tr>
<tr>
<td>691</td>
<td>854</td>
<td>-83.67033</td>
</tr>
<tr>
<td>692</td>
<td>855</td>
<td>-84.31090</td>
</tr>
<tr>
<td>693</td>
<td>856</td>
<td>-85.07222</td>
</tr>
</tbody>
</table>

694 rows × 3 columns

In [17]:

```
In [17]:
plt.figure()

In [17]:
ax = plt.figure().add_subplot(111)

In [17]:
x = test_gold.index

In [17]:
y = test_gold['right ankle angle']

In [17]:
plt.plot(x, y)

In [17]:
plt.xlabel('Frame number')

In [17]:
plt.ylabel('Right Ankle Angle Measured by Golden System')

In [17]:
```

```python
# smooth_ankle = savgol_filter(right['right ankle angle'], 25, 2)

# right['ankle filtered data'] = smooth_ankle

# plt.plot(right.index, smooth_ankle)

# show golden system

```

```python
a = 2
b = 2
gggggggg. plots_num = a * b

hhhhhhh. cycle_length = 173

iiiiiiiii. fig = plt.figure(figsize=(15,15))

jjjjjjjjj. for i in range(plots_num):

kkkkkkk. if i == 0:

lllllllll. start_idx = 0

mmmmmmmm. end_idx = cycle_length + start_idx

nnnnnnn. plt.subplot(a,b,i+1)

ooooooo. plt.title('subplots {}{}{} : cycle {}'.format(a,b,i+1,i+1))
```
3.3.2 Check the ankle angle for our system

In [18]:

```python
plt.figure(figsize = [15,5])
plt.subplot(1,2,1)
sns.lineplot(data = right, x = right.index,y='right ankle angle')
plt.title('Right Ankle Angle before applying filter')
plt.subplot(1,2,2)
smooth_ankle = savgol_filter(right['right ankle angle'], 25, 2)
right['ankle filtered data'] = smooth_ankle
plt.plot(right.index, smooth_ankle)
plt.xlabel('frame number')
plt.title('Right Ankle Angle after applying filter')
```

Out[18]:

```
Text(0.5, 1.0, 'Right Ankle Angle after applying filter')
```

In [19]:

```python
plt.figure()
sns.scatterplot(data = right, x = right.index,y='right ankle angle')
smooth_ankle = savgol_filter(right['right ankle angle'], 25, 2)
right['ankle filtered data'] = smooth_ankle
plt.plot(right.index, smooth_ankle)
plt.xlabel('Frame number')
plt.title('Right Ankle Angle Measured by Our System')
```
3.3.3 Analysis result comparison

In [20]:
    df = pd.concat([resample_data['right ankle angle'], right['ankle filtered data']], axis = 1)
    test_data = scaler.fit_transform(df[['right ankle angle', 'ankle filtered data']])
    print('mean absolute error is {:.2f}%.format(100*mae(test_data[:,0], test_data[:,1], square=False))
    print('mean squared error is {:.2f}%.format(100*mae(test_data[:,0], test_data[:,1], square=False))

In [21]:
    df = pd.concat([resample_data['right ankle angle'], right['right ankle angle']], axis = 1)
    test_data = scaler.fit_transform(df[['right ankle angle', 'right ankle angle']])
    print('mean absolute error is {:.2f}%.format(100*mae(test_data[:,0], test_data[:,1])))
    print('mean squared error is {:.2f}%.format(100*mae(test_data[:,0], test_data[:,1]))

b. Aim2

Created on Fri Sep 17 13:44:46 2021

@author: bfeng1
import json
import sys
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from scipy.signal import savgol_filter
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from scipy.stats import mode
from scipy.spatial.distance import squareform
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# %

class jump:
    r = 5
    def __init__(self, name, jump_cycle):
        self.name = name
        self.jump_cycle = jump_cycle
    def csv2df(self):
        csv_file = self.name + '.csv'
        df = pd.read_csv(csv_file)
        # cleaning the dataset(drop the rows with ratio is higher than 2.25)
        df['left angle ratio'] = df['Angle1']/df['Angle3']
        df['right angle ratio'] = df['Angle2']/df['Angle4']
        df.drop(df[df['left angle ratio'] > 2.25].index, inplace = True)
        df.drop(df[df['left angle ratio'] < 0.75].index, inplace = True)
        df['smoothed1'] = savgol_filter(df['left angle ratio'], 25, 2)
df['smoothed2'] = savgol_filter(df['right angle ratio'], 25, 2)
return df

def finetune(self):
    df = jump.csv2df(self)
jump_cycle = self.jump_cycle
new_results = []
for domain in jump_cycle:
    current_list = []
    for inx in domain:
        start = inx - jump.r
        end = inx + jump.r
        temp = df[start:end]
        max_val = temp['left angle ratio'].max()
        ind = temp[temp['left angle ratio'] == max_val].index.values.astype(int)
        try:
            ind = ind[0]
        except:
            ind = 0
        current_list.append(ind)
    new_results.append(current_list)
check = (jump_cycle == new_results)
if check is False:
    print('old cycle {}: {}'.format(self.name, jump_cycle))
    print('new cycle {}: {}'.format(self.name, new_results))
elif check is True:
    print('The jump cycle has been finetuned')
return new_results

def resample_df(self,n = 135, replace = False):

df_list = []
jump_cycle = self.jump_cycle
df = jump.csv2df(self)
for i in range(len(jump_cycle)):
    temp = df[jump_cycle[i][0]:jump_cycle[i][1]]
    resample_data = resample(temp, n_samples = n, replace = replace, random_state = 0).sort_index()
    resample_data = resample_data.reset_index()
    df_list.append(resample_data)
return df_list

def vis(self):
    df_list = jump.resample_df(self)
a = (len(df_list)+1)//2
b = 2
plt.figure(figsize = (14,22))
for i in range(len(df_list)):
    plt.subplot(a,b,i+1)
    plt.title('subplots {{}{}{}} : cycle {{}{}{}}'.format(a,b,i+1,i+1))
    plt.xlabel('frame number')
    plt.ylabel('Left angle ratio')
sns.scatterplot(data = df_list[i], x = df_list[i].index, y = 'left angle ratio')
sns.lineplot(data = df_list[i], x = df_list[i].index, y = 'smoothed')
print('the process is done for the jump {}'.format(self.name))

# %%
#
# create lists to store the names of csv files
# create jump cycle (manually select range, then autocorrect by algorithm)
# good_jump_cycle =
[[154,309],[398,539],[651,786],[825,980],[1018,1158],[1188,1337],[1374,1524],[1555,1698],[1737,1881],[1895,2054]]
# # cycle1: [010262,010456], [010469, 010638], [010655,010821],[010829,010998],[011168,011331], [011331, 011497],[011497,011659],[011670,011849],[011849,012015]
# inner_jump_cycle=[
[397,562],[562,742],[742,902],[902,1060],[1060,1232],[1232,1398],[1398,1583],[1583,1760]]
# # cycle1: [001550,001700], [001716, 001902], [001930,002095], [002128,002300],[002330,002520],[002540,002709],[002729, 002900],[002916,003078],[003085,03249]
# outer_jump_cycle =
[[379,552],[579,767],[767,973],[991,1171],[1171,1351],[1364,1527],[1543,1697]]

class KnnDtw(object):
    
    """K-nearest neighbor classifier using dynamic time warping as the distance measure between pairs of time series arrays

Arguments
---------

n_neighbors : int, optional (default = 5)
    Number of neighbors to use by default for KNN

max_warping_window : int, optional (default = infinity)
    Maximum warping window allowed by the DTW dynamic
subsample_step : int, optional (default = 1)
    Step size for the timeseries array. By setting subsample_step = 2,
    the timeseries length will be reduced by 50% because every second
    item is skipped. Implemented by x[:, ::subsample_step]

def __init__(self, n_neighbors=5, max_warping_window=10000, subsample_step=1):
    self.n_neighbors = n_neighbors
    self.max_warping_window = max_warping_window
    self.subsample_step = subsample_step

def fit(self, x, l):
    """Fit the model using x as training data and l as class labels

    Arguments
    ---------
    x : array of shape [n_samples, n_timepoints]
        Training data set for input into KNN classifier

    l : array of shape [n_samples]
        Training labels for input into KNN classifier
    """

    self.x = x
    self.l = l

def _dtw_distance(self, ts_a, ts_b, d = lambda x,y: abs(x-y)):
    """Returns the DTW similarity distance between two 2-D
timeseries numpy arrays.

Arguments
---------

`ts_a, ts_b` : array of shape `[n_samples, n_timepoints]`

Two arrays containing `n_samples` of timeseries data
whose DTW distance between each sample of `A` and `B`
will be compared

`d` : DistanceMetric object (default = `abs(x-y)`)

The distance measure used for `A_i - B_j` in the
DTW dynamic programming function

Returns
-------

DTW distance between `A` and `B`

```python
# Create cost matrix via broadcasting with large int
ts_a, ts_b = np.array(ts_a), np.array(ts_b)
M, N = len(ts_a), len(ts_b)
cost = sys.maxsize * np.ones((M, N))

# Initialize the first row and column
cost[0, 0] = d(ts_a[0], ts_b[0])
for i in range(1, M):
    cost[i, 0] = cost[i-1, 0] + d(ts_a[i], ts_b[0])
for j in range(1, N):
    cost[0, j] = cost[0, j-1] + d(ts_a[0], ts_b[j])
```
# Populate rest of cost matrix within window
for i in range(1, M):
    for j in range(max(1, i - self.max_warping_window),
                    min(N, i + self.max_warping_window)):
        choices = cost[i - 1, j - 1], cost[i, j - 1], cost[i - 1, j]
        cost[i, j] = min(choices) + d(ts_a[i], ts_b[j])

# Return DTW distance given window
return cost[-1, -1]

def _dist_matrix(self, x, y):
    """Computes the M x N distance matrix between the training
dataset and testing dataset (y) using the DTW distance measure

Arguments
---------
x : array of shape [n_samples, n_timepoints]
y : array of shape [n_samples, n_timepoints]

Returns
-------
Distance matrix between each item of x and y with
    shape [training_n_samples, testing_n_samples]
"""

    # Compute the distance matrix
dm_count = 0
# Compute condensed distance matrix (upper triangle) of pairwise dtw distances
# when x and y are the same array
if(np.array_equal(x, y)):
    x_s = np.shape(x)
    dm = np.zeros((x_s[0] * (x_s[0] - 1)) // 2, dtype=np.double)
    dm_count = 0
    for i in range(0, x_s[0] - 1):
        for j in range(i + 1, x_s[0]):
            dm[dm_count] = self._dtw_distance(x[i, ::self.subsample_step],
                                               y[j, ::self.subsample_step])
            dm_count += 1

    # Convert to squareform
    dm = squareform(dm)

    return dm

# Compute full distance matrix of dtw distnces between x and y
else:
    x_s = np.shape(x)
    y_s = np.shape(y)
    dm = np.zeros((x_s[0], y_s[0]))
    for i in range(0, x_s[0]):
        for j in range(0, y_s[0]):
            dm[i, j] = self._dtw_distance(x[i, ::self.subsample_step],
                                           y[j, ::self.subsample_step])
            # Update progress bar
dm_count += 1

return dm

def predict(self, x):
    """Predict the class labels or probability estimates for
    the provided data
    
    Arguments
    --------
    x : array of shape [n_samples, n_timepoints]
        An array containing the testing data set to be classified
    
    Returns
    ------
    2 arrays representing:
        (1) the predicted class labels
        (2) the knn label count probability
    """

dm = self._dist_matrix(x, self.x)
# Identify the k nearest neighbors
knn_idx = dm.argsort()[:, :self.n_neighbors]

# Identify k nearest labels
knn_labels = self.l[knn_idx]

# Model Label
mode_data = mode(knn_labels, axis=1)
mode_label = mode_data[0]
mode_proba = mode_data[1]/self.n_neighbors

return mode_label.ravel(), mode_proba.ravel()

# %%
good = ['good1', 'good2', 'good4', 'good6']
inner = ['inner1', 'inner2', 'inner3']
outer = ['outer1', 'outer2']
with open('list_info.txt', 'r') as file:
    input_lines = [line.strip() for line in file]
all_csv = good + inner + outer
info = {}
info['name'] = all_csv
info['cycle'] = input_lines
# %%
# structure dataset for algorithm training
good_dataset = []
inner_dataset = []
outer_dataset = []
n = 135
for i in range(len(all_csv)):
    temp = jump(info['name'][i], json.loads(info['cycle'][i]))
    # temp.finetune()
    # temp.vis(n)
    if i < len(good):
        good_dataset += temp.resample_df()
    elif i < len(good + inner):
        inner_dataset += temp.resample_df()
    else:
        outer_dataset += temp.resample_df()
total_x = good_dataset+inner_dataset+outer_dataset
for i in range(len(total_x)):
    total_x[i]["series_id"] = i
X = pd.concat(total_x)

# compare time-series signal for good jump and bad (inner+outer) jump
# load the label file

y = pd.read_csv('lable.csv')
label_encoder = LabelEncoder()
encoded_labels = label_encoder.fit_transform(y.jump)
y['label'] = encoded_labels

# create feature column
feature_columns = X.columns.tolist()[2:]
# construct sequence
sequences = []
for series_id, group in X.groupby('series_id'):
    sequence_features = group[feature_columns]
    label = y[y.series_id == series_id].iloc[0].label
    sequences.append((sequence_features, label))

def create_data(sequences, test_size = 0.2, feature = 'left angle ratio', random_state = 0):
    train_sequences, test_sequences = train_test_split(sequences, test_size = 0.2, random_state=random_state)
    train_X = np.empty(shape = (len(train_sequences),135), dtype = 'object')
    train_y = []
    for i in range(len(train_sequences)):
        train_X[i] = train_sequences[i][feature]
        train_y.append(train_sequences[i]["label"])
    return train_X, train_y
test_X = np.empty(shape = (len(test_sequences),135), dtype = 'object')
test_y = []
for i in range(len(train_sequences)):
    temp_x = train_sequences[i][0][feature].to_list()
    train_X[i][:] = temp_x
    train_y.append(train_sequences[i][1])
for i in range(len(test_sequences)):
    temp_x = test_sequences[i][0][feature].to_list()
    test_X[i][:] = temp_x
    test_y.append(test_sequences[i][1])
train_y = np.array(train_y)
test_y = np.array(test_y)
return train_X, test_X, train_y, test_y

#%
random_state = [0,2,5,14,3,7]
features = ['Angle1', 'Angle2', 'Angle3', 'Angle4', 'left angle ratio','right angle ratio']
for feature in features:
    matrix = np.array([[0,0],[0,0]])
    score = 0
    score_list = []
    for i in random_state:
        m = KnnDtw(n_neighbors=2, max_warping_window=15)
        train_X, test_X, train_y, test_y = create_data(sequences, feature = feature, random_state=i)
        m.fit(train_X, train_y)
        label, proba = m.predict(test_X)
        temp_score = accuracy_score(label,test_y)
        matrix = np.add(confusion_matrix(test_y, label), matrix)
# tn, fp, fn, tp = confusion_matrix(test_y, label).ravel()
# false_positive_rate.append(fp/(fp+tn))
# false_negative_rate.append(fn/(fn + tp))
score_list.append(temp_score)
score += temp_score
print('the accuracy of the classifier for feature {}: {}%'.format(featu
re, score/len(random_state)*100))
matrix = matrix / len(random_state) / len(test_y)
print('Use feature {}, the confusion matrix is: {}'.format(feature, matrix))
# print('false positive rate: {}'.format(np.mean(false_positive_rate)))
# print('false negative rate: {}'.format(np.mean(false_negative_rate)))
# print('True positive rate: {}'.format(np.mean(false_positive_rate)))
# print('True negative rate: {}'.format(np.mean(false_negative_rate)))

#%%
t1 = X[X['series_id'] == 3].reset_index()
t2 = X1[X1['series_id'] == 3].reset_index()
#%%% plot figs for the two jump dataset
plt.figure(figsize = [20,10])
plt.subplot(1,2,1)
plt.title('(a) Before downsampling', fontsize = 30)
plt.plot(t2.index, t2['smoothed2'])
plt.ylabel('Angle Ratio', fontsize = 30)
plt.xlabel('Frame Number', fontsize = 30)
plt.subplot(1,2,2)
plt.title('(b) After downsampling', fontsize = 30)
plt.plot(t1.index, t1['smoothed2'])
plt.xlabel('Frame Number', fontsize = 30)